

Power Systems

More information about this series at <http://www.springer.com/series/4622>

Francisco M. Gonzalez-Longatt
José Luis Rueda
Editors

PowerFactory Applications for Power System Analysis

Editors

Francisco M. Gonzalez-Longatt
School of Electronic, Electrical and Systems
Engineering
Electrical Power Systems
Loughborough University
Loughborough
UK

José Luis Rueda
Department of Electrical Sustainable
Energy
Intelligent Electrical Power Systems
Delft University of Technology
Delft
The Netherlands

ISSN 1612-1287

Power Systems

ISBN 978-3-319-35619-8

DOI 10.1007/978-3-319-12958-7

ISSN 1860-4676 (electronic)

ISBN 978-3-319-12958-7 (eBook)

Springer Cham Heidelberg New York Dordrecht London

© Springer International Publishing Switzerland 2014

Softcover reprint of the hardcover 1st edition 2014

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made.

Printed on acid-free paper

Springer International Publishing AG Switzerland is part of Springer Science+Business Media
(www.springer.com)

Foreword

In 1987, the Kraftwerke Mainz-Wiesbaden, Germany (KMW AG), became the first official and registered user of the DIgSILENT software. It is not known if the fact that they were the first and single entry on the DIgSILENT PowerFactory reference list constituted apprehension on their behalf, or on behalf of the developers due to future client commitment. However, the fact that after some 27 years, KMW AG's PowerFactory installations are still maintained is a definite indicator that any possible initial doubt was unwarranted. In fact, over the past several years, the user base has grown considerably, establishing a large user community of highly qualified, experienced and creative users.

It has been the intention of the PowerFactory developers from the very beginning to provide a highly flexible power system analysis tool applicable to practically all areas of power generation, transmission, distribution and industry. Since late 1990, renewable generation based on wind and photovoltaics, along with new HVDC transmission technologies, has driven a new and fast-growing sector whose simulation needs include complex, non-traditional applications. In addition, the integration of power system analysis tasks with asset management systems and online applications (GIS, SCADA) as well as closed-loop applications (especially in Smart Grids) has increased the demand for flexible process signal interfaces and bidirectional data exchange capabilities. PowerFactory extends this flexibility even further via the provision of powerful scripting tools allowing the user to develop specific applications according to individual needs.

This first issue of the *PowerFactory Applications for Power System Analysis* serves as an impressive demonstration of the wide range of power system analysis applications that can be handled by the PowerFactory software. In addition, the solutions described for a certain analysis task will motivate and help other users to make use of the software's flexibility and give guidance regarding possible solution options.

We thank all authors who have contributed to this book, initiated by Francisco Gonzalez-Longat and José Luis Rueda, for sharing their valuable PowerFactory experiences and solutions. The topics presented are also of great interest to the PowerFactory development team in their endeavours to optimize further releases and develop even more flexible functions.

August 2014

Dr.-Ing. Martin Schmiege
Managing Director bei DIGSILENT GmbH
Stuttgart Und Umgebung
Deutschland

Contents

1	Load Flow Calculation and Its Application	1
	Muhammad Raza	
2	Modelling of Transmission Systems Under Unsymmetrical Conditions and Contingency Analysis Using DIgSILENT PowerFactory	27
	J.M. Roldan-Fernandez, Francisco M. Gonzalez-Longatt, José Luis Rueda and H. Verdejo	
3	Probabilistic Power Flow Module for PowerFactory DIgSILENT	61
	Saeed Teimourzadeh and Behnam Mohammadi-Ivatloo	
4	Unbalanced Power Flow Analysis in Distribution Systems Using TRX Matrix: Implementation Using DIgSILENT Programming Language	85
	Paulo M. De Oliveira-De Jesus, Andres A. Rojas Q and Francisco M. Gonzalez-Longatt	
5	DC Optimal Power Flow Formulation Using the Power Transmission Distribution Factors—A DIgSILENT Programming Language Application	111
	Víctor Hinojosa-Mateus, Leonardo Pérez-Andrades and Jovan Ilić	
6	Assessing the Renewable Energy Sources Integration Through a Series of Technical Performance Indices Using DIgSILENT PowerFactory DPL	135
	A.I. Nikolaidis, Francisco M. Gonzalez-Longatt and C.A. Charalambous	

7	Modeling of Automatic Generation Control in Power Systems . . .	157
	V. Pavlovsky and A. Steliuk	
8	Gas Turbine Modelling for Power System Dynamic Simulation Studies	175
	Lasantha Meegahapola and Damian Flynn	
9	Implementation of Simplified Models of DFIG-Based Wind Turbines for RMS-Type Simulation in DIgSILENT PowerFactory	197
	José Luis Rueda, Abdul W. Korai, Jaime C. Cepeda, István Erlich and Francisco M. Gonzalez-Longatt	
10	Parameterized Modal Analysis Using DIgSILENT Programming Language	221
	Sergio Pizarro-Gálvez, Héctor Pulgar-Painemal and Víctor Hinojosa-Mateus	
11	Probabilistic Approach for Risk Evaluation of Oscillatory Stability in Power Systems	249
	José Luis Rueda, Jaime C. Cepeda, István Erlich, Abdul W. Korai and Francisco M. Gonzalez-Longatt	
12	Mean–Variance Mapping Optimization Algorithm for Power System Applications in DIgSILENT PowerFactory	267
	Jaime C. Cepeda, José Luis Rueda, István Erlich, Abdul W. Korai and Francisco M. Gonzalez-Longatt	
13	Application and Requirement of DIgSILENT PowerFactory to MATLAB/Simulink Interface	297
	Shadi Khaleghi Kerahrودي, Mohsen M. Alamuti, F. Li, G.A. Taylor and M.E. Bradley	
14	Advanced Applications of DPL: Simulation Automation and Management of Results	323
	Matthias Stifter, Serdar Kadam and Benoît Bletterie	
15	Interfacing PowerFactory: Co-simulation, Real-Time Simulation and Controller Hardware-in-the-Loop Applications	343
	Matthias Stifter, Filip Andrén, Roman Schwalbe and Werner Tremmel	

16	PowerFactory as a Software Stand-in for Hardware in Hardware-In-Loop Testing	367
	Radhakrishnan Srinivasan	
17	Programming of Simplified Models of Flexible Alternating Current Transmission System (FACTS) Devices Using DIgSILENT Simulation Language	391
	Jaime C. Cepeda, Esteban D. Agüero and Delia G. Colomé	
18	Active and Reactive Power Control of Wind Farm Based on Integrated Platform of PowerFactory and MATLAB.	421
	Min-hui Qian, Da-wei Zhao, Da-jun Jiang, Ling-zhi Zhu and Jin Ma	
19	Implementation of Simplified Models of Local Controller for Multi-terminal HVDC Systems in DIgSILENT PowerFactory.	447
	Francisco M. Gonzalez-Longatt, J.M. Roldan, José Luis Rueda, C.A. Charalambous and B.S. Rajpurohit	
20	Estimation of Equivalent Model for Cluster of Induction Generator Based on PMU Measurements.	473
	Francisco M. Gonzalez-Longatt, José Luis Rueda, C.A. Charalambous and P. De Oliveira	

Introduction

Electric power systems have been conceived to convert energy from primary sources into electrical energy and to deliver it to industrial, commercial and residential users. The transportation of electrical power is carried out through transmission, subtransmission and distribution systems, which constitute predominantly AC facilities operating essentially at constant frequency and voltage. Nevertheless, due to transitional targets in different regions worldwide towards sustainable economies by means of large-scale incorporation of renewable energy-based generation sources, the systems are undergoing significant structural and topological changes, in which new power electronic-based technologies, such as for instance, voltage-sourced converter *high-voltage direct current* (VSC-HVDC) multi-terminal transmission links will be increasingly incorporated into the transmission networks.

The operation and planning of such complex systems requires in-depth engineering studies to understand the possible causes of threats to targeted steady-state and dynamic performance criteria as well as to devise prospective solutions from both operational and control viewpoints. So far, these studies rely largely on reliable modelling of system components and availability of suitable tools for simulation and analysis of different types of phenomena. In view of this, sophisticated software packages, such as *DIGSILENT PowerFactory*, have emerged to integrate state-of-art scientific approaches, which assist power engineers in both academic and industry-oriented research endeavours.

PowerFactory offers an amenable object-oriented platform for modelling and simulation of generation, transmission, distribution and industrial systems. Its modelling approach adopts a hierarchical scheme, which combines graphical and scripting modelling techniques. This includes a variety of standard models and the possibility of creating user defined models through *DIGSILENT Simulation Language* (DSL). The software also provides a number of functionalities for simulation, including, load flow calculation, optimal power flow calculation, estate estimation, contingency analysis, protection, RMS/EMT simulation, reliability assessment, to name a few. Besides, based on available basic functions, the

DIGSILENT Programming Language (DPL) allows defining the execution of user defined tasks, e.g. repetitive calculations and post-processing of results, in an automated manner. PowerFactory also supports a set of interfaces with other packages (e.g. MATLAB, Python) for data exchange and remote control.

This book aims at covering theoretical and practical aspects of the *how to's* of the development of modelling and simulation-based advanced applications for different types of power system analysis by exploiting the features of PowerFactory. Based on the experience gained throughout the last decade by top expert users and application developers of PowerFactory, the book provides a set of comprehensive guidelines, which are structured by merging innovative solution strategies. A step-by-step procedure is employed to explain the rationale behind the presented applications, which is supported by concise theoretical discussions in order to empathize physical understanding of particular phenomena involved in each case study. All implementations, outlined in the chapters, which were done using the PowerFactory version 15.1.3, are attached as complementary material of the book in pfd files in order to help the reader to follow the explanations given in each chapter and to provide a starting point for the development of additional applications. The book constitutes a valuable reference for formal instruction of power systems for undergraduate and postgraduate students as well as for engineers working in power system operation and planning in different institutions related to the power engineering field.

Chapter 1 provides a thorough review on key issues for developing load flow analysis-based applications in PowerFactory and illustrates three exemplary applications, including the study of topology changes, control of reactive power flow and frequency variation and load profiling. Chapter 2 demonstrates the main capabilities of PowerFactory on steady-state analysis under unbalanced conditions, such as asymmetrical non-transposed transmission lines and presents a discussion on its implications. Preliminary DPL-based applications are presented in Chap. 3, which deals with the implementation of Monte Carlo and Point Estimate methods for probabilistic load flow analysis, and in Chap. 4, which outlines the implementation of a generalized model to solve five-wire power flow problems in distribution systems using the TRX matrix approach.

The DPL implementation of a DC optimal power flow calculation using power transmission distribution factors is described in Chap. 5, including a comparative analysis and validation with respect to the calculations performed in MatPower environment. Chapter 6 introduces a DPL-based application for estimation of technical performance indices for assessing the impact of renewable energy sources.

Preliminary DSL-based applications are given in Chaps. 7–9. A comprehensive review of frequency control and the implementation of a simplified automatic generation frequency controller are provided in Chap. 7, whereas the modelling of combined cycle gas turbine for dynamic simulation studies is presented in Chap. 8. Relevant aspects for modelling of the third order model of double fed induction generators and a dynamic equivalent for wind power plants are discussed in Chap. 9.

Chapters 10–12 introduce advanced DPL applications for the study of low frequency oscillations in power systems. Namely, Chap. 10 concerns the

implementation of a script for performing parameterized modal analysis in order to determine critical operating conditions, whereas Chap. 11 presents a probabilistic approach for risk assessment associated to the occurrence of poorly damped oscillatory modes. In Chap. 12, the theory and implementation of a novel heuristic optimization algorithm, namely the Mean-Variance Mapping Optimization, is given.

Chapters 13–16 illustrate the integration and communication of PowerFactory with other software and simulations programs. Special emphasis on the use of Matlab/Simulink interface and co-simulations, real-time simulation and controller hardware-in-the-loop application for testing are presented.

Chapters 17–19 provide advanced applications of DSL on modelling of very specific components in power systems. DSL implementation of Flexible AC Transmission Systems (FACTS) devices is presented in Chap. 17. Chapter 18 describes the main modelling aspects of active and reactive power controllers combining DSL and Matlab/Simulink. Multi-terminal High voltage Direct current will be an important technology in future transmission systems, thus, important aspects of modelling and simulation are presented in Chap. 19.

Finally, Chap. 20 overviews a couple of good contributions for researchers: the use of one not-well-known PowerFactory analysis function, namely the Parameter Estimation function, and reading data from text files using ElmFile. These elements are used for model parameter identification in the case of one wind farms.

Committed readers will notice that there is some overlap between the chapters, which has been intentionally kept to illustrate how some approaches could share some common implementation steps, despite of being conceived for different purposes. We hope that the book proves to be a useful source of information on the use of PowerFactory and, at the same time, provide the basis for discussion among readers and users with diverse expertise and backgrounds. Given the great variety of applications, which could not be completely covered in a single edition, it is expected that a second edition of the book will be made available soon.

We would like to thank all authors and invited reviewers of the individual chapters for their continuous and valuable support in the different stages of the preparation of this book. A especial website, www.pf4powersystemsapp.com, will be made available from 1 January 2015 to foster active discussions and the possibility of receiving feedback on the book content. Potential collaboration, organization of scientific workshops and exchange of ideas for future works on power system modelling and simulation can also be conducted through this website.

Francisco M. Gonzalez-Longatt
José Luis Rueda

Chapter 1

Load Flow Calculation and Its Application

Muhammad Raza

Abstract This chapter demonstrates the application of load flow analysis using electrical power system planning software. PowerFactory is used as a software tool for network planning, designing, and application of load flow analysis. Description about the load flow command and different components available in PowerFactory for modeling are given, and how to model the power system network is explained. Furthermore, project planning principle is explained through three case studies. In first case study, network topology for low-voltage network is explained and how to select the network component to fulfill the network constraints is demonstrated. In second case study, medium-voltage network is developed and different power control options are demonstrated to determine the power and voltage operating point of the sources. In addition, method of controlling reactive power flow and frequency variation due to imbalance in power have been simulated. In third case study, application of a load profiling has been developed for low-voltage distribution network. In this study, variation of the load throughout a year has been analyzed.

Keywords Load flow tools and application • Newton–Raphson method • Power flow equation • PowerFactory • Distribution network • Electrical networking planning and designing • Load profiling

Electronic supplementary material The online version of this chapter (doi:10.1007/978-3-319-12958-7_1) contains supplementary material, which is available to authorized users.

M. Raza (✉)

Centre d'Innovació Tecnològica en Convertidors Estàtics i Accionaments, Departament d'Enginyeria Elèctrica, Universitat Politècnica de Catalunya, Barcelona, Spain
e-mail: muhammad.raza@citcea.upc.edu

1.1 Introduction

Power flow calculation is a steady-state analysis of electrical power system to calculate the voltage magnitude and angle at all busbars, and power flow in all branches. Load flow analysis is an essential tool in the process of planning, designing, and operation of power systems under different operating conditions and equipment configuration. Based on load flow calculations, different results can be predicted, such as line losses, transformer loading, power exchange between two or more grids, and required voltage control range of transformers and generators. Load flow results are usually required as initial state for other calculations such as RMS simulation and harmonic analysis; also multiple load flow calculation is required for contingency and reliability analysis.

The derivation of power flow equation is based on admittance of the network. In [1], method of formulating admittance matrix is covered in detail. Generally, power flow equation can be expressed using (1.1) and (1.2).

$$P_l = U_l \sum_{m=1}^n U_m \{G_{lm} \cdot \cos(\delta_l - \delta_m) + B_{lm} \cdot \sin(\delta_l - \delta_m)\} \quad (1.1)$$

$$Q_l = U_l \sum_{m=1}^n U_m \{G_{lm} \cdot \sin(\delta_l - \delta_m) - B_{lm} \cdot \cos(\delta_l - \delta_m)\} \quad (1.2)$$

To solve the system, the number of unknown variables is reduced to number of equation by categorizing network buses into three types. These types are so-called load bus (also known as PQ bus), generator bus (commonly known as PV bus), and slack bus (also called as SL bus). Known and unknown parameters associated with buses are defined in Table 1.1. In the next section, how these categorization in PowerFactory can be made are explained.

- **Load Bus:** All the busbars at which voltage magnitude and angle are not controlled or undefined are called PQ bus. At load bus, fixed power to be injected or consumed is given; unknown quantity voltage magnitude and angle are then calculated. Usually, bus connected with load is defined as PQ bus.
- **Generator Bus:** Busbar at which voltage magnitude is controlled or kept constant is called a PV bus. At generator bus, only active power and voltage magnitude is given as known quantity; thus, power equation is solved for voltage angle and reactive power. A busbar connected with active source is usually defined as PV bus.

Table 1.1 List of known and unknown variables according to bus type

Bus type	Given parameter	Unknown parameter
Slack bus	U, δ	P, Q
Generator bus	P, U	Q, δ
Load bus	P, Q	U, δ

- **Slack Bus:** Any busbar at which voltage magnitude and angle kept constant at defined value is called SL bus. The angle of slack node acts as a reference angle in the network. Slack bus in a network acts as a balancing bus; where missing or excess power is generated or consumed by the active source.

Power flow Eqs. (1.1) and (1.2) are nonlinear function; therefore, numerical iteration method is required to find the solution. There are different methods available for solving nonlinear equation [2–4], but for load flow calculation two approaches are well known: Gauss–Seidel method and Newton–Raphson method [1, 5, 6]. In PowerFactory, Newton–Raphson method is implemented for calculating load flow solution.

1.2 Network Models in PowerFactory for Load Flow

PowerFactory is one of the prominent computer aid engineering software for planning, designing, and operation of power system. It has a vast range of library to model electrical power component and different tools for detail analysis. In this section, transformation of the load flow problems into PowerFactory and performing steady-state analysis with different options and design criteria are explained. There are many possibilities to model electrical component either in detail or in simple form in PowerFactory, but only basic modeling component and those options that are necessary for load flow analysis are covered here. For detail, it is recommended to consult the software user manual [7].

In general, from modeling point of view, PowerFactory electrical component model can be categorized into four categories.

- **Busbar:** It is a fundamental electrical network component at which voltage magnitude and angle are calculated. It is also commonly referred as terminal and junction node. In addition, it is possible to model the busbar as a station in detail that includes the different arrangement of terminals and switches such as single-bus system with tie breaker and double-busbar system with bypass busbar. In PowerFactory, it is identified by the class called *ElmTerm*.
- **Source/Generator:** Generators are modeled as voltage-controlled sources that control the voltage magnitude and active power injection into the busbar. The busbar connected to the generator is commonly referred as PV bus. In the network, at least one of the generators must be configured as reference machine, i.e., control the voltage magnitude and angle. The busbar connected to the reference machine is set as SL bus. Most common generator/source models in PowerFactory are synchronous machine (*ElmSym*), double-fed induction machine (*ElmAsmsc*), static generator (*ElmGenstat*), external grid (*ElmXnet*), AC voltage source (*ElmVac*), and AC current source (*ElmIac*).

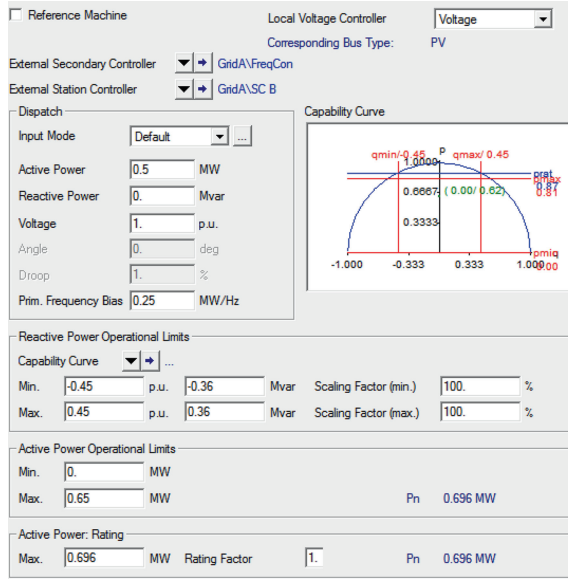
- Shunt Filter/Load:** These electrical network components supply schedule active and reactive power, and the busbar connected with it is known as PQ bus. Generally, a load is modeled in PowerFactory using class *ElmLod*, but additionally, there are other models available depending upon network voltage level such as MV load (*ElmLodmv*) and LV load (*ElmLodlv*). Shunt filter (*ElmShnt*) can be modeled as resistive (R), inductive (L), and capacitive (C) filter either or combination of them. In PowerFactory, asynchronous machine (*ElmAsm*) model is used for motor representation. Motor can either be modeled as single- or double-cage motor.
- Series Branches:** Series branches are the electrical components that connect the two terminals to form the network. It includes lines (*ElmLne*), transformers (*ElmTr**), series reactance (*ElmSind*), series capacitance (*ElmScap*) and filters (*ElmSfilt*), and power control devices. Each of them has subcategories for detail modeling of the network.
 Most common elements in PowerFactory for the development of the network to formulate the load flow problem and required input data are listed below:
- Busbar:** Input data for busbar for load flow calculation are rated line-to-line voltage, e.g., 10 kV. Beside that in PowerFactory, other network informations are also required such as system type (AC), and phase technology (ABC, i.e., three-phase balance network). All these informations defined the network basic property.
- Line:** Transmission line is modeled as PI-network for calculation, and input data required by line are series resistance, series inductance, and shunt capacitance. Usually, these values are given in per km of length. In addition, derating factor and number of parallel line also affect the current-carrying capacity and total impedance of line.
- Load:** Load can be modeled in different ways such as constant impedance, constant current, or constant power. Constant power load is voltage-independent model and represented with stiff voltage characteristics. This method is mostly used in load flow calculation, and it is suitable for medium- to high-voltage network analysis but not for low-voltage network as load power depends on voltage variation. For the representation of the active power demand of motor, load must be modeled as constant current that changes linearly with voltage. For low-voltage load such as lighting or household electrical equipment, constant-impedance model of load is required that changes proportionally to the square of the voltages. For detail analysis, either of load models individually does not cover all the aspect of consumer load. Characteristics of all models can be combined by using the ZIP model of load [7]. In PowerFactory, load type (*TypLod*) must be defined for load ZIP modeling. Input parameters are inserted on load flow page of load type as shown in Fig. 1.1.

Fig. 1.1 PowerFactory setting for voltage dependency load modeling on load flow page of load type

Voltage Dependence P			
Coefficient aP	<input type="text" value="0."/>	Exponent e_aP	<input type="text" value="0."/>
Coefficient bP	<input type="text" value="0."/>	Exponent e_bP	<input type="text" value="1."/>
Coefficient cP	<input type="text" value="1."/>	Exponent e_cP	<input type="text" value="2."/>
Voltage Dependence of Q			
Coefficient aQ	<input type="text" value="0."/>	Exponent e_aQ	<input type="text" value="0."/>
Coefficient bQ	<input type="text" value="0."/>	Exponent e_bQ	<input type="text" value="1."/>
Coefficient cQ	<input type="text" value="1."/>	Exponent e_cQ	<input type="text" value="2."/>

- **Shunt Filter:** It is a representation of R, L, and C or combination of these filters. Usually, shunt filter is modeled as admittance for load flow calculation and the value can be given either directly as layout parameters or calculated from design parameters such as rated voltage, rated power, quality factor, and resonance frequency.
- **Generator:** For load flow calculation, generator is modeled as current-injected source. It is important to define voltage controller scheme for generator, i.e., droop characteristic (DV), fixed set point (PV), or uncontrolled (PQ). If the busbar connected with the generator set to PV mode, then only one generator can control the voltage of that busbar; on the other hand, if the voltage controller scheme is set as droop control, more than one generator can control the voltage of the busbar. Active dispatch and voltage set point must be defined according to input mode. Additionally, to investigate the loading and operation limit violation, it is required to set the active and reactive power maximum and minimum limits. In PowerFactory, it is also possible to implement external station controller and power–frequency controller to control the reactive and active power of the generator. Station control system is used to simulate the behavior as supervisor control system of generators’ reactive power. Similarly, power–frequency controller controls the frequency at a busbar by controlling one or more generators’ active power flow. Load flow page of a static generator (*ElmGen-stat*) is shown in Fig. 1.2.
- **External Grid:** As explain earlier, there must be at least one slack busbar in the network. In PowerFactory, busbar type is defined by the element connected to it. To set a busbar as slack, mode type of generation unit connected to it must be set to SL. Voltage magnitude and angle values are required as input data.

Fig. 1.2 Load flow setting page of PowerFactory static generator model



1.3 Project Planning Using Load Flow Calculation

Generally, power system transmission network is subdivided into three systems for planning, designing an operational point of view. This categorization based on transmission voltage levels, i.e., high-voltage (HV), medium-voltage (MV), and low-voltage (LV) network. Voltage level associated with each network category is listed in Table 1.2. High-voltage network system is used for transferring large electrical power over long distance from city to city or one country to another and connects large power stations. Medium-voltage network usually distributes power within urban and rural area and supplies power to industries having large motors. Low-voltage network supplies power to household customer and small industries [8]. Foremost step in network planning is to construct the single line diagram which is based on geography, and it shows the distribution of load, generation, and layout of the network. Topologically, power systems are constructed and operated as radial, ring-main, and meshed system. Each of these network topologies has some advantages and disadvantages; network design is usually chosen according to the planning criteria such as cost, reliability, and contingency. More detail can be found in [8].

1.3.1 Case Study: Low-Voltage Network

Radial system is the simplest configuration of electrical network. In radial system, there is only one feeding point and usually it is employed at low-voltage-level network. The advantage of this configuration is low investment cost and simple operation, but as a disadvantage, this configuration does not support $n - 1$ criterion;

Table 1.2 Power system network level

Name	Nominal voltage
High voltage	380 kV (400 kV)
	220 kV
	110 kV
Medium voltage	30 kV
	20 kV
	10 kV
	6 kV
Low voltage	400/230 V

that is, in case of failure of a line, the downstream load of that line cannot be restored. On the other hand, radial ring-main topology which is obtained by connecting the line end back to same station as shown in Fig. 1.3 provides reserves for outage of each sections.

PowerFactory project of this case study is available, having title “Load Flow Case LV Network.pfd.” All the required equipment types and necessary data are included in the project. Network configuration is set as the final solution of the project. Six feeders are defined in the project. To activate the coloring according to the feeder, on menu, click “View → Diagram Coloring.” Standard coloring dialog box will open; select option “Other” on basic page. Choose from the list box “Topology” and “Feeders” and press OK.

The list of feeders can be seen by clicking on toolbar option “Edit Relevant Objects for Calculation” and then “Feeders (*.ElmFeeder).” List of other elements can also be displayed in a similar manner. In this configuration, there are six

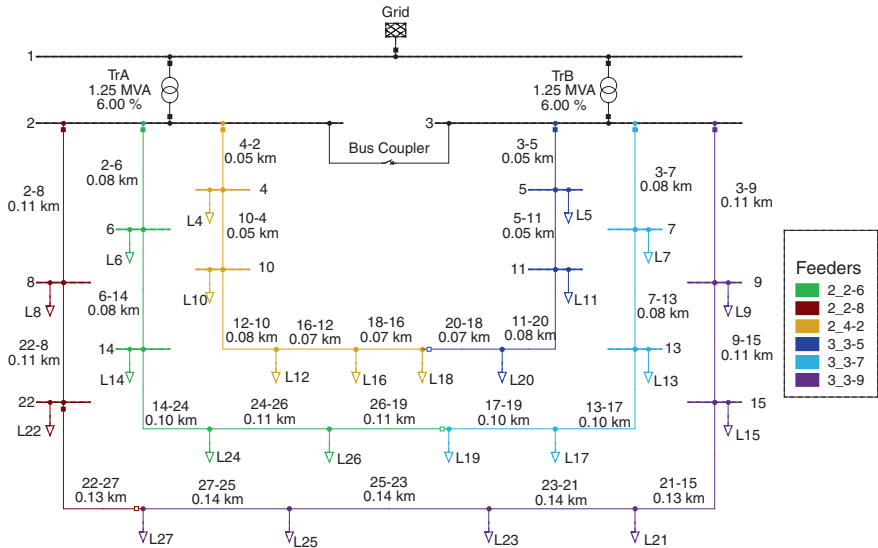


Fig. 1.3 Single line diagram of LV network

outgoing feeders, i.e., line 2–8, 2–6, 4–2, 3–5, 3–7, and 3–9. The term feeder is referred to the outgoing connection from MV or LV substation such as overhead line or cables. Power is transferred via two distribution transformers from MV to LV network. Thus, it provides possibility to continue supply to the customer via one transformer by closing coupling switch in case other transformer is malfunctioning or in schedule services. This means that the thermal loading of each transformer must be at 50 % in average in normal operation. External grid represents the MV distribution system with an ability to withstand load variation of LV network; thus, busbar 1 acts as a slack bus.

Feeders in radial ring-main configuration offer reserves in case of fault from another feeder which is connected to the same substation via open switches as demonstrated in Fig. 1.3. Therefore, it is necessary to keep the loading of each line section lower than its maximum permissible loading to be able to supply the extra load during fault.

Also, system losses and voltage drop can be minimized by changing the location of open switch within the loop. For network planning, total number of consumers, load per consumer, and geographical distance between loads must be known. Also, different line types for selections must also be available. Line types of different cross section are listed in Table 1.3. Also, line types are available in global library of PowerFactory. Network is planned for 50 Hz three-phase balance AC system.

Similarly, transformer specifications are given in Tables 1.4 and 1.5. Additionally, some distribution transformer may equip with tap changer. Through tap changer, additional winding can be added either at HV or LV side, which leads to rise or drop in the voltage level and/or produces phase shift. Here, same type is assigned to both transformers.

Using load and line data that are given in Tables 1.6 and 1.7, respectively, LV network can be established in PowerFactory. In normal operation, switches at

Table 1.3 Line type input data for low-voltage network

Cross section (mm ²)	Rated voltage (kV)	Rated current (kV)	Resistance (Ohm/km)	Reactance (Ohm/km)	Susceptance (μS/km)
50	1.0	0.185	0.3908	0.0785	223.0531
70	1.0	0.230	0.2703	0.0754	245.0442
95	1.0	0.275	0.1950	0.0754	257.6106
120	1.0	0.315	0.1556	0.0723	282.7433
150	1.0	0.355	0.1264	0.0723	285.8849
185	1.0	0.400	0.1013	0.0723	285.8849
240	1.0	0.465	0.0787	0.0723	301.5929
300	1.0	0.525	0.0636	0.0723	314.1593
400	1.0	0.600	0.0494	0.0723	339.2920

Table 1.4 Transformer type data for low-voltage network

Power (MVA)	HV voltage (kV)	LV voltage (kV)	Short-circuit voltage (%)	X/R	Vector group	No load current (%)	No load losses (kW)
1.25	10.0	0.4	6.0	7.073	Yy0	0.192	2.4

Table 1.5 Transformer tap setting

Type	Side	Additional voltage per tap (%)	Phase shift	Tap min. position	Tap neutral position	Tap max. position
Asymmetrical phase shifter	LV	2.5	0.0°	-2	0	+2

Table 1.6 Input data for all loads

Input mode	Active power (MW)	Power factor (inductive)	Technology
P, cos(phi)	0.04	0.98	3Ph-D

Table 1.7 Line input data for low-voltage network

Name	Busbar		Rated voltage (kV)	No. of parallel line	Length (km)	Derating factor
	From	To				
3-5	3	5	1.00	1.00	0.05	1.00
5-11	5	11	1.00	1.00	0.05	1.00
3-7	3	7	1.00	1.00	0.08	1.00
7-13	7	13	1.00	1.00	0.08	1.00
13-17	13	17	1.00	1.00	0.10	1.00
17-19	17	19	1.00	1.00	0.10	1.00
27-25	27	25	1.00	1.00	0.14	1.00
25-23	25	23	1.00	1.00	0.14	1.00
23-21	23	21	1.00	1.00	0.14	1.00
21-15	21	15	1.00	1.00	0.13	1.00
3-9	3	9	1.00	1.00	0.11	1.00
9-15	9	15	1.00	1.00	0.11	1.00
22-8	22	8	1.00	1.00	0.11	1.00
2-8	2	8	1.00	1.00	0.11	1.00
2-6	2	6	1.00	1.00	0.08	1.00
6-14	6	14	1.00	1.00	0.08	1.00
14-24	14	24	1.00	1.00	0.10	1.00
24-26	24	26	1.00	1.00	0.11	1.00
26-19	19	26	1.00	1.00	0.11	1.00
11-20	11	20	1.00	1.00	0.08	1.00
20-18	20	18	1.00	1.00	0.07	1.00
18-16	18	16	1.00	1.00	0.07	1.00
16-12	16	12	1.00	1.00	0.07	1.00
12-10	12	10	1.00	1.00	0.08	1.00
10-4	10	4	1.00	1.00	0.05	1.00
4-2	4	2	1.00	1.00	0.05	1.00
22-27	22	27	1.00	1.00	0.13	1.00

Fig. 1.4 PowerFactory load flow command basic setting

Calculation Method

- ☒ AC Load Flow, balanced, positive sequence
- ☐ AC Load Flow, unbalanced, 3-phase (ABC)
- ☐ DC Load Flow (linear)

Reactive Power Control

- ☐ Automatic Tap Adjust of Transformers
- ☐ Automatic Shunt Adjustment
- ☐ Consider Reactive Power Limits
- ☐ Consider Reactive Power Limits Scaling Factor

Temperature Dependency: Line/Cable Resistances

- ☒ ...at 20°C
- ☐ ...at Maximum Operational Temperature

Load Options

- ☐ Consider Voltage Dependency of Loads
- ☐ Feeder Load Scaling
- ☐ Consider Coincidence of Low-Voltage Loads

Scaling Factor for

Night Storage Heaters %

busbars 18, 19, and 27 and bus coupler are open, thus operating all the feeders in radial configuration. All loads are modeled as constant power system. Initially, assign the line type of minimum cross-section (low-priced) to all lines so that the amount of current flowing through the lines is according to loads and loading of each lines can be determined. Select load flow command from the toolbar or from menu by clicking “Calculation → Load Flow → Load Flow”. Command class is referred as *ComLdf*. Adjust the option as shown in Figs. 1.4 and 1.5, and execute the load flow calculation. These settings execute the balance AC load flow calculation with no reactive power control and using reference machine as an active power control.

Load flow results will be shown in single line diagram. Overloaded cables and low-voltage busbars can be identified via diagram coloring feature. Losses in the network in initial calculation would be high, as many of lines will be overloaded and voltage drop at the end of the feeders might be greater than limits.

In low-voltage network, voltage drop at the end of the feeder must be within the tolerance of $\pm 10\%$ of rated voltage. One of the criteria of the network planning is the consideration of utilization factor which can be directly measured via component loading.

For example, if the line transformer loading is near 20 %, then it considers as overdesign as transformer with higher rated power is installed which leads to unnecessary higher cost. Similarly, a cable can be operated with 100.0 % loading and for short duration of time at higher loading but excess heat and stress on the cable lead to shortening of cable life, which is then considered as underdesign. Generally, loading of the components near 70 % is preferable. PowerFactory

Fig. 1.5 PowerFactory load flow command power control setting

Active Power Control

☒ as Dispatched

☐ according to Secondary Control

☐ according to Primary Control

☐ according to Inertias

☐ Consider Active Power Limits

Balancing

☒ by Reference Machine

☐ by Load at Reference Bus

☐ by Static Generator at Reference Bus

☐ Distributed Slack by Loads

☐ Distributed Slack by Generation (Synchronous Generators)

Reference Bus

Reference Busbar

▼

►

...

Angle

0.

deg

provides many data-handling tools that can be used for defining filters according to custom criteria.

Display the list of lines, on flexible data page of the list, define custom variable by clicking “Define Flexible Data”, and choose loading variable (c:loading). Sort the list of lines in descending order by clicking on the loading column header. Note down lines that have loading above 70 %. Similarly, list of busbars (**ElmTerm*) can be displayed and sorted in ascending order according to voltage magnitude by defining variable (m:u). List of overloaded lines and low-voltage busbars is shown in Table 1.8. It can be noticed that there are lines that are above 100 % loading, and maximum voltage drop at busbars is up to 60 % of rated voltage. Limit violations can be mitigated by choosing cables of higher cross section for overloaded lines; this may improve the voltage drops as well.

It is not only sufficient to select the cable according to normal operational current, but also it is important to consider the worst-case scenario. Worst case in the design would be if either busbar 2 or busbar 3 goes out of service; in this situation, total load has to be supplied by only one transformer. The criteria for lines and transformer would be not to exceed their thermal limit; that is, loading must not be higher than 100 % and voltage drop at feeders end busbar must not drop below 0.8 p.u.

According to these criteria, cable types with the recommended cross-section are listed in Table 1.9. To study worst-case scenario, set busbar 3 to out of service and close switches at busbars 18, 19, and 27. Execute the load flow with same setting and observe the results. With recommended setting in worst case, transformer loading would be 86.4 % and all the lines will have less than 100 % loading. Maximum voltage drop will be up to 0.841 p.u. When the fault is not on the main supplying busbars (bus 2 or 3), there are multiple options via disconnector (i.e., bus coupler, switches at busbars 18, 19, and 27, and feeder switches) to reroute the supply of the consumers that are on the fault branch. For self-exercise, find the

Table 1.8 List of cables above 70 % thermal limits and busbars above 10 % voltage drop limits

Line	Loading (%)	Max. current (kA)	Busbar	Rated voltage (kV)	Measured voltage (kV)	Measured voltage (p.u.)	Voltage drop (%)
3–9	272.49	0.5041	27	0.400	0.235	0.588	–41.200
9–15	236.84	0.4382	25	0.400	0.245	0.612	–38.792
21–15	197.55	0.3655	23	0.400	0.264	0.659	–34.067
4–2	172.79	0.3197	21	0.400	0.291	0.728	–27.189
23–21	153.86	0.2846	15	0.400	0.324	0.810	–18.989
10–4	139.76	0.2586	18	0.400	0.356	0.891	–10.892
3–7	138.65	0.2565	19	0.400	0.357	0.893	–10.732
2–6	138.23	0.2557	9	0.400	0.357	0.893	–10.670
12–10	105.96	0.1960	26	0.400	0.358	0.895	–10.525
25–23	105.68	0.1955	16	0.400	0.360	0.899	–10.093
7–13	105.26	0.1947					
6–14	104.94	0.1941					
3–5	99.18	0.1835					
16–12	71.16	0.1316					
13–17	70.90	0.1312					
14–24	70.69	0.1308					

Table 1.9 Recommended cross-section of cable types for lines

Line name	Cable type
	Cross section (mm ²)
3–9, 2–8	400
3–5, 3–7, 9–15, 22–8, 2–6, 4–2	300
5–11, 7–13, 21–15, 22–27, 6–14, 10–4	240
13–17, 14–24, 11–20, 12–10	185
17–19, 27–25, 23–21, 24–26, 20–18, 16–12	150
25–23	120
26–19, 18–16	95

recommended switches' states in order to supply all the loads when transformer TrB and line 26–19 both are out of service.

As explained earlier, for low-voltage network, preferable load model is constant impedance. In constant-impedance model, actual amount of power consumed by the load depends on voltage level of the connected busbar. If voltage drops, power consumed by the load would be lower than rated value. Besides accuracy of consumer load representation at LV network, constant-impedance model is also useful for solving load flow algorithm when the losses in the network are too high.

In this case study, initially while planning network, same cable type (i.e., 50 mm²) to all lines was assumed.

Table 1.10 Feeder load scaling setting

Name	Load scaling	Scale factor	Power factor
2_2–8	Manual	1.0	No scaling
2_2–6	Manual	1.1	No scaling
3_3–5	Manual	1.2	No scaling
3_3–7	Manual	1.3	No scaling
3_3–9	Manual	1.4	No scaling
2_4–2	Manual	1.5	No scaling

If load flow command is executed for worst-case scenario with initial setting, load flow will not converge. The solution can be found by modeling all loads as constant impedance. In the project, load type with the setting shown in Fig. 1.1 is assigned to all loads. Enable load flow option “Consider Voltage Dependency of Load” on basic page as shown in Fig. 1.4. Execute the load flow command; process will converge. It is to be noticed that power consumption by loads is reduced and it depends upon the connected busbar voltage. Detailed report of the network can be created by executing “Output of Result (*ComSh*)” command. Click “Output Calculation Analysis” icon from the toolbar, and execute the command with option “Grid Summary”.

In normal operation with recommended cable types, total active and reactive power including demand and losses that must be supplied by medium network grid can be determined from external grid model connected to slack busbar.

According to simulation, total demand of low-voltage network is 985.1 kW and 235.2 kVar. During planning, it is also sometimes essential to study the future load increment. Future loads usually are not given as definite value, and it is not necessary to change the load value manually. Scale factor can be introduced for all feeders, and it will be applied on all corresponding loads. In PowerFactory, on feeder load flow page, assign the value according to Table 1.10. To consider the feeder load scaling during load flow calculation, select the option “Feeder Load Scaling” on basic page of load flow command. After execution of load flow, it can be noticed that power flow from each feeding point increases proportionally. Furthermore, due to the increase in power flow, voltage drop increases within the feeders and at feeding busbars too (buses 2 and 3).

As explained earlier, distributed transformer may also have tap setting and usually it increases or decreases tap value automatically according to the set point and voltage range. Adjust the transformer tap control setting on both transformers load flow page according to Fig. 1.6. To enable the automatic transformer control during load flow calculation, select option “Automatic Tap Adjust of Transformer” on basic page of load flow command. After calculating load flow, it can be noticed that voltages at feeder busbar has improved, consequently improving voltage level at all load busbars. Shunt filter is not modeled in the project, but similar to transformer tap control, shunt auto-tap control can be activated by selecting option “Automatic Shunt Adjustment”.

Fig. 1.6 Automatic transformer tap control setting

Tap Changer 1

Neutral: 0 Min:-2 Max: 2

Additional Voltage per Tap 2.5 %

Phase of du 0. deg

Tap Position

☐ According to Measurement Report

Controller, Tap Changer 1

External Tap Controller ...

External Station Controller ...

☒ Automatic Tap Changing

Tap Changer discrete

Controlled Node is at LV

Phase Pos.Seq.

Control Mode V

Setpoint local

☐ Remote Control

Voltage Setpoint p.u.

Lower Bound p.u.

Upper Bound p.u.

Controller Time Constant s

Line Drop Compensation (LDC) none

Thermal Loading Limit

Max. Loading %

1.3.2 Case Study: Medium-Voltage Network

Most of the medium-voltage networks are in ring-meshed configuration as shown in Fig. 1.7. Often, small generators are connected to the network at medium-voltage level. Here, “Grid A” is set as slack generation and “Grid B” is set in PQ mode to represent the equivalent model of low-voltage network level with negative active and reactive power of 985.1 kW and 235.2 kVar, respectively.

Here, network level is 10 kV. PowerFactory project for this case study is available having title “Load Flow Case MV Network.pfd.” Lines and its type parameters as well as transformer and it types parameters are defined in the project. Generator settings are listed in Table 1.11. An additional load is connected at “busbar 36” having value 3,000.0 kW and 1,700.0 kVar. Calculate the three-phase balance load flow calculation with active power control option as dispatch and power balancing by reference machine. Disable all reactive power control options. It can be observed that all the lines and generators loading are less than 100 %. Active power supplies by generators are according to the set points, and all the busbars’ voltage levels are within the range of ± 3 % of nominal voltage, but generators are consuming relatively high reactive power, and network reactive power demand is only fulfilled by “Grid A.” Therefore, here, the planning objective is to determine the appropriate voltage set points for the generator so that reactive power becomes positive and improves the power factor of “Grid A.” Load flow results without power control are shown in

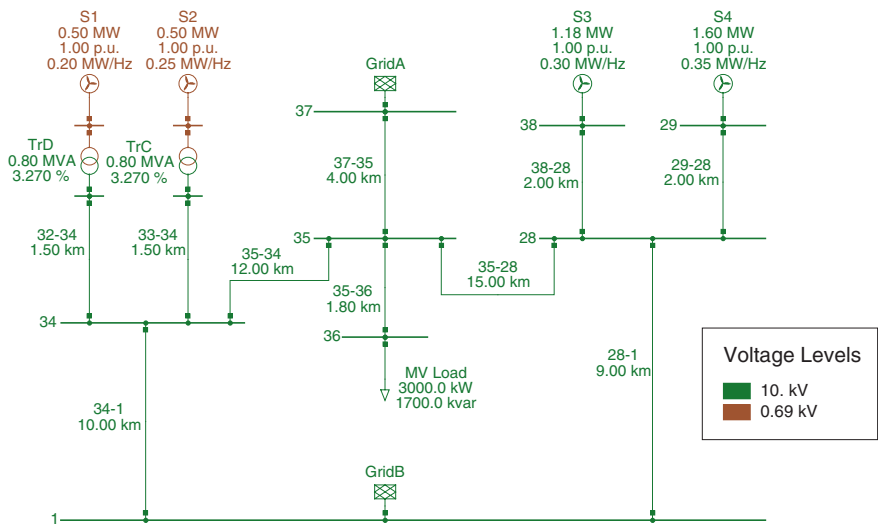


Fig. 1.7 Medium-voltage network in ring-meshed configuration

Table 1.11 Generator setting for medium-voltage network

Name	Bus type	Power (MVA)	Power factor	Active power (MW)	Voltage set point (p.u.)	Pri freq. Bias (MW/Hz)	Reactive power operational limits (MVar)		Active power operational limits (MW)	
							Min	Max	Min	Max
S1	PV	0.8	0.93	0.50	1.0	0.20	-0.38	0.38	0.0	0.70
S2	PV	0.8	0.87	0.50	1.0	0.25	-0.36	0.36	0.0	0.65
S3	PV	1.5	0.95	1.18	1.0	0.30	-0.90	0.90	0.0	1.40
S4	PV	2.0	0.97	1.60	1.0	0.35	-1.12	1.12	0.0	1.90

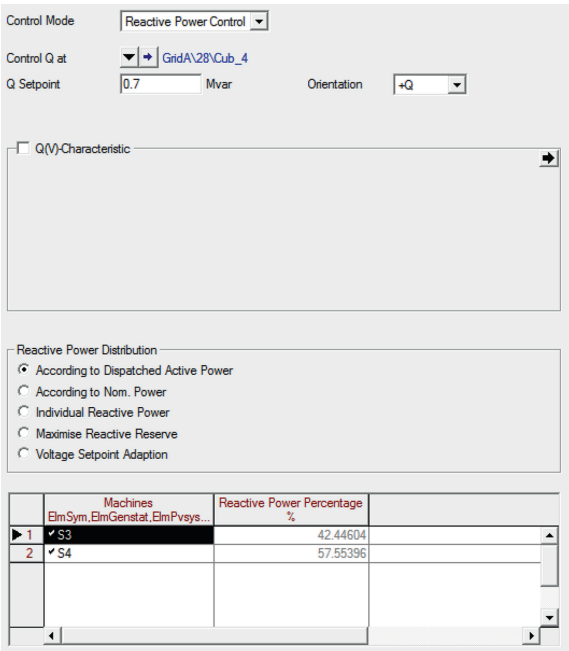
Table 1.12 Power flows of sources in medium-voltage network

Source	Active power (kW)	Reactive power (kVar)	Busbar voltage (p.u.)
Grid A	389.99	2,960.29	1.0
S1	500.00	-59.36	1.0
S2	500.00	-59.36	1.0
S3	1,180.00	-505.19	1.0
S4	1,600.00	-949.14	1.0

Table 1.12. One way to achieve this is by assuming and changing voltage set points of generators manually and recalculating the load flow. This process is repeated and inefficient. In PowerFactory, external station controller (*ElmStactrl*) can be defined to automate this process (Fig. 1.8).

Station controller acts on sources to control the reactive power flow according to the desired characteristic. The objective of station controller can be to control the

Fig. 1.8 PowerFactory station controller “SC A” setting



voltage level at a busbar, maintain power factor, or control reactive power flow in a branch. In the example, two station controllers “SC A” and “SC B” are defined. “SC A” controls generators S3 and S4, and “SC B” controls generators S1 and S2. New station controller can be created by selecting S1 and S2 and then press “Right click → Define → Station Control;” adjust the setting as shown in Fig. 1.9, and press OK. If station controller already exists, you can assign generators to it by selecting generators and pressing “Right click → Add → Station Control.” Setting for station controller “SC A” is shown in Fig. 1.8.

Note “SC A” is controlling reactive power flow through line “35–28” at the side of “busbar 28” and “SC B” is controlling power factor at “busbar 34” at the line “35–34.” In this setting, reactive power flow is enforced from generators toward load connected at “busbar 36” and reducing the amount of reactive power supply from the “Grid A.”

After calculating load flow with the station controllers, it can be seen from Table 1.13 that the demand of reactive power required by the load is now fulfilled by the generators and power factors of all sources are improved. Results also provide the required voltage set points for the generators as well. Results can be verified by disabling the station controllers and assigning the voltage set points to the corresponding generators.

In above example, power is balanced by slack node and generation unit connected to that bus is assumed to have infinite power, but in reality all generation units have active power limits and usually power is balanced by increasing or

Fig. 1.9 PowerFactory station controller “SC B” setting

Control Mode

Power Factor Control

Control Q at

GridA\34\Cub_4

Orientation

+Q

☒ cosphi(P)-Characteristic

Overexcited

Min. Power Factor

0.9

at

Active Power

5.

 MW

Underexcited

Min. Power Factor

0.9

at

Active Power

15.

 MW

Reactive Power Distribution

☒ According to Dispatched Active Power

☐ According to Nom. Power

☐ Individual Reactive Power

☐ Maximise Reactive Reserve

☐ Voltage Setpoint Adaption

	Machines ElmSym, ElmGenstat, ElmPvsys...	Reactive Power Percentage %
1	S1	50.
2	S2	50.

decreasing individual generator power. If in the network, generation is more than the load, then the grid frequency rises, and if generation is less than demand, then overall network frequency drops. This phenomenon can be studied in PowerFactory using Load Flow (*ComLdf*) active power control options “According to Secondary Control” and “According to Primary Control.” Power control option can be changed on load flow page of load flow command. See Fig. 1.5 for detail.

In power balancing by primary control, all generators contribute in balancing the power according to their gain K_{pf} defined on load flow page (*ElmXnet*, *ElmSym*, *ElmGenstat*, etc.); for details see Fig. 1.2. Here, it is important to set the reference machine (*ElmXnet*) into other than SL mode, and it must have nonzero primary frequency biasing. Disable all the station controllers and assign the 0.1 MW/Hz primary frequency biasing to the external grid “Grid A” (*ElmXnet*). Generators parameters are set according to Table 1.11 except voltage set points which are taken from Table 1.13. Enable “Out-of-Service” flag for “Grid B,” and connect load (*ElmLod*) to busbar 1 of value 985.1 kW and 235.2 kVar.

From Table 1.13, it can be noticed that the load demand in the network is 345.58 kW higher than total generation. Thus, change in grid frequency and contribution of each source is calculated using (1.3) and (1.4). After calculating load flow with primary frequency control, the resulting generation from all sources is listed in Table 1.14. Simulation results may differ due to network losses which vary according to power flow direction. Load flow calculation with “According to Secondary Control” active power control option has same principle of balancing the power as of

Table 1.13 Power flow results with station controller

Source	Active power (kW)	Reactive power (kVar)	Busbar voltage (p.u.)	Loading (%)
Grid A	345.58	141.90	1.000000	–
S1	500.00	135.96	1.029818	64.8
S2	500.00	135.96	1.029818	64.8
S3	1,180.00	381.10	1.035410	82.7
S4	1,600.00	516.75	1.036453	84.1

Table 1.14 Load flow results using primary frequency control

Source	Initial generation (kW)	Additional generation (kW)	Total power generation (kW)
Grid A	0.0	29.64	29.64
S1	500.0	59.29	559.29
S2	500.0	74.11	574.11
S3	1,180.0	88.92	1268.92
S4	1,600.0	103.76	1703.76

primary control, but additionally, frequency of the network is controlled by defining external secondary controller on load flow page of all generators (*ElmSym*, *Elm-GenStat*, etc.), see Fig. 1.2 for details. Here, it is required to define the secondary frequency biasing to 0.1 MW/Hz on load flow page of external grid (Grid A).

$$\Delta f = \frac{\sum P_{\text{Load}} - \sum P_{\text{Generation}}}{\sum K_{\text{pf}}} = \frac{345.58}{1.2 \times 1,000} = 0.288 \text{ Hz} \quad (1.3)$$

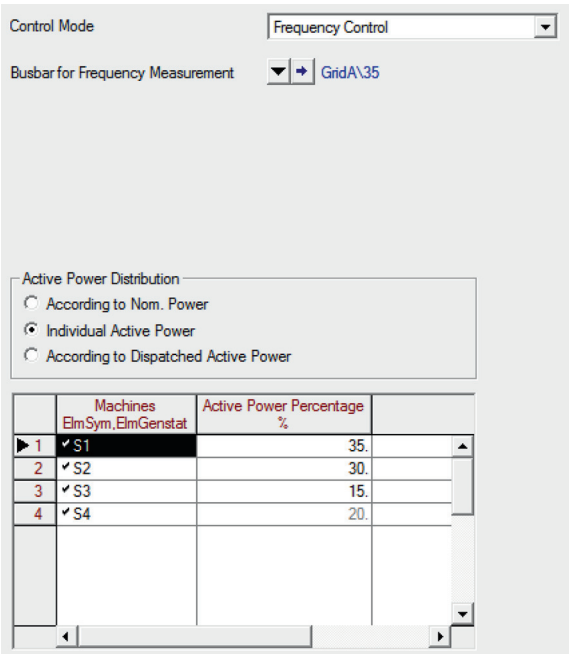
$$P = P_{\text{set}} + K_{\text{pf}} \cdot \Delta f \quad (1.4)$$

Calculate the load flow by setting active power control as “According to Secondary Control.” Results would be the same as shown in Table 1.14. Now, define the power–frequency controller (*ElmSecctrl*) and assign the contribution of each generator according to Fig. 1.10. Power–frequency controller can be defined by selecting all static generators and press “Right click → Define → Power Frequency Controller.”

In the given project, frequency controller is already defined and it can be enabled by unchecking the out-of-services flag. Calculate the load flow; frequency control will try to maintain the frequency at the rated value by increasing or decreasing the generation according to the assign percentage.

The result of load flow calculation using secondary control with external power–frequency controller is shown in Table 1.15. The frequency deviation of the network is controlled to zero, and the contribution of each generators is increased.

Fig. 1.10 PowerFactory power–frequency control setting



From planning point of view, the percentage increase in generation is assigned according to the loading of generators in normal operation. It is also possible that the amount of active power exceeds the maximum limits and generators become overloaded.

In this case, there is a possibility to limit the active power by enabling the option “Consider Active Power Limits” on power control page of load flow command. When frequency control system is not required, such as power balancing with the option as dispatched, PowerFactory provides several other balancing methods as well like by load at reference bus, distributed slack by load, and distributed slack by generation. These options are useful to optimize the losses in the network. Options “Distributed Slack by Load” and “Distributed Slack by Generation” are also effective to converge the load flow result when high load in large network exists and load flow do not converge due to unrealistic data and imbalance in power.

Table 1.15 Load flow results using power–frequency controller

Source	Initial generation (kW)	Additional generation (kW)	Total power generation (kW)
Grid A	0.0	0.0	0.0
S1	500.0	$345.6 \times 0.35 = 120.96$	620.96
S2	500.0	$345.6 \times 0.30 = 103.68$	603.68
S3	1,180.0	$345.6 \times 0.15 = 51.840$	1,231.84
S4	1,600.0	$345.6 \times 0.20 = 69.120$	1,669.12

1.3.3 Case Study: Load Profile Analysis

Load profile or load forecasting is an analysis curve in which variation of demand over a particular period of time can be illustrated. In this analysis, effects on a network stability of load variation are predicted, which is based on previous data usually acquired through electrical bills. Most commonly, profiling is divided into three categories: short-term forecasting is usually performed from 1 h to a week, medium-term forecasting is performed from weeks to a year, and long-term forecasting is usually performed greater than a year. General procedure to analyze the load variation is by applying the load factor on the feeder or loads and calculates the load flow at given time over a period of time. As a result, voltage variation, feeding transformer loading, line loading, and required amount of electrical generation can be estimated [9].

To perform load profile analysis, a DPL script is created to execute the load flow command over the period of a year with 24-h time step. The analysis has been performed on the network shown in Fig. 1.3. In this study, load profile data have been taken from the UK national electrical company “National Grid” record available at [10]. In the profile, data is provided for a whole year at one-hour interval. The load factor of every hour represents the fraction of yearly energy used in that hour and the sum of calendar year is one. For the study, two profiles have been considered, i.e., “Demand” and “Standard Service,” and instead of applying it as hourly, daily fraction has been calculated by summing the hourly data of each day for the whole year. Before implementing DPL script, it is required to set up the network configuration and parameter. To perform this analysis, it is required to change the load with respect to the time. This can be achieved in PowerFactory by defining time-dependent characteristic on the parameter. When the data are given explicitly for a particular load or as per load, then characteristic could be defined on the active and reactive power parameter, but more often, system is analyzed per feeder load [11]. Therefore, characteristics on feeder load scaling on each feeders are applied. Use the following steps to define the characteristics in PowerFactory.

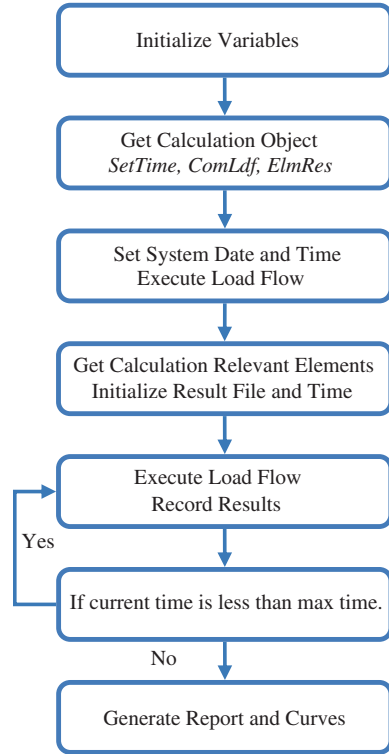
- Define Time Scale:** In the scale folder of project library of equipment type, press new from data manager menu and select time scale (*TriTime*). At time scale dialog, assign appropriate name and select “day of year” as the unit of time trigger. Append the scale value to 365, and assign the cell value from 1 to 365. The time scale will automatically select the time object (*SetTime*) from the study case. Time object is the object in PowerFactory that controls and sets the date and time for all analysis. There are different options available to change date and time. Object parameter “dayofyear” is of interest since per day change in time is required. The button date and time on the dialog will update the “SetTime” to the current system date and time.

- **Define Characteristic:** In the equipment type folder of project library, press new from data manager menu and choose “Characteristics” from radio options and select “Parameter Characteristic-Vector (*ChaVec*).” In *ChaVec* dialog, assign the time scale to the scale pointer that is created in previous step. Set cell values using national grid load profile demand data. Choose usage as relative and linear approximation. Assign the name to this characteristic as “Yearly F1.” Similarly, define second vector characteristic using standard service data and set name as “Yearly F2.” Time scale for the second characteristic would be the same. Characteristic data can be taken from the Excel file provided. Use data from color cell.
- **Assign Characteristics to Feeder Load Scaling:** Feeder load scaling feature has been demonstrated in 1.3.1. Same procedure will be applied here as well, except that load flow command will be executed through DPL script. There are six feeders in the network; to assign the characteristics, right click on scaling factor value (scale0) on feeder load flow page, new characteristic, and then reference and select “Yearly F1.” Scaling method must be manual. Assign “Yearly F1” characteristic to feeders “2_2–8, 2_2–6, and 3_3–5” and “Yearly F2” characteristic to feeders “3_3–7, 3_3–9, and 2_4–2.” Notice, here, scale value for each feeders has been changed, and it is important to note it from the project provided.
- **Load Flow Command Setting (*ComLdf*):** Make sure feeder load scaling option on basic option page of load flow command is enabled.
- **Setup Result File (*ElmRes*):** Create the result file to store the load flow results at each time step. The result file can be created in the study case by pressing new at data manager menu, and in other option, select result (*ElmRes*) in element.
- **Variations (*IntScheme*) and Expansion Stages (*IntSstage*):** Initially, variation must be deactivated.

Through DPL script, peak and off peak loads and the day it will occur, feeder bus voltage, minimum voltage occurrence in feeders, losses in the feeders, and overloaded cables are analyzed [12]. These parameters will provide the essential information to check the network stability, required amount of energy supply, and critical element that violate system constraint. Some of the result can directly acquire through load flow variables, and others can be calculated using those results.

Load flow variables to record results are as follows: cData, cTime, dayofyear for object *SetTime*, c:loading for *ElmLne* and *ElmTr**, m:u for *ElmTerm*, c:LossP, and c:umin for *ElmFeeder*, and m:P:bus1 and m:Q:bus1 for *ElmXnet*. The flow diagram of the DPL script is shown in Fig. 1.11. Script is provided with the project. The file of project file for this study is “Load Profile.pfd.” For network evaluation, results have been visualized using virtual instrument (*SetVipage*, *VisPlot*) graphical board and table report (*ComTablereport*). Using *VisPlot*, five plots have been generated, namely transformer loading, feeder busbar, minimum busbar voltage, demand curve, and overloading lines. Through demand curve, it can be observed that during the month of July and August, there will be high demand and in April and May, there will be lowest power demand. In addition, maximum power demand would be 1,900 kW approximately. Very often, there are minimum and maximum voltage

Fig. 1.11 Flow diagram of load profile analysis



constraint at the feeder busbars. If these limits are violated, transformer tap position has to be changed to improve it. “Feeder terminal” curve demonstrates the feeder busbar voltage variation throughout the year. There are two main feeder busbars in the network, i.e., busbars 2 and 3. Result shows that both busbars’ voltages remain within the range of $\pm 3\%$ of rated voltage throughout the year. Thus, there is no significant requirement to change the transformer tap position. On the other hand, in plot, “minimum terminal voltage” which is the plot of the minimum voltage of all busbars within the feeder provides the information that at peak load for feeder “3_3–9” the busbar voltage could drop up to 0.8 p.u.

During peak period, voltage drop can either be improved by transformer tap position or by shifting some load to another feeder. Changing tap setting will affect the voltage level to other feeders which could lead feeding busbar voltage to exceed the upper limit. For additional study details, see project documentation.

As observed in “Feeder Terminal” plot, feeding busbars have voltage levels within the limits; therefore, more practical approach is to shift some of the loads to adjacent feeder. Voltage profile of the feeder is the curve of the voltage with respect to the distance of the busbars from feeding busbar. Voltage profile provides the information about how much voltage has dropped per busbar distance. This is useful to estimate the number of load to shift from one feeder to another. To do so,

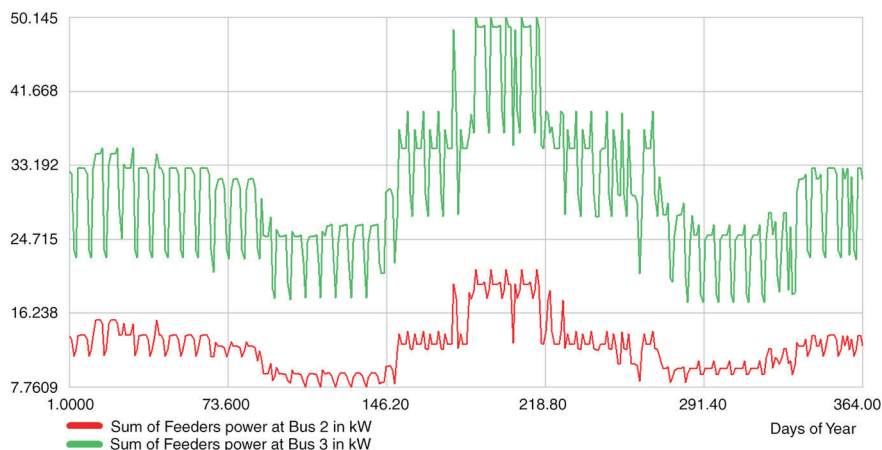


Fig. 1.12 Net active power losses in feeders connected at busbar 2 and in feeders connected at busbar 3

first change the date to peak load date by setting “Day of Year = 195” in *SetTime* object and execute load flow calculation. Open the list of feeders in the data manager, and right click on “3_3–9,” and then “Show → Voltage profile.” Similarly, generate the voltage profile for feeder “2_2–8.” It can be observed from the generated plots that the busbars 25 and 27 are lower than the limit of 0.85 p.u and maximum drop at feeder “2_2–8” is 0.97 p.u; thus, “2_2–8” has sufficient capacity to withstand the extra load. Load can be shifted by creating the switch on line “25–23” at busbar 25 side by pressing “Right click → Create switch.” Open the newly created switch, and close the switch at busbar 27. Re-execute the load flow calculation, and generate the voltage profile curve for both feeders. It can be observed that both feeders’ end busbar voltages are above than 0.9 p.u. This has also improved the overloading of the lines. From the plot of “transformer loading” and “overloading line” that are generated at the initial switch position, critical elements can be identified as well.

According to the network plan, loads are divided into two groups that are fed by two busbars at low-voltage side, i.e., busbars 2 and 3. Thus, it is convenient to study the total power losses per group. Figure 1.12 demonstrates the response of feeder net active power losses fed by busbars 2 and 3. Two curves have been calculated as the summation of the feeder net power losses (c:LossP) connected at busbar 2 and summation of the feeder net power losses connected at busbar 3 over the whole period of time. Plot shows the total losses in the network throughout the year. In addition, voltage profile can be generated per feeding group, i.e., according to busbars 2 and 3. To do so, two additional feeders must be defined at transformers’ connection point. Right click on the transformer “TrA” connection at busbar 2 side, and press “Define → Feeder.” Make sure orientation is set to “Busbar” on basic data page. Similarly, define another feeder on “TrB.” Execute load flow at peak day,

i.e., “Day of Year = 195.” Now, generate the voltage profile for newly created feeders. Through voltage profile curves, comparison between feeders’ path in term of lines loading and busbar voltage can be done.

Very often, electrical network alters throughout the year for specific period of time either due to demand or due to schedule maintenance. It is also very important to take those variation into account in load profiling. This can be done in PowerFactory using variation (*IntScheme*) and expansion stage (*IntSstage*). In the project, three expansion stages are defined that set different loads to out-of-services and put them back into service according to activation time. In data manager, click individual stages to notice loads that are set out-of-services. Variation validity period is from Aug 22, 2014, to Sep 22, 2014. To consider the load services variation for load profile analysis, activate the variation “LoadProfile” by right clicking on variation and press activate. Execute the load profile script. Make sure switches are at initial position and status. Resulting plots clearly show the effects of load change during the variation period; thus, further conclusion can be made based on the curves.

1.4 Conclusion

Load flow analysis is the starting point for the planning, design, and operation of the electrical network. Power system analysis software such as PowerFactory provides advanced tools to conduct different studies. In low-voltage network, design constraints depend upon type of topology employed. The main criteria for LV network planning are thermal and voltage limits. Load flow studies for LV network are usually conducted for finding solution for cable sizing, transformer tape setting, and selection of optimal switch location to minimize the losses. Security of supply for LV network is a critical issue due to topology structure of supplying energy to loads. Therefore, often, network is studied for $n-1$ criteria as well. In LV network, small variation in load has more effect on busbar voltage as compared to HV or MV network; therefore, it is recommended to model voltage dependent load. In medium-voltage network, one of the objectives of performing steady-state analysis is to determine the optimum power and voltage set points for the generators. In addition, load profiling is an important analysis for electrical utility companies to estimate the future energy demand as well as to determine the effects of loads on network long-term stability.

References

1. Grainger JJ, Stevenson WD (1994) Power system analysis. McGraw-Hill, New York
2. Chong EKP, Zak SH (2001) An introduction to optimization. Wiley, New York
3. Newton-Raphson Power Flow: Derivation Using Taylor series (2014) http://www.openelectrical.org/wiki/index.php?title=Newton-Raphson_Power_Flow

4. Deng JJ, Zhao TQ, Chiang HD, Tang Y, Wang Y (2013) Convergence regions of Newton method in power flow studies: numerical studies, international symposium on circuits and systems (ISCAS). IEEE, Beijing
5. IEEE Recommended Practice For Industrial and Commercial Power Systems Analysis (1998) IEEE, USA
6. Trias A (2012) The holomorphic embedding load flow method. Power and Energy Society General Meeting, IEEE, San Diego
7. DIgSILENT GmbH (2013) User manual powerfactory v15.0, Germany
8. Schlabbach J, Rofalski KH (2008) Power system engineering. Wiley, Germany
9. Feinberg EA, Genethliou D (2005) Applied mathematics for restructured electrical power systems: optimization, control, and computational intelligence: Load Forecasting, Springer Science Plus Business Media, USA (Chap. 12)
10. National grid: load profiles (1st May 2014), http://www.nationalgridus.com/niagaramohawk/business/rates/5_load_profile.asp
11. Paoletti S, Garulli A, Vicino A (2012) Electric load forecasting in the presence of active demand, 51st annual conference on decision and control (CDC), IEEE, Maui, HI
12. Li R, Gu C, Li Y, Zhang F (2012) Implementation of load profile test for electricity distribution networks. Power and Energy Society General Meeting, IEEE, San Diego

Chapter 2

Modelling of Transmission Systems Under Unsymmetrical Conditions and Contingency Analysis Using DIgSILENT PowerFactory

J.M. Roldan-Fernandez, Francisco M. Gonzalez-Longatt,
José Luis Rueda and H. Verdejo

Abstract DIgSILENT PowerFactory is a powerful software which includes a power system analysis function designed to cope with large power system power flows, and it handles both DC and AC lines, including all phase combinations (3ph, 2ph and single phase), with/without neutral conductor and ground wires, for both single circuit and mutually coupled parallel circuits. Although power systems are designed and normally operated in balanced (symmetrical) three-phase sinusoidal conditions, there are certain situations that can cause undesired conditions, namely the unbalanced conditions. Uneven distribution of single-phase loads is one of the main unbalanced conditions in distribution level. The objective of this chapter is to provide an extensive review of the main features of steady-state analysis which are included in PowerFactory. Steady-state analysis in PowerFactory covers the normal

Electronic supplementary material The online version of this chapter (doi:10.1007/978-3-319-12958-7_2) contains supplementary material, which is available to authorized users.

J.M. Roldan-Fernandez

Escuela Superior de Ingeniería, Universidad de Sevilla, 20134 Sevilla, Spain

e-mail: jmroldan@us.es

F.M. Gonzalez-Longatt (✉)

School of Electronic, Electrical and Systems Engineering, Loughborough University,

LE11 3TU Loughborough, UK

e-mail: fglongatt@fglongatt.org

J.L. Rueda

Department of Electrical Sustainable Energy, Delft University of Technology,

Mekelweg 4, 2628 CD Delft, The Netherlands

e-mail: J.L.RuedaTorres@tudelft.nl

H. Verdejo

Universidad de Santiago de Chile, Av. Ecuador N°3519, Estación Central,

Santiago de Chile, Chile

e-mail: humberto.verdejo@usach.cl

operation (*ComLdf*) and the contingency analysis (*ComSimoutage*). This chapter has been divided into three distinct parts. The first part presents the main features of modelling overhead transmission lines (*ElmTow*, *TypTow*, *TypGeo*). Secondly, power flow analysis (*ComLfd*) under unbalanced conditions is illustrated. Finally, the third part of this chapter deals with the security assessment or contingency analysis for single time-phase contingency analysis. The deterministic assessment of failure effects under contingencies within a single period (*ComOutage*, *ComNmink*, *ComSimoutage*) is also analysed.

Keywords Contingency analysis • Distribution factors • Overhead line • Unbalanced conditions

2.1 Introduction

A classic assumption about the power systems is that they are designed to be *symmetrical* and *balanced*. A symmetrical set of three-phase voltages of three phases is a set of three voltages in which each voltage is sinusoidal and has the same amplitude, and the set is arranged in such a sequence that the angular phase difference between each member of the set and the one following it, and between the last member and the first, can be expressed as the same multiple of the characteristic angular phase difference of $2/3$ radians. Assuming a set of positive sequence voltage current applied to an electric device, symmetrical refers to the use of the same impedance per phase in that component, e.g. same stator resistance in phases *a*, *b* and *c* in a synchronous generator.

On the other hand, balance conditions refer to the symmetry in electrical quantities (voltage, current, power, etc.), i.e. a circuit in which there are substantially equal currents, either alternating or direct, in all main wires and substantially equal voltages between main wires and between each main wire and neutral (if one exists). A real power system working under normal operational conditions can be assumed symmetrical and balanced; however, there are asymmetrical and unbalanced conditions that must be included in the case of an exact model representation.

One of the intrinsic causes of asymmetric impedance is transmission lines, e.g. asymmetrical geometric configuration of transmission line without transposition is the main source or asymmetrical impedance in power systems. Terminal conditions as phase technologies (3ph, 2ph and single phase, with/without neutral conductor and ground wires) establish unbalance conditions in three-phase symmetrical power system. Uneven distribution of single-phase loads is one of the main unbalanced conditions in distribution level.

The planning, design and operation of power systems require such calculations to analyse the steady-state (quiescent) performance of the power system under various operating conditions and to study the effects of changes in equipment configuration [1, 2]. One of the most common computational procedures used in

power system analysis is the load flow calculation. A number of operating procedures can be analysed, including contingency conditions, such as the loss of a generator, a transmission line, a transformer or a load. These load flow solutions are performed using computer programs designed specifically for this purpose. DIgSILENT PowerFactory is a powerful software that includes a power system analysis function designed to cope with large power system power flows, and it handles both DC and AC lines, including all phase technologies (3ph, 2ph and single phase), with/without neutral conductor and ground wires, for both single circuit and mutually coupled parallel circuits [3].

This chapter is designed to present the main features of power flow analysis on unbalanced conditions using power system analysis function in PowerFactory (*ComLfd*).

2.2 Overhead Line Model

DIgSILENT PowerFactory is very flexible power system analysis software, and it has a very wide range of modelling features in terms of transmission lines. PowerFactory provides models from DC to AC lines over all possible phase technologies (3ph, 2ph and single phase, with/without neutral conductor and ground wires) for both single circuit and mutually coupled parallel circuits.

Table 2.1 shows an overview of all supported options and the corresponding element/type combination. The line element, *ElmLne*, is the most basic branch elements, and it is used to represent the model of the overhead transmission lines. The line element can be used to define single-circuit lines of any phase technology according to Table 2.1.

The number of parallel transmission lines without mutual coupling between each other can be adjusted using the parameter *Number of Parallel Lines*. If the mutual coupling between parallel lines is to be considered, then a line coupling element *ElmTow* has to be defined. In that case, the line element *ElmLne* points to a line

Table 2.1 Overview of line models as available in PowerFactory [3]

System	Phase technology	Element	Type
DC	Unipolar	<i>ElmLne</i>	<i>TypLne</i>
AC, single circuit	1-ph	<i>ElmLne</i>	<i>TypLne</i>
	2-ph	<i>ElmLne</i>	<i>TypLne</i>
	3-ph	<i>ElmLne</i>	<i>TypLne, TypTow, TypGeo</i>
	1-ph with neutral	<i>ElmLne</i>	<i>TypLne</i>
	2-ph with neutral	<i>ElmLne</i>	<i>TypLne</i>
	3-ph with neutral	<i>ElmLne</i>	<i>TypLne</i>
AC, mutually coupled circuits	Any combination of phase technologies	<i>ElmTow</i>	<i>TypTow, TypGeo</i>

coupling element *ElmTow*, which in turn refers to the corresponding tower type *TypTow* or tower geometry type *TypGeo*. However, models based on line types (*TypLne*) are by default *non-frequency dependent*. The user defines the electrical parameters per unit length of the line at a fixed power frequency. These parameters remain unchanged; if the frequency of the simulation changes, i.e. differs from the power frequency, then the program will adjust the reactance and susceptance of the line according to the new frequency. The inductance and capacitance remain, however, unchanged. For certain functions (harmonic load flow, frequency sweeps), the PowerFactory user still has the option to define a frequency characteristic to the parameters in the line type.

For three-phase lines (either single or multiple parallel circuits), the user can choose between two different types of models: *lumped* or *distributed parameters*.

A transmission line is defined as a short-length line if its length is less than 80 km (50 miles). In this case, the shunt capacitance effect is negligible and only the resistance and inductive reactance are considered, and a model based on lumped parameters can be used without any prejudice of the results. If the transmission line has a length between 80 km (50 miles) and 240 km (150 miles), the line is considered a medium-length line and its single-phase equivalent circuit can be represented in a nominal pi circuit configuration. Medium-length line can be modelled using the classical lumped parameters, and there is not mayor negative effect on the accuracy of the obtained results. However, if the line is larger than 240 km, the model must consider parameters uniformly distributed along the line. The appropriate series impedance and shunt capacitance are found by solving the corresponding differential equations, where voltages and currents are described as a function of distance and time. Long transmission lines require the use of distributed parameter models on its calculation, and the model is provided by PowerFactory. For long transmission lines, the distributed parameter model gives highly accurate results and should be the preferred option, while the model with lumped parameters gives accurate enough results for short lines. A line is considered long as long as its length becomes the same order of magnitude of the length of wave of the voltage/current at the grid frequency.

The main features of those models are presented in next subsections.

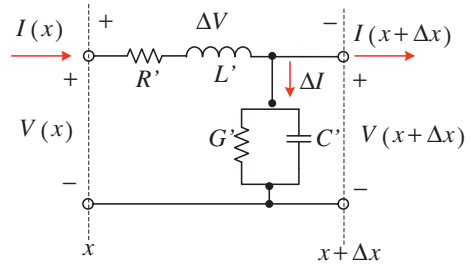
2.2.1 Distributed Parameter Model

The incremental transmission line model is shown in Fig. 2.1. In the case of sinusoidal steady state, it can be modelled by means of two equations:

$$\frac{\partial V}{\partial x} = I(x)Z' \quad (2.1)$$

$$\frac{\partial I}{\partial x} = V(x)Y' \quad (2.2)$$

Fig. 2.1 Incremental model for a line of elemental length



where Z' and Y' are the impedance and the admittance per unit length, respectively, Δx is an elemental length and V and I are the voltage and current, respectively. It must be noted that Z' and Y' are frequency-dependent parameters.

Taking second derivatives (2.1) with respect to x and rearranging the equations to separate the voltage from the current magnitudes, the system of differential can be rewritten as follows:

$$\begin{aligned} V(x) &= K_1 \cdot e^{\gamma \cdot x} + K_2 \cdot e^{-\gamma \cdot x} \\ Z_C \cdot I(x) &= -K_1 \cdot e^{\gamma \cdot x} + K_2 \cdot e^{-\gamma \cdot x} \end{aligned} \quad (2.3)$$

with

$$Z_C = \sqrt{\frac{Z'}{Y'}} \quad (2.4)$$

$$\gamma = \sqrt{Z' \cdot Y'} = \alpha + j\beta \quad (2.5)$$

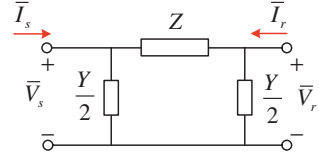
where γ is known as the *propagation factor* and Z_C is the *characteristic impedance* [1]. K_1 and K_2 are integration constants which can be determined from the border conditions at each end of the line. Following the sign convention used in Fig. 2.1, (2.3) can be expressed in the matrix form:

$$\begin{bmatrix} V_r \\ I_r \end{bmatrix} = \begin{bmatrix} \cosh \gamma l & -Z_C \sinh \gamma l \\ \frac{1}{Z_C} \sinh \gamma l & -\cosh \gamma l \end{bmatrix} \begin{bmatrix} V_s \\ I_s \end{bmatrix} \quad (2.6)$$

where subscript r is a magnitude from the receiving end and s from the sending end. An equivalent circuit for (2.6) can be determined as shown in Fig. 2.2. The impedance and admittance of the equivalent circuit are as follows:

$$\begin{aligned} Z &= Z_C \sinh \gamma l \\ Y &= \frac{\cosh \gamma l - 1}{Z_C \sinh \gamma l} \end{aligned} \quad (2.7)$$

Fig. 2.2 Equivalent pi circuit for the line with distributed parameters



2.2.2 Lumped Parameter Model

Lumped parameter model is a simplified model of the distributed parameter model. This model is suitable to describe medium to short lines. The general formulation discussed in this section is valid for any phase configuration by appropriate dimensioning of the impedance and admittance matrices, even though the description is based on a three-phase line without neutral conductor.

According to Fig. 2.3, the equations of the voltages and currents at the sending and receiving ends of the line could be formulated in terms of impedance and admittance matrices. The dimension of the matrices depends on the phase configuration. Therefore, the longitudinal voltage drop along the line (ΔV_i , $i = A, B$ and C) is given by the impedance matrix in the following form:

$$\begin{bmatrix} \bar{V}_{s,A} \\ \bar{V}_{s,B} \\ \bar{V}_{s,C} \end{bmatrix} - \begin{bmatrix} \bar{V}_{r,A} \\ \bar{V}_{r,B} \\ \bar{V}_{r,C} \end{bmatrix} = \begin{bmatrix} \Delta V_A \\ \Delta V_B \\ \Delta V_C \end{bmatrix} = \begin{bmatrix} \bar{Z}_s & \bar{Z}_m & \bar{Z}_m \\ \bar{Z}_m & \bar{Z}_s & \bar{Z}_m \\ \bar{Z}_m & \bar{Z}_m & \bar{Z}_s \end{bmatrix} \begin{bmatrix} \bar{I}_A \\ \bar{I}_B \\ \bar{I}_C \end{bmatrix} \quad (2.8)$$

Following the sign convention assumed in Fig. 2.3, the current at the sending and receiving ends of the line is calculated in terms of the *admittance matrix* as follows:

$$\begin{aligned} \begin{bmatrix} \bar{I}_{s,A} \\ \bar{I}_{s,B} \\ \bar{I}_{s,C} \end{bmatrix} &= \begin{bmatrix} \Delta \bar{I}_{s,A} \\ \Delta \bar{I}_{s,B} \\ \Delta \bar{I}_{s,C} \end{bmatrix} + \begin{bmatrix} \bar{I}_A \\ \bar{I}_B \\ \bar{I}_C \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \bar{Y}_s & \bar{Y}_m & \bar{Y}_m \\ \bar{Y}_m & \bar{Y}_s & \bar{Y}_m \\ \bar{Y}_m & \bar{Y}_m & \bar{Y}_s \end{bmatrix} \cdot \begin{bmatrix} \bar{V}_{s,A} \\ \bar{V}_{s,B} \\ \bar{V}_{s,C} \end{bmatrix} + \begin{bmatrix} \bar{I}_A \\ \bar{I}_B \\ \bar{I}_C \end{bmatrix} \\ \begin{bmatrix} \bar{I}_{r,A} \\ \bar{I}_{r,B} \\ \bar{I}_{r,C} \end{bmatrix} &= \begin{bmatrix} \Delta \bar{I}_{r,A} \\ \Delta \bar{I}_{r,B} \\ \Delta \bar{I}_{r,C} \end{bmatrix} - \begin{bmatrix} \bar{I}_A \\ \bar{I}_B \\ \bar{I}_C \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \bar{Y}_s & \bar{Y}_m & \bar{Y}_m \\ \bar{Y}_m & \bar{Y}_s & \bar{Y}_m \\ \bar{Y}_m & \bar{Y}_m & \bar{Y}_s \end{bmatrix} \cdot \begin{bmatrix} \bar{V}_{r,A} \\ \bar{V}_{r,B} \\ \bar{V}_{r,C} \end{bmatrix} - \begin{bmatrix} \bar{I}_A \\ \bar{I}_B \\ \bar{I}_C \end{bmatrix} \end{aligned} \quad (2.9)$$

Fig. 2.3 Equivalent pi circuit of the line for lumped parameters

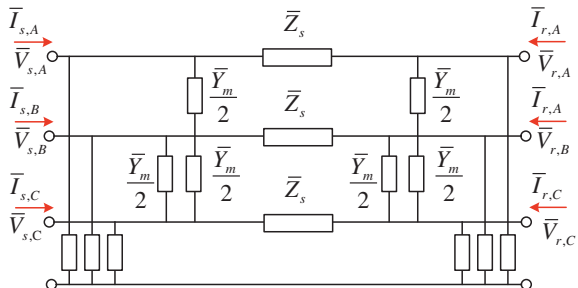
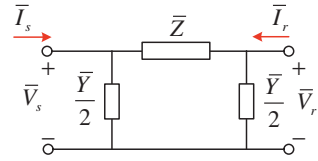


Fig. 2.4 Equivalent single-phase pi circuit for lumped parameters



Equations (2.8) and (2.9) completely define the pi model of the line for lumped parameters. \bar{Y}_s is the sum of all admittance connected to the corresponding phase, and \bar{Y}_m is the negative value of the admittance between two phases. \bar{Z}_s and \bar{Z}_m are the phase impedance and mutual impedance between two phases, respectively.

The circuit shown in Fig. 2.3 can be reduced to the single-phase pi circuit shown in Fig. 2.4.

PowerFactory includes the modelling of lumped parameter model, and it calculates the impedance (Z) and admittance (Y) of the equivalent circuit defined in the *line type (TypLne)* following the equations:

$$\begin{aligned} Z &= Z'_1 l = (R'_1 + j\omega L'_1)l \\ Y &= Y'_1 l = (G'_1 + j\omega C'_1)l \\ G'_1 &= B'_1 \tan \delta_1 \end{aligned} \quad (2.10)$$

where l is the length of the line in km and R'_1 , L'_1 , G'_1 and C'_1 are the line parameters per length unit. The conductance G'_1 can be defined in terms of the insulation factor $\tan \delta_1$. Sending and receiving voltages and currents can be written using the transmission matrix form as follows:

$$\begin{bmatrix} U_s \\ I_s \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} U_r \\ -I_r \end{bmatrix} \quad (2.11)$$

where

$$\begin{aligned} A &= 1 + \frac{1}{2} Z' Y' l^2 \\ B &= Z' l \\ C &= \left(1 + \frac{Z' Y' l^2}{4} \right) Y' l \\ D &= A \end{aligned} \quad (1.12)$$

This lumped parameter model is a simplified model of the distributed parameter model. It can be obtained by series expansion of the hyperbolic functions in (2.7).

The accuracy of the lumped model then depends on the weight of the truncated terms in the series expansion, which in turn depends on the factor (*frequency* \times *length*).

For overhead lines less than 250 km and power frequency, this approximation is very satisfactory and the error can be neglected. For longer lines or higher frequencies, a distributed parameter model will then give a more accurate solution.

Longer lines can be alternatively modelled connecting line sections in cascade.

In *PowerFactory*, the input parameters in the line type (*TypLine*) are defined in terms of positive and zero sequence impedance and admittance. Thus, the conversion from matrix impedance and matrix admittance (2.13) in sequence component is done by the complex transformation matrix as follows:

$$[\bar{Z}_{ABC}] = \begin{bmatrix} \bar{Z}_s & \bar{Z}_m & \bar{Z}_m \\ \bar{Z}_m & \bar{Z}_s & \bar{Z}_m \\ \bar{Z}_m & \bar{Z}_m & \bar{Z}_s \end{bmatrix} \quad [\bar{Y}_{ABC}] = \begin{bmatrix} \bar{Y}_s & \bar{Y}_m & \bar{Y}_m \\ \bar{Y}_m & \bar{Y}_s & \bar{Y}_m \\ \bar{Y}_m & \bar{Y}_m & \bar{Y}_s \end{bmatrix} \quad (2.13)$$

where

$$[\bar{T}_s] = \begin{bmatrix} 1 & 1 & 1 \\ 1 & a^2 & a \\ 1 & a & a^2 \end{bmatrix} \rightarrow [\bar{T}_s]^{-1} = \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \\ 1 & a & a^2 \\ 1 & a^2 & a \end{bmatrix} \quad (2.14)$$

$$[\bar{Z}_{012}] = [\bar{T}_s]^{-1} \times [\bar{Z}_{abc}] \times [\bar{T}_s]$$

Now, the sequence parameters can be calculated as follows:

$$\begin{aligned} [\bar{Z}_{012}] &= \begin{bmatrix} \bar{Z}_0 & 0 & 0 \\ 0 & \bar{Z}_1 & 0 \\ 0 & 0 & \bar{Z}_2 \end{bmatrix} = \begin{bmatrix} \bar{Z}_s + 2\bar{Z}_m & 0 & 0 \\ 0 & \bar{Z}_s - \bar{Z}_m & 0 \\ 0 & 0 & \bar{Z}_s - \bar{Z}_m \end{bmatrix} \\ [\bar{Y}_{012}] &= \begin{bmatrix} \bar{Y}_0 & 0 & 0 \\ 0 & \bar{Y}_1 & 0 \\ 0 & 0 & \bar{Y}_2 \end{bmatrix} = \begin{bmatrix} \bar{Y}_s + 2\bar{Y}_m & 0 & 0 \\ 0 & \bar{Y}_s - \bar{Y}_m & 0 \\ 0 & 0 & \bar{Y}_s - \bar{Y}_m \end{bmatrix} \end{aligned} \quad (2.15)$$

Example 1: Comparison between distributed and lumped parameter models for transmission lines The WSCC 3-machine system, which is well known as P.M Anderson 9-bus, is chosen as case study, and main data of this system appear in Refs. [1, 2] and widely used in the literature for testing purposes. The power system consists of 3 generators, 6 lines, 3 two winding power transformers and 3 loads totalling of 315 MW and 115 Mvar. The static and dynamic data of the system can be found in [1]. The base MVA is 100, and system frequency is 60 Hz.

The test system is depicted in Fig. 2.5, and quantities represented on it are results of classical steady-state calculations. The load flow function (*ComLdf*) is used to compare the accuracy between distributed and lumped models.

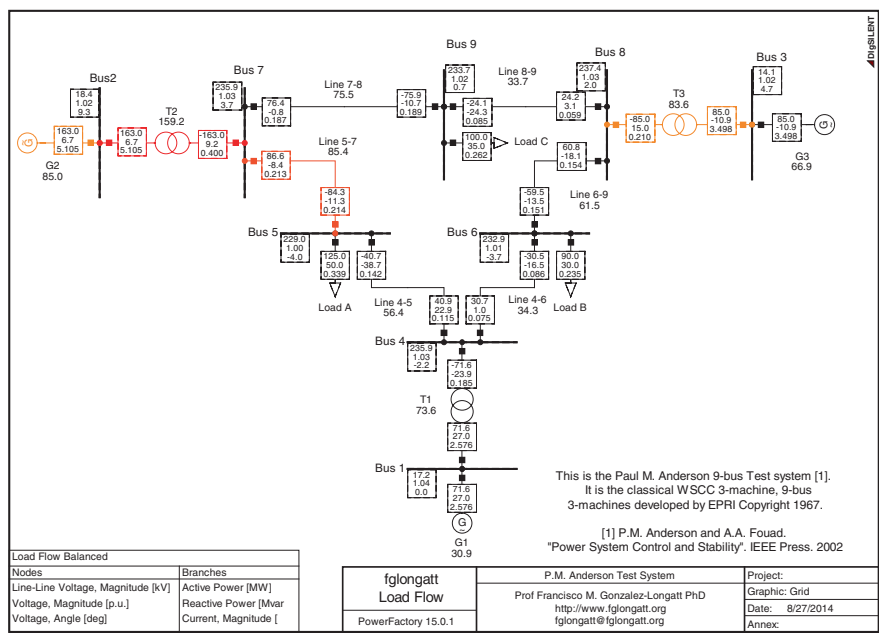


Fig. 2.5 Example 1: test system diagram and parameters

Selection between distributed and lumped models on transmission line is very simple, and it is done at the dialog box of the element transmission line (*ElmLne*). For comparison purposes, two scenarios are created where all transmission lines are using the same model: (i) lumped and (ii) distributed parameters. The calculation method is set to be classical AC load flow using balanced positive sequence and formulation based on power balance equations.

The comparing of results command (*CommDiff*), which is included in PowerFactory, is used to compare results between the two scenarios previously defined. The comparing of results is defined on relative error using as base the scenario of lumped load. Results are presented as percentages as is shown on legend block, bottom left in Fig. 2.6. Colour legend is set in order to help identify the largest changes.

Results show the largest changes are related to power angle of bus voltages ($\phi_{iu} = -7.44\%$ at bus 9), the used of a distributed model implies an increase on the power angles and also there is an effect on the reactive power flows.

The use of lumped parameter model offers several advantages:

- It is extremely easy to be implemented and requires the lowest calculation intensity.
- It is simple and can be used in any phase configuration or number of parallel circuits (dimensions on matrices Z and Y depend on that).

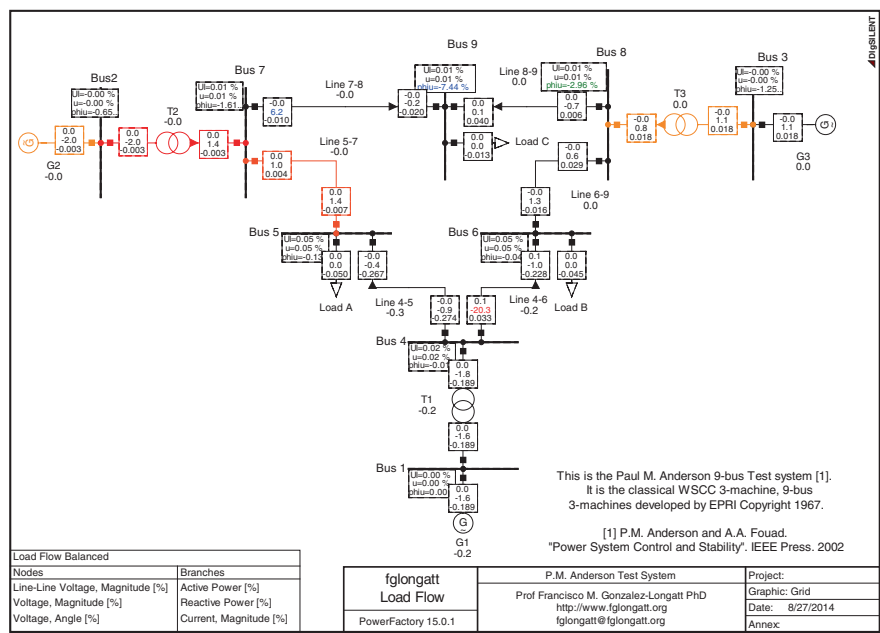


Fig. 2.6 Comparison of load flow results using lumped and distributed models on transmission lines: Example 1

- It is possible to connect several transmission lines in cascade connection (line routes).
- This model can be used on simulation related to transient or dynamic processes (with limitations).

On the other hand, this simple model has few disadvantages:

- There are considerable errors in the use of lumped parameter when the transmission line length is greater than 150 km at 50 Hz and 15 km at 600 Hz.
- Model does not represent frequency dependence on R and L in the case of studies related to frequency response or harmonics.

Model of transmission lines using distributed parameter provides the best results in calculations with overhead transmission lines, considering or not transposition and frequency dependence. Distributed model must be used in any study where frequency changes are relevant to the results, e.g. harmonic load flow and frequency response. There are two important disadvantages on the use of distributed parameter model: (i) it is computationally intensive and (ii) it cannot be used for transient simulations where frequency changes on Z' and Y' .

2.2.3 Tower Model

One important feature of PowerFactory is the way to model lines. Overhead transmission lines and cables are treated alike; they are both instances of the generalized line element (*ElmLne*). However, a simple transmission line can be modelled in very different ways, depending on type of the line selected (more details can be found in the PowerFactory user's manual [3]):

- The simplest way is the *TypLne* object type. It was used on the previous section to describe the line directly providing the electrical parameters (the user can select if the type is defined for an overhead line or a cable).
- The tower model or tower types (*TypTow* and *TypGeo*) are used where the geometrical description (coordinates) and conductor electrical parameters are clearly specified or known, and the electrical parameters are calculated from these data.

The use of one type or other depend on the data available or simulation purposes, for very simple simulations *TypLne* can be used without major difficulties or effecting results but if there is any specific phenomenon (e.g. mutual coupling) of interest, this towers type must be used.

Example 2: Comparison between line model and tower model for transmission lines A typical 230-kV, 60-Hz transmission line is considered in this example, it is named *Test System 2*, and geometrical configuration and main characteristics are indicated in Fig. 2.7.

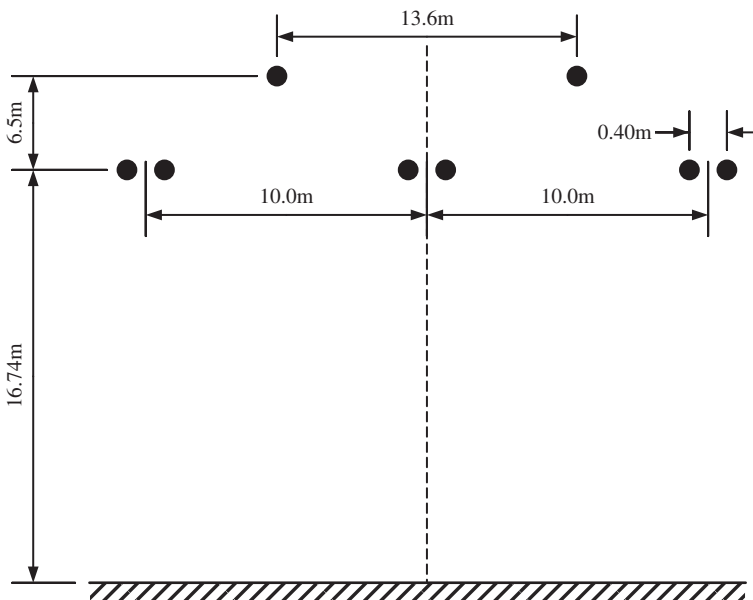


Fig. 2.7 *Test System 2* typical geometrical disposition of a 230-kV transmission line

This transmission line is used initially to evaluate the accuracy of the results provided by PowerFactory in terms of sequence impedance. The transmission line is modelled in ATP (*Alternative Transient Program*), and the routine line constant is used to obtain the sequence impedance and then compared with the results obtained from PowerFactory.

Initially, the transmission line shown in Fig. 2.7 must be modelled in PowerFactory using a tower model. This is a very simple procedure, and it is clearly and explained in detail in Sect. 9.3.4, Example Line Couplings on the PowerFactory User Manual. Here, for space limitations, the main aspects are presented and discussed.

The first step is to create the conductor types (*TypCon*) for phase and earth conductors. Phase conductor ACSR 1033.5 KCM 54/7, $r_{dc,p} = 0.0684$ Ohm/Km, $RMG_p = 1.28$ cm. Earth conductor: Alumoweld 7#9, $r_{dc,g} = 2.362$ Ohm/km, $RMG_g = 0.0567$ cm. Figure 2.8 shows all details of the created conductor types for this example.

Now, the tower geometry type (*TypeGeo*) must be defined following the dimensions provided in Fig. 2.7, and the implementation in PowerFactory is shown in Fig. 2.9.

PowerFactory is power simulation software, and it allows retrieving all the electrical parameter calculations for the transmission line in the output windows. Pressing the “Calculate” bottom, the following matrix is displayed: (i) *Natural impedance matrix*, (ii) *reduced impedance matrix* (3×3), (iii) *symmetrical impedance matrix* (sequences 0, 1 and 2), (iv) *reduced admittance matrix* (3×3) and (v) *symmetrical admittance matrix* (sequences 0, 1 and 2). For illustrative purposes, Fig. 2.10 shows symmetrical impedance and admittance matrixes (sequences 0, 1 and 2) as presented on the output window.

One of the ATP’s special features is to efficiently calculate phenomena in transmission lines. For actual multiphase overhead transmission lines, the following models are applicable: (multistage) lumped pi model, distributed parameter, untransposed-type model, distributed parameter, transposed-type model and frequency-dependent model—J. Marti model. In EMTP (Electromagnetic Transients Program) and ATP, these model data are directly calculated in the subroutine named *Line Constants*, introducing geometrical parameters of transmission lines.

Tables 2.2 and 2.3 show the comparison results between PowerFactory and ATP for sequence impedance and admittance, respectively. Relative error (%) is calculated in each case, using PowerFactory as base for comparisons.

Results provided by PowerFactory and ATP routine are quite similar in magnitudes; in fact, the largest difference is in the case of the positive sequence admittance (3.47 %).

Now, the typical 230-kV transmission line presented in Fig. 2.7 is used to substitute all transmission in the *Test System 1*, WSCC 3-machine system. Classical load flow function in PowerFactory are calculated in order to compare results between the use of *TypLne* object type and tower type (*TypTow*). Figure 2.11 shows changes (%) on the load flow solution; bus voltage angles exhibit the largest

Conductor Type - Equipment Type Library\ACSR 1033.5 KCM 54/7.TypCon *

Basic Data	Name	ACSR 1033.5 KCM 54/7		OK
Load Flow	Nominal Voltage	230.	kV	Cancel
VDE/IEC Short-Circuit	Nominal Current	1.047	kA	
Complete Short-Circuit	Number of Subconductors	2		
ANSI Short-Circuit	Bundle Spacing	0.4	m	
IEC 61363	Conductor Model <input checked="" type="radio"/> Solid Conductor <input type="radio"/> Tubular Conductor			
RMS-Simulation	(Sub-)Conductor			
EMT-Simulation	DC-Resistance (20°C)	0.0648	Ohm/km	
Harmonics/Power Quality	GMR (Equivalent Radius)	128.	mm	
Optimal Power Flow	Outer Diameter	523.7	mm	
Reliability	<input type="checkbox"/> Skin effect			
Generation Adequacy				
Description				

Conductor Type - Equipment Type Library\Alumoweld 7#9.TypCon

Basic Data	Name	Alumoweld 7#9		OK
Load Flow	Nominal Voltage	230.	kV	Cancel
VDE/IEC Short-Circuit	Nominal Current	0.2	kA	
Complete Short-Circuit	Number of Subconductors	1		
ANSI Short-Circuit	Conductor Model <input checked="" type="radio"/> Solid Conductor <input type="radio"/> Tubular Conductor			
IEC 61363	(Sub-)Conductor			
RMS-Simulation	DC-Resistance (20°C)	2.362	Ohm/km	
EMT-Simulation	GMR (Equivalent Radius)	5.67	mm	
Harmonics/Power Quality	Outer Diameter	11.44	mm	
Optimal Power Flow	<input type="checkbox"/> Skin effect			
Reliability				
Generation Adequacy				
Description				

Fig. 2.8 Conductor types (TypCon) for phase and earth conductors of a typical 230-kV transmission line

changes together with the reactive power flows. *TypTow* provided a more detailed and realistic model as consequence is expected to provide less optimistic results. The use of the tower-type model is a computationally efficient calculation providing more realistic results. Authors suggest the use of this model as much as known data allow.

Tower Type - Equipment Type Library\Tower Type 230 kV.TypTow

Basic Data

Load Flow

VDE/IEC Short-Circuit

Complete Short-Circuit

ANSI Short-Circuit

IEC 61363

RMS-Simulation

EMT-Simulation

Harmonics/Power Quality

Optimal Power Flow

Reliability

Generation Adequacy

Description

General Geometry

Name: Tower Type 230 kV

Nominal Frequency: 60. Hz

Number of Earth Wires: 2

Number of Line Circuits: 1 Transposition: none

Input Mode

☒ Geometrical Parameter

☐ Electrical Parameter

Earth Resistivity: 100. Ohmm

Types of Earth Conductors:

	Conductor Types	TypCon
Earth Conductor 1	Alumoweld 7#9	
Earth Conductor 2	Alumoweld 7#9	

Conductor Types of Line Circuits:

	Conductor Types	TypCon	Num. of Phases	Transposition
Circuit 1	ACSR 1033.5 KCM 54/7		3.	

Coordinate of Earth Conductors [m]:

	X	Y
Earth Conductor 1	3.2	23.54
Earth Conductor 2	16.8	23.54

Coordinate of Line Circuits [m]:

	X1	X2	X3	Y1	Y2	Y3
Circuit 1	0.	10.	20.	16.74	16.74	16.74

OK

Cancel

Calculate

Fig. 2.9 Tower geometry types (TypGeo) of a typical 230-kV transmission line *Test System 2*

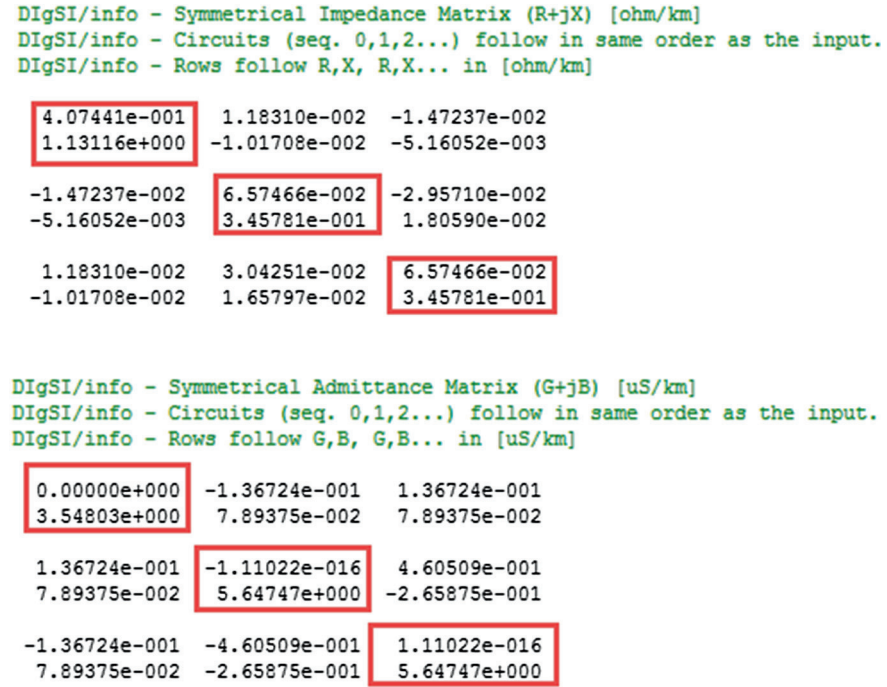


Fig. 2.10 Illustrative results of symmetrical impedance and admittance matrix *Test System 2*

Table 2.2 Results comparison: sequence impedance (Z)

	PowerFactory		ATP line constant		Error	
	R [Ω/km]	X [Ω/km]	R [Ω/km]	X [Ω/km]	R [%]	X [%]
Seq 0	0.40744	1.13115	0.40861	1.14514	0.28750	1.23700
Seq 1, 2	0.06574	0.34578	0.06663	0.35092	1.35500	1.48700

Table 2.3 Results comparison: sequence admittance (Y)

	PowerFactory	ATP line constant	Error
	Y [μΩ ⁻¹ /Km]	Y [μΩ ⁻¹ /Km]	Y [%]
Seq 0	3.5480	3.4648	2.3456
Seq 1, 2	5.6475	5.4512	3.4758

2.3 Power Flow Analysis Under Unbalanced Conditions

The load flow function in PowerFactory allows several features, AC systems, unbalanced, 3-phase (*abc*). This analysis function performs load flow calculations for a multiphase network representation. It can be used for analysing unbalances of 3-phase systems, e.g. introduced by unbalanced loads or non-transposed lines, or

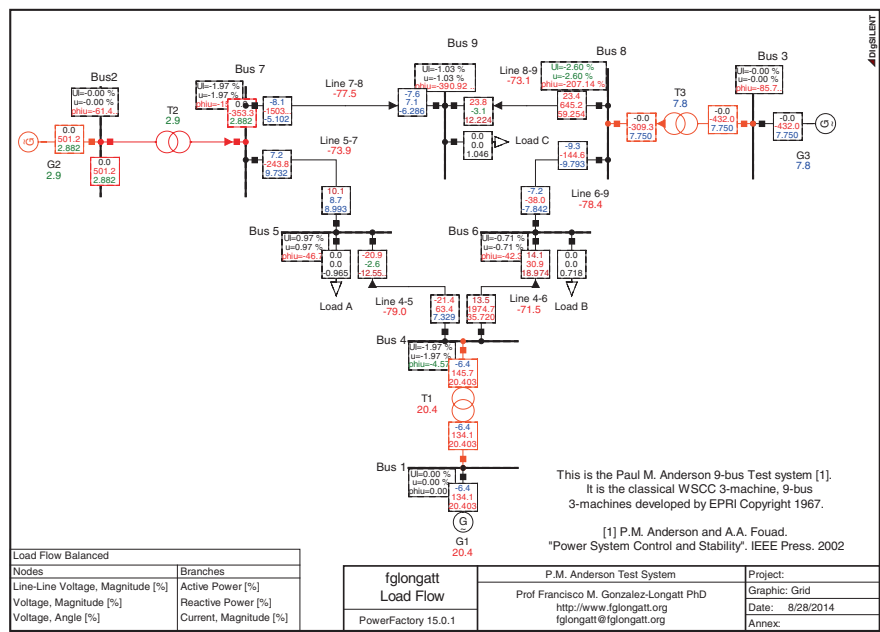


Fig. 2.11 Comparison of load flow results using *TypLne* object type and *TypTow* on transmission lines: Example 1

for analysing all kinds of unbalanced system technologies, such as single-phase or two-phase systems (with or without neutral return).

In PowerFactory, the nodal equations used to represent the analysed networks are implemented using two different formulations, *Newton–Raphson with current equations* or *Newton–Raphson with power equations*. The selection of the method used to formulate the nodal equations is user-defined and should be selected based on the type of network to be calculated. The standard Newton–Raphson algorithm using the power equation formulation is typically used for large, meshed transmission systems with a relatively high *X/R* ratio, especially when heavily loaded, and typically, this formulation usually converges best. However, distribution systems are very different to transmission system. “Current Equations” formulation usually allows a better convergence in distribution systems, especially unbalanced distribution systems. It must be noticed, this is not a general rule, each specific case of no convergence must be specifically analysed.

The unbalanced AC load flow could be performed for a multiphase network representation. It can be used in order to analyse unbalances of 3-phase systems, e.g. introduced by unbalanced loads or non-transposed lines, or for analysing all kinds of unbalanced system technologies, such as single-phase or two-phase systems (with or without neutral return). This option is available only for AC load flow calculations.

Example 3: Magnetically Coupled transmission lines The line coupling element (*ElmTow*) can be used to simulate magnetically coupled transmission lines. Transmission lines in multiple circuits sharing the same support structure (tower) or with very short distance between them are magnetically coupled. In this example, an academic, three-phase, double-circuit, 230-kV, non-transposed transmission system is shown in Fig. 2.12 (*Test System 3*), as distances. Distances shown in Fig. 2.12 are only for academic purposes and should not be taken as rule, and that configuration is designed in order to increase the magnetic and electric coupling between circuits. *Test System 1*, WSCC 3-machine system, has six transmission lines operating at 230 kV, and in this example, tower shown in Fig. 2.12 is issued in order to provide coupling between circuits.

A very simple procedure is followed in PowerFactory to create a *Line Couplings Element* (*ElmTow*) and the *TypLne* object type, and illustrative information is presented in Figs. 2.13 and 2.14. More details of the process to be followed can be found in PowerFactory user's manual [3].

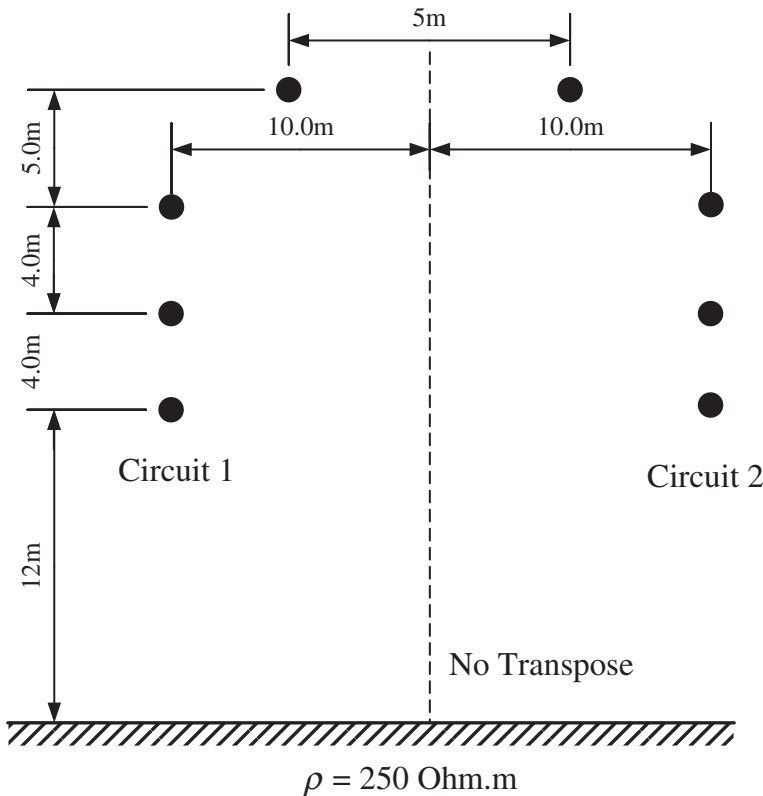


Fig. 2.12 *Test System 3* Illustrative case of a double-circuit, 230-kV transmission line

Tower Type - Equipment Type Library\Tower Type Double Circuit.TypTow

General | Geometry

Name: Tower Type Double Circuit

Nominal Frequency: 60 Hz

Number of Earth Wires: 2

Number of Line Circuits: 2 Transposition: none

Input Mode: ☒ Geometrical Parameter ☐ Electrical Parameter

Earth Resistivity: 250 Ohmm

Types of Earth Conductors:

	Conductor Types	TypCon
Earth Conductor 1	Alumoweld 7#9	
Earth Conductor 2	Alumoweld 7#9	

Conductor Types of Line Circuits:

	Conductor Types	TypCon	Num. of Phases	Transposition
Circuit 1	ACSR 1033.5 KCM 54/7		3	<input type="checkbox"/>
Circuit 2	ACSR 1033.5 KCM 54/7		3	<input type="checkbox"/>

Geometry

Coordinate of Earth Conductors [m]:

	X	Y
Earth Conductor 1	5	25
Earth Conductor 2	15	25

Coordinate of Line Circuits [m]:

	X1	X2	X3	Y1	Y2	Y3
Circuit 1	0	0	0	12	16	20
Circuit 2	20	20	20	12	16	20

Fig. 2.13 Tower types (*TypTow*) used on double-circuit, 230-kV transmission line *Test System 3*

All transmission in the *Test System 1*, WSCC 3-machine system, is using the double-circuit transmission systems as shown in Fig. 2.12. Unbalanced load flow function in PowerFactory is used to calculate the magnitude of line-to-neutral

Line Couplings - Grid\Line Couplings 4-5 and 4-6.ElmTow

Name: Line Couplings 4-5 and 4-6

Route: ...

OK Cancel Calculate

Basic Data EMT-Simulation Description

☐ Out of Service

Line Model: ☒ Lumped Parameter (PI) ☐ Distributed Parameter

Number of Overhead Line Systems: 1

Geometries:

Type	TypTow	TypGeo	Distance	m
G1	Tower Type Double Circuit		0	

Earth Wires:

Type	TypCon	Max.Sag	m
G1/E1	Alumoweld 793	0	
G1/E2	Alumoweld 793	0	

Line Names:

Circuit	ElmLine	ElmRoute	Polarity	Type	TypCon	Max...	Transposition	Phasing
G1/C1	Line 4-5	Terminal i	ACSR 1033.5 KCM 54/7	0		A-B-C		
G1/C2	Line 4-6	Terminal i	ACSR 1033.5 KCM 54/7	0		A-B-C		

Fig. 2.14 Tower element (*ElmTow*) used on double-circuit, 230-kV transmission line *Test System 3*

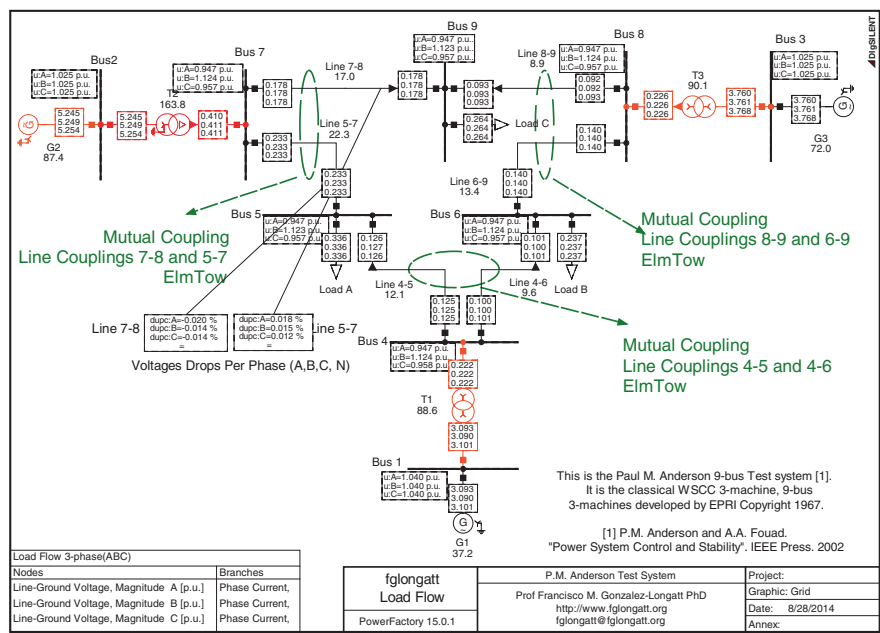


Fig. 2.15 Unbalanced load flow results considering magnetic coupling between transmission systems: Example 3

voltages (p.u) in all nodes and magnitude of phase currents (kA). Figure 2.11 shows the results of the unbalanced load flow solution; bus voltage magnitudes exhibit a considerable unbalance where phase *B* has the highest line-to-neutral voltage ($u_B = 1.124$ p.u.).

The detected voltage imbalance is caused by the shunt currents created by the capacitive coupling with earth and grounding wires. On the other hand, magnetic coupling is providing a relatively low unbalance; voltage drops per phase are presented in Fig. 2.11 where phases A and B show the largest voltage drop across the series impedance (Fig. 2.15).

2.4 Contingency Analysis

In general terms, *contingency analysis* can be defined as the evaluation of the security degree of a power system. Contingency analysis is generally related to the analysis of abnormal system conditions. This is a crucial problem, both in planning and in daily operation. A common criterion is to consider contingencies as a single outage of any system element (generator, transmission line, transformer or reactor) and evaluate the post-contingency state. This is known as the $N - 1$ security criterion. Other contingencies to be taken into account are simultaneous outages of

double-circuit lines that share towers in a significant part of the line path. The outage of the largest generator in an area and any of the interconnection lines with the rest of the system is another contingency to be analysed.

Contingency analyses are used to determine the state of the network after an outage of one ($N - 1$) or multiple elements ($N - k$). Therefore, a load flow must be performed for each selected contingency. This chapter deals with the most basic but typically used contingency analysis: deterministic contingency analysis. PowerFactory contingency analysis module offers two contingency analysis methods [3].

- **Single Time-Phase Contingency Analysis:** The deterministic assessment of failure effects under given contingencies, within a single time period. Here, only one post-fault load flow is analysed per contingency case.
- **Multiple Time-Phase Contingency Analysis:** Deterministic assessment of failure effects under given contingencies. It is performed over different time periods, each of which defines a time elapsed after the contingency occurred. It allows the definition of user-defined post-fault actions.

In both cases, the prefault and post-fault load flows are compared to the specified loading and voltage limits and the reports are generated from the comparison between prefault and post-fault load flows. In PowerFactory, the term *Fault Case* is used to define a contingency. Two concepts must be defined in order to understand the functionality of this module [3]:

- **Contingencies:** These are objects in PowerFactory of the class *ComOutage* which are used to represent contingencies. They are defined by a set of events which represent the originating faults over time and the following fault clearing and post-fault actions.
- **Time Phases:** These represent points in time at which the steady-state operational point of the network under analysis is calculated. Each time phase is defined via a user-defined post-contingency time. The post-contingency time defines the end of a phase, that is, the point in time at which the steady state of the network is calculated.

As mentioned before, the single time-phase contingency analysis function first performs a *prefault load flow*. Following this, it performs a corresponding post-contingency load flow for a single time phase and contingency [3]. The function calculates the initial consequences of the contingencies, regardless of the operational measures to mitigate violations in the system. Moreover, automatic transformer tap changer and switchable shunts can be considered as long as their time constants are smaller than the current *post-contingency time*. The results of the contingency analysis with multiple time phases correspond to the steady-state operational point of the network being studied, at every post-contingency time for each of the defined contingencies. Compressive details about procedure to perform a contingency analysis using PowerFactory can be found in the PowerFactory user's manual [3].

The *Contingency Analysis* command (*ComSimoutage*) performs a load flow calculation to determine the operational point of the network under no-fault

conditions. The command contains *Contingency Cases* (*ComOutage* objects) which define one or more elements that are taken out of service simultaneously. Following the calculation of the base load flow, a contingency load flow for each of these contingencies is calculated. This calculation considers the post-fault thermal ratings of branch elements, transformer tap changer controller time constants and automatic shunt compensators [3].

Contingency cases can be generated by two means: the *Contingency Definition* command (*ComNmink*) or via the definition and use of fault cases and fault groups. In the first case, the contingencies can be created using the *Contingency Definition* command available in its toolbar icon. Another way is by right clicking on a selection of elements in the single line diagram and selecting the option:

Calculate/Contingency Analysis comSimoutage. The corresponding dialog is shown in Fig. 2.16.

In the second case, contingency cases can be created using references to user-defined *fault cases* and *fault groups* from the *Operational Library*.

Either an $N - 1$ or $N - 2$ outage simulation for the selected elements can be prepared. Additional $n - k$ outage for mutually coupled lines/cables is available. Moreover, the Contingency Definition command optionally allows selecting lines/cables, transformers, series reactors, series capacitors and/or generators so as to create contingencies.

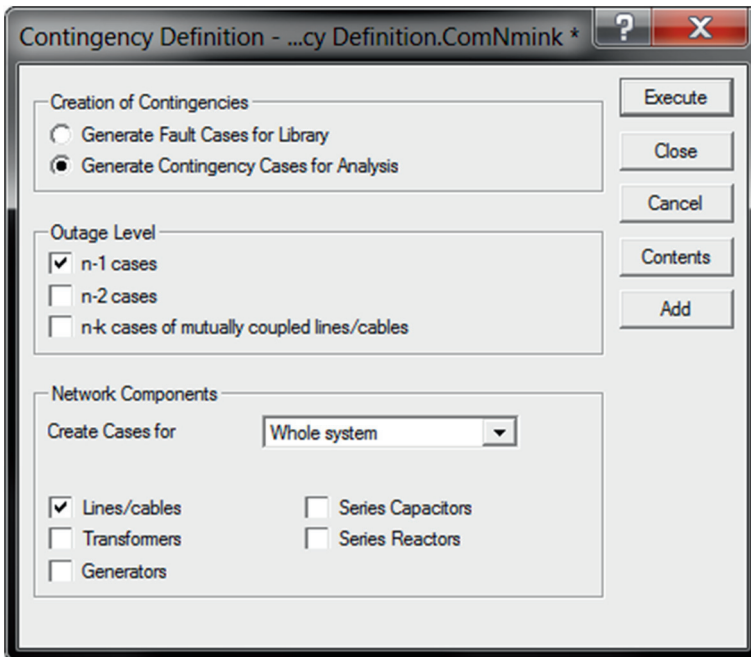


Fig. 2.16 Contingency definition command (*ComNmink*)

2.4.1 Single Time-Phase Contingency

When the command contingency analysis is executed, the dialog shown in Fig. 2.17 is displayed. The contingency analysis limits can be set individually for each terminal and branch element (in the load flow page of the element's dialogue) or globally in the limits for recording field of the contingency analysis command. The calculated result is stored in the result file whenever one of the constraints (individual or global) is violated. The following options can be selected: (i) *AC Load Flow Calculation*: The classical AC load flow analysis to calculate the state of the power system after each contingency. (ii) *DC Load Flow Calculation*: With this option the linear DC load flow method is used to calculate the power system state after each contingency. (iii) *DC Load Flow + AC Load Flow for Critical Cases*: The contingency analysis will perform two runs. First, it will use a linear DC load flow method to calculate the active power flow per contingency case. If for certain contingencies, loadings are detected to be outside a certain threshold, then for these cases, the contingency analysis will recalculate the post-fault load flow using the iterative AC load flow method [3].

The parameters in this section set the global threshold used to determine whether a calculated result is recorded in the *Results object*.

Contingency cases can be displayed by clicking on *show* button or add by *Add Cases/Groups*. This second button is used to create the contingency cases (*Com-Outcome* objects) based on fault cases and/or fault groups. A fault case contains

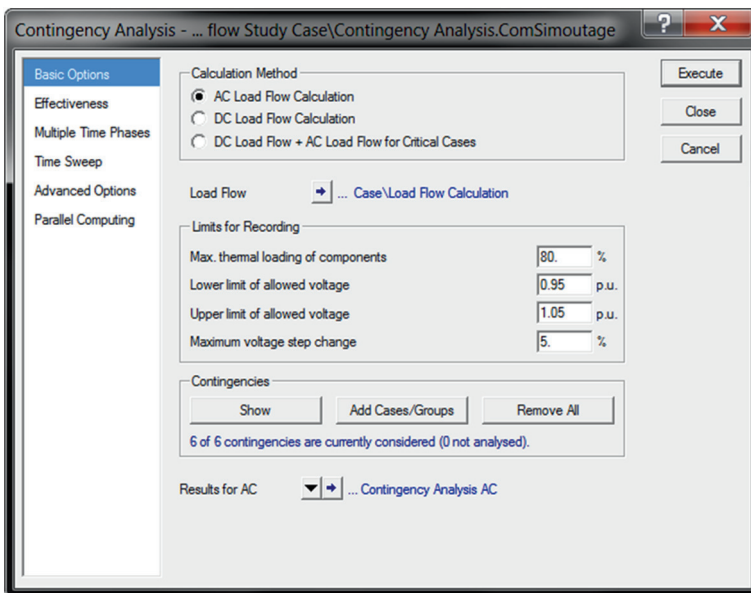


Fig. 2.17 Basic options of contingency analysis command

events: the fault location and (optionally) others (post-fault actions). Fault groups contain a set of references to fault cases. In order to use the *Add Cases/Groups* option, the fault cases and/or groups must have been previously defined in the Operational Library. If these have been defined, then the Add Cases/Groups button can be pressed. As a result, a data browser list of the available fault cases/groups pops up [3].

Depending on the calculation method selected, the reference to the corresponding result file object (*ElmRes*) is defined. The results stored in this file are filtered according to the global threshold set in *Limits for Recording* section of the Basic Data tab and also according to the individual limits defined within each component's respective dialogue.

Example 4: DC Contingency Analysis In power systems, it is possible to use approximate linear model. The method known as contingency analysis with distribution factors is based on *DC load flow*. This model is based on the assumption that the node voltages are $V_i = 1.00$ p.u. at all nodes and losing in this way the capability to track reactive power flows or node voltage. The DC load flow provides a linear relation between active power injections and phase angles of nodal voltages.

$$P_i = \sum_j P_{ij} = \sum_j \frac{V_i V_j}{x_{ij}} \sin \theta_{ij} \approx \sum_j \frac{\theta_i - \theta_j}{x_{ij}} \quad (2.16)$$

where P_i is the net active power injected to bus i , obtained in the general case as the difference between the active power injected by generating elements, S_{Gi} , and the active power absorbed by loads, S_{Li} . P_{ij} and x_{ij} are the branch active power flows and branch reactance, respectively, and θ_i is the bus phase angle.

The Eq. (2.16) can be written as follows:

$$\mathbf{P_I} = \mathbf{B}\boldsymbol{\theta} \quad (2.17)$$

$\mathbf{P_I}$ is net active power injected vector, $\boldsymbol{\theta}$ is the bus phase angle vector and \mathbf{B} is a matrix with the same structure (sparse and symmetrical) of the bus admittance matrix, but its values being computed only in terms of branch reactance. Matrix \mathbf{B} can be expressed by means of the *branch-to-node incidence matrix* \mathbf{A} and the diagonal reactance matrix \mathbf{X} as shown in (2.18):

$$\mathbf{B} = \mathbf{A}\mathbf{X}^{-1}\mathbf{A}^T \quad (2.18)$$

The relationship between the active power injected vector and the branch active power vector can be expressed by means of the branch-to-node incidence matrix \mathbf{A} (matrix reduced by removing the slack row):

$$P_i = \sum_j P_{ij} \Rightarrow \mathbf{P_I} = \mathbf{A} \mathbf{P_f} \quad (2.19)$$

The relationship between the branch active power flow and the phase angle vector can be written using the incidence matrix:

$$P_{ij} = \frac{\theta_i - \theta_j}{x_{ij}} \Rightarrow P_f = [\mathbf{X}^{-1} \mathbf{A}^T] \theta \quad (2.20)$$

Combining (2.17), (2.19), (2.20), an expression to obtain the branch active power vector from the active power injected vector is found:

$$\mathbf{P_f} = [\mathbf{X}^{-1} \mathbf{A}^T \mathbf{B}^{-1}] \mathbf{P_I} = \rho \mathbf{P_I} \quad (2.21)$$

If the system is considered to be linear, the principle of superposition can be applied, and consequently, power flows after a change in the injected powers can be obtained as follows:

$$\mathbf{P_f} = \mathbf{P_f^0} + \rho \Delta \mathbf{P_I} \quad (2.22)$$

where ρ is known as *distribution factor* and represents the flow increase in an element mn (line or transformer) after a unitary increase in the power injected in bus i Eq. (2.23).

$$\rho_{mn}^i = \frac{\Delta P_{mn}^i}{\Delta P_i} \quad (2.23)$$

The flow increase in a branch element, P_{mn} , after the outage of a generator in bus i will be obtained as follows:

$$\Delta P_{mn} = \rho_{mn}^i \Delta P_i = \rho_{mn}^i (-\Delta P_{G_i}) \quad (2.24)$$

where ΔP_{G_i} is the active power generation before the outage.

The WSCC 3-machine system (*Test System 1*) is used to illustrate a comparison between PowerFactory contingency tools and the simplified method of the distribution factors. The test system is depicted in Fig. 2.5, and quantities represented on it are results of classical steady-state calculations.

The power generation in this system in per unit is represented as follows:

$$\begin{aligned} \mathbf{P_I} &= [P_{g2} \ P_{g3} \ 0 \ P_{LA} \ P_{LB} \ 0 \ P_{LC} \ 0]^T \\ &= [1.63 \ 0.85 \ 0 \ -1.25 \ -0.9 \ 0 \ -1 \ 0]^T \end{aligned} \quad (2.25)$$

The active power flow limits ($\mathbf{P_{lim}}$) considered are shown in:

$$\begin{aligned} \mathbf{P}_{\text{lim}} &= [P_{14}^{\text{lim}} \ P_{27}^{\text{lim}} \ P_{39}^{\text{lim}} \ P_{45}^{\text{lim}} \ P_{46}^{\text{lim}} \ P_{57}^{\text{lim}} \ P_{69}^{\text{lim}} \ P_{78}^{\text{lim}} \ P_{89}^{\text{lim}}]^T \\ \mathbf{P}_{\text{lim}} &= [2 \ 2 \ 2 \ 1.5 \ 1.5 \ 1.5 \ 1.5 \ 1.5 \ 1.5]^T \end{aligned} \quad (2.26)$$

Taking into account the topology and the network parameters, matrices \mathbf{A} , \mathbf{X} , \mathbf{B} and $\boldsymbol{\rho}$ can be calculated:

$$\begin{aligned} \text{Bus} \quad \mathbf{A} &= \begin{bmatrix} 1-4 & 2-7 & 3-9 & 4-5 & 4-6 & 5-7 & 6-9 & 7-8 & 8-9 \\ 2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 6 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 7 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 1 \\ 8 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ 9 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & -1 \end{bmatrix} \\ \mathbf{X} &= \begin{bmatrix} 0.0576 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.03125 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.0586 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.085 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.092 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.1610 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.170 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.072 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1008 \end{bmatrix} \\ \mathbf{B} &= \begin{bmatrix} 32 & 0 & 0 & 0 & 0 & -32 & 0 & 0 \\ 0 & 17.065 & 0 & 0 & 0 & 0 & 0 & -17.065 \\ 0 & 0 & 39.995 & -11.765 & -10.870 & 0 & 0 & 0 \\ 0 & 0 & -11.765 & 17.975 & 0 & -6.211 & 0 & 0 \\ 0 & 0 & -10.870 & 0 & 16.751 & 0 & 0 & -5.882 \\ -32 & 0 & 0 & -6.211 & 0 & 52.100 & -13.889 & 0 \\ 0 & 0 & 0 & 0 & 0 & -13.889 & 23.810 & -9.9206 \\ 0 & -17.0648 & 0 & 0 & -5.882 & 0 & -9.9206 & 32.868 \end{bmatrix} \\ \boldsymbol{\rho} &= \begin{bmatrix} -1.0000 & -1.0000 & -1.0000 & -1.0000 & -1.0000 & -1.0000 & -1.0000 & -1.0000 \\ 1.0000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.0000 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.6387 & -0.3848 & 0 & -0.8751 & -0.1351 & -0.6387 & -0.5329 & -0.3848 \\ -0.3613 & -0.6152 & 0 & -0.1249 & -0.8649 & -0.3613 & -0.4671 & -0.6152 \\ -0.6387 & -0.3848 & 0 & 0.1249 & -0.1351 & -0.6387 & -0.5329 & -0.3848 \\ -0.3613 & -0.6152 & 0 & -0.1249 & 0.1351 & -0.3613 & -0.4671 & -0.6152 \\ 0.3613 & -0.3848 & 0 & 0.1249 & -0.1351 & 0.3613 & -0.5329 & -0.3848 \\ 0.3613 & -0.3848 & 0 & 0.1249 & -0.1351 & 0.3613 & 0.4671 & -0.3848 \end{bmatrix} \end{aligned}$$

Now, the distribution factors are used to study generator outages. If the lost generation is assumed by the slack bus,

$$\Delta P_{mn} = \rho_{mn}^i \Delta P_i$$

where $\Delta P_i = -P_{Gi}$, the active power generation before outage. In case of G2 outage, the power flows can be calculated from (2.22):

$$\begin{aligned} \mathbf{P}_f &= \mathbf{P}_f^0 + \rho \cdot [-1.63 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^T \\ \mathbf{P}_f &= [P_{14} \quad P_{27} \quad P_{39} \quad P_{45} \quad P_{46} \quad P_{57} \quad P_{69} \quad P_{78} \quad P_{89}] \\ \mathbf{P}_f &= [2.3 \quad 0 \quad 0.85 \quad 1.42 \quad 0.88 \quad 0.17 \quad -0.02 \quad 0.17 \quad -0.83]^T \end{aligned}$$

The results show that transformer T1 is overloaded (2.30 p.u.) when the generator 2 is out of service. It can be achieved the same results by means of PowerFactory. Firstly, the contingency definition function must be defined by selecting *N-1 cases* and *Create cases for generators* (see Fig. 2.17) and *ComSimoutage* function (Fig. 2.17). If the contingency analysis function (*comSimoutage*) is executed using a DC and AC load flow as calculation method, tables as shown in Fig. 2.18 can be obtained (*report contingencies function*). As can be seen, DC load flow provides enough accuracy in this case.

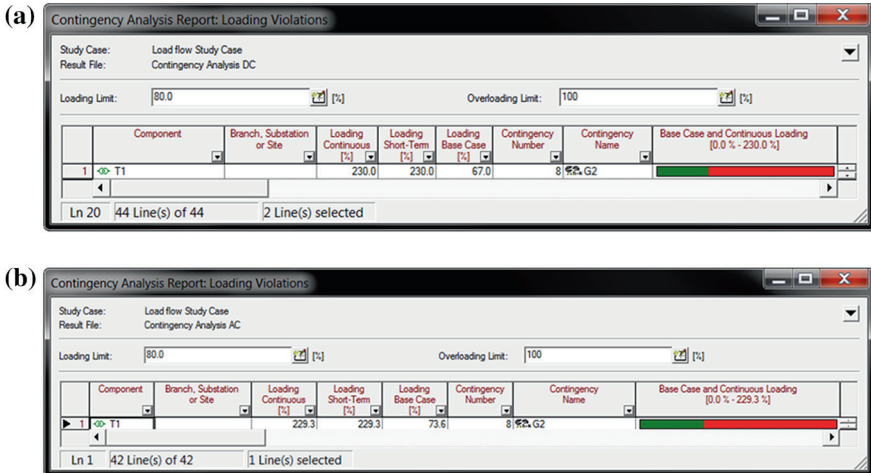


Fig. 2.18 Contingency report, generator 2 outage: **a** DC load flow and **b** AC load flow. **a** Calculation method: DC load flow and **b** calculation method: AC load flow

2.5 Using Distribution Factors to Study the Outage of a Transmission Line

The distribution factors calculated above can be used in order to get the new active power flows when a transformer/line is out of service. Post-contingency flows can be obtained replacing the branch out of service by two fictitious active power injections as depicted in Fig. 2.19. The fictitious injections must coincide with the power flow after the outage:

$$\begin{aligned} P_{ij} &= P_{ij}^0 + \rho_{ij}^i \Delta P_i + \rho_{ij}^j \Delta P_j = P_{ij}^0 + \left(\rho_{ij}^i - \rho_{ij}^j \right) P_{ij} \\ P_{ij} &= \Delta P_i = -\Delta P_j = \frac{P_{ij}^0}{1 - \rho_{ij}^i + \rho_{ij}^j} \end{aligned} \quad (2.27)$$

This approach allows us to avoid modifying matrix **B** and consequently matrix ρ . Then, the active power flow through a branch nm after ij outage is obtained as follows:

$$P_{mn} = P_{mn}^0 + \rho_{mn}^i \Delta P_i + \rho_{mn}^j \Delta P_j \quad (2.28)$$

Using (27) and (28), power flow through the nm branch is as follows:

$$P_{mn} = P_{mn}^0 + \frac{\rho_{mn}^i - \rho_{mn}^j}{1 - \rho_{ij}^i + \rho_{ij}^j} P_{ij}^0 = P_{mn}^0 + \rho_{mn}^{ij} P_{ij}^0 \quad (2.29)$$

ρ_{mn}^{ij} is known as the branch nm distribution factor when the branch ij is out of service. In the nine-bus system, suppose that lines fail, active power flows can be estimated using the distribution factors for each case. The distribution factors considering the line outages are as follows:

$$\rho_{\text{lines}} = \begin{matrix} & \begin{matrix} \rho_{mn}^{45 \text{ off}} & \rho_{mn}^{46 \text{ off}} & \rho_{mn}^{57 \text{ off}} & \rho_{mn}^{69 \text{ off}} & \rho_{mn}^{78 \text{ off}} & \rho_{mn}^{89 \text{ off}} \end{matrix} \\ \begin{matrix} 4-5 \\ 4-6 \\ 5-7 \\ 6-9 \\ 7-8 \\ 8-9 \end{matrix} & \begin{bmatrix} 7.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1.0 & 6.4 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 3.23 & 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 & 3.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 & 1.0 & 8.5 & 1.0 \\ 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 5.8 \end{bmatrix} \end{matrix}$$

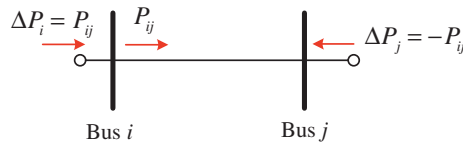


Fig. 2.19 Modelling a branch outage using fictitious injections

Table 2.4 Post-contingency power flow after line outage

	Outage line					
Power flow	4–5	4–6	5–7	6–9	7–8	8–9
4–5	N/A	0.67	1.25	−0.23	−0.38	0.62
4–6	0.67	N/A	−0.58	0.9	1.05	0.05
5–7	−1.25	−0.58	N/A	−1.48	−1.63	−0.63
6–9	−0.23	−0.9	−1.48	N/A	0.15	−0.85
7–8	0.38	1.05	1.63	0.15	N/A	1
8–9	−0.62	0.05	0.63	−0.85	−1	N/A

The diagonal elements have not to be taken into account as they represent the flow increase in a line after the outage of the same line. The analyses of the transformer outage have not been performed since they correspond to radial branch connected with generators. Therefore, the outages of those transformers are comparable to analysing the generator outage.

Then, multiplying the line distribution factors by the preoutage flow before the line fault, the post-contingency flow changes are obtained (Table 2.4). As can be seen, there are *two power flow violations*. If the contingency analysis is executed selecting the mentioned lines, similar results are obtained compared with the distribution factor methodology (Fig. 2.20).

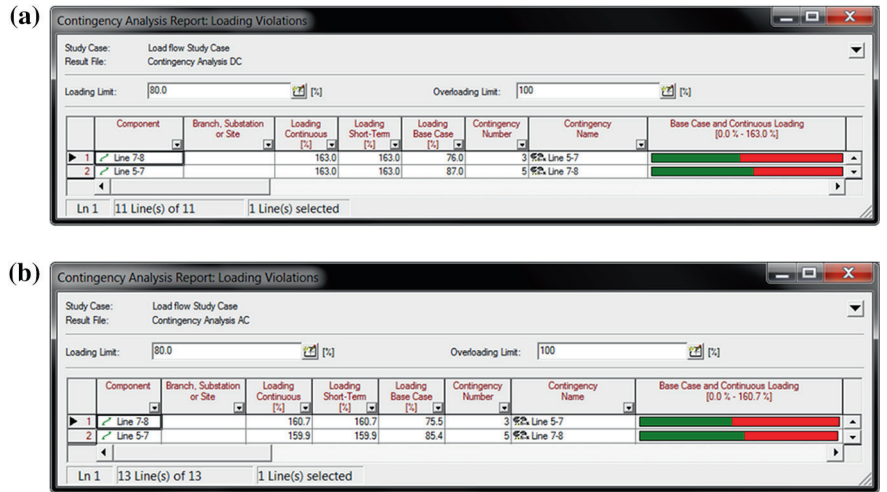


Fig. 2.20 Contingency report, lines outage: **a** DC load flow and **b** AC load flow. **a** Calculation method: DC load flow and **b** calculation method: AC load flow

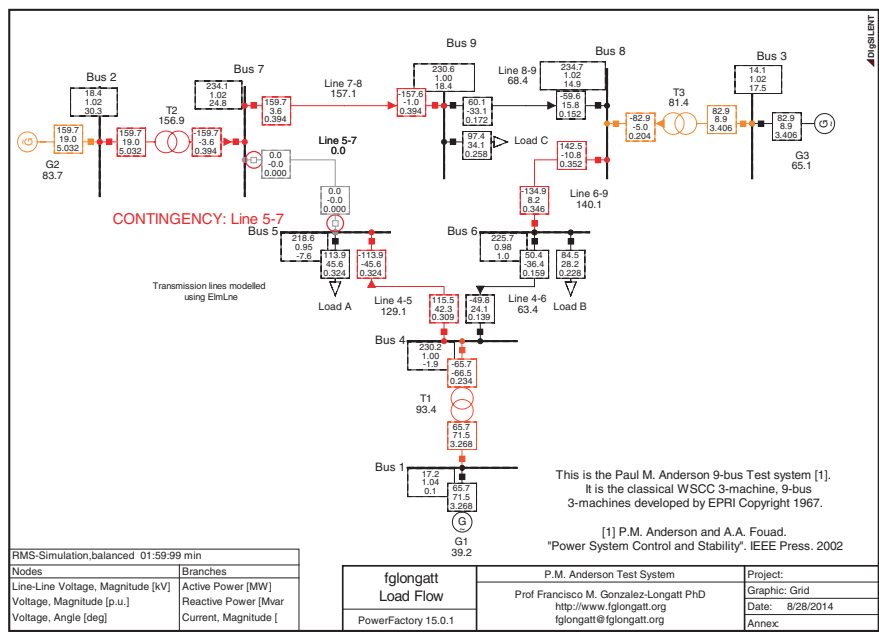


Fig. 2.21 State of test system 1 120s after sudden disconnection of transmission line 5–7

RMS simulation in PowerFactory is used to evaluate the loading of transmission line 7–8 considering an outage in line 5–7. Figure 2.21 shows all voltages and power at 120 s after the sudden disconnection of line 5–7, it can be seen that loading on transmission line 7–8 reaches 157.1 %, which is near to the calculated value (163 %) in Table 2.4, and contingency analysis results are presented in Fig. 2.21. It must be noticed that there is a small discrepancy between the loadings using different methods. Results presented at Fig. 2.21 are just a snapshot of the dynamic situation taken at 120 s; in order to get a full picture about the loading condition during the contingency, a time-domain plot is shown in Fig. 2.22.

Figure 2.3 shows oscillatory behaviour in the loading conditions immediately after the sudden disconnection of line 7–8, and the maximum loading condition (199.274) is reached 0.452 s after the contingency. Oscillations are overdamped, the amplitude of the oscillation decreases over the time, and it is easy to see an asymptotic behaviour around a loading condition of 160 % as was predicted in the previous calculations.

Figure 2.22 includes plot of the frequency in each generator and the calculation of the frequency of inertia centre. The contingency, outage of line 5–7, produces a change in the system topology which triggers a sudden power imbalance in this

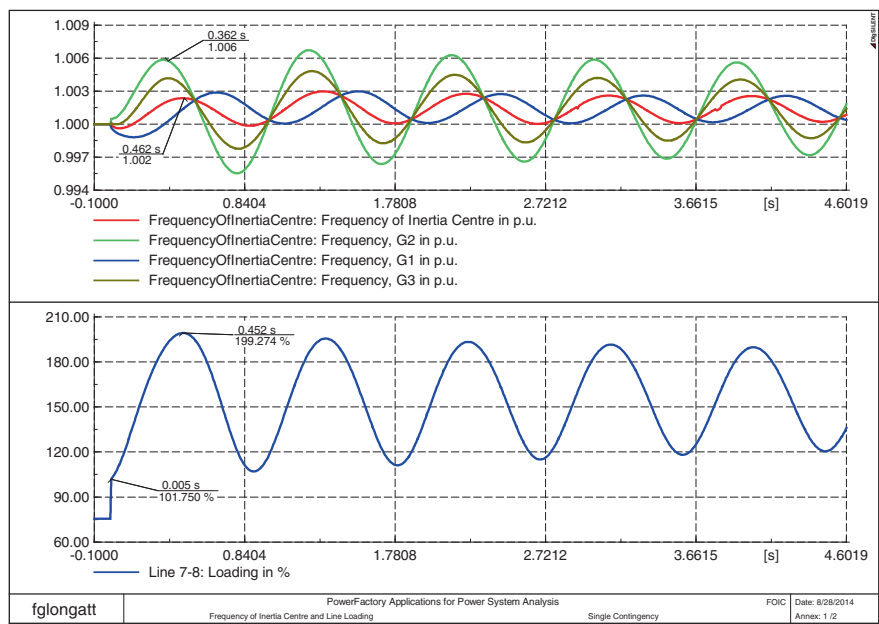


Fig. 2.22 Dynamic response of the system frequency and loading at line 7–8 after sudden disconnection of transmission line 5–7

system, causing a frequency response of the system. During a system frequency disturbance, the frequency created by each generator is related to the generator’s inertia; however, the *frequency of centre of inertia* (FOIC) can be used as indicator of the systemic system frequency response. PowerFactory is able to provide individual measurements of frequency using the PLL element (*ElmPhi_pll*); however, to obtain the FOIC, few calculations are performed during the simulation following the equations presented on [4]. *DIgSILENT Simulation Language* (DSL) has been used to calculate of FOIC during RMS simulations, this implementation is shown in Fig. 2.23, and results are shown in Fig. 2.22.

Plot of FOIC shows how the system frequency oscillates after the disconnection of line 5–7, generators modify its rotational speed to reach a new equilibrium point, and during this oscillation process, individual frequencies are changing and the FOIC as well. It can be noticed how the FOIC is slowly returning to the nominal frequency (1.00 p.u.).

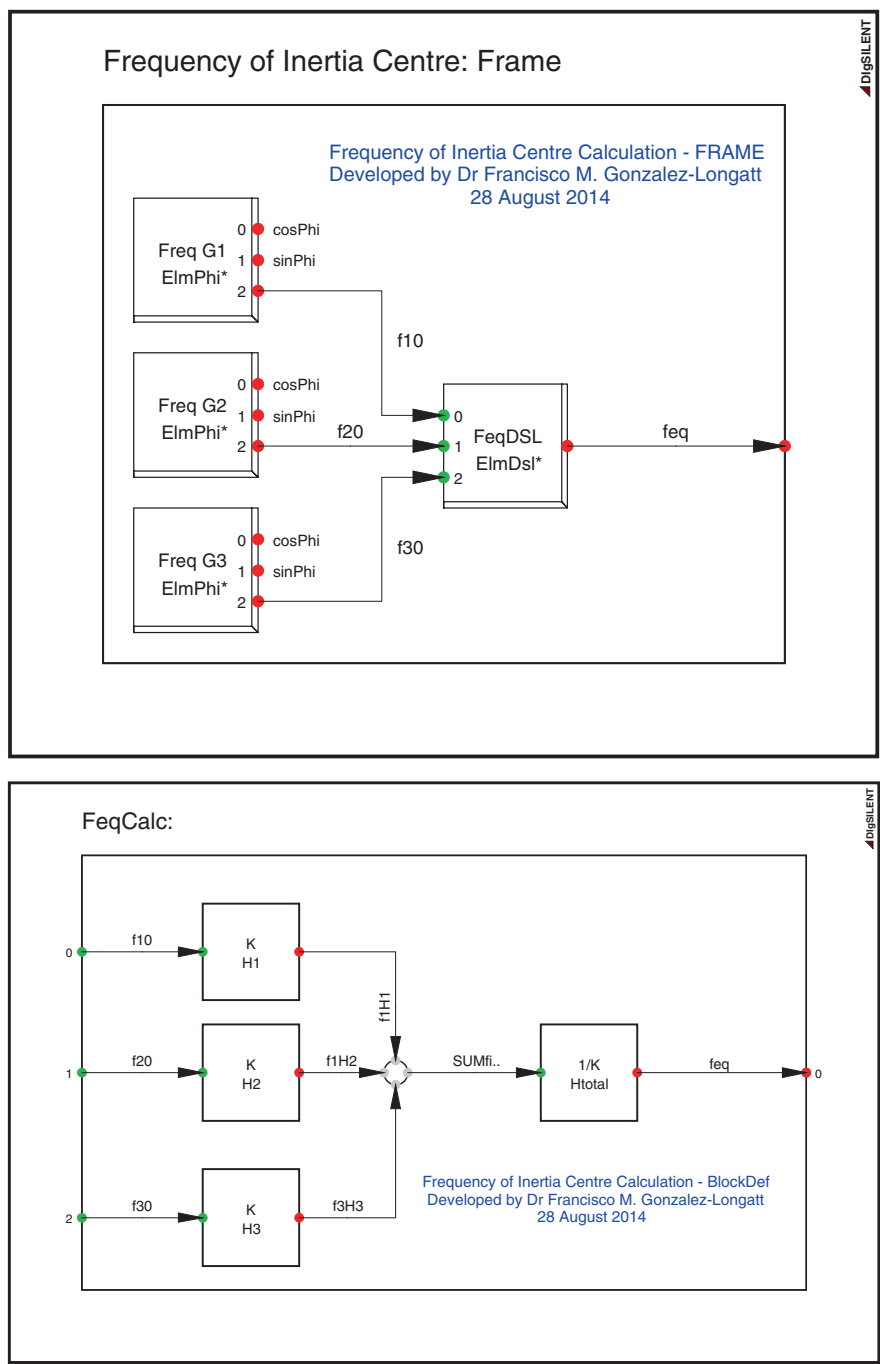


Fig. 2.23 DSL implementation to calculate frequency of inertia centre

References

1. Stagg GW, El-Abiad AH (1968) Computer methods in power system analysis. McGraw-Hill, New York
2. IEEE Recommended Practice for Power System Analysis (IEEE Brown Book) (1980) ANSI/IEEE Std 399-1980, pp 1–223
3. Power Factory User's Manual DIgSILENT PowerFactory (2008) Version 14.0, DIgSILENT GmbH, Gomaringen, Germany
4. Sauer PW, Pai MA (1998) Power system dynamics and stability. Prentice Hall, Upper Saddle River

Chapter 3

Probabilistic Power Flow Module for PowerFactory DIgSILENT

Saeed Teimourzadeh and Behnam Mohammadi-Ivatloo

Abstract Electric power system could be known as the most complex, large-scale human-built assembly. Power system engineers are deal with several challenging problems such as power system operating, planning, and protection. Different types of studies should be carried out in order to handle different problems of power system. However, like many other engineering endeavors, future operating conditions, and operating criteria could not be predicted with certainty. Load increment, generation scheduling, and power system topology are most uncertain conditions in which power system engineers are deal with them. Therefore, in order to drive a reliable system, along with deterministic studies, probabilistic studies should be carried out. Since power system includes multiple numbers of linear and nonlinear elements, computer-based software packages are sufficiently helpful. In most of the computer packages such as DIgSILENT PowerFactory and PSSE/PTI, deterministic analysis tools such as deterministic power flow (DLF) are available. However, probabilistic characteristic of these studies is not included in the software. In this chapter, a probabilistic power flow (PLF) analysis module for DIgSILENT PowerFactory is proposed. The proposed module is implemented using DPL. Two different PLF analysis methods are incorporated in the proposed module including Monte Carlo simulation and point estimate method. The user can select the desired algorithm for PLF analysis. A line-by-line explanation is provided for the written DPL code. The application of the proposed module is investigated using illustrative example incorporating IEEE 14-bus standard test system.

Electronic supplementary material The online version of this chapter (doi:10.1007/978-3-319-12958-7_3) contains supplementary material, which is available to authorized users.

S. Teimourzadeh · B. Mohammadi-Ivatloo (✉)
Faculty of Electrical and Computer Engineering, University of Tabriz,
29 Bahman Blvd, Tabriz, Iran
e-mail: bmohammadi@tabrizu.ac.ir

S. Teimourzadeh
e-mail: teimourzadeh91@ms.tabrizu.ac.ir

Keywords Deterministic power flow • Probabilistic power flow • Monte Carlo simulation • Point estimate method

3.1 Introduction

Most of the engineering problems are subject to uncertainties owing to either inherent randomness or inaccurate considered assumptions in modeling approach [1]. Considering probabilistic aspects of the engineering problems allows engineers to have a better feeling about the future outputs of the problem. Electrical power system, as the largest human-built system, is always subjected to the various types of uncertainties, i.e., daily load variation, generation outage, faults, and failures. Due to the presence of different decision makers, the level of the uncertainties will be even more in the restructured electrical power systems. Effective uncertainty modeling tools are required to control and minimize the risks associated with the design and operation of power systems [2].

Power flow studies are the most efficient tools which help power system engineers to investigate system state variables, i.e., magnitude and angle of the bus voltages. However, deterministic power flow (DLF) studies provide system variables under a certain load and generation circumstances. Load and generation uncertainties are not considered in DLF studies. As the results of power flow studies are used as essential inputs for decision making of system planners, considering the effective uncertainties is vital. Probabilistic power flow (PLF) can provide more information about the possible future conditions of the power system [3, 4]. In the probabilistic load flow studies, the input and system variables of power system are considered as random variables. Hence, the possible ranges of power flow results could be identified.

In order to handle the PLF problem, different approaches have been proposed in the literature. The proposed methods could be categorized into three main categories as follows: Monte Carlo simulation (MCS)-based methods [5–7], analytical methods [8–12], and approximate approaches [10, 13–16]. The MCS-based methods do not require approximation of the power flow problem and solve the problem via repetitive process. The main drawback of the MCS method is the requirement of large number of simulations, which needs large amount of computation time and storage. However, with the development of the efficient computation tools in the recent years, the MCS is used frequently in PLF studies [5–7]. In the analytical approaches for PLF, the power flow equations are linearized to make working with probability density functions easier. The approximation methods did not require linearization of the power flow equations. Instead, in these methods, the number of evaluation points is decreased using several techniques such as point estimate method [14]. Latin hypercube sampling (LHS) is also used for PLF studies with the limited number of simulations [7, 17, 18]. The LHS method is combined with Nataf transformation in [17] and Cholesky decomposition

in [18] for PLF analysis. For more details on the efficiencies and deficiencies of the mentioned approaches, refer to [1]. In this chapter, MCS method and point estimate method are used for PLF analysis and the detailed descriptions of them are provided in the following sections.

3.2 Monte Carlo Simulation

MCS is an iterative approach which utilizes cumulative density function (CDF) of random variables to determine the final result. In this method, firstly, certain numbers of samples should be generated corresponding to the density function of random variables. Then, fitness function should be evaluated for all generated samples. Afterward, mean value of the outputs is set as result of the iteration. Discussed process should be derived until the termination criterion satisfies. Various termination criterions are proposed in the literature; the maximum numbers of iterations and output variation-based criterions are the most popular criterions. Figure 3.1 depicts the flowchart of the MCS. More details on MCS are available in [2].

3.3 Point Estimate Method

The main drawback of the MCS-based studies is the fact that a great number of iterations are needed to reach the desired convergence [1]. However, approximate approaches require a fewer number of iterations to attain the desired convergence. Point estimate-based methods, as the approximate approaches, are the most popular and efficient approximate methods. The $2m$ and $2m + 1$ point estimate methods are widely proposed and utilized in the literature [1, 13–15]. The point estimate-based approaches offer to evaluate the fitness function by using some specially determined points. Analytical investigations prove that, despite light computational burden, desired convergence characteristic is accessed. The main steps of point estimate method are discussed as follows.

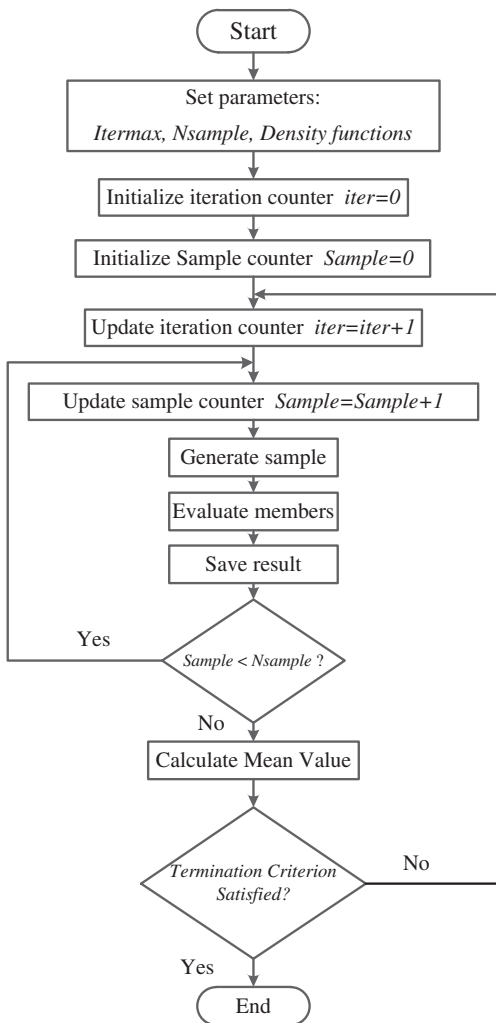
In this method, k points should be estimated for each random variable. Therefore, if there are M input variables, the fitness function should be evaluated $k \times M$ times. During each evaluation process, input vector is calculated as follows:

$$(\mu_{p1}, \mu_{p2}, \dots, P_{pl}, \dots, \mu_{pM}) \quad (3.1)$$

where μ_{pi} is the mean value of the i th input variable. Additionally, $P_{l,k}$ is calculated as follows:

$$P_{pl} = \mu_{pl} + \xi_{l,k} \sigma_{pl} \quad (3.2)$$

Fig. 3.1 Flowchart of Monte Carlo simulation



where μ_{pl} and σ_{pl} are the mean and standard deviation of the i th input random variable, respectively. The $\xi_{l,k}$ is the standard location which is related to the number of estimated points. For instance, for $2m + 1$ point estimate method, standard locations could be calculated using (3.3–3.5) [19].

$$\xi_{l,1} = \frac{\lambda_{l,3}}{2} + \sqrt{\lambda_{l,4} - \frac{3}{4}\lambda_{l,3}^2} \quad (3.3)$$

$$\xi_{l,2} = \frac{\lambda_{l,3}}{2} - \sqrt{\lambda_{l,4} - \frac{3}{4}\lambda_{l,3}^2} \quad (3.4)$$

$$\zeta_{l,3} = 0 \quad (3.5)$$

where $\lambda_{l,3}$ and $\lambda_{l,4}$ are the skewness and kurtosis of l th input variable. After estimating sample points, the fitness function should be evaluated for all estimated points. Finally, expected values of outputs are reported as final results. In this method, expected values are computed as follows:

$$E[Z] = \sum_{l=1}^m \sum_{k=1}^K \omega_{l,k} Z(l, k) \quad (3.6)$$

where Z is the vector of output random variables and $E[Z]$ is the expected value of vector Z . The $\omega_{l,k}$ are weighting coefficients and for $2m + 1$ point estimate method, these coefficients could be calculated as follows:

$$\omega_{l,k} = \frac{(-1)^{3-k}}{\zeta_{l,k}(\zeta_{l,1} - \zeta_{l,2})} \quad k = 1, 2 \quad (3.7)$$

$$\omega_{l,3} = \frac{1}{m} - \frac{1}{\lambda_{l,4} - \lambda_{l,3}^2} \quad (3.8)$$

where m is the number of input random variables. Figure 3.2 presents the flowchart of the point estimate method.

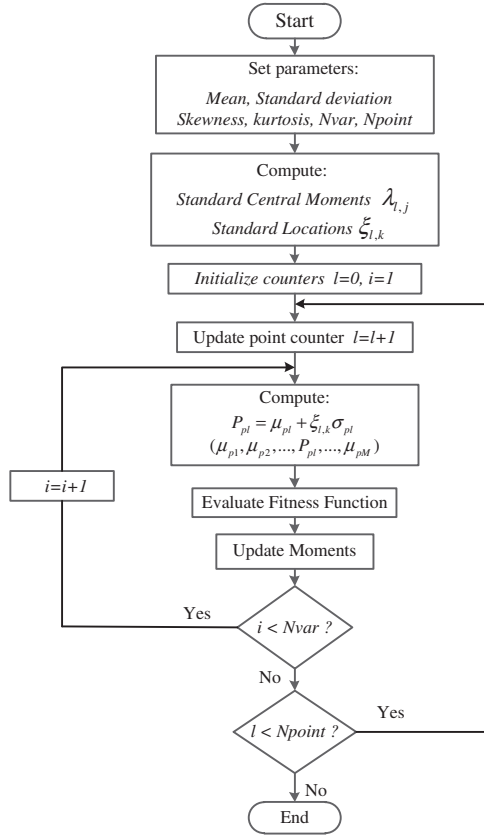
3.4 Implementation of the Probabilistic Power Flow Analysis

In this section, step-by-step implementation of the PLF analysis is presented. In the following, implementation process of both Monte Carlo-based PLF analysis and point estimate-based PLF analysis is discussed.

3.4.1 Monte Carlo Simulation

Step 1. First of all, CDF of all random variables should be prepared. In this chapter, the loads are assumed as the input random variables. The loads are assumed as normally distributed random variables with the mean value of base load and optional standard deviation.

Fig. 3.2 Flowchart of the point estimate method



- Step 2. MCS is an iterative algorithm which needs excessive number of iterations to access acceptable convergence. In this step, maximum number of iterations should be set. Additionally, MCS method utilizes samples to evaluate the fitness function. The maximum number of samples, also, is set in this step.
- Step 3. In this step, samples should be determined. In order to determine the samples in a specific iteration, firstly, two random numbers should be generated. Afterward, samples are generated as follows [2]:

$$S_i = \mu + \sigma \sqrt{-2 \ln(u_1)} \cos(2\pi u_2) \quad (3.9)$$

where S_i is the i th sample. The μ_i and σ_i are the mean value and the standard deviation of the random variable, respectively. Additionally, u_1 and u_2 are uniformly distributed random number in the range of (0, 1). In the probabilistic power calculations, the μ_i is the base load and σ_i is considered as an input data.

- Step 4. In this step, the fitness function should be evaluated by using all of the determined samples. In the PLF calculations, the fitness function is the DLF analysis.
- Step 5. *Check sampling criterion:* The sampling criterion is the maximum number of samples. The algorithm will go to Step 6 if the maximum sample number is reached; otherwise, it continues from Step 3.
- Step 6. Expected values of the outputs are computed in this step. In the case of the PLF analysis, bus voltages and transmission line loadings are considered as the outputs. For a specific iteration, the computed expected values are assigned as the results of the iteration.
- Step 7. *Check stopping criterion:* The terminating criterion is the maximum number of iterations. The algorithm will be terminated if the maximum iteration number is reached; otherwise, it continues from Step 3.

3.4.2 Point Estimate Method

- Step 1. Basic input data such as number of random variables (N_{var}), number of estimated points (N_{point}), and probabilistic characteristics of random variables, i.e., the mean, standard deviation, skewness, and kurtosis, are determined. In this chapter, $2m + 1$ method is utilized. Therefore, three points should generated using (3.1)–(3.5). In addition, system total load is considered as the random variable and the N_{var} equals to 1.
- Step 2. In this step, standard central moments ($\lambda_{l,j}$) and standard locations ($\xi_{l,k}$) should be determined. The $\lambda_{l,1}$, $\lambda_{l,2}$, $\lambda_{l,3}$, and $\lambda_{l,4}$ are the mean, standard deviation, skewness, and kurtosis of the l th random variable, respectively. Moreover, the standard locations could be calculated using (3.3–3.5).
- Step 3. The input vector of the power flow analysis should be computed in this step. The input vector could be computed using (3.1).
- Step 4. In this step, the fitness function should be evaluated. The DLF analysis is considered as the fitness function. Therefore, the DLF is derived for the input vector. The algorithm will go to Step 5, if the maximum variable number is reached; otherwise, it continues from Step 3.
- Step 5. The termination criterion is the maximum number of points. The algorithm will be terminated if the maximum point number is reached; otherwise, it continues from Step 3.

3.5 DPL Implementation of Probabilistic Power Flow Analysis

In this section, implementation of the PLF analysis by using DiGSILENT programming language (DPL) is discussed. The prepared PLF module includes a routine which has two subroutines. The main routine is an interface between users and DiGSILENT to acquire some additional data. In the following, the main routine is discussed.

The proposed main routine is provided as follows:

```

1.  int Method,Itermax,Nsample,PF_Mode;
2.  double Sd,Sk,Ku;
3.  !_____
4.  Method=1;           ! Enter 1 for Monte-Carlo Simulation
                       ! Enter 2 for Point-Estimate Method
5.  Itermax=500;        ! Max. Number of Monte-Carlo Iterations
6.  Nsample=5;          ! Number Of Samples at Each Iteration
7.  Sd=10;              ! Standard Deviation in percent
8.  Sk=0.3041;          ! Skewness
9.  Ku=2.5392;          ! Kurtosis
10. PF_Mode = 0;        ! Enter 0 Balanced Power Flow
                       ! Enter 1 Unbalanced Power Flow
11. !_____
12. if (Method=1)
13. {
    a.  MCS:Itermax=Itermax;
    b.  MCS:Nsample=Nsample;
    c.  MCS:Sd=Sd;
    d.  MCS:PF_Mode=PF_Mode;
    e.  MCS.Execute();
14. }
15. if (Method=2)
16. {
    a.  PEM:Sk=Sk;
    b.  PEM:Ku=Ku;
    c.  PEM:Sd=Sd;
    d.  PEM:PF_Mode=PF_Mode;
    e.  PEM.Execute();
17. }
18. else
19. {
20.  exit();
21. }

```

Lines 1 and 2 are used to define the variables which are utilized in the proposed DPL script. In line 4, the method of PLF analysis is identified. The term “*Method = 1*” provides MCS-based PLF analysis. On the other hand, in order to derive point estimate-based PLF calculations, the term “*Method = 2*” should be entered. In lines 5 and 6, a maximum number of iterations and number of samples for MCS-based PLF analysis are identified, respectively. Different moments of the random variable should be entered from lines 7 to 10. Additionally, DLF method is set in line 10. The term “*PF_Mode = 0*” stands for balanced DLF analysis. Alternatively, “*PF_Mode = 1*” provides unbalanced DLF calculations. The main routine includes two subroutines entitled “*MCS*” and “*PEM*.” The *MCS* and *PEM*

provide MCS-based and point estimate-based PLF analysis, respectively. The settings, which are acquired using lines 4–10, should be transferred to its corresponding subroutine. Therefore, if Monte Carlo simulation method is chosen, lines 13.a–13.e transfer relevant data to the *MCS*. Similarly, if point estimate method is chosen, lines 16.a–16.e transfer relevant data to the *PEM*. The subroutines are executed by using either the term “*MCS.Execute()*” or “*PEM.Execute()*”. In the following, details of both MCS and PEM subroutines are discussed. It is recommended to use DIgSILENT user manual in order to get familiar with the proposed DPL scripts.

3.5.1 MCS Subroutine

As previously mentioned, MCS subroutine provides MCS-based PLF calculations. The proposed DPL script is as follows:

```

1.  int error,iter,i,ii,iii,n,n_bus,n_line;
2.  object PF,0,V,02,03;
3.  double Pt,Qt,Pt2,Qt2,M,L0,u1,u2,x,a,b,sum1,sum;
4.  set S,Bus,SB,S_bus,S_line;
5.  string name;
6.  ClearOutput();
7.  !
8.  S=AllRelevant('*.ElmLod');
9.  0=S.First();
10. PF=GetCaseObject('*.ComLdf');
11. PF:iopt_net = PF_Mode;
12. S_bus=AllRelevant('*.ElmTerm');
13. S_line=AllRelevant('*.ElmLne');
14. SB=AllRelevant();
15. 02=SB.Firstmatch('ElmTerm');
16. 03=SB.Firstmatch('ElmLne');
17. !
18. n = S.Count();
19. n_bus=S_bus.Count();
20. n_line=S_line.Count();
21. Vol.Init(1,n_bus);
22. Vs.Init(Nsample,n_bus);
23. Vf.Init(Itermax,n_bus);
24. Is.Init(Nsample,n_line);
25. If.Init(Itermax,n_line);
26. Iol.Init(1,n_line);
27. per.Init(2,n);
28. LOAD.Init(2,n);
29. !
30. Pt=0;
31. Qt=0;
32. for(i=1;i<=n;i+=1)
33. {
    a. Pt+=0:plini;
    b. Qt+=0:qlini;

```

```

c. LOAD.Set(1,i,0:plini);
d. LOAD.Set(2,i,0:qlini);
e. O=S.Next();
34. }
35. O=S.First();
36. for(i=1;i<=n;i+=1)
37. {
a. per.Set(1,i,(0:plini)/Pt);
b. per.Set(2,i,(0:qlini)/Qt);
c. O=S.Next();
38. }
39. !
40. for (iter=1;iter<=Itermax;iter+=1)
41. {
42. for (i=1;i<=Nsample;i+=1)
43. {
a. u1=Random(0, 1);
b. u2=Random(0, 1);
c. Pt2=Pt*(1+(Sd/100)*sqrt(-2*ln(u1))*cos(2*pi()*u2));
d. Qt2=Qt*(1+(Sd/100)*sqrt(-2*ln(u1))*cos(2*pi()*u2));
e. O=S.First();
f. for(ii=1;ii<=n;ii+=1)
g. {
I. a=per.Get(1,ii);
II. O:plini=a*Pt2;
III. b=per.Get(2,ii);
IV. O:qlini=b*Qt2;
V. O=S.Next();
h. }
i. PF.Execute();
j. O2=SB.Firstmatch('ElmTerm');
k. for(ii=1;ii<=n_bus;ii+=1)
l. {
I. M=O2:m:u1;
II. Vs.Set(i,ii,M);
III. O2=SB.Nextmatch();
m. }
n. O3=SB.Firstmatch('ElmLne');
o. for(ii=1;ii<=n_line;ii+=1)
p. {
I. L0=O3:c:loading;
II. Is.Set(i,ii,L0);
III. O3=SB.Nextmatch();
q. }
44. }
45. for(ii=1;ii<=n_bus;ii+=1)
46. {
a. sum=0;
b. for(iii=1;iii<=Nsample;iii+=1)
c. {
I. sum1=Vs.Get(iii,ii);
II. sum+=sum1;
d. }
e. Vf.Set(iter,ii,sum/Nsample);
47. }
48. for(ii=1;ii<=n_line;ii+=1)
49. {
a. sum=0;
b. for(iii=1;iii<=Nsample;iii+=1)
c. {
I. sum1=Is.Get(iii,ii);
II. sum+=sum1;
d. }
e. If.Set(iter,ii,sum/Nsample);
50. }

```

```

51. }
52. !
53. for(ii=1;ii<=n_bus;ii+=1)
54. {
    a. sum=0;
    b. for(iii=1;iii<=Itermax;iii+=1)
    c. {
        I. sum1=Vf.Get(iii,ii);
        II. sum+=sum1;
    d. }
    e. Vol.Set(1,ii,sum/Itermax);
55. }
56. for(ii=1;ii<=n_line;ii+=1)
57. {
    a. sum=0;
    b. for(iii=1;iii<=Itermax;iii+=1)
    c. {
        I. sum1=If.Get(iii,ii);
        II. sum+=sum1;
    d. }
    e. Iol.Set(1,ii,sum/Itermax);
58. }
59. !
60. O=S.First();
61. for(ii=1;ii<=n;ii+=1)
62. {
    a. O:plini=LOAD.Get(1,ii);
    b. O:qlini=LOAD.Get(2,ii);
    c. O=S.Next();
63. }
64. !
65. ClearOutput();
66. output('Expected Value of Voltage Mag. ');
67. printf('\n');
68. for(i=1;i<=n_bus;i+=1)
69. {
70. M=Vol.Get(1,i);
71. printf('Expected Voltage Mag. of Bus %d is %f p.u.',i,M);
72. printf('\n');
73. }
74. output('Expected Value of Line Loading Percent');
75. printf('\n');
76. for(i=1;i<=n_line;i+=1)
77. {
78. LO=Iol.Get(1,i);
79. printf('Expected Loading of Line %d is %f percents',i,LO);
80. printf('\n');
81. }

```

Lines 1–6 are used to define necessary variables. In lines 8–17, essential objects and sets are acquired. For instance, the term “*AllRelevant(*.class-name)*,” returns a set with relevant objects, according to the given class name. Hence, the term “*S = AllRelevant(*.ElmLod)*,” returns a set which is related to the load class and

store it in S . The term “ $O = S.First();$ ” is used to save the first object of the set S in object O . In addition, in line 10, the term “ $PF = GetCaseObject('*.ComLdf');$ ” is utilized to define object PF . Therefore, this object contains the class “ $*.ComLdf$ ” which presents load flow calculation. Afterward, line 11 is used to set the calculation method of load flow study. The term “ PF_Mode ” is defined in line 10 of main routine. In line 18, the number of objects that are stored in the set S is acquired using the term “ $n = S.Count();$ ”. Since the set S represents the loads of the system, n presents the number of system loads. The DIGSILENT user manual could be efficient to investigate the functions and class names. Some vectors and matrices are needed to store various outputs. In DPL script, the matrices and vectors should be initialized. Lines 21–28 are defined to initialize the vital matrices and vectors. For instance, the term “ $Vs.Init(Nsample, n_bus);$ ” initializes the matrix entitled Vs , with the dimensions of $Nsample$ and n_bus . System total active and reactive loads are calculated by lines 30–34. In line 33.a, for the n th load, the amount of active load is acquired using the term “ $O.plini;$ ”. Afterward, the acquired value is added to the conventional value of Pt . Similar process is performed for reactive load using the line 33.b. Moreover, the contribution of each load in forming the total load of the system is computed using lines 37.a–37.c.

The sampling procedure and relevant calculations along with sample evaluation and expected value computations are provided using lines 41–51. Lines 43.a–43.q provide the sampling process. The sampling process is discussed in Sect. 4.1. Moreover, the averaging process to compute the expected values of the output variables is performed using lines 45–51. Lines 45–47 are used to compute the expected value of bus voltage magnitudes. Similarly, lines 48–51 provide computing the expected line loadings.

Finally, system total load is set to its initial value, and outputs are displayed using lines 62.a–62.c and 65–81, respectively.

3.5.2 PEM Subroutine

The PEM subroutine of the main routine provides the point estimate-based PLF calculations. In the following, proposed DPL script is presented.

```

1.  int error,iter,i,ii,iii,n,n_bus,n_line;
2.  object PF,O,V,O2,O3;
3.  double Pt,Qt,Pt2,Qt2,M,L0,x,a,b,sum1,sum,e1,e2,w1,w2,w3,Pinter,mm,pp,qq;
4.  set S,Bus,SB,S_bus,S_line;
5.  string name;
6.  ClearOutput();
7.  !
8.  S=AllRelevant('*.ElmLod');
9.  O=S.First();
10. PF=GetCaseObject('*.ComLdf');
11. PF:iopt_net = PF_Mode;
12. S_bus=AllRelevant('*.ElmTerm');
13. S_line=AllRelevant('*.ElmLne');
14. SB=AllRelevant();
15. O2=SB.Firstmatch('ElmTerm');
16. O3=SB.Firstmatch('ElmLne');
17. !
18. n = S.Count();
19. n_bus=S_bus.Count();
20. n_line=S_line.Count();
21. Vol.Init(1,n_bus);
22. Vs.Init(3,n_bus);
23. Is.Init(3,n_line);
24. per.Init(2,n);
25. LOAD.Init(2,n);
26. eMAT.Init(3);
27. wMAT.Init(3);
28. Pmat.Init(3);
29. Qmat.Init(3);
30. !
31. Pt=0;
32. Qt=0;
33. for(i=1;i<=n;i+=1)
34. {
    a. Pt+=0:plini;
    b. Qt+=0:qlini;
    c. LOAD.Set(1,i,0:plini);
    d. LOAD.Set(2,i,0:qlini);
    e. O=S.Next();
35. }
36. O=S.First();
37. for(i=1;i<=n;i+=1)
38. {
    a. per.Set(1,i,(0:plini)/Pt);
    b. per.Set(2,i,(0:qlini)/Qt);
    c. O=S.Next();
39. }
40. !
41. !!Point Estimation
42. e1=(Sk/2)+sqrt(Ku-(3/4)*(pow(Sk,2)));
43. e2=(Sk/2)-sqrt(Ku-(3/4)*(pow(Sk,2)));
44. eMAT.Set(1,e1);
45. eMAT.Set(2,e2);
46. eMAT.Set(3,0);
47. w1=1/(e1*(e1-e2));
48. w2=-1/(e2*(e1-e2));
49. w3=(1/1)-1/(Ku-pow(Sk,2));
50. wMAT.Set(1,w1);
51. wMAT.Set(2,w2);
52. wMAT.Set(3,w3);
53. !
54. for (i=1;i<=3;i+=1)
55. {
    a. Pinter=eMAT.Get(i);
    b. Pt2=Pt*(1+(Sd*Pinter/100));
    c. Qt2=Qt*(1+(Sd*Pinter/100));

```

```

d. Pmat.Set(i,Pt2);
e. Qmat.Set(i,Qt2);
f. O=S.First();
g. for(ii=1;ii<=n;ii+=1)
h. {
    I. a=per.Get(1,ii);
    II. O:plini=a*Pt2;
    III. b=per.Get(2,ii);
    IV. O:qlini=b*Qt2;
    V. O=S.Next();
i. }
j. PF.Execute();
k. O2=SB.Firstmatch('ElmTerm');
l. for(ii=1;ii<=n_bus;ii+=1)
m. {
    I. M=O2:m:u1;
    II. Vs.Set(i,ii,M);
    III. O2=SB.Nextmatch();
n. }
o. O3=SB.Firstmatch('ElmLne');
p. for(ii=1;ii<=n_line;ii+=1)
q. {
    I. LO=O3:c:loading;
    II. Is.Set(i,ii,LO);
    III. O3=SB.Nextmatch();
r. }
56. }
57. for(ii=1;ii<=n_bus;ii+=1)
58. {
    a. sum=0;
    b. for(iii=1;iii<=3;iii+=1)
    c. {
        I. sum1=Vs.Get(iii,ii);
        II. sum+=sum1;
        III. mm=wMAT.Get(iii);
        IV. sum+=mm*sum1;
    d. }
    e. Vf.Set(1,ii,sum/3);
    f. Vf.Set(1,ii,sum);
59. }
60. for(ii=1;ii<=n_line;ii+=1)
61. {
    a. sum=0;
    b. for(iii=1;iii<=3;iii+=1)
    c. {
        I. sum1=Is.Get(iii,ii);
        II. mm=wMAT.Get(iii);
        III. sum+=mm*sum1;
    d. }
    e. If.Set(1,ii,sum);
62. }
63. !_____
64. O=S.First();
65. for(ii=1;ii<=n;ii+=1)
66. {
    a. O:plini=LOAD.Get(1,ii);
    b. O:qlini=LOAD.Get(2,ii);
    c. O=S.Next();
67. }
68. !_____
69. ClearOutput();
70. output('Estimated points are as follows:');
71. printf('\n');
72. for(i=1;i<=3;i+=1)
73. {

```

```

a. pp=Pmat.Get(i);
b. qq=Qmat.Get(i);
c. printf('Point Number %d: Active Power= %f and Reactive Power=
%f', i, pp, qq);
74. }
75. output('Expected Value of Voltage Mag. ');
76. printf('\n');
77. for(i=1; i<=n_bus; i+=1)
78. {
a. M=Vf.Get(1, i);
b. printf('Expected Voltage Mag. of Bus %d is %f
c. p.u', i, M);
79. }
80. printf('\n');
81. output('Expected Value of Line Loading Percent');
82. printf('\n');
83. for(i=1; i<=n_line; i+=1)
84. {
a. L0=If.Get(1, i);
b. printf('Expected Loading of Line %d is %f
c. percents', i, L0);
85. }

```

Most of the utilized DPL codes of this section are discussed in MCS subroutine. Nevertheless, some different parts are discussed in the following.

In the proposed DPL script, lines 1–40 are similar to the MCS subroutine. However, during lines 41–52, based on Eqs. (3.3–3.8), standard locations and weighting coefficients are computed. The computed standard locations (lines 42–43) are stored in set *eMAT* using lines 44–46. Similarly, the computed weighting coefficients (lines 47–49) are stored in set *wMAT* using lines 50–53.

Moreover, lines 55.a–55.f are utilized to compute the estimated points. Afterward, the loads should be set to the estimated points. Discussed process is performed using lines 55.h.I–55.h.V. Then, the power flow study is performed using term “*PF.Execute()*,” and the estimated points are evaluated. The voltage magnitudes and line loadings, as the output variables, are stored in sets *Vs.set* and *Is.set* using lines 55.l–55.r.

Similar to the MCS subroutine, in lines 57–62, the expected values of output variables are computed. Additionally, system loads are restarted using line 65. Finally, the results are reported using lines 69–85.

3.6 Illustrative Examples

In this section, three illustrative examples are presented in order to verify accuracy of the proposed methods. The IEEE 14-bus standard test system is simulated in the power factory DIgSILENT and utilized to confirm the efficiency of the proposed methods. The system data of the mentioned test system are available in [20]. However, some minor modifications are performed. In the modified form of the IEEE 14-bus standard test system, the voltage magnitudes of all PV buses are set to 1.05 pu. In this section, firstly, DLF is derived on the mentioned test system. Afterward, Monte Carlo and point estimate-based PLF calculations are carried out.

For all cases, the system total load with normal density distribution is assumed as the random variable and base load is considered as the mean value. However, the standard deviation is an arbitrary parameter which is chosen by the user.

3.6.1 Case I: Deterministic Load Flow

As previously mentioned, in this example, the DLF analysis is derived. In this example, balanced AC load flow analysis is implemented. Table 3.1 reports voltage magnitude and voltage angles for different buses of the test system. According to Table 3.1, all buses are working in their acceptable limits. Additionally, loading percent of the various branches are presented in Table 3.2. According to Table 3.2, all branches are operating in less than 100 % loading level.

3.6.2 Case II: Monte Carlo Probabilistic Load Flow

In this example, Monte Carlo-based PLF analysis is investigated. In order to derive Monte Carlo-based PLF analysis, the method should be chosen in the main routine. Therefore, the term “*Method = 1;*” should be set in 4th line of the main routine. In addition, the value of standard deviation should be identified. As previously mentioned, standard deviation is an arbitrary value which should be set by the user. The mentioned parameter should be set in line 9 of the main routine. In this example, PLF analysis is carried out by considering standard deviation of 15 % for

Table 3.1 Voltage magnitude and angle (deterministic power flow results)

Bus number	Voltage magnitude (p.u)	Voltage angle (deg)
Bus 1	1.050	0.00
Bus 2	1.040	-5.13
Bus 3	1.010	-12.95
Bus 4	1.020	-10.58
Bus 5	1.024	-9.09
Bus 6	1.050	-15.00
Bus 7	1.050	-13.77
Bus 8	1.050	-13.77
Bus 9	1.020	-15.44
Bus 10	1.020	-15.65
Bus 11	1.032	-15.45
Bus 12	1.034	-15.87
Bus 13	1.029	-15.92
Bus 14	1.008	-16.70

Table 3.2 Branch loadings (deterministic power flow results)

Line number	Loading percent
Line(1)	25.36
Line(2)	60.74
Line(3)	33.54
Line(4)	58.79
Line(5)	19.91
Line(6)	45.23
Line(7)	51.22
Line(8)	73.90
Line(9)	23.56
Line(10)	21.57
Line(11)	36.95
Line(12)	33.16
Line(13)	78.14
Line(14)	7.942
Line(15)	26.37
Line(16)	39.56
Line(17)	31.57
Transformer(1)	43.53
Transformer(2)	15.60
Transformer(3)	31.04

system total load. Moreover, it should be mentioned that the MCS is done with 5 samples per iteration and 500 iterations. Similar to standard deviation, sample and iteration numbers are arbitrary parameters which could be set using lines 6 and 8 of the main routine, respectively. Voltage magnitudes of bus numbers 5 and 11 during deriving each iteration of MCS are depicted in Fig. 3.3. According to Fig. 3.3, owing to the load variation, voltage magnitudes of various buses are altering.

In the PLF analysis, the expected value and standard deviation of the outputs are considered as the final results. Figure 3.4 presents the expected value of the voltage magnitudes of bus numbers 5 and 11. In Fig. 3.4, mean value of each iteration is computed with respect to the pervious iterations. According to Fig. 3.4, it could be illustrated that the expected voltage magnitudes (EVMs) of bus numbers 5 and 11 are 1.025 and 1.032 (p.u), respectively. Moreover, final results of Monte Carlo-based PLF analysis are reported in Tables 3.3 and 3.4.

According to Table 3.3, for PQ buses, the voltage magnitudes have small variations in the vicinity of the mean values. In addition, comparison between the results of Tables 3.1 and 3.3 illustrates that EVMs of some buses are different from those calculated by DLF analysis. The EVM could be calculated by computing the mean value of the output variables. Moreover, the loading percent of branches are reported in Table 3.4. Second and fifth columns of Table 3.4 present the expected loading percent (ELP) of transmission lines. Comparison between deterministic results that are presented in Tables 3.2 and 3.4 confirms that lines loading are

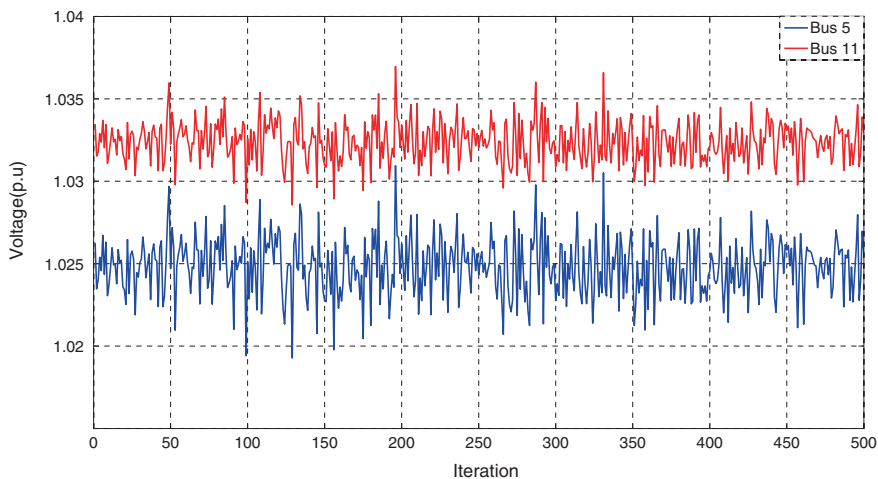


Fig. 3.3 Voltage magnitude of bus numbers 5 and 11 during each iteration

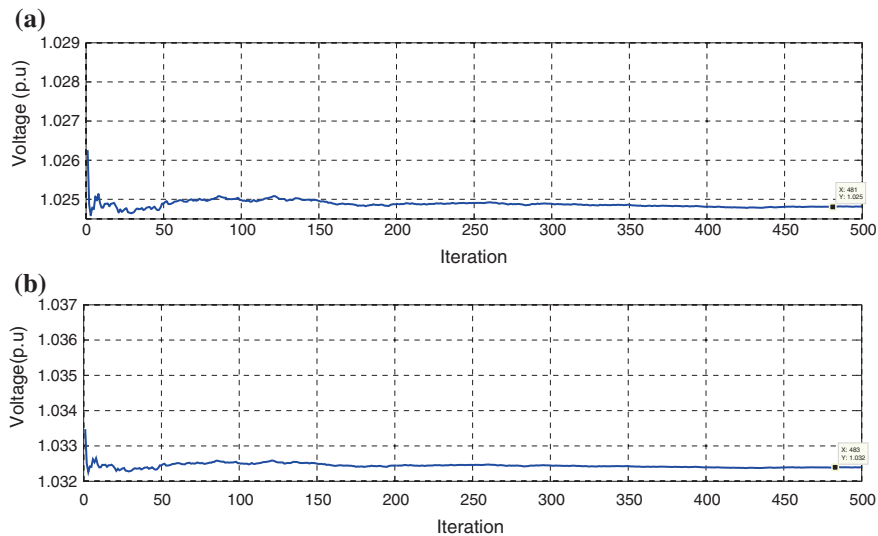


Fig. 3.4 The expected voltage magnitudes of bus numbers 5 (a) and 11 (b)

somehow similar. However, some branches have considerable standard deviations. For instance, *Line (2)* that is connected to the slack bus has considerable standard deviation.

Table 3.3 Voltage magnitude (probabilistic power flow results)

Bus number	EVM	Standard deviation
Bus 1	1.050000	6.45E-15
Bus 2	1.040000	8.22E-15
Bus 3	1.010000	6.22E-15
Bus 4	1.019891	0.001782
Bus 5	1.024819	0.001753
Bus 6	1.050000	6.45E-15
Bus 7	1.050000	6.45E-15
Bus 8	1.050000	6.45E-15
Bus 9	1.025501	0.001744
Bus 10	1.022224	0.00197
Bus 11	1.032395	0.001246
Bus 12	1.034099	0.001121
Bus 13	1.028521	0.001522
Bus 14	1.008346	0.002974

Table 3.4 Branch loadings (probabilistic power flow results)

Line number	ELP	Standard deviation
Line(1)	25.3844	2.27
Line(2)	60.73101	4.67
Line(3)	33.54901	2.10
Line(4)	58.84013	4.15
Line(5)	20.10876	1.51
Line(6)	45.22532	2.98
Line(7)	51.21575	3.79
Line(8)	73.90754	5.18
Line(9)	23.53704	1.59
Line(10)	21.59663	1.55
Line(11)	36.95489	2.63
Line(12)	33.14841	2.32
Line(13)	78.11202	5.50
Line(14)	7.945386	0.57
Line(15)	26.38333	1.91
Line(16)	39.55121	2.76
Line(17)	31.66516	2.17

3.6.3 Case III: Point Estimate Probabilistic Load Flow

In this example, point estimate-based PLF analysis is studied. In order to derive point estimate-based PLF analysis, the term “*Method = 2;*” should be set in line 4 of the main routine. Similar to the Monte Carlo-based PLF analysis, some settings

Table 3.5 The standard locations and weighting coefficients

Standard locations	Value	Weighting coefficients	Value
$\xi_{l,1}$	1.723623	$\omega_{l,1}$	0.184584
$\xi_{l,2}$	-1.419523	$\omega_{l,2}$	0.224126
$\xi_{l,3}$	0	$\omega_{l,3}$	0.591290

should be identified beforehand. The standard deviation, skewness, and kurtosis are the settings which should be identified by the user. In this study, the values of the standard deviation, skewness, and kurtosis are set on 15 %, 0.3041, and 2.5392, respectively. These values are considered based on the studies on the practical power systems. Nevertheless, any other values could be adopted. In order to derive point estimate-based PLF analysis, firstly, standard locations and weighting coefficients should be computed using Eqs. (3.3–3.8). Table 3.5 presents the standard locations and weighting coefficients for this example.

The estimated points could be computed by using Eq. (3.2) with respect to Table 3.5. In this example, system total load is 259 MW and 81.4 MVAR. The estimated points are presented in Table 3.6. According to Table 3.6, three points are estimated, and for each point, the DLF analysis should be carried out and the results should be saved. Afterward, the results are used to compute the expected value and the standard deviation of outputs. According to Table 3.5, the standard location of the first, second, and third points is positive, negative, and zero, respectively. Hence, the first point should be greater than system's base load and the second one should be lower than system's base load. In addition, the third point should be equal to the base load of system. Table 3.6 confirms this claim.

Table 3.7 reports voltage magnitudes of the system buses using the estimated points. According to Table 3.7, the results of point #3 are similar to Table 3.1. However, owing to load increment that is estimated by point #1, the voltage magnitudes corresponding to the point #1 are lower in comparison with those of point #3. On the other hand, the voltage magnitudes of the second point are higher than those of the third point. The results of Table 3.7 could be verified with respect to the load increment and decrement which are estimated by the first and second points, respectively. Moreover, expected value and standard deviation of the voltage magnitudes are reported in Table 3.8. The expected values and standard deviations are computed using weighting coefficients of Table 3.5.

Table 3.6 The estimated points

Point number	Active power (MW)	Reactive power (MVAR)
1	325.96	102.44
2	203.85	64.06
3	259.00	81.40

Table 3.7 Voltage magnitude related to the estimated points

Bus number	Point #1	Point #2	Point #3
Bus 1	1.05000	1.05000	1.05000
Bus 2	1.04000	1.04000	1.04000
Bus 3	1.01000	1.01000	1.01000
Bus 4	1.01281	1.02528	1.02000
Bus 5	1.01779	1.03009	1.02496
Bus 6	1.05000	1.05000	1.05000
Bus 7	1.05000	1.05000	1.05000
Bus 8	1.05000	1.05000	1.05000
Bus 9	1.01874	1.030869	1.025527
Bus 10	1.01461	1.028299	1.022242
Bus 11	1.02759	1.03624	1.032402
Bus 12	1.029795	1.037566	1.034095
Bus 13	1.022671	1.033227	1.028520
Bus 14	0.996875	1.017523	1.008363

Table 3.8 Expected voltage magnitude (probabilistic power flow results)

Bus number	EVM	Standard deviation
Bus 1	1.05000	2.11E-08
Bus 2	1.04000	0
Bus 3	1.01000	0
Bus 4	1.01986	0.003971
Bus 5	1.02479	0.003919
Bus 6	1.05000	2.11E-08
Bus 7	1.05000	2.11E-08
Bus 8	1.05000	2.11E-08
Bus 9	1.02547	0.00386
Bus 10	1.02219	0.00435
Bus 11	1.03237	0.00275
Bus 12	1.03407	0.00247
Bus 13	1.02849	0.00335
Bus 14	1.00829	0.00657

According to the results of Table 3.8, there are some differences between the results of Tables 3.8 and 3.3, which present the results of MCS. The MCS is an iterative solution and takes large number of iterations to converge the result. In *Case II*, MCS is derived only for 500 iterations. However, the point estimate method finds the most promising solution without any iterative process. Finally, Table 3.9 reports the expected branch loadings using point estimate-based PLF analysis.

Table 3.9 Expected branch loadings (probabilistic power flow results)

Line number	ELP	Standard deviation
Line(1)	25.4224	5.02
Line(2)	60.8107	10.29
Line(3)	33.5847	4.64
Line(4)	58.9084	9.17
Line(5)	20.1253	3.40
Line(6)	45.2764	6.58
Line(7)	51.2797	8.36
Line(8)	73.9954	11.43
Line(9)	23.5647	5.02
Line(10)	23.5647	3.50
Line(11)	21.6225	3.44
Line(12)	36.9993	5.80
Line(13)	33.1881	5.11
Line(14)	78.2059	12.13
Line(15)	7.95503	1.26
Line(16)	26.4155	4.23
Line(17)	39.5985	6.10

3.7 Discussion

This chapter presents a PLF analysis for power factory DIgSILENT. Here are some important features that should be considered about this chapter:

- This chapter deals with the PLF analysis.
- In this chapter, Monte Carlo-based and point estimate-based power flow analyses are discussed.
- A DPL-based module is designed for DIgSILENT PowerFactory.
- The provided module is a flexible module, and for different cases and purposes, all setting could be changed by users.
- The designed module is capable of handling both Monte Carlo-based and point estimate-based PLF analysis.
- The Monte Carlo-based PLF analysis requires just standard deviation of total load as input data. Nevertheless, owing to its iterative feature, it requires large number of iterations to converge.
- The point estimate-based probabilistic power is the most promising analysis and has low computational burden. However, it requires additional information of the random variables, i.e., skewness and kurtosis.
- In the provided module, it is proposed to utilize point estimate-based analysis when all information about random variable is available. On the other hand, for those with little data, the Monte Carlo-based analysis is the best solution.

3.8 Conclusion

In this chapter, a new module is developed for PLF analysis using DIgSILENT PowerFactory. The DPL is used for modeling the effect of load uncertainties in power flow analysis. Two methods based on the Monte Carlo simulation and point estimate method are incorporated in the developed module. The user can select the used procedure for PLF calculation and perform the analysis with different conditions. The written program in DPL is described in detail, and its implementation is investigated using the illustrative IEEE 14-bus standard test system.

References

1. Morales JM, Perez-Ruiz J (2007) Point estimate schemes to solve the probabilistic power flow. *Power Syst IEEE Trans* 22:1594–1601
2. Anders GJ (1989) Probability concepts in electric power systems
3. Borkowska B (1974) Probabilistic load flow. *Power Apparatus Syst IEEE Trans* 93 (3):752–759
4. Sexauer JM, Mohagheghi S (2013) Voltage quality assessment in a distribution system with distributed generation—a probabilistic load flow approach. *Power Deliv IEEE Trans* 28:1652–1662
5. Rubinstein RY, Kroese DP (2011) *Simulation and the monte carlo method*, vol 707. Wiley, New York (Wiley.com)
6. Jorgensen P, Christensen JS, Tande JO (1998) Probabilistic load flow calculation using Monte Carlo techniques for distribution network with wind turbines. In: *Proceedings of the 8th international conference on harmonics and quality of power*, vol 2, pp 1146–1151
7. Hajian M, Rosehart WD, Zareipour H (2013) Probabilistic power flow by Monte Carlo simulation with latin supercube sampling. *Power Syst IEEE Trans* 28:1550–1559
8. Allan R, Leite da Silva A, Burchett R (1981) Evaluation methods and accuracy in probabilistic load flow solutions. *Power Apparatus Syst IEEE Trans* 2539–2546
9. Meliopoulos AS, Cokkinides GJ, Chao XY (1990) A new probabilistic power flow analysis method. *Power Syst IEEE Trans* 5:182–190
10. Zhang P, Lee ST (2004) Probabilistic load flow computation using the method of combined cumulants and Gram-Charlier expansion. *Power Syst IEEE Trans* 19:676–682
11. Williams T, Crawford C (2013) Probabilistic load flow modeling comparing maximum entropy and Gram-Charlier probability density function reconstructions. *Power Syst IEEE Trans* 28:272–280
12. Yan C, Jinyu W, Shijie C (2013) Probabilistic load flow method based on Nataf transformation and Latin hypercube sampling. *Sustain Energy IEEE Trans* 4:294–301
13. Su CL (2005) A new probabilistic load flow method. In: *Power engineering society general meeting*, 2005. IEEE, pp 389–394
14. Su C-L (2005) Probabilistic load-flow computation using point estimate method. *Power Syst IEEE Trans* 20:1843–1851
15. Verbic G, Cañizares CA (2006) Probabilistic optimal power flow in electricity markets based on a two-point estimate method. *Power Syst IEEE Trans* 21:1883–1893
16. Soleimanpour N, Mohammadi M (2013) Probabilistic load flow by using nonparametric density estimators. *Power Syst IEEE Trans* 28:3747–3755
17. Chen Y, Wen J, Cheng S (2013) Probabilistic load flow method based on Nataf transformation and Latin hypercube sampling. *Sustain Energy IEEE Trans* 4:294–301

18. Yu H, Chung CY, Wong KP, Lee HW, Zhang JH (2009) Probabilistic load flow evaluation with hybrid latin hypercube sampling and Cholesky decomposition. *Power Syst IEEE Trans* 24:661–667
19. Hong H (1998) An efficient point estimate method for probabilistic analysis. *Reliability Eng Syst Saf* 59:261–267
20. <https://www.ee.washington.edu/research/pstca/>

Chapter 4

Unbalanced Power Flow Analysis in Distribution Systems Using TRX Matrix: Implementation Using DIgSILENT Programming Language

**Paulo M. De Oliveira-De Jesus, Andres A. Rojas Q
and Francisco M. Gonzalez-Longatt**

Abstract In this chapter, a generalized model to solve multigrounded distribution networks under DIgSILENT platform taking into account current flows and voltages across neutral and ground paths is presented. The fundamental idea developed in this work is solving the 5-wire power flow problem using a unique matrix called TRX that includes all feeder characteristics related to phase, neutral, and equivalent ground paths. Proposed power flow routines are run from MATLAB engine using DIgSILENT databases. The method is suitable to be applied on unbalanced aerial and underground distribution systems with distributed generation. Proposed methodology has been successfully applied in an illustrative test system.

Keywords Backward/forward sweep • Load flow • Power flow • Distribution system analysis • DIgSILENT Programming Language • MATLAB

Electronic supplementary material The online version of this chapter (doi:10.1007/978-3-319-12958-7_4) contains supplementary material, which is available to authorized users.

P.M. De Oliveira-De Jesus (✉) • A.A. Rojas Q
Conversion and Energy Delivery Department, Simon Bolivar University,
AP. 89000, Caracas, Venezuela
e-mail: pdeoliveira@usb.ve

A.A. Rojas Q
e-mail: arojasq@usb.ve

F.M. Gonzalez-Longatt
School of Electronic, Electrical and Systems Engineering,
Loughborough University, Loughborough LE11 3TU, UK
e-mail: longatt@ieee.org

4.1 Introduction

DIgSILENT PowerFactory is able to solve several configurations of unbalanced distribution systems: three-phase, three-phase–four-wire, two-phase, bi-phase, single-wire earth return [1]. All network models are based upon three-phase models where the neutral wire and the ground effect is usually merged into phase conductor using *Kron's reduction* [2] and *Carson equations* [3]. In this chapter, a new application to solve the detailed 5-wire multigrounded power flow model taking into account all current flows and voltages across phase, neutral, and ground paths is presented. The 5-wire power flow application is not supported in current versions of DIgSILENT PowerFactory.

Several methods have been reported in the literature to solve multigrounded unbalanced distribution system power flow. We can identify two major solving approaches: firstly Newton–Raphson-based methods [4–7] and methods based upon extensions of the two-step backward–forward (BW/FW) sweep method [8, 9].

The computing engine used in this proposal is an extension of TRX formulation for the basic power flow approach (positive sequence and three phases) [10]. In this formulation, the iterative process is merged into one unique evaluation formula. Only one sweep or iteration is required to get all currents and voltages. The extended TRX matrix constitutes a complete database by including information of network topology structure as well as branch impedances (phase, neutral, and ground) of each distribution feeder. This representation is suitable to be efficiently stored in DIgSILENT database in order to be exchanged under common information model [11]. The proposal is suitable to be applied on aerial and underground distribution systems, radially, weakly, or strongly meshed, operated with distributed generation. The methodology has been applied using a representative test system.

This document is organized as follows: Sect. 4.2 describes the 5-wire multigrounded electric system model. Section 4.4 presents the methodology. Case study is discussed in Sect. 4.5. Conclusions are drawn in Sect. 4.6. Description of the distribution line exact model is included in Appendix A. The DPL code used to build the power flow database is presented in Appendix B. In Appendix C, DPL script for displaying final results is presented. Finally, I/O file structure is provided in Appendix D.

4.2 The 5-Wire Multigrounded Distribution System Model

The calculation of 5-wire power flow is based upon a detailed power system model for loads, generators, and networks.

4.2.1 The Network Model

In this contribution, the detailed network model is represented by a unique matrix called **TRX** comprising the following items:

- Self- and mutual phase line impedances (without Carson effect)
- Self- and mutual neutral line impedances
- Self- and mutual ground line impedances
- Earthing pole impedance
- Topology representation (system sending and receiving nodes)
- Grounding reduction factors representation

All of these components could be acquired or calculated from DigSILENT's database. Self- and mutual impedances could be calculated from power line configurations, i.e., distance spacings on overhead and underground three-phase, two-phase, and single-phase lines. Earth pole impedance could be calculated depending on system grounding geometry and local resistivity. Topology representation could be easily built from sending and receiving bus list using the well-known upper triangular matrix [12]. Finally, grounding reduction factor calculation is required at each node in order to assess current flows through ground and neutral paths.

The 5-wire multigrounded electric system model with m buses can be represented by a complex matrix **TRX**, where all entries are real numbers, using the following structure.

$$\mathbf{TRX} = \mathbf{T}^T \cdot \mathbf{Z} \cdot \mathbf{T} \cdot \mathbf{M} \quad (4.1)$$

Branch data are specified through $5m \times 5m$ rectangular impedance matrix **Z**:

$$\mathbf{Z} = \text{Diag} \left[\hat{\mathbf{Z}}^{01} \quad \dots \quad \hat{\mathbf{Z}}^{ij} \quad \dots \quad \hat{\mathbf{Z}}^{(m-1)m} \right] \quad (4.2)$$

where $\hat{\mathbf{Z}}^{ij}$ is per unit primitive series impedance matrix 5×5 corresponding to i - j line section, three phases, one neutral, and equivalent ground conductor. Details of calculation of all entries of these primitive matrices $\hat{\mathbf{Z}}^{ij}$ are included in the Appendix A.

The $5m \times 5m$ **T** matrix is the upper triangular topology matrix and built according Kirchhoff current laws (KCL) [12].

M is a $5m \times 5m$ matrix containing the reduction factors. The reduction factors are defined as the ratio between currents flowing through grounding systems at node i , I_{G_i} and the unbalanced current at node j .

The concept of a **TRX** matrix gathers into a unique object of all feeder characteristics related to network topology and structure of phase, neutral, and equivalent ground paths. It could be easily built from database exchange protocol, *wires*, and *topology* classes under IEC61968 [11].

4.2.2 Power Injection Model

Phase power injections at each load buses are represented as follows:

$$\mathbf{S} = \begin{bmatrix} \bar{S}_{p1} \\ \vdots \\ \bar{S}_{pj} \\ \vdots \\ \bar{S}_{pm} \end{bmatrix} = \begin{bmatrix} P_{pl} + jQ_{pl} \\ \vdots \\ P_{pj} + jQ_{pj} \\ \vdots \\ P_{pm} + jQ_{pm} \end{bmatrix} \quad (4.3)$$

where active and reactive powers are given as total power injected at node j , and related to the per phase generated and demanded powers as follows:

$$P_{pj} = P_{Gpj} - P_{Dpj} \quad p = a, b, c \quad j = 1, \dots, m \quad (4.4)$$

$$Q_{pj} = Q_{Gpj} - Q_{Dpj} \quad p = a, b, c \quad j = 1, \dots, m \quad (4.5)$$

Note that generated powers at load nodes turn the network from passive to active by including distributed generation injections.

4.3 Implementation

There exist different possibilities to integrate DIgSILENT with other simulators. In this case, a direct interface to link DIgSILENT database with MATLAB calculation engine mode is required [13]. This proposal is developed in order to extend the capabilities of existing power flow solution coded in MATLAB with the advantages of the standardized DIgSILENT database. Figure 4.1 shows a diagram with the general procedure used in this paper to communicate both programs.

The data exchange process is carried out using two buffer files:

1. *casedata.txt* file for connecting DIgSILENT database to MATLAB workspace and
2. *casesolution.txt* file for transferring the power flow solution from MATLAB to DIgSILENT

The communication functions are written using DIgSILENT programming language (DPL).

The procedure is applied in six steps:

1. Load DIgSILENT current project database `5wire.pfd`
2. Execute the DPL script called **DPL5wire** to extract required data from DIgSILENT current project database (details in Appendix B).
3. The power flow database is stored in `casedata.txt` file.

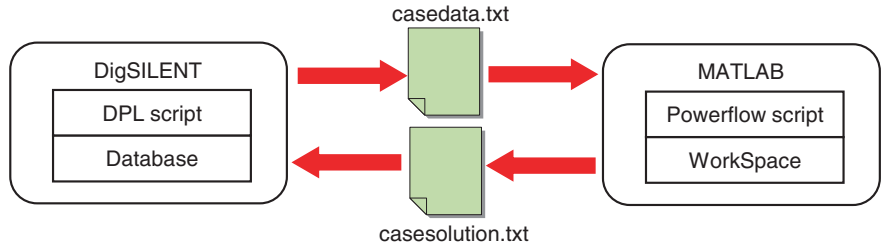


Fig. 4.1 Data exchange between DIgSILENT and MATLAB

4. Execute the program `MATLAB5wire.m` under MATLAB platform: The power flow is run for current case study. The power flow is coded in MATLAB (details in Appendix C):
5. The power flow solution is stored in `casesolution.txt` file.
6. Execute the DPL script called **DPL_MATLAB_DATA** in order to load the solution file in DIgSILENT command window display (details in Appendix D).

In Fig. 4.2, the general flowchart procedure is shown.

4.3.1 General Considerations

The DPL script is written taking into account the following general aspects:

- The following sets must be defined *set*, objects (*object*), integer variables (*int*), double-precision variables (*double*), and characters (*string*). The list of *set*, *object* *double*, *int*, and *string* is presented in Appendix B.
- By using the *fopen* command, the `casedata.txt` file is created, *txt*.
- Network elements are extracted from the database. A database query is performed using *AllRelevant* command, specifying the corresponding grid element to be extracted *ElmNet*.

The grid elements are identified as follows:

- *ElmLne*: Transmission lines
 - *ElmTerm*: System busbars
 - *ElmLod*: Loads
- Each element is an object with different attributes suitable to be extracted. For instance, for the element *ElmLne*, the object *pLne* is created, and then using the

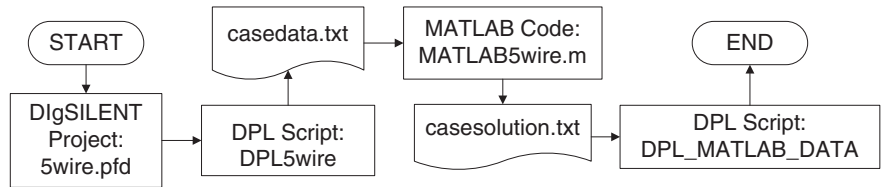


Fig. 4.2 General project implementation flowchart

while statement different attributes as receiving node tag i , sending node tag j , feeder length, nominal voltage, etc. Other important attributes are extracted using the object *TypTow*: pole geometric configurations per phase and neutrals and ground soil resistivity characteristics and the object *TypCon* used to extract conductor characteristics as geometric mean radius (GMR), resistances, and conductor diameter. Using *ElmXnet* object, the slack bus is extracted; *usetp* specifies slack bus modulus, *phiini* is slack bus angle, and *bustp* indicates busbar type: PV, PQ, and Slack.

- For element *ElmLod*, following attributes are extracted: active *plinir*, *plinis*, *plinit*, and reactive power demands *qlinir*, *qlinis*, and *qlinit* for phases r , s , and t .

4.3.2 DPL Code: Step-by-Step Description

The DPL code used in this application to build the *casedata.txt* file is included in Appendix B. The DPL code contains five sections:

- Section INTRODUCTION
- Section LINES
- Section CONDUCTORS
- Section TOWERS
- Section LOADS

Detailed description of the DPL code is provided.

4.3.2.1 Section INTRODUCTION

Section INTRODUCTION is coded as follows:

```
1. Fc = GetCaseObject('ComLdf');
2. Fc.Execute();
3. fopen('C:\casedata.txt','w',0);
4. Grids = AllRelevant('*ElmNet');
```

The function **GetCaseObject** is used to extract active project current functions; in this case, **ComLdf** is applied in object F_c . Power flow algorithm is calculated using **Execute** command. In line 3, using *fopen* command, the file *casedata.txt* is created with *w* permission. In line 4, the function **AllRelevant** is used to extract all elements that belong to grid current project by using the class *ElmNet* and save in set *Grids*.

4.3.2.2 Section LINES

Section LINES is coded as follows:

```
1. Lines = pGrd.GetContents('*ElmLne*',1);
2. pLne = Lines.First(); !Call to objects Lines
```

```

3. while (pLne) {
4.  pTyp_line = pLne:typ_id;
5.  busi = pLne.GetNode(0,0);
6.  busj = pLne.GetNode(1,0);
7.  kVnom = busi.Unom();
8.  lname = pLne:loc_name;
9.  busi_name = busi:loc_name;
10. busj_name = busj:loc_name;
11. long = pLne:dline; !Length
12. fprintf(0,'%s',busi_name);
13. fprintf(0,'%s',busj_name);
14. pLne = Lines.Next();
15. }
16. fprintf(0,'%f',long);
17. fprintf(0,'%g',kVnom);

```

In line 1, all entries in element **ElmLne** are acquired by applying the command **GetContents**. From line 3 to line 15, a *while* loop is applied in order to extract all line terminals by using the command **GetNode**, all voltage levels by using the command **Unom**, all line name terminals by using the parameter *loc_name*, and its length by applying the parameter *dline*. The parameter *loc_name* is related to strings *busi_name* y *busj_name*, and parameter *dline* is related to double-precision variable *long*. From line 16 to line 17, extracted entries (line lengths and voltage levels) are written on *casedata.txt* file. Voltage level is specified with parameter *kVnom*.

4.3.2.3 Section CONDUCTORS

Section CONDUCTORS is coded as follows:

```

1. Lib_Cond = GetLocalLib('TypCon');
2. Cond = Lib_Cond.GetContents();
3. pCond = Cond.First(); !Call to objects belong to Cond
4. while (pCond) {
5.  Cname = pCond:loc_name;
6.  cond_gmr = pCond.GetVal(gmr,'erpha');
7.  cond_res = pCond.GetVal(rdc,'rpha');
8.  if (.not.cond_gmr) {
9.   fprintf(0,'%f',gmr);
10.  fprintf(0,'%f',rdc);
11. };
12. pCond = Cond.Next();
13. };

```

In lines 1–2, command **GetLocalLib** is used to get all entries of Conductor Type library **TypCon** by applying the general command **GetContents**. In line 3, *pCond*

object is created. From lines 4 to 13, a *while* loop is applied to get all conductor values *pCond* through general command *GetVal*. This command has two input parameters: *erpha* and *rpha*. The parameter *erpha* is related to GMR, and *rpha* is related to direct current resistance. In lines 9–10, GMR and DC resistance of all conductors are acquired and written in *casedata.txt* file.

4.3.2.4 Section TOWERS

Section TOWERS is coded as follows:

```

1. Towers_set = AllRelevant('* TypTow');
2. TowersG = Towers_set.First(); !Call to objects belong
   to Tower
3. coord_xy_p = TowersG.GetSize('xy_c', fp, cp);
4. coord_xy_e = TowersG.GetSize('xy_e', fn, cn);
5. if (.not.coord_xy_p)
6. c = 0; s = "";
7. while (c < cp) {
8. coord_xy_p = TowersG.GetVal(x, 'xy_c', r, c);
9. fprintf(0, '%g', x);
10. if (.not.coord_xy_p) c += 1; }
11. s1 = ""; c = 0;
12. while (c < cn) {
13. coord_xy_e = TowersG.GetVal(y, 'xy_e', r, c);
14. fprintf(0, '%g', Y);
15. if (.not.coord_xy_e) c += 1;
16. }
17. for(TowersG = Towers_set.First(); TowersG; Tow-
   ersG = Towers_set.Next()) {
18. rground = TowersG:rearth;
19. fprintf(0, '%f', rground);
20. }
```

In line 1, command **AllRelevant** is used to get all entries of Pole Type library **TypTow** by applying the general command **GetContents**. In line 2, elements of set *Towers_set* are called. In lines 3–4, vector size of **xy_c** and **xy_e** is obtained using **GetSize** command. From lines 7 to 10, a *while* loop is executed in order to get all int variables *y*. Double-precision phase conductor coordinates *coord_xy_p* are acquired using **GetVal** command. Between lines 12 and 14, the same procedure is applied to get neutral coordinates. A *for* parameter is executed to extract soil resistivity **rearth** parameter, and its value is stored in a double-precision variable called *rground*.

4.3.2.5 Section LOADS

Section LOADS is coded as follows:

```

1. Loads = pGrd.GetContents('* .ElmLod', 1);
2. pLds = Loads.First();
3. while (pLds){
4. Load1 = pLds.GetParent();
5. busi_name = pLds:loc_name;
6. lap = pLds:plnir;
7. lbp = pLds:plinis;
8. lcp = pLds:plinit;
9. laq = pLds:qlnir;
10. lbq = pLds:qlinis;
11. lcq = pLds:qlinit;
12. fprintf(0, '%f\n%f\n%f\n%f\n%f\n%f\n', lap, lbp, lcp,
    laq, lbq, lcq);
13. pLds = Loads.Next();
14. }

```

In line 1, all entries in element **ElmLod** are acquired by applying the command **GetContents**. In line 2, elements of set *loads* are called. From lines 3 to 15, a *while* loop is executed in order to get **plnir**, **plinis**, **plinit**, **qlnir**, **qlinis**, and **qlinit** parameters. Their values are stored as double-precision variables *lap*, *lbp*, *lcp*, *laq*, *lbq*, and *lcq*. Variables *lap*, *lbp*, and *lcp* represent active powers at phases *r*, *s*, and *t*, respectively. Variables *laq*, *lbq*, and *lcq* represent reactive powers at phases *a*, *b*, and *c*, respectively.

4.4 The 5-Wire Power Flow Algorithm

This section presents the fundamental steps required to perform the 5-wire power flow study. The general flow chart of the method coded in MATLAB is depicted in Fig. 4.3

Step 1: **Data Setup:** The input files must be processed in order to apply the algorithm.

1. Bus renumbering and **T** matrix configuration (according to [12])
2. Defining **Z**, **M**, and **TRX** matrices

Step 2: **Method Initialization:** At the beginning ($k = 0$) of the iterative process, all system voltages must be set equal to zero.

Step 3: **Current Calculations:** The relationship between injected currents \mathbf{I}^k and branch currents \mathbf{J}^k is set through an upper triangular matrix **T** accomplishing the KCL.

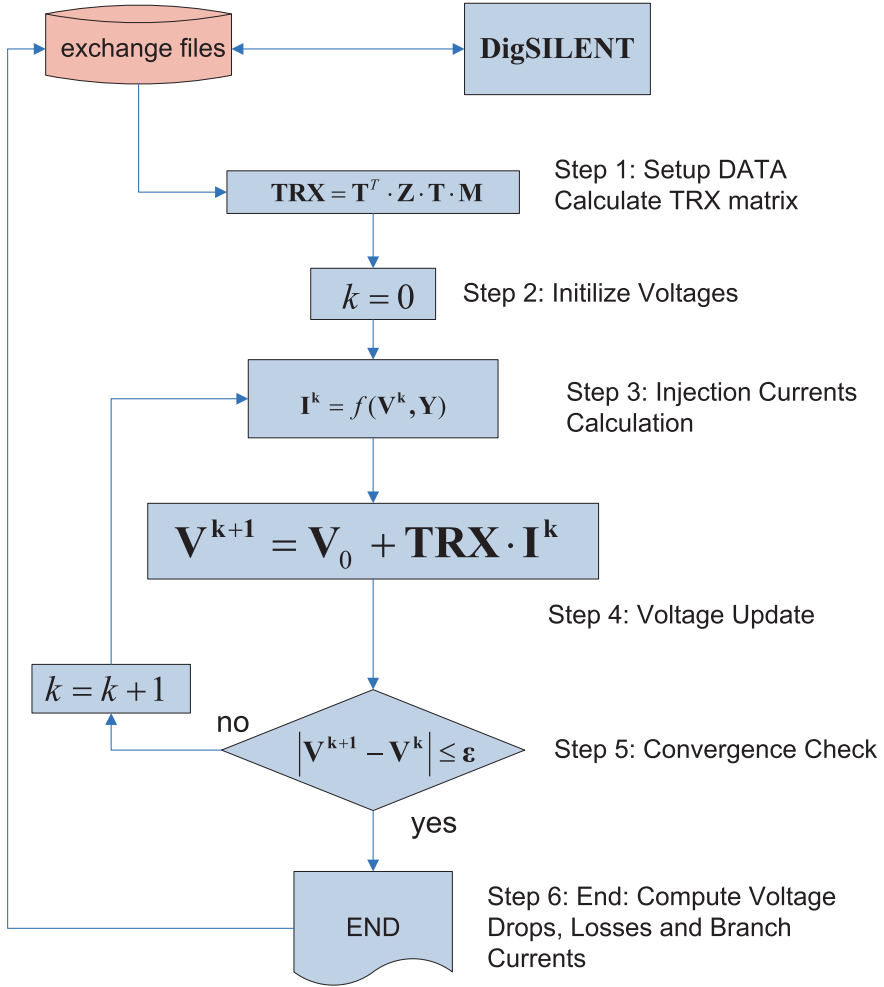


Fig. 4.3 The TRX 5-wire power flow algorithm

$$\mathbf{J}^k = -\mathbf{T} \cdot \mathbf{M} \cdot \mathbf{I}^k \quad (4.6)$$

Each element \bar{I}_{pj}^k of \mathbf{I}^k associated with node J and phase $p = a, b, c$ is calculated as function of injected powers \bar{S}_{pj} and its voltage profile \bar{V}_{pj}^k are as shown below:

$$\bar{I}_{pj}^k = -\frac{\bar{S}_{pj}^*}{\bar{V}_{pj}^{k*} - \bar{V}_{nj}^{k*}} + \sum_{r=a,b,c,n,g} \bar{Y}_{pr}^{ij} \cdot \bar{V}_{rj}^k \quad (4.7)$$

where \bar{Y}_{pp}^{ij} is the shunt of admittance at line section $i-j$ at phase p . Equivalent injected currents at neutral and ground point are calculated as follows:

$$\bar{I}_{nj}^k = \bar{I}_{gj}^k = - \sum_{p=a,b,c} \bar{I}_{pj}^k$$

Step 4: **Voltage Update:** The state of the system \mathbf{V}^{k+1} is updated at each iteration k accomplishing the Kirchhoff voltage laws (KVL) using previously calculated branch current vector \mathbf{J}^k :

$$\mathbf{V}^{k+1} = \mathbf{V}_0 - \mathbf{T}^T \cdot \text{diag}(\mathbf{Z}) \cdot \mathbf{J}^k \quad (4.8)$$

where

$$\text{diag}(\hat{\mathbf{Z}}) = \begin{bmatrix} \hat{\mathbf{Z}}^{01} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \hat{\mathbf{Z}}^{ij} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots \end{bmatrix} \quad (4.9)$$

It must be noted that $\mathbf{0}$ is 5×5 matrix with all entries set as 0. Using injected current Eq. 4.6, the state of the system Eq. 4.8 can be rewritten and a more general expression can be written as follows:

$$\mathbf{V} = \mathbf{V}_0 + \mathbf{TRX} \cdot \mathbf{I} \quad (4.10)$$

Injected currents could also be calculated considering constant current or constant impedance loads. Due to lack of space, these representations are not expressly included; however, general formulation for three-phase formulation can be easily extended to 5-wire model [10].

Step 5: **Convergence Check-in:** In order to perform convergence check-in, recent updated voltages are compared with previous voltages in all nodes as follows:

$$\varepsilon \leq \left| \bar{V}_{qj}^{k+1} - \bar{V}_{qj}^k \right| \quad j = 1, \dots, m \quad q = a, b, c, n \quad (4.11)$$

Step 6: **Final Calculations:** When convergence is reached, voltages \mathbf{V} and injected currents \mathbf{I} are available; then, branch currents \mathbf{J} and system power losses can be easily calculated including losses in the neutral and ground path.

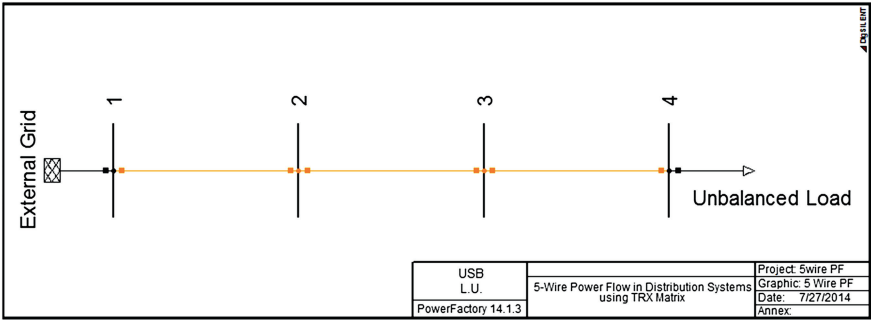


Fig. 4.4 Three-section distribution test system

4.5 Testing: 3-Section Distribution Test System

The proposed implementation of a 5-wire power flow analysis under DIGSILENT platform has been applied in the well-known N -section distribution test system. System model details are taken from [14, 15]. All simulations consider 4-wire configuration feeding unbalanced loads. Power system base is $S_{base} = 3\text{MVA}$. Ground impedance at node 0 is zero. For illustration purposes, the application has been applied considering three sections ($N = 3$) as shown in Fig. 4.4.

The I/O files (casedata.txt and casesolution.txt) can be found in Appendix E.

The number of iterations required to converge ($\varepsilon < 10^{-7}$) is 6. Power losses are 217.97 kW and 429.63 kVar. Current flows through neutral and ground paths are depicted in Fig. 4.5.

Note that distribution current distribution led to neutral-to-ground voltage at load node around 153 V. Nodes 2 and 3 show neutral-to-ground voltage around 102 and

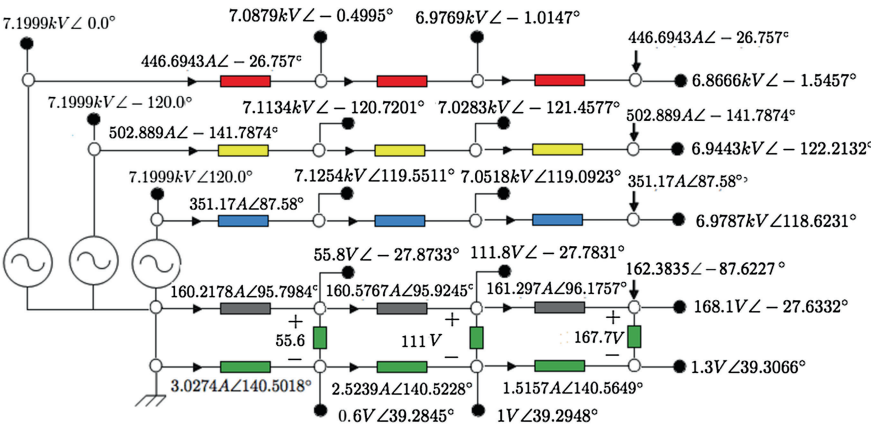


Fig. 4.5 Three-section distribution test system—current distribution

Table 4.1 Three-section distribution test system—results

Node	V_a (kV)	V_b (kV)	V_c (kV)	V_n (V)	V_g (V)
0	7.1999	7.1999	7.1999	000.0	000.0
1	7.0879	7.1134	7.1254	55.8	000.6
2	6.9769	7.0283	7.0518	111	001.0
3	6.8666	6.9443	6.9787	168.1	001.3
Node	θ_a (deg)	θ_b (deg)	θ_c (deg)	θ_n (deg)	θ_g (deg)
0	0.00	-120.00	120.00	0	0
1	-0.48	-120.73	119.54	-27.87	39.28
2	-1.01	-121.45	119.09	-27.78	39.29
3	-1.54	-122.21	118.62	-27.63	39.39

50 V, respectively. These voltages are local. General solution for phase, neutral, and ground voltages is presented in Table 4.1.

In this case, V_a , V_b , V_c , V_n , and V_g are line-to-ground voltages (at Node 0). Then, $V_n = 168.1$ V is referred to the reference node 0.

4.6 Conclusion

A DiGSILENT's power flow application capable of distribution system analysis by preserving the 5-wiring multigrounded configuration of distribution networks is proposed. This approach is particularly powerful for advanced smart grid applications such as power quality or power loss analysis. For a given distribution feeder, the matrix **TRX** gathers into a unique object of all feeder characteristics related to network topology and structure of phase, neutral, and equivalent ground paths.

Appendix A—Distribution Line Model

The nature of the distribution line requires an exact model in order to identify the self- and mutual impedances of the conductor taking into account the ground path for the unbalanced currents. Figure 4.6 shows a schematic distribution line section between nodes i and j (branch $i-j$). It consists of 4 wires: three phases (a , b , c) and neutral (n) with their respective self- and mutual impedance terms.

To include multigrounded neutrals, an equivalent conductor (g) that emulates the effect of ground return is required. The earth is a conductor of non-uniform conductivity. Thus, the ground current distribution is not uniform. The efforts aim to calculate the impedance of the phase and neutral conductors with ground-return effect. Researchers have faced and overcome this problem using different

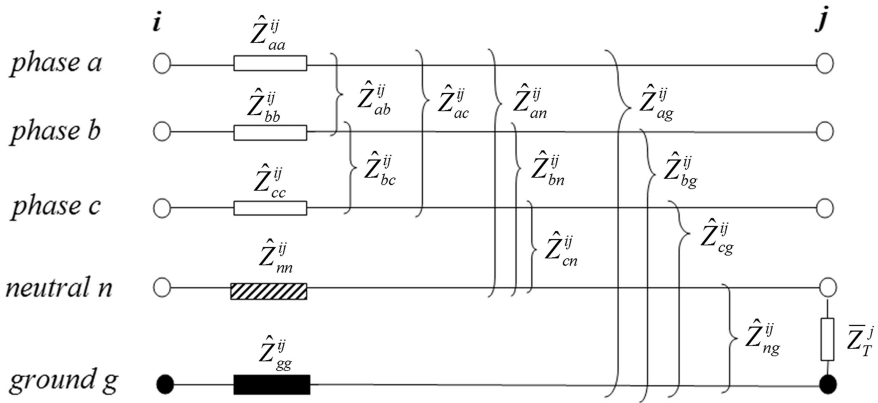


Fig. 4.6 Four-wire multigrounded distribution segment line

assumptions and methods. Carson's series-based asymptotic approximation equations [3] are generally accepted as the best, considering the following assumptions: The conductors are parallel to the ground, and the earth is a solid with a plane surface, infinite in extent, and of uniform conductivity [16]. Advanced approaches have been also proposed to avoid truncation errors due to cases with wide separation among conductors, frequency higher than power frequency, or low earth resistivity [17]. The key issue is how to model equivalent conductor that emulates the effect of ground return in order to obtain self- and mutual ground impedances \hat{Z}_{gs}^{ij} and \hat{Z}_{qg}^{ij} for $q = a, b, c, n$.

According to Anderson [18], the extended primitive impedance matrix $\hat{\mathbf{Z}}^{ij}$ in Ω per unit length is given by

$$\hat{\mathbf{Z}}^{ij} = \begin{bmatrix} \hat{Z}_{aa}^{ij} & \hat{Z}_{ab}^{ij} & \hat{Z}_{ac}^{ij} & \hat{Z}_{an}^{ij} & \hat{Z}_{ag}^{ij} \\ \hat{Z}_{ba}^{ij} & \hat{Z}_{bb}^{ij} & \hat{Z}_{bc}^{ij} & \hat{Z}_{bn}^{ij} & \hat{Z}_{bg}^{ij} \\ \hat{Z}_{ca}^{ij} & \hat{Z}_{cb}^{ij} & \hat{Z}_{cc}^{ij} & \hat{Z}_{cn}^{ij} & \hat{Z}_{cg}^{ij} \\ \hat{Z}_{na}^{ij} & \hat{Z}_{nb}^{ij} & \hat{Z}_{nc}^{ij} & \hat{Z}_{nn}^{ij} & \hat{Z}_{ng}^{ij} \\ \hat{Z}_{ga}^{ij} & \hat{Z}_{gb}^{ij} & \hat{Z}_{gc}^{ij} & \hat{Z}_{gn}^{ij} & \hat{Z}_{gg}^{ij} \end{bmatrix} \quad (4.12)$$

Self- and mutual impedances in Ω per unit length with no ground effect of all phases and neutral are given by Eqs. 4.13 and 4.14, respectively:

$$\hat{Z}_{qq}^{ij} = r_q + j4\pi \times 10^{-4} f \zeta \ln \frac{1}{GMR_q} \quad q = a, b, c, n \quad (4.13)$$

$$\hat{Z}_{ql}^{ij} = j4\pi \times 10^{-4} f \zeta \ln \frac{1}{D_{ql}} \quad q, l = a, b, c, n \quad (4.14)$$

where

GMR_q	Geometric mean radius of conductor q in feet
r_q	AC resistance of conductor q in Ω /mile
ξ	Constant equal to 1.609, if \hat{Z}_{qq}^{ij} and \hat{Z}_{ql}^{ij} are given in Ω /mile, and equal to 1.000, if \hat{Z}_{qq}^{ij} and \hat{Z}_{ql}^{ij} are given in Ω /km
D_{ql}	Distance between conductor q and l in feet
f	Frequency in Hz

Ciric [9] proposed a model where ground self-impedance \hat{Z}_{gg}^{ij} is explicitly represented. In this proposal, values of \hat{Z}_{gg}^{ij} are given in Ω per unit length and identified as the frequency-dependent terms in the modified Carson's equations according to the following expression:

$$\hat{Z}_{gg}^{ij} = \pi f \xi \times 10^{-4} \left[\pi - j8 \times 0.03868 + j4 \ln \frac{2}{5.6198 \times 10^{-3}} \right] \quad (4.15)$$

Equation 4.15 is presented as a frequency-dependent complex number with fixed resistive and inductive parts. However, according to [18], it can be demonstrated that inductive ground behavior depends on both frequency and geometric assumptions. In this case, Eq. 4.15 could be considered as a particular case. If the skin effect is ignored, self-impedance in Ω per unit length of the equivalent ground conductor can be written as follows [18]:

$$\hat{Z}_{gg}^{ij} = \pi^2 \xi \times 10^{-4} f + j4\pi \xi \times 10^{-4} f \left[\ln \frac{2s}{D_{sd}} - 1 \right] \quad (4.16)$$

where D_{sd} is the GMR of ground equivalent conductor and s is the length of wire, both in the same units. In Eq. 4.16, $\pi^2 \xi \times 10^{-4} f = 0.0953$ and $4\pi \xi \times 10^{-4} f = 0.12134$ in Ω /mile. It must be noticed that Eqs. 4.15 and 4.16 should be equivalent, and therefore, there exist several values of D_{sd} and s in order to accomplish both equations.

In this document, for the sake of simplicity, it is assumed \hat{Z}_{gg}^{ij} has no inductive part. This arbitrary assumption is also applied by [18] to derive self- and mutual ground equivalent impedances. As a result, self-ground impedance in Ω per unit length used in the remainder of this document is given by the following expression.

$$\hat{Z}_{gg}^{ij} = \pi^2 \xi \times 10^{-4} f \quad (4.17)$$

Mutual impedance between phase and neutral conductors in Ω per unit length with respect to ground \hat{Z}_{qg}^{ij} may be directly obtained from simplified Carson's equations as written in Kersting's book [2].

A direct relation between Carson's impedances Z_{qq}^{ij} and primitive impedances \widehat{Z}_{qq}^{ij} in Ω per unit length can be written as follows [18]:

$$Z_{qq}^{ij} - \widehat{Z}_{qq}^{ij} = Z_{ql}^{ij} - \widehat{Z}_{ql}^{ij} = \widehat{Z}_{gg}^{ij} - 2\widehat{Z}_{qg}^{ij} \quad (4.18)$$

$$Z_{qq}^{ij} = r_q + \pi^2 \xi \times 10^{-4} f + j4\pi \times 10^{-4} f \xi \left[\ln \frac{1}{GMR_q} \dots \right. \\ \left. + 7.6728 + \frac{1}{2} \ln \frac{\rho_{ij}}{f} \right] \quad q = a, b, c, n \quad (4.19)$$

$$Z_{ql}^{ij} = \pi^2 \xi \times 10^{-4} f + j4\pi \times 10^{-4} f \xi \left[\ln \frac{1}{D_{ql}} \dots \right. \\ \left. + 7.6728 + \frac{1}{2} \ln \frac{\rho_{ij}}{f} \right] \quad q, l = a, b, c, n \quad (4.20)$$

Then, from Eqs. 4.18–4.20, mutual ground impedances in Ω per unit length are given by

$$\widehat{Z}_{qg}^{ij} = 2\pi \xi \times 10^{-4} f \left[\frac{1}{2} \ln \frac{\rho_{ij}}{f} - 7.6728 \right] \quad q = a, b, c, n \quad (4.21)$$

where ρ_{ij} is the earth resistivity at section line i – j in Ω -m.

The grounding resistance of a node j between neutral and ground is given in Ω :

$$\overline{Z}_T^j = \frac{\rho_{ij}}{2\pi l} \left[\ln \frac{4l}{r_{\text{rod}}} - 1 \right] \quad (4.22)$$

where l is the ground rod length in meters and r_{rod} is the radius of ground rod in meters. For commercial ground rods used in distribution systems 10 ft and 5/8 in, ground impedance can be approximated by $\overline{Z}_T^j \approx \rho_{ij}/3.35$ [19]. For instance, for $\rho_{ij} = 100 \Omega$ -m, $\overline{Z}_T^j \approx 30 \Omega$

Finally, in order to complete the 4-wire multigrounded model for a given section i – j , primitive shunt admittance in μS per unit length is given by $\widehat{\mathbf{Y}}^{\text{ij}} = j\mathbf{B}^{\text{ij}} = jw\mathbf{C}^{\text{ij}}$ where $w = 2\pi f$ is the angular frequency and \mathbf{C}^{ij} is the capacitance primitive matrix. Details about self- and mutual capacitance calculations can be found in [2].

$$\widehat{\mathbf{Y}}^{\text{ij}} = \begin{bmatrix} \widehat{Y}_{aa}^{ij} & \widehat{Y}_{ab}^{ij} & \widehat{Y}_{ac}^{ij} & \widehat{Y}_{an}^{ij} & 0 \\ \widehat{Y}_{ba}^{ij} & \widehat{Y}_{bb}^{ij} & \widehat{Y}_{bc}^{ij} & \widehat{Y}_{bn}^{ij} & 0 \\ \widehat{Y}_{ca}^{ij} & \widehat{Y}_{cb}^{ij} & \widehat{Y}_{cc}^{ij} & \widehat{Y}_{cn}^{ij} & 0 \\ \widehat{Y}_{na}^{ij} & \widehat{Y}_{nb}^{ij} & \widehat{Y}_{nc}^{ij} & \widehat{Y}_{nn}^{ij} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (4.23)$$

Appendix B—DPL5Wire Code

Script developed in DPL for exchanging files with MATLAB platform.

```
!DPL Program for TRX multifilar approach

set Grids, Lines,Nodes,Loads,Cond,Towers_set;

object pLne, pGrd, pNds,pLds,bus,bus1,pVp,busp,cargal,pBase,
      pTow,pCond,pTyp_line,busi,busj,Fc,Lib_Tow,Lib_Cond,TowersG;

double lap,lbp,lcp,laq,lbq,lcq,unom,long,coord_xy_e,coord_xy_p,x,
      cond_gmr,gmr,kVnom,sang,us,xl,yl,cond_res,y,rground;

string busi_name, busj_name,Lname,s,bustype,Cname,s1,s2;

int fp,cp,fn,cn,r,c,rdc;

Fc = GetCaseObject('ComLdf'); Fc.Execute();
fopen('C:\casedata.txt','w',0);

Grids =AllRelevant('*.ElmNet'); }

!LINES
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
pGrd = Grids.First(); Lines = pGrd.GetContents('*.*ElmLne*',1);

pLne= Lines.First(); while (pLne){
    pTyp_line=pLne:typ_id;
    busi = pLne.GetNode(0,0);
    busj = pLne.GetNode(1,0);
    kVnom=busi.Unom(); !Nominal Voltage
    Lname = pLne:loc_name;
    busi_name = busi:loc_name;
    busj_name = busj:loc_name;
    long = pLne:dline; !Length
    !fprintf(0,'%g\t%s\t%s\t%f',kVnom,busi_name,busj_name,long);
    fprintf(0,'%s',busi_name);
    fprintf(0,'%s',busj_name);
    pLne = Lines.Next();
}
fprintf(0,'%f',long);
fprintf(0,'%g',kVnom);

!CONDUCTORS
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Lib_Cond=GetLocalLib('TypCon'); Cond=Lib_Cond.GetContents();
pCond=Cond.First();
while (pCond) {
    Cname = pCond:loc_name;
    cond_gmr=pCond.GetVal(gmr,'erpha');
    cond_res=pCond.GetVal(rdc,'rpha');
    if (.not.cond_gmr){
        fprintf(0,'%f',gmr);
    }
}
```

```

        fprintf(0,'%f',rdc);
    };
    pCond=Cond.Next();
};

!TOWERS
!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Towers_set= AllRelevant('*.TypTow'); !Tower Set
TowersG=Towers_set.First(); !Call to objects belong to Tower
coord_xy_p=TowersG.GetSize('xy_c',fp,cp);
coord_xy_e=TowersG.GetSize('xy_e',fn,cn);

if (.not.coord_xy_p)
c=0; s='';
while (c<cp){
coord_xy_p=TowersG.GetVal(x,'xy_c',r,c);
fprintf(0,'%g',x);
if (.not.coord_xy_p) c+=1; } sl='';
c=0;
while (c<cn){
coord_xy_e=TowersG.GetVal(y,'xy_e',r,c);
fprintf(0,'%g',y);
if (.not.coord_xy_e) c+=1; }

for(TowersG=Towers_set.First();TowersG;TowersG=Towers_set.Next()) {
rground=TowersG:rearth;
fprintf(0,'%f',rground);
}
!LOADS
!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
pGrd = Grids.First(); Loads = pGrd.GetContents('*.ElmLod',1);
pLds=Loads.First(); while (pLds){
!pLds.ShowFullName();
Loadl=pLds.GetParent();
busi_name=pLds:loc_name;
lap=pLds:plinir;
lbp=pLds:plinis;
lcp=pLds:plinit;
laq=pLds:qlinir;
lbq=pLds:qlinis;
lcq=pLds:qlinit;
fprintf(0,'%f\n%f\n%f\n%f\n%f',lap,lbp,lcp,laq,lbq,lcq);
pLds=Loads.Next();
}
fclose (0);

```

After querying data from DIgSILENT exchanging file:

```

clear all
% clc
% DATA=load('casedata.txt');
% %clc
% GMRp=DATA(11)*0.0032808;%feet
% rp=DATA(12)*1.61;%ohm/mile
% GMRn=DATA(9)*0.0032808;%feet
% rn=DATA(10)*1.61;%ohm/mile
% f=60;%Hz
% rvd=DATA(21);%
% eta=1.6093;%Impedances are Given in ohm/mile
% %Spacings

```

```

% X1p=DATA(13);X2p=DATA(14);X3p=DATA(15);
% Y1p=DATA(16);Y2p=DATA(17);Y3p=DATA(18);
% X1n=DATA(19);Y1n=DATA(20);
% Dab=(abs(X1p)-abs(X2p))*3.2808;%feet
% Dbc=(abs(X2p)+abs(X3p))*3.2808;%feet
% Dac=Dab+Dbc;%feet
% Dcn=sqrt((X3p-X1n)^2+(Y3p-Y1n)^2)*3.2808;%feet
% Dbn=sqrt((X2p-X1n)^2+(Y2p-Y1n)^2)*3.2808;%feet
% Dan=sqrt((X1p-X1n)^2+(Y1p-Y1n)^2)*3.2808;%feet
% hqa=29;%feet
% hqb=29;%feet
% hqc=29;%feet
% hqn=25;%feet
% si=0;
% zp(1,1)=rp+i*(4*pi*0.0001*f*eta*(log(inv(GMRp))));%ohm/mile
% zp(2,2)=rp+i*(4*pi*0.0001*f*eta*(log(inv(GMRp))));%ohm/mile
% zp(3,3)=rp+i*(4*pi*0.0001*f*eta*(log(inv(GMRp))));%ohm/mile
% zp(4,4)=(rn+i*(4*pi*0.0001*f*eta*(log(inv(GMRn))));%ohm/mile
% zp(1,2)=i*(4*pi*0.0001*f*eta*(log(inv(Dab))));%ohm/mile
% zp(1,3)=i*(4*pi*0.0001*f*eta*(log(inv(Dac))));%ohm/mile
% zp(1,4)=i*(4*pi*0.0001*f*eta*(log(inv(Dan))));%ohm/mile
% zp(2,3)=i*(4*pi*0.0001*f*eta*(log(inv(Dbc))));%ohm/mile
% zp(2,4)=i*(4*pi*0.0001*f*eta*(log(inv(Dbn))));%ohm/mile
% zp(3,4)=i*(4*pi*0.0001*f*eta*(log(inv(Dcn))));%ohm/mile
% zp(2,1)=zp(1,2);%ohm/mile
% zp(3,1)=zp(1,3);%ohm/mile
% zp(4,1)=zp(1,4);%ohm/mile
% zp(3,2)=zp(2,3);%ohm/mile
% zp(4,2)=zp(2,4);%ohm/mile
% zp(4,3)=zp(3,4);%ohm/mile
% zp(1,5)=i*(.5*log(f/rvd)-7.6728)*2*pi*f*.0001*eta;%ohm/mile,
% zp(2,5)=i*(.5*log(f/rvd)-7.6728)*2*pi*f*.0001*eta;%ohm/mile,
% zp(3,5)=i*(.5*log(f/rvd)-7.6728)*2*pi*f*.0001*eta;%ohm/mile,
% zp(4,5)=i*(.5*log(f/rvd)-7.6728)*2*pi*f*.0001*eta;%ohm/mile,
% zp(5,5)=pi^2*f*.0001*eta;%ohm/mile
% zp(5,1)=zp(1,5);%ohm/mile
% zp(5,2)=zp(2,5);%ohm/mile
% zp(5,3)=zp(3,5);%ohm/mile
% zp(5,4)=zp(4,5);%ohm/mile
%
% Sbase=3;%MVA Base at High Voltage
% Vbase=12.47/sqrt(3);%kV base
% Ibase=1000*Sbase/Vbase;
% Zbase=Vbase^2/Sbase;%Zbase high in ohms
%
% zt0=.00001;%grounding impedance at root node (source)
% zt1=100.00001;%ground impedance at bus 1
% zt2=100.00001;%ground impedance at bus 2
% zt3=100.00001;%ground impedance at bus 3
%
% L=DATA(7)*0.6213;
% n=3;
% ZL=zp*L/Zbase;%section impedance ohms
%
% %powers at bus 1
% Salm=0;Sblm=0;Sc1m=0;
% fpa1=0;fpa1=0;fpc1=0;
%
% %powers at bus 2
% Sa2m=0;Sb2m=0;Sc2m=0;
% fpa2=0;fpa2=0;fpc2=0;
%
% %powers at bus 3
% Pa3m=DATA(22);Pb3m=DATA(23);Pc3m=DATA(24);
% Qa3m=DATA(25);Qb3m=DATA(26);Qc3m=DATA(27);
% Sa3m=sqrt(Pa3m^2+Qa3m^2);
% Sb3m=sqrt(Pb3m^2+Qb3m^2);

```

```

% Sc3m=sqrt(Pc3m^2+Qc3m^2);
% fpa3=Pa3m/Sa3m;
% fpb3=Pb3m/Sb3m;
% fpc3=Pc3m/Sc3m;
%
% %Build the T matrix
% for i=1:n*5
%     % for k=1:n*5
%         % if i==k
%             % Tabcng(i,k)=1;
%         % end
%     % end
% end
% for m=5:5:(n-1)*5
%     % for i=1:n*5
%         % for k=1:n*5
%             % if k<=(n-1)*5+5-m
%                 % if i==k
%                     % Tabcng(i,k+m)=1;
%                 % end
%             % end
%         % end
%     % end
% end
%
% j=sqrt(-1);
% %j=0;
% Sa3=-(Sa3m*fpa3+j*Sa3m*(1-fpa3^2)^.5)/Sbase;
% Sb3=-(Sb3m*fpb3+j*Sb3m*(1-fpb3^2)^.5)/Sbase;
% Sc3=-(Sc3m*fpc3+j*Sc3m*(1-fpc3^2)^.5)/Sbase;
% Pa3=real(Sa3);
% Pb3=real(Sb3);
% Pc3=real(Sc3);
% Qa3=imag(Sa3);
% Qb3=imag(Sb3);
% Qc3=imag(Sc3);
%
% Sa2=-(Sa2m*fpa2+j*Sa2m*(1-fpa2^2)^.5)/Sbase;
% Sb2=-(Sb2m*fpb2+j*Sb2m*(1-fpb2^2)^.5)/Sbase;
% Sc2=-(Sc2m*fpc2+j*Sc2m*(1-fpc2^2)^.5)/Sbase;
% Pa2=real(Sa2);
% Pb2=real(Sb2);
% Pc2=real(Sc2);
% Qa2=imag(Sa2);
% Qb2=imag(Sb2);
% Qc2=imag(Sc2);
%
% Sa1=-(Sa1m*fpa1+j*Sa1m*(1-fpa1^2)^.5)/Sbase;
% Sb1=-(Sb1m*fpb1+j*Sb1m*(1-fpb1^2)^.5)/Sbase;
% Sc1=-(Sc1m*fpc1+j*Sc1m*(1-fpc1^2)^.5)/Sbase;
% Pa1=real(Sa1);
% Pb1=real(Sb1);
% Pc1=real(Sc1);
% Qa1=imag(Sa1);
% Qb1=imag(Sb1);
% Qc1=imag(Sc1);
% a=complex(cos(2*pi/3),sin(2*pi/3));
% a2=a^2;
% Vom=1;
% Vo=zeros(5*n,1);
% for i=1:5:5*n
%     % Vo(i,1)=(Vom);
%     % Vo(i+1,1)=(Vom*a2);
%     % Vo(i+2,1)=(Vom*a);
% end
% V123=Vo;
% Zabcng=zeros(5*n,5*n);

```

```

% for j=1:5:n
%   for i=1:5
%     for k=1:5
%       Zabcng(i+j-1,k+j-1)=(ZL(i,k));
%     end
%   end
% end
% %Currebt Divisors calculations
% A=zt3+ZL(5,5)*Zbase;
% B=zt0+ZL(5,5)*Zbase;
% C=ZL(4,4)*Zbase;
% D=ZL(4,4)*Zbase;
% E=zt2;
% F=ZL(4,4)*Zbase;
% G=zt1;
% H=ZL(5,5)*Zbase;
% Fp=(F+C+C*G+F*G)/F;
% Cp=(F+C+C*G+F*G)/C;
% Gp=(F+C+C*G+F*G)/G;
% trid=G+C+H+E*(D+A)/(D+E+A);
% trid3=(G*trid)/(G+trid);
% zy11=(trid/(G+trid))*(F/(F+B+trid3));
% zy21=((A+D)/(E+A+D))*(G/(G+trid))*(F/(F+B+trid3));
% zy31=((E)/(E+A+D))*(G/(G+trid))*(F/(F+B+trid3));
% a1=(E*(D+A)/(E+D+A));
% a2=B*Cp/(B+Cp);
% a3=((A+D)/(E+A+D));
% a4=((E)/(E+A+D));
% trid2=Fp*(H+a1)/(Fp+H+a1);
% zy22=(Gp/(Gp+a2+trid2))*(Fp/(Fp+H+a1))*a3;
% zy32=(Gp/(Gp+a2+trid2))*(Fp/(Fp+H+a1))*a4;
% zy12=(Gp/(Gp+a2+trid2))*(Cp/(B+Cp)-Fp/(Fp+H+a1));
% Bpp=(B*Cp)/(B+Cp)+H*Fp/(Fp+E+H);
% Cpp=Gp;
% Epp=Fp*E/(Fp+E+H);
% App=A+H*E/(Fp+E+H);
% Dpp=D;
% Dppp=(Epp*Cp+Cp*Dpp+Dpp*Epp)/Dpp;
% Eppp=(Epp*Cp+Cp*Dpp+Dpp*Epp)/Epp;
% Cppp=(Epp*Cp+Cp*Dpp+Dpp*Epp)/Cp;
% beta= Dppp*Bpp/(Dppp+Bpp)+Cppp*App/(Cppp+App);
% zy33=(Cppp/(Cppp+App))*(Eppp/(Eppp+beta));
% W=zy33*H-(1-zy33)*C;
% P=F*(1-zy33)-B*zy33;
% M=G+F+B;
% zy23=(P*G-W*M)/((C+E+H)*M+(B+F)*G);
% zy13=(P-(B+F)*zy23)/M;
% Tr=zeros(5*n,5*n);
% for i=1:5*n
%   Tr(i,i)=1;
% end
% Tr(4,4)=(1-zy11);
% Tr(4,9)=-zy12;
% Tr(4,14)=-zy13;
% Tr(5,5)=zy11;
% Tr(5,10)=zy12;
% Tr(5,15)=zy13;
% Tr(9,4)=-zy21;
% Tr(9,9)=(1-zy22);
% Tr(9,14)=-zy23;
% Tr(10,5)=zy21;
% Tr(10,10)=zy22;
% Tr(10,15)=zy23;
% Tr(14,4)=-zy21;
% Tr(14,9)=-zy32;
% Tr(14,14)=(1-zy33);
% Tr(15,5)=zy31;

```

```

% Tr(15,10)= zy32;
% Tr(15,15)= zy33;
% TRXabcng=Tabcng'*Zabcng*Tabcng*Tr;
% delta=1;
% V123x=Vo;
% iter=0;
% while min(abs(delta))> .0000001
%   iter=iter+1;
%   I(1)=conj(Sa1/(V123(5*n-4-10)));
%   I(2)=conj(Sb1/(V123(5*n-3-10)));
%   I(3)=conj(Sc1/(V123(5*n-2-10)));
%   I(6)=conj(Sa2/(V123(5*n-4-5)));
%   I(7)=conj(Sb2/(V123(5*n-3-5)));
%   I(8)=conj(Sc2/(V123(5*n-2-5)));
%   I(6+5)=conj(Sa3/(V123(5*n-4)));
%   I(7+5)=conj(Sb3/(V123(5*n-3)));
%   I(8+5)=conj(Sc3/(V123(5*n-2)));
%   Id1=(-I(1)-I(2)-I(3));
%   Id2=(-I(1+5)-I(2+5)-I(3+5));
%   Id3=(-I(1+10)-I(2+10)-I(3+10));
%   I(4)=Id1;%
%   I(5)=Id1;%
%   I(9)=Id2;%
%   I(10)=Id2;%
%   I(14)=Id3;%
%   I(15)=Id3;%
%   V123=Vo+TRXabcng*I.';
%   if I(4)*I(5)*I(9)*I(10)*I(11)+I(15)==0
%     delta(1)=V123(1)-V123x(1);
%   else
%     delta=V123-V123x;
%   end
%   V123x=V123;
% end%end while
% Jabcng=-(Tabcng*Tr*I. ');
% mJabcng=abs(Jabcng)*Ibase;
% aJabcng=angle(Jabcng)*180/pi;
% mV123=abs(V123);
% aV123=angle(V123)*180/pi;
% mV123real=abs(V123)*Vbase*1000;
% Ing=(Tr*I. ');
% iter;
% NEV2=abs(Ing(10)*zt2)*Ibase; %tension local volts
% DVn=1000*abs(V123(9))*Vbase; %voltage drop at neutral wire volts
% DVg=1000*abs(V123(10))*Vbase; %voltage drop at ground equivalent wire volts
% %% Loss Study
% DemandP=-Sbase*(Pa2+Pb2+Pc2+Pa1+Pb1+Pc1+Pa3+Pb3+Pc3);%Total Active Power Load
% DemandQ=-Sbase*(Qa2+Qb2+Qc2+Qa1+Qb1+Qc1+Qa3+Qb3+Qc3);%Total Reactive Power Load
% GenerP=Sbase*real(Vo(1)*conj(Jabcng(1))+Vo(2)*conj(Jabcng(2))+Vo(3)*conj(Jabcng(3)));
% GenerQ=Sbase*imag(Vo(1)*conj(Jabcng(1))+Vo(2)*conj(Jabcng(2))+Vo(3)*conj(Jabcng(3)));
% LossP=(GenerP-DemandP)*1000;%Loss in kW
% LossQ=(GenerQ-DemandQ)*1000;%Loss in kvar
% VT=vertcat(mV123real,aJabcng,LossP,LossQ);
% FID=fopen('caseresults.txt','w');
% for i=1:length(VT)
%   fprintf(FID,'%f\n',VT(i,1));
% end
% fclose(FID);

```

Appendix C—DPL Script for Displaying Final Results in DIgSILENT

```
double V1a,V1b,V1c,V1n,V1g,V2a,V2b,V2c,V2n,V2g,V3a,V3b,V3c,V3n,V3g,
T1a,T1b,T1c,T1n,T1g,T2a,T2b,T2c,T2n,T2g,T3a,T3b,T3c,T3n,T3g,
Ploss,Qloss;
string V1_a,V1_b,V1_c,V1_n,V1_g,V2_a,V2_b,V2_c,V2_n,V2_g,
V3_a,V3_b,V3_c,V3_n,V3_g,T1_a,T1_b,T1_c,T1_n,
T1_g,T2_a,T2_b,T2_c,T2_n,T2_g,T3_a,T3_b,
T3_c,T3_n,T3_g,Node1;
fopen('C:\caseresults.txt','r',1);
fscanf(1,'%d\n%d\n%d\n%d\n%d\n%d\n%d\n%d\n%d\n%d\n%d\n%d\n',V1a,V1b,V1c,V1n,
V1g,V2a,V2b,V2c,V2n,V2g,V3a,V3b,V3c,V3n,V3g);
fscanf(1,'%d\n%d\n%d\n%d\n%d\n%d\n%d\n%d\n%d\n%d\n%d\n%d\n',T1a,T1b,T1c,T1n,
T1g,T2a,T2b,T2c,T2n,T2g,T3a,T3b,T3c,T3n,T3g);
fscanf(1,'%d\n%d',Ploss,Qloss);
printf('%s\n%s\n%s\n%s\n', V1_a,V1_b,V1_c,V1_n,V1_g);
printf('%f\n%f\n%f\n%f\n', V1a,V1b,V1c,V1n,V1g);
printf('%s\n%s\n%s\n%s\n', V2_a,V2_b,V2_c,V2_n,V2_g);
printf('%f\n%f\n%f\n%f\n', V2a,V2b,V2c,V2n,V2g);
printf('%s\n%s\n%s\n%s\n', V3_a,V3_b,V3_c,V3_n,V3_g);
printf('%f\n%f\n%f\n%f\n', V3a,V3b,V3c,V3n,V3g);
printf('%s\n%s\n%s\n%s\n',T1_a,T1_b,T1_c,T1_n,T1_g);
printf('%f\n%f\n%f\n%f\n', T1a,T1b,T1c,T1n,T1g);
printf('%s\n%s\n%s\n%s\n', T2_a,T2_b,T2_c,T2_n,T2_g);
printf('%f\n%f\n%f\n%f\n', T2a,T2b,T2c,T2n,T2g);
printf('%s\n%s\n%s\n%s\n', T3_a,T3_b,T3_c,T3_n,T3_g);
printf('%f\n%f\n%f\n%f\n', T3a,T3b,T3c,T3n,T3g);
printf('%f\n', Ploss,Qloss);
fclose(1);
```

Appendix D—Case Study: I/O Files Structure

See Tables 4.2 and 4.3.

Table 4.2 Casedata.txt file elements

Casedata. txt—var	MATLAB equivalent var.	Unit	Description
1	N.A.	N.A.	Node 1. Using for T matrix construction
2	N.A.	N.A.	Node 2. Using for T matrix construction
3	N.A.	N.A.	Node 2. Using for T matrix construction
4	N.A.	N.A.	Node 3. Using for T matrix construction
3	N.A.	N.A.	Node 3. Using for T matrix construction
4	N.A.	N.A.	Node 4. Using for T matrix construction
0.6096	L	km	Length
12.47	$kVll$	kV	Nominal voltage
2.4811	$GMRn$	mm	Geometric mean radius for neutral cond.
0.3678	rn	$\frac{\Omega}{\text{km}}$	DC Resistance for neutral cond.
7.43712	$GMRp$	mm	Geometric mean radius for phase cond.
0.19006	rn	$\frac{\Omega}{\text{km}}$	DC Resistance for phase cond.
−1.2192	$X1p$	m	x -coordinate for phase a
−0.4572	$X2p$	m	x -coordinate for phase b
0.9144	$X3p$	m	x -coordinate for phase c
8.5344	$Y1p$	m	y -coordinate for phase a
8.5344	$Y2p$	m	y -coordinate for phase b
8.5344	$Y3p$	m	y -coordinate for phase c
0	Xln	m	x -coordinate for neutral cond.
7.62	Yln	m	y -coordinate for neutral cond.
100	rvd	$\Omega \cdot \text{m}$	Soil resistivity
2.7	$Pa3m$	MW	Active power demand in phase a
3.325	$Pb3m$	MW	Active power demand in phase b
2.125	$Pc3m$	MW	Active power demand in phase c
1.3077	$Qa3m$	MVAr	Reactive power demand in phase a
1.0928	$Qb3m$	MVAr	Reactive power demand in phase b
1.3170	$Qc3m$	MVAr	Reactive power demand in phase c

Table 4.3 Casesolution.txt elements

Casesolution.txt—var	DPL code var. description
<i>V1a</i>	Voltage at node 1—phase <i>a</i>
<i>V1b</i>	Voltage at node 1—phase <i>b</i>
<i>V1c</i>	Voltage at node 1—phase <i>c</i>
<i>V1n</i>	Voltage at node 1—neutral <i>n</i>
<i>V1 g</i>	Voltage at node 1—ground <i>g</i>
<i>V2a</i>	Voltage at node 2—phase <i>a</i>
<i>V2b</i>	Voltage at node 2—phase <i>b</i>
<i>V2c</i>	Voltage at node 2—phase <i>c</i>
<i>V2n</i>	Voltage at node 2—neutral <i>n</i>
<i>V2 g</i>	Voltage at node 2—ground <i>g</i>
<i>V3a</i>	Voltage at node 3—phase <i>a</i>
<i>V3b</i>	Voltage at node 3—phase <i>b</i>
<i>V3c</i>	Voltage at node 3—phase <i>c</i>
<i>V3n</i>	Voltage at node 3—neutral <i>n</i>
<i>V3 g</i>	Voltage at node 3—ground <i>g</i>
<i>T1a</i>	Angle at node 1—phase <i>a</i>
<i>T1b</i>	Angle at node 1—phase <i>b</i>
<i>T1c</i>	Angle at node 1—phase <i>c</i>
<i>T1n</i>	Angle at node 1—phase <i>n</i>
<i>T1g</i>	Angle at node 1—phase <i>g</i>
<i>T2a</i>	Angle at node 2—phase <i>a</i>
<i>T2b</i>	Angle at node 2—phase <i>b</i>
<i>T2c</i>	Angle at node 2—phase <i>c</i>
<i>T2n</i>	Angle at node 2—phase <i>n</i>
<i>T2g</i>	Angle at node 2—phase <i>g</i>
<i>T3a</i>	Angle at node 3—phase <i>a</i>
<i>T3b</i>	Angle at node 3—phase <i>b</i>
<i>T3c</i>	Angle at node 3—phase <i>c</i>
<i>T3n</i>	Angle at node 3—phase <i>n</i>
<i>T3 g</i>	Angle at node 3—phase <i>g</i>
<i>Ploss</i>	Total power active loss
<i>Qloss</i>	Total power reactive loss

References

1. DiGSILENT (2012) PowerFactory users manual 14.1—power system analysis functions, vol II. DiGSILENT GmbH, Technical Report
2. Kersting WH (2002) Distribution system modeling and analysis. CRC Press, Boca Raton
3. Carson JR (1926) Wave propagation in overhead wires with ground return. Bell Syst Tech J 5:539–554
4. Monfared M, Daryani AM, Abedi M (2006) Three phase asymmetrical load flow for four-wire distribution networks. In: Power systems conference and exposition, pp 1899–1903

5. Penido DRR, Araujo LR, Pereira JLR, Garcia PAN, Carneiro J (2004) Four wire Newton-Raphson power flow based on the current injection method. In: Proceedings of 2004 IEEE power engineering society power systems conference and exposition, vol 1. pp 239–242
6. Penido DRR, Araujo LR, Carneiro S (2008) Three-phase power flow based on four-conductor current injection method for unbalanced distribution networks. *IEEE Trans Power Syst* 23 (2):494–503
7. Penido DRR, de Araujo LR, Junior SC, Rezende Pereira JL (2013) A new tool for multiphase electrical systems analysis based on current injection method. *Electr Power Energy Syst* 44:410–420
8. Ciric RM, Padilha-Feltrin A, Ochoa LF (2003) Power flow in four-wire distribution networks. *IEEE Trans Power Syst* 18(4):1283–1290
9. Ciric RM, Ochoa LF, Padilha-Feltrin A (2004) Power flow in distribution networks with earth return. *Electr Power Energy Syst* 26:373–380
10. De Oliveira-De Jesus PM, Alvarez MA, Yusta JM (2013) Distribution power flow method based on a real quasi-symmetric matrix. *Electr Power Syst Res* 95:148–159
11. IEC 61968-1 (2003) Application integration at electric utilities—system interfaces for distribution management, part 1: interface architecture and general requirements. International Electrotechnical Commission, pp 1–7
12. Shirmohammadi D, Hong HW, Semlyen A, Luo GX (1988) A compensation-based power flow method for weakly meshed distribution and transmission networks. *IEEE Trans Power Deliv* 3(2):753–762
13. MATLAB/Simulink, The language of technical computing. Available at <http://www.mathworks.com>. Accessed on Feb 2013
14. Sunderman WG, Dugan RC, Dorr DS (2008) The neutral-to-earth voltage (NEV) test case and distribution system analysis. In: Power and energy society general meeting—conversion and delivery of electrical energy in the 21st Century, 20–24 July 2008
15. Kersting WH (2008) A three-phase unbalanced line model with grounded neutrals through a resistance. In: Power and energy society general meeting—conversion and delivery of electrical energy in the 21st Century, 20–24 July 2008
16. Blackburn JL (1993) Symmetrical components for power systems engineering. In: Series on electrical engineering and electronics. Marcel Dekker, New York, pp 296–297
17. Beaty HW (2001) Handbook of electric power calculations. In: Series on electrical engineering and electronics. McGraw-Hill, New York, pp 9.13–9.14 (Chapter 9: Overhead transmission lines and underground cables)
18. Anderson PM (1995) Analysis of faulted power systems, power systems engineering series. IEEE Press, Piscataway, pp 71–83
19. IEEE recommended practice for grounding of industrial and commercial power systems. IEEE Std. 142-1991, June 1992

Chapter 5

DC Optimal Power Flow Formulation Using the Power Transmission Distribution Factors—A DIgSILENT Programming Language Application

Víctor Hinojosa-Mateus, Leonardo Pérez-Andrades and Jovan Ilić

Abstract This research presents an application of different formulations to solve the DC optimal power flow problem. The classical DC optimization problem is reformulated using the power transmission distribution factors (PTDF). The main advantage of this formulation is the reduction of the decision variables in the optimization problem. In this chapter, the interior-point algorithm to solve the DC optimization problem is programmed using DIgSILENT Programming Language (DPL). In order to evaluate and compare the solution obtained by the algorithm, the simulations results are validated using *MatPower*. The proposed formulation has been implemented and validated with different test power systems. The results of the proposed algorithm demonstrate both the feasibility of applying the methodology to the DC optimal power flow problem and the potential applicability of the approach in medium- and large-scale power systems. The simulations performed in this chapter validate the formulation proposed in this study to solve the DCOPF problem. Finally, this application adds a new dimension to the DIgSILENT *Power Factory* by allowing the creation of new optimization algorithms to solve operation and planning problems.

Keywords Optimal power flow • Distribution factors • DC formulation • Interior-point algorithm

Electronic supplementary material The online version of this chapter (doi:10.1007/978-3-319-12958-7_5) contains supplementary material, which is available to authorized users.

V. Hinojosa-Mateus (✉) · L. Pérez-Andrades
Department of Electrical Engineering, Universidad Técnica Federico Santa María,
Valparaíso, Avenida España 1680, Valparaíso, Chile
e-mail: victor.hinojosa@usm.cl

L. Pérez-Andrades
e-mail: leonardo.perez.andrades@gmail.com

J. Ilić
Department of Electrical and Computer Engineering, Carnegie Mellon University, Forbes
Avenue 5000, Pittsburgh, USA
e-mail: jilic@ece.cmu.edu

Nomenclature

X	decision variables
X_{\min}, X_{\max}	variable bounds
n	number of decision variables
Φ_e, Φ_i	number of equality and inequality constraints
N_b	number of buses
N_G	number of thermal units
N_L	number of transmission lines
$C(P_i)$	fuel cost of the i -th thermal unit
P_i	active power supplied by thermal unit i
P_i^{\min}	minimum active power of the unit i
P_i^{\max}	maximum active power of the unit i
f_{ij}	total power flow in the transmission line i - j
f_{ij}^{\max}	maximum power flow of the transmission line i - j
x_{ij}	reactance of the branch i - j
P_k	net active power injections at bus k
P_k^d	active load of the bus k
D	total load of the system
θ_k	voltage bus angle at bus k
B	full susceptance matrix
S	branch-node incidence matrix
A, A_r	full and reduced bus incidence matrix

5.1 Introduction

The concept of the optimal power flow (OPF) was first proposed by Carpentier [1] in the early 1960s based on the economic dispatch problem. The OPF problem has been largely discussed in the specialized literature, for instance in 2013 [2–4]. A wide range of models have been developed and adopted to formulate different kinds of optimization problems through different objective functions, different sets of decision variables, and different kinds of constraints [4–7].

The OPF problem is concerned with the optimization of steady-state power system performance, subject to various equality, and inequality constraints. The OPF analysis aims to determine the power system operation state according to cost, planning, or reliability criteria without violating system and equipment operating limits. It is one of the most intensely used tool in many power engineering applications for network optimization, voltage control, state estimation, generation planning, transmission planning, and markets studies [8].

The alternating current—OPF (ACOPF) problem is very important. In real power systems, a one percent improvement in dispatch derived from better solutions to this problem could save approximately 1–5 billion dollars per year in the USA, or 4–20 billion dollars per year in the world [9]. An ACOPF that accurately

models all constraints and controls, with an objective function of minimizing cost, would inherently meet the objectives of minimizing generator fuel costs, minimizing losses, and minimizing control actions. When it is not feasible to run a full ACOPF problem due to time constraints, computing power, or lack of a robust solution algorithm, a common substitute is to decouple the problem and iterate with an approximation of the nonlinear constraints.

5.1.1 Direct Current (DC) Optimal Power Flow (DCOPF)

The problem involved in solving an AC power flow can be illustrated by the use of direct current (DC) network. A DCOPF problem is an approximation for an underlying ACOPF under several simplifying restrictions regarding voltage magnitudes, voltage angles, admittances, and reactive power. For the DC approximation, the following three assumptions are considered:

- (i) Branches can be considered lossless (in particular branch resistances and charging capacitances are negligible);
- (ii) Bus voltage magnitudes in the buses are close to 1.0 pu; and
- (iii) Voltage angle differences across branches are small enough. In addition, a further simplification of the AC power flow equations involves simply dropping the Q versus V relation altogether [10]. In this case, the voltage magnitudes and reactive power are eliminated from the problem, so the real power flows are linear functions of the voltage angles.

In the literature review, the most accepted model used by researchers in power systems problems is the so-called DC model. Linear solvers, such as the simplex method, are widely available for linear versions of the OPF problem. In electrical power systems, the DC formulation is broadly used by researchers to solve operation and planning problems such as unit commitment [11, 12], OPF [13, 14], security-constrained OPF [15, 16], security-constrained unit commitment [17, 18], generation planning [19, 20], transmission planning [21–23], and co-optimization of generation and transmission planning [24].

5.1.2 Literature Review

There are many approaches to solve the OPF problem. Some researchers have included several simplifications such as DC analysis, linear and continuous cost function, among other considerations. This has reduced the complexity of the problem to obtain a lower computation time when the OPF problem is applied to a large-scale power system.

Using DC network modeling assumptions and limiting polynomial costs to second order, the problem can be modeled using a quadratic cost function with

nodal power balance and transmission power flows as linear constraints. In order to transform the quadratic objective function, the researchers have considered linearizing the fuel cost generation using a linear function. In the technical literature, the linear fuel cost is applied to solve operation problems (unit commitment and OPF) and planning problems, while the quadratic objective function is used in OPF problems.

(a) **Mathematical formulation**

In this study, two mathematical formulations have been used to solve the DCOPF problem. The main difference of the formulations is the number of decision variables in the optimization problem.

- The first formulation is the classical DCOPF formulation which uses the power generation and the voltage phase angles as decision variables.
- The authors have achieved an efficient formulation using Y bus matrix and power transmission distribution factors (PTDF), so the network balance constraint is transformed in a global power balance equation, and the power flow on the transmission lines is a function of distribution factors and net power injection.

The DCOPF problem using the PTDF-based formulation solves the optimization problem considering only the active power generation as decision variable. The main advantage of this formulation is the reduction of the decision variables, so the algorithm can speed up the convergence process. This is a very important factor, because in each electricity control room (ISO), the OPF problem and/or approximation must be solved many times per day (for instance every 5 min).

An added benefit of using the distribution matrix would be loss estimation. The drawback of this approach is that the solution does not compute the voltage bus angles to draw some conclusions about the system, but using the nodal admittance matrix (Y bus) and the optimal power generation, it is possible to quickly calculate these angles.

(b) **Optimization algorithm**

Advances in algorithms are changing linear programming, and a new class of algorithm is emerging. Changes began in 1984 when Karmarkar published his paper [27] introducing interior-point methods to solve linear and nonlinear programming problems. In [28], the study compares the interior-point method with classic simplex method solved with *Minos*. The results in that study show the interior-point method is not as effective on smaller dimension problems, but as problem get larger, it becomes more effective than the simplex method. In this chapter, the DCOPF problem is carried out using an interior-point algorithm. The algorithm has no problems solving the quadratic objective function, and linear and nonlinear constraints.

(c) **Software**

The methodology has been implemented in DIgSILENT *PowerFactory* program using DIgSILENT Programming Language (DPL). The programming code was

validated with three test power systems. In order to evaluate the solution obtained by the algorithm, the results are modeled and validated with *MatPower*. The DCOPF computed by *MatPower* uses the classical formulation [29], and the interior-point algorithm is implemented in pure-MATLAB code [30].

Hinojosa and Moreno [31] perform a particle swarm algorithm to solve the DCOPF and an approximation of the ACOPF problem, so previous results are very encouraging considering that the *DIgSILENT PowerFactory* program can be used as a tool to carry out and develop some optimization algorithms. With this study, the authors add a new dimension to the *DIgSILENT PowerFactory* program by allowing the creation of mathematical optimization algorithms.

The chapter is organized as follows: Sect. 5.2.1 shows the classical DCOPF formulations used in the technical literature. Section 5.2.2 presents and introduces other formulation to solve the DCOPF problem based on the PTDF. Section 5.3.3 gives an overview of the algorithm and computational implementation issues. Section 5.5 provides numerical results of the algorithm applied to electrical test power systems. It is quite evident from the comparisons that the proposed methodology provides the better formulation. Finally, conclusions are derived in Sect. 5.6.

5.2 Optimal Power Flow Formulation

In the beginning, the optimization problem was formulated as a large-scale, non-convex, and nonlinear programming by Dommel and Tinney [32]. In the OPF problem, the control variables can be adjusted in pursuit of the optimal solution. These variables consist of voltage magnitudes at generator buses, transformer tap settings, and shunt devices switching. The dependent variables (depend on control variables) include voltage magnitudes in load buses, and voltage phase angles except the angle in the slack bus.

5.2.1 Classical DCOPF Formulation

A DC problem is an approximation for the alternating current network (AC). The classical DC optimization problem can be represented by the following formulation:

- (i) **Fuel cost of thermal units (objective function):** The total fuel cost of the units can be expressed in terms of the real power output.

$$Z = \min \sum_{i=1}^{N_g} C(P_i) \quad (5.1)$$

The fuel cost can be defined by $C(P_i) = a_i + b_i P_i + c_i P_i^2$; where a_i , b_i , and c_i are the fuel cost coefficients of the i -th thermal unit.

The optimization problem is subject to the following linear constraints:

- (ii) **Network balance:** There is a balance for the active power for each k -th bus of the transmission system (nodal power balance constraints).

$$P_k - P_k^d - B * \theta_k = 0 \quad (5.2)$$

where the voltage angle setting at reference bus is $\theta_{\text{ref}} = 0$.

- (iii) **Generation output limits:** For all generators, the active power output must be restricted within their lower and upper limits.

$$P_i^{\min} \leq P_i \leq P_i^{\max} \quad (5.3)$$

- (iv) **Transmission limits:** The power flow through the transmission line must be limited within its capacity limits (thermal limit constraints).

$$-f_j^{\max} \leq \frac{\theta_i - \theta_j}{x_{ij}} \leq f_j^{\max} \quad (5.4)$$

The decision variables in this optimization problem are the voltage angles and the active power generation. In the above formulation, the voltage bus angles could be constrained.

5.2.2 DCOPT Formulation Using the Power Transmission Distribution Factors (PTDF)

In this case, the objective function and the generation output constraint are the same. In addition, the network balance constraint and the transmission limits can be represented by the following formulation:

- (i) **Network balance:** The power balance constraint can be transformed to a single expression. In economic dispatch formulation, this constraint is called global power balance equation. Note that Eq. (5.5) does not depend on voltage bus angles.

$$\sum_{i=1}^{N_g} P_i - \sum_{k=1}^{N_b} P_k^d = \sum_{i=1}^{N_g} P_i - D = 0 \quad (5.5)$$

- (ii) **Transmission limits:** The DC power flow model can be used to compute the sensitivities of branch flows with respect to changes in nodal real power injections. In the technical literature, these factors are called *injection shift*

factors, generation shift factors, or PTDF [10]. These $n_{ij} \times n_k$ sensitivity matrices carry an implicit assumption about the slack distribution. The PTDF matrix can be calculated using the Eq. (5.6):

$$PTDF = \text{diag}(B) * A_r * (A_r^T * \text{diag}(B) * A_r)^{-1} \quad (5.6)$$

The power flow through the transmission lines can be computed using the PTDF so that the power flow constraints are a function of the PTDF matrix and the net power injection.

$$-F^{\max} \leq PTDF * (P - P^d) \leq F^{\max} \quad (5.7)$$

The decision variables in this optimization problem are the active power generation supplied by each unit.

5.3 Primal-dual Interior-point Solver

In the past decade, primal-dual algorithms have emerged as the most important and useful algorithm from the interior-point class. The interior-point method has become more and more popular in the OPF field because of its excellent computational performances. The theoretical foundations for interior-point methods consist of three crucial building blocks: (1) Isaac Newton's method for solving nonlinear equations, and hence for unconstrained optimization; (2) Joseph Lagrange's methods for optimization with equality constraints; and (3) Anthony Fiacco and Garth McCormick's barrier method for optimization with inequality constraints.

The general nonlinear optimization problem can be represented as follows:

$$\min_X f(X) \quad (5.8)$$

Subject to:

$$\begin{aligned} G(X) &= \mathbf{0} \\ H(X) &\leq \mathbf{0} \\ X_{\min} &\leq X \leq X_{\max} \end{aligned} \quad (5.9)$$

where linear and nonlinear constraints are included in $G(X)$ and $H(X)$ constraints.

5.3.1 First Order Optimality Conditions

Firstly, slack variables Z are added to inequality constraints to convert them to equality constraints. The nonnegative constraints on slack variables are eliminated by placing them in a barrier objective function using the perturbation parameter γ , which approaches to zero when the solution is accomplished by the iterative algorithm.

The first order optimality (Karush–Kuhn–Tucker) conditions for this problem are satisfied when the partial derivatives of the Lagrangian (L) above are set to zero.

$$\begin{bmatrix} L_X^T \\ L_Z^T \\ L_\lambda^T \\ L_\mu^T \end{bmatrix} = \begin{bmatrix} f_X^T + G_X^T \lambda + H_X^T \mu \\ [\mu]Z - \gamma e \\ G(X) \\ H(X) + Z \end{bmatrix} = 0 \quad (5.10)$$

5.3.2 Newton Step

The first order optimality conditions are solved using Newton's method.

$$\begin{bmatrix} L_{XX} & 0 & G_X^T & H_X^T \\ 0 & [\mu] & 0 & [Z] \\ G_X & 0 & 0 & 0 \\ H_X & I & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta X \\ \Delta Z \\ \Delta \lambda \\ \Delta \mu \end{bmatrix} = - \begin{bmatrix} L_X^T \\ [\mu]Z - \gamma e \\ G(X) \\ H(X) + Z \end{bmatrix} \quad (5.11)$$

This set of equations can be simplified and reduced to a smaller set of equations [29] by solving explicitly for $\Delta\mu$ in terms of ΔZ and for ΔZ in terms of ΔX , so the matrix solution can be transformed into the following system:

$$\underbrace{\begin{bmatrix} M & G_X^T \\ G_X & 0 \end{bmatrix}}_K \begin{bmatrix} \Delta X \\ \Delta \lambda \end{bmatrix} = \begin{bmatrix} -N \\ -G(X) \end{bmatrix} \quad (5.12)$$

Defining the matrix M and N as follows:

$$M \equiv L_{XX} + H_X^T [Z]^{-1} [\mu] H_X \quad (5.13)$$

$$N \equiv L_X^T + H_X^T [Z]^{-1} (\gamma e - [\mu] H(X)) \quad (5.14)$$

5.3.3 Primal-dual Interior-point Algorithm

The following pseudo-code has been included to summarize the basic steps of the algorithm:

Algorithm: General primal-dual interior point method

1: Set the initial point (X^0, λ^0, Z^0 and μ^0)

2: Set $iter := 0$

3: **repeat**

4: Set

$$\gamma^{iter} \rightarrow 0.1 * \frac{Z^T \mu}{n_i}$$

5: Solve system (5.12), (5.13) and (5.14) to compute $\Delta X^{iter}, \Delta \lambda^{iter}, \Delta Z^{iter}$ and $\Delta \mu^{iter}$

6: Set

$$X^{iter+1} \rightarrow X^{iter} + \alpha_p \Delta X$$

$$Z^{iter+1} \rightarrow Z^{iter} + \alpha_p \Delta Z$$

$$\lambda^{iter+1} \rightarrow \lambda^{iter} + \alpha_d \Delta \lambda$$

$$\mu^{iter+1} \rightarrow \mu^{iter} + \alpha_d \Delta \mu$$

choosing

$$\alpha_p = \min \left(\xi * \min \left(-\frac{Z_m}{\Delta Z_m} \right), 1 \right)$$

$$\alpha_d = \min \left(\xi * \min \left(-\frac{\mu_m}{\Delta \mu_m} \right), 1 \right)$$

7: Set $iter := iter + 1$

8: **until** Convergence

*The parameter ξ is a constant scalar with a value slightly less than one. The scalar is a safety factor to guarantee that the next point meets the strict positivity conditions with typical value of 0.99995.

5.3.4 Analysis of the Problem Dimension

In order to compute the decision variables (ΔX) and the Lagrangian multipliers ($\Delta \lambda$), the inverse of the matrix K (5.12) must be solved, so the complexity of the problem depends on the dimension of this matrix.

- (1) **Classical formulation (M1):** This problem considers as decision variables voltage angles and active power generation so that the number of decision variables is $n = N_b + N_G$. In addition, the number of equality constraints is $\Phi_e = N_b + N_L + N_G$, and the number of inequality constraints is $\Phi_i = 2N_L + 2N_G$.
For the classical formulation, the dimension of the square K matrix is $\dim(K) = (N_b + N_G) + (N_b + 1) = 2N_b + N_G + 1$.
- (2) **DCOPF using the PTDF-based formulation (M2):** This problem considers as decision variables only active power generation supplied by each unit so that the number of decision variables is $n = N_G$. The number of equality constraints is $\Phi_e = 1$, and the number of inequality constraints is $\Phi_i = 2N_L + 2N_G$. The dimension of the K matrix is $\dim(K) = N_G + 1$, so there are $2N_b$ variables lower than the classical formulation. The authors recommend its application to a medium- and large-scale power system in order to obtain the lower simulation time.

5.4 DIgSILENT Programming Language

The DPL offers an interface to the user for the automation of tasks in the *Power-Factory* program. The main idea of this interface is that the user can define its own automation commands (or scripts) develop some programs and new calculation functions. The DPL method distinguishes itself from the command batch method in several aspects. DPL offers:

- (i) Decision and flow commands,
- (ii) Definition and use of user-defined variables,
- (iii) A flexible interface for input–output and for accessing objects,
- (iv) Mathematical expressions.

The DPL command object (*ComDpl*) is the central element which is connecting different parameters, variables, or objects to various functions or internal elements and then puts out results or changes parameters. DPL command objects provide an interface for the configuration, preparation, and use of DPL scripts. Thus, the DPL script will run a series of operation and start calculation or other function inside the script (subroutines).

5.4.1 Creating a New DPL Command

A DPL Command *ComDpl* can be created by using the “New Object” button in the toolbar of the data manager. When a new DPL command is created, the dialogue is shown and the parameters, objects, and script can be specified. A DPL example called *DCOPF_PTFD.ComDpl* is shown in Fig. 5.1. This DPL command performs a DC optimal power flow using the PTDF-formulation. It can be seen on the *Input Parameters* section, the slack bus, the nominal power system data, and the parameters for cost generation system, among other parameters. The variables *slack*, *Sbase*, *Vbase*, etc. are used in the DPL programming code. In addition to the input parameters section, external, and internal objects can be used and executed.

5.4.2 Interior-point Flowchart

In Fig. 5.2, it is included a flowchart which shows the main stages of the primal-dual interior-point algorithm for solving the DC optimal power flow.

Basic Options | Advanced Options | Script | Description | Version

Name

DCOPF_PTFD

General Selection

▼ | ► | ...

Input parameters:

	Type	Name	Value	Unit	Description
▶ 1	int	slack	1		Slack bus
2	double	Sbase	100		MVA_Base
3	double	Vbase	230		voltage base
4	double	C2_g1	0.11		a_G1
5	double	C2_g2	0.085		a_G2
6	double	C2_g3	0.1225		a_G3
7	double	C1_g1	5		b_G1
8	double	C1_g2	1.2		b_G2
9	double	C1_g3	1		b_G3
10	double	C0_g1	150		c_G1
11	double	C0_g2	600		c_G2
12	double	C0_g3	335		c_G3

External Objects:

	Name	object	Description
▶ 1			

Execute

Close

Cancel

Save

Check

Contents

Fig. 5.1 DPL command

Each process of the flowchart is explained in the following steps:

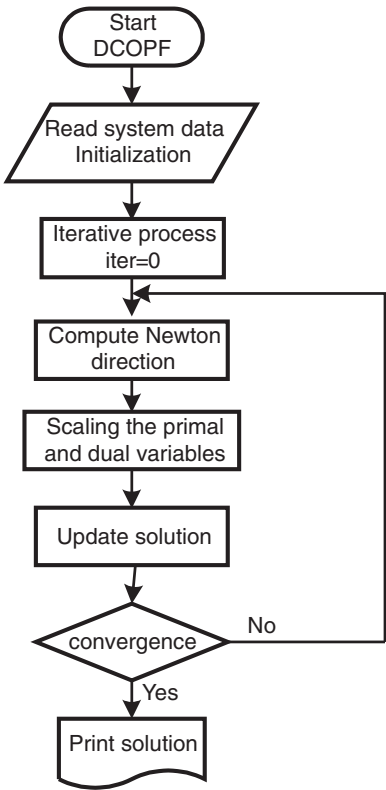
Step 1: *Script Section*

DPL scripts are written by the user through the DPL editor which resides inside the DPL command object. The DPL script language uses syntax quite similar to the programming language C++ [33]. The syntax can be divided into the following parts:

- (i) Variable definitions,
- (ii) Assignments and expressions, and,
- (iii) Program flow instructions.

Before starting the simulation process, variables, sets, and objects must be defined. In addition, DPL commands can call other DPL commands as subroutines in order to optimize the programming algorithm. The DPL contents are used to include subroutines, matrix objects, and filter sets. In Fig. 5.3 are shown the following elements: (1) objects and sets used by the algorithm to handle the power system elements (buses, power units, loads, transmission lines, and generator cost data); (2) variable definitions of the interior-point algorithm; and (3) subroutines, matrix objects, and filters used by the main script (DCOPF_PTDF.ComDpl).

Fig. 5.2 Interior-point scheme



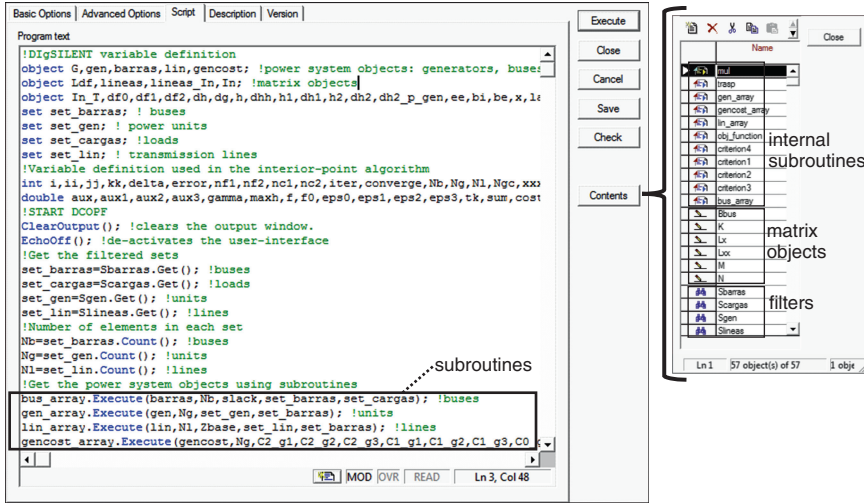


Fig. 5.3 Variables and elements used in the algorithm

The power system objects (*gen*, *barras*, *lin*, and *gencost*) are defined in the second line of Fig. 5.3, and the data are saved in objects using the subroutines: *bus_array.ComDpl*, *gen_array.ComDpl*, *lin_array.ComDpl*, and *gencost_array.ComDpl*.

The authors use filter sets (*SetFilt*) to obtain the electrical parameters of the systems: buses (*Sbarras.ElmTerm*), loads (*Scargas.ElmLod*), power units (*Sgen.ElmSym*), and transmission lines (*Slneas.ElmLne*). Moreover, the vector and matrix objects (*Bbus.IntMat*, *K.IntMat*, etc.) that the DPL command uses for computing the iterative algorithm (interior-point algorithm) are shown in Fig. 5.3.

Step 2: Computing Newton directions

A short script is shown in Fig. 5.4 in which it is possible to see the construction of the matrixes components (\mathbf{M} , \mathbf{G}_X^T , \mathbf{G}_X), the inverse command, the decision variables (power generation supplied by each unit), and the Lagrangian multiplier (called marginal cost in the PTDF-based formulation). An error message is used in the programming code to verify the singularity of the matrix \mathbf{K} .

The statements in a DPL script are separated by semicolons and are grouped together by braces.

Step 3: Finding step lengths

The search direction may not be a good direction as it regularly happens that only a small step can be taken along this direction before the positivity constraints ($Z > 0$) are violated. In order to maintain strict feasibility of the trial solution, the algorithm truncates the Newton step by scaling the primal and dual variables using α_p and α_d , respectively. The computation of the primal scaling factor (α_p) is shown in Fig. 5.5.

```

for (ii=1; ii<=nf1; ii+=1) { !M matrix
    for (jj=1; jj<=nc1; jj+=1) {
        aux=M.Get(ii,jj);
        K.Set(ii,jj,aux);
    }
}
for (ii=1; ii<=nf2; ii+=1) { !Gx matrix
    for (jj=1; jj<=nc2; jj+=1) {
        aux=dg.Get(ii,jj);
        K.Set(Ng+ii,jj,aux);
    }
}
for (ii=1; ii<=nc2; ii+=1) { !Gx^T matrix
    for (jj=1; jj<=nf2; jj+=1) {
        aux=dg_T.Get(ii,jj);
        K.Set(ii,Ng+jj,aux);
    }
}
error=K.Invert(); !Inverse of the K matrix
if (error) !error message
    Error('The matrix has not inverse.');
```

mul.Execute(K,nf1+nf2,nc1+nf2,bb,nc1,nf2,dx_L); !solution of (5.12)

```

for (ii=1;ii<=Ng;ii+=1) { !Get the decision variables (power generation)
    aux=dx_L.Get(ii,1);
    dx.Set(ii,1,aux); !delta_x
}
aux=dx_L.Get(Ng+1,1); !Get the Lagrangian multiplier (marginal cost)
dlam.Set(1,1,aux); !delta_lambda
```

Fig. 5.4 Solution of the system using Newton's method

Fig. 5.5 Computation of the primal and dual scaling factors

```

! alpha_p
jj=0;
for (ii=1; ii<=2*(Nl+Ng); ii+=1) {
    aux=dz.Get(ii,1);
    if ({aux<0} .and. {ii>0}) {
        jj+=1;
        aux1=z.Get(ii,1);
        aux2=-(aux1/aux);
        alfaz_p.Set(jj,1,aux2);
    }
}
aux=1e3;
xxx=alfaz_p.SizeX();
for (ii=1; ii<=xxx; ii+=1) {
    aux1=alfaz_p.Get(ii,1);
    if (aux1<aux)
        aux=aux1;
}
if (aux<1)
    alfa_p=aux;
else
    alfa_p=1;
```

Step 4: *Updating solution*

Once the Newton step has been computed using the matrixes ΔX , ΔZ , $\Delta \lambda$, and $\Delta \mu$, the variables x (decision variables), z (slack variables), lam (marginal cost), and u (Lagrangian multipliers associated with inequality constraints) must be iteratively updated. Figure 5.6 shows the updating process.

Fig. 5.6 Updating process of the interior-point variables

```
!Updating process: 1) decision variables (X)
for (ii=1; ii<=Ng; ii+=1) {
    aux=x.Get(ii,1);
    aux1=dx.Get(ii,1);
    aux2=aux+alfa_p*aux1;
    x.Set(ii,1,aux2);
}
! 2) slack variables (Z)
for (ii=1; ii<=2*(Nl+Ng); ii+=1) {
    aux=z.Get(ii,1);
    aux1=dz.Get(ii,1);
    aux2=aux+alfa_p*aux1;
    z.Set(ii,1,aux2);
}
! 3) Marginal cost (lambda)
for (ii=1; ii<=1; ii+=1) {
    aux=lam.Get(ii,1);
    aux1=dlam.Get(ii,1);
    aux2=aux+alfa_d*aux1;
    lam.Set(ii,1,aux2);
}
! 4) Lagrangian multipliers
for (ii=1; ii<=2*(Nl+Ng); ii+=1) {
    aux=u.Get(ii,1);
    aux1=du.Get(ii,1);
    aux2=aux+alfa_d*aux1;
    u.Set(ii,1,aux2);
}
```

Step 5: *Checking convergence*

In Fig. 5.7 is shown the convergence criteria used by the algorithm: (a) feasibility condition (*feascond*), gradient condition (*gradcond*), complementary condition (*compcnd*), and cost condition (*costcond*). The algorithm solution converges when these conditions are lower than 10^{-6} (*epso0*, *epso1*, *epso2*, and *epso3*). Finally, when the conditions are accomplished, the script prints the optimal solution and a DC power flow is executed to display the result in the power system grid.

The DPL command (DCOPF_PTDF.ComDpl) is executed in order to solve the DC-OPF problem using the PTDF-formulation. When the DPL script is successfully executed, the output window of DlgSILENT *PowerFactory* prints different messages of the interior-point algorithm: number of iterations, objective function (fuel cost), and convergence criteria in order to give information about the iterative process. The results of the iterative process are shown in Fig. 5.8.

5.5 Illustrative Examples and Simulation Results

Three power systems are considered to inspect and verify the proposed formulation. Several experiments have been conducted with different constraint in order to validate and determine the best performance of the mathematical formulation.

```
! Check Tolerance criteria
criterion1.Execute (g,h,x,z,feascond);
criterion2.Execute (Lx,lam,z,u,gradcond);
criterion3.Execute (z_T_u,normax,compcond);
criterion4.Execute (f,f0,costcond);
! Printing information
printf_info=print;
if (printf_info=1)
    printf('%3d %12.3f %12.4g %12.4g %12.4g
%12.4g',iter,f,feascond,gradcond,compcond, costcond);
! Convergence criteria
if({feascond<eps0}.and.{gradcond<eps1}.and.{compcond<eps2} and.{costcond<eps3}) {
    converge = 1;
    printf('Converged! \nDC-OPF solution:');
    ii=1;
    for (G=set_gen.First(); G>0; G=set_gen.Next()) {
        aux=x.Get(ii,1);
        G.pgini=Sbase*aux;
        printf('%s = %.4f [MW]',G:loc_name,Sbase*aux);
        ii+=1;
    }
    Ldf=GetCaseObject('ComLdf');
    Ldf:iopt_net=2; !DC PF command
    Ldf.Execute(); !DC power flow solution
}
else {
    f0 = f;
    if (iter>1000)
        Error('No converged!');
}
```

Fig. 5.7 Check convergence process

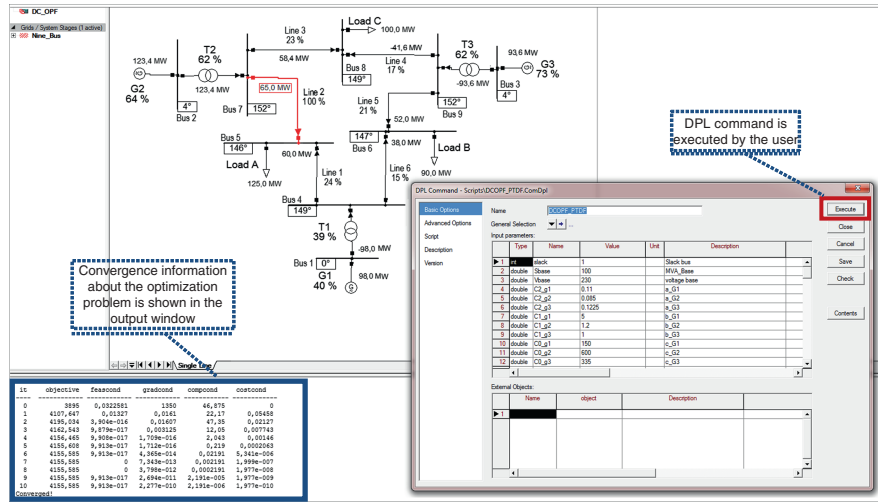


Fig. 5.8 Execution process example

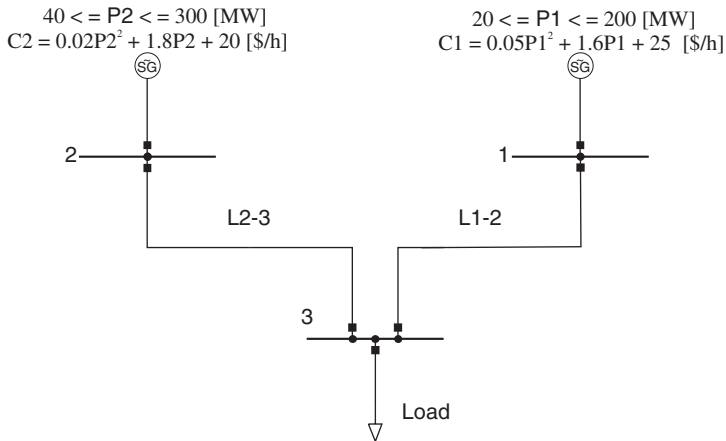


Fig. 5.9 Test system 1: 3-bus electrical system example

5.5.1 DCOPF Formulation Using the Power Transmission Distribution Factors

The first test system used for validation purposes has three buses and two transmission lines. In Fig. 5.9, the single line diagram of the test system is provided, and the load (350 MW) and the generation system characteristic are included in the diagram.

In this first example, the reactance for each transmission line is 11.132 and 20.57 Ω , respectively. Both transmission lines have a capacity of 200 MW. The optimization algorithm converged in 9 iterations (*feascond* = 0.00, *gradcond* = 1.37×10^{-16} , *compcond* = 2.86×10^{-07} , and *costcond* = 3.82×10^{-10}). The convergence criteria for the conditions are 10^{-6} .

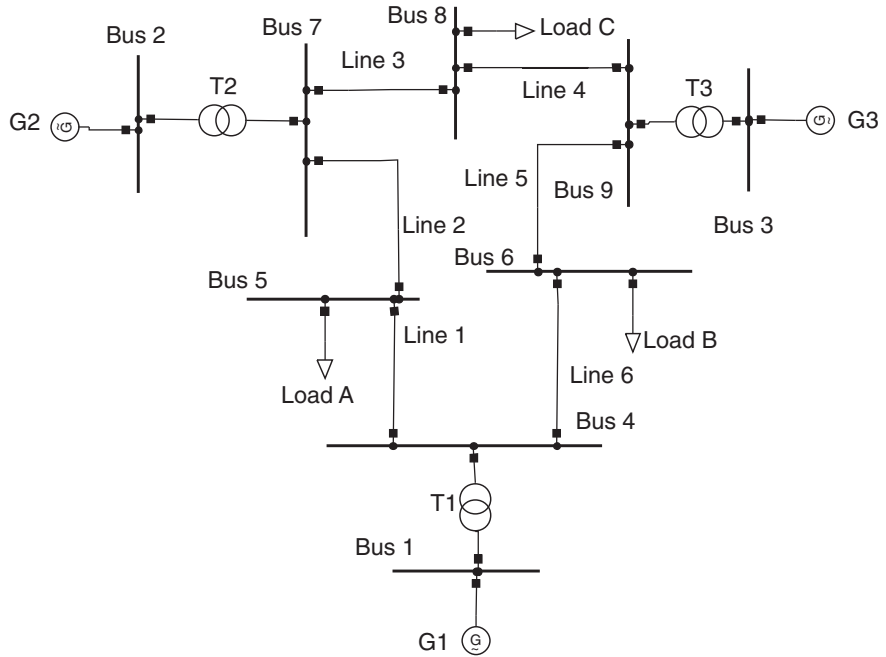
The optimal cost of both solutions is 2,570 \$/h, and the power generation is $P_1 = 150$ and $P_2 = 200$ MW. The results are also compared with the interior-point algorithm used by *MatPower*, and both solutions (M1 and M2) are the same.

With this example, it is possible to test and validate the DCOPF solution based on the PTDF-formulation.

5.5.2 PTDF-based Formulation Applied to IEEE Systems

In this section, two power systems test are modeled to verify the application of the PTDF-based formulation:

1. **DIgSILENT 9-bus system:** This test system consists of three generators and nine transmission lines. The system is packaged with *DIgSILENT PowerFactory*. In Fig. 5.10, the electrical transmission network is shown.



UTFSM V.H. Hinojosa PowerFactory 15.0.5	Nine-Bus System	Project: DCOPF
	DC optimal power flow solution using a PTDF-based formulation	Graphic: Single Line
		Date: 8/9/2014 Annex: Nine Bus

Fig. 5.10 Test system 2: DiGSILENT 9-bus transmission network

The transmission network data can be obtained from DiGSILENT *PowerFactory*. The technical parameters for the generation system are given in Table 5.1.

- (i) Formulation M1 and M2 were programmed using DPL. The PTDF-based formulation converges in 7 iterations ($feascond = 1.017 \times 10^{-16}$, $gradcond = 1.891 \times 10^{-16}$, $compcnd = 9.943 \times 10^{-06}$, and $costcond = 2.201 \times 10^{-16}$). The classical DCOPF formulation also converges in 7 iterations. The dimension of the matrix K is 4 using the PTDF-based formulation, and 22 using the classical formulation. Therefore, there is an important reduction (5 times approximately) in the dimension of the problem. The optimal cost is 4,131.0266 \$/h using both formulations (M1 and M2). The power generation is $P_1 = 86.565$ MW, $P_2 = 134.378$ MW, and $P_3 = 94.058$ MW. The results are compared with *MatPower* which use the classical formulation (M1), and the optimal solution is the same so that both formulations are equivalent. Tables 5.2 and 5.3 show the power flow solution: the voltage angles for each bus, the marginal cost for the slack bus (λ), and transmission power flows.

Table 5.1 Generation system data for the 9-bus system

Parameter	G_1	G_2	G_3
a_i , \$/MW ² h	0.11	5	150
b_i , \$/MWh	0.085	1.2	600
c_i , \$/h	0.1225	1	335
P_{\min} , MW	10	10	10
P_{\max} , MW	250	300	270

Table 5.2 Power flow solution for the 9-bus system

Bus #	Voltage		G_i	Load	λ
	Magnitude	Angle	P (MW)	P (MW)	(\$/MWh)
1	1.0	0.000	86.565	–	24.044
2	1.0	6.040	134.378	–	–
3	1.0	4.003	94.058	–	–
4	1.0	–2.857	–	–	–
5	1.0	–5.430	–	125	–
6	1.0	–4.635	–	90	–
7	1.0	1.228	–	–	–
8	1.0	–1.338	–	100	–
9	1.0	0.845	–	–	–

Table 5.3 Power flows on the transmission lines

Line	From	To	P (MW)
1	5	4	–52.83
2	5	7	–72.17
3	7	8	62.20
4	8	9	–37.80
5	9	6	56.26
6	6	4	–33.74
7	4	1	–86.56
8	7	2	–134.38
9	9	3	–94.06

In this case, there is not congestion in the transmission system based on the rated current of each line.

- (ii) In order to analyze the performance of the algorithm with respect to the maximum power flow, the transmission line 7-2 has been limited with a current of 0.163 kA; the rated current of that line is 0.627 kA (base case). The DCOPF problem is solved, and the optimal cost is 4,155.5845 \$/h using both formulations. The power generation is $P_1 = 97.992$ MW, $P_2 = 123.443$ MW, and $P_3 = 93.564$ MW.

Figure 5.11 shows the optimal solution. The red color shows that the transmission line 7-2 is congested. The additional cost (24.5549 \$/h) occurs because the network’s transmission system is limited (cost of congestion) so

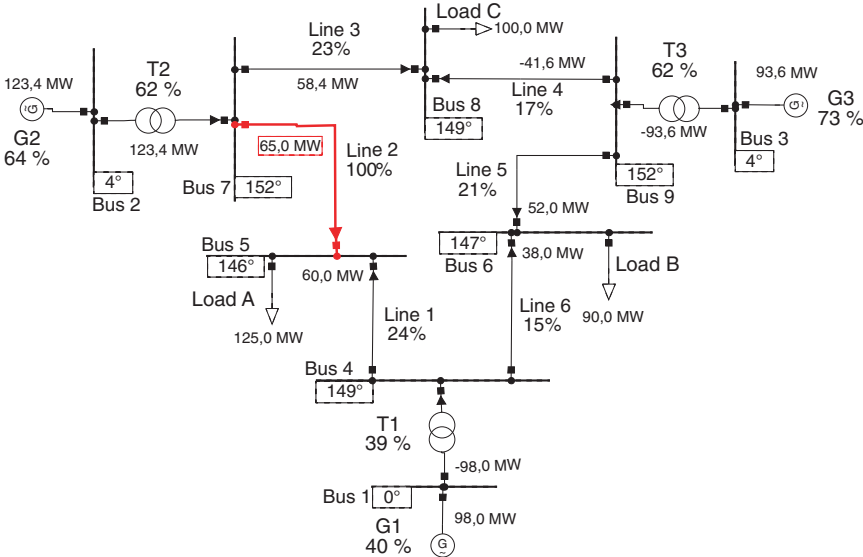


Fig. 5.11 DCOPF solution for the DiGSILENT 9-bus system

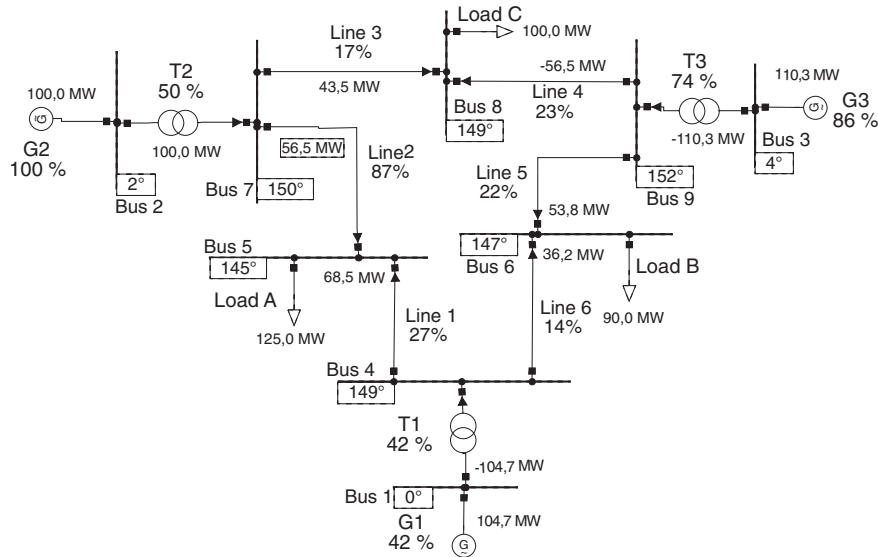


Fig. 5.12 DCOPF solution considering the maximum power in unit 2

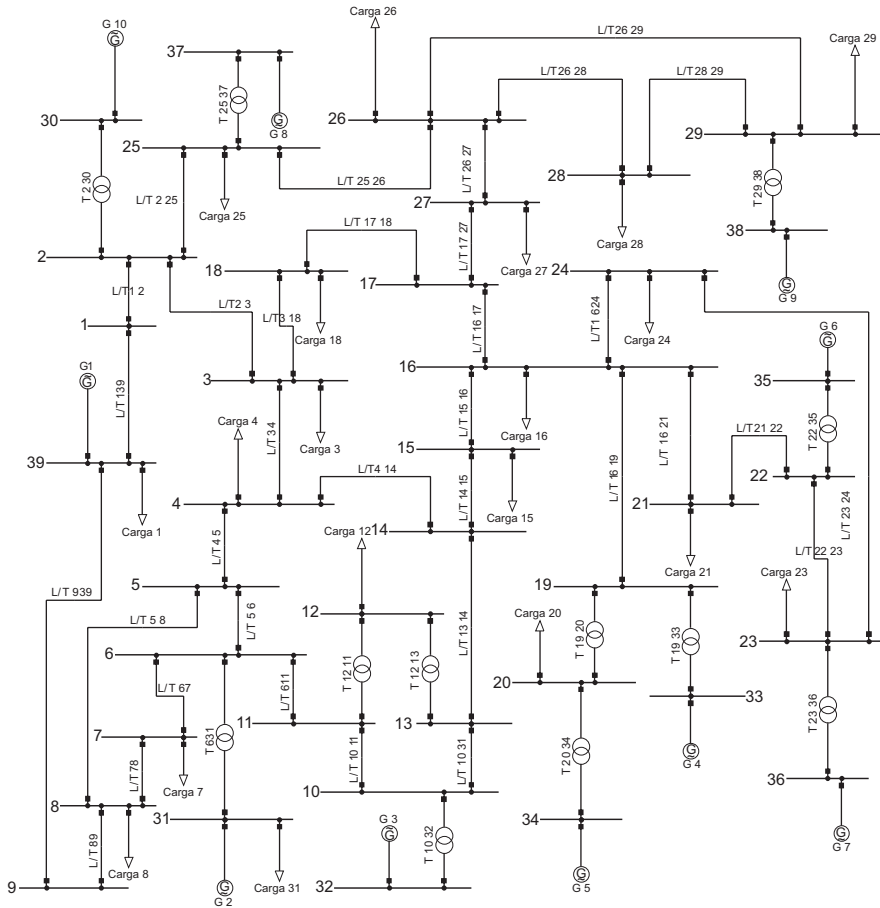


Fig. 5.13 Test system 3: New England transmission network

that it is necessary to use more expensive generation (G_2 and G_3) to supply the load of the customers.

- (iii) The authors have modeled other case where the maximum power of unit 2 has been limited to 100 MW. The DCOPF problem is solved, and the optimal cost is 4,299.9758 \$/h. The power generation is $P_1 = 104.677$ MW, $P_2 = 100.0$ MW, and $P_3 = 110.323$ MW. Figure 5.12 shows the optimal solution. The results obtained by *DIgSILENT PowerFactory* and *MatPower* are the same (cost, power generation level, and transmission power flows). In the power flow solution, it can be seen that the active power of unit 2 (G_2) is constrained. In this operation case, the power flow through the transmission line 2-7 decreases, so the loading is 87 %.

Classical and PTDF-based formulations work very well considering both transmission and generation inequality constraints.

2. IEEE 39-bus system: The second test system consists of ten generators and forty-six transmission lines. The system is packaged with *MatPower*. In Fig. 5.13, the electrical transmission network modeled in *DIgSILENT PowerFactory* is shown. The technical parameters for the generation system and the transmission network are obtained from *MatPower*.

The optimal cost using both formulations is 41,370.515 \$/h, and the power generation in each bus is $P_{30} = 300.0$ MW, $P_{31} = 646.0$ MW, $P_{32} = 700.0$ MW, $P_{33} = 652.0$ MW, $P_{34} = 508.0$ MW, $P_{35} = 687.0$ MW, $P_{36} = 580.0$ MW, $P_{37} = 564.0$ MW, $P_{38} = 756.77$ MW, and $P_{39} = 756.77$ MW. Both formulations converge in 17 iterations using *DIgSILENT PowerFactory*, but the dimension of the K matrix using the PTDF-based formulation is lower than the classical formulation. The results are also compared with *Matpower*, and it is verified that both solutions are the same.

5.6 Conclusions

This chapter has presented an application of two different formulations to figure out the DC optimal power flow problem. The formulations have been implemented and validated with three test power systems using an interior-point solver. The interior-point algorithm has been used due to its excellent computational performance, and the algorithm has been totally programmed using DPL. In this study, it is proposed an efficient formulation using the PTDF. In the PTDF-based formulation, the network balance constraints are converted in a global power balance equation, and the transmission constraints are a function of the PTDF and the net power injection. Several experiments were conducted to determine the performance of the proposed methodology. The main advantage of this formulation is the lower complexity in the optimization problem so that the algorithm can speed up the convergence process. Finally, the DPL adds a new dimension to the *DIgSILENT Power Factory* program by allowing the creation of optimization algorithms to solve operation and planning problems. Nowadays, the authors are implementing and including some constraints in order to model and simulate the transmission expansion planning problem in the *DIgSILENT Power Factory* program.

Acknowledgments This study was supported in part by the Chilean National Commission for Scientific and Technological Research (CONICYT) under grant Fondecyt 1130793 and Basal FB0008, and by the Universidad Técnica Federico Santa María, Chile, under project No. USM 22.13.42.

References

1. Carpentier J (1962) Contribution a l'Etude du dispatching Economique. *Bull Soc Fr Electr* 8:431–447
2. Mohapatra A, Bijwe PR, Panigrahi BK (2013) Optimal power flow with multiple data uncertainties. *Electr Power Syst Res* 95:160–167
3. Khorsandi A, Hosseinian SH, Ghazanfari A (2013) Modified artificial bee colony algorithm based on fuzzy multi-objective technique for optimal power flow problem. *Electr Power Syst Res* 95:206–213
4. Hinojosa VH, Araya R (2013) Modeling a mixed-integer-binary small population evolutionary particle swarm algorithm for solving the optimal power flow problem in electric power systems. *Appl Soft Comput* 13:3839–3852
5. Momoh J, Adapa R, El-Hawary M (1993) A review of selected optimal power flow literature to 1993 i. nonlinear and quadratic programming approaches. *IEEE Trans Power Syst* 14:96–104
6. Pizano A, Fuerte CR, Ruiz D (2011) A new practical approach to transient stability-constrained optimal power flow. *IEEE Trans Power Syst* 26(3):1686–1696
7. Momoh JA, Koessler RJ, Bond MS, Stott B, Sun D, Papalexopoulos AD, Ristanovic P (1997) Challenges to optimal power flow. *IEEE Trans Power Syst* 12(1):444–455
8. Torelli F, Vaccaro A, Xie N (2013) A novel optimal power flow formulation based on the Lyanupunov theory. *IEEE Trans Power Syst* 28(4):4405–4415
9. Cain MB, O'Neill P, Castillo A (2012) History of optimal power flow and formulations. FERC staff technical paper, Dec 2012
10. Wood AJ, Wollenberg BF (1996) Power generation, operation and control. Wiley, New York
11. Fotuhi-Firuzabad M, Billinton R, Khan ME (1999) Extending unit commitment health analysis to include transmission considerations. *Electr Power Syst Res* 50(1):35–42
12. Lee FN, Huang J, Adapa R (1994) Multi-area unit commitment via sequential method and a dc power flow network model. *IEEE Trans Power Syst* 9(1):279–287
13. Yu J, Yan W, Li W, Wen L (2008) Quadratic models of AC-DC power flow and optimal reactive power flow with HVDC and UPFC controls. *Electr Power Syst Res* 78(3):302–310
14. Biskas PN, Bakirtzis AG, Macheras NI, Pasialis NK (2005) A decentralized implementation of dc optimal power flow on a network computers. *IEEE Trans Power Syst* 20(1):25–33
15. Amjady N, Sharifzadeh H (2011) Security constrained optimal power flow considering detailed generator model by a new robust differential evolution algorithm. *Electr Power Syst Res* 81(2):740–749
16. Ardakani AJ, Bouffard F (2013) Identification of umbrella constraints in dc-based security-constrained optimal power flow. *IEEE Trans Power Syst* 28(4):3924–3934
17. Lei X, Lerch E, Xie CY (2002) Frequency security constrained short-term unit commitment. *Electr Power Syst Res* 60(3):193–200
18. Kalantari A, Restrepo JF, Galiana FD (2013) Security-constrained unit commitment with uncertain wind generation: the loadability set approach. *IEEE Trans Power Syst* 28(2):1787–1796
19. Tohidi Y, Aminifar F, Fotuhi-Firuzabad M (2013) Generation expansion and retirement planning based on the stochastic programming. *Electr Power Syst Res* 104:138–145
20. Dehghan S, Amjady N, Kazemi A (2014) Two-stage robust generation expansion planning: a mixed integer linear programming model. *IEEE Trans Power Syst* 29(2):584–597
21. Vinasco G, Tejada D, Silva EFD, Rider MJ (2014) Transmission network expansion planning for the colombian electrical system: Connecting the Ituango hydroelectric power plant. *Electr Power Syst Res* 110:94–103
22. Hesamzadeh MR, Yazdani M (2014) Transmission capacity expansion in imperfectly competitive power markets. *IEEE Trans Power Syst* 29(1):62–71

23. Hinojosa VH, Galleguillos N, Nuques B (2013) A simulated rebounding algorithm applied to the multi-stage security-constrained transmission expansion planning in power systems. *Int J Elect Power Energy Syst* 47:168–180
24. Khodaei A, Shahidehpour M (2013) Microgrid-based co-optimization of generation and transmission planning in power systems. *IEEE Trans Power Syst* 28(2):1582–1590
25. Dommel H, Tinny W (1968) Optimal power flow solutions. *IEEE Trans Power Appar Syst* PAS-87(10):1866–1876
26. PSERC (2011) Software matpower. <http://www.pserc.cornell.edu/matpower/i>. Accessed 12 Dec 2011
27. Karmarkar N (1984) A new polynomial-time algorithm for linear programming. *Combinatoria* 4(8):373–395
28. Astfalk G, Lustig I, Marsten R, Shanno D (1992) The interior-point method for linear programming. *IEEE Softw* 9(4):61–68
29. Zimmerman RD, Murillo-Sanchez CE, Thomas RJ (2011) Matpower: steady-state operations, planning, and analysis tools for power systems research and education. *IEEE Trans Power Syst* 26:12–19
30. <http://www.pserc.cornell.edu/matpower/>
31. Hinojosa VH, Moreno A (2008) Optimal power flow solution using evolutionary algorithms—a DigSILENT application. In: *Proceedings of Ecuaciel—XXII power systems conference*, Ecuador, 2008 (in Spanish)
32. Dommel H, Tinny W (1968) Optimal power flow solutions. *IEEE Trans Power Appar Syst* PAS-87:1866–1876
33. DigSILENT power factory (2010) Power factory user’s manual—version 14.0, DigSILENT GmbH, 2010

Chapter 6

Assessing the Renewable Energy Sources Integration Through a Series of Technical Performance Indices Using DIgSILENT PowerFactory DPL

A.I. Nikolaidis, Francisco M. Gonzalez-Longatt
and C.A. Charalambous

Abstract The increasing penetration of intermittent energy sources in existing power systems heavily impacts the planning and operational processes of system operators. Therefore, refined approaches are required to address the techno-economic issues introduced by the stochastic nature of renewable energy sources, in an attempt to harness the maximum benefit possible. To address these challenges, this work capitalizes on a set of technical performance indices. These indices are attempting to quantify some operational impacts by providing an operational and/or long-term planning tool for assessing the renewable energy penetration in transmission networks. The modeling and simulation case studies are facilitated within the powerful grid simulator DIgSILENT which enables for fast and reliable execution of a wide spectrum of computing procedures. In particular, this chapter showcases how DIgSILENT's programming language (DPL) is utilized to create several automated scenarios that may explore potential topologies of a test system as per the systems' operators and planners' possible interests.

Keywords Renewable energy • DPL • Technical performance indices • Stochastic power flow

Electronic supplementary material The online version of this chapter (doi:10.1007/978-3-319-12958-7_6) contains supplementary material, which is available to authorized users.

A.I. Nikolaidis (✉) · C.A. Charalambous
Department of Electrical and Computer Engineering, University of Cyprus,
75 Kallipoleos Avenue, P.O. Box 20537, 1678 Nicosia, Cyprus
e-mail: nikolaidis.alexandros@ucy.ac.cy

F.M. Gonzalez-Longatt
School of Electronic, Electrical and Systems Engineering, Loughborough University,
LE11 3TU, Loughborough, UK
e-mail: fglongatt@fglongatt.org

6.1 Introduction

Renewable energy sources (RES) are gradually becoming an integral part of the energy mix in the vast majority of countries around the world. They constitute a key element in the efforts of dealing with the finite nature of fossil fuels. Concurrently, they are progressively replacing CO₂-emitting technologies, thus contributing to the decarbonization of future power systems.

To this end, it is noted that wind and solar penetration is continuously increasing at a faster rate than any other electricity generation technology. Total renewable power capacity worldwide exceeded 1,470 GW in 2012. Globally, wind power accounted for about 39 % of renewable power capacity added in 2012, followed by hydropower and solar PV, each accounting for approximately 26 %. As prices for renewable energy technologies continue to fall, a growing number of renewables are expected to be gradually reaching grid parity [1].

Due to the fact that renewable energy generation is intermittent, traditional power systems are undergone a major transition toward an operational state involving large amounts of variable energy sources. However, the intrinsic nature of power systems calls for suitable methods that are able to counteract the increased level of uncertainty which arises from the stochastic behavior of renewable energy. To this end, power system planning and operation procedures are being continually updated in order to ensure a constructive integration of renewable energy generation [2, 3].

Thus, the ultimate planning goal is to harness the available RES potential in such a way that does not compromise the system's reliability whilst, at the same time, reduces the dependency on fossil-fueled technologies. Nevertheless, integrating large amounts of renewable energy is not a simple task due a number of technical and economic constraints imposed [4]. The vital issue that needs to be addressed is, *how the existing infrastructure can cope or even benefit from the increasing penetration of renewable energy?*

To this aim, two factors are crucially important in evaluations pertaining to renewable energy integration: (a) the RES plants' *size* and (b) *location* (both geographical as well as electrical) [5–8]. It is, thus, important to use such tools that are able to explore possible configurations under a range of scenarios that may examine systems' expansion and RES integration plans from generation down to distribution level.

The general framework of the RES integration analysis performed in this work aims at capturing the stochastic behavior of loads and RES generation levels in order to provide a spherical view of the expected impact of RES penetration on the system. Therefore, a probabilistic approach is adopted. The analysis technique used is generally known as stochastic power flow (SPF) [9]. There exist two approaches to the SPF concept, namely the *analytical* and the *numerical*. The analytical SPF method relies on using convolution techniques, with probability density functions (PDFs) of input stochastic variables that enable the calculation of the PDFs of output stochastic variables, such as the system states and line flows.

Conversely, the numerical approach of SPF is an iterative process that selects a random value for each of the (stochastic) system variables (i.e., loads and RES generation specifics), based on a specified probability distribution. Thus, the optimal power flow is subsequently executed based on these random values. The result of each optimal power flow is recorded, and the process is repeated. A typical SPF (based on Monte Carlo simulations) calculates the power flows numerous times, each time using different randomly selected values.

To this extent, DIGSILENT PowerFactory has been established as a versatile power system analysis software for applications in generation, transmission, distribution, and industrial systems. It is a valuable tool for power system planners as its capabilities cover a very wide spectrum of studies necessary for any power system optimization process.

The stochastic behavior of the loads and RES generation calls for appropriate probabilistic techniques that can capture the overall effect of integrating renewable energy. Therefore, the SPF concept could prove an effective way of approaching the RES integration issues. Moreover, a series of technical performance indices could be utilized (as found in relevant academic literature [6, 7]) in order to expand on the SPF concept through the use of DPL. These indices compare the system's initial state (base scenario) to each potential future configuration on three technical aspects, namely power losses, voltage deviation, and line loading.

6.1.1 Chapter Contribution

This chapter expands on the SPF concept through a case study of integrating wind power in the WSCC 3-machine, 9-bus system [10] (see Fig. 6.1). This test system is used as the benchmarking network to materialize the subsequent indices evaluation. In particular, the system consists of three conventional generators that supply three major loads. The installed capacity sums to 820 MW, whilst the mean load is at 315 MW. Moreover, it is assumed that a total 100 MW of rated wind capacity is installed in the system as per Table 6.1.

Specifically, each load follows a normal distribution $X \sim N(\mu, \sigma^2)$ with $\mu = 1$ and $\sigma = 0.07$ on a per unit scale. The wind generation follows a Weibull distribution with scale parameter $\lambda = 0.3$ and shape parameter $k = 2$ (this parameter selection emulates the per unit wind generation level with approximately 26.5 % capacity factor). It is noted that all loads and wind generation sources are considered uncorrelated (i.e., all variables are mutually independent). The selected scenarios are shown in Table 6.1 and are collated to the test system shown in Fig. 6.1. Nevertheless, these scenarios are indicative and aim to act as a preliminary basis to illustrate how RES integration studies can be performed.

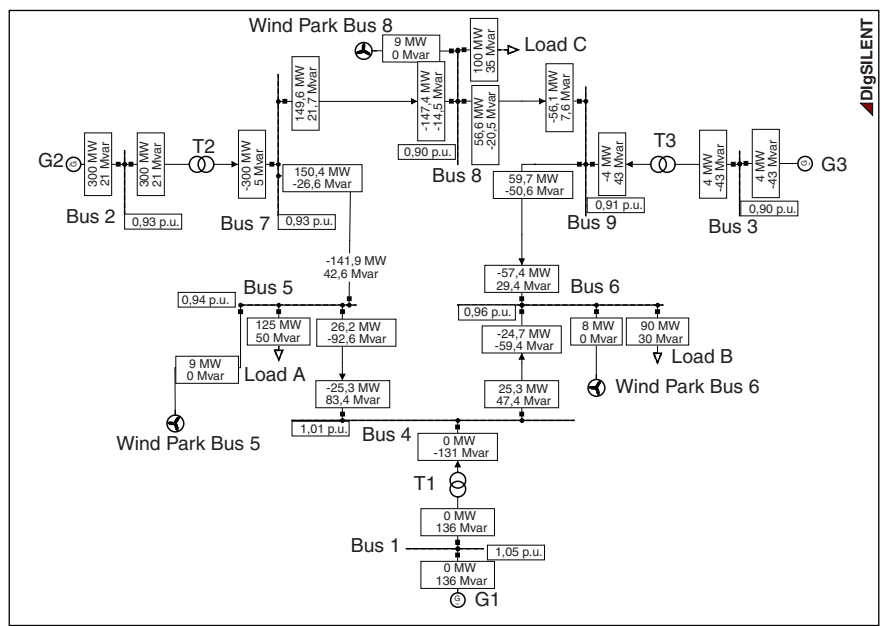


Fig. 6.1 Single line diagram of the test system: WSCC 3-machine, 9-bus system [10]

Table 6.1 Wind penetration details

Wind installed capacity at bus 5	Wind installed capacity at bus 6	Wind installed capacity at bus 8
35 MW	30 MW	35 MW

6.1.2 Theoretical Background

6.1.2.1 Power Losses Indices

The first two technical performance indices relate to the system’s total power losses. Planners should always bear in mind the technical and economic impact of losses on their system’s planning and operating objectives. Strategically placed RES penetration has been proven an effective way of reducing the overall system losses [8, 11]. This raises the value of investing in decentralized RES generation as a way of locally offsetting the system’s load. This will in turn reduce the needs for bulk power transmission over long distances. Thus, the line power losses indices (i.e., ILp and ILq) compare the total active and reactive losses between the potential RES

configuration and the base scenario. Their mathematical formulation is shown in (6.1) and (6.2):

$$IL_p = 1 - \frac{\text{Re}\{\text{Losses}\}}{\text{Re}\{\text{Losses}^0\}} \quad (6.1)$$

$$IL_q = 1 - \frac{\text{Im}\{\text{Losses}\}}{\text{Im}\{\text{Losses}^0\}} \quad (6.2)$$

where Losses refers to the total power losses of the RES scenario under examination, whereas Losses⁰ refers to the total power losses of the base scenario (scenario without RES generation). Near unity indices values imply a positive effect (minimization of losses) of the RES penetration, whereas negative values imply an increase of power losses.

6.1.2.2 Voltage Deviation Index

Voltage levels are crucial for the system's operation. Keeping voltage within acceptable range enables reliable power transmission whilst maintaining power quality. The maximum allowable deviation is set by the operational constraints of each network as per the system's requirements. It is often assumed that a ± 5 or ± 10 % deviation from the nominal network voltage is acceptable. The IVD index refers to the maximum voltage deviation between the network's buses. Nevertheless, a flat voltage profile is usually desirable. In this regard, positive IVD values imply a flattening of the voltage profile, whereas negative values imply a wider voltage deviation. In mathematical terms, the index is expressed as in (6.3).

$$IVD = (V_{\max}^0 - V_{\min}^0) - (V_{\max} - V_{\min}) \quad (6.3)$$

where V_{\max}^0 and V_{\min}^0 refer to the maximum voltage level of the base and RES scenario, respectively, whilst V_{\max} and V_{\min} refer to the minimum voltage level.

6.1.2.3 Line Loading Index

The ILL index refers to the loading level of the system's transmission lines. Minimizing line loading through strategically placed RES penetration may be an effective way of deferring transmission and distribution investment [2]. This may also reduce the size of the investment costs pertaining to the installation of new transmission lines necessary to compensate the system's tolerance on load growth. Thus, positive ILL values imply that line flows have been reduced. Inversely, negative values would imply that the examined configuration leads to a larger

loading of the system's lines. In mathematical terms, the index can be expressed as in (6.4).

$$ILL = \max \left(\frac{SL_m^0}{SR_m} \right)_{m=1}^{NL} - \max \left(\frac{SL_m}{SR_m} \right)_{m=1}^{NL} \quad (6.4)$$

where SL_m^0 and SL_m refer to the loading of line m for the base and RES scenario, respectively, whilst SR_m is the rated ampacity of line m . NL refers to the number of lines of the system.

This index serves as the means to identify whether a candidate system configuration raises or reduces the loading level of the system's most loaded transmission line. Therefore, in cases where the loading level of a heavily loaded transmission line drops, a line flow "relief" takes place. When the loading level of the most loaded transmission line increases, an upgrade of the particular line may be needed in the future. Thus, the decrease/increase in line loading reflects on the tolerance of the system's transmission capabilities to load growth. It, thus, provides an indirect sign for an associated upgrading requirement.

6.1.3 Creating a Synthesized DPL Script for RES Integration Studies

It is usual for a large, synthesized DPL script to consist of multiple smaller scripts acting as subroutines that execute a certain required process and return their results to the main program for further calculations. In this work, this programming approach is deemed necessary for the execution of multiple optimal power flows and indices calculation as per the SPF requirements. Therefore, the subroutine scripts must be written and contained in the main program as objects in order for the process to be executed correctly. To execute a subroutine, we simply call the respective object (DPL script) in the code and use the *Execute()* method [12].

Since the indices are based on a direct comparison between the base and RES scenario, it is logical that, for each of the iterations, both the base and RES scenario's optimal power flows must be executed. Therefore, the load scaling must be applied to both scenarios, whilst the RES generation scaling is valid only for the RES scenario (since the base scenario is considered to exclude any RES plants). The process of the concept as implemented in DPL is shown in Fig. 6.2.

Before executing the process, the following scripts (i.e., *ComDpl* objects) should be created under the respective names in order to execute the RES integration studies:

- **MAIN.ComDpl:** This script is the main program that calls all the relevant subroutines, calculates, and stores the indices values for each of the user-defined number of iterations.

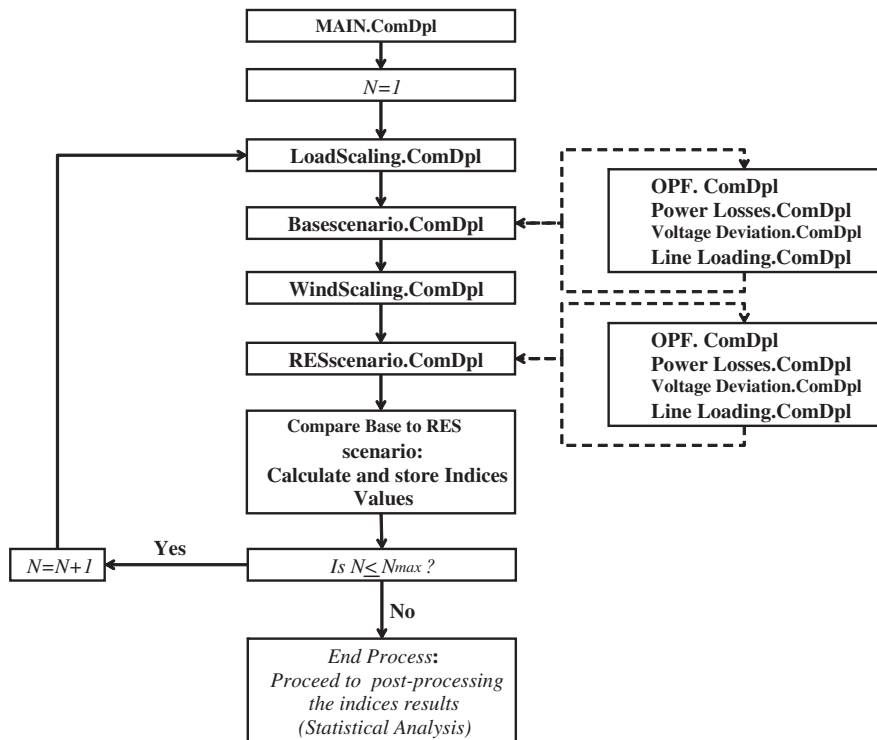


Fig. 6.2 Process implementation in DPL

- **LoadScaling.ComDpl:** This script scales all the system's loads according to their respective distribution.
- **Basescenario.ComDpl:** This script is responsible for executing the base scenario optimal power flow and returns all the relevant results to the main program.
- **WindScaling.ComDpl:** This script scales all the wind parks' generation according to their respective distribution.
- **RESscenario.ComDpl:** This script is responsible for executing the RES scenario optimal power flow and returns all the relevant results to the main program.
- **OPF.ComDpl:** This script runs the AC optimal power flow.
- **PowerLosses.ComDpl:** This script takes as input the optimal power flow results and returns the active and reactive losses.
- **VoltageDeviation.ComDpl:** This script takes as input the optimal power flow results and returns the maximum voltage deviation.
- **LineLoading.ComDpl:** This script takes as input the optimal power flow results and returns the maximum line loading level.

6.1.4 *MAIN.ComDpl*

This script is the core of the synthesized structure of the RES integration case study of this chapter. It performs the technical performance indices calculations by calling all the appropriate subroutines needed for the analysis for a user-defined number of iterations. Subsequently, it stores the results in an .xls file under a user-defined name. Therefore, the script (shown in Fig. 6.3) acts as the driver of the case study and handles its results.

In order to reproduce this chapter's case study, the user should set the respective parameters as follows:

- `FileName`: set the name of the result-storing file,
- `Iterations`: set the number of Monte Carlo iterations equal to 1,000.

6.1.5 *LoadScaling.ComDpl*

Before an optimal power flow is executed, the loads should be scaled as per the user's interests. As mentioned above, the SPF relies on scaling the loads through a random number selection based on their respective distribution. Therefore, the DPL script should find and scale all the relevant objects (as per the user's requirements), thus preparing the parameters that are involved in the optimal power flow calculation.

In the following example, the DPL script performs a separate random scaling of all the system's loads. The first task of a scaling script would be to create two sets. The command "*AllRelevant*" is a DPL internal method that returns a set with calculation relevant objects. Therefore, a set must be created that should contain all the system's load(s). The associated DPL object class is "**.ElmLod*". The set of calculation relevant objects will be determined by the currently active study case and grid.

The scaling can be performed on a per unit basis or by directly determining the active and reactive requirements of each load, depending on the user's convenience. The relevant object field(s) for the per unit scaling of each load is `oLoad:scale0`, whilst for a direct load determination are `oLoad:plini` and `oLoad:qlini` (for the active and reactive requirements, respectively). The example that follows uses the per unit scaling.

In order to generate the random numbers, the DPL internal method `fRand` is utilized which returns a random number according to a specific probability distribution. The `fRand` command supports the random number generation from three kinds of distributions, namely the uniform, the normal, and the Weibull distribution. The syntax of `fRand(mode,p1,p2)` determines the characteristics of the underlying distribution. Table 6.2 shows the syntax details as found in [12].

```

! Variable declaration

object SumGrid,SumGrid0;

int i,j,Iterations;

double ILp,ILq,IVD,ILL,LossesP0,LossesQ0,LossesP,LossesQ,VD,VD0,Max0,Max;

string FileName;

ClearOutput();!Clears the output window

Iterations = 1000; ! Sets the number of Monte Carlo iterations

FileName ='C:\DIgSILENT\Results.xls';! Declares the name of the result-storing file

fopen(FileName,'w',0) ;! Opens the file where the results will be saved

fprintf(0,'%s %s %s %s','ILp','ILq','IVD','ILL'); ! Defines the indices' headers for
the user's convenience when viewing the results file

for (j=1;j<(Iterations+1);j=j+1){! Starts the Monte Carlo iteration loop

LoadScaling.Execute();!Executes the load scaling script

Basescenario.Execute(LossesP0,LossesQ0,SumGrid0,VD0,Max0);! Executes the base scenario
script and returns the relevant results for the indices calculations

WindScaling.Execute();! Executes the wind scaling script

RESSscenario.Execute(LossesP,LossesQ,SumGrid,VD,Max);! Executes the RES scenario script
and returns the relevant results for the indices calculations

ILp = 1 - LossesP/LossesP0;! Calculates the active power losses index ILp

ILq = 1 - LossesQ/LossesQ0;! Calculates the reactive power losses index ILq

IVD=VD0-VD;! Calculates the voltage deviation index IVD

ILL = 1 - (Max/Max0); ! Calculates the line loading index ILL

fprintf(0,'%5f %5f %5f %5f',ILp,ILq,IVD,ILL);! Stores the indices results in the
file

}

! Closes the file

fclose(0);

```

Fig. 6.3 MAIN.ComDpl code

Table 6.2 *fRand* syntax details

	Uniform (mode = 0)	Normal (mode = 1)	Weibull (mode = 2)
P1	Min	Mean	Shape
P2	Max	Standard deviation	Scale

Figure 6.4 shows the script's code. In order to reproduce this chapter's case study, the user should set the respective parameters as follows:

- **LoadDistribution**: set the mode of the *fRand* equal to 1 (i.e., normal distribution),
- **Loadmu**: set the mean of the load distribution equal to 1 (i.e., the P1 parameter of the *fRand*),
- **Loadsigma**: set the standard deviation of the load distribution equal to 0.07 (i.e., the P2 parameter of the *fRand*),
- **oLoad:scale0**: set this equal to *fRand*(**LoadDistribution**, **Loadmu**, and **Loadsigma**) in order to perform the random scaling of each load.

```

! Variable declaration

int LoadDistribution;

double Loadmu, Loadsigma;

set sLoads;

object oLoad;

LoadDistribution = 1; ! Sets the fRand mode parameter of the loads

Loadmu = 1; ! Sets the mean of the loads' distribution

Loadsigma = 0.07; ! Sets the standard deviation of the loads' distribution

sLoads = AllRelevant('*.ElmLod'); ! Creates a set containing all the system's loads

oLoad = sLoads.First(); ! Gets the first object (i.e. load)

while (oLoad){

    oLoad:scale0 = fRand(LoadDistribution, Loadmu, Loadsigma); ! Per unit random scaling of the
    load based on the distribution and parameter selection

    oLoad = sLoads.Next(); ! Gets the next object (i.e. load)

}

```

Fig. 6.4 LoadScaling.ComDpl code

6.1.6 *Basescenario.Dpl*

The base scenario program must be constructed in a “script-within-a-script” manner; this means that the base scenario must itself be a subroutine that contains subroutines. The base scenario script is responsible for excluding all RES generators from the optimal power flow calculation. Therefore, the script, shown in Fig. 6.5, should create a set containing all the RES generators of the system (“*.ElmGenstat” class) and set them out-of-service by using the appropriate object field before executing the optimal power flow calculation.

In order to reproduce this chapter’s case study, the user should set the respective parameters as follows:

- oStat:outserv: set this field equal to 1 in order to exclude all RES generators from the system.

```
! Variable declaration
set sStat;

object oStat;

int i;

sStat=AllRelevant('*.ElmGenstat');! Creates a set containing all the wind generators
of the system

oStat=sStat.First();! Gets the first object (wind generator) of the set

while (oStat){

if (oStat:outserv=0){! Check if the wind generator is in service. If yes,

oStat:outserv=1;! it sets it out of service

}

oStat=sStat.Next();      ! Gets the next object (wind generator)

}

i=OPF.Execute(SumGrid0);! Executes the optimal power flow script and returns the
Summary Grid

PowerLosses.Execute(SumGrid0,LossesP0,LossesQ0);! Executes the poewr losses script and
returns the active and reactive losses

VoltageDeviation.Execute(SumGrid0,VD0); ! Executes the voltage deviation script and
returns the difference between the maximum and minimum bus voltage level

LineLoading.Execute(Max0);! Executes the line loading script and returns the loading
of the system's most loaded line
```

Fig. 6.5 Basescenario.ComDpl code

6.1.7 *WindScaling.ComDpl*

In order to scale the wind generation of each of the system's wind parks, a set must be created which will contain the system's RES generator(s), which can be addressed as the `*.ElmGenstat` class. The scaling is performed by the `oWind:pgini` and `oWind:qgini` object fields (active and reactive generation, respectively). In order to apply a per unit scaling for the wind generators, the following concept is applied: The rated capacity of each *individual* wind generator is set equal to 1 MVA (the relevant field for this is `oWind:sgn`). The total wind park capacity (i.e., the number of parallel wind generators) is then set equal to the desired total rated capacity (in this example, the object field `oWind:ngnum` is set for each individual wind park as in Table 6.1). Once the rated capacity of the plant has been set, the per unit scaling can be performed by randomly selecting numbers from a Weibull distribution with appropriate characteristics. The random numbers are inserted in the `oWind:pgini` field of the object (the `oWind:qgini` is set equal to 0 in order to emulate a power factor equal to 1).

The script's code is shown in Fig. 6.6. In order to reproduce this chapter's case study, the user should set the respective parameters as follows:

- `WindDistribution`: set the mode of the `fRand` equal to 2 (i.e., Weibull distribution),
- `WindShape`: set the shape parameter of the Weibull distribution equal to 0.3,
- `WindScale`: set the scale parameter of the Weibull probability equal to 2,
- `W1`: set the installed capacity of the wind park installed at bus 5 equal to 35,
- `W2`: set the installed capacity of the wind park installed at bus 8 equal to 35,
- `W3`: set the installed capacity of the wind park installed at bus 6 equal to 30,
- `oWind:sgn`: set the rated capacity of each wind park equal to 1,
- `oWind:ngnum`: set the number of parallel machines of each wind park equal to `W1`, `W2`, and `W3`, respectively,
- `oWind:pgini`: set the active power generated by each wind park equal to `fRand(WindDistribution, WindShape, and WindScale)` in order to perform the random scaling of each wind park's generation,
- `oWind:qgini`: set the reactive power generated by each wind park equal to 0.0001 (this is to avoid optimal power flow convergence issues),
- `oWind:outserv`: set the out-of-service parameter of each wind park equal to 0 (i.e., ensures that the wind park is in service).

6.1.8 *REScenario.ComDpl*

The RES scenario program must also be constructed in a "script-within-a-script" manner, similar to the base scenario program. The script's code is shown in Fig. 6.7.

```

! Variable declaration

int WindDistribution;

double WindScale, WindShape, W1, W2, W3;

set sWind;

object oWind;

sWind=AllRelevant('*.ElmGenstat');! Creates a set containing all the wind generators

WindDistribution = 2;! Sets the fRand mode parameter of the wind generators

WindShape = 0.3;! Sets the shape parameter of the wind generators' distribution

WindScale = 2;    !Sets the scale parameter of the wind generators' distribution

W1=35;! Insert the desired rated capacity of the first wind park

W2=35;! Insert the desired rated capacity of the second wind park

W3=30;! Insert the desired rated capacity of the third wind park

oWind=sWind.Obj(0);! Gets the first object (wind park)

oWind:outserv=0;! Sets the object in service

oWind:sgn=1;! Sets the rated capacity of the first wind park

oWind:ngnum=W1;! Sets the number of parallel generators (i.e. rated capacity) of the
first wind park

```

Fig. 6.6 WindScaling.ComDpl code

6.1.9 OPF.ComDpl

The SPF application requires a DPL script (i.e., a *ComDpl* object) that automates the optimal power flow execution. The script shown below uses the DPL internal method `GetCaseCommand` ('ComOpf') that calls the optimal power flow calculation feature of *PowerFactory*. Subsequently, the script executes the optimal power flow and gets the summary grid (another DPL internal method [12]) which contains all the relevant results of the calculation. Since the numerical SPF requires a number of iterations of the same process, this script (shown in Fig. 6.8) could be called as many times as needed (in conjunction with the scaling script) in order to execute the iterative optimal power flow calculation.

In order to reproduce this chapter's case study, the user should set the respective parameters as follows:

```

oWind:pgini=fRand(WindDistribution,WindShape,WindScale);! Random scaling of the active
power generation level of the first wind park based on the respective distribution
selection

oWind:qgini = 0.0001;! Scales the reactive power generation to avoid convergence
issues

oWind=sWind.Obj(2);! Gets the second object (wind park)

oWind:outserv=0;! Sets the object in service

oWind:sgn=1;! Sets the rated capacity of the second wind park

oWind:ngnum=W2;! Sets the number of parallel generators (i.e. rated capacity) of the
second wind park

oWind:pgini=fRand(WindDistribution,WindShape,WindScale);! Random scaling of the active
power generation level of the second wind park based on the respective distribution
selection

oWind:qgini = 0.0001;! Scales the reactive power generation to avoid convergence
issues

oWind=sWind.Obj(2);! Gets the first object (wind park)

oWind:outserv=0;! Sets the object in service

oWind:sgn=1;! Sets the rated capacity of the first wind park

oWind:ngnum=W3;! Sets the number of parallel generators (i.e. rated capacity) of the
third wind park

oWind:pgini=fRand(WindDistribution,WindShape,WindScale);! Random scaling of the active
power generation level of the thiornd wind park based on the respective distribution
selection

oWind:qgini = 0.0001;! Scales the reactive power generation to avoid convergence
issues

```

Fig. 6.6 (continued)

- oOPF:iopt_ACDC: set the option of the optimal power flow method equal to 0 in order to ensure that the AC optimal power flow is executed,
- oOPF:iobj_obj: set the option of the optimal power flow's objective function equal to 'Minimization Of Costs'.


```

! Variable declaration

int i;

i=OPF.Execute(SumGrid);! Executes the optimal power flow script and returns the
Summary Grid

PowerLosses.Execute(SumGrid,LossesP,LossesQ);! Executes the poewr losses script and
returns the active and reactive losses

VoltageDeviation.Execute(SumGrid,VD); ! Executes the voltage deviation script and
returns the difference between the maximum and minimum bus voltage level

LineLoading.Execute(Max);! Executes the line loading script and returns the loading of
the system's most loaded line

```

Fig. 6.7 RESSscenario.ComDpl code

```

! Variable declaration

object oOPF;

int i;

ResetCalculation(); ! Resets any previous calculations

oOPF=GetCaseCommand('ComOpf'); ! Gets the object that refers to the OPF calculation

oOPF:iopt_ACDC=0; ! Activates the AC OPF calculation

oOPF:iopt_obj='Minimization Of Costs'; ! Sets the OPF objective function

i=oOPF.Execute(); ! Executes the OPF

SumGrid=SummaryGrid();! Gets the summary grid results of the OPF execution

```

Fig. 6.8 OPF.ComDpl code

6.1.10 *PowerLosses.ComDpl*

The power losses script is responsible for calculating the total active and reactive losses of each optimal power flow. It must, thus, create a set of all the system's lines ("*.ElmLne" class) and, subsequently, calculate the respective losses depending on the current flowing through each line. Figure 6.9 showcases the script's code.

```

! Variable declaration

object O, Omax;

set S;

double TotalP, TotalQ, I, R, X;

S=AllRelevant('*.ElmLne');! Creates a set containing all the system lines

TotalP=0;! Initializes variable for calculating total active power losses

TotalQ=0;! Initializes variable for calculating total reactive power losses

O=S.First();!Gets the first object of the set

while (O){

I=O.m:I:bus1;! Gets the current flowing through the line

R=O.R1;! Gets the resistance of the line

X=O.X1;! Gets the reactance of the line

LossesP = 3*I*I*R;! Calculates the active power losses of the line

LossesQ = 3*I*I*X;! Calculates the reactive power losses of the line

TotalP=TotalP+LossesP;! Adds the active line losses to the total active losses of the
system

TotalQ=TotalQ+LossesQ; ! Adds the reactive line losses to the total reactive losses of
the system

O=S.Next();! Gets the next object (line)

}

LossesP=TotalP;

LossesQ=TotalQ;

```

Fig. 6.9 PowerLosses.ComDpl code

6.1.11 VoltageDeviation.ComDpl

This script is responsible for calculating the maximum voltage deviation between all energized system buses. Therefore, a set must be created containing the buses that are in service (“*.ElmTerm” class). Once the set is created, the script seeks the

maximum and minimum bus voltage level (on a per unit basis) and returns their difference as its output. The object field containing the per unit voltage level of each bus is `ObjectName:m:u`. The script's code is shown in Fig. 6.10.

```
! Variable declaration

set sBuses;

object oBus;

double Max,Min;

sBuses=AllRelevant('*.ElmTerm',0);! Creates a set containing all buses in service

Max=-100;!Initializes maximum voltage level

oBus = sBuses.First();!Gets the first object of the set

while (oBus){

if (oBus:m:u>Max){

Max=oBus:m:u;!Updates maximum voltage level

}

oBus=sBuses.Next();

}

Min=100; !Initializes minimum voltage level

oBus = sBuses.First();

while (oBus){

if (oBus:m:u<Min){

Min=oBus:m:u;!Updates minimum voltage level

}

oBus=sBuses.Next();!Gets the next object (bus)

}

VD= 1 - (Max-Min);
```

Fig. 6.10 VoltageDeviation.ComDpl code

```

! Variable declaration

double Loading;

object oLine;

set sLines;

sLines = AllRelevant('*.ElmLne'); ! Creates a set containing allsystem lines

Max=0;    !Initializes variable

oLine = sLines.First();! Gets first object (line) of the set

while (oLine){

if (oLine:c:loading>Max){!Checks maximum

Max=oLine:c:loading;! Updates maximum

}

oLine=sLines.Next();!Gets next object (line)

}

```

Fig. 6.11 LineLoading.ComDpl code

6.1.12 LineLoading.ComDpl

This script is responsible for seeking the maximum loaded line of the system and returning the loading value as output. Therefore, a set containing all the lines (“**ElmLne*” class) of the system must be created. The object field that contains the loading of each line is `ObjectName:c:loading`. Figure 6.11 showcases the script’s code.

6.2 Case Study Results

Having described and developed all the necessary scripts for the RES integration case study, the results of the technical performance indices are shown in Figs. 6.12, 6.13, 6.14, and 6.15. The indices values are put in a descending order thus creating an empirical cumulative distribution function. Therefore, the user can statistically treat the case study’s results as shown in Table 6.3. It is logical that increasing the number of Monte Carlo iterations will increase the accuracy of the process.

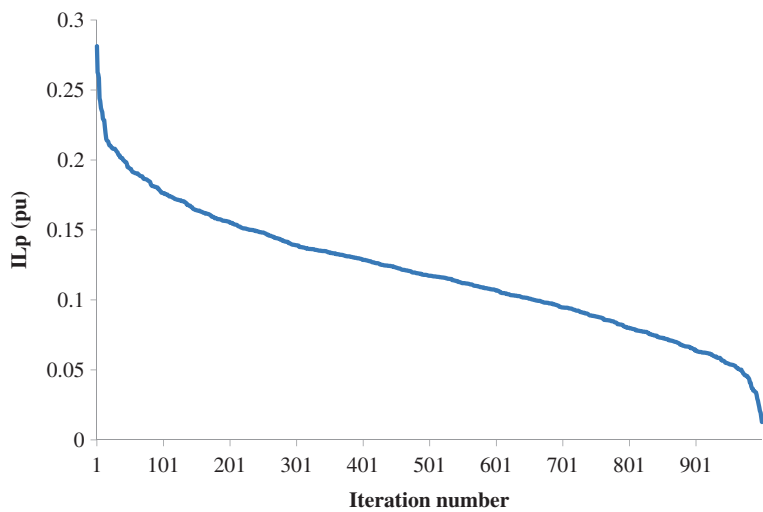


Fig. 6.12 ILp results of the RES integration case study

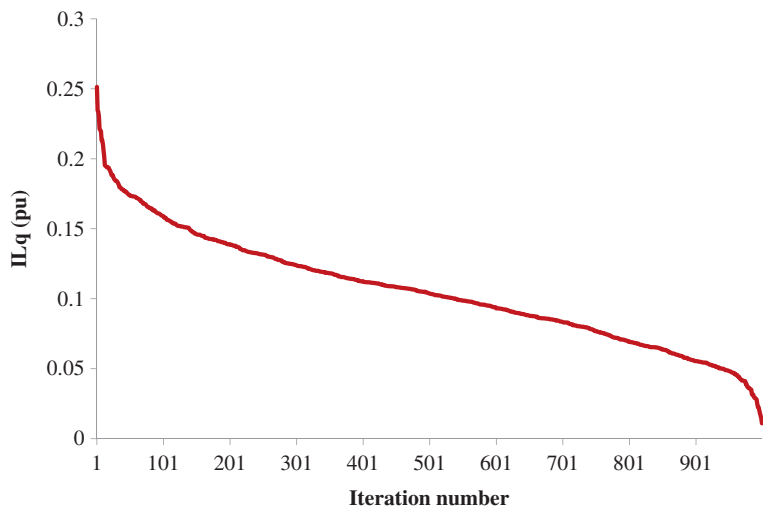


Fig. 6.13 ILq results of the RES integration case study

Nevertheless, after a certain number of iterations, this increase in accuracy is not worth the extra computational time and resources. The number of iterations was in this case study set equal to 1,000 in order to provide generic results without accuracy concerns. It is, therefore, possible that the reproduction of this chapter’s case study may lead to slightly different indices results. However, each user can adjust this number as per his own objectives.

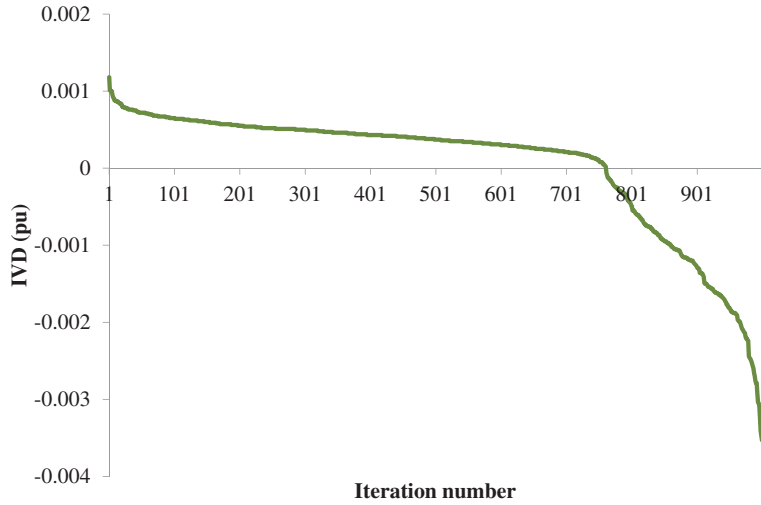


Fig. 6.14 IVD results of the RES integration case study

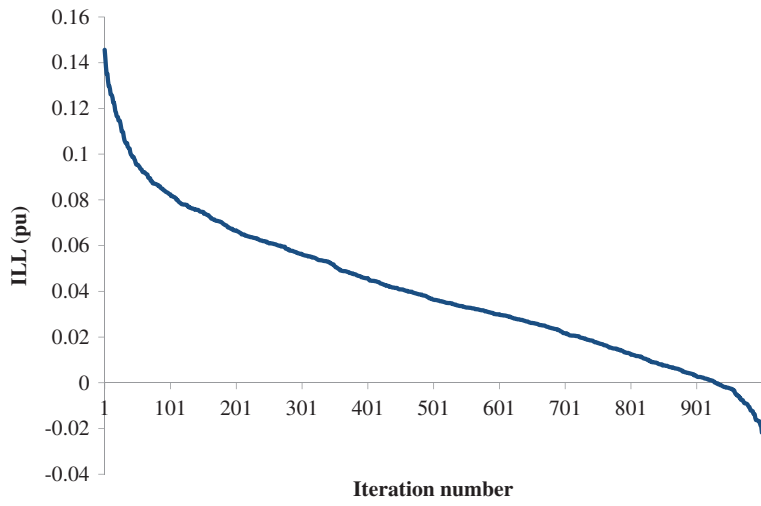


Fig. 6.15 ILL results of the RES integration case study

Table 6.3 Statistical analysis of the technical performance indices results

Index	Mean	Median	Maximum	Average
ILp	0.11943	0.11729	0.28129	0.01281
ILq	0.10560	0.10376	0.25133	0.01105
IVD	0.00004	0.00037	0.00118	-0.00353
ILL	0.04079	0.03637	0.14559	-0.02182

The ILp and ILq results show that introducing dispersed wind generation, especially when located at load centers, leads to loss reduction due to the reduced transmission line currents. Decentralized generation may, therefore, contribute to power losses cost minimization. However, the RES penetration effect on the voltage profile of a power system is strongly dependent on the existence of reactive power support near the load (e.g., shunt capacitors, reactors, and generating units). This is reflected in the IVD index results that are not as smooth as the ILp and ILq indices results due to the specific topology of the test system. Specifically, the fact that there exist three machines which can provide reactive power support to the system buses directly influences the RES penetration effect on the voltage profile. Moreover, line loadings are also decreased since the loads are partially offset by local generation.

6.3 Conclusion

RES integration in existing systems is a very complex issue. DigSILENT can be a very useful tool in applying such methodologies on power systems with a high level of precision and flexibility. This was demonstrated through the programming language DPL that can be of great benefit for creating and automating several power system calculations. Under this framework, system planners and operators can investigate the capabilities and limitations of their power systems thus avoiding undesirable situations that could potentially harm the systems' well-being.

In particular, conclusions of this work are heavily based on the SPF concept. This is considered as one of the means to capture the random behavior of the system loads and RES generation specifics. Each scenario was examined and evaluated based on the impact of RES penetration on power losses, voltage deviation, and line loading.

References

1. REN 21 (2013) Renewables 2013: global status report. Technical Report
2. David TC, Wang Luis F, Ochoa Gareth Harrison (2010) DG impact on investment deferral: network planning and security of supply. *IEEE Trans Power Syst* 25(2):1134–1141
3. Lingfeng Wang (2008) Integration of renewable energy sources: reliability-constrained power system planning and operations using computational intelligence. PhD dissertation, Texas A&M University, College Station
4. Jenkins N, Allan R, Crossley P, Kirschen D, Strbac G (2008) Embedded generation. In: Johns AT, Warne DF (eds) *Institution of Engineering and Technology*, London
5. Abookazemi K, Hassan MY, Majid MS (2010) A review on optimal placement methods of distribution generation sources. In: *Proceedings of IEEE international conference on power and energy (PECon)*, pp 712–716
6. Ochoa Luis F, Padilha-Feltrin Antonio, Harrison Gareth (2006) Evaluating distributed generation impacts with a multi-objective index. *IEEE Trans Power Deliv* 21(3):1452–1458

7. Nikolaidis AI, Gonzalez-Longatt FM, Charalambous CA (2013) Indices to assess the integration of renewable energy resources on transmission systems. In: Proceedings of conference papers in energy, vol 2013, pp 1–8
8. Ochoa Luis F, Harrison Gareth P (2011) Minimizing energy losses: optimal accommodation and smart operation of renewable distributed generation. *IEEE Trans Power Syst* 26 (1):198–204
9. Gonzalez-Longatt F, Roldan JM, Rueda JL, Charalambous CA (2012) Evaluation of power flow variability on the Paraguana transmission system due to integration of the first venezuelan wind farm. In: Proceedings of power and energy society general meeting, 2012. IEEE, San Diego, pp 1–8
10. Anderson PM, Fouad AA (2003) Power system stability and control, 2nd edn. IEEE Press, New York
11. Gonzalez-Longatt FM (2007) Impact of distributed generation over power losses on distribution system. In: Proceedings of 9th international conference, electrical power quality and utilisation, Barcelona, pp 1–6
12. DIgSILENT GmbH (2011) DIgSILENT PowerFactory: version 14.1-User's manual. Technical Report, Gomaringen, Germany

Chapter 7

Modeling of Automatic Generation Control in Power Systems

V. Pavlovsky and A. Steliuk

Abstract Frequency stability and active power control is one of the principle tasks of improving the security and reliability of power system operation. The chapter covers the various aspects of simulation of frequency control in power systems. The chapter is divided into three parts. In the first part, the elements of the frequency control are considered and main equations are presented. In the second part, PowerFactory software model of automatic generation control (AGC) is considered. In the third part, the results of load flow and transient simulation are presented.

Keywords Frequency • Power flow • Automatic generation control • Primary and secondary frequency controls • Power system

7.1 Introduction

Nowadays, the renewables, primarily the wind and solar power plants, are widely used in power systems all over the world. One of the features of the renewables is their variable and stochastic generation power. It is obvious that penetration of the renewables complicates power system control. Today, improvement of the power system dispatch, power quality, providing the flexible control of the voltage, and reactive power are actual and important tasks [1]. Besides, one of the actual tasks is the improving of frequency control. It requires a simulation of the automatic generation control (AGC) operation considering the different dynamic characteristics of the generating units participating in the frequency control, active power spinning reserves planning, etc. [2, 3]. Thus, the simulation of the frequency control is based

Electronic supplementary material The online version of this chapter (doi:10.1007/978-3-319-12958-7_7) contains supplementary material, which is available to authorized users.

V. Pavlovsky · A. Steliuk (✉)
DMCC Engineering Ltd., Peremogy Ave, 56, Kiev 03056, Ukraine
e-mail: anton.stelyuk@dmcc.com.ua

on developing the appropriate AGC software model. In the chapter, the simplified AGC model is presented. However, this model allows to simulate the frequency control in power systems.

7.2 Theoretical Background of the Frequency Control

It is well known that the frequency in the power system depends on the active power equilibrium of generation and demand. Changing of the power demand or generation power leads to active power imbalance and consequently to the frequency and net interchange power deviation from their set point values. Let us consider the frequency control in the event of load increasing. In order to analyze the frequency control stages, we shall consider the frequency plot and time diagram (Fig. 7.1).

At the initial time, the increasing of the demand power in the power system is compensated by the kinetic energy of the rotating masses of the generators and motors at a decreased frequency. Several seconds later, the turbine governors and the loads react to the frequency deviation due to their dynamic characteristics. As a result, it leads to the active power balance at the deviated frequency. During *the primary control*, the dynamic frequency deviation depends on the following factors [4]:

1. The kind of disturbance (e.g., disconnection of generator or a smooth change of the active power);
2. A number of generating units participating in the primary frequency control;
3. Primary spinning reserve and its distribution among the generating units;
4. Dynamic characteristics of the load.

The frequency deviation after the primary control depends on the droops of the turbine governors as well as the characteristics of the load (including self-regulating effect). Thus, the primary control stabilizes the frequency at a deviated value, which differs from its initial one [4–6]. All synchronously operating interconnected power systems (IPS) have to participate in the primary control including the IPS, in which the imbalance has occurred, with the action time from several (4–5 s) to 30 s [4].

In order to restore the normal frequency and scheduled net interchange power, *the secondary frequency control* is used. The frequency and power flows are controlled by the generation power change that allows compensating the active power imbalance. The AGC can control the frequency or the net interchange power with the frequency correction that depends on the AGC operation mode. It should be noted that the power imbalance is to be compensated only by the AGC of the IPS, in which the imbalance has occurred. The action time of the secondary frequency control is from 30 s to 15 min [4]. Besides, as shown in Fig. 7.1, during the primary and secondary control, the frequency is to be maintained within the “trumpet” curves [4].

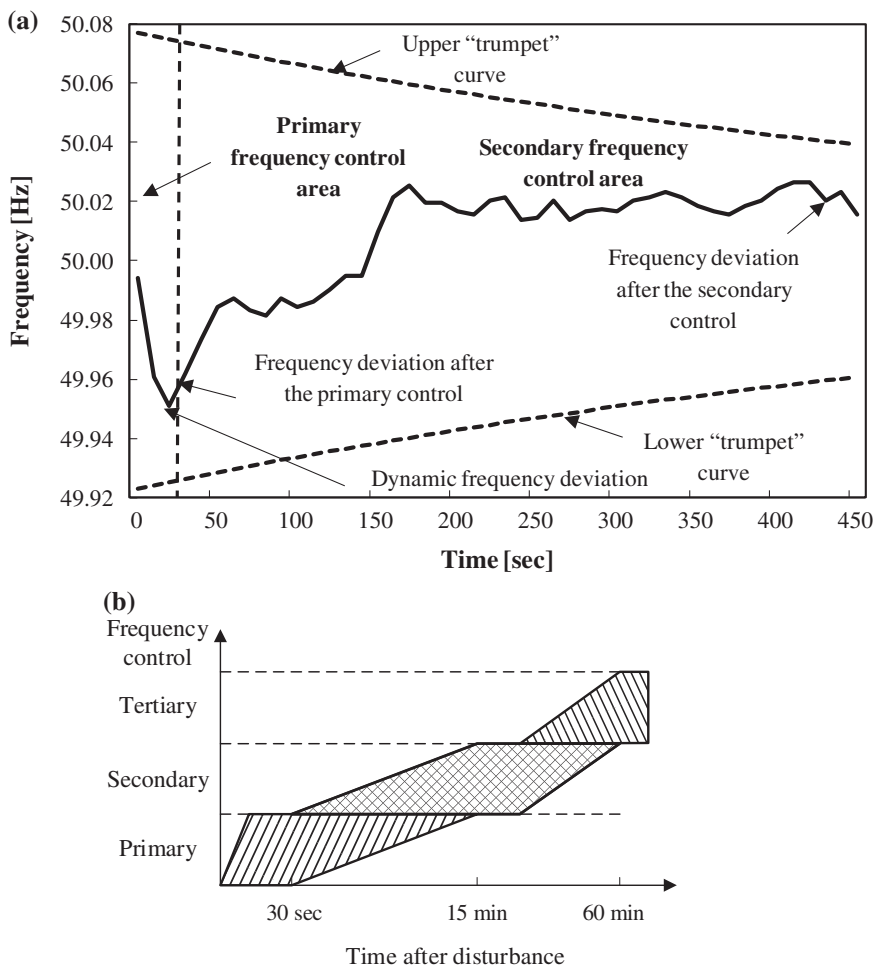


Fig. 7.1 Frequency control illustration **a** frequency change during the primary and secondary controls (SCADA measurements), **b** time diagram of frequency control

The tertiary frequency control is intended to restore the active power spinning reserves used during the primary and secondary control and to provide the optimal distribution of these reserves. The action time of the tertiary control is from 15 to 60 min.

The structure of the AGC of the IPS (used in the developed model in PowerFactory software) is shown in Fig. 7.2. There are three control levels of the active power and frequency control. The upper *system control level* is presented by the AGC of the IPS. The input signals are the current frequency f in the power system and tie-line interchanges P_{flowk} , $k = 1, \dots, P$.

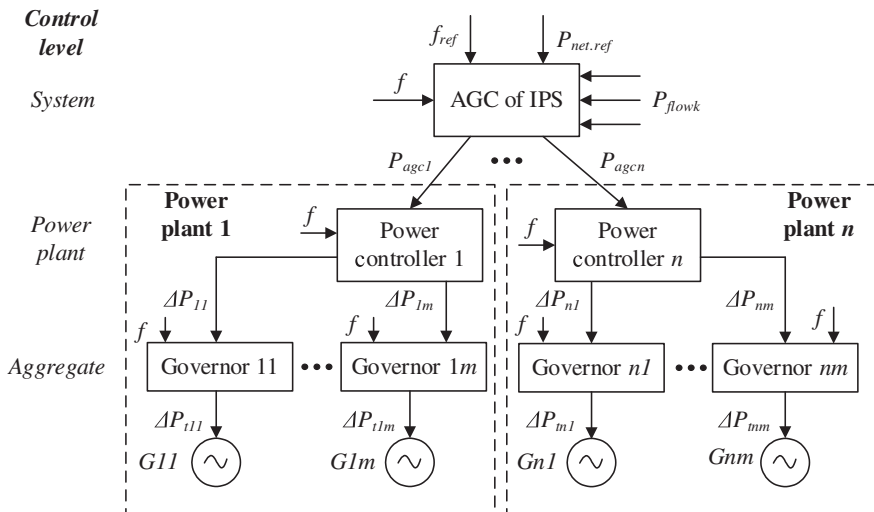


Fig. 7.2 The structure of the automatic generation control

Based on tie-line interchanges P_{flowk} , the net interchange power is calculated as follows:

$$P_{net} = \sum_{k=1}^P P_{flowk} \quad k = 1, 2, \dots, P. \quad (7.1)$$

In the AGC of the IPS (at the *system control level*), the frequency as well as the net interchange power deviations is defined as follows:

$$\Delta f = f - f_{ref} \quad (7.2)$$

$$\Delta P_{net} = P_{net} - P_{net.ref} \quad (7.3)$$

where

f_{ref} is a frequency set point value;

$P_{net.ref}$ is a net interchange power set point value.

Further, considering (7.2) and (7.3), the area control error (ACE) is calculated as follows:

$$ACE = -\Delta P_{net} + K_{bias} \Delta f \quad (7.4)$$

where

K_{bias} is the frequency bias

In the event of the “internal” imbalance in the IPS, ACE defines the power to be compensated by the regulating power plants in this IPS [5, 6]. In the case of the

“external” disturbance due to the different signs of frequency and net interchange power deviations, ACE value tends to zero that provides the selectivity of the AGC operation depending on location of the disturbance [5, 6].

The unscheduled active power setting P_{agc} formed by the proportional-integral (PI) controller on basis of (7.4) is calculated as follows:

$$P_{\text{agc}} = K_P \text{ACE} + K_I \int_{t_1}^{t_2} \text{ACE} dt \quad (7.5)$$

where

K_P is the proportional gain of the PI controller;

K_I is the integral gain of the PI controller;

t_1 and t_2 are the integration limits.

As shown in Fig. 7.2, the control signals P_{agci} , $i = 1, \dots, n$, transmitted to each regulating power plant are defined according to the participation factor α_i of individual power plant in the secondary frequency control:

$$P_{\text{agci}} = \alpha_i P_{\text{agc}} \quad i = 1, 2, \dots, n \quad (7.6)$$

At the *power plant control level*, the signal P_{PCi} formed by the power plant PI controller is calculated as follows:

$$P_{\text{PCi}} = K_P^{\text{PC}} \left(K_f \Delta f + P_{\text{agci}} - \sum_{j=1}^m \Delta P_{\text{Tj}} \right) + K_I^{\text{PC}} \int_{t_1}^{t_2} \left(K_f \Delta f + P_{\text{agci}} - \sum_{j=1}^m \Delta P_{\text{Tj}} \right) dt \quad i = 1, 2, \dots, n \quad (7.7)$$

where

K_P^{PC} is the proportional gain of the power plant PI controller;

K_I^{PC} is the integral gain of the power plant PI controller;

K_f is the coefficient of frequency correction;

$\sum_{j=1}^m \Delta P_{\text{Tj}}$ is the sum of the turbine power change of the generating units participating in the secondary frequency control

The distribution of the control signal P_{PCi} at the i -power plant control level is performed in accordance with the participation factors β_{ij} of the generating units of i -power plant in the secondary frequency control (see Fig. 7.2):

$$\Delta P_{ij} = \beta_{ij} P_{\text{PCi}} \quad i = 1, 2, \dots, n; j = 1, 2, \dots, m \quad (7.8)$$

where

- i is the number of the regulating power plants;
- j is the number of the generating units of the i -power plant;
- ΔP_{ij} is the control signal from the power plant controller

The control signal $\Delta P_{ij,\text{ref}}$ is distributed in such a way that

$$P_{\text{PC}i} = \sum_{j=1}^m \Delta P_{ij} \quad i = 1, 2, \dots, n; j = 1, 2, \dots, m \quad (7.9)$$

and

$$P_{\text{agc}} = \sum_{i=1}^n P_{\text{agci}} = \sum_{i=1}^n \sum_{j=1}^m \Delta P_{ij} \quad i = 1, 2, \dots, n; j = 1, 2, \dots, m \quad (7.10)$$

The calculated control signal ΔP_{ij} from the power controller is transmitted to the turbine governor of the generating unit (*aggregate control level*) via the speed changer motor (see Fig. 7.2). Further, according to the reference control signal ΔP_{ij} , the turbine governor generates a signal of the turbine power change ΔP_{ij} . Thus, the power changing of the generating units restores the normal frequency and scheduled net interchange power.

7.3 Automatic Generation Control Model in PowerFactory

In order to demonstrate the AGC operation, a simplified AGC software model considering the system and the aggregate control levels has been developed in PowerFactory software using the example of IEEE 14-bus test system (see Fig. 7.3). The test model includes 230, 115, and 10 kV networks as well as the following elements:

1. 7 synchronous generators;
2. 2 synchronous compensators;
3. 7 two-winding transformers;
4. 1 three-winding transformer;
5. 16 overhead lines (OHL) including 8 OHL 230 kV and 8 OHL 115 kV;
6. 13 loads.

The IPS 1 and IPS 2 operate synchronously. The AGC in IPS 1 controls the frequency (control criterion— $\Delta f = 0$). The control signal from the AGC controller in IPS 1 is transmitted to the governor of generator 1. In IPS 2, the AGC controls the net interchange power and the frequency. The frequency bias in the AGC of IPS 2 is 300 MW/Hz. In IPS 2, the active power settings from AGC are transmitted to

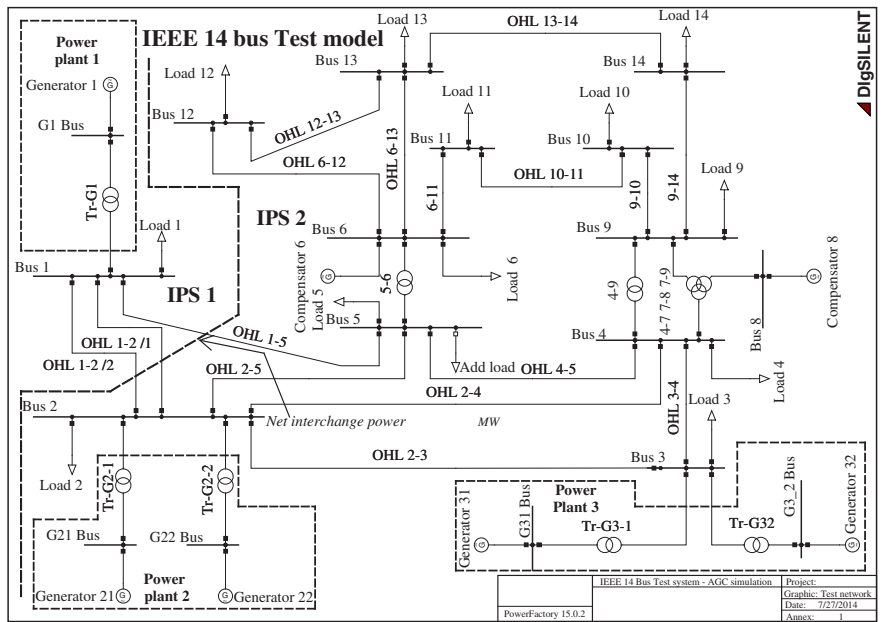


Fig. 7.3
IEEE 14-bus test model with AGC

power plants 2 and 3. The participation factors of these plants and their generators in the secondary frequency control are shown in Table 7.1.

For example, the participation factor of power plant 2 in the secondary frequency control is 0.6 pu, while the participation factors of generators 21 and 22 are 0.65 and 0.35 pu, respectively (their sum equaling one). The nominal active power of each generator is 300 MW.

By multiplying the participation factors of the power plant α_i and generators β_{ij} , the total participation factors of the generators γ_{ij} (at the system control level of the AGC in IPS 2) have been calculated and are shown in Table 7.2. For example, the total participation factor of the generator 21 γ_{21} is calculated as $0.6 \times 0.65 = 0.39$. The remaining total participation factors have been defined similarly.

Table 7.1
Participation factors of the power plants and generators of IPS 2 in the secondary frequency control

Element	Participation factor (pu)
Power plant 2	0.6
Generator 21	0.65
Generator 22	0.35
Power plant 3	0.4
Generator 31	0.45
Generator 32	0.45

Table 7.2 Total participation factors of the generators in the secondary frequency control at the system control level of the AGC in IPS2

Element	Total participation factor (pu)
Generator 21	0.39
Generator 22	0.21
Generator 31	0.18
Generator 32	0.22

As seen in Table 7.2, the sum of the total participation factors equals one.

AGC model implementation in PowerFactory software. The AGC model has been developed by using *DigSILENT simulation language*. Let us consider generalized steps of the AGC model development in PowerFactory. Since AGC model in IPS 2 includes all of the elements of IPS 1, we shall consider the model implementation of AGC in IPS 2 only.

Step 1. *Creating of the AGC Composite Frame.* In order to define the connections between the inputs and outputs of the AGC model elements, the AGC Composite Frames in IPS 1 and IPS 2 have been developed. The diagram of the AGC composite frame in IPS 2 is presented in Fig. 7.4.

This frame contains three power flow and one frequency measurement slots plus AGC controller slot (see Fig. 7.4). The power flows P_{flow1} ,

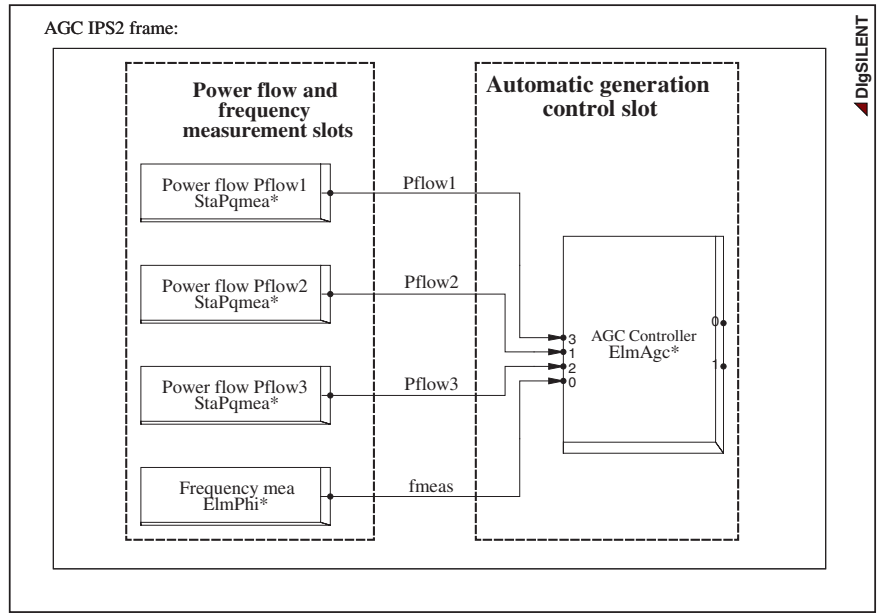


Fig. 7.4 The composite frame of AGC in IPS 2

$P_{\text{flow}2}$, $P_{\text{flow}3}$ through OHL 1–2/1, 1–2/2, 1–5 and the frequency f_{meas} at Bus 5 are measured (see Fig. 7.3). In composite model of AGC in IPS 1, the frequency at Bus 1 is measured only.

Step 2. Creating the Block Definitions of AGC models. Using *Standard Macros* of PowerFactory global *Library* and considering the Eqs. (7.1)–(7.10), the *Block Definitions* of the AGC in IPS 1 and IPS 2 have been created. The Block Definition of AGC in IPS 2 contains the following subblocks (see Fig. 7.5).

Frequency deviation calculation. In this subblock, the frequency deviation df and the frequency component $K_{\text{bias}} df$ of ACE are calculated according to Eqs. (7.2) and (7.4).

Calculation of the net interchange power deviation. In this subblock, on basis of tie-line interchanges $P_{\text{flow}1}$, $P_{\text{flow}2}$, and $P_{\text{flow}3}$, the net interchange power P_{net} is defined as follows:

$$P_{\text{net}} = P_{\text{flow}1} + P_{\text{flow}2} + P_{\text{flow}3}$$

Further, the deviation of the net interchange power dP_{net} from its reference value P_{netref} (see Fig. 7.5) is calculated in per units:

$$dP_{\text{net}} = \frac{P_{\text{net}} - P_{\text{netref}}}{P_{\text{base}}}$$

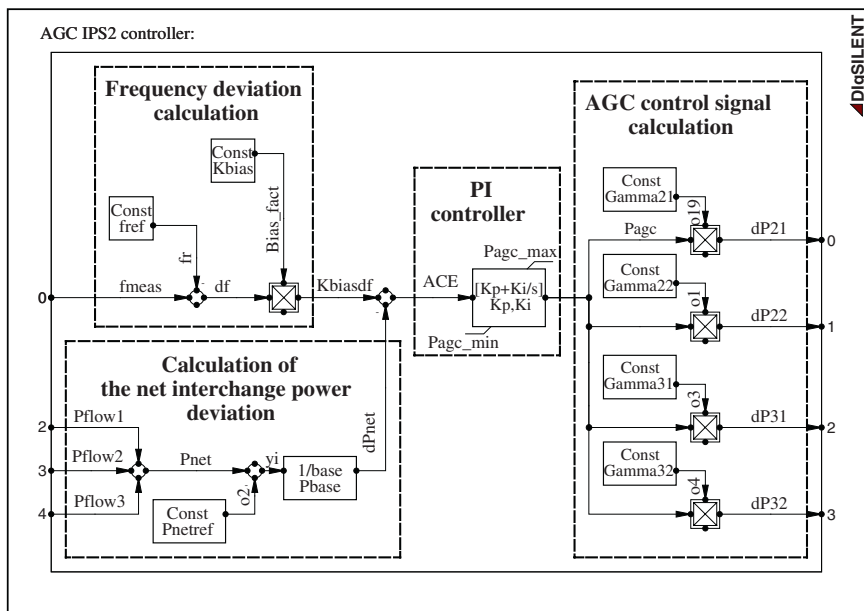


Fig. 7.5 Block Definition of the AGC in IPS 2 in PowerFactory

where

P_{base} is a base power equaling the total nominal active power of the generators in IPS 1 and IPS 2

Taking into account the signs of $K_{\text{bias}} df$ and dP_{net} (see Fig. 7.5), the ACE in IPS 2 is calculated as follows:

$$\text{ACE} = -dP_{\text{net}} + K_{\text{bias}} df$$

PI controller. In this subblock, the AGC active power setting P_{agc} is calculated considering the minimum $P_{\text{agc_min}}$ and maximum $P_{\text{agc_max}}$ limits of the output signal. The PI controller is presented by the following equations:

$$P_{\text{agc}.P} = K_P \text{ACE}$$

$$P_{\text{agc}.I} = K_I \int_{t_1}^{t_2} \text{ACE} dt$$

$$P_{\text{agc}} = P_{\text{agc}.P} + P_{\text{agc}.I}$$

$$P_{\text{agc.min}} \leq P_{\text{agc}} \leq P_{\text{agc.max}}$$

where

$P_{\text{agc}.P}$ is a proportional component of the PI controller output signal P_{agc} ;

$P_{\text{agc}.I}$ is an integral component of the PI controller output signal P_{agc} .

Using *PowerFactory DlgSILENT simulation language*, the initial condition can be written in the following way:

$$\text{inc}(x) = 0$$

where

x is a state variable of the PI controller

AGC control signal calculation. In this subblock, the PI controller output signal P_{agc} is distributed among the regulating units as follows (see Fig. 7.5):

$$dP_{ij} = \text{Gamma}_{ij} P_{\text{agc}} \quad i = 2, 3; j = 1, 2$$

where

dP_{ij} is the control signal formed by AGC in IPS 2;

Gamma_{ij} is the total participation factor in secondary frequency control (see Table 7.2)

Further, the control signals dP_{ij} from AGC in IPS 2 are transmitted to the governor of the j -generator of i -power power plant participating in the secondary frequency control.

- Step 3. *Definition of the AGC Common Models and measurements points.* On basis of developed *Block Definitions*, the *AGC Common Models* with a set of parameter values have been created. The active power flows are measured by *PQ Measurement* elements (which belong to a Class called *StaPqmea*) on OHLs of the interface between IPS 1 and IPS 2, while *Phase Measurement Devices PLL-Type* (Class called *ElmPhi_pll*) are used for the frequency measurement.
- Step 4. *Creating the Composite Models of AGC in IPS 1 and IPS 2.* Using the developed *AGC Composite Frames* (see *Step 1*), the *Composite Models* of AGC in IPS 1 and IPS 2 have been created, in which the *AGC Common Models* and measurement devices are combined.
- Step 5. *AGC model verification for the different disturbances.* One of the important stages of the model development is its validation for different disturbances. In order to validate the developed AGC model, the frequency control has been simulated using the built-in *Power-Frequency Controller* (load flow modeling) and proposed AGC model (RMS simulation).

7.4 Secondary Control Modeling by Load Flow Command

In this chapter, two cases of disturbances are studied. Case A corresponds to the additional load increasing at Bus 5, while in Case B, the change of the net interchange power set point from -90.25 to -110 MW in AGC of IPS 2 is considered.

The initial load of the individual generator in IPS 2 is 150 MW, generator 1 in IPS 1 being a reference machine. In order to model the load flow according to the secondary frequency control, power-frequency controllers for IPS 1 and IPS 2 need to be created as follows:

- Step 1. Activate the project “Automatic generation control start” (in the main menu, select **File**→**Activate Project...**, and in the displayed dialog window, choose the project and press the **OK** button).
- Step 2. Create a power-frequency controller of IPS 1 (open *Data Manager* (press the *Open Data Manager...* icon on the left of the main toolbar), then click the *New Object* icon; in the *Elements* section of the *Element selection* dialog window, select the *Controller/Motor Driven Machines* radio button. In the *Controller Type* pop-up window, select *Secondary Controller*; in the *Element* pop-up window select, *Power-Frequency Controller (ElmSecctrl)* and press the **OK** button).
- Step 3. Define the control mode of the power-frequency controller in IPS 1 (in the *Power-Frequency Controller* dialog window, select *Load Flow* tab, in the *Control Mode* pop-up window select *Frequency Control*).

- Step 4. Define Bus 1, at which the frequency in IPS 1 is measured (click on the arrow down button and choose the *Select...* command. In the displayed *Please select Busbar/Terminal* dialog window, select Bus 1 and press the **OK** button).
- Step 5. Define the generator participating in the frequency control in IPS 1 (in the *Active Power Distribution* section of *Power-Frequency Controller* dialog window, choose the *Individual Active Power* radio button. In order to define the regulating generator in IPS 1 right, click on the *Machines ElmSym,ElmGenStat* cell, and from the context-sensitive menu, select *Select Element/Type....* Further, in the displayed dialog window, select *Generator 1* and press the **OK** button).
- Step 6. Create the *Power-Frequency Controller* in IPS 2 (see Step 2).
- Step 7. Define the control mode of the power–frequency controller in IPS 2 (in *Power-Frequency Controller* dialog window, select the *Load Flow* tab; then in the *Control Mode* pop-up window, select *Power-Frequency Control*).
- Step 8. Define Bus 5, at which the frequency in IPS 2 is measured (how to define a busbar for frequency measurement see Step 4).
- Step 9. Define the *Boundary* in IPS 2 (check the *Boundary* radio button, click on the arrow down button and choose the *Select...* command. In the displayed *Please Select Boundary* dialog window, select *Net interchange power* boundary and press the **OK** button. Define *Power Set-point* in the AGC of IPS 2 ($P_{\text{net.ref}} = -90.25$ MW) and *Frequency Bias* ($K_{\text{bias}} = 300$ MW/Hz)). For the information on how to create the *Boundary*, refer to Chap. 13 of the PowerFactory User Manual.
- Step 10. Define the generators participating in the secondary frequency control in IPS 2 (see Step 5). In the displayed dialog window of the generator selection, choose the generators of power plants 2 and 3.
- Step 11. Define the participation factors of these generators in the secondary frequency control (double click on the *Active Power Percentage %* cell of the appropriate generator, define the total participation factor γ_{ij} for each of the generators (see Table 7.2), and then press the **OK** button). Let us calculate the initial load flow and the one set according to the secondary frequency control for Cases A and B.
- Step 12. Calculate the initial load flow (press the *Calculate Load Flow* icon on the main toolbar, check the *Consider Voltage Dependency of Loads* of *Load Options* section. Check also that in the *Active Power Control* tab of the *Load Flow Calculation* dialog window, the *as Dispatched* radio button is checked and then press **Execute** button).
- Step 13. Case A—switch on the additional load by clicking *Add load* switch at Bus 5. Case B—define the new set point value (–110 MW) of the net interchange power in *Power-Frequency Controller* of IPS2 (see Step 9).
- Step 14. Calculate the load flow according to the secondary frequency control (see Step 12. Check that in the *Active Power Control* tab of the *Load*

Flow Calculation dialog window the *according to Secondary Control* radio button is checked—in this case, the above power–frequency controllers are considered).

The calculation results of the initial load flow and the one set according to the secondary control for Cases A and B are presented in Tables 7.3 and 7.4. The generation power change of the generating unit is calculated as

$$\Delta P_{\text{gen}} = P_{\text{gen}}^{\text{SFC}} - P_{\text{gen}}^{\text{IL}}$$

where

$P_{\text{gen}}^{\text{SFC}}$ is the generation power according to secondary frequency control;

$P_{\text{gen}}^{\text{IL}}$ is the generation power of the initial load.

Table 7.3 Calculation results of the initial load flow and the one set according to the secondary frequency control in the event of a power demand increase

Element	Initial generation (MW)	Generation after secondary frequency control (MW)	Generation power change (MW)	Calculated participation factor (pu)
Power plant 1	224.9	225.1	0.2	1.00
Generator 1	224.9	225.1	0.2	1.00
Power plant 2	300	325.6	25.6	0.6
Generator 21	150	166.6	16.6	0.65
Generator 22	150	159	9	0.35
Power plant 3	300	317.1	17.1	0.4
Generator 31	150	157.7	7.7	0.45
Generator 32	150	159.4	9.4	0.55
Total IPS 2	600	642.7	42.7	1.00

Table 7.4 Calculation results of the initial load flow and the load flow after secondary frequency control in the event of the net interchange power set point change in power–frequency controller of IPS2

Element	Initial generation (MW)	Generation after secondary frequency control (MW)	Generation power change (MW)	Calculated participation factor (pu)
Power plant 1	224.9	245.4	20.5	1.00
Generator 1	224.9	245.4	20.5	1.00
Power plant 2	300	287.8	−12.2	0.6
Generator 21	150	142.1	−7.9	0.65
Generator 22	150	145.7	−4.3	0.35
Power plant 3	300	291.8	−8.2	0.4
Generator 31	150	146.3	−3.7	0.45
Generator 32	150	145.5	−4.5	0.55
Total IPS 2	600	579.6	−20.4	1.00

The calculated participation factor of the generator in the secondary frequency control is defined as follows

$$\beta = \frac{\Delta P_{\text{gen}}}{\Delta P_{\text{PP}}}$$

where

ΔP_{PP} is the generation power change of the appropriate power plant

Accordingly, the participation factor of the power plant in the secondary frequency control α is calculated as

$$\alpha = \frac{\Delta P_{\text{PP}}}{\Delta P_{\text{IPS}}}$$

where

ΔP_{IPS} is the total generation power change in IPS 1 and IPS 2

As seen in Tables 7.3 and 7.4, the participation factors of the power plants and generators in the secondary frequency control are fully consistent with the initial ones shown in Table 7.1.

7.5 Transient Simulation of the Frequency Control in Power Systems

In this section, using the developed AGC model, the simulation is performed in the event of the additional load increasing (Case A) and power set point change from -90.25 to -110 MW in AGC of IPS 2 (Case B). The participation of the generators in the secondary frequency control is defined by the total participation factors shown in Table 7.2. In order to simulate the frequency control, it is necessary to perform the following steps.

- Step 1. Select the disturbance for the simulation (press the *Calculate Initial Conditions* icon on the main toolbar. In the *Events* pop-up menu of the *Selection of Simulation Events* section, choose *Select....* In the displayed dialog window, select the event corresponding to the appropriate case study and press the **OK** button). For more information on how to define the simulation events in PowerFactory, refer to Chap. 25 of PowerFactory User Manual.
- Step 2. Check the options for the initial load flow calculation (click the right arrow of *Load flow* in the *Calculation of Initial Conditions* dialog window. In the displayed *Load Flow Calculation* dialog window, check the option *Consider voltage dependency of loads* and press the **Close** button).

- Step 3. Calculate the initial conditions (press the **Execute** button of the *Calculation of Initial Conditions* dialog window).
- Step 4. Simulate the frequency control for a given disturbance (press the *Start Simulation...* icon on the main toolbar, define the simulation time and press **Execute** button).
- Step 5. The simulation results are displayed in *Test network* as well as the *Frequency*, *Generation power*, *OHL power flow*, and *Net interchange power* tabs. If necessary, the user can define the additional variables to plot in the Virtual instruments panel.

As seen, the participation factors for the transient simulation (see Tables 7.5 and 7.6) are fully consistent with the initial ones presented in Table 7.1. The obtained simulation results for Cases A and B are depicted in Figs. 7.6 and 7.7.

Table 7.5 Simulation results for the initial load flow and the load flow after the secondary frequency control in the event of a power demand increase in IPS2

Element	Initial generation (MW)	Generation after secondary frequency control (MW)	Generation power change (MW)	Calculated participation factor (pu)
Power plant 1	224.9	225	0.1	1.00
Generator 1	224.9	225	0.1	1.00
Power plant 2	300	325.2	25.2	0.6
Generator 21	150	166.4	16.4	0.65
Generator 22	150	158.8	8.8	0.35
Power plant 3	300	316.9	16.9	0.4
Generator 31	150	157.6	7.6	0.45
Generator 32	150	159.3	9.3	0.55
Total IPS 2	600	642.1	42.1	1.00

Table 7.6 Simulation results for the initial load flow and the load flow after the secondary frequency control in the event of the net interchange power set point change in AGC of IPS2

Element	Initial generation (MW)	Generation after secondary frequency control (MW)	Generation power change (MW)	Calculated participation factor (pu)
Power plant 1	224.9	245.3	20.4	1.00
Generator 1	224.9	245.3	20.4	1.00
Power plant 2	300	287.8	-12.2	0.6
Generator 21	150	142.1	-7.9	0.65
Generator 22	150	145.7	-4.3	0.35
Power plant 3	300	291.8	-8.2	0.4
Generator 31	150	146.3	-3.7	0.45
Generator 32	150	145.5	-4.5	0.55
Total IPS 2	600	579.6	-20.4	1.00

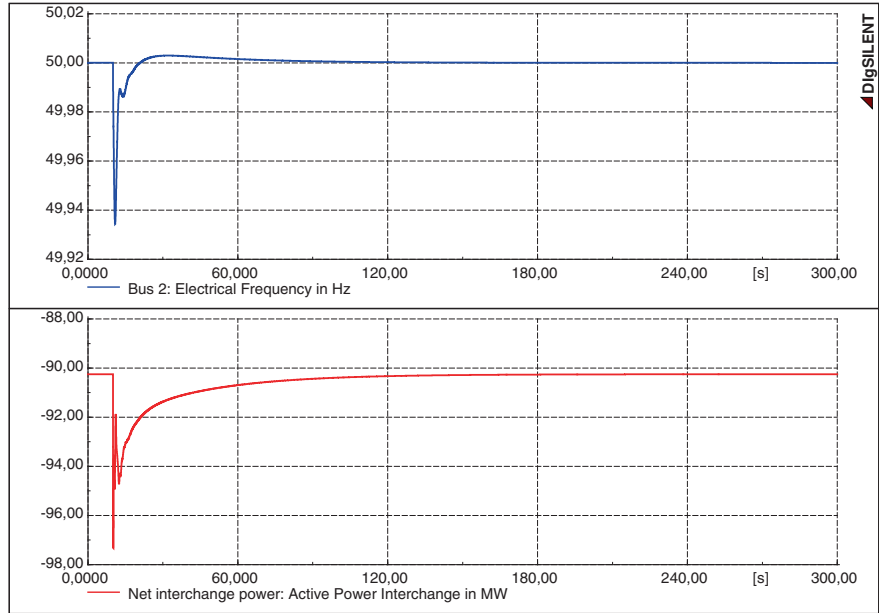


Fig. 7.6 Simulation results—frequency and net interchange power (demand increase in IPS2)

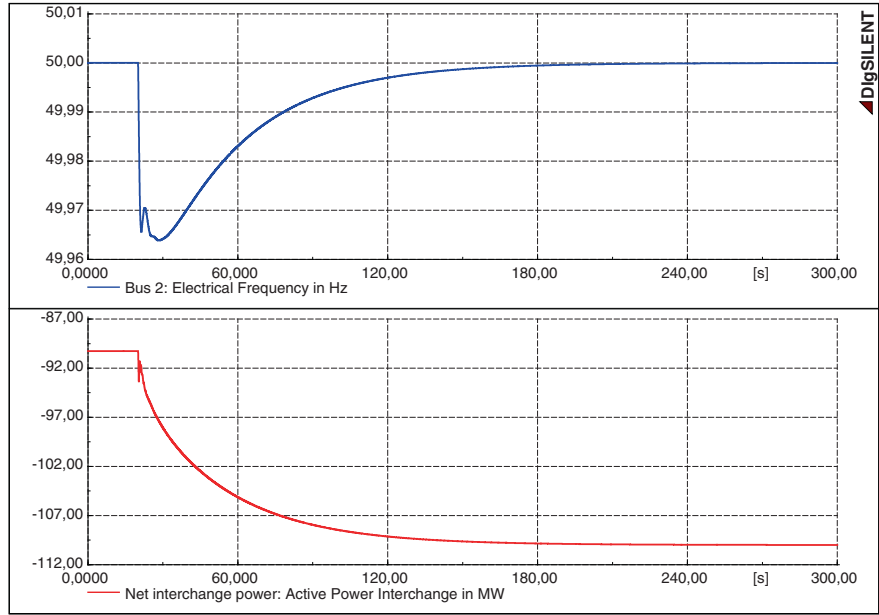


Fig. 7.7 Simulation results—frequency and net interchange power (net interchange power set point change in AGC of IPS2 to -110 MW)

It should be noted that the detailed AGC model of the IPS of Ukraine has been developed by DMCC Engineering Ltd. It is used to study the impact of the wind and solar power plants on the frequency control in the Ukrainian IPS [7, 8].

References

1. Windtech International (2014) Understanding active power control from wind power. Available from <http://www.windtech-international.com/articles/understanding-active-power-control-from-wind-power>
2. Hasan N (2012) An overview of AGC strategies in power system. *Int J Emerg Technol Adv Eng* 2(8):56–64
3. Arya Y, Kumar N, Mathur HD (2012) Automatic generation control in multi area interconnected power system by using HVDC links. *Int J Power Electron Drive Syst* 2(1):67–75
4. P1—Policy 1: Load-frequency control and performance. Available from https://www.entsoe.eu/fileadmin/user_upload/_library/publications/entsoe/Operation_Handbook/Policy_1_final.pdf
5. Kundur P (1994) *Power system stability and control*. McGraw Hill Inc, New York, 1173 p
6. Machowski J, Bialek JW, Bumby JR (2008) *Power system dynamics: stability and control*, 2nd edn. Wiley, New York, 630 p
7. Kyrylenko O, Pavlovsky V, Steliuk A, Lukianenko L (2012) Simulation of the automatic generation control in the interconnected power system of Ukraine. *Tech Electrodyn* 6:44–50
8. Kyrylenko O, Pavlovsky V, Steliuk A (2014) AGC software model validation for identification of renewables impact on frequency control in the IPS of Ukraine. In: *Proceedings of IEEE international conference on intelligent energy and power systems*, pp 141–144

Chapter 8

Gas Turbine Modelling for Power System Dynamic Simulation Studies

Lasantha Meegahapola and Damian Flynn

Abstract Gas turbines possess unique frequency response characteristics, as compared to other conventional (synchronous) generation technologies, which can significantly influence power system dynamics and stability during disturbances in the network. Many power systems are seeing increasing penetrations of combined cycle gas turbines (CCGTs), along with variable renewables, such as wind and solar generation, so an improved understanding of gas turbine characteristics during high- and low-frequency events and network faults is required in order to mitigate potential adverse system impacts. In particular, the impact of ambient conditions on unit rating, the temperature control system, gas turbine compressor intake, and associated components such as inlet guide vanes (IGVs) must be accurately modelled, particularly when operating at high output, in order to accurately assess the impact of CCGTs on stability issues. Therefore, this chapter aims to provide a comprehensive guide to developing a CCGT model using the DiGSILENT dynamic simulation language (DSL), including model initialisation. In addition, modelling of an open cycle gas turbine (OCGT) will be presented. Finally, a dynamic comparison of the CCGT and OCGT models will be examined using a demonstration test system.

Electronic supplementary material The online version of this chapter (doi:10.1007/978-3-319-12958-7_8) contains supplementary material, which is available to authorized users.

L. Meegahapola (✉)

School of Electrical, Computer and Telecommunications Engineering, Faculty of Engineering and Information Sciences, University of Wollongong, Northfields Avenue, Wollongong 2500, Australia
e-mail: lasantha@ieee.org

D. Flynn

Department of Electrical, Electronic and Communications Engineering, University College Dublin, Belfield, Dublin 4, Ireland
e-mail: damian.flynn@ucd.ie

Keywords Combined cycle gas turbines (CCGT) • Dynamic simulation language (DSL) • Frequency stability • Open cycle gas turbines (OCGT)

8.1 Introduction

During the last three decades, combined cycle gas turbines (CCGTs) have received immense industry attention as one of the economically viable and environmentally friendly thermal power generation technologies due to their high efficiency, flexibility, and low greenhouse gas emissions. In parallel with these developments, renewable power generators have been widely integrated to thermal power networks due to increased pressure to reduce greenhouse gas emissions from power generation [1]. However, the highly variable and uncertain output from non-conventional renewable power generators (i.e. wind and solar power generation) has resulted in challenging issues during system operation. Consequently, open cycle gas turbines (OCGTs) have become a viable fast-start generation technology, with many power networks, operating with a significant penetration of renewable power generation, deploying OCGTs to effectively manage renewable power variability during system operation.

The dynamic response characteristics of gas turbines during frequency excursions have been discussed by a number of researchers as they may adversely impact on system frequency stability [2–6]. During low-frequency events, the gas turbine (both CCGT and OCGT) power output reduces due to the reduction in the pressure ratio across the gas turbine: a result of the reduced airflow to the combustion chamber as the grid synchronised compressor slows down during the low-frequency event. In addition, the dynamics associated with the temperature control scheme of the CCGT also lead to a further reduction in unit output during low-frequency events by reducing the fuel flow to the combustion chamber. Various system operators have set operational guidelines to alleviate such adverse impact from CCGTs during low-frequency events [7]. In addition, during high-frequency events, such as short-circuit faults and sudden load reduction events, the gas turbine compressor speeds up, as it is synchronised with the grid, which will ultimately force more air into the combustor, and the gas turbine may be prone to a lean blowout (LBO) condition [8, 9]. Therefore, power utility engineers require an in-depth understanding of the gas turbine dynamics during both low- and high- frequency events.

8.2 Modelling of a Combined Cycle Gas Turbine Plant

A CCGT plant is comprised of a gas turbine, a heat recovery steam generator (HRSG), and a steam turbine (ST) unit, with the exhaust gases from the gas turbine extracted to generate steam for the ST unit. CCGTs have two typical configurations:

(i) single-shaft and (ii) multi-shaft configuration. In a single-shaft arrangement, the air compressor, the gas turbine, and the ST are connected to the same shaft, while in the multi-shaft design the steam turbine is driven by a separate shaft from that of the main gas turbine shaft. The former single-shaft arrangement provides slightly higher unit efficiency, while the latter allows multiple gas turbines and/or steam turbines to be connected together for increased unit flexibility.

A schematic diagram of a multi-shaft CCGT plant is illustrated in Fig. 8.1: air at atmospheric pressure and temperature conditions is drawn into the compressor, where the air temperature and pressure is raised, before being mixed with the fuel in the combustion chamber. The resulting hot exhaust gas flow is then expended through the turbine while driving the generator and the compressor shaft.

A number of studies have been conducted on gas turbine modelling for dynamic and stability studies [10–15]. The work presented by Rowen [10] was one of the pioneering studies in the early literature, and subsequently that model was further improved by including variable inlet guide vanes (IGVs) to control the airflow to the combustion chamber. An IEEE working group has presented a dynamic model

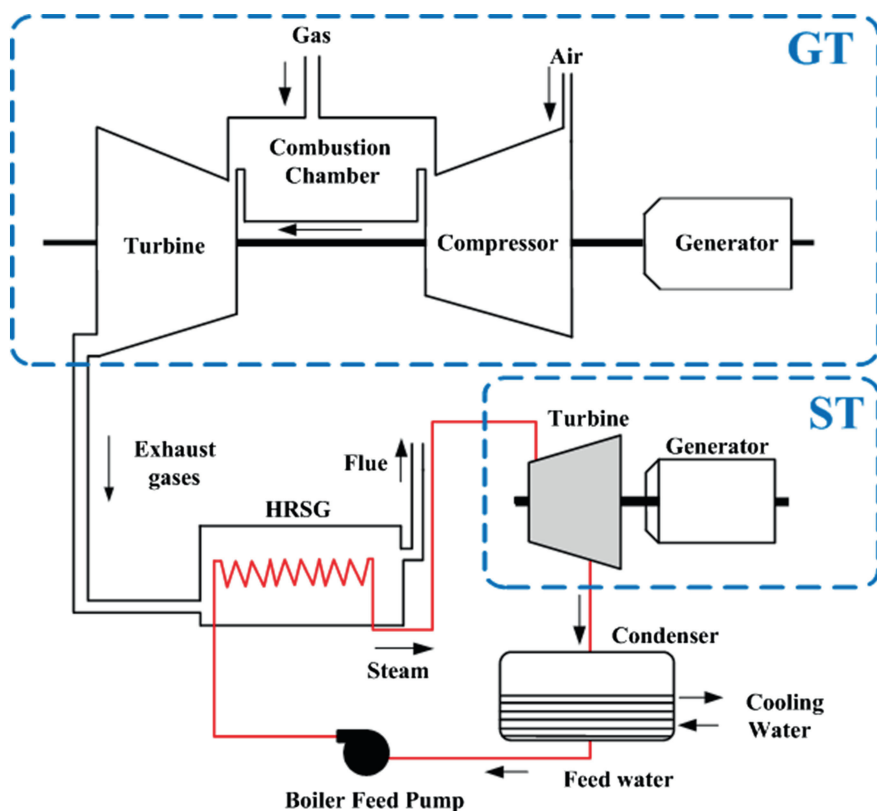


Fig. 8.1 The layout of the multi-shaft CCGT plant

for a single-shaft gas turbine with IGVs while considering physical thermodynamic properties and laws [11]. In [12, 13], the authors placed a special emphasis on modelling gas turbines for frequency stability studies and developed a physical model to accurately determine the gas turbine power output during various frequency excursions. A CIGRE working group has also developed a gas turbine model for power system dynamic studies based on Rowen's model; however, the turbine dynamics were represented by a second-order block in contrast to the mathematical function in Rowen's model for the torque calculation [14]. A range of modelling concepts has been applied by researchers to model CCGTs for power system dynamic studies. For example, in the IEEE model, the dynamics of the CCGT plant are represented through physical thermodynamic properties/laws associated with the turbine, combustor and various control loops [11]. However, Rowen's model uses mathematical functions to represent the dynamics associated with the turbine and combustor [10]. The accuracy of Rowen's model for evaluating the system frequency response of a CCGT has been verified in [4]. Comparisons of various gas turbine dynamic models, including parameter estimation and validation procedures, are presented in [16–18].

8.3 Dynamic Modelling of a CCGT Plant

A number of components are required when developing a CCGT plant in DIgSILENT PowerFactory (i.e. synchronous generator, prime mover and governor, and excitation system), which can be combined through a *Composite Frame*, whereby each component is represented by a *block*. A schematic of a *Composite Frame* for a generation plant is shown in Fig. 8.2

Where ve , pt , $xspeed$, ut , $upss$, fe , $cosn$, and $sgnn$ denote excitation voltage (pu), turbine power (pu), generator speed (pu), generator terminal voltage (pu), power system stabiliser output (pu), electrical frequency (pu), nominal power factor, and nominal apparent power (MVA), respectively. Specific dynamics related to the CCGT are built into the governor and prime mover *block* ($ElmPcu^*$), which determines the turbine power output (pt). As this chapter mainly focuses on developing the prime mover model for the CCGT plant, modelling of the excitation system, synchronous generator, and power system stabiliser will not be discussed. The model presented is mainly based on Rowen's CCGT model for frequency dynamic studies [4], and a schematic diagram of the CCGT prime mover model is shown in Fig. 8.3

Where P_{step} , $tempc$, $temof$, $temigv$, WX , and $combout$ denote active power set point (pu), temperature control system signal, temperature reference (°F), exhaust gas temperature for VIGV control (°F), corrected airflow based on VIGV position, and gas flow after combustion, respectively. The CCGT model is comprised of two main control loops (i.e. speed control and temperature control). The temperature control loop can be divided into two components: (i) temperature measurement system and (ii) airflow control system using IGVs. For reasons of clarity, the CCGT

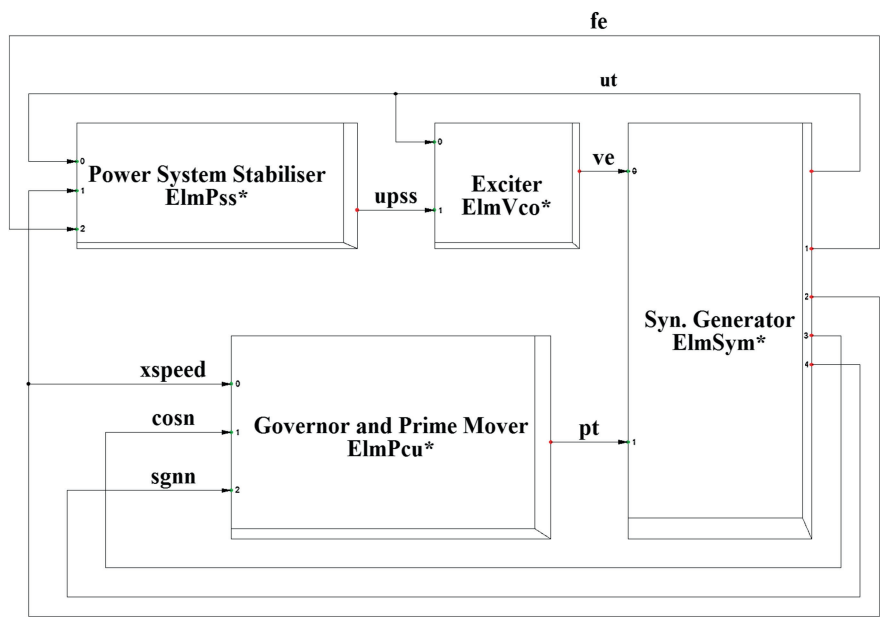


Fig. 8.2 The Composite Frame of a generation plant

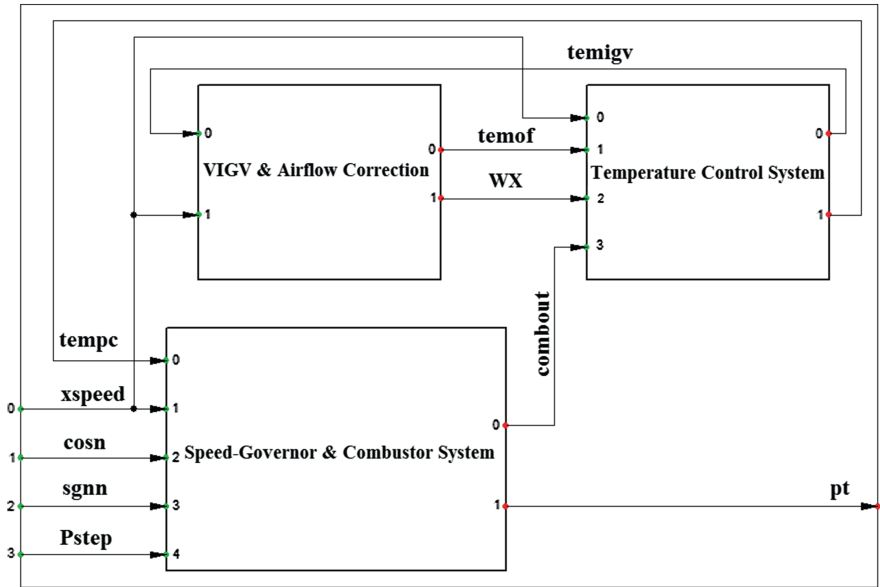


Fig. 8.3 The CCGT speed governor and prime mover model

Table 8.1 Parameter values of the CCGT model [4, 8]

Parameter	Value
Governor droop (K_{drp})	25
Speed controller time constant (T_{drp})	0.05 s
Fuel control delay (T_{fd})	0.0625 s
Valve positioner time constant (T_v)	0.05 s
Fuel system delay (T_f)	0.4 s
Maximum fuel limit (fl_max)	0.7692 pu
Minimum fuel limit (fl_min)	0.15 pu
Combustion reaction delay (E_{cr})	0.01 s
Compressor discharge time constant (T_c)	0.1 s
Turbine exhaust transport delay (E_{id})	0.01 s
Temperature controller time constant (T_t)	450 s
Temperature controller gain (K_{tc})	3.3
Thermocouple time constant (T_{tr})	0.5 s
Radiation shield prop. factor (K_{rs})	0.8
Radiation shield integrator factor (K_{ri})	0.2
Radiation shield time constant (T_{rs})	15 s
IGV controller time constant (T_{d})	20 s
IGV controller gain (K_{i})	4
IGV actuator time constant (T_{i})	3 s
IGV maximum (ig_max)	1
IGV minimum (ig_min)	0.46
Rated exhaust temperature (TC)	950 °F

prime mover model is presented in three sections, namely speed governor and combustor system, temperature control system, and VIGV and airflow control system. However, it should be noted that the model can be designed as a single *block*. The parameter values of the CCGT prime mover model are based on [4, 10] and are given in Table 8.1.

The speed governor model and the combustor system model, developed in DiGSILENT PowerFactory, are shown in Fig. 8.4 and the parameter values are given in Table 8.1.

The generator speed (ω) is compared with the nominal system speed (ω_{ref}), with the resulting error being fed to the speed controller, which is represented by a first-order transfer function. The CCGT is assumed here to have a droop governor whose output is proportional to the speed error, however, if an isochronous governor is employed a proportional-integral (PI) controller will instead be appropriate. The following DSL code is employed in the block definition (BlkDef) of the speed governor block in order to determine the droop governor response:

```
limits(T_drp)=[0,)           !define limit for the time constant
x.=select(T_drp>0,(K_drp*yi-x)/T_drp,0)!define the derivative of state-var.
yo=select(T_drp>0,x,K_drp*yi) !define the output of the block
```

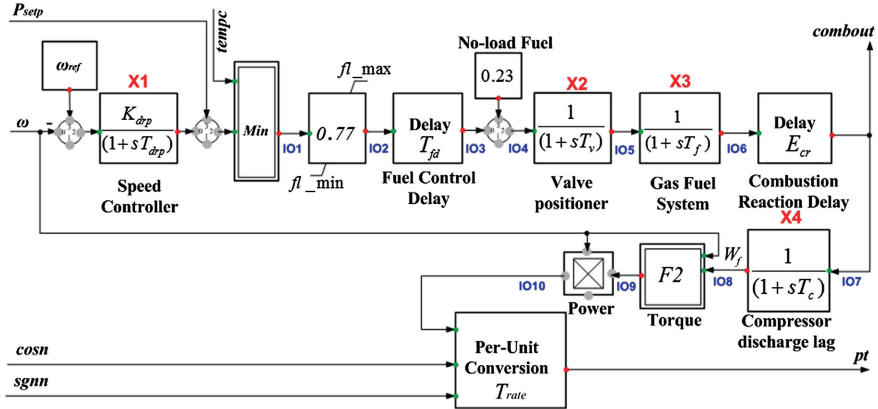



Fig. 8.4 The speed governor and combustor system

It should be noted that a standard notation of y_i and y_o to denote the input and output of a block definition has been used throughout this chapter. The load reference set point (P_{setp}) is added to the speed governor response to determine the fuel demand requirement for steady-state operation. It must be noted that the P_{setp} can be further modified based on the ambient temperature and the pressure variations as the gas turbine power output is influenced by ambient conditions [4]. However, that was not included in the model shown in Fig. 8.4. For gas turbines, however, the fuel requirement does not solely depend on the speed governor response, as the GT exhaust temperature must be maintained at an optimal value in order to achieve the highest plant efficiency. Therefore, a separate control system is also in place to control the temperature of the exhaust gases. The minimum output between the governor and temperature control signals is selected to determine the fuel demand requirement for the combustor at the minimum function selector block (Min), before being scaled by a constant subjected to fuel limits. The following DSL code is implemented at the *block reference*:

```
yo=lim(0.77*yi,fl_min,fl_max) !setting the fuel limits
```

The fuel system has a control delay (T_{fd}) to respond to the fuel demand variation, which is incorporated in the fuel demand signal using the *block definition* of the fuel control delay, defined as follows:

```
yo= delay(yi,T_fd) !apply a delay for the output
```

Then, the no-load fuel requirement is added to the fuel demand signal, and the combined output is communicated to the valve positioner to move the valve actuator to release the required fuel quantity to the combustor. In liquid fuel

systems, a bypass valve is also present, which will not be considered here. The fuel valve positioner is represented by a first-order transfer function with a time constant (T_v). Although the valve directly modulates the fuel flow to the combustor, transport time lags present in the fuel system pipework are represented by a first-order transfer function with a time constant of (T_f). Once the fuel is injected to the combustor, there is a finitely small time period for heat release, which is denoted by a delay known as the combustor reaction delay. However, the time lag between the compressor discharge and turbine inlet is much longer and is represented by a first-order transfer function with the time constant (T_c).

After incorporating all the associated time lags in each sub-system, the turbine torque is calculated using the following function ($F2$):

$$F2 = 1.3 \times (W_f - 0.23) + 0.5 \times (1 - \omega) \quad (8.1)$$

where W_f denotes the exhaust gas flow from the turbine. The mechanical power output is calculated by multiplying the torque by the turbine shaft speed (ω), which is then normalised based on the generator active power rating (T_{rate}), generator rating ($sgnn$), and nominal power factor ($cosn$) of the synchronous generator. The normalised turbine mechanical power output (pt) is used as the mechanical power input of the synchronous generator. The DSL code for the normalising *block definition* is defined as follows:

```
!set the limit for the turbine power output &
!normalise the turbine power output
yo=lim(select(Trate>0,yi*Trate/((sgnn)*cosn),yi),0,1)
```

The CCGT exhaust temperature control system, Fig. 8.5, is comprised of a number of components, such as a thermocouple, radiation shield, and variable inlet guide vanes (VIGVs).

It should be noted again that for reasons of clarity, the airflow control system based on VIGVs is presented separately. A turbine and exhaust system time delay (E_{td}) is incorporated in the gas flow path from the combustor system prior to

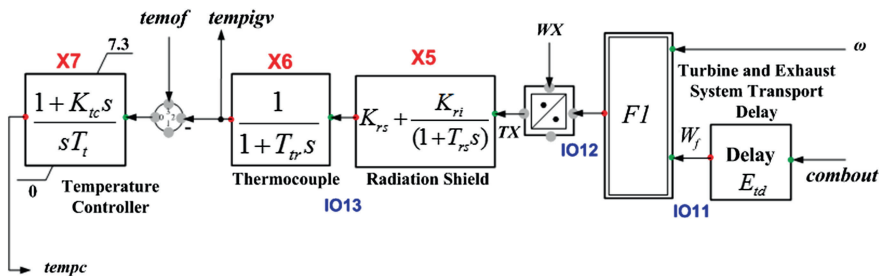


Fig. 8.5 The exhaust temperature measurement and control system

determining the exhaust gas temperature. The following function is utilised to determine the exhaust gas temperature:

$$F1 = (1/[1 + 0.005 \times (15 - T_a)]) \times \left\{ \begin{aligned} &[TC - 453 \times (\omega^2 - 4.21\omega + 4.42) \times 0.82 \times (1 - W_f)] \\ &+ 722 \times (1 - \omega) + 1.94 \times (\text{MaxIGV} - \text{IGV}) \end{aligned} \right\} \quad (8.2)$$

where T_a and TC denote the ambient temperature and rated turbine exhaust temperature, respectively. MaxIGV denotes the maximum angle of the VIGVs while IGV refers to the current VIGV angle.

In order to maintain the exhaust temperature from the gas turbine at the optimal value to achieve highest efficiency, CCGTs are equipped with VIGVs which control the airflow to the compressor. In addition, the air compressor is connected to the same shaft as the generator, thus, variations in the system frequency will affect the shaft speed and hence the airflow to the compressor. The corresponding airflow correction is denoted by WX in Fig. 8.5. Therefore, the turbine exhaust temperature is further modified based on the VIGV position and system frequency.

The temperature measurement system is comprised of two components, namely a thermocouple and radiation shield. The radiation shield is a polished and highly reflective metal sheet which wraps around the thermocouple to reflect radiation away from the thermocouple [18], and so, due to its heat transfer properties, will create a time lag in the measurement which must be incorporated. The DSL code for the radiation shield block is defined as follows:

```
limits(Trs)= (0,)      !define limit for the time constant
x.= (yi-x)/Trs         !define the derivative of state-var.
yo=(x*Kri+Krs*yi)     !define the output of the block
```

The thermocouple itself has its own time lag, which varies with the type and design, and is represented by a first-order transfer function. The measured temperature is compared against the reference exhaust temperature (recognising the overfiring capability of the CCGT), with the error signal enabling exhaust temperature correction based on fuel flow. The temperature controller is a PI controller, and the DSL code for the block definition is denoted as follows:

```
limits(Tt)=(0,)      !define limit for the time constant
x.= yi/T             !define the derivative of state-var.
!define limit for the output and state-variable
yo=lim(yox=limstate(x,0,7.3)+yi*Ktc/Tt,0,7.3)
```

Subsequently, the exhaust gas temperature can be controlled by adjusting the VIGV position and hence the airflow. The airflow control system based on VIGVs is shown in Fig. 8.6 and parameters are given in Table 8.1.

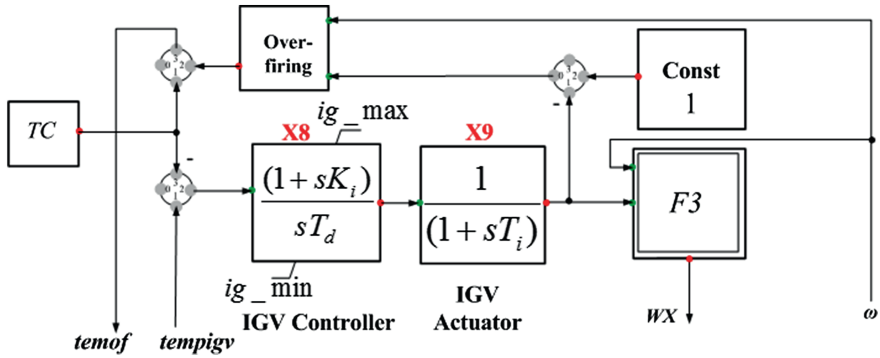


Fig. 8.6 The VIGV and airflow control system

The temperature error between the measured exhaust temperature and the reference exhaust temperature is utilised at the VIGV controller to determine the IGV position. Changing the VIGV position corresponds to a change in the incidence angle at the inlet of the impeller. The VIGV controller is a PI controller with minimum and maximum limits. The DSL code of the VIGV controller block is shown below:

```
limits(T_d)=(0,) !define limit for the time constant
x.= yi/T_d      !define the derivative of state-var.
!define limit for the output and state-variable
yo=lim(yox=limstate(x,ig_min,ig_max)+yi*K_i/T_d,ig_min,ig_max)
```

The VIGV actuator also incorporates a time lag, modelled as a first-order lag. Airflow correction is then performed, to obtain the corrected airflow (WX), based on the VIGV position, $Ligv$, and system frequency, as given by the following function:

$$F3 = \omega \times [519 / (Ta + 460)] \times (Ligv)^{0.257} \quad (8.3)$$

It should be further noted that CCGTs are equipped with an overfiring capability, hence the rated exhaust temperature must be corrected based on the temperature increase corresponds to overfiring capability must be added to the temperature reference signal (i.e. $temof$). Representative parameter values with regard to speed governor and combustor system (Fig. 8.4), exhaust temperature measurement and control system (Fig. 8.5), and VIGV airflow control system (Fig. 8.6) are listed in Table 8.1.

The exhaust gases from the gas turbine feed the heat recovery steam generator (HRSG) to exploit the heat and generate steam, dependent on the gas turbine exhaust flow (W_f) and exhaust gas temperature ($tempigv$). It must be noted that the ST drives a separate synchronous generator in a multi-shaft configuration which then also requires a *Composite Frame*, similar to Fig. 8.2. A schematic of the ST model is shown in Fig. 8.7.

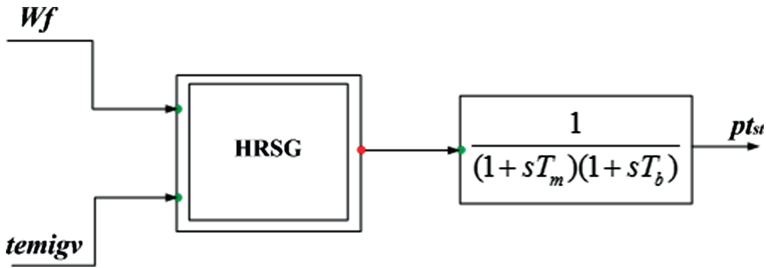


Fig. 8.7 The steam turbine model

The mechanical power output (pt_{st}) of the ST depends on the gas turbine exhaust energy, and the time lags associated with the high-pressure and low-pressure sections of the steam turbine. These time lags correspond to the tube metal heat capacitance (T_m -order of 5 s) and the HRSG time constant (T_b -order of 50–100 s). The large time constants involved imply an insignificant effect on the overall plant response during the study period of interest (10 s) for frequency stability and transient stability applications. Furthermore, as the ST generator is typically operated in a sliding pressure mode, it does not provide any primary response.

8.4 CCGT Model Initialisation

The CCGT model is initialised based upon the generator(s) speed measurement (frequency measurement) and the gas turbine mechanical power output (pt). In Figs. 8.4, 8.5 and 8.6, the essential signals required for model initialisation are represented using the symbol IO, while the associated state variables in each block are represented by the symbol X. Once a network load flow calculation has been performed, the turbine power input required to deliver the active power output can be determined. In addition, the initial system is assumed to be in steady state with the initial generator shaft speed (ω) known. The state variable (X1) associated with the speed controller is initialised by multiplying the speed controller droop from the generator speed deviation, as follows:

```
inc(X1)=Kdrp*(1-w) !initialise based on the initial speed deviation
```

The initialisation routine, from this point onwards, continues from right to left (i.e. turbine power input to signal at the minimum value selector), as the power reference set point (P_{setp}) must be determined based on the mechanical power input to the generator. Therefore, based on the mechanical power input, *reverse* order calculations are performed to determine IO1 as follows:

```

!initialise the turbine power for a given mechanical power input
inc(I010)    = select(Trate>0,pt*(sgnn)*cosn/Trate,pt)
!initialise turbine torque based on calculated turbine power output
inc(I09)     = I010/w
!initialise exhaust gas-flow from the turbine torque
inc(I08)     = ((I09-0.5*(1-w))/1.3)+0.23
inc(I07)     = I08      !initialise combustor discharge flow
inc(x4)      = I07      !initialise compressor discharge state var.
inc(I06)     = I07      !initialise compressor gas-flow
inc(I05)     = I06      !initialise fuel into the combustor
inc(X3)      = I06      !initialise fuel system state variable
inc(I04)     = I05      !initialise fuel flow from the value
inc(X2)      = I04      !initialise state var. for fuel control value
inc(I03)     = I04-0.23 !initialise the fuel requirement from gov.
inc(I02)     = I03      !initialise fuel request
inc(I01)     = I02/0.77 !initialise gov. signal
inc0(psetp)  = I01      !initialise steady-state power output

```

The load reference set point (P_{setp}) denotes the generator steady-state fuel requirement, which can be determined based on IO1 (the steady-state operating fuel requirement). The initialisation routine for the temperature control system is defined as follows:

```

inc(I011)    = I07      !initialise gas flow after combustion
inc(Wx)      = w*(pow(0.095,0.257)) !initialise air-flow correction
inc(I013)    = Kri*x5+Krs*Tx !initialise exhaust gas temperature
inc(Tx)      = X5       !initialise air-flow corrected temperature
inc(X5)      = I012/WX  !initialise radiation-shield state-var.
inc(X6)      = I013     !initialise thermocouple state-var.
inc(X7)      = 7.3      !initialise temperature controller state var.

```

For the airflow control system, only the state variables are initialised, while the internal signals are automatically calculated during the initialisation process:

```

inc(X8)      = igv_min !initialise IGV controller state-var.
inc(X9)      = igva    !initialise IGV actuator state-var.

```

In order to verify that the model has been initialised correctly, it is recommended to complete a simulation for a single machine test system without triggering any fault and confirming that the turbine power output remains stable at the required value.

8.5 Dynamic Modelling of an OCGT Plant

Typically, the exhaust temperature of an OCGT is not controlled at an optimal value, since a heat recovery mechanism is not used during open cycle operation. However, both the CCGT and OCGT are assumed to utilise the same internal parameters for speed and temperature control loops, although the presence of

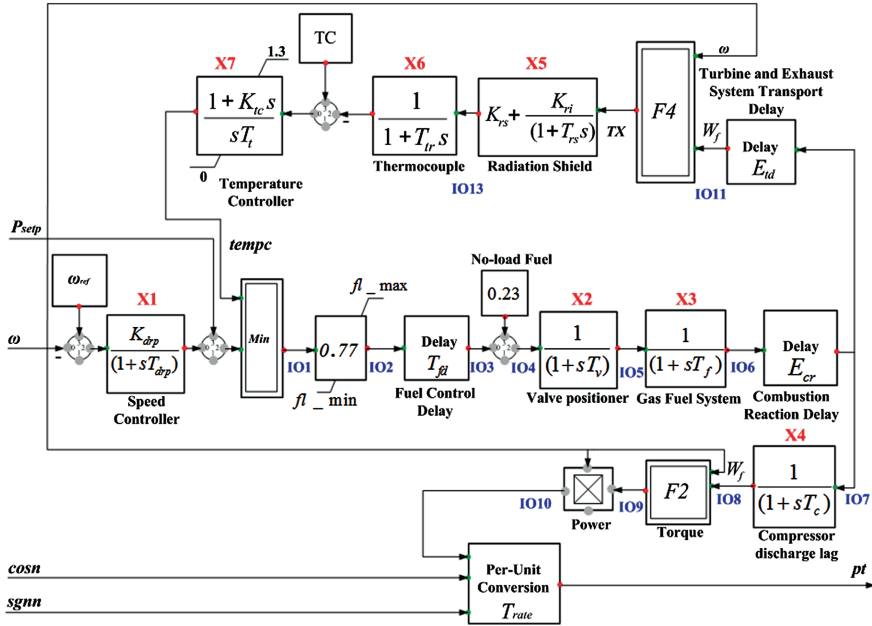


Fig. 8.8 OCGT model developed in DigSILENT PowerFactory

VIGVs and an overfiring capability for CCGTs distinguish the models from each other. Since OCGTs do not have variable VIGVs, the exhaust temperature is calculated using the following function [10]:

$$F4 = TC - 700(1 - W_f) + 550(1 - \omega) \quad (8.4)$$

However, the OCGT turbine power output is determined using the same function as for the CCGT (i.e. $F2$). The OCGT model also requires a *Composite Frame* similar to Fig. 8.2 and the OCGT is built into the speed governor and prime mover model. A schematic diagram of the OCGT model, developed in DigSILENT PowerFactory, is shown in Fig. 8.8. The OCGT model can also be initialised based on the routines defined for the CCGT model in Sect. 4.4, excepting those routines for the VIGV-based airflow controller.

The initialisation routine for the governor and combustor system is same as the CCGT however as the OCGTs are not equipped with IGVs, hence following initialisation routine is used for the OCGT temperature control system.

```
inc(I011)    = I07 !initialise gas flow after combustion
inc(f1)     = Tr-700*(1-I011)+550*(1-w) !initialise F1
inc(x5)     = f1 !initialise radiation-shield state-var.
inc(I013)   = Kri*x5+Krs*f1 !initialise exhaust gas temperature
inc(x6)     = I013 !initialise thermocouple state-var.
inc(x7)     = 1.3 !initialise temperature controller state var.
```

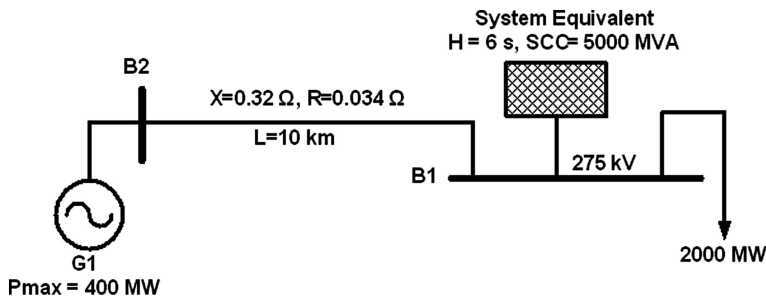


Fig. 8.9 Test system model for model validation

8.6 Case Study: Comparison of CCGT and OCGT Dynamics

8.6.1 Test Network System

A single machine test network was developed in order to investigate the characteristics of a CCGT and OCGT during system frequency excursions. The test system, Fig. 8.9, is comprised of a 2,000 MW load, represented by a constant impedance load model, connected to a 400 MW OCGT/CCGT generator, and a system equivalent, represented by an external grid model. The system equivalent has an inertia constant (H) of 6 s, a short-circuit capacity (SCC) of 5,000 MVA, and a governor frequency droop of 4 %.

The generator model was separately represented as a CCGT and as a OCGT, in order to avoid any dynamic interactions between both units. Furthermore, it is assumed that generator G1 is operating at zero reactive power output and that the CCGT has an inertial constant of 8 s and a rated power output of 400 MW, 500 MVA, while the OCGT has an inertial constant of 4 s and a rated power output of $2 \times 200 \text{ MW}$, $2 \times 250 \text{ MVA}$. The excitation model is based on the IEEE Type 1 excitation system model [19]. Both the CCGT and OCGT capability characteristics were determined based on typical transmission system operator (TSO) specifications [20, 21].

8.6.2 Gas Turbine Response During a Low-frequency Event

A low-frequency event was simulated ($t = 1 \text{ s}$) by connecting a 200 MW load to the test network (a *switch event*), assuming that prior to the event the test system was operating with a 1,800 MW load and that the generator (CCGT/OCGT) was operating at rated power output (400 MW). Figure 8.10 illustrates in turn the CCGT and OCGT responses, with the following per unit conventions used for the various

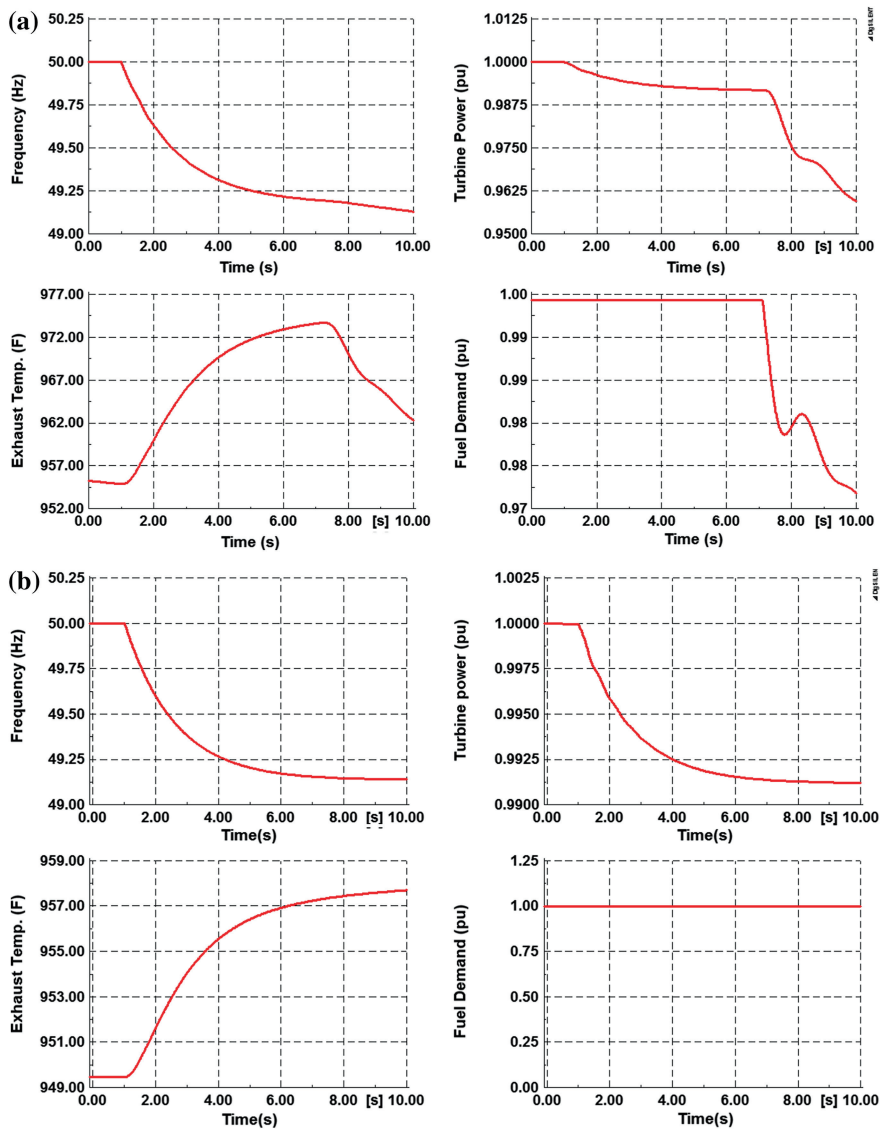


Fig. 8.10 Gas turbine response during a low-frequency event; **a** CCGT, **b** OCGT

traces shown: per unit fuel demand is defined based on the fuel demand required for rated power output; per unit turbine power output defined is based on the rated power output of the generator.

When the 200 MW load is connected to the network ($t = 1$ s), the system frequency falls to a new steady-state level (dependent on the combined droop of the generator and system equivalent), with the rate of frequency fall higher when the

OCGT is in place rather than the CCGT (due to the fact that the CCGT has a much larger inertia and hence inertial response, in comparison with the OCGT). It should be noted that the inertial response temporarily raises the unit output beyond the rated active power output of both machines.

As the system frequency falls, the turbine power output for both the CCGT and OCGT also decreases, a phenomenon which can be explained as follows: when the system frequency falls, the compressor slows down, reducing the airflow to the combustion chamber and hence also the pressure ratio at the turbine. Subsequently, the turbine power output is reduced, since the fuel input to the combustion chamber cannot be increased, as the generator is operating at rated power output (i.e. maximum fuel limit reached). However, the power reduction of 0.008 pu is negligibly small in comparison with the turbine power reduction (which occurs at 6.8 s). Subsequently, after a few seconds, the system frequency decreases to 49.18 Hz for the CCGT case, while the corresponding minimum frequency nadir for the OCGT is slightly higher at 49.22 Hz. Again, this phenomenon can be explained as follows: when the system frequency falls, the compressor slows down while reducing the airflow to the combustion chamber, which then results in a temperature increase of the exhaust gases due to the increase in the fuel-to-air ratio in the combustion chamber. As mentioned earlier, CCGTs, but not OCGTs, are equipped with VIGVs, since they must maintain an optimal exhaust temperature in order to achieve maximum efficiency at the heat recovery stage. Therefore, the exhaust temperature is corrected by altering the VIGV position. With the CCGT operating at rated power output its VIGVs are fully open, such that the fuel-to-air ratio in the combustion chamber cannot be corrected.

Instead, the compressor speed is the only determinant of the fuel-to-air ratio and hence the exhaust temperature increases. Ultimately, the CCGT exhaust gas temperature increases beyond that for the OCGT. The fuel selector selects the minimum value between the temperature control and speed governor signals in order to determine the fuel demand. The CCGT exhaust temperature increases at a much faster rate than that for the OCGT, since VIGVs can no longer control the CCGT exhaust temperature. Subsequently, the CCGT temperature control loop overrides the speed governor signal, and the fuel input to the combustion chamber is reduced, while also reducing the exhaust gas temperature ($t = 6.6$ s), whereupon the turbine power output reduces below that for the OCGT ($t = 6.8$ s). Consequently, the system frequency starts to fall below the system frequency for the OCGT case ($t = 8.1$ s), while worsening the frequency stability issues in the network. The exhaust temperature spread can be used as a signature of LBO for the gas turbine, measured as the temperature change across a 2 s time window [19]. An exhaust temperature spread of 19.5 °F is seen for the CCGT while the equivalent value for the OCGT units is 8.6 °F. In both cases, the exhaust temperature spread is much less than the typical temperature spread required for a LBO condition (i.e. 200 °F).

8.6.3 Gas Turbine Dynamics During Frequency Increase Events

A high-frequency event is simulated at $t = 1.0$ s by disconnecting a 200 MW load from the network, given an initial load of 2,000 MW, and assuming that both the CCGT and OCGT are initially operating at rated power output. Following the load reduction event, the system frequency increases at a higher rate when the OCGT is in place compared to the CCGT, due to the low inertia of the OCGT unit, and so the OCGT governor initially provides a faster response. In steady-state, the system frequency depends on the combined droops of the CCGT/OCGT and the external grid, which is achieved here by a 24 % (96 MW) reduction in turbine output. For this event, the temperature control loop does not override the governor response, and so the CCGT and OCGT experience the same turbine power output reduction.

Following the load reduction event, the system frequency increases at a higher rate when the OCGT is in place compared to the CCGT scenario, due to the low inertia of the OCGT unit, and so the OCGT governor initially provides a faster response. In steady state, the system frequency depends on the combined droops of the CCGT/OCGT and the external grid, which is achieved here by a 24 % (96 MW) reduction in turbine output. For this event, the temperature control loop does not override the governor response, and so the CCGT and OCGT experience the same turbine power output reduction.

As the system frequency increases, the air compressor speeds up, which leads to more air being drawn into the combustion chamber. This results in a low fuel-to-air ratio and consequently the turbine exhaust gas temperature reduces. For OCGTs the exhaust temperature spread is high compared to CCGTs (OCGT -110 °F, CCGT -31 °F), since the latter is equipped with variable IGVs, and so can reduce the airflow by ‘closing’ the vanes when the turbine exhaust temperature falls (see Fig. 8.11). In contrast, OCGTs do not incorporate IGVs to correct the exhaust temperature; hence, its exhaust temperature reduces. When both gas turbines are operated below base load, the same trend can be observed. Therefore, CCGTs indicate a more robust exhaust temperature compared to OCGTs during high-frequency events. Although OCGTs indicate a larger exhaust temperature spread, the overall trend is again a temperature reduction, well within safe limits, and so an LBO condition during a 200 MW load reduction is unlikely.

The exhaust temperature for the CCGT response indicates much larger temperature oscillations, due to the continuous temperature control action performed by the IGVs (see Fig. 8.11). Consequently, a large load reduction event, resulting in a high ROCOF, may ultimately lead to a higher exhaust temperature spread and hence increased likelihood of an LBO condition. This is because a change in fuel flow is largely instantaneous; however, due to the inertia of the rotating mass of the gas turbine, an increased airflow may reduce the fuel-to-air ratio below the lean flammability limit [9].

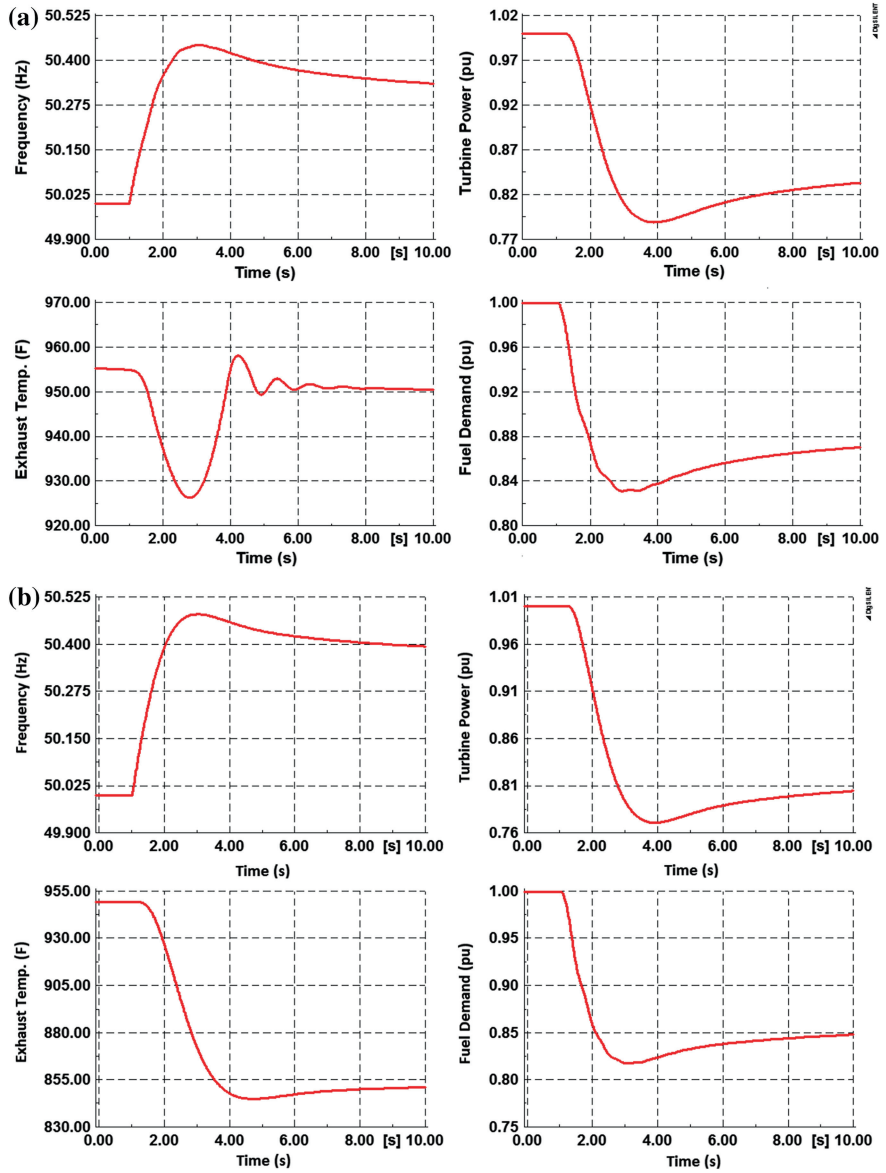


Fig. 8.11 The gas turbine response during a frequency increase event; a CCGT, b OCGT

8.6.4 Dynamic Behaviour of Gas Turbines During Short-Circuit Events

During a short-circuit events (e.g. 3-phase short circuit), generators are subjected to large electro-mechanical oscillations due to the large voltage drop across a wide area of the power network. Therefore, during short-circuit faults, large frequency excursions and high ROCOFs can be observed. The ROCOF was measured using the model outlined in [22] and the modelling of the ROCOF measurement model will not be discussed here. Figure 8.12 illustrates the dynamic behaviour of the CCGT and OCGT units for a 150 ms short-circuit fault in the test system at busbar B2 ($t = 1$ s), with both units operating at base load.

According to Fig. 8.12, the system frequency has significantly increased during the fault for both scenarios, particularly with the OCGT in place: a maximum ROCOF of 0.89 Hz/s for the OCGT and 0.83 Hz/s for the CCGT. The frequency rise has resulted in a substantial reduction in fuel inflow to the combustor, coupled with an accelerating compressor pumping more air to the combustor, and ultimately the turbine power output is substantially reduced. However, the turbine power output reduction (i.e. 25 %) is very similar for both the CCGT and OCGT, given that the fuel demand reduction is similar.

After fault clearance, the system frequency quickly recovers to nominal, with a corresponding increase in fuel flow to the combustion chamber. However, now the CCGT IGVs are at their minimum value, and the compressor also slows down while reducing the airflow to the combustion chamber. This substantial airflow reduction impacts the air to fuel ratio, causing a steep increase in the exhaust temperature ($t = 2.5$ s). In this particular scenario, the exhaust temperature spread for the CCGT is ≈ 66 °F, which is below the LBO exhaust temperature spread [23], however characteristic exhaust temperature behaviour during a LBO event can be observed. For the OCGT, the exhaust temperature spread is only 38 °F, indicating that CCGTs are much more susceptible to LBO during faults, due to the airflow control actions of the IGVs.

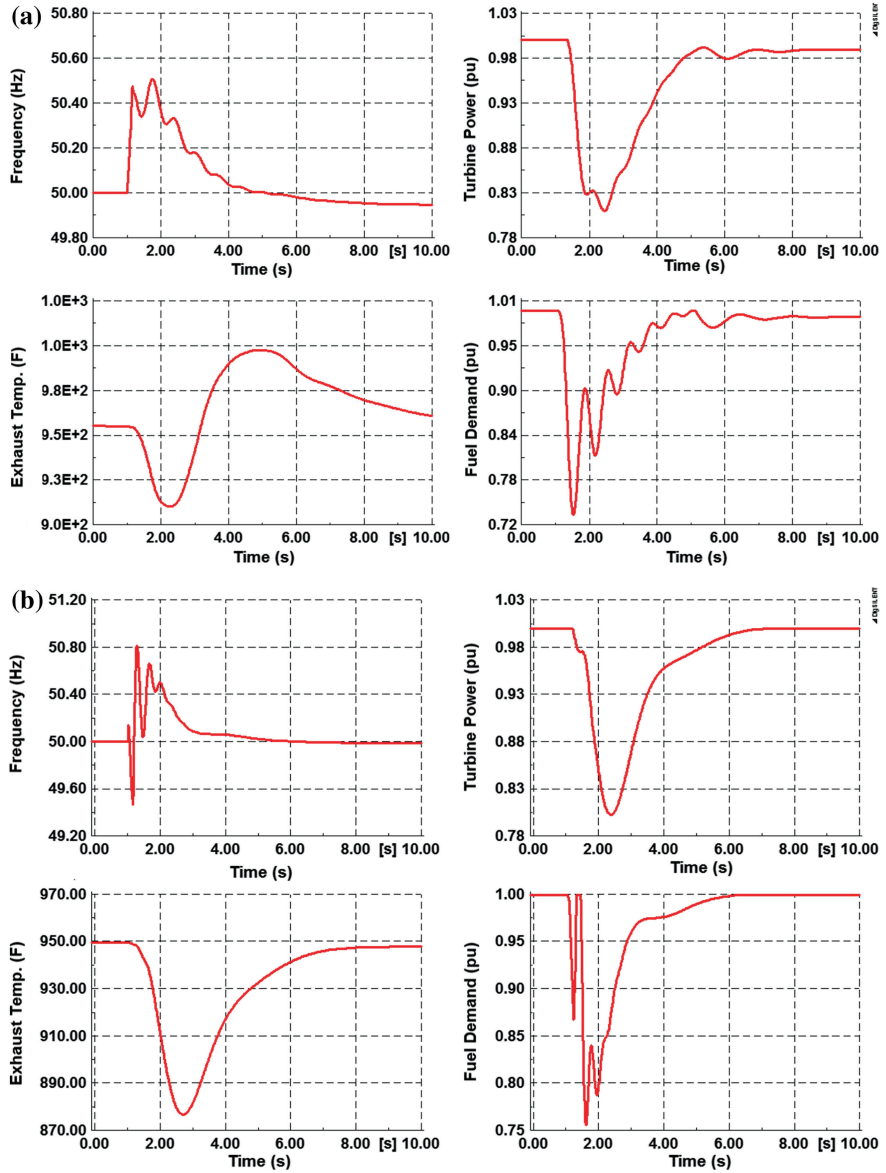


Fig. 8.12 Dynamic behaviour of gas turbine units during a short-circuit fault; **a** CCGT, **b** OCGT

References

1. Fox B, Flynn D, Bryans L et al (2007) Wind power integration: connection and system operation aspects. IET, London
2. Bagnasco A, Delfino B, Denegri GB et al (1998) Management and dynamic performances of combined cycle power plants during parallel and islanding operation. *IEEE Trans Energy Conv* 13(2):194–201
3. Kakimoto N, Baba K (2003) Performance of gas turbine-based plants during frequency drops. *IEEE Trans Power Syst* 18(3):1110–1115
4. Lalor G, Ritchie J, Flynn D et al (2005) The impact of combined-cycle gas turbine short-term dynamics on frequency control. *IEEE Trans Power Syst* 20(3):1456–1464
5. Meegahapola L, Flynn D (2011) Frequency dynamics during high CCGT and wind penetrations. Paper presented at the 21st Australasian Universities power engineering conference (AUPEC), Brisbane, Australia
6. Meegahapola L (2014) Characterisation of gas turbine dynamics during frequency excursions in power networks. Available via IET Digital-library. <http://digital-library.theiet.org/content/journals/10.1049/iet-gtd.2013.0824>. Accessed 15 July 2014
7. Doyle J (2012) Grid requirements on CCGT plants. CIGRE, Ireland
8. NERC Industry Advisory (2008, June) Turbine combustor lean blowout. USA
9. FRCC (2008) FRCC system disturbance and under frequency load shedding event report. USA
10. Rowen WI (1983) Simplified mathematical representations of heavy-duty gas turbines. *J Eng Power* 105(1):865–869
11. IEEE Working Group on Prime Mover and Energy Supply Models for System Dynamic Performance Studies (1994) Dynamic models for combined cycle plants in power system studies. *IEEE Trans Power Syst* 9(3):1698–1708
12. Kunitomi K, Kurita A et al (2001) Modeling frequency dependency of gas turbine output. In: *Proceedings of IEEE power engineering society winter meeting*, Columbus, USA
13. Kunitomi K, Kurita A (2003) Modeling combined-cycle power plant for simulation of frequency excursions. *IEEE Trans Power Syst* 18(2):724–729
14. CIGRE Task Force (2003) Modeling of gas turbines and steam turbines in combined cycle power plants
15. Yousefi I, Yari M et al (2013) Modelling, identification and control of a heavy duty industrial gas turbine. In: *Proceedings of 2013 IEEE international conference on mechatronics and automation (ICMA)*, Kagawa, Japan
16. Soon KY, Milanovic JV, Hughes FM (2008) Overview and comparative analysis of gas turbine models for system stability studies. *IEEE Trans Power Syst* 23(1):108–118
17. Hannett LN, Khan AH (1993) Combustion turbine dynamic model validation from tests. *IEEE Trans Power Syst* 8(1):152–158
18. Tavakoli MRB, Vahidi B, Gawlik W (2009) An educational guide to extract the parameters of heavy duty gas turbines model in dynamic studies based on operational data. *IEEE Trans Power Syst* 24(3):1366–1374
19. IEEE Standard 421.5-2005 (2006) IEEE recommended practice for excitation system models for power system stability studies
20. System Operator Northern Ireland (2008) Minimum function specification for centrally dispatched OCGT units
21. System Operator Northern Ireland (2010) Minimum function specification for centrally dispatched CCGT units
22. Affonso CM, Freitas W, Xu W, Da Silva LCP (2005) Performance of ROCOF relays for embedded generation applications. *IEE Proc Gener Transm Distrib* 152(1):109–114
23. PAL Turbine Services (2013) Turbine tips: root causes of high exhaust temperature spreads. Available via Poundlucier. <http://www.poundlucier.com/Company/tip.htm>. Accessed May 2013

Chapter 9

Implementation of Simplified Models of DFIG-Based Wind Turbines for RMS-Type Simulation in DIgSILENT PowerFactory

José Luis Rueda, Abdul W. Korai, Jaime C. Cepeda, István Erlich and Francisco M. Gonzalez-Longatt

Abstract Due to its variable nature, the increasing penetration of wind power plants into power systems poses new challenges for reliable and secure operation. Considering that model-based time-domain simulation constitutes a widely used approach for assessing the power system dynamic performance as well as for making proper decisions concerning operational and planning security strategies, there has been a great research effort, especially in the last decade, to cover different issues on modelling of wind generation systems (WGS). Remarkably, the development of models that entail a compromise between accuracy and simplicity is one of the main

Electronic supplementary material The online version of this chapter (doi:10.1007/978-3-319-12958-7_9) contains supplementary material, which is available to authorized users.

J.L. Rueda (✉)

Department of Electrical Sustainable Energy, Delft University of Technology,
Mekelweg 4, 2628 CD Delft, The Netherlands
e-mail: J.L.RuedaTorres@tudelft.nl

A.W. Korai

Institute of Electrical Power Systems, University Duisburg-Essen,
Bismarckstr. 81, 47057 Duisburg, Germany
e-mail: abdul.korai@uni-due.de

J.C. Cepeda

Corporación Centro Nacional de Control de Energía—CENACE,
Av. Atacazo and Panamericana Sur km 0, Quito, Ecuador
e-mail: cepedajaime@ieee.org

I. Erlich

Department of Electrical Engineering and Information Technologies,
University Duisburg-Essen, Bismarckstr. 81, 47057 Duisburg, Germany
e-mail: istvan.erlich@uni-due.de

F.M. Gonzalez-Longatt

School of Electronic, Electrical and Systems Engineering, Loughborough University,
LE11 3TU Loughborough, UK
e-mail: fglongatt@fglongatt.org

concerns for enabling the simulation of large-scale systems. This chapter addresses the implementation of two simplified models of WGs by using the functionalities of DIgSILENT simulation language (DSL). The first one is the reduced third-order model of the doubly fed induction generator (DFIG), for which suitable models for multiple point tracking, the rotor-side controller (RSC), current controller, and speed and pitch controller are adopted. The second model constitutes a generic equivalent model, which can be used for representation of the stationary and dynamic response of wind power plants comprising several DFIGs. RMS-type simulation results are presented to illustrate the suitability of the adopted modelling approaches.

Keywords Control system · Doubly fed induction generator · Dynamic equivalent · Power system dynamic performance

9.1 Introduction

Wind power has positioned as the most promising renewable energy source among all other alternatives considered in the energy transition that is underway in several countries worldwide [1]. Small-size wind power plants (WPPs) were usually connected at distribution system level in the initial developmental stage. Nevertheless, due to the rapid technological progress achieved in recent years, more focus has been put into the planning and installation of WPPs with several hundred megawatts of capacity. Besides, these large-size WPPs are preferably connected at the high-voltage transmission level [8].

Simultaneously, this is also motivating a number of research endeavours for understanding and assessing the impacts of the ongoing changes in power system performance patterns due to the increasing share and variable nature of wind power [12, 15]. To this aim, many studies rely on model-based steady-state and dynamic stability simulations. With considerable large-size of modern interconnected power systems, the execution of time-domain simulations, even for short time windows (e.g. 10 s), by considering detailed modelling of a large number of WPPs, would result in unacceptable computational effort (due to very small integration step times, e.g. smaller than 1 ms) [6].

Thus, especial attention has also been put into the development of simple and generic models for wind generation systems (WGs), which is usually done under the following considerations [7]: (i) adequate representation of the dynamic phenomena under study; (ii) ability for simulating WGs of different manufactures and types; (iii) contribute significantly to decrease the computational memory and time that is expended to run simulations. Based on previous experience of the authors on these issues, the purpose of this chapter is twofold: (i) to overview key aspects, which have relevance for large-scale RMS-type time-domain simulation, concerning the definition of models for the reduced third-order model of the doubly fed induction generator (DFIG) and a dynamic equivalent for WPPs comprising DFIG- or full-size converter (FSC)-based WGs; (ii) to illustrate the implementation

and evaluation of the suitability of these models by using the functionalities of DIgSILENT simulation language (DSL).

This chapter is organized as follows: In Sect. 9.2, the background on the adopted modelling concepts is introduced. In Sect. 9.3, the description and implementation of the models in DIgSILENT PowerFactory is presented. In Sect. 9.4, the performance of the models is assessed and analysed. Finally, Sect. 9.5 summarizes the main concluding remarks and outlines prospective future work.

9.2 Mathematical Model of DFIG

The system of equations for the DFIG consists of two voltage equations, two flux equations and an equation of motion which are given below [9].

Voltage equations:

$$\underline{u}_S = r_S \underline{i}_S + j\omega \underline{\psi}_S + \frac{d\underline{\psi}_S}{dt} \quad (9.1)$$

$$\underline{u}_R = r_R \underline{i}_R + j(\omega_0 - \omega_R) \underline{\psi}_R + \frac{d\underline{\psi}_R}{dt} \quad (9.2)$$

where

\underline{u}_S	Stator voltage
r_S	Stator resistance
\underline{i}_S	Stator current
ω_0	Synchronous speed
$\underline{\psi}_S$	Stator flux linkages
\underline{u}_R	Rotor voltage
r_R	Rotor resistance
\underline{i}_R	Rotor current
ω_R	Speed of the machine
$\underline{\psi}_R$	Rotor flux linkages

Flux equations:

$$\underline{\psi}_S = l_S \underline{i}_S + l_M \underline{i}_R \quad (9.3)$$

$$\underline{\psi}_R = l_M \underline{i}_S + l_R \underline{i}_R \quad (9.4)$$

where

l_S	Stator self-inductance
l_R	Rotor self-inductance
l_M	Mutual inductance between stator and rotor

Equation of motion:

$$\frac{d\omega_R}{dt} = \frac{1}{\theta_m} [k_R (\psi_{Rd} i_{Sq} - \psi_{Rq} i_{Sd}) + t_m] \quad (9.5)$$

where

ψ_{Rd}	Rotor flux linkages aligned with d -axis
i_{Sq}	Stator current aligned with q -axis
ψ_{Rq}	Rotor flux linkages aligned with q -axis
i_{Sd}	Stator current aligned with d -axis
t_m	mechanical torque of the machine

whereas the inductances are defined as

$$l_s = l_h + l_{\sigma s} \quad (9.6)$$

$$l_R = l_h + l_{\sigma R} \quad (9.7)$$

and

l_h	Magnetizing inductance
$l_{\sigma s}$	Stator leakage inductance
$l_{\sigma R}$	Rotor leakage inductance

Equations (9.1)–(9.7) constitute the full (5th)-order model of DFIG. This model considers the fast changes in the stator flux which determines the DC component of the stator currents in case of grid fault. In the rotor circuit, this DC component appears as AC component and has an effect on the DC link voltage of the converter [10]. This effect has to be accounted for in some grid studies where fast transients are of importance.

9.2.1 Quasi-Stationary Model (Reduced Third-Order Model)

The full-order model requires differential equations for the whole network due to the fact that the grid is directly connected to the stator circuit. If the full-order model is used for time-domain simulation, small integration step sizes are required due to the small time constants involved. The small integration time step, in addition to the large number of differential equations, especially for the grid, results in considerable simulation efforts when studying large systems. These disadvantages limit the application of the full-order model to small grids or even to a single machine, infinite bus systems [10]. A reduced-order model can be derived by neglecting the derivative term in the stator equation. The effect of stator flux changes can be

neglected with the assumption that the transformer voltage in the stator winding can be neglected against the much greater speed voltage, that is

$$|j\omega \cdot \underline{\psi}_s| \gg \left| \frac{d\underline{\psi}_s}{dt} \right| \quad (9.8)$$

By considering this, the stator flux derivative in (9.1) can be set to zero. After some algebraic manipulation, stator flux linkage equation is obtained

$$\underline{\psi}_s = \frac{\underline{u}_s - r_s \cdot \underline{i}_s}{j\omega_0} \quad (9.9)$$

By substituting (9.4) into (9.3), the rotor currents can be eliminated to yield

$$\underline{\psi}_s = l_s \cdot \underline{i}_s + \frac{l_h}{l_R} (\underline{\psi}_R - l_h \cdot \underline{i}_s) \quad (9.10)$$

Rearranging (9.9) and (9.10) results in

$$\underline{u}_s = (r_s + j\omega_0 l') \underline{i}_s + j\omega_0 k_R \underline{\psi}_R \quad (9.11)$$

where

$$k_R = \frac{l_h}{l_h + l_{\sigma R}} \quad (9.12)$$

Equation (9.11) corresponds to the reduced third-order model of the DFIG and can also be written as

$$\underline{u}_s = \underline{z}' \underline{i}_s + \underline{u}' \quad (9.13)$$

where \underline{z}' and \underline{u}' denote the impedance and internal voltage of the machine, respectively. From (9.11), it is evident that the internal voltage of the machine is a function of rotor flux. Substituting (9.4) into (9.2) and decomposing it into d - q components, the rotor flux equations are obtained as follows:

$$\frac{d\underline{\psi}_{Rd}}{dt} = -\frac{r_R}{l_R} \underline{\psi}_{Rd} + (\omega_R - \omega_0) \underline{\psi}_{Rq} + k_R r_R i_{sd} + u_{Rd} \quad (9.14)$$

$$\frac{d\underline{\psi}_{Rq}}{dt} = (\omega_R - \omega_0) \underline{\psi}_{Rd} - \frac{r_R}{l_R} \underline{\psi}_{Rq} + k_R r_R i_{sq} + u_{Rq} \quad (9.15)$$

Equation (9.5) along with (9.14) and (9.15) represents mathematical relationships that describe the dynamic behaviour of the machine (reduced third-order model).

9.3 DFIG Model Representation

As is well known, the rotor terminals of a DFIG are fed with a symmetrical three-phase voltage of variable frequency and amplitude fed through a voltage source converter usually equipped with IGBT-based power electronic circuitry [11]. The basic topology including its control system is shown in Fig. 9.1. As a general approach, the space-phaser coordinates with orthogonal direct (d) and quadrature (q) axis is used. The choice of the stator voltage as the reference frame enables the decoupled control of P (d control channel) and Q (q control channel).

A complete model of the DFIG also includes models of the real and reactive power control together with speed and pitch angle control. Moreover, damping control could be also embedded into the speed control structure. These models, however, are not relevant for the purposes of study in this chapter, which is the large-disturbance rotor angle stability (i.e. transient stability) of a power system.

The structures of the wind turbine mechanical model, rotor-side controller (RSC), line-side controller (LSC), and speed and pitch controllers are described in the next subsection. The models presented here summarize the core functionalities which are of relevance for stability studies. Notwithstanding, it would be possible to augment these core functionalities for other different purposes. In addition, it is worth to mention that the structures as presented here reproduce neither any eventual blocking of the converters nor crowbar activation during grid disturbances.

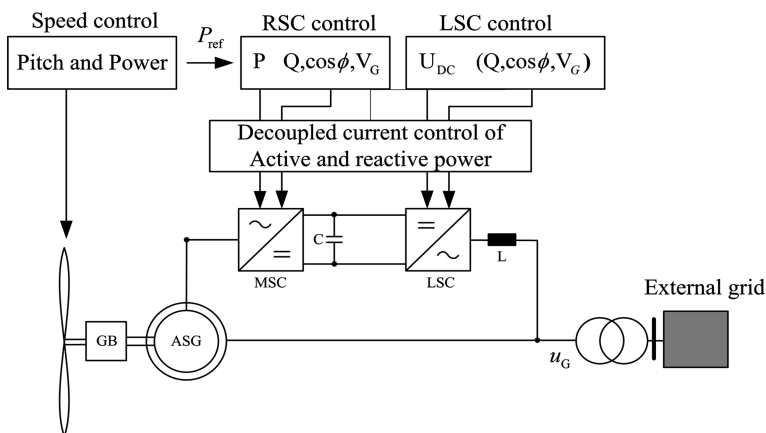


Fig. 9.1 DFIG model representation

9.3.1 Wind Turbine Mechanical Model and Drive Train

The mechanical power extracted from wind by the wind turbine can be calculated as [13]:

$$p_w = \frac{1}{2} \rho A_{\text{rot}} c_p(\lambda, \beta) v_w^3 \quad (9.16)$$

where ρ is the air density, A_{rot} is the cross section through which the air mass is streaming, c_p is the power coefficient, and v_w is the wind speed. Wind turbine manufacturers give the specific value for a turbine as a function c_p of the pitch angle β and tip-speed ratio λ . The drive train is described as a one-mass model. The gearbox mass and the losses are neglected. The tip-speed ratio is defined as follows:

$$\lambda = \omega_T R / v_w \quad (9.17)$$

where R is the radius of rotor blade and ω_T is the speed of the turbine. There is a fixed relationship between ω_T and R given by the gear transmission ratio. The turbine model is based on the steady-state power characteristics of the turbine.

During heavy faults, shaft oscillations should be considered in order to simulate the turbine and generator acceleration response with a sufficient accuracy. For this purpose, the shaft is modelled as a two-mass model representing the two dominant masses: the wind rotor with a larger inertia H_w and the generator with a smaller inertia H_R . The mechanical inertia constant H_R is defined as a parameter in the built-in generator model in PowerFactory. Therefore, according to DIGSILENT PowerFactory [3], the low-speed side of the shaft is modelled by stiffness K_{sh} and a damping coefficient d_{sh} , while the high-speed side is assumed stiff [3]. The following equations describe the drive train coupling of the wind turbine:

$$H_w \frac{d\omega_w}{dt} = t_w - t_R \quad (9.18)$$

$$H_R \frac{d\omega_R}{dt} = t_R + t_{\text{el}} \quad (9.19)$$

$$\frac{d\theta_{\text{WR}}}{dt} = \omega_w - \omega_R \quad (9.20)$$

$$t_R = k_{\text{sh}} \theta_{\text{WR}} + d_{\text{sh}} (\omega_w - \omega_R) \quad (9.21)$$

where t_R and t_{el} are the mechanical and electrical torques of the generator, respectively. The turbine torque is defined as follows:

$$t_w = \frac{p_w}{\omega_w} \quad (9.22)$$

The control system of the DFIG-based WGS can mainly be divided into five parts:

- Speed and pitch control
- Reference value calculation for active and reactive currents for RSC
- RSC current control
- Reference value generation for the active and reactive currents for LSC
- LSC current control

In the following subsections, the main control schemes which are implemented in the actual model are described.

9.3.2 Speed and Pitch Control

Wind energy conversion with DFIG allows operation at variable speed. This can be used to operate the wind turbine at the point of maximum power. This power tracking is achieved by two mechanisms: speed and pitch control. The speed control is shown in the upper part of Fig. 9.2.

It adapts the generator speed according to a tracking characteristic based on a power measurement. The reference value for active power p_{WT_ref} obtained from the speed controller is passed to the RSC control. The pitch control in the lower part limits the generator speed to its rated value for wind speeds above rated wind speed.

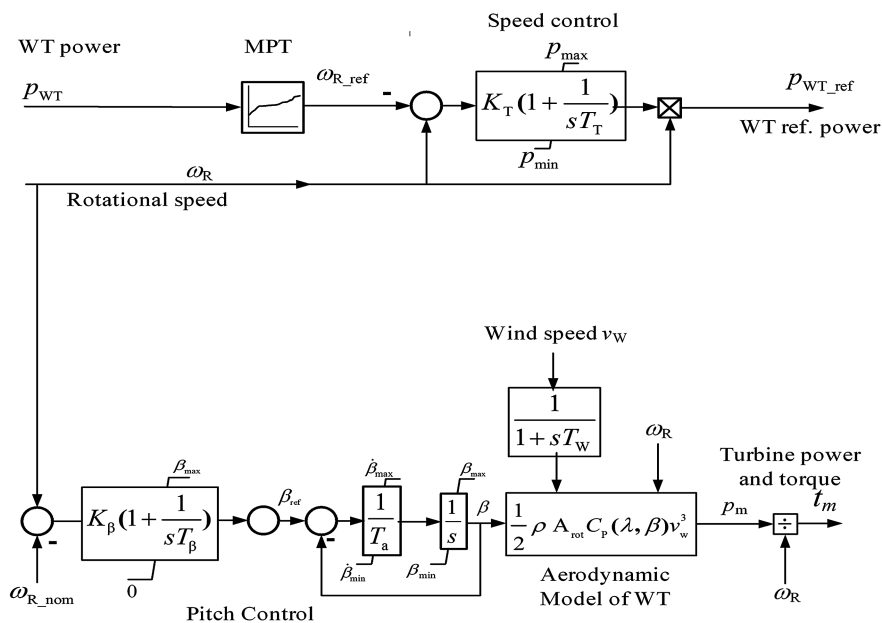


Fig. 9.2 Speed and pitch control of DFIG-based WGS

9.3.3 Calculation of Reference Values for Rotor Currents

The decoupled control of active and reactive power is achieved by the current control of the RSC, which is working in a stator voltage-oriented reference frame. The reference values for d -axis and q -axis components of the rotor currents are calculated according to the structure shown in Fig. 9.3.

The reference active power p_{WT_ref} is used to calculate the d -axis reference current $i_{Rd_ref}^{\angle u_s}$. The q -component $i_{Rq_ref}^{\angle u_s}$ considers the reference value for stationary reactive power transfer q_{WT_ref} and the excitation current of the DFIG ($i_{Rq}^* = -u_s/x_h$). The reference active and reactive powers are passed through delay block to account for communication delays. Besides, the control structure includes a fast-acting controller for voltage support through reactive currents during grid faults. It maintains a deadband of $\pm 10\%$ to preclude controller action until the voltage exceeds this limit. Nevertheless, it should be kept in mind that the deadband and the gain K_p should be chosen depending on specific grid code's stability requirements.

9.3.4 Rotor-Side Converter Current Control

The RSC current control is performed using a feed-forward decoupled control structure as shown in Fig. 9.4 [4].

The reference values $i_{Rd_ref}^{\angle u_s}$ and $i_{Rq_ref}^{\angle u_s}$ are obtained from the calculations explained in the previous subsection. The outputs $u_{Rd}^{\angle u_s}$ and $u_{Rq}^{\angle u_s}$ of the current

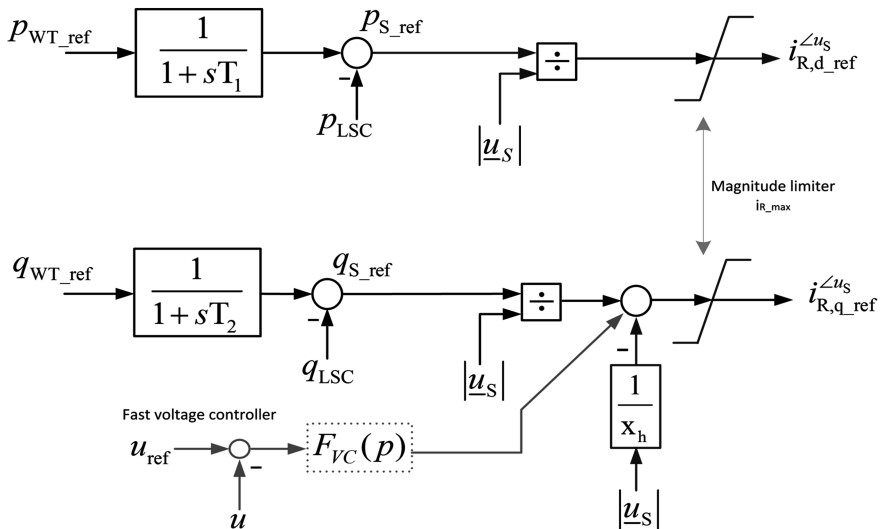


Fig. 9.3 Structure of reference value calculation for active and reactive rotor currents

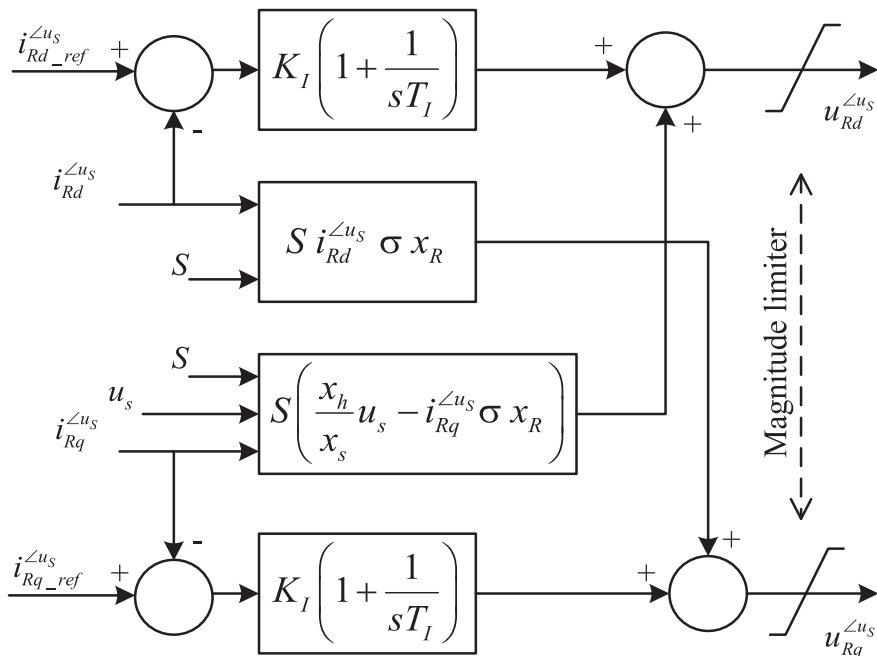


Fig. 9.4 Current control of RSC

controllers represent the set point of the RSC voltage converter. The cross-coupling (feed forward) terms compensate for any disturbances occurring, while PI current controller only accounts for the measurements delays and parameter uncertainties. The inclusion of cross-coupling terms provides fast compensation.

9.3.5 Reference Values of LSC Currents

In the active power channel, following the assumption that DC voltage fluctuations are not considered, the line-side converter will try to inject the same active power into the grid as that provided to the DFIG through the rotor-side converter (i.e. converter losses are neglected). The reactive power control channel of the line-side converter would provide the capability to generate reactive power. Basically, it works like a static compensator (e.g. STATCOM). However, in normal operation, it would be more purposeful to generate reactive power using the RSC. The current transmitted from the rotor to the stator circuits is amplified on account of the characteristic value of the turn ratio of the DFIG. Thus, the same reactive current passing through the rotor-side converter is capable of producing higher reactive power compared to the current through the line-side converter. As a result, in normal operation, the reactive power reference value of the line-side converter is

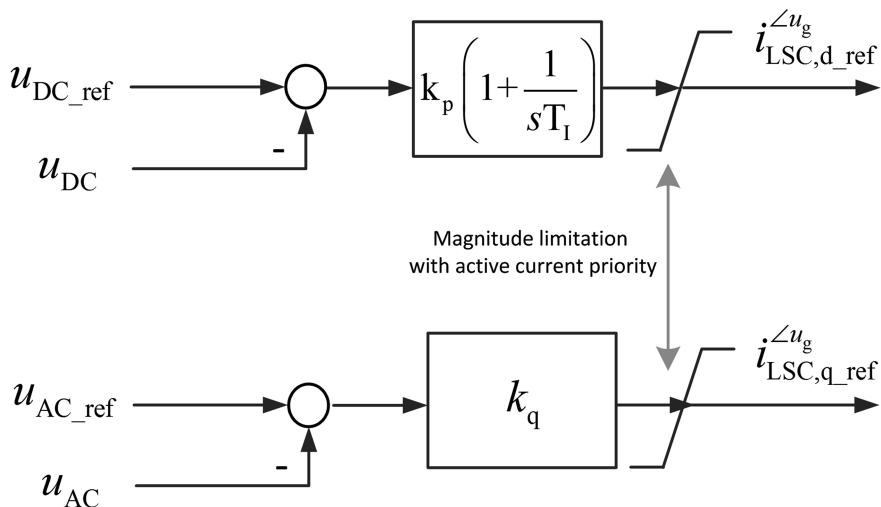


Fig. 9.5 Structure for generating LSC current reference values

usually set to zero and the total reactive current required is supplied by the rotor-side converter. However, during a disturbance or when stator circuit is disconnected for a short period of time, the line-side converter can be activated to provide reactive power support.

The LSC utilizes a similar decoupled current control as the RSC. The d -component of the LSC current $i_{d_LSC}^{\angle u_g}$ is used to control the voltage in the DC circuit of the converter u_{DC} . The q -component $i_{q_LSC}^{\angle u_g}$ is used to support the grid voltage during grid faults according to the reactive current distribution between generator and LSC. Figure 9.5 illustrates the concept for generating LSC current reference values. The reference voltage values $u_{LSC,d_ref}^{\angle u_g}$ and $u_{LSC,q_ref}^{\angle u_g}$ are the set values for the LSC which is modelled using the PWM converter net element in PowerFactory.

9.3.6 Line-Side Converter Current Control

The LSC current control is performed using a decoupled control structure as illustrated in [4]. The reference values $i_{LSC,d_ref}^{\angle u_g}$ and $i_{LSC,q_ref}^{\angle u_g}$ are obtained from the calculations explained in the previous subsection. The outputs $u_{LSC,d_ref}^{\angle u_g}$ and $u_{LSC,q_ref}^{\angle u_g}$ of the current controllers represent the set points of the line-side voltage source converter. The feed-forward terms in LSC current control have not been included on the account that the PI current controller with gain and integration time constant are able to compensate for any error resulting from the disturbances [5].

being out of the scope of this chapter, it is worth mentioning that a Norton model can be alternatively used.

The voltage source is supplied by two delay blocks (one each for the real and imaginary components), which represent the electrical machine and the converter delay.

The active and reactive current control is performed by using PI blocks, which operate in terminal voltage-oriented coordinates, which entails that the d -component of the current corresponds with the active current and the q -component with the negative of the reactive current. For sake of simplicity, it is considered that the active current reference is synthesized from the active power, which is assumed constant, whereas the reactive current reference is provided by a terminal voltage controller. The assumption concerning with constant active power is valid for 1–2 s following a grid disturbance. For longer simulation time spans, modelling of the speed controller including the equation of motion might be necessary. The magnitude of the current reference is limited to the maximum allowed (i_{\max}) for the converter system. The way the limitation is done depends on the underlying priority, which is defined by (9.23)–(9.26).

Active current priority:

$$i_{\text{Pref}} = i_{\text{Pref}}^* \quad (9.23)$$

$$i_{\text{Qref}} = \sqrt{(i_{\max})^2 - (i_{\text{Qref}}^*)^2} \quad (9.24)$$

Reactive current priority:

$$i_{\text{Qref}} = i_{\text{Qref}}^* \quad (9.25)$$

$$i_{\text{Pref}} = \sqrt{(i_{\max})^2 - (i_{\text{Qref}}^*)^2} \quad (9.26)$$

The above priorities are chosen based on the terminal voltage magnitude [7]. In normal operating conditions, the priority is on active current. By contrast, reactive current is prioritized following grid faults, especially if the voltage drops below 80–90 % of the nominal value. In this way, the model is also conceived to emulate reactive power support from WPPs, which is commonly required in several utility grid codes [14].

9.5 Implementation in DIgSILENT PowerFactory

The DFIG-based WT and WPP models are created in DIgSILENT PowerFactory 15.0. The following sections provide key details on the implementation of dynamic these models with special focus on the description of composite models (*ElmComp*) and initialization of the controllers.

9.5.1 Dynamic Equivalent

Thevenin voltage source representation of the WPP is performed by using the static generator (*ElmGenstat*) model behaving as a controlled voltage source. Modelling of the schemes associated with WPP, including the calculation of the voltage source's input reference voltages, is done through composite model frames embedded with composite block definitions. The WPP's composite model frame consists of the following blocks as shown in Fig. 9.7: (i) static generator (*ElmGenstat*); (ii) WPP control: controller layout (including calculation of input reference voltages); (iii) *P_fixed*: estimation of the initial active power at WPP terminal; (iv) *Q_fixed*: estimation of the initial reactive power at WPP terminal; (v) measurement of active and reactive power at WPP terminal (*StaPqmea*); (vi) voltage measurement at WPP terminal (*StaVmea*).

The DSL implementation of the WPP inner control system is shown in Fig. 9.8. To initialize the model, the following aspects are taken into consideration:

- The system starts from steady state (determined by load flow calculation)
- All the derivatives are set to zero in the controller transfer functions
- All the limits, deadbands and saturation blocks are ignored

The initialization is done from the right-hand side (output) to the left-hand side (input) and based on results from load flow calculation. The following equations represent the behaviour of the static generator when simulated as a controlled voltage source [3].

$$u1r_in + j \cdot u1i_in = \bar{u}1 + \bar{j}1 \cdot \bar{z} \quad (9.27)$$

$$\bar{z} = R + j \cdot X \quad (9.28)$$

where *u1r_in* and *u1i_in* are the controlled positive sequence voltage's real and imaginary parts in p.u., respectively. These are determined initially by load flow calculation based upon the *P* and *Q* values of the static generator. *R* and *X* are the

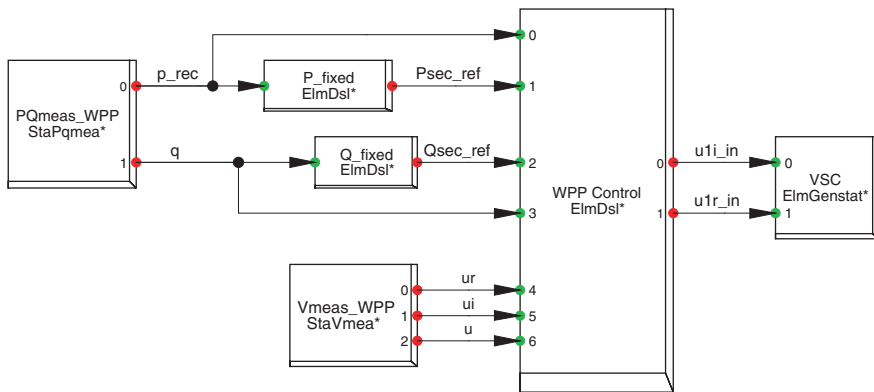


Fig. 9.7 Structure of the WPP dynamic equivalent in PowerFactory


```
inc(x)=(-u1r_in*sinphi+u1i_in*cosphi)/Ki !Upper PI block: x (state variable)
```

The initial condition of x_q (associated to the lower PI controller) is obtained in a similar manner:

```
inc(x1)=(u1r_in*cosphi+u1i_in*sinphi)/Ki !Lower PI block: x1 (state variable)
```

The initial conditions of first-order delay blocks and input voltage reference signal are given by

```
inc(x2)=u1r_in !Upper delay block: x2 (state variable)
inc(x3)=u1i_in !Lower delay block: x3 (state variable)
inc(u_ref)=u !Reference terminal voltage magnitude
```

9.5.2 DFIG Turbine Model and RSC

Figure 9.9 shows the model of the wind turbine and DFIG rotor-side converter, which was built as a composite model and contains the following components: (i) DFIG: It includes the standard PowerFactory model of DFIG (*ElmAsmsc*); (ii) current measurement: coordinate transformation into stator voltage-oriented reference frame; (iii) I_r _ctrl: current controller; (iv) feed-forward term for RSC current controller and coordinate transformation into rotor-oriented reference frame;

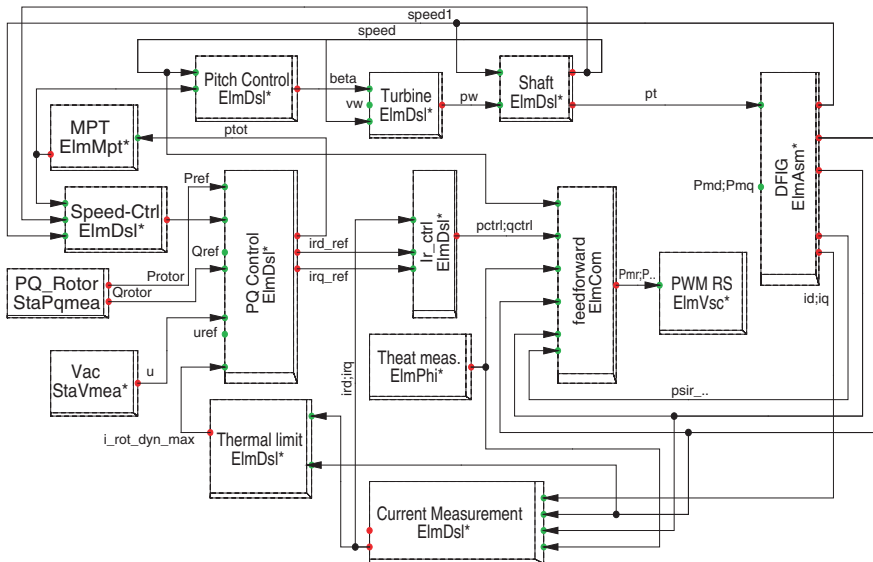


Fig. 9.9 Structure of the turbine and RSC models in PowerFactory

(v) Vac (*StaVmea*): stator voltage measurement; (vi) PQ_rotor: (*StaPqmea*): active and reactive power measurement of rotor-side converter; (vii) PQ control: generation of active and reactive reference currents for RSC; (viii) Theat_meas: angle measurement (*ElmPhi_pll*); (ix) MPT controller (not used in this chapter); (x) Speed-Ctrl: Speed controller; (xi) pitch controller; (xii) aerodynamic model of the turbine; (xiii) model of the shaft; (xiv) PWM RS: PWM converter (*ElmVscmono*); and (xv) thermal limit: rotor current limiter.

9.5.3 Initialization of Turbine Model and RSC

The initialization of all the RSC and turbine dynamic models of DFIG is done in a similar fashion like in Sect. 9.5.1. The initialization equations for each dynamic block of RSC and turbine models are defined as follows:

RSC current controller:

```
inc(xd)=up
inc(xq)=uq
inc(ird_ref)=ird
inc(irq_ref)=irq
inc0(up)=0
inc0(uq)=0
inc0(ird)=0
inc0(irq)=0
inc(bypass)=0
inc(i_synch)=0
```

Generation of active and reactive reference currents for RSC:

```
inc(Pref)=id_ref*u+Protor
inc(Qref)=(iq_ref+Ratio*u)*u +Qrotor
inc(x)=u
inc(du)=0
inc(xp)=Pref
inc(xq)=Qref
inc(uref)=u
inc0(id_ref)=0.5
inc0(iq_ref)=0
```

Speed controller:

```
inc(xi)=Pref/speed
inc(speed_ref)=speed
inc(xpc)=Pref
inc(i_enable)=1
inc(xramp)=1

vardef(Kptrq)='Speed Controller Proportional Gain';
vardef(Kitrq)='1/s'; 'Speed Controller Integral Gain';
vardef(Tpc)='s'; 'Output Limiter Time Constant';
vardef(p_max)='p.u.'; 'Active Power Upper Limit';
vardef(p_min)='p.u.'; 'Active Power Lower Limit';
vardef(dp_max)='p.u./s'; 'dP/dt Upper Limit';
vardef(dp_min)='p.u./s'; 'dP/dt Lower Limit';
```

Pitch controller:

```
inc(xp)=beta_min
inc(xip)=beta_min
inc(xic)=beta_min
inc(speed_ref)=speed
inc(beta)=beta_min

vardef(Kpp)='p.u.';'Blade Angle Controller Gain'
vardef(Kip)='s';'Blade Angle Controller Time Constant'
vardef(Tp)='s';'Servo Time Constant'
vardef(dbeta_max)='deg/s';'Max. dbeta/dt'
vardef(dbeta_min)='deg/s';'Min. dbeta/dt'
vardef(beta_max)='deg';'Max. beta'
vardef(beta_min)='deg';'Min. beta'
vardef(speed_ref)='p.u.';'Speed Reference'
```

Aerodynamic model of the turbine:

```
pwind=rhoAr*Cp*pow(vw,3)
Cp=sapprox2(beta,lambda,matrix_i)
lambda=Kb*speed/vw

inc(vw)=8
inc0(beta)=0
```

Model of the shaft:

```
inc(dphi12)=pt*Pgbase/Ptbase/speed_gen/Ktg
inc(xH)=speed_gen
inc(pw)=pt*Pgbase/Ptbase
inc(speed_tur)=speed_gen
```

9.5.4 LSC Model

Figure 9.10 shows the model of the LSC, which was built as a composite model and contains the following components: (i) LSC model: it includes the PWM converter (*ElmVscmono*); (ii) DC voltage controller; (iii) AC voltage controller; (iv) PLL: angle measurement (*ElmPhi_pll*); (v) DC voltage measurement (*StaVmea*); and (vi) AC voltage measurement (*StaVmea*).

9.5.5 Initialization of the LSC

The initialization equations of each dynamic block of LSC are presented below.

DC voltage controller:

```
inc(xidref)=id_ref*Tudc/Kudc
inc(udc_ref)=abs(udc)
inc(xr)=udc
inc(reset)=0

vardef(Kudc)='p.u.';'Proportional Gain'
vardef(Tudc)='s';'Integral Time Constant'
vardef(Min_idref)='p.u.';'Min. d-axis current reference'
vardef(Max_idref)='p.u.';'Max. d-axis current reference'
```

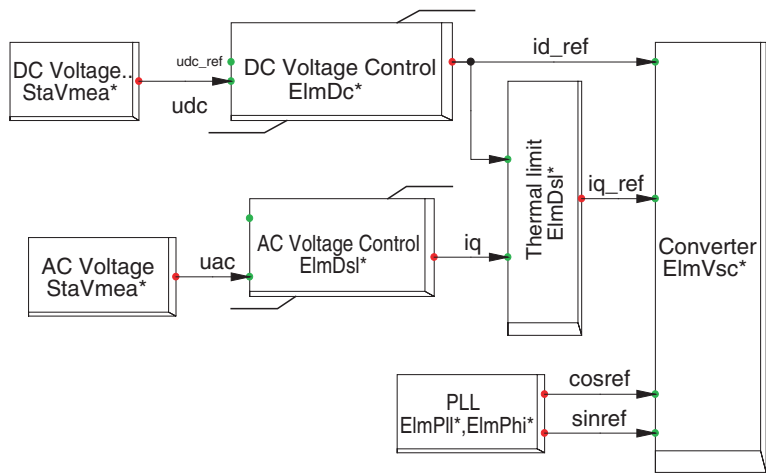


Fig. 9.10 Structure of the LSC model in PowerFactory

AC voltage controller:

```
inc(uac_ref)=uac
inc(xr)=uac
```

9.6 Implementation in DlgSILENT PowerFactory

The system shown in Fig. 9.11 has been used for testing purposes. The values of the parameters of all the components can be found in the implementation files *rueda02.pfd* (used for simulating the DFIG-based wind power plant) and *rueda03.pfd* (used for simulating dynamic equivalent). It has been modelled based upon the data from [6]. The time response in terms of bus voltage magnitude as well as of injected active and reactive power at bus WPP_CB33 (i.e. point of common coupling of the DFIG or WPP equivalent) to a 3-phase grid fault at bus B220 for the duration of 200 ms are analysed for illustrative purposes. The DFIG parameters were based on reference values of a 1.45-MW WG and are listed in Table 9.1. 60 DFIG-based

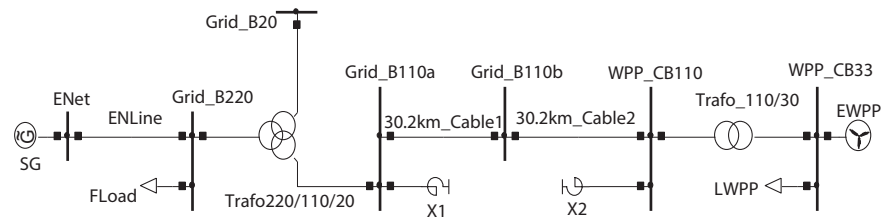


Fig. 9.11 Test system used for simulations with DFIG and WPP models

Table 9.1 DFIG parameters [3]

DFIG parameters	Value
Rated stator voltage (kV)	0.69
Rated rotor voltage (V)	1,863
Rated apparent power (MVA)	1.5
Rated speed (rpm)	1,485
No. pole pairs	2
Stator resistance (p.u.)	0.01
Stator reactance (p.u.)	0.1
Rotor reactance (p.u.)	0.1
Rotor resistance (p.u.)	0.01
Magnetizing reactance (p.u.)	3
Generator inertia (kgm^2)	75.4

WGs are operating in parallel in this test grid generating total of 87 MW of active power. The WPP represents the equivalent wind power generation of 90 MW.

The parameters of the dynamic equivalent, which were determined by applying the identification procedure given in [2], are summarized in Table 9.2.

9.6.1 Performance of the DFIG Model

During the fault, the voltage at the bus WPP_CB33 decreases as shown in Fig. 9.12a, the LSC and RSC sense the voltage drop and the outer loop of both the controllers generate the command for more reactive power. The total active and reactive power contribution of the DFIG is shown in Fig. 9.12b. It can be observed that the injected active power settles quickly at the pre-fault value.

Furthermore, from Fig. 9.12c, it can be seen that the LSC works as STATCOM and supports the grid voltage by injecting the reactive power. Therefore, the DFIG-based power plant is capable of voltage support by providing reactive power. Moreover, from the figure, it can be seen that there is an oscillatory shape in time evolution of the injected active power, which is attributed to the deadband in the voltage controller.

Table 9.2 Parameters of the WPP equivalent

Parameter	Value
u_k (%)	34.89384
P_{cu} (kW)	12,149.70
T_V (s)	0.573272
T_I (s)	0.040493
K_I (p.u.)	15.47351
K_{VC} (p.u.)	2.484421
i_{\max} (p.u.)	1.198953

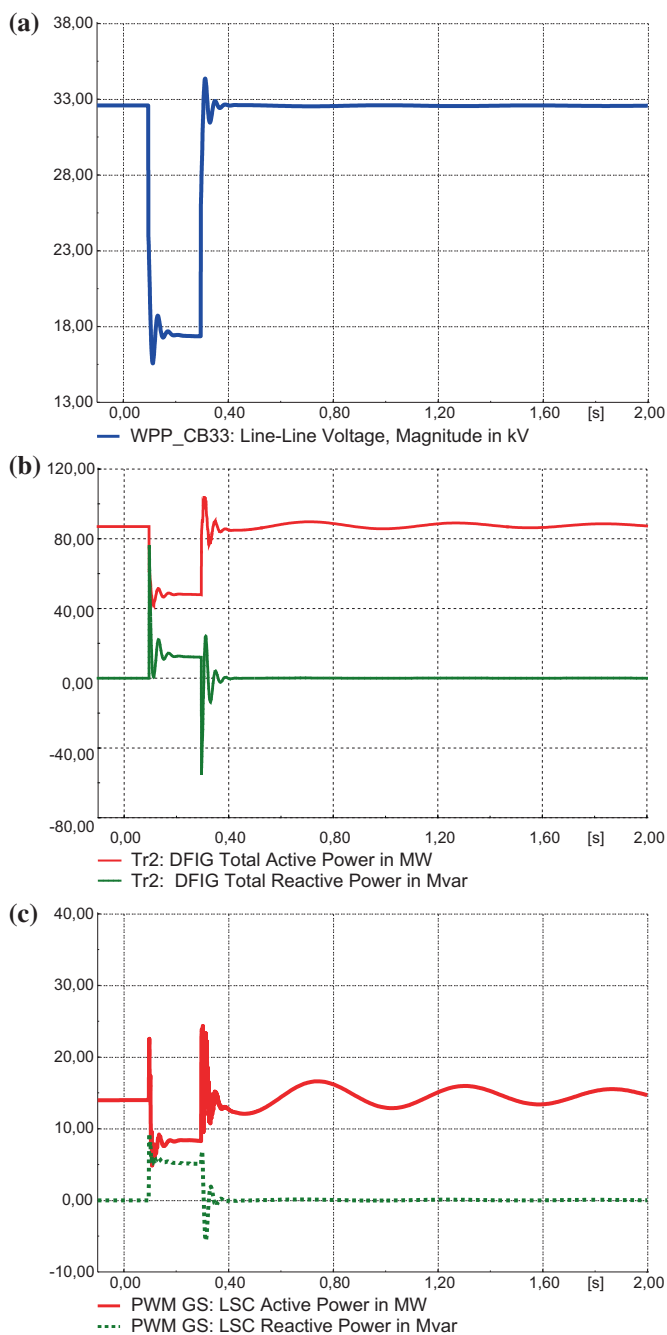


Fig. 9.12 Response of the DFIG-based WG during 3-phase fault at bus B220

9.6.2 Performance of the Dynamic Equivalent

Figure 9.13 shows the response of the WPP equivalent during the fault. It can be observed that the WPP supports the grid voltage by injecting the reactive power. Basically, it also works as STATCOM in this case, which is in correspondence with the previous findings concerning the expected response of power plants comprising DFIG-based generators.

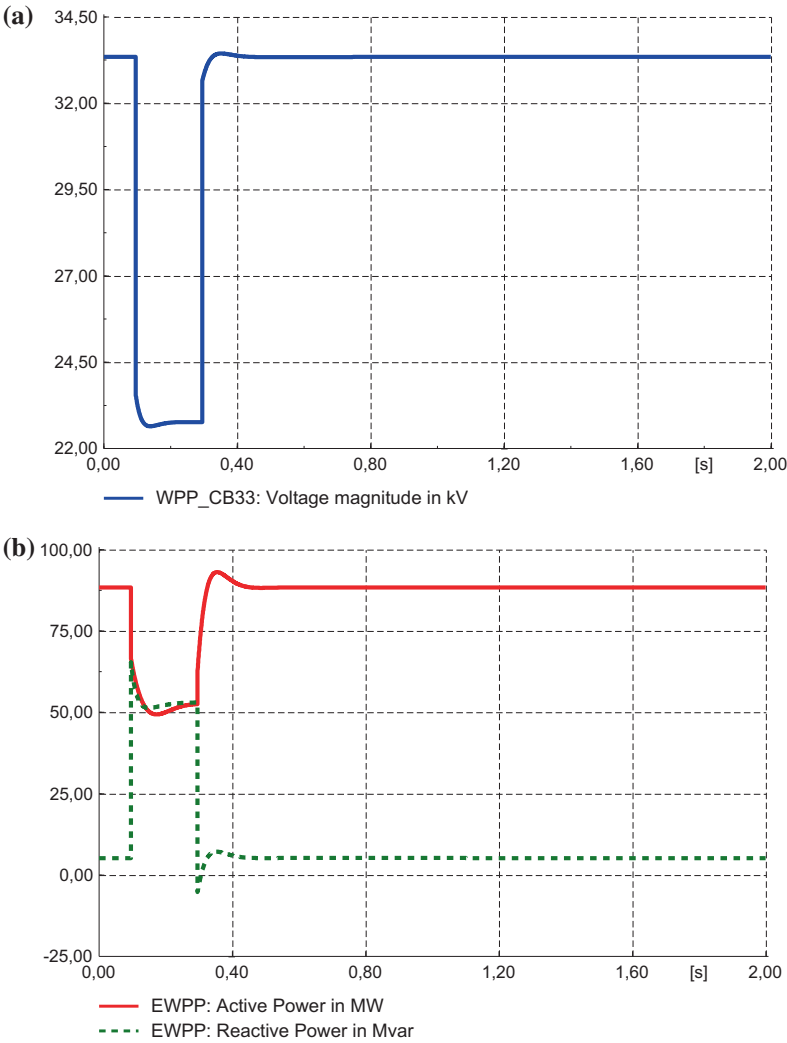


Fig. 9.13 Response of the WPP dynamic equivalent during 3-phase fault at bus B220

9.7 Conclusions

In this chapter, the implementation of simplified models of DFIG-based wind generators and wind power plant equivalents for RMS-type simulation in DiGSILENT PowerFactory was presented. The implementation and initialization of the models in PowerFactory was thoroughly explained. Based on a small test power system, the performances of both the models were analysed and discussed. The simulation results showed that both models are suitable to reproduce the response of wind power plants, including the capability for reactive power support, when voltage drops occur at the point of common coupling due to external faults. Moreover, it is worth pointing out that the WPP equivalent model should be extended to include speed control, including the equation, if longer simulation time spans are considered.

References

1. Ackermann T (ed) (2012) Wind power in power systems, 2nd edn. Wiley, West Sussex
2. Cepeda JC, Rueda JL, Erlich I (2013) VMOS-based approach for identification of dynamic equivalents from PMU measurements. In: Proceedings of the IEEE Grenoble PowerTech, pp 1–6
3. DiGSILENT PowerFactory User Manual (2013) DiGSILENT GmbH. Gomaringen, Germany
4. Engelhardt S, Feltes C, Fortmann J, Kretschmann J, Erlich I (2009) Reduced order model of wind turbines based on doubly-fed induction generators during voltage imbalances. In: 8th international workshop on large-scale integration of wind power into power systems as well as on transmission networks for offshore wind farms
5. Erlich I, Shewarega F (2006) Modeling of wind turbines equipped with doubly-fed induction machines for power system stability studies. IEEE PES power systems conference and exposition (PSCE '06), pp 978–985
6. Erlich I, Shewarega F, Feltes C, Koch F, Fortmann J (2012) Determination of dynamic wind farm equivalents using heuristic optimization. In: Proceedings of the IEEE power and energy society general meeting, pp 1–8
7. Fortmann J, Engelhardt S, Kretschmann J, Feltes C, Erlich I (2014) New generic model of DFIG-based wind turbines for RMS-type simulation. IEEE Trans Energy Convers 29 (1):110–118
8. Higgins P, Foley AM (2013) Review of offshore wind power development in the United Kingdom. In: Proceedings of 12th international conference on environment and electrical engineering, pp 589–593
9. Koch F, Erlich I, Shewarega F (2003) Dynamic simulation of large wind farms integrated in a multi-machine network. In: IEEE power and energy society general meeting
10. Kretschmann J, Wrede H, Mueller-Engelhardt S, Erlich I (2006) Enhanced reduced order model of wind turbines with DFIG for power system stability studies. IEEE international power and energy conference, pp 303–311
11. Muller S, Deicke M, De Doncker RW (2002) Doubly fed induction generator systems for wind turbines. IEEE Ind Appl Mag 8(3):26–33
12. Rueda JL, Erlich I (2011) Impacts of large scale integration of wind power on power system small-signal stability. In: Proceedings of the 4th international conference on electric utility deregulation and restructuring and power technologies, pp 673–681

13. Suwan M, Neumann T, Feltes C, Erlich I (2012) Educational experimental rig for DFIG based wind turbine. IEEE PES general meeting, pp 1–8
14. Tennet TSO GmbH (2012) Grid code—high and extra high voltage. Available via. <http://www.tennet.eu/de/en/customers/grid-customers/grid-connection-regulations.html>. Accessed 14 Mar 2014
15. Villa WM, Rueda JL, Torres S, Peralta WH (2012) Identification of voltage control areas in power systems with large scale wind power integration. In: Proceedings of the sixth IEEE/PES transmission and distribution: Latin America conference and exposition, pp 1–7

Chapter 10

Parameterized Modal Analysis Using DIgSILENT Programming Language

Sergio Pizarro-Gálvez, Héctor Pulgar-Painemal
and Víctor Hinojosa-Mateus

Abstract A parameterized modal analysis is proposed to evaluate power system small-signal stability and a simulation procedure using DIgSILENT programming language (DPL) is presented for this purpose. Modal analysis, or small-signal stability analysis, refers to the ability of a system to withstand small perturbations around an equilibrium point without reaching instability or displaying sustained oscillations. This is an important problem in real systems as sustained oscillations may cause mechanical failures in generating units and make the system vulnerable to the point of losing stability. In order to define the most critical scenario, the standard approach employed by the industry is to consider the system at its maximum loading. This chapter proposes that critical scenarios should be further sought as in even lighter loading conditions, low-frequency oscillations may dangerously appear. Using DPL, the system modal analysis is parameterized in order to search for critical operating points. To validate the importance of employing a parameterized modal analysis, a real case is studied: the Northern Chile Interconnected System. The obtained results applying parameterized modal analysis show that the system in a real operation only requires minor deviations from the programmed operating points to incur in dangerous low-frequency oscillations. These results validate the importance of searching for critical scenarios and DPL scripting is used

Electronic supplementary material The online version of this chapter (doi:10.1007/978-3-319-12958-7_10) contains supplementary material, which is available to authorized users.

S. Pizarro-Gálvez
SYSTEP Ingeniería y Diseños, Las Condes, 7550171 Santiago, Chile
e-mail: spizarro@systep.cl

H. Pulgar-Painemal (✉)
Department of Electrical Engineering and Computer Science, The University
of Tennessee, Knoxville, Knoxville, TN 37996, USA
e-mail: hpulgar@utk.edu

V. Hinojosa-Mateus
Department of Electrical Engineering, Universidad Técnica Federico Santa María,
2390123 Valparaíso, Chile
e-mail: victor.hinojosa@usm.cl

to program the steps of the proposed analysis. The scripts and applications presented in this chapter set the basis for additional development to identify critical scenarios for modal analysis in other real systems.

Keywords Power system dynamics • Electromechanical modes • RMS simulations • Small-signal stability • Power system damping

10.1 Introduction

Current power systems today are stressed due to an increasing loading, lack of generation and transmission plans due to deregulation, integration of significant amount of renewable energy (solar and wind), among others. These facts have caused the systems to operate closer to their security limits. In particular, in weak power systems such as those of some Latin-American countries, the stability becomes a problem mainly related to insufficient damping of electromechanical oscillations [1]. Electromechanical oscillations may dangerously cause mechanical failures in generating units and are also a precursor of large blackouts—these are devastating not only due to technical reasons, but also due to their negative economic consequences. During the last two decades, there have been many large blackouts around the world such as the US Northeast Blackout in August 2003 [2] and the India Blackout in July 2012 [3].

Electromechanical oscillations are studied using modal analysis, also known as small-signal stability analysis [4, 5]. Electromechanical oscillations are characterized in the range of 0.1–3 Hz and are associated with the oscillations of the rotating masses in synchronous generators and their corresponding oscillations of generated power. Most of the grid codes mandate that a power system must have all its modes properly damped with a damping ratio above 10 %. In case of a single contingency, e.g., generator or line outage, the damping ratio for all modes must still remain above 5 %. Note that power systems have diverse dynamics involving different timescales such as those associated with the electromagnetic or the electromechanical phenomenon [3–7]. As the manifestation of the electromagnetic phenomenon lasts in a time horizon in the order of milliseconds, it is not numerically efficient to model these dynamics in detail. On the contrary, these dynamics in power system analysis are typically assumed to be extremely fast and represented by algebraic equations. Consequently, the power system model is represented by a set of differential–algebraic equations, set that is solved when DIGSILENT RMS simulation is invoked.

In order to fulfill the grid code requirements, independent system operators have to anticipate the existence of electromechanical oscillations for different scenarios and take measures to avoid them. To identify the most critical conditions, a traditional approach employed by the industry consists in considering the operating point with the highest system loading. Unfortunately, as power systems are highly

nonlinear, this is not a recommendable approach. This chapter proposes to identify critical scenarios by performing a parameterized modal analysis, in which, the parameter should represent one of the main driver of oscillations. Typically, loading level is chosen as parameter, but other parameter may be also employed. A simulation methodology is presented to perform a parameterized modal analysis using DIgSILENT programming language (DPL). To validate the importance of employing a parameterized modal analysis, a real case is studied: the Northern Chile Interconnected System. The obtained results show that the system in a real operation only requires minor deviations from the programmed operating points to incur in dangerous low-frequency oscillations. These results validate the importance of searching for critical scenarios. DPL scripts and applications presented in this chapter set the basis for additional development to identify critical scenarios for modal analysis in other real systems.

10.2 Power System Dynamic Modeling and Stability

Electrical power systems are characterized by a large dimensionality, the occurrence of dynamics in different timescales, high parameter dependence, among others. Dimensionality is an evident characteristic; current-interconnected power systems, with thousands of elements that interact with each other, are considered to be largest systems created by the humankind. With respect to the involved dynamics, there are several distinctive physical phenomena that simultaneously take place [7, 8]. For example,

- **Electromagnetic dynamics:** Phenomenon related to the magnetic flux linkages in elements such as electrical machines, transformers and transmission lines. These dynamics occur in the timescale of milliseconds.
- **Electromechanical dynamics:** Phenomenon related to the rotating masses in generators and motors and their corresponding power oscillations injected to or absorbed from the grid. These dynamics occur in the timescale of seconds.

From a mathematical point of view, a power system dynamic behavior can be described by the following nonlinear model with N state variables and p parameters:

$$\dot{z} = w(z, \mu) \quad (10.1)$$

where

- | | |
|-----------------------------------|------------------------------------|
| $z \in \mathbb{R}^{N \times 1}$ | is a vector of state variables |
| $\mu \in \mathbb{R}^{p \times 1}$ | is a vector of parameters |
| w | is a vector of nonlinear functions |

Dynamics related to lightning or reactive power switching are very fast, and their modeling requires the representation in both time and space—traveling waves. Therefore, they are not represented by this differential model. In addition, very slow

dynamics such as those associated with thermal phenomenon in power plant boilers are not considered. With respect to the parameters, these can be classified in three categories: permanent, adjustable, and variable. Permanent parameters are constants of the system model, such as resistances and reactance of electrical machines, transformers, and lines. Adjustable parameters are related to controller's gains and references. Variable parameters can be associated with power system loading or, in the case of renewable generation, solar irradiance or wind speed. Due to the timescale issue, this formulation has two inconveniences when it comes to obtain a numerical solution: the possible occurrence of numerical instabilities and the excessive required simulation time—stiff systems. To avoid these problems, two models are derived by timescale separation: one model is employed in electromagnetic transient simulation (EMS), and the other is employed in electromechanical simulation—known as RMS simulation in DigSILENT PowerFactory.

10.2.1 Electromechanical Modeling

In order to derive a proper model for RMS simulation, the fast dynamics associated with the electromagnetic phenomenon need to be alternatively represented because a full characterization may cause numerical problems. Let $x \in \mathbb{R}^{n_s \times 1}$ be the state-variable vector related to the electromechanical dynamics and $y \in \mathbb{R}^{n_a \times 1}$ be the state-variable vector related to the electromagnetic dynamics— n_s and n_a are the numbers of variables, respectively. The terms of the state-variable vector z can be rearranged such that $z^T = [x^T y^T]$. Therefore, the full model is transformed to:

$$\dot{z} = w(z, \mu) \quad \Rightarrow \quad \dot{x} = w_1(x, y, \mu) \quad \wedge \quad \dot{y} = w_2(x, y, \mu) \quad (10.2)$$

An explicit solution for y in terms of x is sought, i.e., the manifold $y = h(x)$, which satisfies its own differential equation. Thus:

$$\dot{y} = \frac{\partial y}{\partial x} \dot{x} = \frac{\partial h}{\partial x} w_1(x, h(x), \mu) = w_2(x, h(x), \mu) \quad (10.3)$$

Finally, the system model can be reduced to the following:

$$\dot{x} = w_1(x, h(x), \mu) \quad (10.4)$$

This mathematical procedure solves the timescale issue; however, it is not practical as an explicit solution for y is hard to find. In power system modeling, in order to obtain a proper model for RMS simulation, a common approach is to use a zero-order manifold to approximately model the fast dynamics, and therefore, the following set of differential–algebraic equations (DAEs) can be obtained [7]:

$$\begin{aligned}\dot{x} &= f(x, y, \mu) \\ 0 &= g(x, y, \mu)\end{aligned}\tag{10.5}$$

In this set of DAEs, f and g are vector functions that model electromagnetic dynamics as infinitely fast. While the terms of the vector y correspond to algebraic variables related to the electromagnetic phenomenon, the terms of the vector x correspond to state variables related to the electromechanical phenomenon.

10.2.2 Linearized Model

For a given set of parameters, assume that the system is operating at an equilibrium point (x^e, y^e) . Thus, from Eq. (10.5), this point satisfies the following:

$$0 = f(x^e, y^e, \mu)\tag{10.6}$$

$$0 = g(x^e, y^e, \mu)\tag{10.7}$$

Small-signal stability refers to the system behavior when small perturbations are applied. Therefore, from a mathematical point of view, the system is well represented through a linear approximation of the model in a neighborhood of (x^e, y^e) . A Taylor series expansion of Eq. (10.5) around (x^e, y^e) becomes:

$$\dot{x} = f(x^e, y^e, \mu) + \nabla f_x(x^e, y^e, \mu)\Delta x + \nabla f_y(x^e, y^e, \mu)\Delta y + H.O.T.\tag{10.8}$$

$$0 = g(x^e, y^e, \mu) + \nabla g_x(x^e, y^e, \mu)\Delta x + \nabla g_y(x^e, y^e, \mu)\Delta y + H.O.T.\tag{10.9}$$

where $\Delta x = x - x^e$ and $\Delta y = y - y^e$. Using this definition, it turns out that $\dot{x} = \frac{d}{dt}(x - x^e) = \Delta \dot{x}$. Then, by neglecting the higher-order terms (H.O.T.), the following linear representation is obtained:

$$\begin{bmatrix} \Delta \dot{x} \\ 0 \end{bmatrix} = \begin{bmatrix} \nabla f_x & \nabla f_y \\ \nabla g_x & \nabla g_y \end{bmatrix} \bigg|_{(x^e, y^e, \mu)} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}\tag{10.10}$$

This representation still involves algebraic variables—variables that are allowed to respond infinitely fast and therefore their dynamics do not belong to the electromechanical phenomenon. By applying Kron's reduction to Eq. (10.10), algebraic variables are expressed explicitly in terms of the state variables as follows:

$$0 = \nabla g_x \Delta x + \nabla g_y \Delta y \Rightarrow \Delta y = -(\nabla g_y^{-1} \nabla g_x) \bigg|_{(x^e, y^e, \mu)} \Delta x\tag{10.11}$$

and the system is reduced to the following:

$$\Delta \dot{x} = \underbrace{(\nabla f_x - \nabla f_y \nabla g_y^{-1} \nabla g_x)}_{A_s} \Big|_{(x^e, y^e, \mu)} \Delta x = A_s(x^e, y^e, \mu) \Delta x \quad (10.12)$$

Note that for all x and y , the nonsingularity of the matrix ∇g_y is required. The matrix $A_s(x^e, y^e, \mu) \in \mathbb{R}^{n_s \times n_s}$ is called the system matrix, and it is dependent on the considered set of parameters μ and the system equilibrium point (x^e, y^e) . Eigenvalues of A_s give us information about the system dynamic behavior around the equilibrium point (x^e, y^e) . Note that other special dynamic behaviors such as limit cycles or chaotic phenomenon are not captured by this linearized model.

10.2.3 Modal Analysis

The model presented in Eq. (10.12) is a linear homogenous system—it only exhibits a zero-input response. The equilibrium point is precisely defined by $\Delta x = 0$ and $\Delta y = 0$, i.e., $x = x^e$ and $y = y^e$. In order to apply a small perturbation, an initial condition different than the equilibrium point is considered. Thus, the perturbed system is mathematically represented by the following initial value problem:

$$\begin{aligned} \Delta \dot{x} &= A_s \Delta x \\ \Delta x(t_0) &= \Delta x_0 \neq 0 \end{aligned} \quad (10.13)$$

The explicit solution of Eq. (10.13) is given by the matrix equation:

$$\Delta x(t) = e^{A_s(t-t_0)} \Delta x_0, \quad \forall t \geq t_0 \quad (10.14)$$

In general, A_s is a full matrix, and obtaining the exponential of A_s is not an easy task. An alternative solution is obtained as follows. Without loss of generality, consider that A_s has n_s distinct real eigenvalues $\{\lambda_1, \lambda_2, \dots, \lambda_{n_s}\}$ with their corresponding right eigenvectors $\{v_1, v_2, \dots, v_{n_s}\} \in \mathbb{R}^{n_s \times 1}$. Let M be the transformation matrix, $M = \{v_1, v_2, \dots, v_{n_s}\} \in \mathbb{R}^{n_s \times n_s}$. As the set of right eigenvectors are linearly independent, then the matrix M is full column rank and, therefore, invertible. Let q be the vector of the transformed state variables defined by $q = M^{-1} \Delta x$. Multiplying Eq. (10.13) from the left by M^{-1} , the following transformed model is obtained:

$$M^{-1} \Delta \dot{x} = M^{-1} A_s \Delta x \quad (10.15)$$

$$\begin{aligned} \Rightarrow \dot{q} &= (M^{-1} A_s M) q \triangleq \Lambda q \\ q(t_0) &= M^{-1} \Delta x_0 \end{aligned} \quad (10.16)$$

It can be proved that the transformed system matrix $\Lambda = (M^{-1}A_sM)$ is a diagonal matrix of the form $\Lambda = \text{diag}\{\lambda_1, \lambda_2, \dots, \lambda_{n_s}\}$. Consequently, this transformation decouples the system, and every transformed state variable is determined independently as $q_i(t) = q_i(t_0)e^{\lambda_i(t-t_0)}$, $\forall i \in \{1, 2, \dots, n_s\}$ and $\forall t \geq t_0$. The term $e^{\lambda_i(t-t_0)}$ is called the oscillation mode i . In the matrix form,

$$q(t) = e^{\Lambda(t-t_0)}q(t_0) = \begin{bmatrix} e^{\lambda_1(t-t_0)} & 0 & \dots & 0 \\ 0 & e^{\lambda_2(t-t_0)} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & e^{\lambda_{n_s}(t-t_0)} \end{bmatrix} M^{-1}\Delta x_0 \quad (10.17)$$

By transforming back to the original state variables, the following explicit solution is obtained:

$$\Delta x(t) = M e^{\Lambda(t-t_0)} M^{-1} \Delta x_0 \quad (10.18)$$

In case of having repeated eigenvalues, the transformed matrix Λ is not diagonal, but it has a Jordan form [9]. In the case of having a pair of complex eigenvalues, the complex terms of M , $e^{\Lambda(t-t_0)}$, and M^{-1} are canceled out to obtain real values for $\Delta x(t)$. Therefore, considering either repeated eigenvalues or complex pairs of eigenvalues (10.18) is still the valid solution of the system. In a power system, however, eigenvalues are typically distinct. In this particular case, this solution expresses that every single state variable is a linear combination of the oscillation modes, i.e.,

$$\Delta x_i(t) = \sum_{j=1}^{n_s} k_{ij} e^{\lambda_j(t-t_0)}, \quad \forall i \in \{1, 2, \dots, n_s\} \quad (10.19)$$

where k_{ij} is a constant that depends on M , M^{-1} , and Δx_0 . Note that some perturbations, or in other words some initial values Δx_0 , may excite or not a particular oscillation mode. Also, if a perturbation excites a particular mode λ_j , this mode may appear with some intensity in a particular state variable x_i . If this intensity is marginal, then the coefficient k_{ij} tends to zero. The coefficient k_{ij} not only tells us about how important is the mode j in the dynamic behavior of the state variable i , but also gives us information about the phase that the mode has in the variable. In order to have a better description of the dynamic behavior exhibited by the state variables, the concepts of participation factor, mode shape, oscillation vector, and damping ratio are defined.

1. *Participation factor* [10]: The participation factor (p_{jk}) is the sensitivity of an eigenvalue (λ_j) with respect to a diagonal entry (a_{kk}) of the system matrix (A_s). During the study of small perturbations, it can be proved that the participation factor p_{jk} is defined by [7]:

$$p_{jk} = \frac{\partial \lambda_j}{\partial a_{kk}} \approx \frac{w_j(k)v_j(k)}{w_j^T v_j} \quad (10.20)$$

where v_j and w_j are the right and left eigenvectors associated with the eigenvalue λ_j . The terms $v_j(k)$ and $w_j(k)$ correspond to the k th term of the right and left eigenvector, respectively. The magnitude of the participation factor is related to the intensity that the j th mode has on the state variable x_k . By definition, the right and left eigenvectors are orthogonal. If in addition, the left eigenvector w_j is defined such that $w_i^T v_j = \delta_{ij}$, where δ_{ij} is the Kronecker's delta function, then the set of left and right eigenvectors become orthonormal. It turns out that $W = M^{-T}$ where W is the matrix of left eigenvectors formed as $W = [w_1, w_2, \dots, w_n]$. Due to the orthonormality, the participation factor p_{jk} is now simply defined by $p_{jk} = w_j(k)v_j(k)$. Alternatively, a matrix of participation factors can be defined as

$$PF = W \circ M \quad (10.21)$$

where \circ is the element-wise product between W and M . Note that $PF(j, k) = p_{jk}$.

2. *Mode shape*: The mode shape (MS) is defined as the relative activity of the state variables when a particular mode is excited [3]. Based on Eq. (10.18), the angle of $v_j(i)$ —the i th term of the right eigenvector v_j —gives a measure of the phase displacement of the mode over the variable. Without loss of generality, assume that a particular mode j only participates in the state variables x_m and x_n . The phase displacement of the mode over the variables is given by $\angle v_j(m)$ and $\angle v_j(n)$. In case that $\angle v_j(m) - \angle v_j(n) = 0^\circ$, the oscillation in these variables is said to be in phase. In case that $\angle v_j(m) - \angle v_j(n) = 180^\circ$, the oscillation in these variables is said to be in counter-phase.
3. *Oscillation vector*: DIGSILENT PowerFactory gives the option of plotting participation factors. However, this graphical representation may be misleading as only the magnitude of the participation factors gives relevant information. There exists the risk that someone may use the angle of the participation factors to determine the relative displacement among the state variables. In order to avoid any confusion, a new definition is proposed in this chapter which is called an *oscillation vector*. An oscillation vector of the state variable x_i with regard to the mode j is defined as the following:

$$OS_{ji} = |p_{ji}| \angle v_j(i) \quad (10.22)$$

When the oscillation vectors are plotted, the vector magnitudes represent the participation of the mode over the variables, while the angle differences describe how the oscillation is manifested in the state variables, e.g., in phase or counter-phase.

4. *Damping ratio*: A damping ratio is a dimensionless measure of how damped an oscillation mode is. Considering a pair of complex eigenvalues $\lambda = \alpha \pm j\beta$, then the damping ratio of the oscillation is defined as follows:

$$\sigma = \frac{-\alpha}{\sqrt{\alpha^2 + \beta^2}} = -\cos(\angle \lambda) \quad (10.23)$$

Note that while positive damping ratios are related to stable systems, negative damping ratio are associated with unstable systems. In a second-order system, and in the presence of a pair of complex eigenvalues, it can be proved that an approximated solution of the oscillation is given by [8]

$$\Delta x(t) = \frac{\Delta x_0}{\cos(\phi)} e^{\frac{-\sigma}{\sqrt{1-\sigma^2}}\beta t} \cos(\beta t - \phi) \quad (10.24)$$

where $\phi = \sin^{-1}(\alpha/\sqrt{\alpha^2 + \beta^2})$. From this equation, it can be estimated how fast an oscillation is damped for a given damping ratio. For example, in case of having 5 % of damping ratio, an oscillation would be damped about 27 % in the first cycle of the oscillation. In power system dynamics, the most critical modes are the electromechanical ones. Grid codes typically impose that system modes must at least have a damping ratio of 10 %. Only in the existence of a single contingency, damping ratios may have a minimal value of 5 %.

10.2.4 Classification of Electromechanical Modes

Electromechanical modes are those oscillatory phenomena related to the exchange of kinetic energy among synchronous generators—energy stored at the rotating masses. Mathematically speaking, these modes are identified as those having the largest participation factors related to either the rotor angle or angular speed of synchronous generators. The exchange of kinetic energy in these electromechanical oscillations may be manifested in different ways which leads to the following classification [11]: *local mode*, typically a generator oscillates in counter-phase with the rest of the generators at a frequency between 0.8 and 1.5 Hz; *intraplant mode*, the generators of a power plant oscillate in counter-phase at a frequency between 2 and 3 Hz; and *interarea mode*, a group of generators in one area oscillate in phase and altogether in counter-phase with a group of generators from another area at a frequency between 0.1 and 0.7 Hz.

10.2.5 Parameterizing Modal Analysis

As shown in Eq. (10.12), the system matrix is dependent on the vector of parameters. When a parameter is varied, not only the equilibrium point changes but also the dynamic response of the system. Therefore, a system parameter is chosen and allowed to vary in discrete steps and in a predefined range. After the modal analysis is repeated for all parameter values, eigenvalues are traced in the complex plane and the dynamic responses are analyzed. During this study, the system may exhibit eigenvalues with a low damping ratio or may experience more drastic alterations such as a change in its stability or the appearance of multiple equilibrium points. Due to the latter, this study is also called bifurcation or branching analysis. However, in bifurcation analysis, other dynamic phenomena are also considered, e.g., limit cycles.

In a parameterized modal analysis, assume that the system is stable. Consider that a trajectory of a pair of complex eigenvalues moves from left to right in the complex plane when a certain parameter is varied. It is said that the system has reached a *Hopf bifurcation point* when this pair of complex eigenvalues is located exactly at the imaginary axis. The Hopf bifurcation is specified by the parameter value that causes this condition, and it defines a critical stability condition for the system. After the system reaches a Hopf bifurcation point, the system state variables may be orbiting in limit cycles or the system may have multiple equilibrium points—only the latter can be captured when a parameterized modal analysis is executed, requiring nonlinear theory to analyze limit cycles [12].

There exists another dynamic behavior that can be studied using a parameterized modal analysis: the existence of a limit-induced bifurcation. This type of bifurcation is related to the sudden loss of stability when a state variable hits its limit [13]. This phenomenon has been related to the shortage of reactive power in generators and has been also observed in wind farms [14]. Unfortunately, DIGSILENT Power-Factory does not consider variable limits in modal analysis as variables are relaxed by not taking into account hard constraints such as the technical limits of generating units. Note that DIGSILENT only prints a warning in the output window when a limit is hit.

10.3 DPL Scripting

The DPL offers an interface to the user for the automation of tasks in the Power-Factory program. The main idea of this interface is that the user can define its own automation commands (or scripts), develop some programs and new calculation functions. The DPL method distinguishes itself from the command batch method in several aspects. DPL offers the following:

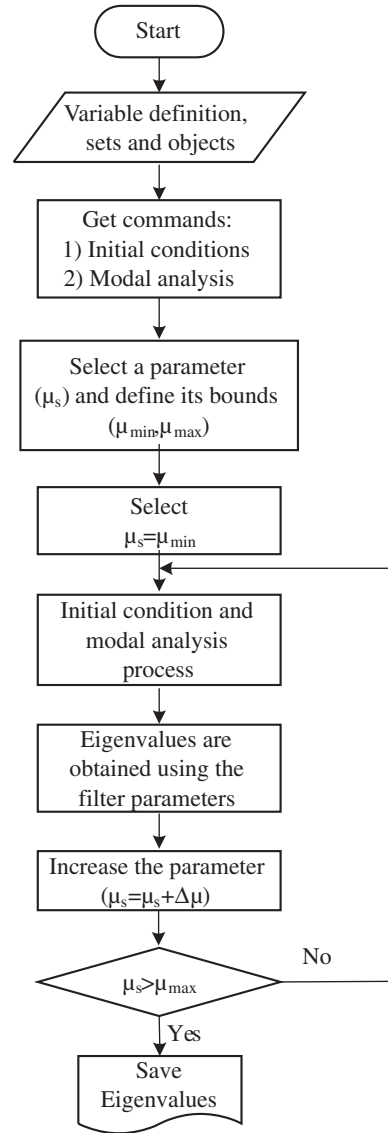
- (i) Decision and flow commands,
- (ii) Definition and use of user-defined variables,
- (iii) Flexible interface for input–output and for accessing objects,
- (iv) Recognition of mathematical expressions and functions.

The DPL command object (*ComDpl*) is the central element which is connecting different parameters, variables, or objects to various functions or internal elements and then puts out results or changes parameters. DPL command objects provide an interface for the configuration, preparation, and use of DPL scripts. These objects may take input parameters, variables, and/or objects, may pass these to functions or subroutines, and may then output results. Thus, the DPL script will run a series of operation and start calculation or other functions inside the script. In order to program a parameterized modal analysis, a script using DPL is used. The variable definitions, assignments and expressions, standard functions, program flow instructions, functions and subroutines, available commands and the DPL can be consulted in the DIgSILENT PowerFactory Manual [15].

10.3.1 Parameterized Modal Analysis Using DPL

Time-domain simulations in PowerFactory are initialized using the *PowerFactory* function (*ComInc*) in order to determine the initial conditions for all power system elements including all controller units and mechanical components. These initial conditions represent the steady-state operating point at the beginning of the simulation, fulfilling the requirement that the derivatives of all state variables of the power system elements, e.g., generating machines, controllers, and loads, are zero. The authors have included a flowchart in Fig. 10.1 showing the main stages accomplished by the DPL script. In this flowchart, the fifth block defines the parameter which is being considered in the parameterized modal analysis. As mentioned before, this parameter should be chosen as the one that is driven some particular dynamic behavior. For example, the parameter could be the active power of an important consumer, a specific power transfer from an injection point to a consumption point, or the wind speed in the case of wind turbines or solar irradiance in the case of solar power. When the parameter is selected, the range in which the parameter is allowed to vary is defined. In this case, the parameter is allowed to vary between a minimum and a maximum value defined by μ_{\min} and μ_{\max} , respectively. The parameter is set to its minimum value and the modal analysis is performed. When the results are obtained, the parameter is increased by $\Delta\mu$ and if the new parameter value is less than the maximum value, the modal analysis is repeated.

Fig. 10.1 Procedure to perform a parameterized modal analysis



10.3.2 Creating a New DPL Command

When a new DPL command is created, a dialogue window is used to specify the parameters, objects, and the script. As an example, a DPL command object employed in this chapter is shown in Fig. 10.2. As being presented later, this DPL command performs a parameterized modal analysis applied to the Northern Chile

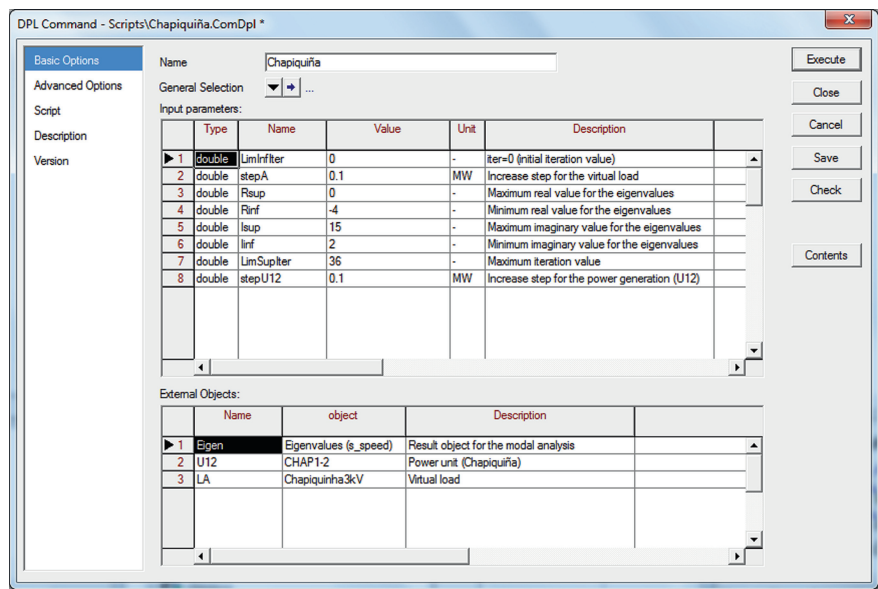


Fig. 10.2 Modal analysis DPL example (.ComDpl)

Interconnected System. In this case, eigenvalues are calculated, while the loading of CHAP power plant is increased in steps of 0.1 MW (*stepU12* as shown in the figure).

The section *Input Parameters* is used to define variables that are accessible in the DPL command. The variables *Rsup*, *Rinf*, *Isup*, and *Iinf* are used to filter the calculated eigenvalues. While *Rsup* and *Isup* contain the maximum values of the real and imaginary parts of the eigenvalues, respectively, *Rinf* and *Iinf* contain the minimum values. In order to study only the effect of the loading of CHAP power plant units (CHAP1-2.*ElmSym*), a virtual load is connected at the terminal of the plant. The virtual load is included in the same bus to compensate the increase of the power generation. In this manner, the power flows through the network and the equilibrium points at other machines and load should remain unchangeable when the loading of CHAP power plant units is varied. In this case, the changes in the loading are chosen as a parameter in the parameterized modal analysis (μ_s). The parameter *stepU12* corresponds to the increase of the active power generation, and parameter *stepA* corresponds to the increase of the virtual load. Note that $\mu_s = \text{stepU12} = \text{stepA}$. *LimInfIter* and *LimSupIter* correspond to the minimum and maximum values of iterations during the parameterized modal analysis. In addition to the Input Parameters section, external objects are used in the script. *U12* (CHAP1-2.*ElmSym*) is the power unit object selected in the eigenvalue analysis, and *LA* (Chapiquinha3 kV.*ElmLod*) is the load used to compensate the increase of

the power generation of the CHAP power plant units. Lastly, *Eigen* is a result object (Eigenvalues (s_speed).*ElmRes*); a result object is required to store the results of the modal analysis.

10.4 Application to a Real System

10.4.1 Northern Chile Interconnected System (SING)

The *Northern Chile Interconnected System* (in Spanish, SING: *Sistema Interconectado del Norte Grande*) is an isolated electrical system of Chile, in the northern regions of Arica-Parinacota, Tarapaca, and Antofagasta. The system is operated by the *Center of Economic Load Dispatch* (in Spanish, CDEC-SING: Centro de Despacho Económico de Carga del SING) whose main goal is to preserve the system security and to ensure the most economical operation for the whole system. During 2013, 78.8 % of the energy was consumed by mining companies, 10.6 % by end consumers, 7.9 % by power plants auxiliary consumption, and 2.7 % by power transmission losses. The SING's main characteristics are presented in Table 10.1, and the SING's one-line diagram is shown in Fig. 10.3. Due to the lack of hydroelectric resources in the northern regions, the system is predominantly thermal with about 99.2 % of its installed capacity being related to coal, fuel, petroleum diesel, and combined cycle power plants. Cogeneration and two small hydro power plants represent only a 0.8 % of the SING installed capacity. Being a small system and having a few consumers with a large fraction of the total demand, the SING experiences very particular dynamic responses in a regular basis [16].

10.4.2 Base Case

The SING dynamic model is obtained from the CDEC-SING's Web site [16]. The system version with regard to all its constitutive elements corresponds to the one

Table 10.1 SING characteristics as of December 2013

Maximum gross generation [MW]	2,226
Gross energy production [GWh]	17,237
Maximum gross demand [MW]	2,060
Gross energy consumption [GWh]	15,882
Transmission losses [GWh]	468
Installed capacity [MW]	4,601
Number of generating units	94
System inertia [s] ^a	300

^a The system inertia is estimated using a power base of 100 [MVA]



Fig. 10.3 One-line diagram of the SING

from April 2013. This model was built in DIgSILENT PowerFactory version 14.1, but it is exported to version 15.0, the most current version up to date. In general, the model considers 407 buses, 188 transmission lines from 220 to 13.8 [kV], and 64 equivalent generating units. CDEC-SING defines three different load profiles in their studies: the high, medium, and low load profiles. These profiles are defined in terms of the total system demand, and therefore, a unit not necessarily has its maximum generation at the high load profile. This aspect is very important, as the high load profile may not imply the most critical scenario from a modal analysis point of view.

Because the maximum demand is about half of the installed capacity, 43 generating units are not committed to generate in the three loading profiles. A special case is Salta Power Plant which is located in Argentina. This power plant is connected to the system through a 345-kV transmission line of 408 km. However, as this plant was not included in the 2013 generation scheduling, it is not considered in this work. The economic load dispatch for all three scenarios is presented in Table 10.2. For the sake of simplicity, voltage references in all generators are set to 1.00 pu.

Table 10.2 Economic load dispatch for the scenarios of high, medium, and low load profile of the SING

Power plant	Unit	High [MW]	Medium [MW]	Low [MW]
Andina	CTA	150	150	150
Angamos	ANG1	230	240	220
Atacama	TG1A	95	95	95
Atacama	TV1C	60	60	60
Angamos	ANG2	230	240	220
Cavancha	CAVA	2.7	2.7	2.7
Chapiquiña	CHAP ^a	3	3	5
Hornitos	CTH	150	150	0
Mejillones	CTM1	149	149	149
Mejillones	CTM2	154	154	154
Norgener	NT01	112,6	96	125,6
Norgener	NT02	120	130	130
Tarapaca	CTTAR	140	140	140
Tocopilla	U12	60	60	80
Tocopilla	U13	60	60	80
Tocopilla	U14	122	122	122
Tocopilla	U15	116	100	116
Tocopilla	U16	220	220	178
Backup gen	Chuquic.	24	24	24
Backup gen	Q. Blanca	33	33	33
Cogeneration	PAM	25	17	25
Total generation		2,256.3	2,245.7	2,116.7

^a Chapiquiña Power Plant has two generating units, but for simplicity, an equivalent unit is considered (CHAP)

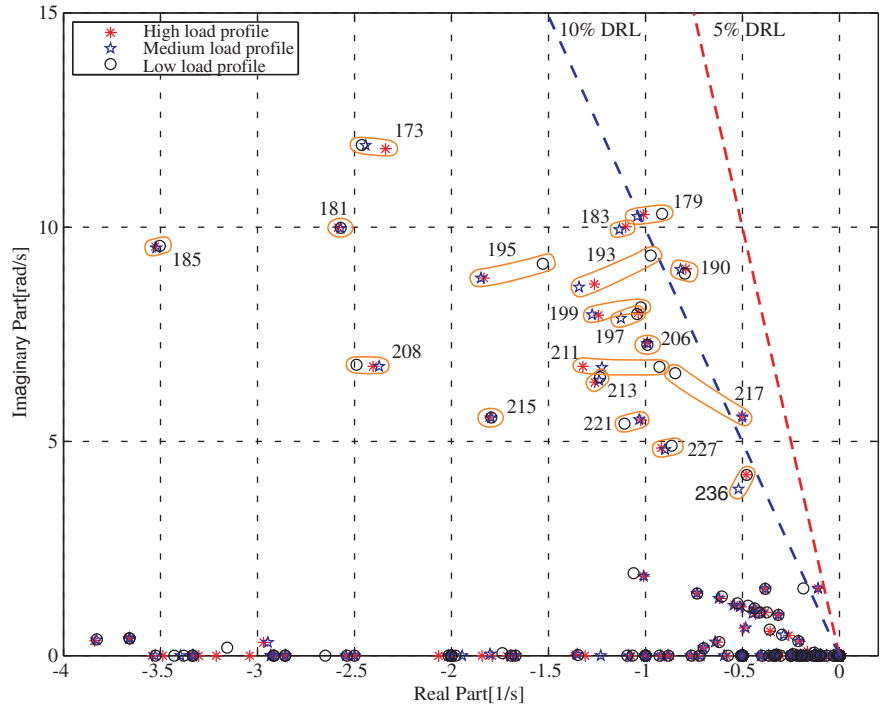


Fig. 10.4 Dominant eigenvalues considering the scenarios of high, medium, and low load profile

Using these three load profiles with their corresponding economic dispatch, the modal analysis is executed. DigSILENT PowerFactory automatically calculates the system equilibrium point, the system matrix, and the eigenvalues when the modal analysis procedure is invoked. In Fig. 10.4, the location of the most dominant eigenvalues is shown for these three scenarios and also the 5 and 10 % damping ratio locus (DRL) are drawn to easily identify the most critical eigenvalues. DigSILENT PowerFactory randomly assigns a tag number to every eigenvalue for identification purposes; using participation factors, it has been verified that when a different scenario is studied, although the numerical value of an eigenvalue may change, the assigned number corresponds to the same dynamic phenomenon. The dynamic characterization of the three most critical eigenvalues is as follows:

1. *Eigenvalue No. 179*: Interplant mode related to the kinetic energy exchange between the two generating units of Norgener Power Plant. These units oscillate in counter-phase. At the high load profile, the oscillation has a damping ratio of 9.77 % and a frequency of 1.64 Hz.
2. *Eigenvalue No. 190*: Local mode related to the kinetic energy exchange between the units NTO1 and NTO2 of Norgener Power Plant and the system. Units NTO1 and NTO2 jointly oscillate in counter-phase with the rest of the generators, but mainly with the unit CTH of Hornitos Power Plant and the unit CTA

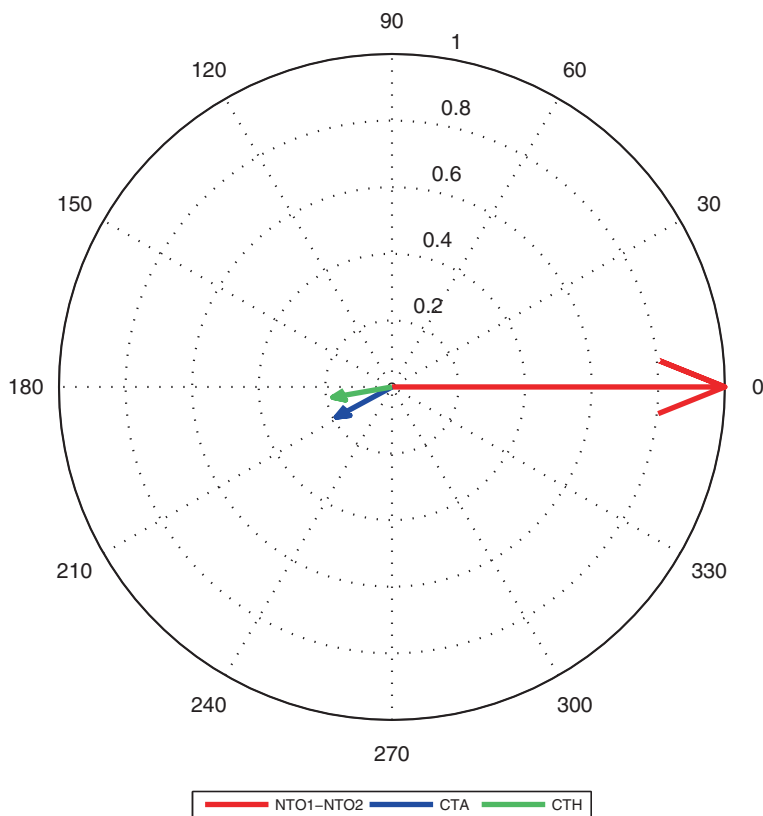


Fig. 10.5 Oscillation vectors related to eigenvalue No. 190

of Andina Power Plant, who have a participation factor of 20 %. At the high load profile, this oscillation is characterized by a damping ratio of 8.74 % and a frequency of 1.44 Hz. The oscillation vectors related to this mode are shown in Fig. 10.5.

3. *Eigenvalue No. 217*: Local mode related to the kinetic energy exchange between the unit CHAP of Chapiquiña Power Plant and the system. This unit is very distant from the rest of the generators, and it is located in the north side of the system. CHAP oscillates in counter-phase with the system. In addition, it is observed that the unit CTTAR of Tarapaca Power Plant is also dragged by this oscillation and has a participation factor of about 5 % with a phase displacement with respect to CHAP of about 29°. The unit CTTAR is also located in the north side, but it is closer to the rest of the generators than the unit CHAP. At the high load profile, this oscillation is characterized by a damping ratio of 8.94 % and a frequency of 0.89 Hz.

Note that some eigenvalues are unaltered by the load profile, e.g., eigenvalues No. 181 and No. 215, while others are considerably affected, e.g., eigenvalues No. 195 and No. 217. As the modes are strongly related to one or more generators, an initial guess to justify this result is to assume that the reduction of damping ratios is due to the increase of the generators loading. Eigenvalues No. 195 and No. 217 are chosen to verify this statement. By using participation factors, note that the eigenvalue No. 195 is strongly related to generators U12 and U13 of Tocopilla Power Plant, and the eigenvalue No. 217 is strongly related to the unit CHAP. In Table 10.2, these aforementioned units have a higher loading in the low load profile. Therefore, while eigenvalue No. 195 increases its damping ratio with a higher loading, eigenvalue No. 217 decreases it. In conclusion, the most critical scenarios from a modal analysis point of view cannot be obtained by simply considering the highest generator loading. In order to identify critical scenarios, it is proposed to perform a parameterized modal analysis around a certain load dispatch in which some generator loadings are defined as parameters.

10.4.3 Effect of Generating Unit Loading Factor

For the sake of brevity, only eigenvalue No. 217 is taken into account. For the purpose of studying the effect of unit CHAP loading, a parameterized modal analysis is executed, using the loading as a parameter. In addition, to ensure that modal changes are only attributed to the unit loading, a virtual load is connected at CHAP power plant bus. This virtual load adjusts its consumption in order to keep a constant net power injected by CHAP to the system—minimal changes in the system equilibrium point are sought by introducing this load. Still, due to the internal losses and flux leakage of the units at CHAP power plant, the modal changes might also be attributed, partially, to the action of the unit's exciter. The system is assumed to be operating under the high load profile. The injected power by CHAP is 3 [MW]. This generated power, P_G , is defined as a parameter, and it is allowed to vary between 3 and 10.2 [MW], in steps of 0.2 [MW]. The virtual load has a demand of $(P_G - 3)$ [MW] with $3 \leq P_G \leq 10.2$ [MW].

10.4.3.1 Creating the DPL

The proposed event considers a critical power plant (U12), CHAP power plant, in order to determine its loading effect in the power system dynamics. Next, the initial condition and modal analysis commands are executed. The simulation uses an iterative procedure to solve load flows and to integrate the dynamic model's state variables simultaneously. The most important eigenvalues are saved in a matrix object. The developed DPL script is shown below. In the code, several comments are included to explain step-by-step the parameterized modal analysis accomplished.

```

object Inc,Modal,ResulModal,GrB,ViPg,oVi; !objects used in the script
double Nvar,Nval,deltaP,pp,pp12,pp13,ccAp1,ccAq1,ccBp1,ccBq1,stepLoadAP,stepLoadAQ;
int xx,ww,yy,a,b,c,aa,a1,bb,kk,cc,Stp,zz;
!Start modal analysis
ClearOutput(); !clears the output window
EchoOff(); !de-activate the user-interface
!Create graph
GrB = GetGraphBoard();
ViPg = GrB.GetPage('Modal analysis',1);
oVi = ViPg.GetVI('Modal analysis','VisXyplot',1);
oVi:vStyle = 5; !point marker
oVi:vWidth = 150; !point width
Inc = GetCaseCommand('ComInc'); !initial conditions
Modal = GetCaseCommand('Modal.ComMod'); !modal analysis
pp12 = U12:e:pgini; !CHAP power plant U12 (power unit)
ccAp1 = LA:e:plini; !Virtual load
bb = LimInf1ter; !minimum and maximum values used by the filter
kk = LimSup1ter;
Inc.Execute(); !initial condition
Modal.Execute(); !modal analysis
LoadResData(Eigen); !load the modal simulation
Nvar = ResNvars(Eigen); !obtain information about the result object
Nval = ResNval(Eigen,0);
A = Nval; !auxiliar variables
B = bb+kk;
C = 2*b;
Xx = 0;
Aa = 0;
Eigenvalues = GetCaseCommand('eigenvalues.IntMat'); !real and imaginary output matrix
ResulModal = this.CreateObject('ElmRes','Results');
ResulModal.AddVars(this,'b:real','b:imaginary');
ResulModal.Init();
oVi.SetVal(ResulModal,'ResFile',0); !configure the plot
oVi.SetVal(this,'pyObj',0);
oVi.SetVal('b:real','xVar',0);
oVi.SetVal('b:imaginary','yVar',0);
for(cc=bb ; cc <= kk; cc=cc+1) { !Iterative process
    aa = aa+1;
    if ( {cc<=kk}.and. {cc>bb} ) {
        U12:e:pgini = U12:e:pgini+stepU12;
        LA:e:plini = LA:e:plini+stepA; }
    else {
        U12:e:pgini = U12:e:pgini;
        LA:e:plini = LA:e:plini;
    }
    Inc.Execute();
    Modal.Execute();
    LoadResData(Eigen);
    Nvar = ResNvars(Eigen);
    Nval = ResNval(Eigen,0);
    printf('Active power(%)=%.3f: P_A(%)=%.3f\n',U12:loc_name,U12:e:pgini,LA:loc_name,LA:e:plini,LA:loc_name,LA:e:qlini);
    a1 = 2*aa;
    yy = 0;
    ww = 0;

```

```

for (xx=0; xx<Nval; xx=xx+1) {           !load the row data
  GetResData(real,Eigen,xx,yy);           !real & imaginary component
  GetResData(imaginary,Eigen,xx,yy+1);
  if ({real<Rsup}.and.{real>Rinf}) {      !criterion to filter the eigenvalues
    if ({imaginary<Isup}.and.{imaginary>Inf}.or.{imaginary>-Isup}.and.{imaginary<-Inf}})
    {
      ww = ww+1;
      eigenvalues.Set(ww,a1-1,real); !eigenvalues result
      eigenvalues.Set(ww,a1,imaginary);
      ResulModal.WriteDraw();
    }
  }
}
}
}
oVi:auto_y = 1;           !set the plot
oVi:auto_x = 1;
ViPg.DoAutoScaleX();
ViPg.DoAutoScaleY();
U12:e:pgini = pp12;       !set the initial values in the power system
LA:e:plini = ccAp1;
Delete(ResulModal);

```

When executing the DPL code by pressing *Execute*, an output matrix is created (eigenvalues.*IntMat*) which contains the filtered eigenvalues for all the values of the parameter. Given a parameter, the calculated and filtered eigenvalues are stored in a submatrix of two column. The first row corresponds to the first eigenvalue, while the last row corresponds to the last eigenvalue. In the first column of this submatrix, the real part of the eigenvalues is stored. In the second column of this submatrix, the imaginary part of the eigenvalues is stored. When the parameter is increased by $\Delta\mu$, the calculated eigenvalues are once again stored as a submatrix of two column, to the left of the last stored submatrix (see Fig. 10.6). In order to observe the trajectories that the eigenvalues take for the different values of the parameter, proceed in the following fashion using scatter plot. Choose the first eigenvalue which is stored in the first row. Take the first pair of columns in the row. Draw in the complex plane a mark considering in the horizontal axis the real part of the eigenvalue (first column of the pair), and in the vertical axis, the imaginary part of the eigenvalue (second column of the pair). Repeat the drawing for the same eigenvalue (same row) but for the next value of the parameter (next pair of columns). When all the pair of columns are completed, the trajectory of the eigenvalue should be visualized in the plot. Repeat this procedure for the next row, which is associated with the following eigenvalue. For visualization, the output matrix and the eigenvalue trajectory plot are shown in Fig. 10.6.

10.4.3.2 Analysis of the Results

The eigenvalue trajectory obtained for this case is shown in Fig. 10.7. Note that eigenvalue No. 217 is clearly affected by unit CHAP loading, but there is no clear indication as either an increasing or decreasing loading level would set the worst-

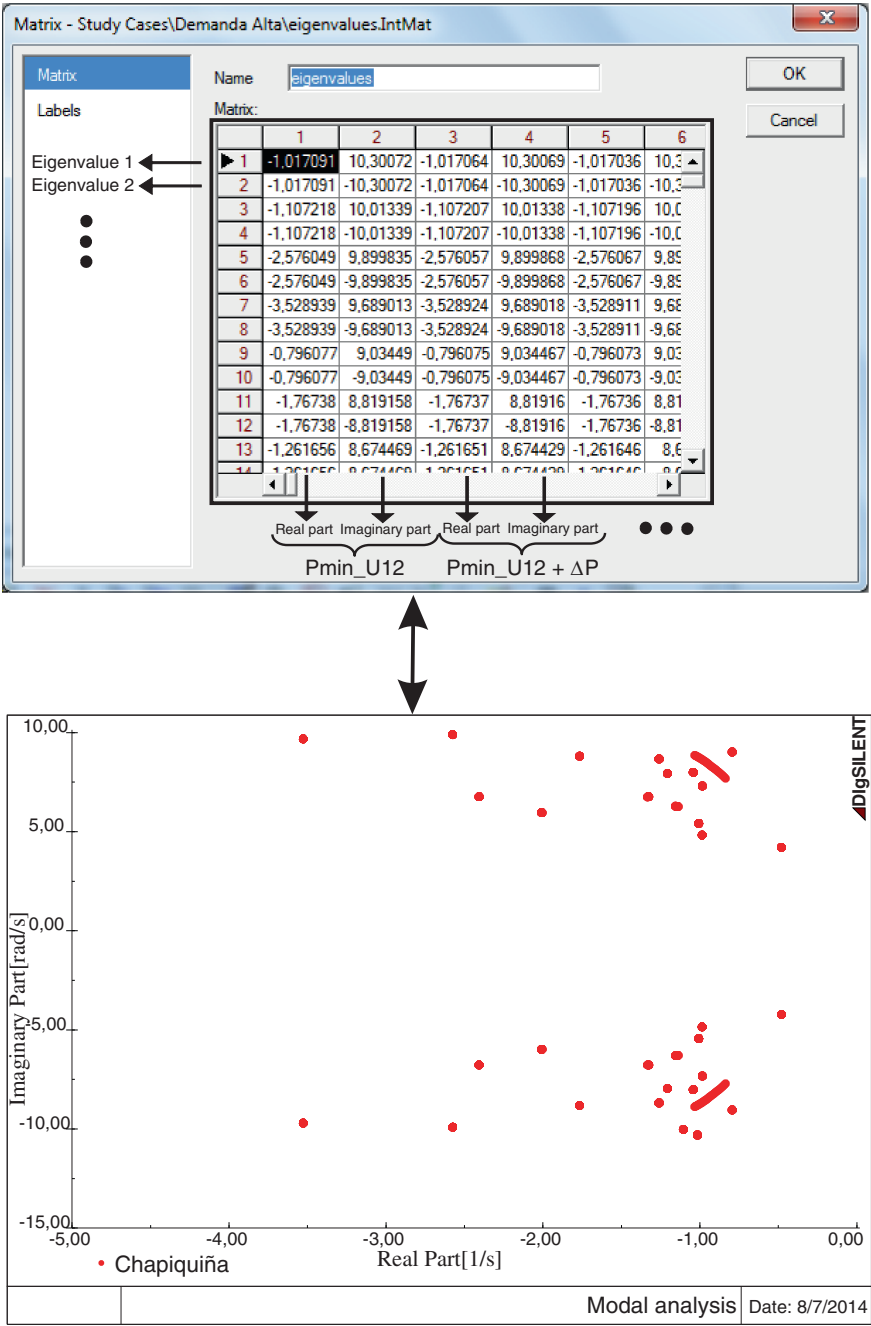


Fig. 10.6 Output matrix after performing a parameterized modal analysis

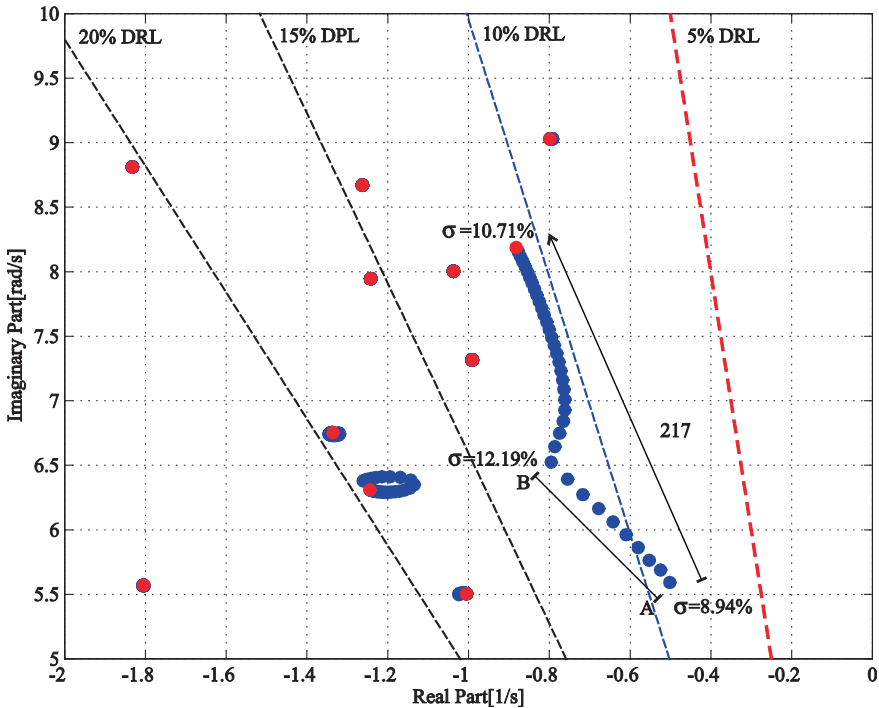


Fig. 10.7 Effect of CHAP power plant loading

case scenario for modal analysis. In fact, the eigenvalue pathway follows an irregular trajectory. From point A to point B, CHAP's generated power is increased from 3 to 5 [MW] and the eigenvalue increases its damping ratio to 12.19 %. If CHAP's generated power is further increased to 10.2 [MW], the eigenvalue tends to decrease its damping ratio, but its trajectory is bounded, reaching a final damping ratio of 10.71 %. In conclusion, loading of unit CHAP does not seem critical, and its generated power variation should not create critical conditions with poorly damped oscillations. This type of analysis should be repeated for any other unit related to eigenvalues that suffer large variations when different load profiles are tested, e.g., units related to eigenvalue No. 193.

10.4.4 Effect of the Mismatch Between Economic Dispatch and Real Operation

In power systems, it is typical to observe deviations between the programmed and real-time operation. Consumers do not necessarily consume their declared power as industrial process and domiciliary usage cannot be predicted with accuracy. As a

result, generating units need to adjust their production in order to closely match the total demand, therefore, large deviations of electric frequency can be avoided. In steady–steady, there are selected units that have to fulfill this role in what it is called the secondary frequency regulation.

For the purpose of identifying loads and generators with the largest mismatch between their programmed and real operation, the SING's real operation during 2012 is analyzed and, in particular, the hourly operation over 100 days is taken into account. These days are chosen randomly. By identifying these loads and generating units, it is also possible to identify certain patterns that correlate increases/decreases of power between generators and loads. For simplicity, these patterns are modeled by the use of power transfers. These power transfers are conceptualized as the increase/decrease of power at one or multiple loads and the corresponding increase/decrease of generated power from those units in charge of the secondary frequency regulation. Four power transfers are identified and presented in Fig. 10.8. The group of generating units that compensate the load variations is dependent on the employed load profile—either low or high. Out of the four identified power transfers, only the transfer L1 is described and analyzed in this document. The characteristics and analysis of the rest of the transfers are reported in [17]. Transfer L1 considers the generating units ANG1, ANG2, and CTA (to regulate frequency) and the equivalent loads at Collahuasi, El Abra, and Radomiro Tomic substations. The transfer deviation, ΔP , is defined as the parameter of study. When the parameter is increased, the loads at Collahuasi, El Abra, and Radomiro Tomic substations are increased in 30, 30, and 40 % of ΔP , respectively. When $\Delta P < 0$, the load is reduced according to these percentages at the aforementioned substations. At the same time, the generated power from ANG1, ANG2, and CTA is changed by 47.6, 47.6, and 13.3 % of ΔP , respectively. The parameter is defined in the range of

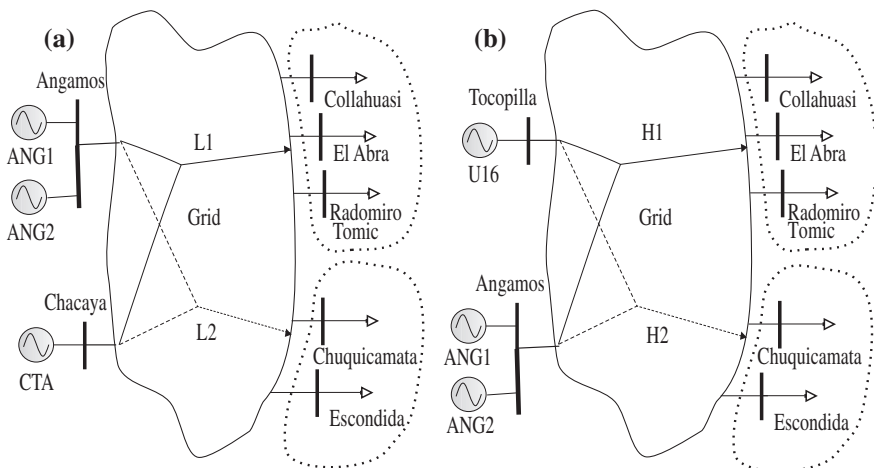


Fig. 10.8 Definition of four power transfers: **a** at a low load profile (transfers L1 and L2), **b** at a high load profile (transfers H1 and H2)

$-115 \leq \Delta P \leq 60$ [MW]. To perform this analysis, a similar DPL script used in Sect. 10.4.3 is employed.

By performing a parameterized modal analysis using ΔP as the parameter, the pathway of eigenvalues in the complex plane is obtained; the result is shown in Fig. 10.9. Most of the dominant eigenvalues are affected in some way by this transfer; however, eigenvalue No. 211 suffers the most important change when the transfer is reduced by 115 [MW]. Note that in the base case, this eigenvalue was not considered a critical one; however, if a mismatch between the programmed and real-time operation coincides with a reduction of 115 [MW] at these buses and generators, then this eigenvalue becomes critical exhibiting a poor damping ratio of 9.55 %. This operating point is not accepted by the Chilean Grid Code. As transfer L1 is defined based on the real operation in 2012, this new critical scenario seems to be likely. Note that, contrary to what the common sense dictates, this critical condition—from a modal analysis point of view—occurs when the system is operating at a lighter load. Further analysis reveals that this poorly damped mode is related to both an excessive bus voltage at Lagunas substation and the under-excited operation of unit CTTAR of Tarapaca Power Plant. As a possible solution to this critical condition, authors suggest evaluating the installation of new automatic

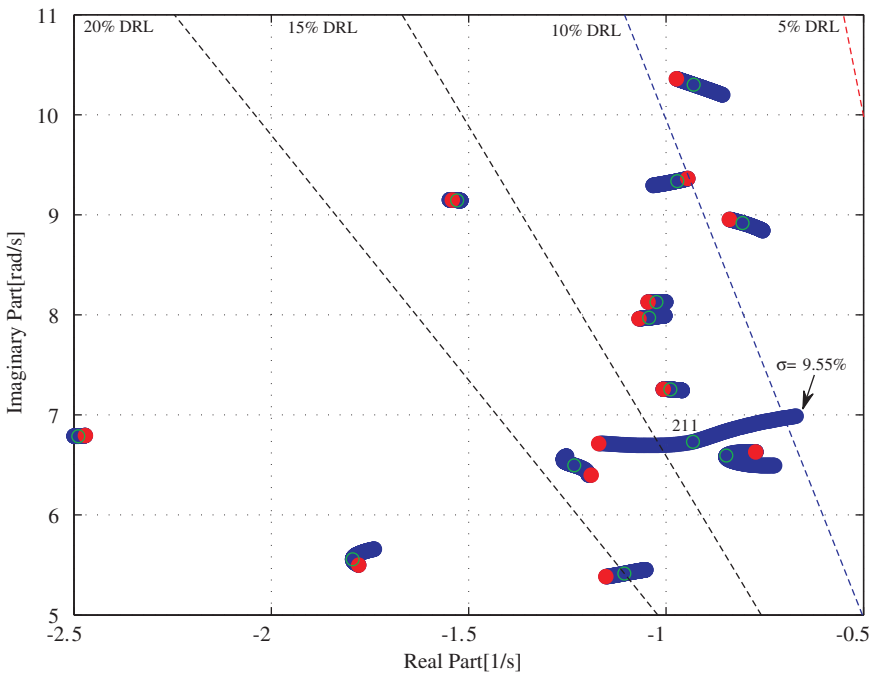


Fig. 10.9 Parameterized modal analysis considering transfer L1 and transfer changes of ΔP . The parameter ΔP is allowed to vary between -115 and 60 [MW]

voltage control systems at Lagunas substation to ease the operation of CTTAR. More details of this analysis are found in reference [17].

10.4.5 Final Remarks

The authors recommend that for establishing the most critical scenarios for modal analysis, not only the standard approach of employing maximum load profile should be considered, but also other likely scenarios that can be obtained based on the expert knowledge of the system. In this work, using historical data, multiple scenarios are identified by using the definition of four power transfers. When these scenarios are defined, a parameterized modal analysis should be performed in order to explore the state space for identifying critical operating points. It is recommendable to initiate the search around particular operating points such as the hour with the highest penetration of renewable generation or the hour with the most demanding reactive power capabilities—voltage regulation issues. In addition, as a parameter, not only power transfers should be considered, but any other parameter that the system experts may consider relevant, e.g., wind speed or solar irradiance in the case of renewable generation. In addition, when power system stabilizers or other control systems are used to improve power system dynamics, a parameterized modal analysis would also be helpful to tune these controllers by defining control gains as parameters.

All of the important studies presented above require the use of powerful simulation tools, accepted by the industry, and also with the capabilities to program the specific routines of the parameterized modal analysis. The DPL environment is shown to be effective and simple to perform this analysis. As DPL has a very similar syntax to C++, no further explanation or specialized training in DPL programming is required. The scripts and applications presented in this document set the basis for additional development to identify critical scenarios for modal analysis in other real systems.

10.5 Conclusion

This chapter has been focused on power system low-frequency oscillations. This phenomenon is very important especially as this may lead to mechanical failures in generating units or push the system to instability. Contrary to the industry approach of taking the highest loading condition as the worst-case scenario for the appearance of electromechanical oscillations, this chapter proposed a methodology to perform a parameterized modal analysis using DPL. This methodology is applied to the *Northern Chile Interconnected System*. The obtained results revealed that the system in a real operation is likely to have low frequency oscillations, not foreseen when the traditional industry approach is used. The system only requires minor

deviations from the programmed operating points to incur in dangerous low-frequency oscillations. Also, it is found that when the system even operates at a lighter loading, low-frequency oscillations may also occur. In this case, the oscillations are related to reactive power capabilities and the under-excited operation of generating units. Therefore, a proper search for critical condition with regard to low-frequency oscillations must be done and a parameterized modal analysis using DPL is found helpful in this matter. The scripts and applications presented in this chapter set the basis for additional development to identify critical scenarios for modal analysis in other real systems. In addition, this procedure can be also used to tune power system controllers to improve power system dynamic operation, such as power system stabilizers.

Acknowledgments The authors thank Ing. Raul Moreno for his valuable comments. Víctor Hinojosa-Mateus thanks the support of the Chilean National Commission for Scientific and Technological Research (CONICYT) under Grant Fondecyt 1130793 and Basal FB0008.

References

1. Rueda J (2009) Assessment and enhancement of small signal stability of power systems considering uncertainties. PhD thesis, Universidad Nacional de San Juan, Argentina
2. Andersson G et al (2005) Causes of the 2003 major grid blackouts in North America and Europe, and recommended means to improve system dynamic performance. *IEEE Trans Power Syst* 20(4):1922–1928
3. Romero JJ (2012) Blackouts illuminate India's power problems. *IEEE Spectr* 49(10):11–12
4. Ning J, Pan X, Venkatasubramanian V (2014) Oscillation modal analysis from ambient synchrophasor data using distributed frequency domain optimization. *IEEE Trans Power Syst* 29:349–358
5. Chen L, Min Y, Chen Y, Hu W (2014) Evaluation of generator damping using oscillation energy dissipation and the connection with modal analysis. *IEEE Trans Power Syst* 29:1393–1402
6. Kundur P (1994) Power system stability and control. McGraw-Hill, New York
7. Sauer PW, Pai MA (1998) Power systems dynamics and stability. Prentice Hall, Upper Saddle River, NJ
8. Machowski J, Bialek JW, Bumby JR (2008) Power systems dynamics, stability and control. Wiley Ltd, Chichester
9. Chen C-T (1998) Linear system theory and design, 3rd edn. Oxford University Press, Inc., New York
10. Perez-Arriaga JI (1981) Selective modal analysis with applications to electric power systems. PhD thesis, Institute of Technology, Massachusetts
11. Pai MA, Sen Gupta DP, Padiyar KR (2004) Small signal analysis of power systems. Alpha Science International, Ltd
12. Seydel R (1994) Practical bifurcation and stability analysis: from equilibrium to chaos, 2nd edn. Springer-Verlag, New York
13. Dobson I, Lu L (1992) Voltage collapse precipitated by the immediate change in stability when generator reactive power limits are encountered. *IEEE Trans Circuits Syst I, Fundam Theory Appl* 39(9):762–766
14. Pulgar-Painemal H, Galvez-Cubillos R (2013) Limit-induced bifurcation by wind farm voltage supervisory control. *Electr Power Syst Res* 103:122–128

15. Factory DP (2010) PowerFactory user's manual. DIgSILENT GmbH Version 14.0
16. CDEC-SING (2014) Center of economic load dispatch of the Northern Chile Interconnected System. [Website] www.cdec-sing.cl. Accessed in March 2014
17. Pizarro-Gálvez S (2014) Análisis modal del Sistema Interconectado del Norte Grande ante escenarios de oferta y demanda, Universidad Técnica Federico Santa María, thesis, Professional Degree in Electrical Engineering [online]. Available <http://www.bib.utfsm.cl/>

Chapter 11

Probabilistic Approach for Risk Evaluation of Oscillatory Stability in Power Systems

José Luis Rueda, Jaime C. Cepeda, István Erlich, Abdul W. Korai
and Francisco M. Gonzalez-Longatt

Abstract The use of probabilistic framework is of great importance for the development of comprehensive approaches, which are suitable for coping with increasing uncertainties in power system operation and planning. While huge effort has been put in the past into the conception of probabilistic methods to deal with stochastic load flow calculation, there is an increasing interest on the development of new approaches to ascertain the implications of changing operating conditions in terms of power system dynamic performance. Among the main concerns on this regard is the determination of the degree of exposure to poorly damped low-frequency oscillations (LFOs), which occur typically in the range of 0.1–1.0 Hz. This chapter concerns the implementation of a Monte Carlo (MC)-based approach

Electronic supplementary material The online version of this chapter (doi:10.1007/978-3-319-12958-7_11) contains supplementary material, which is available to authorized users.

J.L. Rueda (✉)

Intelligent Electrical Power Systems, Department of Electrical Sustainable Energy,
Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands
e-mail: j.l.ruedatorres@tudelft.nl

J.C. Cepeda

Research and Development Department, Corporación Centro Nacional de Control
de Energía – CENACE, Av. Atacazo and Panamericana Sur km 0, Quito, Ecuador
e-mail: cepedajaime@ieee.org

I. Erlich

Institute of Electrical Power Systems, Department of Electrical Engineering and Information
Technologies, University Duisburg-Essen, Bismarckstr. 81, 47057 Duisburg, Germany
e-mail: istvan.erlich@uni-due.de

A.W. Korai

Institute of Electrical Power Systems, University Duisburg-Essen, Bismarckstr. 81,
47057 Duisburg, Germany
e-mail: abdul.korai@uni-due.de

F.M. Gonzalez-Longatt

Electrical Power Systems, School of Electronic, Electrical and Systems Engineering,
Loughborough University, Loughborough LE11 3TU, UK
e-mail: fglongatt@fglongatt.org

for evaluation of oscillatory instability risk by using the functionalities for modelling and programming of DIgSILENT PowerFactory. Particularly, the DIgSILENT programming language (DPL) is used to structure the steps of the MC repetitive procedure, namely sampling of uncertain input variables, automated scenario generation, and storage of eigenanalysis outcomes. The chapter also illustrates the implementation of the PST 16 benchmark system, which has a relative large size, and is appropriated to study different kinds of stability problems, especially LFOs. Based on characteristic parameters of the European power systems, different built-in models available in DIgSILENT are used to model the system components. Numerical experiments performed on this system support the relevance of the MC-based approach.

Keywords Eigenanalysis • Monte Carlo method • Power system dynamic performance • Risk evaluation • Small-signal stability

11.1 Introduction

Because of the changing ways, power systems are being operated in the power industry deregulated environment, evaluating the system performance problem involves much greater uncertainties related to several factors, such as, for instance, fluctuating behaviour of renewable energy sources and greater interaction between the market agents [1]. Within this context, there is also a greater probability of occurrence of system disturbances, which might entail a higher risk of instability threats [2]. Among them are the low-frequency oscillations (LFOs), whose damping degradation can lower the operational efficiency and occasionally led to unstable system operation with major consequences including grid break-ups and widespread blackouts [3]. Probabilistic-based analysis framework is therefore of great importance to properly ascertain the risk of oscillatory instability associated to uncertainties in system operation and planning [4].

Broadly speaking, the existing state-of-art approaches for probabilistic analysis of power system oscillatory performance (i.e. small-signal stability) can be categorized into analytical and simulation type methods. To infer the probabilistic distribution functions (PDFs) of system eigenvalues, the analytical approaches usually assume that their variability can be described by simplified relationships, which are function of some input uncertain parameters [5]. Nevertheless, despite of the reduced computing effort, these approaches might fail to provide reliable estimation of the statistical attributes of the system eigenvalues (i.e. accurate depictions of the true modal variation) [1]. By contrast, the simulation approaches do not resort to significant simplifications and perform the estimation of the PDFs by repetitively evaluating the system performance under varying randomly sampled operating conditions [4]. The computing effort (i.e. large number of numerical simulations) that is spent when running these tools has been questioned in the past as the main

drawback. Notwithstanding, considerable reduction of computing time can be achieved if suitable tools for fast eigenvalue computation of large-sized systems are used [6]. Moreover, higher reduction can be further obtained if the task is distributed by using modern supercomputers or cloud computing along with suitable resource allocation techniques [7].

The work presented in this chapter bases on previous experience of the authors with the development and application of Monte Carlo (MC)-based framework for probabilistic evaluation of the instability risk associated to power system oscillatory performance. As it is well known, this approach belongs to the group of simulation type methods, and its most attractive features resides in the conceptual simplicity, which is crucial for easy implementation in different simulation software packages as well as for application in different systems. Simply put, the MC-based procedure consists in repetitively sampling the uncertain input variables from their PDFs to automatically generate different scenarios, which is followed by evaluation of system oscillatory stability via eigenanalysis and storage of eigenvalue-related information [8].

Section 11.2 provides a brief review on foundations behind the MC-based approach. In Sect. 11.3, the implementation of the approach by using DlgSILENT programming language (DPL) is described, highlighting relevant aspects concerning sampling of uncertain input variables, automated scenario generation, and storage of eigenanalysis outcomes. Section 11.4 outlines the modelling and implementation of the PST 16 benchmark system by using built-in models available in DlgSILENT PowerFactory. Numerical tests are analysed and discussed in Sect. 11.5. Finally, concluding remarks are summarized in Sect. 11.6.

11.2 The MC-Based Procedure

The framework of the MC-based approach for probabilistic evaluation of power system oscillatory instability risk is schematically illustrated in Fig. 11.1. The procedure starts by drawing random samples of continuous (i.e. nodal demands) and discrete (i.e. system component outage occurrence) input variables according to their PDF models. Each trial input vector defines an operating state which is determined by optimal power flow calculation, which accounts for settled utility's operating policy to balance power. Next, linearization-based eigenanalysis is performed for all viable operating states (which strictly fulfil operational requirements) as well as for operating states characterized by allowable overloading percentage of transmission network elements and low-voltage profiles.

For every operating state, the information concerning the global least damped oscillatory mode (e.g. frequency and damping ratio) is stored until a predefined number of simulations (i.e. repetitions) is reached. The global least damped oscillatory mode (ζ_{sys}) is determined throughout every sampled operating state by using (11.1).

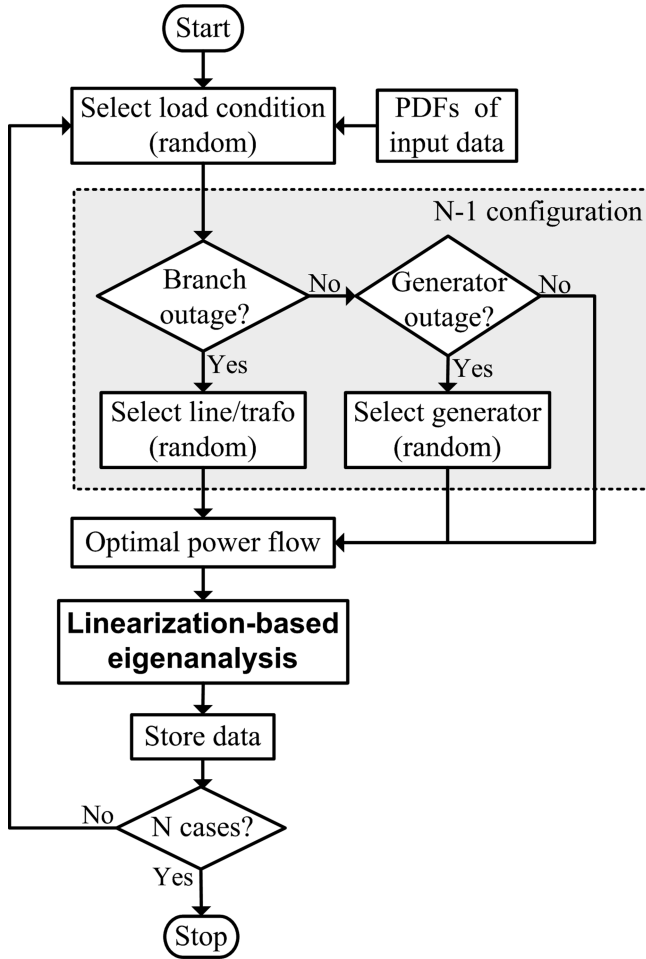


Fig. 11.1 MC-based approach for oscillatory instability risk assessment (reproduced from Rueda and Erlich [8])

$$\zeta_{\text{sys}} = \min_{p=1 \dots nm} (\zeta_p) \quad (11.1)$$

where ζ denotes damping ratio and nm is the number of critical oscillatory modes.

The PDF of ζ_{sys} can be determined by fitting the collected data to a given PDF model. As it is shown in [9], a single PDF model might not entail a good fit to explain the variation of ζ_{sys} . Thus, it is recommended to use a Gaussian mixture model (GMM) [10], whose parameters can be estimated by using the *gmdistribution.fit* function in Matlab environment, or alternatively, the identification procedure introduced in [11]. Based on the resultant PDF and a predefined threshold for

system damping (e.g. $\zeta_{th} = 10\%$), an index that quantifies the global risk of oscillatory instability (ROI) can be derived as follows:

$$ROI = P(\zeta_{sys} < \zeta_{th}) = \int_{\zeta_{lmost}}^{\zeta_{th}} f(\zeta_{sys}) d\zeta_{sys}$$

(11.2)

where $P(.)$ denotes conditional probability, $f(\zeta_{sys})$ is the PDF of ζ_{sys} , and ζ_{lmost} is the lowest value of ζ_{sys} among all sampled operating states.

Interested readers can find a thorough review on modelling of the PDF of input data in [12, 13] and detailed background on probabilistic evaluation of power system small-signal stability in [1].

11.3 Implementation in DPL Language

By using DPL, it is possible to perform all actions required in the flowchart shown in Fig. 11.1, e.g. adjusting or changing network element parameters during the simulation, performing load flow/eigenvalue calculation, as well as collecting and saving the results for further analysis. It is recommended to import the project file *rueda01.pfd* to easily follow the explanations given in remaining of this subsection. Figure 11.2 shows the *basic options* of the main command window of the DPL

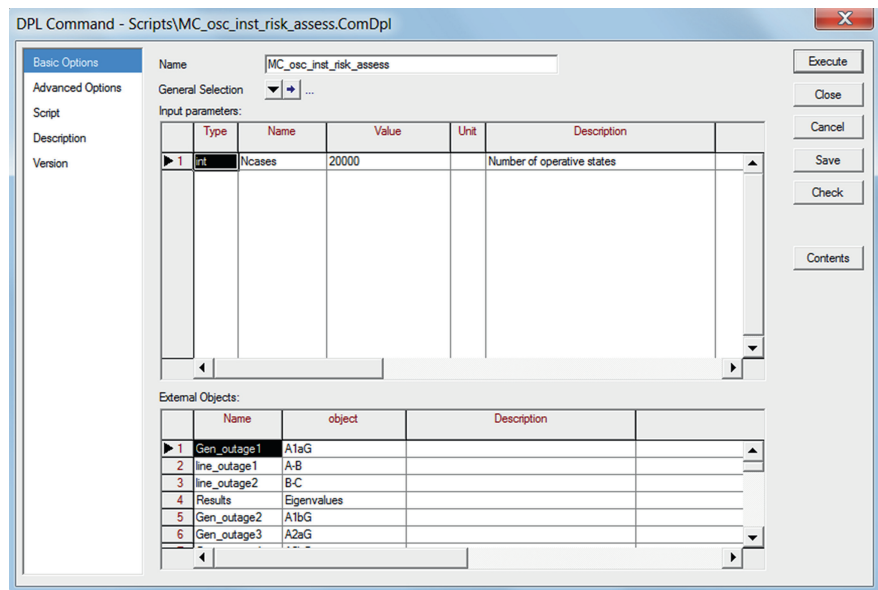


Fig. 11.2 DPL command window

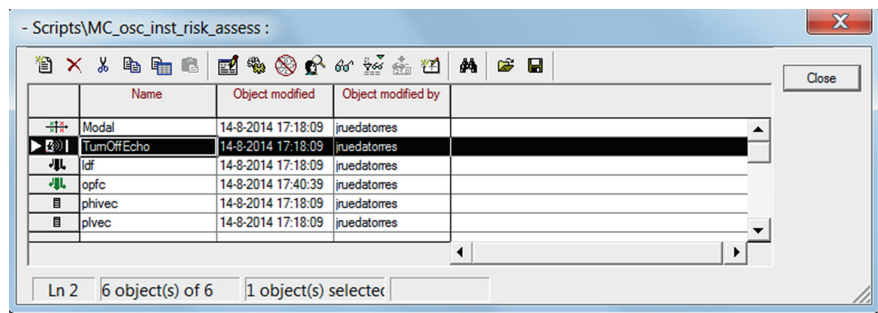


Fig. 11.3 Objects used in the script for internal calculations

script named *MC_osc_inst_risk_assess.ComDpl*. As shown in the figure, the user has the possibility of modifying only one input in the section *input parameters* of the DPL in order to define the total number of MC simulations (i.e. operating conditions) to be carried out.

The *general selection* collects all loads in a container called *set*. The script will then select loads from the container *set* one by one and perform the random variation of demand of active and reactive power as it will be shown afterwards in this subsection. For the test system that will be presented later in Sect. 11.4, the script also calls external objects such as synchronous generators (*ElmSym*), two long tie-lines (*ElmLne*), which will be used in the script to perform random outage of system components, as well as results (*ElmRes*), from which the outcomes of eigenvalue calculation will be read and collected. These objects are defined in the interface section *external objects*.

As shown in Fig. 11.3, by clicking on *contents* in the main command window, the user will find different objects which are launched internally by some commands of the script to perform specific calculations. The object named *ldf* (*ComLdf*) is used to perform load flow calculation, whereas *opf* (*ComOpf*) and *modal* (*ComMod*) are used to perform optimal power flow and eigenvalue calculations, respectively. The vector objects *plvec* and *phivec* (*IntVec*) are used to temporarily store the values of nodal active power demand and power factor, from which new values (and scenarios) will be created. Finally, the object *TurnOffEcho* (*ComEcho*) allows configuring the display messages to show errors and warnings.

Figure 11.4 illustrates the configuration of the *opf* object, which is used to perform the dispatch of generators for a given sampled random loading condition and topology.

As shown in the figure, the objective of the dispatch is minimization of costs, and it is solved by using linear programming and by accounting technical constraints associated to generators and branches of the transmission network. The initialization, advanced options, and iteration control use the default settings. Also, the object *ldf* is configured by using default settings (Fig. 11.5) whereas the object

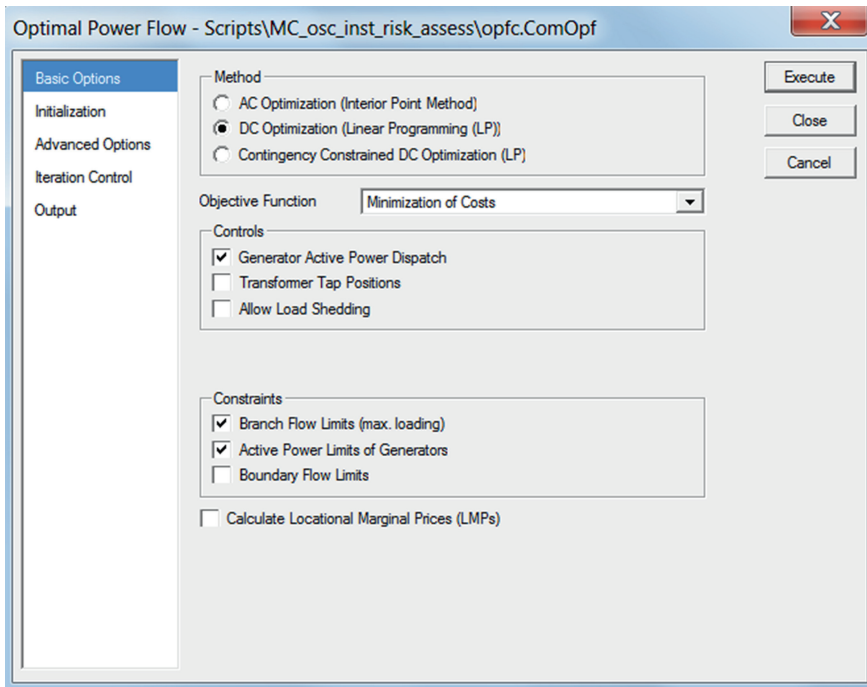


Fig. 11.4 Configuration of OPF object

modal is configured to perform full eigenvalue calculation by using the QR method (Fig. 11.6). Due to the size of the system used for numerical tests in this chapter (220 state variables when the system is modelled without damping controllers and 268 state variables when they are included), the use of this method is preferred. Nevertheless, selective modal analysis (i.e. Arnoldi method) should be used for large power systems (e.g. those having more than 800 state variables) to perform eigenvalue calculation in lesser computing time.

11.3.1 DPL Script

The structure of DPL script follows the iterative procedure shown in Fig. 11.1. Firstly, the script starts with the creation of an output file, which is named as *peigen.csv*, where the results of the MC simulations will be stored. The command `fopen` is used for this purpose:

```
fopen('D:\Chapter11\peigen.csv','w',1);
```

Fig. 11.5 Configuration of load flow calculation object

Next, the vector objects *plvec* and *phivec* are resized according to the number of nodal loads (*ElmLod* objects) in the system as follows:

```
s_load = AllRelevant('*.ElmLod');
k=0;
for (oload=s_load.First(); oload; oload=s_load.Next()){
    k=1+k;
}
plvec.Resize(k);
phivec.Resize(k);
```

Each element of the each vector is filled with the actual values of nodal load active power demand and power factor, respectively. As shown in the following piece of code, the parameters *plini* and *qlini* of each *ElmLod* object are read, so each recorded value of load active power demand in *plvec* corresponds with *plini*, whereas each recorded value of power factor in *phivec* is a function of both *plini* and *qlini*. It is worth pointing out that the values of active power demand and power

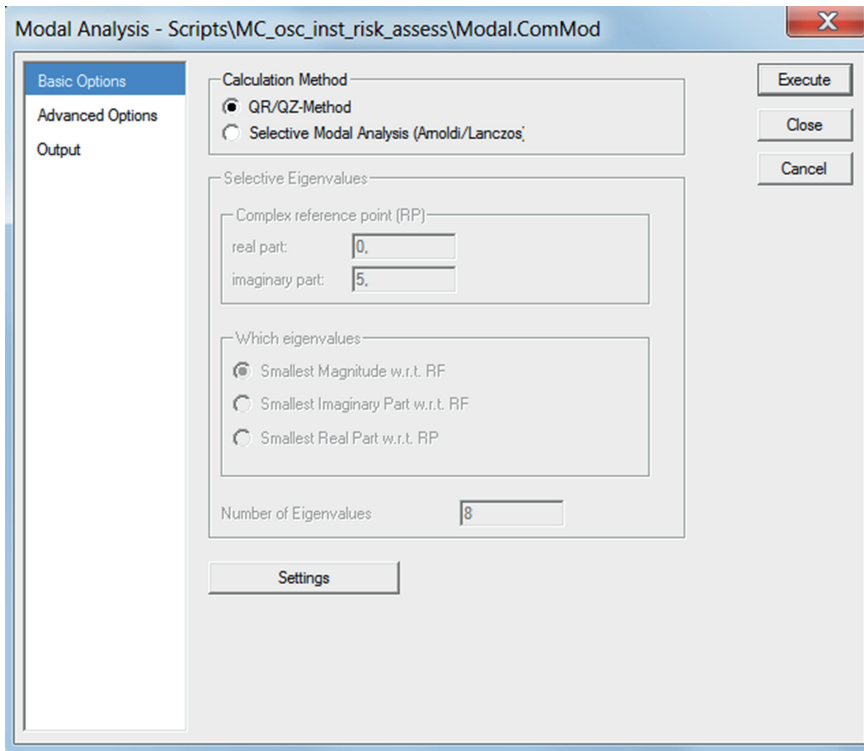


Fig. 11.6 Configuration of eigenvalue calculation object

factor, which are stored in `plvec` and `phivec`, correspond to a highly loaded operating condition of the test system used in this chapter. From these reference values, random values of nodal active and reactive power demand will be generated in every MC simulation.

```
k=0;
for (oload=s_load.First();oload;oload=s_load.Next()){
    k=1+k;
    pold=oload:plini;
    qold=oload:qlini;
    sold=sqrt(sqr(pold)+sqr(qold));
    phiold = pold/sold;
    plvec.Set(k,pold);
    phivec.Set(k,phiold);
}
```

Thereafter, the MC procedure starts by performing the following loop

```
i=0;
for (cases=1;cases<=Ncases;cases+=1){
    i=1+i;
    random scenario generation
    optimal power flow calculation
    eigenvalue calculation
    storage of data
}
```

At every MC simulation, an operating condition (scenario) is sampled by firstly drawing random values of nodal active power demands based on Gaussian PDFs. As shown in the code below, the mean and standard deviation of each Gaussian model are derived from the elements of *plvec*. Note that a random number is uniformly sampled between 0.4 and 1, and it is used to scale each value extracted from *plvec* in order to account for different loading conditions in the system. Also, note that the standard deviation is defined as 5 % of the sampled active power demand. For sake of simplicity, it is assumed that the power factor is keep almost constant. Thus, the value extracted from *phivec* is used to obtain the nodal reactive power demand from the sampled active power demand.

```
sflv = fRand (0,0.4,1);
k=0;
for (oload=s_load.First();oload;oload=s_load.Next()){
    k=1+k;
    pold=plvec.Get(k);
    phiold = phivec.Get(k);
    stdok = pold*sflv*0.05;
    randomp=fRand(1,pold*sflv,stdok);
    oload:plini=randomp;
    qnew=randomp*tan(acos(phiold));
    oload:qlini=qnew;
}
```

Next, an *if...else* statement is used to define whether an outage will occur as suggested in Fig. 11.1:

```

f = fRand (0,0,3);
if({f>0.and.f<=1}){
    caseid=1;
    do nothing (only random Load changes occur)
}else if ({f>1 .and.f<=2}){
    caseid=2;
    random Line outage
}else {
    caseid=3;
    random generator outage
}

```

For the test system used in this chapter, only outage of one of the two parallel circuits of the tie-lines between areas A–B and B–C is considered. The line circuit outage is randomly selected as follows:

```

gg = Random (0,2);
if ({gg>0 .and.gg<=1}){
    line_outage1:nlnum=1;
    Info('Line BC outage');
}
else {
    line_outage2:nlnum=1;
    Info('Line AB outage');
}

```

Only 14 out of 16 power plants of the test system are considered for occurrence of generator outages, which are performed by decreasing the number of available units at a power plant, which is randomly selected as follows:

```

g = Random (0,14);
if ({g>0 .and.g<=1}){
    Gen_outage1:ngnum=4;
    Info('Gen A1aG outage');
}
else if ({g>1 .and.g<=2}){
    Gen_outage2:ngnum=3;
    Info('Gen A2aG outage');
}
... !Rest of DPL script routines for other generators

```

After creation of a given operating condition, optimal load flow is executed to determine the dispatch the active and reactive powers of system generators considering minimization of their operation costs. For this purpose, the following command is used:

```
valid=opfc.Execute();
```

By using the modal analysis functionality, the system is linearized around the actual operating point, and the system state matrix **A** along with the system eigenvalues is computed. This is implemented as follows:

```
Modal = GetCaseObject('ComLdf');
ModalCal=Modal.Execute();
```

The following commands are used to load the outcomes of eigenvalue calculation from the results file in memory as well as to determine the number of system eigenvalues:

```
LoadResData(Results);
Neig = ResNval(Results,0);
```

Recalling (11.1) and (11.2), it is worth emphasizing that the current application concerns determining the statistics of the system's least damped mode. Thus, the following piece of code is defined in the script in order to screen the parameters associated to the least global damped mode from modal analysis results. Among the mode's parameters are real part, imaginary part, frequency, and damping ratio. The values 25 and 0.63 for mode's imaginary part ($\omega = 2\pi f$) define thresholds for filtering those modes between $f = 0.03$ and 4 Hz, respectively. It is worth clarifying that these values can be set to different values depending on the frequency ranges of the LFOs of a particular power system.

```
mindamp=1;
for(ele=0;ele<=Neig-1;ele+=1){
    GetResData(eigreal,Results,ele,0);
    GetResData(eigimag,Results,ele,1);
    if({abs(eigimag)<25}.and.{abs(eigimag)>0.63}){
        freq=abs(eigimag)/(2*pi());
        damp=-eigreal/sqrt(sqr(eigreal)+sqr(eigimag));
        mindamp=min(mindamp,damp);
        if(mindamp=damp){
            imag=eigimag;
            real=eigreal;
            minfreq=freq;}
    }
```


The information associated to the least global damped mode is then stored in the *peigen.csv* file:

```
fprintf(1, '%.6f;%.6f;%.6f;%.6f;%g', real, imag, mindamp, minfreq, caseid);
```

Afterwards, the generator/line that was taken out of service is put back in service for the next iteration in order to avoid the occurrence of simultaneous outages. Finally, the *peigen.csv* file is closed:

```
fclose(1);
```

It is important to note here that the contingencies are created in the script to exclusively apply N-1 criterion. Though being out of the scope of the present application, the sampling of N-2 event can easily be incorporated into the DPL script. It is worth emphasizing that the whole procedure is carried out repetitively for the number of samples (operating states) defined in the main DPL command window.

11.4 The PST 16 Benchmark System

The PST 16-machine test system is shown in Fig. 11.7. It consists of three strongly meshed areas, 66 buses, 16 generators, 28 transformers, and 51 transmission lines. The transmission lines interconnecting the areas are weak because the length of the lines is about 290 km. Thus, this system is useful to study different kinds of stability problems, especially low-frequency oscillations, and has been developed based on characteristic parameters of the European interconnected power system [6]. The system generating park includes hydro, nuclear, and thermal power plants. The total system-wide installed capacity is about 15.93 GW, and the total demand in peak period is 15.57 MW. The transmission network comprises the voltage levels 380, 220, and 110 kV. For hydro, thermal, and nuclear generators, the rated power is 220, 247 and 259 MVA, respectively. The loads are concentrated in area C and power is transferred from area A and B to area C through two long tie-lines.

Modelling of most system components is done by using PowerFactory's built-in models [14], whereas DSL is used to model the controllers attached to each generator and the thyristor-controlled series capacitor (TCSC) located in the tie-line between areas A and C. The implementations were done by using DlgSILENT PowerFactory release 15.0.2.

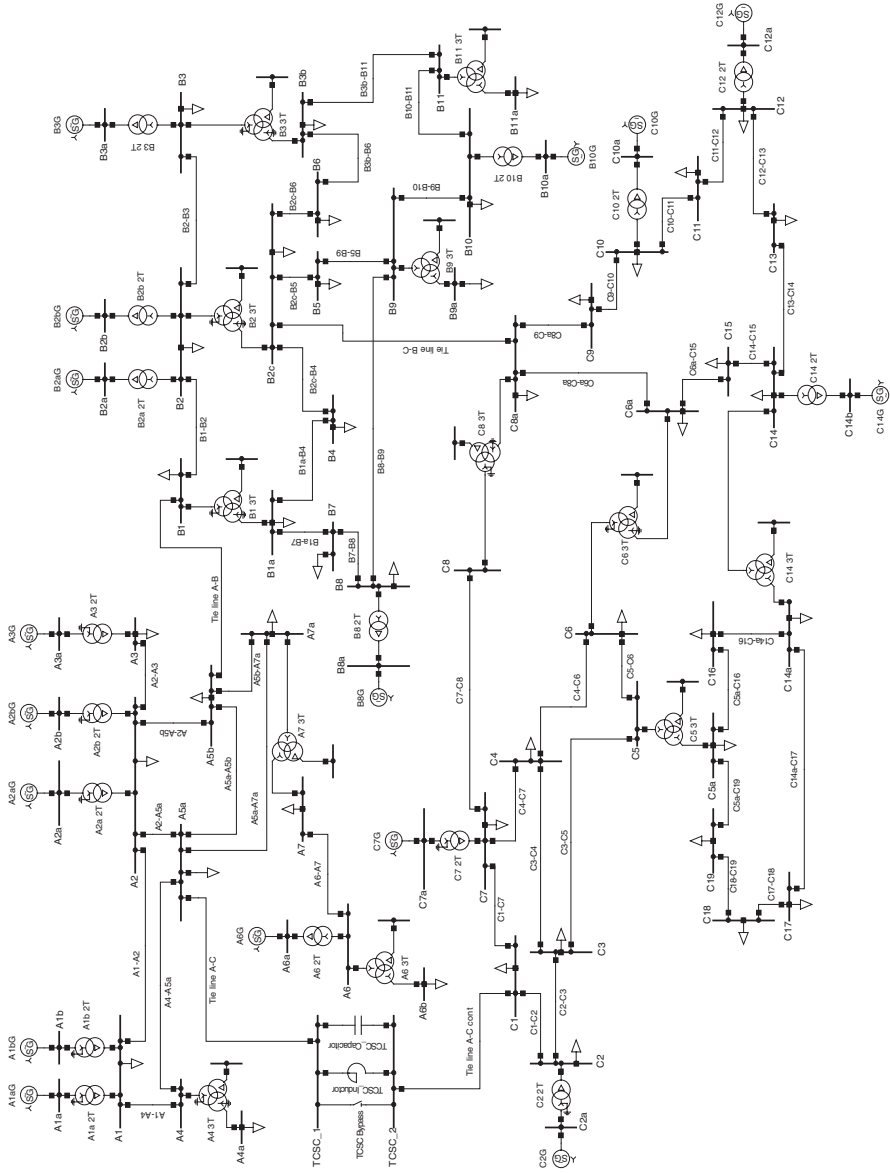


Fig. 11.7 PST 16 benchmark system [6]

11.4.1 Loads and Transmission Network

The general load object *ElmLod* is used to model each load as constant impedance. The detailed information about the general load modelling can be found in the *General Load Technical Reference Manual*.

The elements of the transmission network are represented by lumped parameter (PI) models of the transmission line object *ElmLne* and transformer objects *ElmTr2* and *ElmTr3*. The detailed information about the transmission line/transformer modelling can be found in the *Overhead Line Models/Two-Winding Transformer (3-Phase) Technical Reference Manual*.

As mentioned in the previous subsection, there is a TSC placed in the line interconnecting areas A and C, whose model is described in Chap. 17. The model is implemented by combining a reactor (*ElmSind*) connected in parallel with a series capacitor (*ElmScap*). It is worth indicating that the capacitor is not controlled and its reactance remains constant, while the effective reactance of the TCSC is changed by controlling the reactance of the reactor. More details about modelling of this FACT device can be found in [15, 16].

11.4.2 Modelling of Generators

The object *ElmSym* is used to model each synchronous generator based on the fifth-order model. Detailed information about this model can be found in the *Synchronous Machine Technical Reference manual*. The excitation system is modelled by using the IEEE type 1 model available in DIGSILENT PowerFactory, whereas HYTGOV1 (*gov_HYGOV.BlkDef*) and TGOV1 (*gov_TGOV1.BlkDef*) models are used to represent the hydro and steam turbine governors, respectively. Moreover, a speed input power system stabilizer (PSS) is also attached to each generator. The model of PSS defined in Chap. 12 is adopted, which basically comprises a washout filter, two lead-lag blocks, gain, and output limiters.

11.5 Simulation Results

Numerical experiments were performed on a PC with Intel® Core™ i7-4600U CPU, 2.70 GHz processing speed, and 8 GB RAM. Overall CPU time for MC-based probabilistic eigenanalysis with 10,000 trials was 30.79 min. This computing time is acceptable considering that the studied system has a relative small size compared to real interconnected power systems and that the application of the proposed approach is to be performed offline. The effort could be greater if the approach is applied to large size systems. It is worth clarifying that robust and coordinated tuning of PSSs is beyond the scope of this chapter. Thus, for sake of simplicity and ease of implementation, the parameters of the PSSs at synchronous generators were determined based on the optimization approach presented in Chap. 12 by considering a worst case scenario.

As it is well known, there is a high possibility of occurrence of poorly damped LFOs when the power systems are operating near their technical limits, specially under highly loaded conditions with heavy power transfers [17]. Thus, in this case

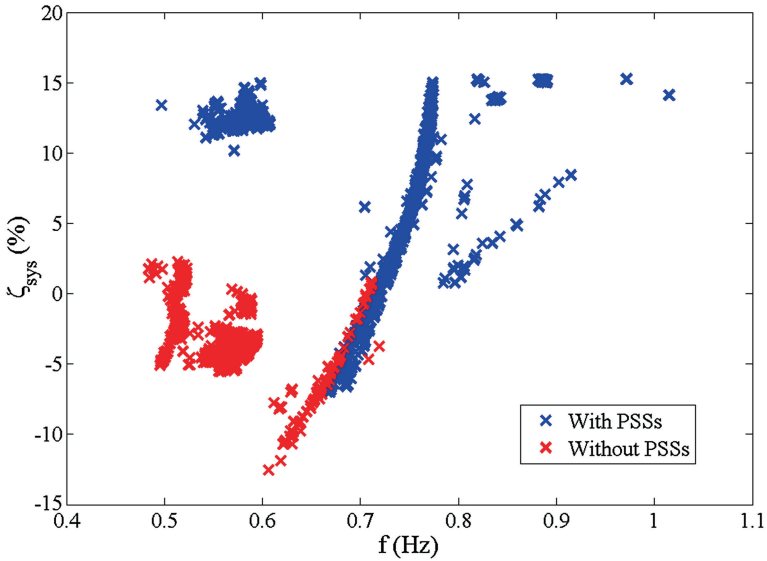


Fig. 11.8 Frequency and damping of the least damped oscillatory mode

study, the uncertainty concerning nodal load variations is modelled by using normal PDFs, whose samples emulate variations around highly loaded conditions. As mentioned previously, the scaled values of active power given for each load in the project file *rueda01.pfd* are used as mean values for the normal PDFs, whereas the standard deviation is assumed to be 5 % around the mean value.

For illustrative comparisons, the MC-based approach was performed by considering the system with and without the PSSs. Figure 11.8 depicts the variation of the frequency and damping ratio associated to the global least damped oscillatory mode for both cases. This mode constitutes an inter-area mode with high participation of generators from areas A and B.

When the PSSs are not considered, the frequency of the global least damped oscillatory mode varies throughout the simulated operating conditions between 0.48 and 0.72 Hz, whereas it varies between 0.49 and 1.02 Hz when the PSSs are included. From the figure, it can be noticed that the addition of the PSS helps in improving the global damping of the systems, but it does not suffice to fully eliminate the possibility of having poorly damped LFOs. This finding is further corroborated in Fig. 11.9, which shows the cumulative PDF (CPDF) resulting from the data collected for ζ_{sys} throughout the MC simulations. From the figure, and considering that $\zeta_{\text{th}} = 10\%$, it can be inferred that there is a high risk of having poorly damped oscillations (i.e. $\zeta_{\text{sys}} < 10\%$). The existence of PSSs, which were tuned by considering only a worst case scenario, does not suffice to fully eliminate the instability risk. To overcome this problem, a multi-scenario-coordinated optimal tuning, such as the one proposed in [9], could be used. This issue is however beyond the scope of this chapter.

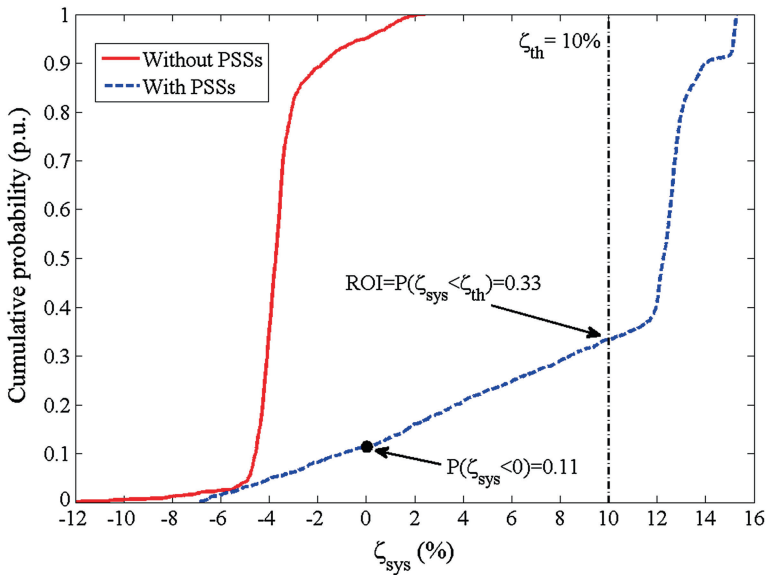


Fig. 11.9 Cumulative PDFs of ζ_{sys} of the PST 16

11.6 Conclusions

In this chapter, the application of MC-based risk assessment for the oscillatory stability was overviewed. The theory behind the MC-based approach and the implementation of the DPL code to carry out MC simulations in PowerFactory was explained. The application of the MC-based approach on the PST 16 benchmark system highlighted the feasibility and suitability of the approach, as a systematic procedure, to estimate the instability risk associated to the global least damped oscillatory mode. The DPL code can be further extended to include a tracking procedure to properly identify and collect the data associated to several critical oscillatory modes throughout the MC simulations. Likewise, the code can be easily adapted to perform the risk assessment by using other power system models. The PST 16 benchmark system can also be extended to perform other kind of stability studies. Future work is currently being pursued to address these issues.

References

1. Rueda JL, Erlich I (2012) Probabilistic framework for risk analysis of power system small-signal stability. In: Proc. Inst. Mech. Eng. [O]: J Risk Reliab—Spec Issue Risk Reliab Model Energy Syst 226(1):118–133
2. Allen A, Keri A, DeGroff A, Kosterev D, Tatro P (2009) Importance of dynamic stability phenomena in power system planning. In: Proceedings of the IEEE power and energy society general meeting, pp 1–6

3. Zhu S, Zhang Y, Le D, Chowdhury AA (2010) Understanding stability problems in actual system—Case studies. In: Proceedings of the IEEE power and energy society general meeting, pp 1–6
4. Rueda JL, Colomé DG, Erlich I (2009) Assessment and enhancement of small signal stability considering uncertainties. *IEEE Trans Power Syst* 24(1):198–207
5. Preece R, Woolley NC, Milanovic JV (2013) The probabilistic collocation method for power-system damping and voltage collapse studies in the presence of uncertainties. *IEEE Trans Power Syst* 28(3):2253–2262
6. Teeuwssen SP (2005) Oscillatory stability assessment of power systems using computational intelligence. PhD Dissertation, University Duisburg-Essen
7. Dong ZY, Zhang P (2010) Emerging techniques in power system analysis. Springer, Berlin
8. Rueda JL, Erlich I (2013) Input/output signal selection for wide-area supplementary damping controllers based on probabilistic eigenanalysis. In: Proc 2nd IFAC Workshop Convergence Inf Technol Control Methods Power Syst, pp 1–6
9. Rueda JL, Cepeda JC, Erlich I (2014) Probabilistic approach for optimal placement and tuning of power system supplementary damping controllers. *IET Gener Transm Distrib*. Available online: 29 May 2014 (doi:10.1049/iet-gtd.2013.0702)
10. Singh R, Pal BC, Jabr RA (2010) Statistical representation of distribution system loads using Gaussian mixture model. *IEEE Trans Power Syst* 25(1):29–37
11. Gonzalez-Longatt F, Rueda JL, Erlich I, Bogdanov D, Villa W (2012) Identification of Gaussian mixture model using mean variance mapping optimization: Venezuelan Case. In: Proceedings of the IEEE PES innovative smart grid technologies (ISGT) Europe conference, pp 1–6
12. Billinton R, Li W (1994) Reliability assessment of electric power systems using Monte Carlo methods. Plenum Press, New York
13. Papaefthymiou G (2007) Integration of stochastic generation in power systems. PhD Dissertation, Delft University of Technology
14. DIgSILENT PowerFactory User Manual (2013) DIgSILENT GmbH. Gomaringen, Germany
15. Cai LJ, Erlich I (2003) Simultaneous coordinated tuning of PSS and FACTS controller for damping power system oscillations in multi-machine systems. *IEEE Power Technol* 2:6
16. Milano F (2008) Documentation of power system analysis toolbox (PSAT) 2.0.0
17. Rueda JL (2009) Assessment and enhancement of small signal stability of power systems considering uncertainties. PhD dissertation, National University of San Juan, Argentine

Chapter 12

Mean–Variance Mapping Optimization Algorithm for Power System Applications in DIgSILENT PowerFactory

**Jaime C. Cepeda, José Luis Rueda, István Erlich, Abdul W. Korai
and Francisco M. Gonzalez-Longatt**

Abstract The development and application of heuristic optimization algorithms have gained a renewed interest due to the limitations of classical optimization tools for tackling several hard-to-solve problems in different engineering fields. Due to the complex nature of power system dynamics, electrical engineering optimization problems usually present a discontinuous multimodal and non-convex landscape that necessarily has to be handled by heuristic optimization algorithms. While most of the pioneer heuristic optimization approaches, such as *genetic algorithms*, *particle swarm optimization*, and *differential evolution*, are undergoing different types of modifications and extensions in order to improve their performance, great focus is also being put into the development of new approaches aiming at conceptual

Electronic supplementary material The online version of this chapter (doi:10.1007/978-3-319-12958-7_12) contains supplementary material, which is available to authorized users.

J.C. Cepeda
Corporación Centro Nacional de Control de Energía—CENACE,
Av. Atacazo and Panamericana Sur km 0, Quito, Ecuador
e-mail: cepedajaime@ieee.org

J.L. Rueda
Department of Electrical Sustainable Energy, Delft University of Technology,
Mekelweg 4, 2628 CD Delft, The Netherlands
e-mail: J.L.RuedaTorres@tudelft.nl

I. Erlich
Department of Electrical Engineering and Information Technologies,
University Duisburg-Essen, Bismarckstr. 81, 47057 Duisburg, Germany
e-mail: istvan.erlich@uni-due.de

A.W. Korai
Institute of Electrical Power Systems, University Duisburg-Essen, Bismarckstr. 81,
47057 Duisburg, Germany
e-mail: abdul.korai@uni-due.de

F.M. Gonzalez-Longatt (✉)
School of Electronic, Electrical and Systems Engineering, Loughborough University,
Loughborough LE11 3TU, UK
e-mail: fglongatt@fglongatt.org

simplicity, easy adaptability for a variety of optimization-based applications, and outstanding performance. The mean–variance mapping optimization (MVMO) is a recent contribution to the family of evolutionary optimization algorithms. Its novel search mechanism performs within a normalized range of the search space for all optimization variables and follows a single parent–offspring pair approach. Besides, MVMO is characterized by a continuously updated knowledge archive storing the n -best solutions achieved so far, from which a special mapping function, which accounts for the mean and variance of the optimization variables, is applied for mutation operation. In this way, the algorithm proceeds by projecting randomly selected variables onto the corresponding mapping function that guides the solution toward the best set achieved so far. Despite the orientation on the best solution, the algorithm keeps on searching globally. This chapter addresses key aspects concerning the implementation of MVMO by using DigSILENT programming language (DPL). An exemplary application on the coordinated tuning of power system supplementary damping controllers is presented and discussed in order to highlight the feasibility and effectiveness of structuring MVMO-based applications in DigSILENT PowerFactory environment.

Keywords DigSILENT programming language • Damping control • Heuristic optimization • Mean–variance mapping optimization • Power system stability

12.1 Introduction

Many real-world optimization problems are often of nonlinear, non-convex, discontinuous, multimodal (i.e., multiple local optima), and mixed integer nature, which are not to be handled by classical optimization methods [7]. These features motivate the exploration of new solution strategies, for which heuristic optimization framework is an attractive alternative. Heuristic optimization algorithms generally adopt a particular search mechanism to repetitively generate new candidate solutions based on the success of preceding solutions [13]. Some of the operations underlying the search mechanism usually involve several random factors with the aim of encouraging a certain degree of search diversity. Thus, due to their stochastic nature, heuristic optimization algorithms do not strictly guarantee global optimality, but usually provide near-to-optimal or good enough solutions in reasonable time frames [14]. Most state-of-the-art heuristic optimization algorithms possess an open architecture, so they are undergoing further development, especially in recent years, in order to enhance their performance [3]. Moreover, great interest is also being put into the development of new algorithmic procedures, which allow efficiently tackling complex and high-dimensional optimization problems.

Mean–variance mapping optimization (MVMO) is a recently introduced evolutionary optimization algorithm [5]. Its search mechanism performs within a

normalized range of the search space for all optimization variables and adopts a single parent–offspring pair approach. The core idea of MVMO bases on a special mapping function applied for mutating the offspring on the basis of mean and variance of the set comprising the n -best solutions attained so far and saved in a continually updating archive. Thanks to the strategic control of the shape of the mapping function throughout the search process, MVMO achieves a suitable balance between search diversification and intensification, which is translated into fast convergence behavior with minimum risk of premature convergence [9].

Preliminary application of MVMO for tackling different power system optimization problems, such as the optimal dispatch of energy and reserve [2], the solution of the optimal reactive power allocation problem [6], transmission expansion planning [10], and identification of dynamic equivalents [1], has demonstrated its potential, as a general-purpose optimization tool, for achieving satisfactory solutions with reduced computing effort. This motivates the investigation and design of new MVMO-based applications, which are of especial interest for those power engineers working with advanced simulation packages. In view of this, this chapter explores and discusses key aspects concerning the implementation of MVMO into DIgSILENT PowerFactory based on the available resources of DIgSILENT programming language (DPL). Except for function evaluation, the calculation stages involved in MVMO's search procedure, i.e., initialization and offspring creation, are structured in a master DPL script, which can be employed, without major modifications, for solving any kind of optimization problem in DIgSILENT PowerFactory environment. The function evaluation is performed in a separate subroutine (slave DPL), which is designed by the user to determine the values of the objective function and constraints defined for a specific optimization problem. The master script subordinates the subroutine for function evaluation in order to determine the fitness of a candidate solution. Besides, the chapter provides an application example for the problem of optimal coordinated tuning of damping controllers [i.e., power system stabilizers (PSS)] by using a benchmark power system.

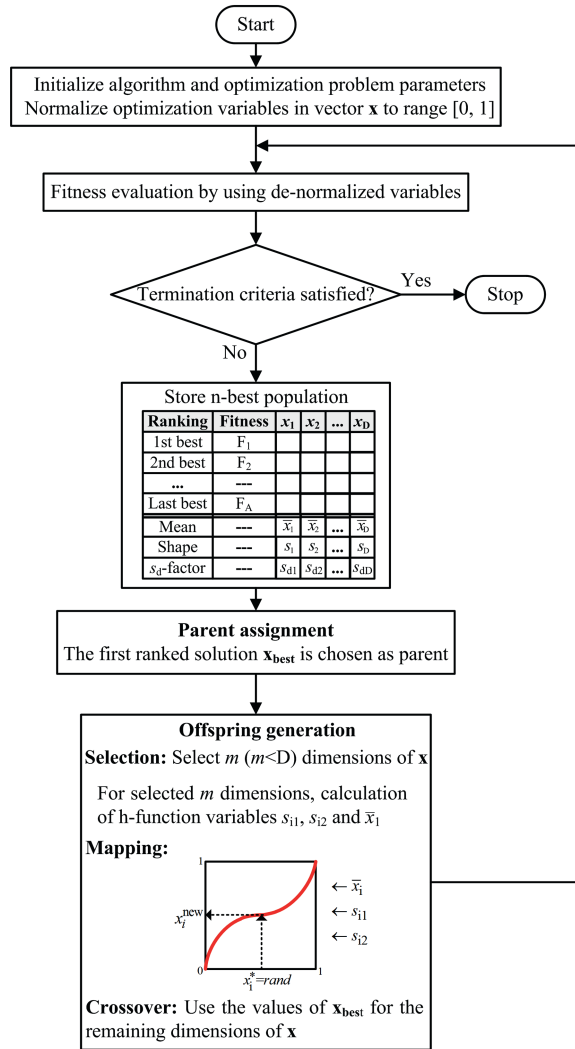
Following this introduction, the outline of the chapter is as follows: Sect. 12.2 presents a review of MVMO theory. Section 12.3 overviews and discusses the implementation of the DPL scripts. Section 12.4 describes the application example and provides a discussion on the experimental results of the test case study. Finally, Sect. 12.5 summarizes the main concluding remarks.

12.2 Rationale Behind MVMO

The general algorithmic procedure of MVMO is illustratively sketched in Fig. 12.1.

MVMO operates on a single solution rather than a set of solutions like other state-of-the-art evolutionary algorithms. It aims at performing prompt and accurate optimization with a minimum amount of objective function evaluations.

Fig. 12.1 Search procedure of MVMO



Broadly speaking, the procedure performs as follows [5]: It starts with an initialization stage, where the algorithm's parameter settings are defined, the initial solution is generated by randomly sampling the decision/optimization variables within their $[\min, \max]$ bounds, and each decision variable is normalized to $[0, 1]$. Next iterative loop is entered, which comprises fitness evaluation of candidate solution, fill/update the solution archive (i.e., inclusion or exclusion of candidate solutions in the archive depending on their fitness), parent assignment (based on global best solution achieved so far), and creation of new candidate solutions

(i.e., application of crossover and mutation operators). The iterative process stops upon fulfillment of a predefined termination criterion. The distinctive features of MVMO are summarized as follows:

- The range of the search space for all optimization variables is restricted to $[0, 1]$. This is a precondition for using the mapping function. Besides, it ensures that the generated offspring is always within the search boundaries. However, fitness evaluation is carried out in the original physical dimension (i.e., transformation from $[0, 1]$ to the original $[\min, \max]$ bounds is performed in this stage).
- A compact and dynamically updated solution archive is used as the knowledge base for guiding the searching direction (i.e., adaptive memory). The n -best individuals that MVMO has found so far are saved in the archive and sorted in a descending order of fitness.
- A single parent–offspring pair concept.
- A novel mapping function that is used for mutating genes in the offspring based on the mean and variance of the solution archive.

12.2.1 Initialization

The parameters used in different operations in MVMO, which are summarized in Table 12.1, are set to predefined values. The table also provides some reference values for these parameters, which were determined from application of MVMO to several optimization test problems. Notwithstanding, it is worth mentioning that tuning of these parameters might be necessary for a given application. This can be done, for instance, by performing sensitivity analysis with respect to a single parameter change within few optimization trials.

Besides, the initial candidate solution is generated by randomly sampling the decision variables within their bounds as follows:

$$x_i^{\text{ini}} = x_i^{\min} + \text{rand}(x_i^{\max} - x_i^{\min}), \quad i = 1 \dots D \quad (12.1)$$

Alternatively, the initial values of the decision variables can be set to specific values, which could be known beforehand. Each decision variable x_i is then normalized to the range $[0, 1]$.

Table 12.1 Parameters used in MVMO algorithm [5]

Name	Symbol	Value
Archive size	A_{sz}	$[2, 5]$
Scaling factor	f_s	$[0.9, 10]$
Asymmetry factor	AF	$[1, 10]$
–	s_d	$[10, 75]$
–	k_d	$0.0505/D + 1$

D number of decision variables

12.2.2 Fitness Evaluation and Termination Criterion

The decision variables are denormalized to their original [min, max] range and subsequently fed into the mathematical model of the optimization problem for evaluation of the equations determining the values of the objective function and constraints (if any). In case of unconstrained optimization problems, the fitness f^* corresponds with the objective function value associated with the candidate solution being evaluated, whereas for constrained problems, it can be calculated by using a constraint handling strategy. For the sake of illustration, f^* can be approximated by using (12.2), which constitutes a static approach [14].

$$f^*(\mathbf{x}) = f(\mathbf{x}) + \sum_{i=1}^{N_{ic}} \gamma_i \cdot \max\{0, g_i(\mathbf{x})\} + \sum_{j=1}^{N_{ec}} \vartheta_j \cdot \max\{0, |h_j(\mathbf{x})| - \varepsilon\} \quad (12.2)$$

where \mathbf{x} is the candidate solution vector, $f(\mathbf{x})$ is the objective function, $g_i(\mathbf{x})$ is the i th inequality constraint, $h_j(\mathbf{x})$ is the j th equality constraint, and ε is a tolerance value. γ_i and ϑ_j constitute penalty coefficients, whereas N_{ic} and N_{ec} denote the number of constraints.

The MVMO search process is terminated upon completion of the termination criterion, which is usually defined as a prespecified number of fitness evaluations. Alternatively, it could be decided to stop the process if there is no improvement of fitness over several successive fitness evaluations.

12.2.3 Solution Archive

The solution archive constitutes the knowledge base of MVMO, whose information is crucial for guiding the searching direction. Basically, the n -best solutions that MVMO has found so far, along with their corresponding fitness, shape, and d factors, are stored in the archive. The archive size is fixed for the entire process and has to be defined by the user beforehand. Experience with application on several optimization benchmarks and power system problems reveals that an archive size of 2–5 is usually sufficient. The archive is filled up progressively over the iteration steps with the solutions sorted in a descending order of fitness, so that the first-ranked solution constitutes the best found so far. Once the archive is filled up, an update is performed only if the fitness of the new generated solution is better than those in the archive. As fitness improves over iterations, the archive members keep changing. Mean and shape variables are calculated after every update of the archive for each optimization variable x_i by using (12.3) and (12.4), respectively.

$$\bar{x}_i = \frac{1}{n} \sum_{j=1}^n x_i(j) \quad (12.3)$$

$$s_i = -\ln(v_i) \cdot f_s \quad (12.4)$$

where the variance v_i is computed as follows:

$$v_i = \frac{1}{n} \sum_{j=1}^n (x_i(j) - \bar{x}_i)^2 \quad (12.5)$$

At the beginning, \bar{x}_i corresponds with the initialized value of x_i , and v_i is set to 1. The shape variable s_i is one of the inputs of the mapping function, which strongly influences the shape its geometric characteristic. This is why the calculation of s_i considers the scaling factor f_s , which allows controlling the form of the mapping function and thus the search process.

12.2.4 Parent Selection, Crossover, and Mutation

The first-ranked position in the solution archive is chosen as the parent $\mathbf{x}^{\text{parent}}$ for the child solution to be created. For the next generation (i.e., iteration), the child solution vector $\mathbf{x}^{\text{child}} = [x_1, x_2, x_3, \dots, x_D]$ is created by combining a subset of D-m directly inherited dimensions from $\mathbf{x}^{\text{parent}}$ (i.e., crossover) and m selected dimensions that undergo mutation operation through mapping function based on the actual values of the parameters \bar{x}_i and s_i . The value of m is problem dependent, whereas the selection of the m variables to be mutated is done by using a random sequential selection strategy, which is illustrated in Fig. 12.2. The new value of each selected dimension x_i is determined by using (12.6).

$$x_i = h_x + (1 - h_1 + h_0) \cdot x_i^* - h_0 \quad (12.6)$$

where x_i^* is a variable varied randomly with uniform distribution within $[0, 1]$ and the h refers to mapping function, which is defined as follows:

$$h(\bar{x}_i, s_{i1}, s_{i2}, x) = \bar{x}_i \cdot (1 - e^{-x \cdot s_{i1}}) + (1 - \bar{x}_i) \cdot e^{-(1-x) \cdot s_{i2}} \quad (12.7)$$

h_x , h_1 , and h_0 are the outputs of the mapping function, which are determined by substitution as follows:

$$h_x = h(x = x_i^*), \quad h_0 = h(x = 0), \quad h_1 = h(x = 1). \quad (12.8)$$

Both input and output of the mapping function are always defined in the interval $[0, 1]$. The shape of the mapping function is determined by the mean \bar{x}_i and the shape factors s_{i1} and s_{i2} . Broadly speaking, the variation of \bar{x}_i entails shifting the mapping curve between the lower and upper bounds of the normalized search range, whereas the variation of s_{i1} and s_{i2} affects its degree of bent, which is crucial to put more emphasis on either exploration (i.e., high bent shape) or exploitation (flattered shape)

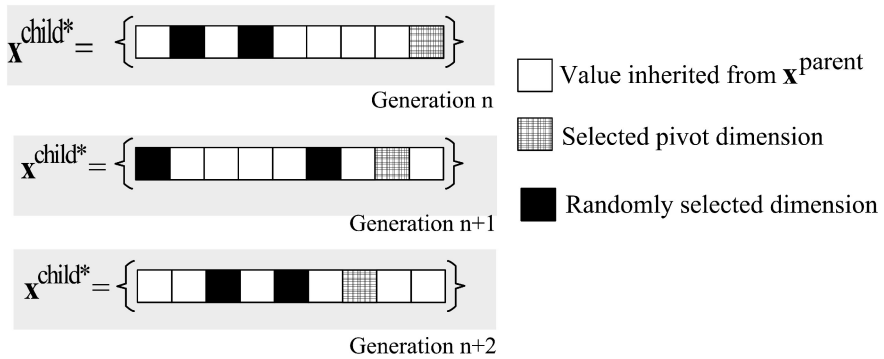


Fig. 12.2 Random sequential selection strategy. Example built considering a 9-dimensional solution vector and $m = 3$ over three successive generations. From one generation to another, the strategy performs by sequentially selecting one (pivot) of the m variables from the last to the first dimension of the parent vector array, whereas the remaining $m - 1$ variables are selected randomly

in different stages of the search process. Therefore, the following strategy is used in MVMO to strategically assign the values of s_{i1} and s_{i2} in order to properly exploit the asymmetric characteristic of the mapping function, which leads to enhanced searching performance with suitable balance between exploration and exploitation:

$$\begin{aligned}
 &\text{if } x_i^{\text{best}} < \bar{x}_i \\
 &\quad s_{i1} = s_i; \quad s_{i2} = s_i \cdot \text{AF} \\
 &\text{elseif } x_i^{\text{best}} > \bar{x}_i \\
 &\quad s_{i2} = s_i; \quad s_{i1} = s_i \cdot \text{AF} \\
 &\text{end.}
 \end{aligned} \tag{12.9}$$

As the optimization progresses, and if m is set to a small value with respect to D , it can prematurely happen that all individuals stored in the solution archive do not differ with respect to particular optimization variables. This is translated into zero or near to zero v_i , cf. (12.5), so the shape variable s_i tends a very high value, cf. (12.4), which is reflected in a flattered mapping curve and entails that the optimization does not progress (i.e., stagnation). Thus, the following strategy is adopted in order to prevent premature stagnation of the search, which considers that the last nonzero v_i is used to compute s_i :

$$\begin{aligned}
 &s_{i1} = s_{i2} = s_i \\
 &\text{if } s_i < s_d \\
 &\quad s_d = s_d \cdot k_d; \quad s_{i1} = s_d \\
 &\text{elseif } s_i > s_d \\
 &\quad s_d = s_d / k_d; \quad s_{i1} = s_d \\
 &\text{end.}
 \end{aligned} \tag{12.10}$$

It is worth pointing out that \bar{x}_i , v_i , s_{i1} , and s_{i2} associated with a candidate solution are not calculated before a certain number of solutions are available in the solution archive. They can be calculated immediately after two solutions have been already stored in the archive. Alternatively, it can decide to do it once the archive is filled up completely. In this stage, the search is performed with $s_{i1} = s_{i2} = 0$, which corresponds with a straight line between zero and one as a shape of the mapping function. In this case, the mean value in this case does not entail any effect on the mapping function.

12.3 MVMO Implementation in DPL Language

DPL offers an interface to develop automatic tasks in DIgSILENT PowerFactory. This functionality allows the creation of new user-defined calculation functions [4].

The main features of DPL comprise a particular programming language, similar to C++ that offers arithmetic and standard function availability, as well as other types of typical programming functions (e.g., logic functions, loops, conditionals), including the handling of vectors and matrices.

The DPL command object (*ComDpl*) constitutes the central element. This object allows the connection of different parameters, variables, or objects to various functions or internal elements in order to give results or change parameters.

The input to the DPL script can be predefined input parameters, single objects from the single line diagram, elements of the database, or a set of objects or elements [4]. This input information can then be evaluated using functions and internal variables inside the script. Also, internal objects can be used and executed (i.e., calculation commands, subscripts also released in DPL, filter sets, etc.).

The DPL script is designed to run a set of operations to communicate itself with the database in order to read and/or change settings, parameters, or results directly in the database objects. This functionality is used to implement the MVMO algorithm in order to carry out a number of possible power system optimization routines that require systematic running of specific power commands (such as load flow, time-domain simulation, modal analysis, among others). This chapter focuses on solving the coordinated tuning of supplementary damping controllers represented by PSS connected to the automatic voltage regulators (AVR) of the generators (*ElmSym*). This type of optimization problem actually requires running the *modal analysis* command (*ComMod*) in order to determine, in each evaluation, the linearization-based eigenanalysis [12] and, after, compute the corresponding objective function.

The following subsections describe, in detail, the implementation of the MVMO into DPL.

12.3.1 Main Script—MVMO Implementation Procedure

After creating the DPL command *MVMO.ComDpl*, the main dialogue (*Input parameters*) is used in order to specify the main parameters of the MVMO that can be modified by the user to control the optimization.

Figure 12.3 presents the *basic options* dialogue of *MVMO.ComDpl*, where *Input parameters* section contains the number of population to be saved in the archive, the number of restrictions (if there exists), the number of function evaluations (defined MVMO stop criterion), the initial scaling factor f_s , and the output counter that defines the interval of evaluations after which the results will be printed in the PowerFactory *Output Window*. The programming variables are included in the DPL programming code (*Script* section), whereas several internal objects, such as vectors (*IntVec*), matrices (*IntMat*), results (*ElmRes*), and the function evaluation subroutine (*Obj_function.ComDpl*), are included in *Contents* section.

The DPL script has been structured in order to incorporate all the stages presented in the MVMO search procedure depicted in Fig. 12.1.

First, all the internal variables have to be declared, as follows:

```
!!! MVMO internal variable declaration
int nparam,ele,dddd,Ddddd,asymetry,Fx_best,no_in,no_inin,ele1;
int scaling1,scaling2,scaling,fitsize,xtfeas,isbetter,ixbetter;
int n_sim,nnn,nnn1,nnp,changed,i_position,njump,Time_o,Time_f,Dpl_time;
double ran_n,x_n,x_den,mslope,Nrandomly,ffx,xtobjective,xtfitness;
double sumfit,vartemp,weight;
double vartemp1,vartemp2,vartemp3,vartemp4,vartemp5,H,H0,H1;
string folder,file1,file2,parametros;
int contador;
set OldMons;
object mon;
object GrB,ViPg,Plot;
```

Since the number of the control variables (i.e., variables to be optimized) depends on the specific optimization problem, defining a generalized *ElmRes* object is not feasible. Thus, it has been decided to save the results in two different text files: *results.txt* for saving the function evaluation convergence and *parameters.txt* for saving the best optimized parameters or variables for each evaluation. Therefore, the full path of a folder where both files will be saved has to be firstly specified at the beginning of the DPL script (after the variable declaration). This folder specification is a mandatory input data to allow the DPL script running.

```
!!! SPECIFICATION OF FOLDER TO SAVE RESULTS
folder=sprintf('%s','C:\jcepeda\MVMO_Chapter');
```

After the full path definition of the folder, the *txt* files are created, as follows.

```
!!! CREATION OF TWO RESULT FILES
file1=sprintf('%s%s',folder,'\results.txt');    ! File for saving convergence results
file2=sprintf('%s%s',folder,'\parameters.txt'); ! File for saving optimized variables
```

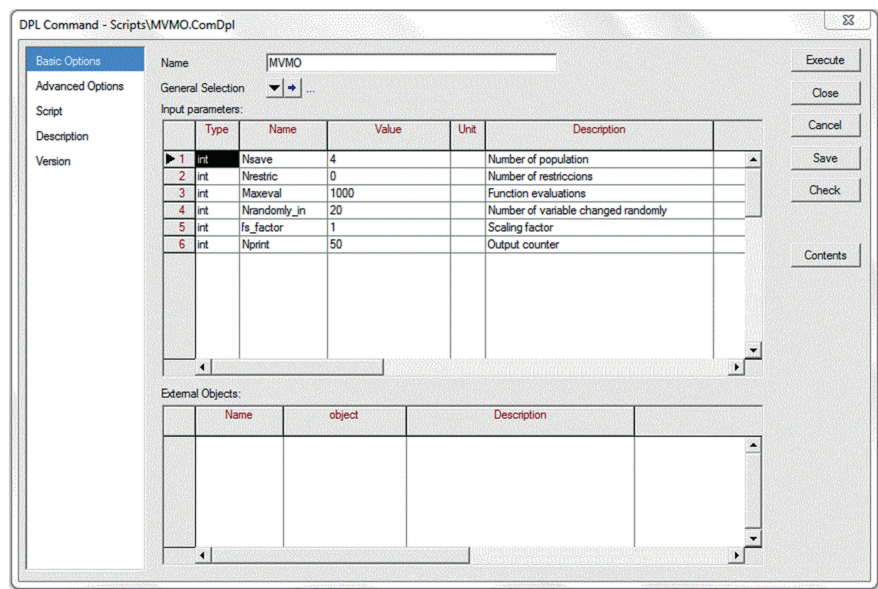



Fig. 12.3 MVMO DPL command basic options

In the procedure of running a heuristic optimization algorithm, knowing the elapsed time of the entire optimization is usually of interest. Then, some command lines have been included for computing the elapsed time as well as to freeze the user interface (to accelerate simulations), as follows.

```
Time_o=GetTime(4);    !Defining the initial processor time
EchoOff();            !Freezing the user-interface

...                  !DPL script routines

EchoOn();             !Re-activating the user interface
Time_f=GetTime(4);    !Defining the final processor time
Dpl_time=Time_f-Time_o; !Computing the elapsed time
printf('%s: %g','Processor Time',Dpl_time); !Printing the elapsed time
```

Although the results of the optimization will be saved in the previously specified *txt* files, the convergence of the objective function will also be saved in the internal results' object *Convergence.ElmRes*. This saving will allow the internal plot of the *Objective Function versus Function evaluations* graphic.

For this purpose, in *Advance Options* section, two *Result parameters* representing the *Objective Function Convergence* (*ffx_conv*) and the *Number of Function evaluations* (*ffx_eval*) have to be defined, as shown in Fig. 12.4.

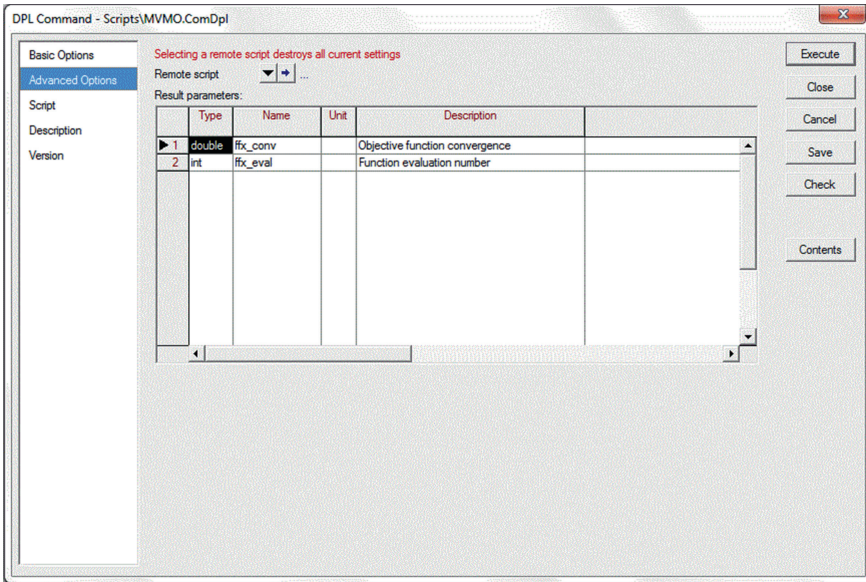


Fig. 12.4 MVMO DPL command advanced options

These two variables have to be created into the *Convergence.ElmRes* object by applying the following code.

```
! Saving convergence results

OldMons = Convergence.GetContents();
mon=OldMons.FirstFilt('Xvar.IntMon');

if(mon = NULL){
    mon = Convergence.CreateObject('IntMon', 'Xvar');
}

if(mon){
    mon.obj_id=this;
    mon.AddVar('b:ffx_eval');
    mon.AddVar('b:ffx_conv');
}
```

After, several subroutines that allow structuring all the stages presented in the MVMO search procedure depicted in Fig. 12.1 have been implemented, as presented in the following lines:

Stage 1: Parameter and memory space definition

The definition of internal parameters is firstly stated in this routine. The objective of this definition is to provide MVMO with the required parameters which are suggested not to be changed by the user.

```
! Parameter and memory space definition
contador=1;
sumfit=0;
dddd=25;
asymetry=2.5;
Fx_best=1e50;
no_in=0;    !Initial number of variables selected for mutation
no_inin=0;
Nrandomly=Nrandomly_in;
```

One of the main aspects of the optimization algorithm is the memory space definition since it gives the appropriate allocation of the solution archive, as well as the space of each one of the matrices associated with the optimization variables, including the search space of each variable, and also other control parameters such as the possible constraints. A total of 21 vectors (*Feas.IntVec*, *Fitness.IntVec*, *Meann.IntVec*, *Mobjective.IntVec*, *V_irand.IntVec*, *V_temp.IntVec*, *considered.IntVec*, *ggx.IntVec*, *id_greater.IntVec*, *idcontrol.IntVec*, *irand.IntVec*, *mean_temp.IntVec*, *v_isfeas.IntVec*, *var_temp.IntVec*, *variance.IntVec*, *vsel.IntVec*, *vv.IntVec*, *x_denorm.IntVec*, *x_denorm_best.IntVec*, *x_norm.IntVec*, *x_norm_best.IntVec*) and 2 matrices (*Paralim.IntMat*, *bests.IntMat*) have been included as internal objects of *MVMO.ElmDpl* in order to accomplish all the required space definition.

From all the above-mentioned vectors and matrices, only *Paralim.IntMat* constitutes a mandatory input data to be specified by the user prior to the running of the optimization task. This matrix contains the [min, max] bounds of each one of the optimization variables. Thus, the search range of each variable has to be defined in this matrix considering that the size of the matrix depends on the number of variables ($2 \times n$, where n is the number of optimization variables). All other vectors or matrix will be adequately resized as space definition requirements depending on the MVMO parameters defined in the basic options as well as the n number of rows of *Paralim.IntMat*.

```

! Memory space definition
nparam=Paralim.NCol(); ! returns the no of columns in matrix
Dddd=0.0505/nparam+1; ! Initial value of alternative shape
bests.Resize(Nsave,nparam); ! Nsave rows (no of population) and n columns
Mobjective.Resize(Nsave); ! vector of Nsave rows (no of population)
Feas.Resize(Nsave); ! Feas resize (no of population)

Fitness.Resize(Nsave); ! for Nsave population size, set Fitness vector to 1e10
for(ele=1;ele<=Nsave;ele+=1){
    Fitness.Set(ele,1e10);
}

considered.Resize(nparam); ! Considered vector with n values from nparam
variance.Resize(nparam); ! variance vector with n values from nparam
Meann.Resize(nparam); ! mean vector with n values from nparam
for(ele=1;ele<=nparam;ele+=1){
    considered.Set(ele,1); ! set value at index i (i,v) to V
    variance.Set(ele,1); ! set considered and variance vector to 1
}

vsel.Resize(Nrandomly); ! Vsel vector Number of variable changed randomly
vv.Resize(Nrandomly); ! vv vector with Number of variable changed randomly
for(ele=1;ele<=Nrandomly;ele+=1){
    vsel.Set(ele,ele);
    vv.Set(ele,ele);
}

v_isfeas.Resize(Nrestric); ! vector v_isfeas equal to number of restrictions
for(ele=1;ele<=Nrestric;ele+=1){
    v_isfeas.Set(ele,ele);
}

if (Nrandomly > nparam) { ! if # var. changed randomly > # parameters to be optimized
    Nrandomly=nparam; !variable changed randomly = Number of parameters to be optimized
}

x_norm.Resize(nparam); ! make vector x_norm equal to nparam
x_denorm.Resize(nparam); ! make vector x_denorm equal to nparam
idcontrol.Resize(nparam); ! make vector idcontrol equal to nparam
mean_temp.Resize(nparam); ! make vector mean_temp equal to nparam
var_temp.Resize(nparam); ! make vector var_temp equal to nparam
V_temp.Resize(nparam); ! make vector V_temp equal to nparam

```

Stage 2: Initialization of parameters

The initial candidate solution is generated by randomly sampling the decision variables within their bounds and then previously normalized to the range [0, 1].

```

! Initialization
for(ele=1;ele<=nparam;ele+=1){ ! for every parameters to be optimized
    ran_n= fRand(0); !generate a random number (uniform distribution)
    x_norm.Set(ele,ran_n); ! set x-norm values to random numbers
    x_norm_best.Set(ele,ran_n); ! set x-norm_best values to random numbers
    x_denorm_best.Set(ele,ran_n); ! set x-denorm_best to random numbers
    Meann.Set(ele,ran_n); ! set Meann to random numbers
}
Meann.Set(ele,ran_n); ! set Meann to random numbers
}

```

Stage 3: Optimization main loop

The main optimization loop presented in Fig. 12.1 is composed of several routines that allow the control of the optimization procedure, the function evaluation execution, the constraint administration, as well as the mutation and offspring generation. The commands involved in this main loop are presented as follows.

```
!Main loop: Optimization, Objective function and Restrictions

for(n_sim=1;n_sim<=Maxeval;n_sim+=1){ ! Running for each iteration to Maxeval

nnn=1000;
nnn1=6000;
nnp=2;

if(n_sim<=nnn){          ! if total runs are equal to 1000, then
Nrandomly=Nrandomly_in; ! make Number of variable changed randomly = Nrandomly_in
}else if({n_sim>nnn}.and.{n_sim<=nnn1}){ ! or if total runs > 1000 & < 6000 then
mslope= (Nrandomly-nnp)/(nnn-nnn1);      ! modify mslope
Nrandomly=round(mslope*(n_sim-nnn)+Nrandomly_in);
}else if(n_sim>nnn1){
Nrandomly=nnp;
}

if (Nrandomly<nnp){
Nrandomly=nnp;
}

for(ele=1;ele<=nparam;ele+=1){ ! for each parameters to be optimized
scaling1=Paralim.Get(2,ele);    ! get max parameters values and store in scaling1
scaling2=Paralim.Get(1,ele);    ! get min parameters values and store in scaling2
scaling=scaling1-scaling2;      ! get the diff between max and min limits of param
x_n=x_norm.Get(ele);           ! get n random values from x_norm in x_n
x_den=x_n*scaling+scaling2;     ! denormalize all values (n)
x_denorm.Set(ele,x_den);       ! set denormalized values x_denorm vector
}

!!! Call Objective function
Obj_function.Execute(ffx);      ! Execute the subroutine of Function Evaluation
xtobjective=ffx;                ! set result from objective fuction

!!! Penalty for poor fitness
fitsize=ggx.Size();            ! set fitsize to equal to vector ggx (penalty values)
for(ele=1;ele<=fitsize;ele+=1){ ! for each constraint
vartemp=ggx.Get(ele);          ! set vartemp equal to vector ggx value
if(vartemp>0){                  ! if vartemp is greater than 0
sumfit+=sqr(vartemp);          ! add square of vartemp (ggx value at that iteration)
}
xtfitness=ffx+1e5*sumfit;      ! if restrictions don't meet, apply the penalty
}
sumfit=0;

!!! Check the feasibility
if(xtfitness<xtobjective){      ! if objective function value is equal to xfitness
xtfeas=1;                      ! make xfeas equal to 1
}
```

```

    }else{
        xtfeas=0;                ! other set it zero
    }

!!! Rule-based elitism
isbetter=0;
vartemp=Feas.Get(1);           ! set vartemp equal to 1st value of Feas
vartemp1=Mobjective.Get(1);    ! set vartemp1 equal to 1st value of Mobjective
if({xtfeas}.and.{vartemp}.and.{xtobjective<vartemp1}){
    isbetter=1;                ! set Isbetter equal to 1
}else if({.not.xtfeas}.and.{.not.vartemp}.and.{xtobjective<vartemp1}){
    isbetter=1;                ! set Isbetter equal to 1
}else if({xtfeas}.and.{.not.vartemp}){
    isbetter=1;                ! set Isbetter equal to 1
}
if(isbetter=1){                ! if isbetter=1 then
    Fx_best=xtobjective;        ! make Fx_best equal to value of objective fucntion
}

!!! Storing the n-best solutions to the archive
no_in+=1;                      ! for each iteration, increase no_in by 1
changed=0;
ixbetter=0;
vartemp=x_norm.Size();        ! set vartemp equal randomly generated variables

!!! Applicable for First iteration only
if(no_in=1){                    ! First time will be true
    for (ele=1;ele<=vartemp;ele+=1){ ! for each n parameters to be optimized
        vartemp1=x_norm.Get(ele);    ! set vartemp1 equal to parameter values
        bests.Set(1,ele,vartemp1);    ! set matrix bests equal to vartemp1
    }
    Mobjective.Set(1,xtobjective);    ! set objective function in Mobjective (1st)
    Fitness.Set(1,xtfitness);         ! set fitness value in Fitness(1st)
    Feas.Set(1,xtfeas);               ! set xtfeas in Feas (1st)
    weight=1/Nsave;                   ! Weight is the mean factor
    no_inin+=1;                       ! increase no_in by 1
}

!!! After first iteration
}else{
    i_position=0;
    for(ele=1;ele<=Nsave;ele+=1){    ! for Nsave populations
        if(changed=1){                ! if changed is 1
            break;                    ! then break the loop
        }
        !!! Rule-based elitism        ! other wise
        ixbetter=0;
        vartemp=Feas.Get(ele);        ! set vartemp equal to value of Feas
        vartemp1=Mobjective.Get(ele); ! set vartemp1 equal to value of Mobjective
        if({xtfeas}.and.{vartemp}.and.{xtobjective<vartemp1}){
            ixbetter=1;
        }else if({.not.xtfeas}.and.{.not.vartemp}.and.{xtobjective<vartemp1}){
            ixbetter=1;
        }else if({xtfeas}.and.{.not.vartemp}){
            ixbetter=1;
        }
        if(ixbetter=1){
            Fx_best=xtobjective;        ! make Fx_best equal to value of objective function
        }
        if(ixbetter=1){
            i_position=ele;            ! change i_position with each iteration
            changed=1;                 ! make changed equal to 1 and
            no_inin+=1;                ! incese no_inin by 1
        }
    }
}
}

```

```

if(changed=1){
if(Nsave>=i_position+1){
for(ele=Nsave;ele>=i_position+1;ele-=1){ ! run for loop from Nsave to 1
vartemp=Mobjective.Get(ele-1); ! set vartemp equal ele-1 value Mobjective
Mobjective.Set(ele,vartemp); ! set Mobjective value equal to vartemp
vartemp=x_norm.Size(); ! make vartemp equal to x_norm (n)
for(ele1=1;ele1<=vartemp;ele1+=1){
vartemp1=bests.Get(ele-1,ele1); ! set vartemp1 with elem. of bests
bests.Set(ele,ele1,vartemp1); ! bests rows are replaced from top
}
vartemp=Fitness.Get(ele-1); ! set vartemp equal to ele-1 row of Fitness
Fitness.Set(ele,vartemp); ! set Fitness ele row with ele-1 row
vartemp=Feas.Get(ele-1); ! same for Feas
Feas.Set(ele,vartemp);
}
}

vartemp=x_norm.Size(); ! make vartemp equal to x_norm size (n)
for (ele=1;ele<=vartemp;ele+=1){
vartemp1=x_norm.Get(ele); ! set vartemp1 with x_norm
bests.Set(i_position,ele,vartemp1); ! set first row with vartemp1 (x_norm)
}
Mobjective.Set(i_position,xtobjective); ! set Mobjective with obj. function
Fitness.Set(i_position,xtfitness); ! same for fitness with xfitness
Feas.Set(i_position,xtfeas); ! same for Feas with feas

!!! Actualize x_norm_best
vartemp=x_norm.Size(); ! make vartemp equal to x_norm size (n)
for(ele=1;ele<=vartemp;ele+=1){
vartemp1=bests.Get(1,ele); ! set vartemp1 with 1st row and n columns of bests
x_norm_best.Set(ele,vartemp1); ! set x_norm_best with vartemp1
}

!!! Calculation of mean and variance
if(no_inin>=Nsave){ ! if no_inin is > Nsave
if(i_position<>1){ ! if i_position is not 1
vartemp=idcontrol.Size(); ! make vartemp equal to idcontrol size
for(ele=1;ele<=vartemp;ele+=1){
idcontrol.Set(ele,1); ! set each row of idcontrol with 1
}
}else{ ! if i_position is 1
vartemp=idcontrol.Size(); ! make vartemp equal to idcontrol size
for(ele=1;ele<=vartemp;ele+=1){
vartemp1=considered.Get(ele); ! make vartemp1 equal to considered
vartemp2=max(0,vartemp1); ! max value between 0 and considered
idcontrol.Set(ele,vartemp1); ! set idcontrol equal to considered
}
}
}

!!! MEAN
for(ele=1;ele<=nparam;ele+=1){
vartemp=0;
for(ele1=1;ele1<=Nsave;ele1+=1){
vartemp1=bests.Get(ele1,ele);
vartemp+=vartemp1; ! add each variable in best
}
vartemp2=weight*vartemp; ! mean formula
mean_temp.Set(ele,vartemp2); ! mean_temp is equal to mean value
}

for(ele=1;ele<=nparam;ele+=1){
vartemp=idcontrol.Get(ele);
if(vartemp=1){
vartemp1=mean_temp.Get(ele);
Meann.Set(ele,vartemp1);
}
}

```

```

    }
}

!! VARIANCE
for(ele=1;ele<=nparam;ele+=1){
    vartemp=0;
    for(ele1=1;ele1<=Nsave;ele1+=1){
        vartemp1=bests.Get(ele1,ele);           ! for each variable in best
        vartemp2=mean_temp.Get(ele);
        vartemp3=weight*sqr(vartemp1-vartemp2); ! variance formula
        vartemp+=vartemp3;
    }
    var_temp.Set(ele,vartemp);
    if(vartemp>0){
        V_temp.Set(ele,1);
    }else{
        V_temp.Set(ele,0);
    }
}
for(ele=1;ele<=nparam;ele+=1){
    vartemp=idcontrol.Get(ele);
    vartemp2=V_temp.Get(ele);
    vartemp3=min(vartemp,vartemp2);
    if(vartemp3=1){
        vartemp1=var_temp.Get(ele);
        variance.Set(ele,vartemp1);
    }
}
}

!! PRINT AND SAVE
vartemp=Mobjective.Get(1);           ! get first value of objective function
vartemp2=Feas.Get(1);
ffx_conv=vartemp;
ffx_eval=n_sim;
Convergence.WriteDraw();             ! Internally save convergence results (for plotting)

fopen(file1,'a',1);                  ! Open file for saving convergence results
fprintf(1,'%g %g %g',n_sim,vartemp,vartemp2); ! Save convergence results
fclose(1);                           ! Close file for saving convergence results

if({n_sim=contador*Nprint}.or.{n_sim=1}){ ! Print convergence in Output Window
    parametros=sprintf('%g %g %g',n_sim,vartemp,vartemp2);
    printf('%s',parametros);
    if(n_sim>1){
        contador+=1;
    }
}

fopen(file2,'a',2);                  ! Open file for saving optimized variables
for(ele=1;ele<=nparam;ele+=1){
    scaling1=Paralim.Get(2,ele);
    scaling2=Paralim.Get(1,ele);
    scaling=scaling1-scaling2;
    x_n=bests.Get(1,ele);
    x_den=x_n*scaling+scaling2; ! denormalize
    x_denorm_best.Set(ele,x_den); ! Save denormalized parameters
    if(ele=1){
        parametros=sprintf('%g',x_den);
    }else{
        parametros=sprintf('%s %g',parametros,x_den);
    }
}
fprintf(2,'%s',parametros);          ! Save optimized variables results
fclose(2);                           ! Close file for saving optimized variables

```



```

!! MUTATION 1
for(ele=1;ele<=nparam;ele+=1){
    vartemp=x_norm_best.Get(ele);
    x_norm.Set(ele,vartemp);      ! Set x_norm with x_norm_best
    considered.Set(ele,0);
}

njump=1;

for(ele=1;ele<=Nrandomly_in;ele+=1){ ! for variables to be mutated
    vartemp=vs1.Get(ele);           ! set vartemp to random values from vs1
equal to
    if(vartemp>nparam){
        id_greater.Set(ele,1);
        vartemp1=vs1.Get(ele);
        vartemp2=vartemp1-nparam;
        vs1.Set(ele,vartemp2);
    }else{
        id_greater.Set(ele,0);      ! set id_greater equal to 0
    }
}

irand.Init(Nrandomly-1);
V_irand.Init(Nrandomly);
vartemp=vs1.Get(1);
V_irand.Set(1,vartemp);
for(ele=2;ele<=Nrandomly;ele+=1){
    vartemp=irand.Get(ele-1);
    V_irand.Set(ele,vartemp);
}

for(ele=1;ele<=Nrandomly-1;ele+=1){
    vartemp=1;
    vartemp4=0;
    while(vartemp){
        vartemp2=fRand(0);
        vartemp3=round(1+(nparam-1)*vartemp2);
        for(ele1=1;ele1<=Nrandomly;ele1+=1){
            vartemp5=V_irand.Get(ele1);
            if(vartemp3=vartemp5){
                vartemp4+=1;
            }
        }
        if(vartemp4=0){
            vartemp=0;
            irand.Set(ele,vartemp3);
            V_irand.Set(ele+1,vartemp3);
        }
        vartemp4=0;
    }
}
for(ele=2;ele<=Nrandomly;ele+=1){
    vartemp=irand.Get(ele-1);
    vs1.Set(ele,vartemp);
}
for(ele=1;ele<=Nrandomly;ele+=1){
    vartemp=vs1.Get(ele);
    considered.Set(vartemp,1);
}
for(ele=1;ele<=Nrandomly;ele+=1){
    vartemp=vs1.Get(ele);
    vartemp+=njump;
    vs1.Set(ele,vartemp);
}

```

```

!! MUTATION 2
for(ele=1;ele<=nparam;ele+=1){
    vartemp=considered.Get(ele);
    if(vartemp){
        vartemp1=fRand(0);
        x_norm.Set(ele,vartemp1);
    }
}

for(ele=1;ele<=nparam;ele+=1){
    vartemp=considered.Get(ele);
    if(vartemp){
        vartemp3=variance.Get(ele);
        vartemp1=-ln(vartemp3)*fs_factor; !!!ss1
        vartemp2=vartemp1; !!!ss2
        vartemp4=x_norm_best.Get(ele);
        vartemp5=Meann.Get(ele);
        if(vartemp4<vartemp5){
            vartemp2=asymetry*vartemp2;
        }else if(vartemp4>vartemp5){
            vartemp1=asymetry*vartemp1;
        }else{
            if(vartemp2>=dddd){
                dddd=dddd*Ddddd;
            }else{
                dddd=dddd/Ddddd;
            }
            vartemp1=dddd;
        }
        !! Function mutation
        vartemp3=Meann.Get(ele);
        vartemp4=x_norm.Get(ele);
        ! Mapping Functions
        H=vartemp3*(1-exp(-vartemp4*vartemp1))+(1-vartemp3)*exp((vartemp4-1)*vartemp2);
        H0=(1-vartemp3)*exp(-vartemp2);
        H1=vartemp3*exp(-vartemp1);
        vartemp5=H+H1*vartemp4+H0*(vartemp4-1);
        x_norm.Set(ele,vartemp5);! update x_norm vector with new function values
    }
}
}
}

```

Stage 4: Creating the convergence plot

Once the optimization process has been completed (i.e., when the maximum number of function evaluations *Maxeval* has been reached), the plot of the *Objective Function versus Function evaluations* graphic is automatically deployed. For this purpose, the following code is included.

```

! Creating convergence plot
GrB = GetGraphBoard();
if (GrB=NULL) { output('No Graphics Board open'); exit(); }

Plot = NULL;
ViPg = GrB.GetPage(this:loc_name,1);
if (ViPg) {
    ViPg.SetResults(Convergence);
    Plot=ViPg.GetVI('qot','VisPlot',1);
    if (Plot) {
        Plot:use_x=1;
    }
}

```

```

    Plot.SetXVar(this,'b:ffx_eval');
    Plot.AddVars(this,'b:ffx_conv');
}
}
if (ViPg=NULL.or.Plot=NULL) {
    Warn('Failed to create plot, continue calculation ...');
}

```

12.3.2 Subroutine for Function Evaluation

The master DPL presented in the previous subsection has been structured for solving any kind of optimization problem in DigSILENT PowerFactory environment. For this purpose, the function evaluation needs to be performed in a separate subroutine (slave DPL), which is designed by the user to determine the values of the objective function and constraints depending on the specific optimization problem. Therefore, each function evaluation subroutine will be structured differently. The only mandatory conditions are as follows: (i) The DPL has to be named *Obj_function.ComDpl*, and (ii) it needs a *Result parameter* called *ffx* to be defined in *Advanced Options* section [this variable is the objective function that will be called by the main MVMO DPL script using this command: *Obj_function.Execute (ffx)*]. Figure 12.5 shows the DPL command window for the defined slave DPL.

For the sake of application illustration, this chapter uses MVMO to tackle the problem of optimal coordinated tuning of PSS, which aims at achieving a satisfactory overall system damping performance for all oscillatory modes (OMs). Mathematically, the problem has the following format [12]:

$$\min \text{OF} = |\zeta_{\text{th}} - \zeta_{\text{sys}}| \quad (12.11)$$

$$\zeta_{\text{sys}} = \min_{p=1 \dots \text{nm}} (\zeta_p) \quad (12.12)$$

subjected to

$$\mathbf{x}_{\min} \leq \mathbf{x} \leq \mathbf{x}_{\max} \quad (12.13)$$

where ζ_{th} is the minimum acceptable damping threshold (e.g., $\zeta_{\text{th}} > 15\%$) and ζ_{sys} is the minimum global damping of the system (the least of the nm system OMs). The vector \mathbf{x} constitutes the solution to the optimization problem (i.e., gain and time constants of the PSS model).

The block diagram of a typical PSS (which has been used in this chapter) is shown in Fig. 12.6. The PSS output signal, *upss*, is summed to the reference of the excitation system for a given generator. The washout time constant T_W and the output limits $\pm y_{\text{lim}}$ are considered to be known beforehand, i.e., $T_W = 10$ s, $y_{\text{lim}} = \pm 0.1$ [8, 11]. The parameters of the PSSs are optimized considering typical limits: $K \in [0, 100]$; $T_1, T_3 \in [0.01, 0.2]$; $T_1/T_2, T_3/T_4 \in [1, 15]$. These search bounds will be specified in the vector *Paralim.IntMat*.

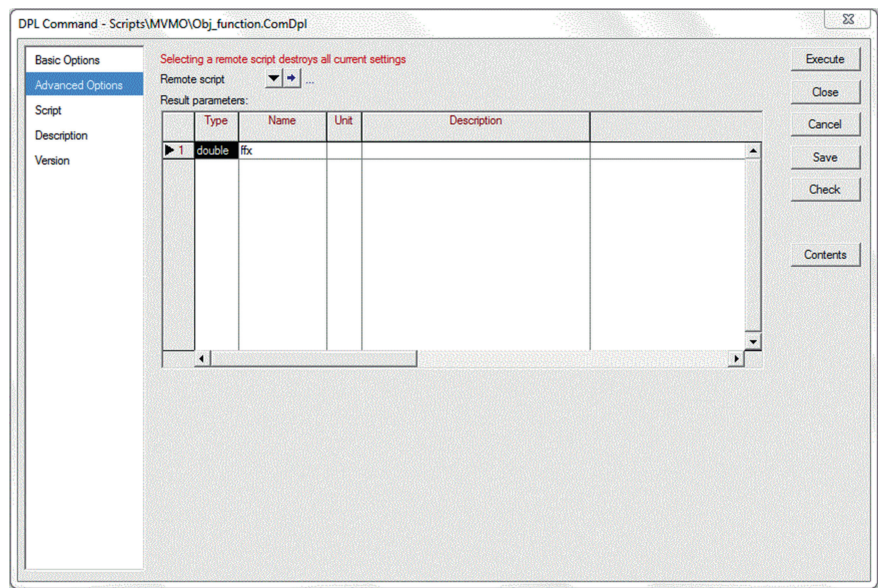


Fig. 12.5 Objective function subroutine DPL command advanced options

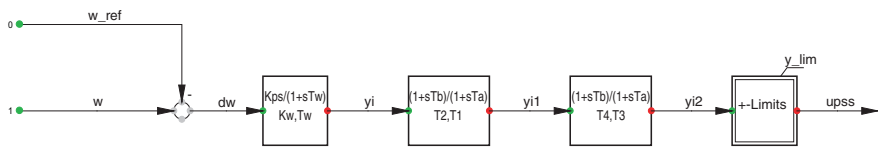


Fig. 12.6 PSS block diagram

For this specific application, it is necessary to modify the parameters of each PSS installed in the system. Thus, the *DSL common model* [4] representing each PSS is linked to the slave DPL as an *External Object*. In addition, the $x_{denorm}.IntVec$ containing the control variables for each iteration, as well as the *modal analysis results* object of the active *study case* (*Eigenvalues.ElmRes*), is also included as *External Objects* in order to viable the modal analysis execution in each iteration, whose eigenanalysis results allow computing the ζ_{sys} . Figure 12.7 presents the snapshot of the slave DPL command basic options, where the links with the specified *External Objects* are also shown.

First, the internal variable declaration and the definition of the damping threshold have to be specified.

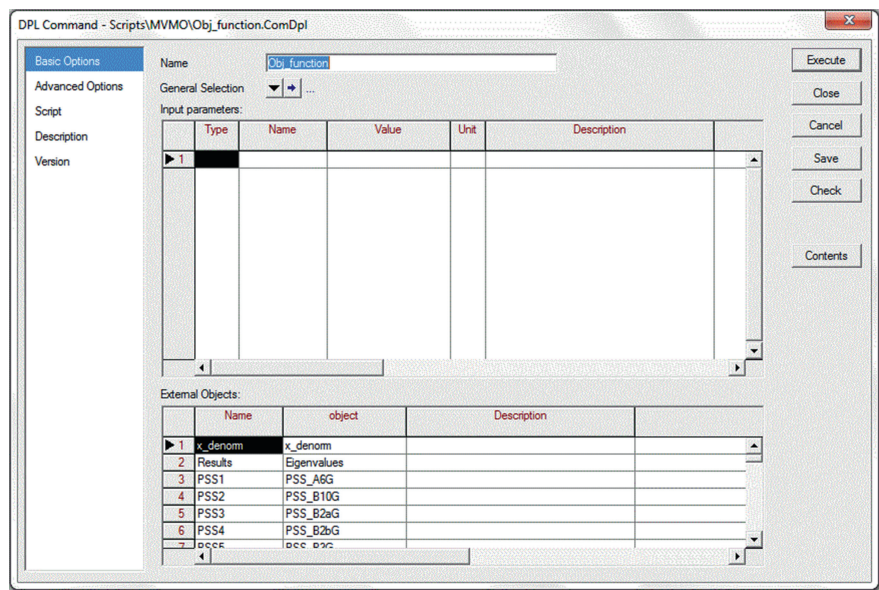


Fig. 12.7 Objective function subroutine DPL command basic options

```
! Objective Function Variable Declaration
int ele,nparam,cases,ele1,Neig;
double param,var1,var2,ele2;
double damplim, eigreal,eigimag,mindamp,minfreq,freq,damp;

! Defining the damping threshold
damplim = 0.15; ! Damping threshold equal to 15%
```

Afterward, the values of the control variables of each PSS (i.e., gains and time constants) have to be directed to the corresponding *Common Model*. The following lines represent the addressing of variables for one PSS (i.e., PSS1); however, similar commands will be used for all the required PSSs.

```
!!!! OBJECTIVE FUNCTION: Minimizing {abs(damplim-mindamp)}
nparam=x_denorm.Size();

for(ele=1;ele<=nparam/5;ele+=1){

    if (ele=1){
        PSS1:Kw=x_denorm.Get(1);
        var1=x_denorm.Get(2);
        var2=x_denorm.Get(3);
        PSS1:T1=var2;
        PSS1:T2=var2/var1;
        var1=x_denorm.Get(4);
        var2=x_denorm.Get(5);
        PSS1:T3=var2;
        PSS1:T4=var2/var1;
    }
    ...
}
```

!Rest of DPL script routines for other PSSs

The script that allows computing the objective function represented by (12.11) is shown as follows. Take into account that the *Calculation of Initial Conditions* (*Initialjc.ComInc*) and the *Modal Analysis* (*Modal.ComInc*) have been included in *Contents* section as *Internal Objects*.

```
! Computing eigenanalysis
mindamp=1;

Initialjc.Execute();
Modal.Execute();
LoadResData(Results);
Neig = ResNval(Results,0);

for(ele=0;ele<=Neig-1;ele+=1){
  GetResData(eigreal,Results,ele,0);
  GetResData(eigimag,Results,ele,1);
  if ({abs(eigimag)>0}){
    freq=abs(eigimag)/(2*pi());
    damp=-eigreal/sqrt(sqr(eigreal)+sqr(eigimag));
    mindamp=min(mindamp,damp);
  }
}

! Computing Objective Function
ffx = abs(damplim-mindamp)
```

12.3.3 Script for Load of Results

The subroutine for function evaluation allows loading each control variable to the corresponding address in the *DSL Common Model* of the PSS in each evaluation of the MVMO optimization process. However, when the entire process has finished, it is necessary to load the best parameters (stored in *x_denorm_best.IntVec*) into each PSS. For this purpose, a DPL script for loading the optimization results has been structured (*Load_Best_Param.ComDpl*).

This script presents the same structure as the previously presented code lines used to direct the values of the control variables of each PSS (i.e., gains and time constants) to the corresponding *Common Model*. The only difference is that the DPL linked external vector is now the *x_denorm_best.IntVec* instead of *x_denorm.IntVec*.

```

!-----Load Best Param
int ele,nparam,ele1;
double param,var1,var2;
!-----Load Best Param

nparam=x_denorm.Size();

for(ele=1;ele<=nparam/5;ele+=1){

    if (ele==1){
        PSS1:Kw=x_denorm.Get(1);
        var1=x_denorm.Get(2);
        var2=x_denorm.Get(3);
        PSS1:T1=var2;
        PSS1:T2=var2/var1;
        var1=x_denorm.Get(4);
        var2=x_denorm.Get(5);
        PSS1:T3=var2;
        PSS1:T4=var2/var1;
    }
    ...
}
!Rest of DPL script routines for other PSSs

```

12.4 Application Examples

In this section, the results of optimal coordinated tuning of PSSs in a test power system are presented. For this aim, the implemented MVMO and the stated objective function have been run. The QR method is used for full eigenvalue computation via *Modal Analysis*.

Finally, comparisons of eigenanalysis results and time-domain simulations have been also included in order to highlight the excellent performance of the tuning.

12.4.1 Test System

A slightly modified version of the PST 16-machine test system, consisting of three strongly meshed areas, 80 buses, 16 generators, 28 transformers, 52 transmission lines, and a thyristor-controlled series compensator (TCSC), has been used in the simulations (details of the TCSC implementation can be found in Chap. 17). Also, all generators are equipped with PSSs. This test system has been presented in Chap. 11 of this book, where interested readers can find details of its implementation.

12.4.2 Simulation Results

In this chapter, it is considered that the system has sufficient damping when the critical modes have $\zeta \geq 15\%$ (i.e., $\zeta_{th} = 15\%$). The purpose of this criterion is to ensure that no OM in the system has a damping ratio less than the specified threshold.

First, *Modal Analysis* is run considering that all the PSSs are disconnected. Table 12.2 presents the critical OMs (i.e., damping less than ζ_{th}) where it is possible to observe even the existence of an interarea negatively damped OM.

From the previous results, the necessity of carrying out a rational PSS tuning is easily obtained. Thus, the MVMO-based PSS tuning methodology presented in this chapter is applied to the test power system.

The first stage is to define all the MVMO basic parameters, including the definition of the *Paralim.IntMat* elements (i.e., limits of each optimization variable). In this case, a total of 80 control variables have been defined (i.e., 5 variables times 16 PSSs). The limits of each variable have been specified in Sect. 12.3.2.

Following the described procedure, the parameters of each PSS are identified by performing the minimization of the objective function shown by (12.11). Figure 12.8 shows the convergence of the objective function after the optimization process has finished (which presented a total elapsed time of 12 min). It can be noticed that the minimum is reached at around 180 function evaluations.

After the optimization process has finished, and the optimized parameters have been loaded, *Modal Analysis* is again run considering the obtained PSS tuning. Table 12.3 presents the five less-damped OMs obtained after PSS tuning. It can be noticed that even the least-damped mode accomplishes the predefined threshold.

Time-domain simulations were also performed to further assess the quality of the obtained results. For this purpose, a three-phase short-circuit was applied in the middle of line B–C at 0.1 s with a duration of 100 ms. Figure 12.9 presents the oscillograms corresponding to the active and reactive power of two thermoelectric generators (i.e., C10G and C12G), whereas Fig. 12.10 presents the oscillograms corresponding to the active and reactive power of two hydroelectric generators (i.e., A1bG and A2aG).

It can be noticed from the plots how the system presents oscillatory instability when the PSSs are not tuning. This unstable condition is caused by the presence of

Table 12.2 Eigenanalysis results without PSSs

Mode	Real part (1/s)	Imaginary part (rad/s)	Damped frequency (Hz)	Damping ratio (%)
88	−0.950	6.426	1.023	14.63
98	−0.806	5.850	0.931	13.65
82	−0.878	6.816	1.085	12.78
92	−0.761	6.233	0.992	12.12
106	−0.628	5.444	0.866	11.47
100	−0.672	5.833	0.928	11.45
96	−0.692	6.003	0.955	11.45
104	−0.622	5.479	0.872	11.28
110	−0.526	4.662	0.742	11.21
108	−0.540	4.900	0.780	10.96
102	−0.357	5.257	0.837	6.77
112	−0.155	3.388	0.539	4.56
114	0.080	3.081	0.490	−2.59

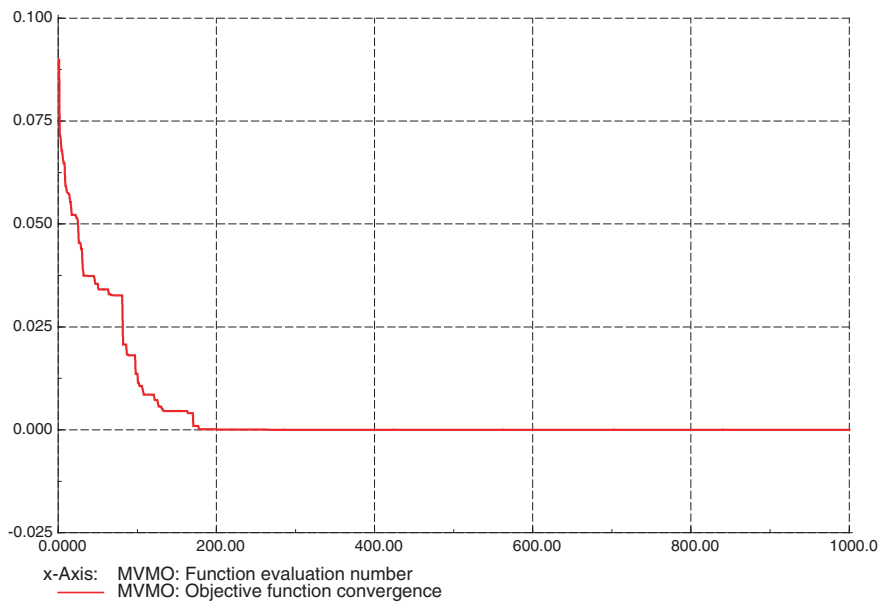


Fig. 12.8 Objective function convergence

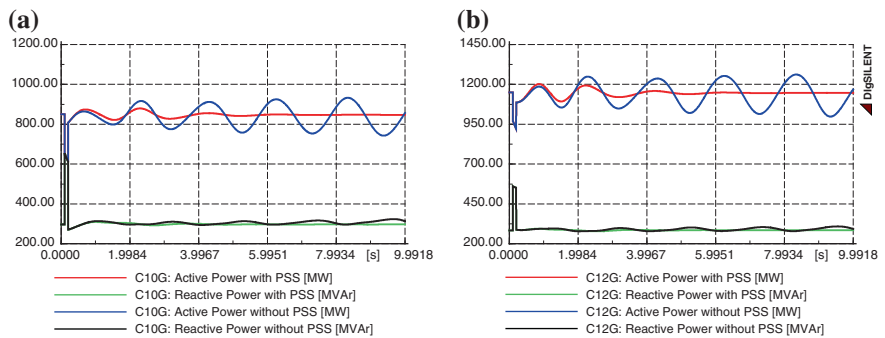


Fig. 12.9 Active and reactive power of thermoelectric. **a** Generator C10G. **b** Generator C12G

Table 12.3 Eigenanalysis results after PSS tuning

Mode	Real part (1/s)	Imaginary part (rad/s)	Damped frequency (Hz)	Damping ratio (%)
129	−0.949	5.865	0.933	15.97
149	−0.471	3.009	0.479	15.47
125	−0.955	6.148	0.978	15.35
135	−0.790	5.165	0.822	15.13
133	−0.859	5.659	0.901	15.00

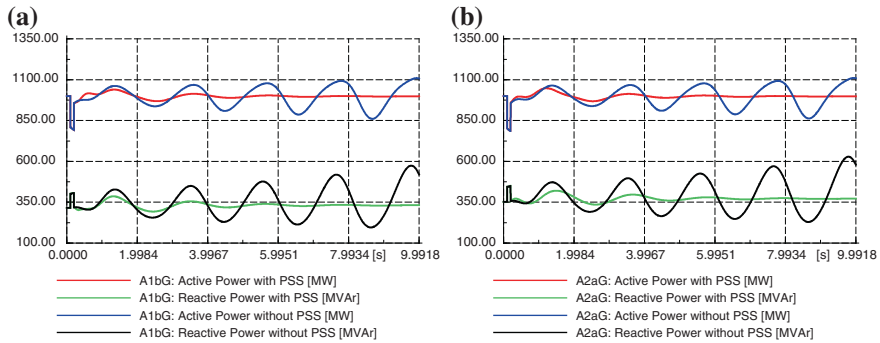


Fig. 12.10 Active and reactive power of hydroelectric. **a** Generator A1bG. **b** Generator A2aG

the interarea negatively damped OM. On the other hand, after PSS's tuning, the system presents a post-contingency stable condition, which ratifies the excellent performance of the supplementary damping-coordinated tuning.

12.5 Concluding Remarks

This chapter presents the detailed implementation of a novel heuristic optimization algorithm, called MVMO into DiGSILENT PowerFactory environment via DPL. The application of the implemented MVMO to solve the problem of optimizing the coordinated tuning of PSSs is also presented in this chapter.

The optimization problem is solved for a specific operating scenario of a slightly modified version of the PST 16-machine test system. Numerical results of modal analysis attest the outstanding performance of the proposed MVMO-based methodology in determining the best set of PSS's parameters in order to accomplish the predefined damping threshold.

Some time-domain simulations have also been presented in order to show the dynamic behavior of the system considering or not the influence of PSSs. The results obtained confirm that the PSS tuning allows obtaining a stable system, which agrees with the results obtained from the modal analysis.

References

1. Cepeda JC, Rueda JL, Erlich I (2012) Identification of dynamic equivalents based on heuristic optimization for smart grid applications. In: Proceedings of the IEEE world congress on computational intelligence, pp 1–8
2. Chamba MS, Añó O (2013) Economic dispatch of energy and reserve in competitive markets using meta-heuristic algorithms. IEEE Latin Am Trans 11(1):473–478

3. Chiong R, Weise T, Michalewicz Z (eds) (2012) Variants of evolutionary algorithms for real-world applications. Springer, Berlin
4. DlgSILENT GmbH (2011) DlgSILENT PowerFactory Version 14.1 User's Manual, Gomaringen, Germany, May 2011
5. Erlich I, Venayagamoorthy GK, Nakawiro W (2010) A mean-variance optimization algorithm. In: Proceedings of the IEEE congress on evolutionary computation, pp 1–6
6. Erlich I, Nakawiro W, Martinez M (2011) Optimal dispatch of reactive sources in wind farms. In: Proceedings of the IEEE PES general meeting, pp 1–7
7. Lee KY, El-Sharkawi MA (eds) (2008) Modern heuristic optimization techniques. Wiley, Hoboken
8. Machowski J, Bialek J, Bumby J (2008) Power system dynamics: stability and control. Wiley, West Sussex
9. Nakawiro W, Erlich I, Rueda JL (2011) A novel optimization algorithm for optimal reactive power dispatch: a comparative study. In: Proceedings of the 4th international conference on electric utility deregulation and restructuring and power technologies, pp 1555–1561
10. Pringles R, Rueda JL (2012) Optimal transmission expansion planning using mean-variance mapping optimization. In: Proceedings of the sixth IEEE/PES transmission and distribution: Latin America conference and exposition, pp 1–8
11. Rogers G (2000) Power system oscillations. Kluwer Academic Publishers, Boston
12. Rueda JL, Guamán WH, Cepeda JC, Erlich I, Vargas A (2013) Hybrid approach for power system operational planning with smart grid and small-signal stability enhancement considerations. IEEE Trans Smart Grid—Spec Issue Comput Intell Appl Smart Grids 4 (1):530–539
13. Simon D (2013) Evolutionary optimization algorithms: biologically inspired and population-based approaches to computer intelligence. Wiley, Hoboken
14. Talbi EG (2009) Metaheuristics: from design to implementation. Wiley, Hoboken

Chapter 13

Application and Requirement of DIgSILENT PowerFactory to MATLAB/Simulink Interface

Shadi Khaleghi Kerahroudi, Mohsen M. Alamuti, F. Li, G.A. Taylor and M.E. Bradley

Abstract Simulation tools are the most economical solution for modelling and design of various components of large-scale power systems. However, as the complexity in the integrated electric power network grows, the need for more comprehensive simulation tools rises. Modern simulation tools have therefore been developed in line with this ever increasing need by means of integration of improved user interfaces. Interfacing of the simulation tool to an external program/mathematical tool extends the capability of the simulation tool in a more effective execution of the simulation, particularly in an area where the external program/algorithm provides more advanced technique and flexibility, e.g. the advanced control system design in MATLAB control toolbox. This chapter describes the process of interfacing DIgSILENT PowerFactory to MATLAB/Simulink program. Two study cases will be presented in this chapter. These study cases are set up and provided to allow the readers to understand the process and steps of linking DIgSILENT PowerFactory to the MATLAB/Simulink program.

Keywords DIgSILENT PowerFactory to MATLAB/Simulink interface • Control system design • TCSC and PSS-based POD

Electronic supplementary material The online version of this chapter (doi: 10.1007/978-3-319-12958-7_13) contains supplementary material, which is available to authorized users.

S.K. Kerahroudi (✉) · M.M. Alamuti · G.A. Taylor
Brunel Institute of Power Systems, BIPS, Brunel University, London, UK
e-mail: shadi.kerahroudi@nationalgrid.com

M.M. Alamuti
e-mail: mohsen.mohammadi@brunel.ac.uk

F. Li · M.E. Bradley
Market Operation, National Grid, London, UK

13.1 Introduction to the MATLAB/Simulink to PowerFactory Interface

Simulation tools have been used extensively by manufacturers, designers and system operators for the analysis and assessment of various aspects of the design and operation of power systems with ease, without resorting to costly and potentially harmful tests on the real system. As the complexity in the integrated electric power network grows, the need for more comprehensive simulation tools rises. The simulation tools are expected to cater for all the standard power system analysis needs including high-end applications in new technologies such as wind power and distributed generation [1, 2]. Therefore, modern simulation tools have been developed in line with this ever increasing need by means of integration of the improved user interfaces. Interfacing between simulation tools, external programs or mathematical tools extends the capability of the simulation tool in the execution of the simulation more effectively, particularly in areas where the external program/algorithm provides more advanced techniques and flexibility, i.e. advanced control system design in MATLAB's control tool box.

13.1.1 Requirement for Interfacing

The standard library of simulation tools which normally includes models of synchronous/asynchronous machines, transmission lines, transformers, semiconductors, power electronic devices and simple processing functions [such as gain blocks, integrators and proportional-integral (PI) controllers] facilitate the fast modelling of the electrical network. However, not all control system components are readily available to perform specific computation or design of a complex control law. Commonly, a comprehensive analysis or design requires computational facilities, advanced control blocks, digital processors and the capability of modelling nonlinear elements, etc. Therefore, it is sometimes essential to combine the two programs so that each could provide particular features either for more detailed modelling of certain aspects of a complex power network or for higher computational capability [1]. For that reason, the majority of existing simulation programs are capable of extending their capabilities by interfacing to external programs/software. In this approach, each software tool follows certain standards to allow the software developed with various vendors to be implemented together through the use of interfaces [2]. The application of interfacing has been reported through the simulation of a power electronic convertor, the design of a complex protection system, the design of an advanced digital control system and simply adding another feature such as optimisation to the main simulator [1]. This chapter describes the process of interfacing PowerFactory to MATLAB/Simulink using two study cases. These examples demonstrate how interfacing between two tools is performed in order to allow full exploitation of both packages' capabilities. Consequently, the powerful

mathematical and control functions including the various toolboxes available in the MATLAB program can be implemented with the fast and powerful modelling and simulation capability of PowerFactory.

13.2 Characteristics of the Test System for the First Study Case

13.2.1 Modelling of the Thyristor-Controlled Compensation (TCSC)

A typical TCSC model consists of a fixed series capacitor bank connected in parallel with a thyristor-controlled reactor (TCR) as shown in Fig. 13.1. When the thyristors of the TCSC are blocked, the TCSC operates as a conventional series compensator with a fixed reactance ' X_C ' and when the thyristors are conducting, the device can vary its net reactance. This series reactance could be adjusted within its limit to keep the specific amount of active power flow across the line. In this study, a simple model of TCSC is implemented using the concept of variable series reactance.

13.2.2 Test System

A single machine infinite bus (SMIB) system, which is presented in Fig. 13.1, is used to show the process of interfacing PowerFactory to MATLAB/Simulink. The test system consists of two AC lines (*ElmLine*) which connect the two areas. The generator (*ElmSym*) in each area is represented by a full order synchronous machine model, which is equipped with both governors and voltage regulators in PowerFactory. One of the AC lines is equipped with a TCSC, where the initial fixed series compensation (FSC) is 30 % of the total line reactance. The TCSC can provide up to 75 % compensation to the total reactance of a single line route by changing the

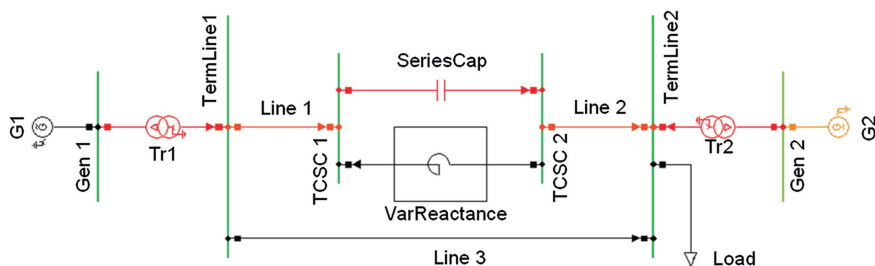


Fig. 13.1 Test system in PowerFactory

reactance of the TCR. The TCR is modelled as a variable series reactance (*ElmSind*) ‘VarReactor’, and the steady-state value of the inductance, which is 20 Ω , is defined in the load flow model. It should be noted that as the internal or external controller provides the control signal (XI, reactance of TCR branch), it is essential that the initial condition of TCSC reactance is defined and set in the design of the controller in PowerFactory and MATLAB/Simulink. The control variable is the reactance of the TCR branch. In fact, the active power in line 1 is regulated by varying the reactance of the series reactor. The control system, in this example, is designed to regulate the power flow on line 1 and line 2 in order to minimise inter-machine/inter-area oscillations under normal operations such as tracking of the set-point changes. (The ‘DIgSILENT PowerFactory’ model of this test system is available within the book CD).

13.3 Overall Structure of the DIgSILENT PowerFactory to MATLAB Interface

In the process of linking PowerFactory to MATLAB/Simulink, the following programs interact throughout the simulation time [4].

- PowerFactory
- MATLAB data models in the form of m.files (m)
- MATLAB/Simulink model (mdl)

The m.files and MATLAB/Simulink models represent the model and algorithms of the external controllers, and the PowerFactory model represents the power system. In order to implement a MATLAB model/algorithm in a PowerFactory project, it needs to be included in a frame similar to the dynamic simulation language (DSL) model definition. The first step is to create a ‘slot’ inside a composite frame (*ElmComp*) in the PowerFactory project where the model/algorithm should be inserted. A block definition (*BlkDef*) is also required to be created in the ‘type’ library, which allows definition of the MATLAB code to be imported by ticking the MATLAB m.file check box in the classification section and providing the address of the m.file (as shown in Fig. 13.3f). (Please note that users need to insert the address of the location where the provided m.file is stored).

In the block definition (*BlkDef*) of this composite frame, all of the required inputs from PowerFactory, the outputs obtained from MATLAB and the state variables must be defined. In addition, all the defined inputs, outputs and state variables in this block must be declared properly in the m.file. In each step of the simulation, PowerFactory inserts all the signals defined as ‘global variables’ into a ‘common workspace’ that can then be used by the external models or algorithms in MATLAB. It is therefore crucial that all signals including the controller parameters, inputs, outputs and state variables are defined as a global variable in the m.file [4]. The function that the MATLAB/Simulink program is tasked to perform can either be a built-in MATLAB function or a user-defined algorithm.

In order to provide an illustrative instruction on how the interface between PowerFactory DIgSILENT and MATLAB/Simulink can be created, two study cases are provided in this chapter in the form of a step-by-step tutorial. The first study case demonstrates the design of an external PI controller for the control of a TCSC line. The second study case presents the steps in the design of a PSS-based power system oscillation damping (POD) control for the TCSC line using the interface capability of the two tools. The provided study describes the implementation and linking of the POD controller in MATLAB/Simulink to damp inter-area oscillations for the simulated two-area four-machine network designed in PowerFactory DIgSILENT.

13.4 Study Case 1

13.4.1 Demonstrating the Process of Interfacing PowerFactory to MATLAB/Simulink

The first study case is set up to show the process and aspects related to interfacing PowerFactory and MATLAB/Simulink programs. In this study case, a PI controller is designed externally in Simulink to regulate the power flow in line 1 and line 2 (which are modelled in PowerFactory) using the MATLAB to PowerFactory interface capability. Regulation of the power in these lines minimises the inter-machine/inter-area oscillations under normal operations by tracking the TCSC's set-point changes. To examine the correct linking between the two tools, the performance of the externally designed PI controller in Simulink is compared with the case when the built-in PI controller from the PowerFactory library is used to regulate the power in line 1 and line 2.

13.4.2 The Built-in PI Controller in PowerFactory

In this case, firstly, a composite frame, shown in Fig. 13.2a, is created and the input (measured power on TCSC line) and output (control signal) of the TCSC control model are defined. Also, a composite model (*ElmComp*) related to the TCSC frame is created, which defines the link of each slot in the TCSC frame to the relevant component (as shown in Fig. 13.2d).

A built-in PI controller (*BlkDef*) from the PowerFactory library is used for the control of the TCSC, which was created in the TCSC control model (Fig. 13.2b) consisting of a PI controller and a limiter (*BlkDef*) [all the equations for the calculation of the initial conditions in the TCSC control model are defined in the block definition (*BlkDef*) of the TCSC control model (Fig. 13.2c)]. The limiter is used to improve the controller's response to large deviations of the input signal. The PI

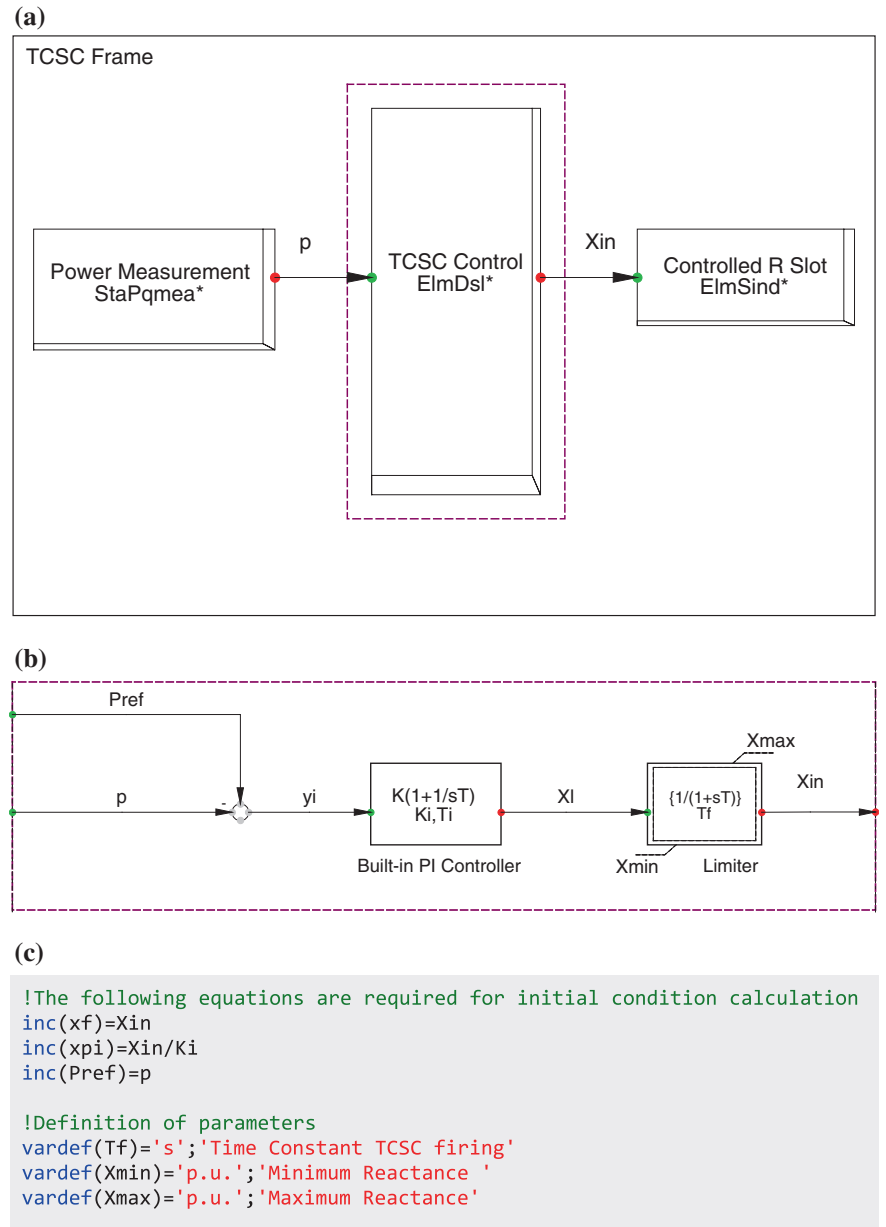


Fig. 13.2 Structure of a built-in PI controller in PowerFactory. **a** Composite TCSC frame in PowerFactory. **b** TCSC control model in PowerFactory. **c** Block definition of TCSC control model (initial condition equation page). **d** Defined composite model for the TCSC model. **e** Common DSL model of TCSC control frame

(d)

Composite Model - GridTCSC FRAME.ElmComp

Name: TCSC FRAME

Frame: Library\Models\TCSC Frame

☐ Out of Service

Slot Definition:

	Slots BlkSlot	Net Elements Elm*Sta*IntRef
1	Controlled R Slot	✓ VarReactance
2	TCSC Control	✓ TCSC Control
▶ 3	Power Measurement	✓ PQ Measurement

Slot Update Step Response Test

OK Cancel Contents

(e)

Common Model - GridTCSC FRAME\TCSC Control.ElmDsl

Name: TCSC Control

Model Definition: Library\Models\TCSC Control

☐ Out of Service ☐ A-stable integration algorithm

	Parameter
Tf Time Constant TCSC firing [s]	0.05
Ki	0.01
Ti	0.002
Xmin Minimum Reactance [p.u.]	12.8
▶ Xmax Maximum Reactance [p.u.]	30.

Export to Clipboard

OK Cancel Events

Fig. 13.2 (continued)

controller has two components, proportional and integrator $K(1 + 1/ST)$, (with the gain of $K_i = 0.01$ and integration time of $T_i = 0.002$). The parameters of the PI controller are defined in a common DSL model (*ElmDsl*) as shown in Fig. 13.2e. It should be noted that the same parameters are used in the design of an external PI controller in Simulink.

13.4.3 The External PI Controller in MATLAB/Simulink

13.4.3.1 Setting up the Required Structure (on the PowerFactory Side) to Create an Interface

As described in Sect. 13.3, the three programs [PowerFactory model, MATLAB m. files (m) and MATLAB/Simulink model (mdl)] interact throughout the simulation time when the interface capability is implemented. In this example, the Simulink model (mdl), shown in Fig. 13.3b, represents the externally designed PI controller. The parameters of the external PI controller are defined in a common DSL model (*ElmDsl*) of PowerFactory, as shown in Fig. 13.3e.

As described in Sect. 13.3, the first step of creating the interface to allow the insertion of the MATLAB/Simulink model is to create a slot (*ElmDsl*) inside the TCSC composite frame in the PowerFactory project, which in this study case is named 'MATLAB TCSC control' shown in Fig. 13.3a. This slot represents the PI controller model in Simulink (mdl). Each slot in the TCSC frame is linked and sourced to the relevant component using the created composite model (*ElmComp*) for the TCSC frame presented in Fig. 13.3c. Since all of the required inputs from PowerFactory, the outputs obtained from MATLAB/Simulink and the state variables must be defined, a block definition (*BlkDef*) for the MATLAB TCSC control slot is created and all inputs, outputs and state variables are defined as shown in Fig. 13.3f. In addition, all the initial conditions (Fig. 13.3g) are defined in this block definition (*BlkDef*).

In this example, the input signals are measured power in line 1 (defined as P in Fig. 13.3a, b, d) and set point of the TCSC line (defined as P_{ref} in Fig. 13.3b) which are sent from PowerFactory to the Simulink model. The output of the Simulink model (y) is the control signal of the PI controller that is sent to PowerFactory at each step of the simulation (defined as X_{in} in Fig. 13.3a, b, d). It is essential that all the inputs, outputs and state variables defined in the created block definition (*BlkDef*) (shown in Fig. 13.4f) are declared as a global variable in the m.file which is presented in Sect. 13.4.3.2.

The parameters of the PI controller in Simulink (mdl) are defined in a common DSL (*ElmDsl*) model in PowerFactory as shown in Fig. 13.3e.

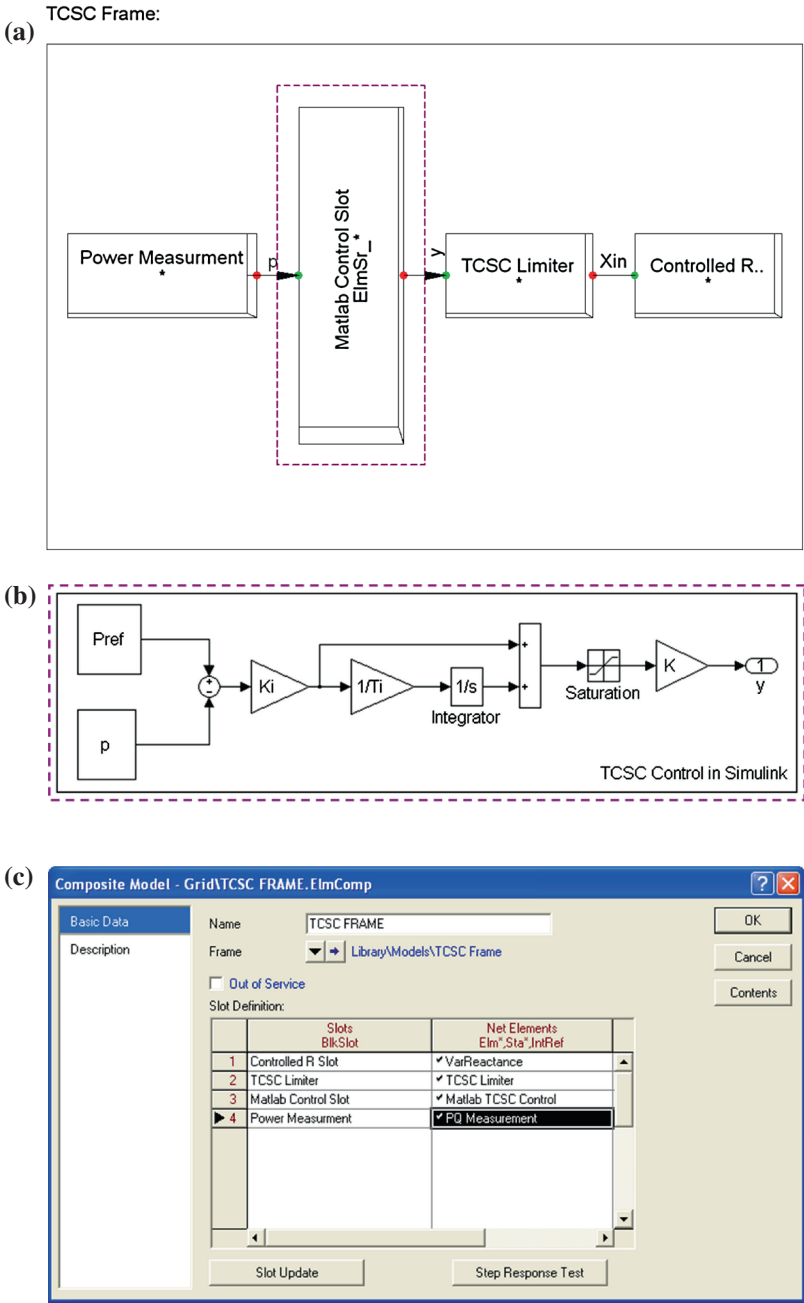


Fig. 13.3 Structure of the interface for design of the PI controller in Simulink. **a** Composite TCSC frame model in PowerFactory. **b** External PI controller in MATLAB/Simulink. **c** Composite model for TCSC frame in PowerFactory. **d** Power measurement (input of TCSC control)—output of TCSC control. **e** Common DSL model for MATLAB TCSC control. **f** MATLAB TCSC control, block definition (basic data page) **g** MATLAB TCSC control block definition (equations page)

(d)

Slot - ...aryModels\TCSC Frame\Power Measurement.BlkSlot [?] [X]

Name: Power Measurement [OK] [Cancel]

Sequence: 3

Block Definition: [v] [→] ...

Filter for:

Class Name: *

Model Name: *

Classification:

☒ Linear

☐ Automatic, model will be created

☒ Local, model must be stored inside

☐ Main Slot

Upper Limitation:

Limiting Input Signals: []

Lower Limitation:

Limiting Input Signals: []

Variables:

Output Signals: p

Input Signals: []

Slot - LibraryModels\TCSC Frame\Controlled R Slot.BlkSlot [?] [X]

Name: Controlled R Slot [OK] [Cancel]

Sequence: 0

Block Definition: [v] [→] ...

Filter for:

Class Name: *

Model Name: *

Classification:

☒ Linear

☐ Automatic, model will be created

☒ Local, model must be stored inside

☐ Main Slot

Upper Limitation:

Limiting Input Signals: []

Lower Limitation:

Limiting Input Signals: []

Variables:

Output Signals: []

Input Signals: X_{in}

Fig. 13.3 (continued)

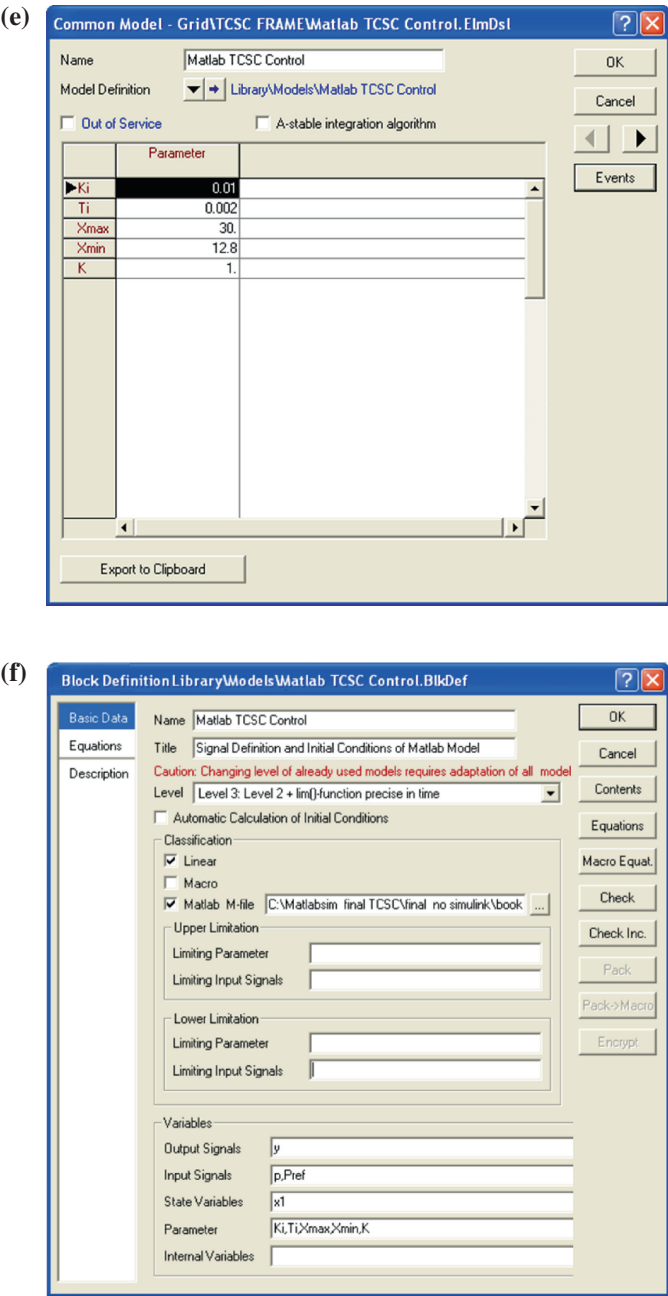


Fig. 13.3 (continued)

(g)

```
!The following equations are required for initial condition calculation

inc(Pref)=p
inc(x1)=y/K

y=0.001
x1.= 0
```

Fig. 13.3 (continued)

13.4.3.2 Initial Condition Definition and Calculation

Setting un-correct initial conditions for newly defined controller frames can stop the program or introduce un-necessary oscillations at the beginning of the simulation which might not be damped quickly enough. The initial conditions have to be defined for any block that contains state variables. This can be implemented within the block itself (i.e. the PI controller) or within the controller frame which contains the mentioned block. There is no strict way to define such initial conditions. However, the user has to set the values in a way that the calculated initial conditions to be as close as possible to the calculated load flow results.

Since the system is in its steady-state condition when initialized, all the error signals (if any feedback controller is employed) have to be zero. The initial condition has to be defined for all the state variables in the frame. Defining the initial values for controller inputs and outputs is optional unless the equation determining

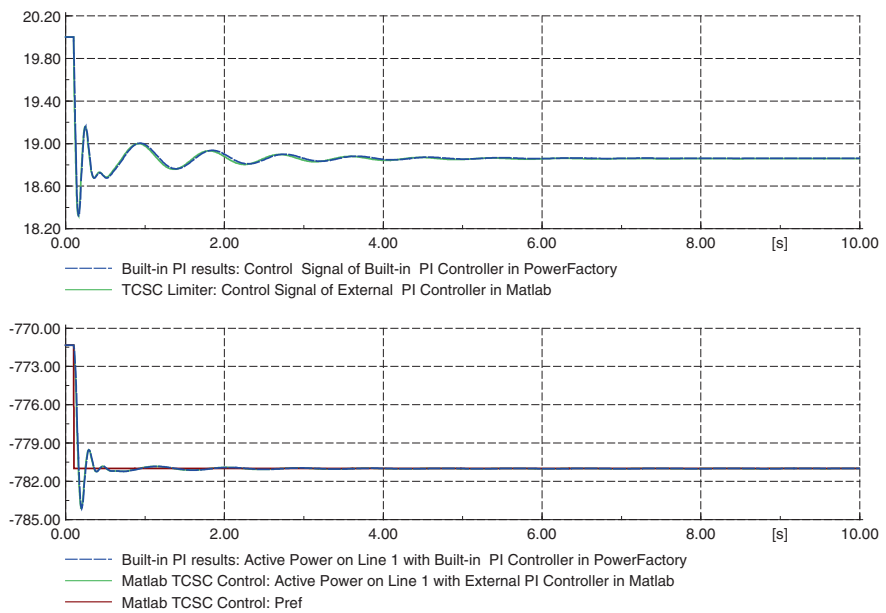


Fig. 13.4 Comparison of built-in and external PI controller performance

the initial values of the state contains any of the input signals. As a general guideline, the back calculation method can be used. In most cases, the initial value of frame output is known to DIGSILENT through the load flow analysis which is executed prior to the initial condition calculation (if not it can be given the initial value). The initial value of the next block in line which has a state variable is given in a way that the calculated output value matches the known output value.

13.4.3.3 Setting up the Required Structure (on MATLAB/Simulink Side) to Create the Interface

After creating the right structure for implementing the interface in PowerFactory, an external PI controller is created in Simulink (as shown in Fig. 13.3b). Since DIGSILENT is only able to call scripts, any Simulink file has to be executed via a script or MATLAB function such as:

```
Function [t, x, y] = TCSCcontrol
global p Pref ki Ti Xmax Xmin k x1

% Get the simulation parameters
options = simget('TCSCcontrol_model');

% Set updated initial conditions
options = simset('InitialState',[x1]);

% Run the simulation and update the output and states value
[t, x, y] = sim('TCSCcontrol_model', [], options);
```

13.4.4 Performance Comparison of Built-in and External Controllers

The performances of both the built-in and external PI controllers following the change of set point of the TCSC line (P_{ref}) are presented in Fig. 13.4. The control signals generated by both controllers and the active power in line 1 that is under the control of PI controllers are shown in Fig. 13.4a, b respectively. The simulation results show that the behaviour and performance of the external PI controller in simulink are identical with it perfectly matching the performance of the built-in PI controller in PowerFactory. This implies that the interface between PowerFactory and MATLAB/Simulink is conducted correctly and that the inputs and outputs are exchanged accurately between the two tools at each simulation step.

13.5 Study Case 2

13.5.1 Design and Implementation of the PSS-Based POD for TCSC Using MATLAB/Simulink Script

In this section, two approaches are implemented in the design of the supplementary PSS-based POD control for the TCSC as an actuator. In the first approach, the POD controller is designed in Simulink and the interface capability of the PowerFactory and MATLAB/Simulink is implemented to link this POD controller to the test power system simulated in PowerFactory, whereas in the second approach, only a MATLAB script is used for the design of the POD control for the TCSC.

13.5.2 Inter-area Oscillation Control with TCSC

Ideally, all generators in the power system are synchronous meaning they rotate with the same angular velocity. In reality, this is not the case and the generators have slightly different rotor speeds or angles to deliver the required energy demand. Disturbances may increase the severity of such oscillations. Apart from the generator PSS, various flexible AC transmission system (FACTS) devices can be equipped with supplementary POD damping controllers to damp the aforementioned inter-machine (area) oscillations [5–8]. For this study case, the process for interfacing a PSS-based POD controller for TCSC to damp the inter-area oscillations is explained.

13.5.3 Characteristics of the Test System for the Second Study Case

The test system used for illustrating the application of TCSC supplementary damping control is given in Fig. 13.5. It is based on a two-area four-machine system

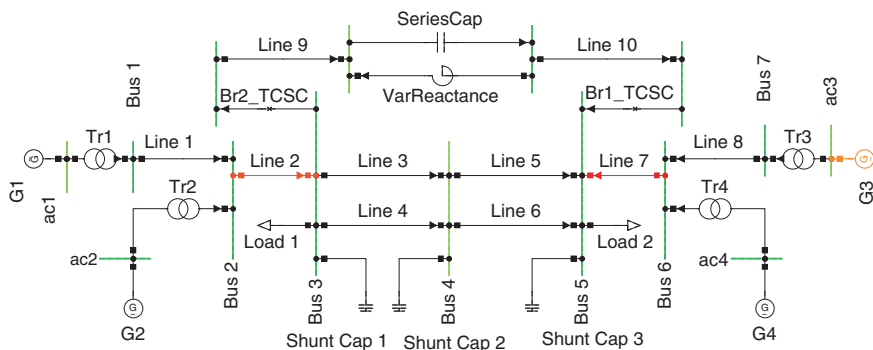


Fig. 13.5 Schematic of the two-area four-machine system with TCSC

with an extra AC line. In the mid-point of the additional AC line there is a TCSC that provides 40 % compensation of the line reactance. (The 'DIgSILENT PowerFactory' model of this test system is available within the book CD).

As described in Sect. 13.2.1, the TCSC is an impedance compensator component that is added in series to an AC transmission line to increase the power transfer capability of the line. It is also able to contribute to the power system stability enhancement by load flow control of the system. For power system stability studies, the thyristor operation is negligible in the simulations. Consequently, for such studies, a TCSC can be represented using a simplified and ideal model with only a fixed capacitor (*ElmScap*) in parallel with a variable reactor (*ElmSind*).

During the steady-state operation of the system, the TCSC reactance is controlled to provide the expected power flow. In addition, the TCSC can also play an important role in improving the damping of low frequency power oscillations. This can be realised by adding a supplementary damping controller that provides a damping signal to dynamically change the TCSC reactance. The design of the supplementary controller is out of the scope of this chapter. In the present case study, a typical PSS-based POD supplementary controller for TCSC is designed and tuned based on the modal and residue analysis.

13.5.3.1 Structure of the Supplementary PSS-Based Damping Controller for the TCSC

In this study case, the supplementary damping controller for the TCSC line is a PSS-based damping controller which has a similar structure to the generators' PSS, incorporating a phase compensation block, washout filter, low pass filter, gain and limiter (as shown in Fig. 13.8). An ideal damping controller shifts the selected eigenvalue of the system into the left side of the complex plain in order to make the system stable and improve oscillation damping. However, the selected mode of interest must be both controllable by the chosen input and observable in the chosen output for the feedback control to have any effect on the mode. Therefore, the selection of suitable feedback variables is critical to the design of any damping controller. In this study, the difference between the generators frequency is implemented as an input signal for the damping controller.

13.5.4 Interfacing PowerFactory and MATLAB/Simulink for Design of POD Control

This section provides a step-by-step tutorial on the application of interfacing DIgSILENT PowerFactory and Simulink in the design of a PSS-based POD control for the TCSC. The provided study case 2 explains the implementation and linking of the POD controller in Simulink to damp inter-area oscillations for the simulated

two-area four-machine network, as designed in PowerFactory. The two-area four-machine network is one of the well-known network model used for stability analysis. In the present study, the unstable or poorly damped modes of the system are excited by creating a temporary three-phase fault followed by the clearance of the fault and disconnection of the AC line. To be able to benefit from this tutorial, the user is required to be familiar with basic modelling in DIgSILENT PowerFactory and MATLAB/Simulink.

13.5.4.1 Setting up the Required Structure for Interface (on the PowerFactory Side)

Since the purpose of this tutorial is to explain how to interface Simulink/MATLAB to PowerFactory, the procedure for the design and implementation of the POD controller within PowerFactory is not explained here. Further details of the controller design can be found in [7]. As previously explained, a supplementary POD controller is used to damp the inter-area oscillations. The provided controller adds the POD signal to the set point of the TCSC and consequently adjusts the value of the variable reactance in order to control the flow of the power through the line and ultimately control the rotor oscillations of the generators. The designed POD controller measures frequencies from two areas of the system. The measurements are obtained at G2 (*ElmSym*) and G4 (*ElmSym*) as presented in Fig. 13.6a, b. Then, the measured frequencies (Fig. 13.6a) are fed into the POD controller block. As shown in Fig. 13.6a, the POD controller block sends the POD signal to the TCSC control. In the TCSC frame, the blocks inside the purple rectangular box (in Fig. 13.6a) represent the components related to design of the POD controller.

The name, inputs and outputs of each block are shown in Fig. 13.6. In the next step, similar to study case 1, required slots (shown in purple rectangular box in Fig. 13.6a) within the TCSC composite frame need to be created to allow the implementation and definition of the MATLAB/Simulink model. Then, the composite model (*ElmComp*) related to the TCSC frame has to be updated, and three newly created slots have to be associated with the relevant components. The ‘ac2 frequency’ and ‘ac4 frequency’ slots (*BlkSlot*) are associated to G2 (*ElmSym*) and G4 (*ElmSym*), respectively, as shown in Fig. 13.6b. Also, a new block definition (*BlkDef*) has to be created for the ‘Damping Controller’ as presented in Fig. 13.6e.

After creating the MATLAB controller block, in the block definition (*BlkDef*), all the variables as well as the link to the MATLAB file have to be defined. This procedure has been shown in Fig. 13.6e. In the Classification section (Fig. 13.6e), the user needs to tick the MATLAB m.file check box and provide the address of the m.file within the box. Next, all the variables including inputs, outputs, state variables and constant parameters of the controller are required to be defined in the variables section. The defined variables have to be initialized as shown in Fig. 13.6f. Since during the steady-state operation of the system, the controller does

not provide any output, the initial condition for the controller output and its state variables are set to zero. This is also the case for the time derivative of the state variables. The command ‘vardef’ provides an extra explanation for each parameter in the controller parameters definition page (*BlkDef*).

13.5.4.2 Setting up the Required Structure for Interface (on MATLAB/Simulink Side)

After implementing the previously explained instructions and setting up the interface in DIgSILENT PowerFactory, the MATLAB script and Simulink file has to be prepared. Since PowerFactory is only able to call scripts, any Simulink file has to be executed via a script or MATLAB function. The following scripts can be used to interface the MATLAB/Simulink file with DIgSILENT. The names are optional, however; the declared parameters have to be exactly the same as those that have been declared in the damping controller block definition in PowerFactory (as shown in Fig. 13.6e).

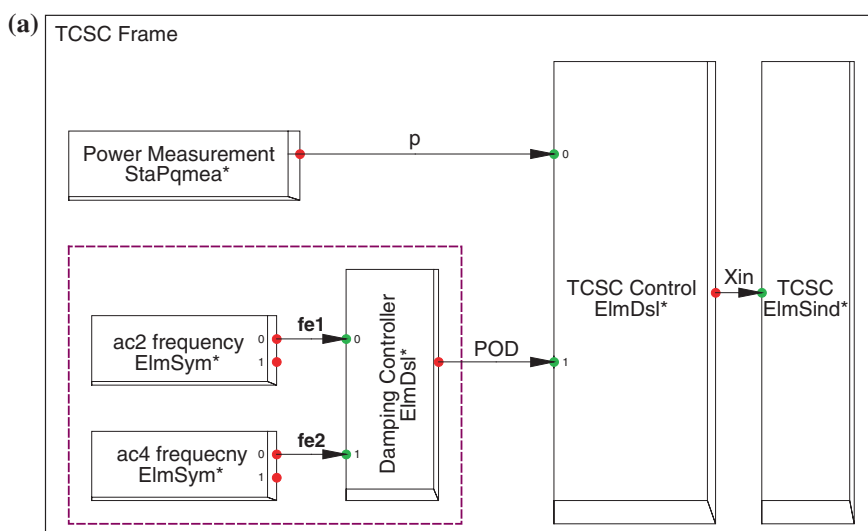


Fig. 13.6 Structure of interface for design of supplementary PSS-based POD. **a** Structure of composite TCSC frame including added slots for POD. **b** POD input signals—G2 and G4 frequency. **c** Defined composite model for the TCSC frame. **d** Damping controller parameters value. **e** Damping controller block definition (basic data page). **f** Damping controller block definition (equations page)

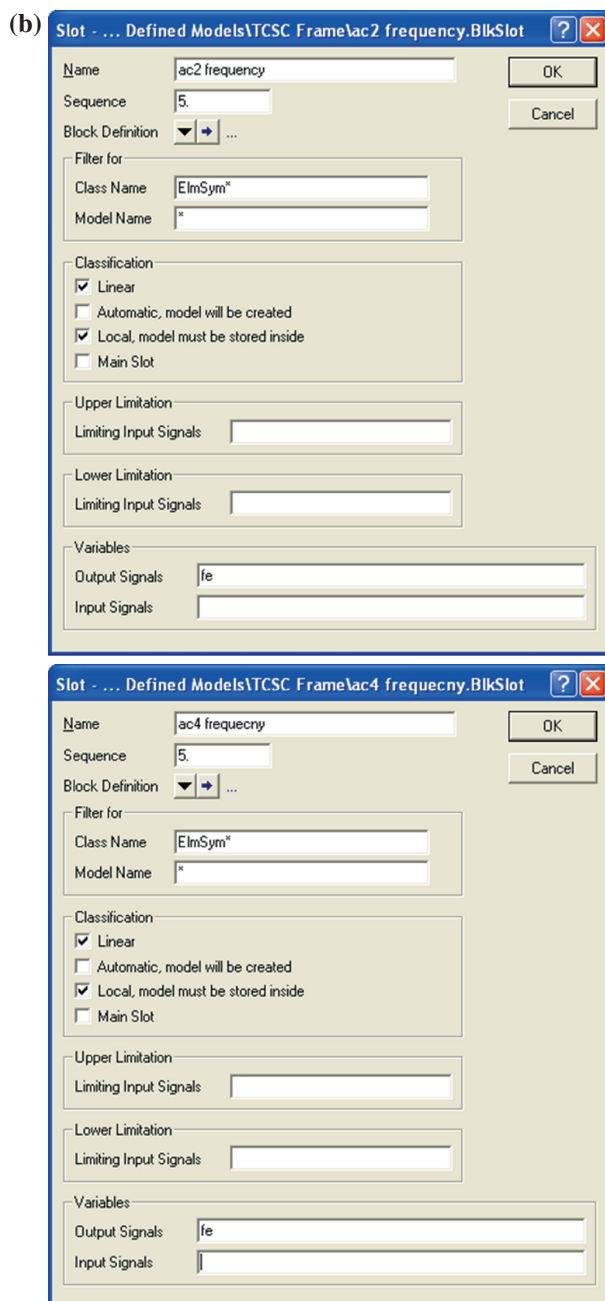


Fig. 13.6 (continued)

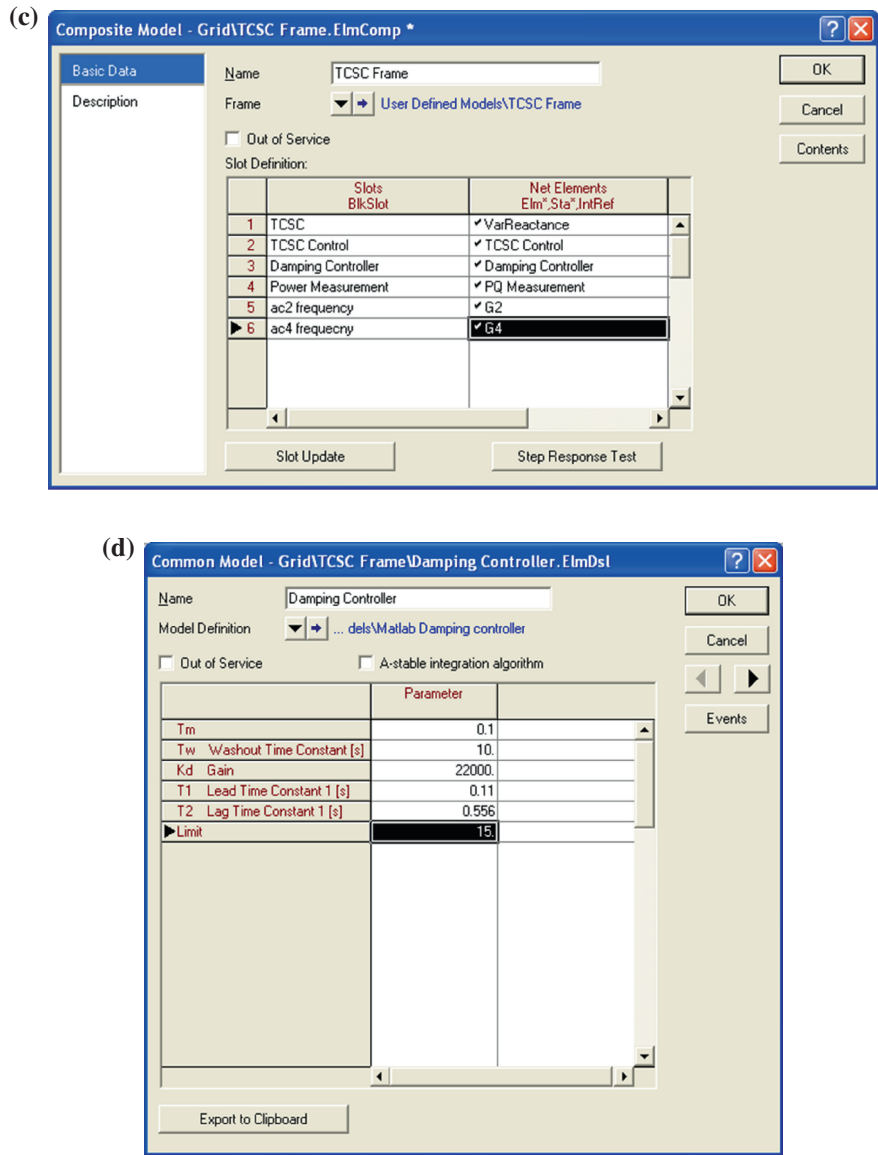


Fig. 13.6 (continued)

(e) Block Definition - User Defined Models\Matlab Damping Controller.BlkD

Basic Data	Name	Matlab Damping Controller	OK
Equations	Title	Signal Definition and Initial Conditions of Matlab Model	Cancel
Description	Caution: Changing level of already used models requires adaptation of all models		
	Level	Level 3: Level 2 + lin()-function precise in time	Contents
	<input type="checkbox"/> Automatic Calculation of Initial Conditions		
	Classification		
	<input checked="" type="checkbox"/> Linear		
	<input type="checkbox"/> Macro		
	<input checked="" type="checkbox"/> Matlab M-file C:\Matlabsim final TCSC\final no simulink\book ...		
	Upper Limitation		
	Limiting Parameter		
	Limiting Input Signals		
Lower Limitation			
Limiting Parameter			
Limiting Input Signals			
Variables			Check
Output Signals			delta_X
Input Signals			fe1,fe2
State Variables			x1,x2,x3
Parameter			Tm,Tw,Kd,T1,T2,Limit
Internal Variables			
			Check Inc.
			Pack
			Pack>Macro
			Encrypt

(f)

```

!Initial condition
inc (fe1) =1      ! Frequency initial condition
inc (fe2) =1      ! Frequency initial condition
inc (x1) =0       ! State variable initial condition
inc (x2) =0       ! State variable initial condition
inc (x3) =0       ! State variable initial condition
!The following equations are required for initial condition calculation
delta_ X=0
x1. =0
x2. =0
x3. =0
!Definition of parameters
vardef (Tw) ='s'; ' Washout Time Constant'
vardef (Kd) ='Gain'
vardef (T1) ='s'; ' Lead Time Constant'
vardef (T2) ='s'; ' Lag Time Constant'

```

Fig. 13.6 (continued)


```
function [t, x, y] = POD
global Tm Tw Kd T1 T2 Limit fe1 fe2 x1 x2 x3

% Get the simulation parameters
options = simget('POD_model');

% Set the updated initial conditions
options = simset('InitialState', [x1,x2,x3]);

% Run the simulation and update the output and states value
[t, x, y] = sim('POD_model', [], options);
```

In each simulation time step, PowerFactory updates the values of the defined parameters (if they are time variable) and state variables. Line 4 of the above code initializes the Simulink file with the values of the state variables taken from the previous time step. Line 5 of the code starts the simulation with the present initial values and stores the new values for time, state variables and outputs.

The designed power oscillation damping controller within the Simulink environment has been shown in Fig. 13.7. In order to appropriately interface with the controller, all parameters have to be identical to those declared in the written script. For every simulation time step, an updated value for the parameters is transferred to the MATLAB/Simulink file. It is important to set the MATLAB simulation time step equal to or smaller than the set time step in DIGSILENT. In the present example, the simulation time step of 0.01 s is selected for both files. If the simulation time is smaller, then its reciprocal must be an integer number. As an example, the Simulink time step can be 0.005, 0.0025 and 0.002.

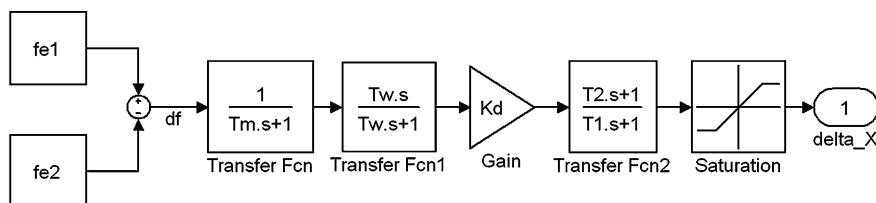


Fig. 13.7 Power oscillation damping block diagram

13.5.5 Interfacing PowerFactory and MATLAB for the Design of POD Using Script (Second Approach)

The previous section explained how to link a Simulink model/file with DIgSILENT PowerFactory. An example of power oscillation damping was used to demonstrate the interfacing procedure. In some cases, where the required components are not available in Simulink, it is necessary for the user to develop their own controller in the form of a script. Additionally, handling the state variables become more complicated as the size of the Simulink file increases and is sometimes impossible. As an example, despite the availability of a model predictive controller component in Simulink, since the controller cannot be loaded when the Simulink file is called, it cannot be executed from DIgSILENT. The only solution in similar scenarios is to use a script for developing the controller and linking to DIgSILENT. A basic knowledge of writing and executing scripts is necessary to proceed with this section.

MATLAB scripts provide a platform for the user to develop and execute the user's own tailored codes in the MATLAB language. The entire library of MATLAB functions can be employed which enables the user to create any desirable function. This section provides a moderately simple case study for the design and implementation of a POD controller using such a script. The implemented controller is essentially the same as that which was introduced in the previous Sect. 14.5.4. Instead of the Simulink model, a MATLAB script is used. The whole script can be seen at the end of the section.

All the steps to set up the connection between MATLAB and DIgSILENT PowerFactory are essentially similar as explained before. In this example, all the calculations are provided within a MATLAB script and no Simulink file is used. The only change here is in the controller block definition page (*BlkDef*) where the *mFilePOD.m* file has to be selected and linked to PowerFactory instead.

The *mFilePOD* function transfers time, state variables and output values to PowerFactory. Similar to the previous section, all defined variables in PowerFactory have to be globally declared. The POD controller is modelled in state space matrices instead of a block diagram. The represented A, B and C matrices are the state space model representation of the block diagram shown in Fig. 13.7. The users may derive the equations themselves. The time matrix (*t*) must have at least two elements starting with zero and ending with a value equivalent to the PowerFactory time step. The output (*y*) and state variable (*x*) matrices need to have a dedicated row for each element of the time matrix. Section two of the code determines the state matrix and updates the state values using the transient state space equations. It is important to note that matrices *x*, *y* and *t* must have a certain size and format. The last section of the code is written to limit the produced output similar to limiter block.

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [t, x, y] = mFilePOD
global Tm Tw Kd T1 T2 Limit fe1 fe2 x1 x2 x3

t = [0;0.01]; % Simulation time steps
dt = t(2) - t(1); % DeltaT definition
u = fe1 - fe2; % Frequency difference as the input
X(:,1) = [x1;x2;x3]; % States value from the previous step

%% State space matrices of the controller (A, B and C)
A = [ -1/Tm 0 0
      -Kd/Tm -1/Tw 0
      -T2*Kd/(T1*Tm) (Tw-T2)/(T1*Tw) -1/T1];

B = [1/Tm; Kd/Tm; T2*Kd/(T1*Tm)];
C = [0 0 1];

X(:,2) = dt*(A*X(:,1) + B*u) + X(:,1); % Updated states
y(1,1) = C*X(:,1); % Updated output
y(2,1) = C*X(:,2); % Updated output
x = X'; % To match DigSILENT format

%% Output saturation (To limit the output y)
for i=1:size(y)
    if (y(i) > Limit)
        y(i) = Limit;
    elseif (y(i) < -1*Limit)
        y(i) = -1*Limit;
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

13.5.6 Comparative Simulation Results on Performance of PSS-Based POD Controller

The results provided in Fig. 13.8 present the effect of the power oscillation damping controller on the response of G2 rotor angle and speed (more results can be seen in the provided DigSILENT PowerFactory file). The performance of a supplementary POD controller for TCSC is observed following the occurrence of a severe disturbance such as a three-phase fault. It is evident that the controller is able to improve the systems power oscillation damping following a severe disturbance and stabilize the system within the reasonable settling time. In addition, the results demonstrate that the performance of both POD controllers designed using Simulink model and m.file are identical.

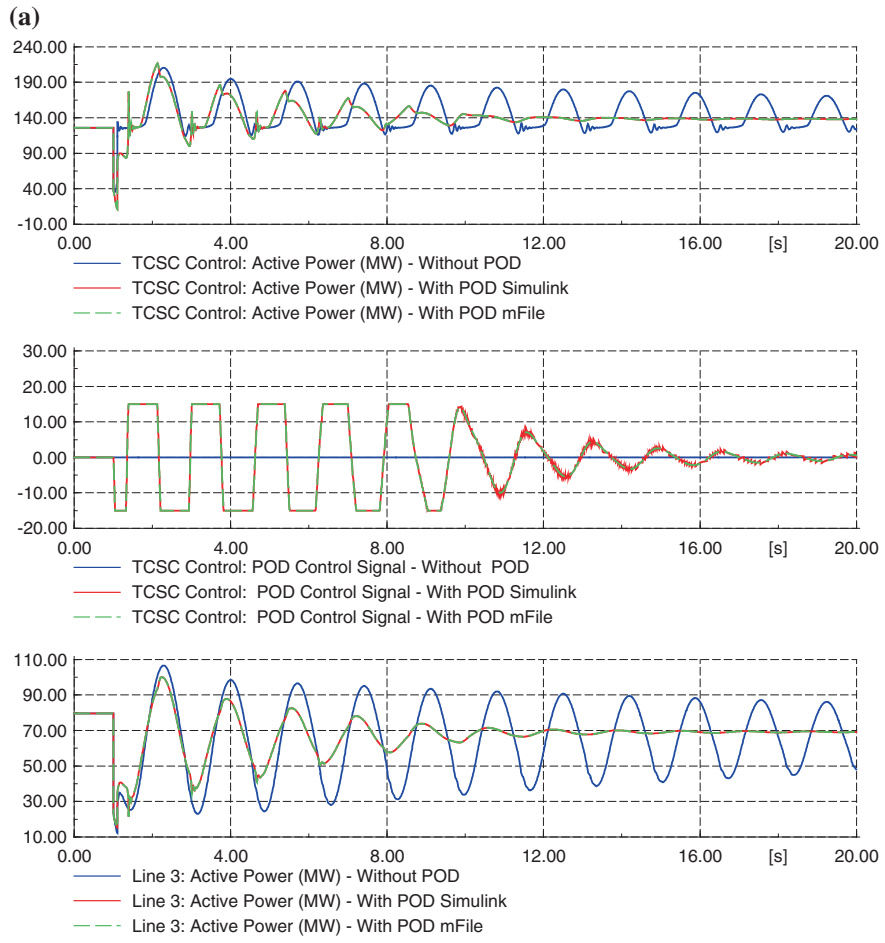


Fig. 13.8 The effect of the damping controller on the stability of the system. **a** Active power on TCSC and line 3 and control signal with/without damping controller signal. **b** Generators rotor angle oscillation with/without POD signal

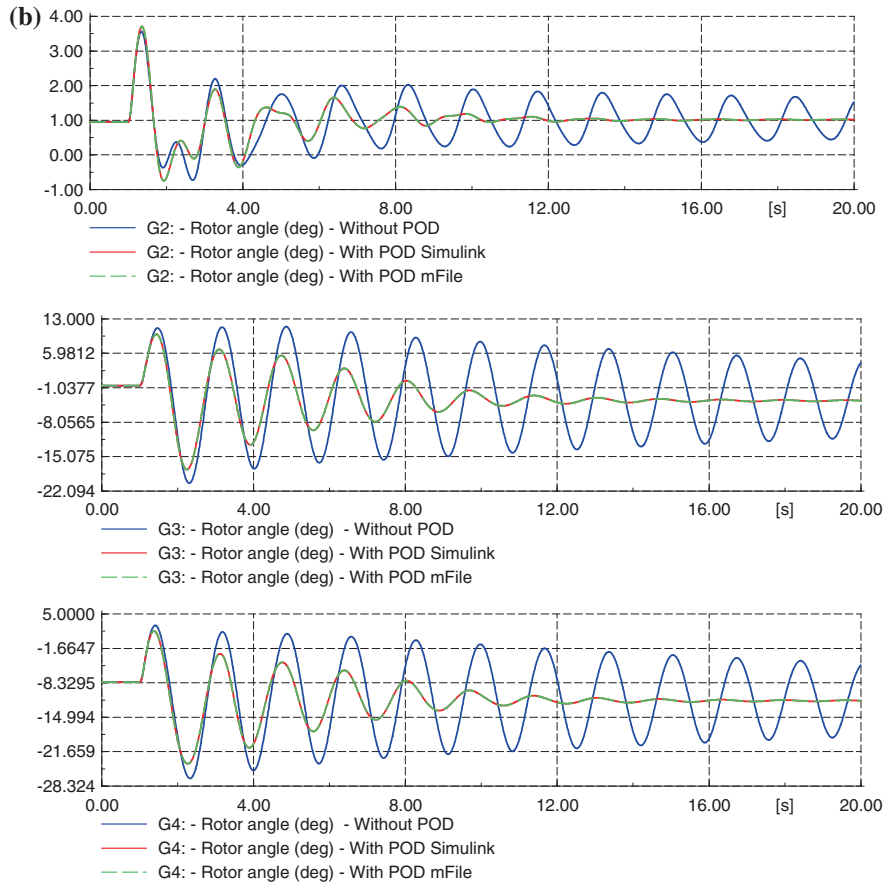


Fig. 13.8 (continued)

13.6 Conclusion

MATLAB/Simulink offers a large control toolbox and a wide range of computerised algorithms. Thus, when one is investigating a new control technique, for power system, either continuous or discrete, it is often more convenient to use MATLAB. Access to MATLAB from the PowerFactory simulation tool is achieved by utilising the available interface capability in both tools. Also, when a power system has been modelled in PowerFactory and a controller is required to perform a number of comprehensive mathematical calculations, then there is a need to interface PowerFactory with the necessary control tool boxes capable of performing such calculations. Therefore, interfacing to the MATLAB program is essential for advanced controller designs that require a certain level of mathematical calculation capability.

Therefore, the description of the structure and process of interfacing between the DlgSILENT PowerFactory and MATLAB/Simulink programs are presented in this chapter using two different study cases.

Acknowledgments The authors would like to thank National Grid Plc, for their support. Also, they would like to gratefully acknowledge the contribution of Dr. Vandad Hamidi in providing some of the simulation models.

References

1. Filizadeh S, Heidari M, Mehrizi-Sani A, Jatskevich J, Martinez JA (2008) Techniques for interfacing electromagnetic transient simulation programs with general mathematical tools IEEE taskforce on interfacing techniques for simulation tools. *IEEE Trans Power Delivery* 23 (4):2610–2622
2. Gole AM, Daneshpooy A (1997) Towards open systems: A PSCAD/EMTDC to MATLAB interface, submitted to IPST '97
3. MATLAB Toolbox Release (2009) The MathWorks, Inc., Natick, Massachusetts. Available <http://www.mathworks.co.uk> (Online)
4. DlgSILENT PowerFactory, Version14.0.522. Available <http://www.digsilent.com> (Online)
5. Hashmani AA (2010) Damping of electromechanical oscillations in power systems using wide area control. PhD thesis, University of Duisburg-Essen
6. Pipelzadeh Y, Chaudhuri B, Green TC (2013) Control coordination within a VSC HVDC link for power oscillation damping: a robust decentralized approach using homotopy. *IEEE Trans Control Syst Technol* 21(4):1270–1279
7. Cai D (2012) Wide area monitoring, protection and control in the future great Britain power system. PhD thesis, University of Manchester
8. Preece R, Milanovic J, Almutairi A, Marjanovic O (2013) Damping of inter-area oscillations in mixed AC/DC networks using WAMS based supplementary controller. *IEEE Trans Power Syst* 28(2):1160–1169

Chapter 14

Advanced Applications of DPL: Simulation Automation and Management of Results

Matthias Stifter, Serdar Kadam and Benoît Bletterie

Abstract In this chapter, some principles of simulation automation are explained, which are helpful for investigating a large number of simulation scenarios. Parameters of data models often need to be changed for a certain range or results need to be post-processed. It is demonstrated how PowerFactory's built-in scripting language is used to automatically assign external profiles (characteristics) and how script parameters can be altered and the simulation controlled externally in batch-style programs. An example project shows the implementation of a local voltage controller by utilising the scripting language. Simulation parameters are taken from Microsoft Excel for parametric studies and results are stored back into the spreadsheet.

Keywords DIgSILENT programming language • DPL • Local voltage control • Reactive power drop control Q(U) • Simulation automation • Simulation scripting

14.1 Introduction

With the *DIgSILENT Programming Language* (DPL), a C-style scripting language exists, which allows to control PowerFactory. A manual and reference is provided in the help section. The advantages of a scripting language are the simple syntax and the versatile usage of commands and functions. A major disadvantage though is the lack of a debugger to analyse the script execution and state of variables. The typical application tasks for DPL scripts can be classified into the following:

Electronic supplementary material The online version of this chapter (doi:10.1007/978-3-319-12958-7_14) contains supplementary material, which is available to authorized users.

M. Stifter (✉) · S. Kadam · B. Bletterie
Energy Department, AIT Austrian Institute of Technology, Giefinggasse 2,
1210 Vienna, Austria
e-mail: matthias.stifter@ait.ac.at

- **Automation of simulation:** starting, repeating and stopping
- **Implementation of controls:** changing the system behaviour
- **Analysis of results:** evaluation and visualisation

Examples of this are as follows:

- Repeatedly execution of steady-state power flow analysis with varying, dynamic profiles
- Short-circuit analysis and, e.g. compute the thermal equivalent short-circuit current for different protection tripping time
- Modal analysis for different network topologies with, e.g. changes in controller settings
- Start transient simulations with different parameters for sensitivity analysis

This chapter will introduce the DPL scripting capability for simulation automation by a demonstration of automated assignment of load characteristics. This has to be done normally manually and is for large networks very time-consuming, error prone and tedious. After that, an excursion to use automation functionality from external applications are undertaken by introducing the *Remote Communication Interface* (RCOM). Since PowerFactory version 15.1, it is possible to include Python scripts and control PowerFactory from outside via the Python API Wrapper.

In the example project, an inverter-based reactive power droop control to perform voltage control will be implemented in DPL. Various simulation scenarios of the control settings can be parameterised in Microsoft Excel, where the results of the simulation are then written back to a table.

14.2 Simulation Automation

Typical simulation scenarios may include the modelling of different generation and load situation in the network (e.g. winter and summer).

Built-in mechanisms in PowerFactory enable the assignment of specific characteristics ('profiles') to various variables, such as the active and reactive power of loads. When managing a decent size of a network model, this assignment of characteristics can be time-consuming, tedious and error prone. Within DPL, this can be easily achieved and used generically for all sorts of controlling individual behaviour of variables by characteristic. Figure 14.1 shows the typical tasks for conducting power-flow-based system analysis and the listing in Fig. 14.2 the execution of the corresponding subscripts. All scripts can be found in the accompanying project under: *Library* → *Scripts* → *RunExtDataTimeSweep*.

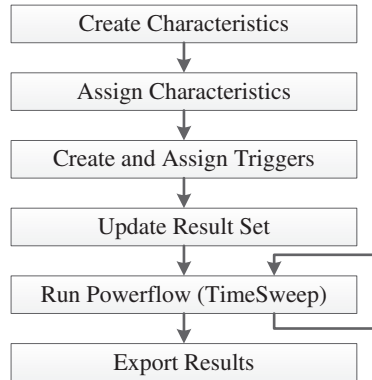


Fig. 14.1 Typical tasks for power flow based analysis with characteristics

```

! ### RunExtDataTimeSweep ###
ResetCalculation();
CreateCharacteristics.Execute();      ! create characteristics
AssignCharacteristics.Execute();      ! link CSV columns
CreateAndAssignTrigger.Execute();     ! create and assign triggers
UpdateResultSet.Execute();            ! update loads in result set
TimeSweep.Execute();                 ! simulate every step
Export.Execute();                     ! save results to CSV
  
```

Fig. 14.2 DPL script for running simulations with external characteristics

14.2.1 Creating and Assigning of External Characteristics

The external characteristic objects have to be created and assigned to the specific load object variables. The listing in Fig. 14.3 shows the steps of first deleting all existing and creating new characteristics for the active and reactive power of all loads.

After creating the external file characteristics for every load, the next step is to automatically assign the external file, respectively the columns of the comma separated value (.csv) file to the right characteristic.

Since the format of the characteristic file is without any header or information for the various columns, an accompanying file *loadList*—a list of the loads to assign—is provided as a parameter of the script. This has the advantage that the references between loads and characteristics are fully controllable from extern: the alphabetically ordered name of loads in every line in the *loadList* corresponds to the columns in the characteristic. In case of balanced models, for every entry in the *loadList*, two columns for active and reactive power will be assigned. The listing in Fig. 14.4 shows this DPL script.

```

! ### CreateCharacteristics ###
object load, 0;                                ! variables for objects
set loadSet, S;                                ! variables for sets
int n;                                          ! number of loads

S = AllRelevant('ChaVecFile');                 ! get all external file characteristics
for(0=S.First();0=S.Next()){                  ! iterate over all characteristics
    Delete(0);                                ! delete all characteristics
}

loadSet = AllRelevant('ElmLod');               ! get all loads into set loadSet
for(load=loadSet.First();load;load=loadSet.Next()){ ! iterate over all loads
    load:i_sym=0;                             ! set to symmetric
    load:plini=0;                             ! set active power to zero
    load:qlini=0;                             ! set reactive power to zero
}

load = loadSet.First();                       ! first load in set
while (load) {                                ! while load in set
    load.CreateObject('ChaVecFile', 'plini'); ! active power characteristic
    load.CreateObject('ChaVecFile', 'qlini'); ! reactive power characteristic
    printf('P,Q characteristic created for load: %s',load:loc_name);
    load = loadSet.Next();                    ! next load in set
}
n = loadSet.Count();                          ! number of loads
printf('P,Q characteristics created: Number of loads: %i', n);

```

Fig. 14.3 DPL script for creating characteristics for active power of loads

14.2.2 Creating and Assigning of Triggers

Since every characteristic needs a trigger, which corresponds to the actual position in the associated scale of the characteristic, a trigger has also to be created and assigned automatically. The subscript *CreateAndAssignTrigger* in the project deletes any existing triggers, creates a new one and assigns it to all characteristics in the project.

14.2.3 Updating Variables of the Result Set

For running a series of power flow calculation, we need also to define the variables of the objects we want to store for later analysis. This is usually done by adding Variable Selection Objects (*IntMon*) to a result set (default result set 'All calculations'). In Fig. 14.5, the DPL command is depicted, showing the reference to the external result set 'Results' (*ElmRes*). Automatically adding the variables of interest (voltage and active power) with DPL is shown in listing Fig. 14.6.

14.2.4 Running the Simulation

After assigning the external characteristics, triggers and results, the simulation can be executed. The probably well-known and by the standard, global library provided

```
! ### AssignCharacteristics ###
set S;                                ! set holds all characteristics
object O;                             ! entry of set (characteristic object)
object load;                          ! load object
int mismatch;                         ! result of string compare
string listEntry,loadName;           ! names
int m;                                ! modulo operation result
int col;                              ! current column to assign to characteristic
col=1;                               ! start with the first column

fopen(loadList,'r',0);                ! open file of sorted load list

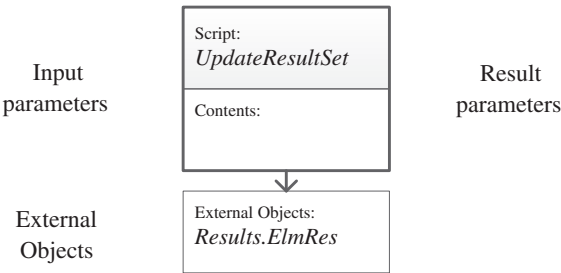
! select all characteristics and sort them alphabetically
S = AllRelevant('*.ChaVecfile');      ! all external file characteristics
S.SortToVar(0,'fold_id','loc_name');  ! sort to load name

for(0=S.First();0;0=S.Next()) ! iterate over all characteristics
{
  load=0:fold_id;                    ! assigned load object
  loadName=load:loc_name;             ! load object's name
  m = modulo(col, 2);                ! read only every second characteristic
  if (m = 1){
    fscanf(0,'%s',listEntry); ! read load name from load list
  }
  mismatch=strcmp(listEntry,loadName); ! compare to load list

  ! assign active power and reactive power column respective
  if(mismatch=0){
    printf(' %s:%s (column: %d)', 0:loc_name, loadName, col);
    O:icol = col;                  ! column number
    O:afac = 0.001;                ! multiplicative factor from kW (csv) to internal MW
    O:bfac = 0;                    ! additive factor
    O:f_name = loadFile;           ! file name of assigned csv file
    O:iopt_sep=0;                  ! use system separators
    O:col_Sep = ',';               ! column separation of csv file
    O:dec_Sep = '.';               ! decimal character of csv file
    O:usage = 2;                   ! 2=absolute
    O:approx = 1;                  ! interpolation
    col = col + 1;
  }
  else {
    col=col+2;                     ! skip the next 2 columns
    printf(' Load %s not found in list!', loadName);
  }
}
fclose(0);                          ! close file
```

Fig. 14.4 DPL script for automatic assignment of external characteristics to loads

Fig. 14.5 DPL command
UpdateResultSet



```
! ### UpdateResultSet ###
set S;                                ! result set
object 0;                             ! object in result set

S = Res.GetContents('* ',0);           ! Get all elements of result set
while (0) {                           ! While there are objects in the set
    Delete(0);                         ! Delete all Objects in Result Set
    0 = S.Next();                      ! Get next object
}

S = AllRelevant('*.ElmLod');          ! Get all loads
for(0=S.First();0;0=S.Next()){
    Res.AddVars(0,'n:U:bus1','m:P:bus1'); ! Specify voltage and power
    printf('  %s added to Results',0:loc_name);
}
```

Fig. 14.6 DPL script for adding result variables to the result set

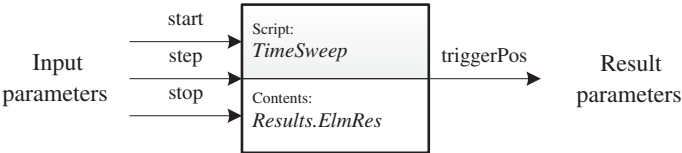


Fig. 14.7 DPL command *TimeSweep*

DPL script *TimeSweep* (*Library* → *Scripts* → *TimeSweep*) provides the mechanism to advance the assigned file trigger to the next line in the characteristic, running the power flow calculation and storing the results. In Fig. 14.7, the input and result parameters, as well as the content (local folder) of the DPL command are depicted. The listing in Fig. 14.8 shows a minimal version of that script.

The result of the simulation can be seen in Fig. 14.9 or in the project at the page *Active Power* where the characteristics of the various loads of the network are shown.

14.3 Batch Simulations via RCOM

Until PowerFactory Version 14.x, the file folder *ENGINE* provided the so-called RCOM interface for sending control commands via remote procedure call. Since version 15.1 RCOM is no longer officially supported and it is recommended to use the Python/API functionality for this purpose. Although no longer supported, one can still copy and paste the *ENGINE* folder from previous version and use the provided *digrcom* executable.

The following listing in Fig. 14.10 shows the use of the RCOM interface for batch processing of various simulation scenarios by setting parameters of the external sources of the characteristic files and starting the DPL script *RunExtDataTimeSweep*.

```

! ### TimeSweep ###
object Ldf;                ! power flow object
object Trigger;            ! Trigger object

Ldf = GetCaseObject('ComLdf');
Trigger = GetCaseObject('*.SetTrigger');

EchoOff();                ! deactivate output
triggerPos = start;        ! start parameter
while ( triggerPos <= stop )
{
  Trigger.ftrigger = triggerPos; ! assign trigger
  Ldf.Execute();              ! power flow
  Results.WriteDraw();        ! store results
  triggerPos += step;         ! increase trigger
  printf('Simulation of step: %d', triggerPos);
}
EchoOn();                 ! activate output

```

Fig. 14.8 DPL script for running power flow with characteristics

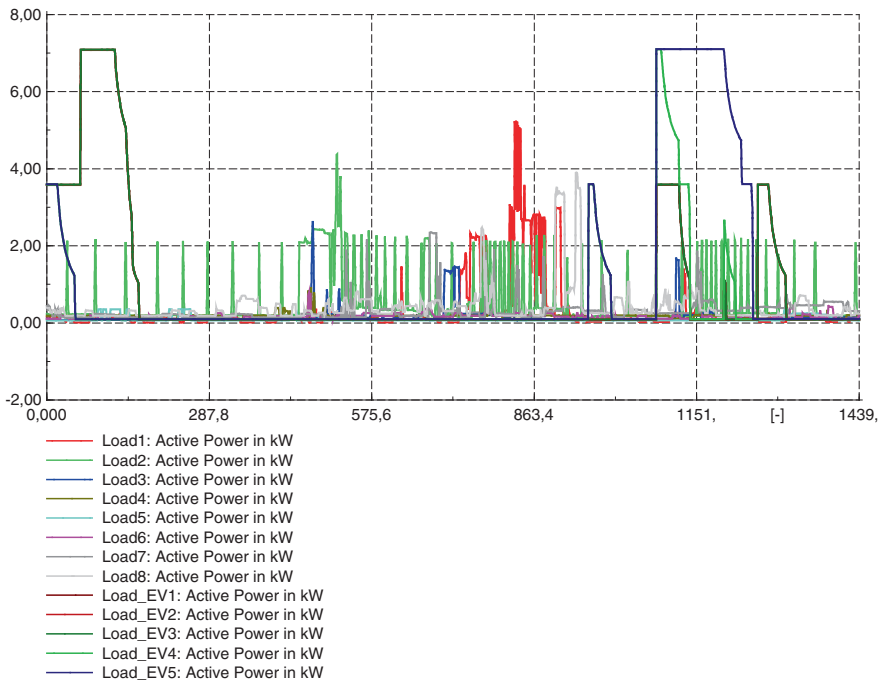


Fig. 14.9 Profiles of the external active power characteristics

The content of this RCOM command file is the parameter for the command:

```
digrcom -d -p ncacn_ip_tcp -n 127.0.0.1 -e 2001 -f="automation.cmd"
```

```
ac \user\project
cd \user\project\Library\Scripts\RunExtDataTimeSweep
man/set obj=AssignCharacteristics.ComDpl variable=IntExpr:0 value=A:\data\loadList.txt
man/set obj=AssignCharacteristics.ComDpl variable=IntExpr:1 value=A:\data\profiles.csv
man/set obj=TimeSweep.ComDpl variable=IntExpr:0 value=1
man/set obj=TimeSweep.ComDpl variable=IntExpr:1 value=1
man/set obj=TimeSweep.ComDpl variable=IntExpr:2 value=1440
cd \user\project\Study Cases\Study Case\
RunExtDataTimeSweep
res/file/all/loc/data f_name=A:\data\result.txt
```

Fig. 14.10 RCOM command file for batch control of PowerFactory

14.4 Example Project

14.4.1 Description

The following project demonstrates a reactive power control of photovoltaic system (PV) to control the voltage at the point of common coupling in a low-voltage distribution network. The typical low-voltage network consists of several loads and PV systems positioned in radial feeders. The PV inverters are capable of adjusting the reactive power output in order to control locally the voltage as required by some grid codes [1, 2]. Among various options, such as using a power factor as a function of the injected active power, the so-called $Q(U)$ characteristics have gained increasing attention in the last years [3–7]. In order to investigate the performance of the different implementations of a $Q(U)$ -control, suitable models and control strategies are needed [8].

The objective of the control implemented in DPL is to find the steady-state solution for the considered network with the inverters controlling the voltage by reactive power consumption of injection. Additionally, active power curtailment ($P(U)$ -control) is used to guarantee a maximal voltage rise by reduction of active power feed in.

In order to solve the problem, an iterative approach is needed (fixed point iteration) since the controller output depends on the voltage ($Q(U)$) and the voltage depends on the reactive power output. Of course, this type of controller can be implemented with the DIGSILENT simulation language (DSL) but requires more efforts since a block definition (*BlkDef*) must be defined for each controller. While this block definition can also be implemented automatically (with DPL), the approach followed in this example is to implement a very simple algorithm. This allows to find the steady-state solution with little configuration efforts: controller settings can be automatically changed for all PV inverter models, new inverters can be easily defined and will be automatically considered by the control, inverters without control can also be defined (e.g. for inverters which don't support reactive power output).

The input parameters for various scenarios are taken from a spreadsheet. After the operation point of the generators have converged the maximum, minimum and total active and reactive power are evaluated and written as a result to the spreadsheet.

14.4.2 Preparation

In order to have the right references to the external measurement data files, the batch file `configure.bat` has to be executed. It will map the network drive A: to the data folder contained in that folder automatically. The batch file can be started from any location (CD or hard drive) to reference the actual folder to the one referenced in the PowerFactory project. All external profiles in the example project are mapped to A: drive, e.g. `A:\data\LoadList.txt`.

14.4.3 Control Characteristic

A typical voltage control characteristic is depicted in Fig. 14.11, where a dead band is defined at the nominal voltage. The slopes for overvoltage (II.) and undervoltage (IV.) are characterised by the voltages $U_{DB,pos}^Q$, $U_{DB,neg}^Q$ —where the control starts to operate—and the maximum reactive power $Q_{max,ind}$ and $Q_{max,cap}$. These reactive

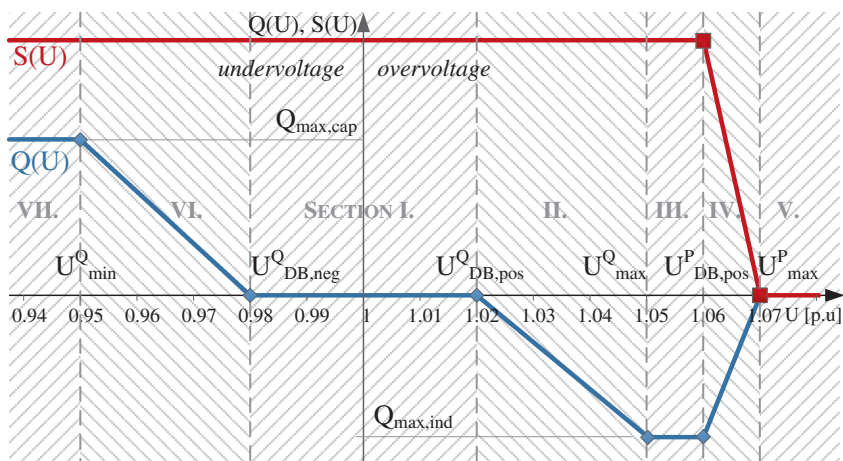


Fig. 14.11 Reactive power characteristic to support local voltage control

power limits have to be determined by the maximum apparent power S_{nom} , and the minimum allowed power factor PF_{min} defined as the cosine of the angle between active power and the apparent power:

$$\text{PF}_{\text{min}} = \cos \varphi = \frac{P_{\text{max}}}{S_{\text{nom}}} \quad (14.1)$$

In the following, the sections are explained in detail:

Dead Band

- I. The $Q(U)$ control characteristic has an area where only active power is provided (dead-band zone).

Overvoltage

- II. If the voltage increases over a certain limit, the reactive power is provided according to the given droop with the equation:

$$Q(U) = k_{\text{overvoltage}}^Q (U - U_{\text{DB,pos}}^Q) \quad (14.2)$$

$$k_{\text{overvoltage}}^Q = - \frac{Q_{\text{max,ind}}}{(U_{\text{max}}^Q - U_{\text{DB,pos}}^Q)} \quad (14.3)$$

$$Q_{\text{max,ind}} = S_{\text{nom}} \sin(\arccos(\text{PF}_{\text{min}})) \quad (14.4)$$

where $k_{\text{overvoltage}}^Q$ is the slope of the $U(Q)$ characteristic in section II in Fig. 14.11.

- III. Maximum negative reactive power: The maximum reactive power is limited when the maximum power factor of the generator is reached to

$$Q_{\text{max,ind}} = - \sin(\arccos(\text{PF}_{\text{min}})) S_{\text{nom}} \quad (14.5)$$

- IV. Maximum voltage limit: If the voltage increases over a certain limit the active power will constantly be reduced ($P(U)$ -control) to counteract a further increase to the point where the total power output is zero. Note that due to the minimal power factor supported by the inverter the reactive power has to be reduced simultaneously.

$$P = k^P U - d^P \quad (14.6)$$

$$k^P = -\frac{S_{\text{nom}} PF_{\text{min}}}{(U_{\text{max}}^P - U_{\text{DB,pos}}^P)} \quad (14.7)$$

$$d^P = -k^P U_{\text{max}}^P \quad (14.8)$$

V. The power output will be set to zero, when the voltage is higher than U_{max}^P .

Undervoltage

- VI. To support the voltage reactive power is produced according to the given gradient of the droop, which can be determined similar to the equations of the overvoltage droop region in II.
- VII. Maximum positive reactive power: The reactive power is limited due to the nominal power factor of the generator:

$$Q_{\text{max,cap}} = \sin(\arccos(PF_{\text{min}})) S_{\text{nom}} \quad (14.9)$$

14.4.4 Control Algorithm

To converge to the steady-state operation point, the DPL script executes a power flow to determine the reactive power for every inverter and a controller evaluation iteratively until the change of the reactive power set point is below a certain epsilon threshold (fix-point iteration). The algorithm is a simple way of finding the steady-state solution of a network with inverters with $Q(U)$ and $P(U)$ control and is very easy to use. It does not claim to be efficient in terms of convergence. Since the iteration starts with determining the individual reactive power support with respect to the local voltage level, the contributions are fairly distributed among the ones which are causing the violation, which is proportional to the local voltage situation.

Figure 14.12 shows the flow diagram for the voltage control algorithm according to the voltage-dependent reactive power characteristic (Fig. 14.11):

1. Power flow is performed to calculate the voltages at the nodes
2. For each generator, the position within the voltage characteristic is determined
3. According to the section on the voltage characteristic, the respective action is performed.

For example, if the voltage is within the section II in Fig. 14.11, ‘ $Q(U)$ droop inductive’: the reactive power set point is determined to the appropriate point on the slope, according to the gradient of the characteristic.

4. Check apparent power constraints: After the set points of the reactive (and active) power are calculated, they are checked if within the given maximum apparent power of the power diagram.
5. Check power factor constraints: The operational limits are checked.

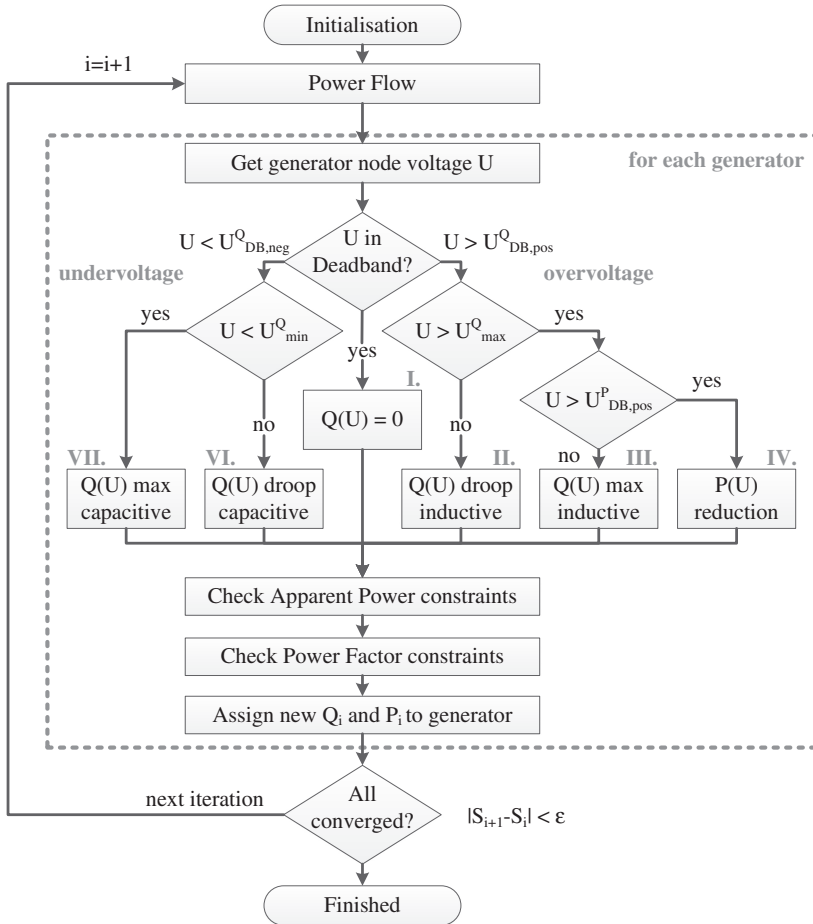


Fig. 14.12 Algorithm for the voltage depended reactive power control $Q(U)$ with droop characteristics and dead band

6. The set points are assigned to the generator.
7. Convergence: Check whether the changes of the generator set points are within the termination criteria. Repeat if not converged with new set point.

The detailed implementation of the algorithm can be found in Sect. 14.4.6.

14.4.5 Run Voltage Control with Input Parameters from Excel

Microsoft Excel tables can be accessed from within DPL scripts for providing input parameters for an automated simulation. In the example project, various settings for

Table 14.1 Example input parameters for voltage control scenarios (Excel)

Gen.	Q(U) characteristics				P(U) characteristic		Generator ratings	
P_{act} (MW)	U_{min}^Q (V)	$U_{\text{DB,neg}}^Q$ (V)	$U_{\text{DB,pos}}^Q$ (V)	U_{max}^Q (V)	$U_{\text{DB,pos}}^Q$ (V)	U_{max}^{QP} (V)	PF_{min} (rad)	S_{nom} (MVA)
0.0005	0.95	0.98	1.02	1.05	1.06	1.07	0.85	0.001
0.0007	0.95	0.98	1.02	1.05	1.06	1.07	0.85	0.001
0.0008	0.95	0.98	1.02	1.05	1.06	1.07	0.85	0.001
0.001	0.95	0.98	1.02	1.05	1.06	1.07	0.85	0.001

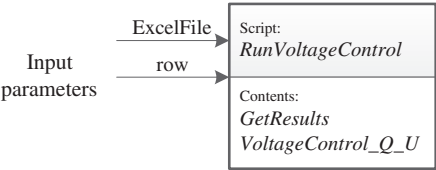


Fig. 14.13 DPL command *RunVoltageControl*

the $Q(U)$ characteristic as well as parameters for the nominal rating powers of the generators are used. The DPL script *RunVoltageControl* reads the rows of the excel worksheet and executes the voltage control script. Simulation results are written back into the table. Table 14.1 shows the input parameter and the settings for the $Q(U)$ characteristics. Figure 14.13 shows the DPL script with its input parameters and the listing in Fig. 14.14 the main parts of the DPL code.

14.4.6 Voltage Control with Reactive Power from PV Inverters

In the DPL script *VoltageControl_Q_U* (Fig. 14.15) the set points of the generator’s reactive and, respectively, active power are determined according to the given characteristics. In the initialisation part, the generator parameters are assigned from the input parameters. The generator output P is stored in a map and retrieved at the start of the iteration. After the initial power flow the generator node voltages are also stored in a map, because assigning new set points to the network model will reset those results (change of calculation relevant objects). Both constraints—maximal nominal apparent power limits and minimal power factor—are checked, after the potential $Q(U)$ operation point has been determined. The change of the set points Q and P between successive iterations is the termination criteria *DeltaS*. In case the algorithm does not converge and alternates between two values (which can happen in weak networks with a high amount of PV generation), it is forced into the convergence with a damping factor **alpha**. The Listing is shown in Fig. 14.16.

```

! ### RunVoltageControl (part) ###
error = xlStart();           ! Start MS Excel
error = xlOpenWorkbook(ExcelFile); ! Open file
error = xlActivateWorksheet(1); ! Activate first sheet

! Start from row declared in input parameter list
while(1){
    ! Read control parameters
    xlGetValue(1, row, P);           ! Read first column
    if(P = 0)                       ! Check for empty row
        break;                     ! Stop iterating on rows
    xlGetValue(2, row, Uqmin);       ! Read values
    ! ... read the input parameters for VoltageControl_Q_U

    ! Start voltage control

VoltageControl_Q_U.Execute(P,Uqmin,Uqdbneg,Uqdbpos,Uqmax,Updbpos,Upmax,PFmin,Snom,DeltaS,alpha,MaxIter);

    ! Get results
    GetResults.Execute();           ! Calculate results
    ! Write Results to excel file
    xlSetValue(13, row, GetResults:Psum);
    ! ... write the results to excel spreadsheet

    ! next scenario
    row += 1;
}

! Save and exit
error = xlSaveWorkbook();
xlTerminate(); ! Terminate MS Excel

```

Fig. 14.14 DPL script for running voltage control with external parameters

In the script *CheckApparentPowerConstraints* the reactive power set point is determined according to the actual voltage U on the characteristic. To ensure that the maximal nominal apparent power rating of the generator is guaranteed, in case the new operation point is outside the power capability curve of the generator, the constraints are checked according to:

$$P = \sqrt{S_{\text{nom}}^2 - Q^2} \quad (14.10)$$

Figure 14.17 shows the input parameter for the script *CheckApparentPowerConstraints*. The input parameter P is modified inside the script and changes are automatically reflected in the calling higher-level script *VoltageControl_Q_U*.

In the script *CheckPowerFactor* the reactive power set point has to be within the operational limits given by the minimal power factor of the generator. The script *CheckPowerFactor* takes care of this by constraining Q , if it exceeds the limits, to:

$$Q = \frac{Q}{\text{abs}(Q)} P \tan(\arccos(\text{PFmin})) \quad (14.11)$$

```

! ### VoltageControl_Q_U (part) ###
do {
    converged=1
    MyLdf.Execute();

    ! Load flow results become invalid when changing network model
    for (PV_obj=PV_set.First();PV_obj;PV_obj=PV_set.Next())
        map_U.Insert(PV_obj:loc_name, PV_obj:m:bus1); ! store in map

    ! Calculate P,Q for each generator
    for (PV_obj=PV_set.First();PV_obj;PV_obj=PV_set.Next()){
        P_old = PV_obj:pgini;           ! store previous
        Q_old = PV_obj:qgini;           ! store previous
        U = map_U.GetValue(PV_obj:loc_name); ! get voltage from map
        P = map_P.GetValue(PV_obj:loc_name); ! Start with deadband
        Q=0;                             ! Start with deadband

        ! === overvoltage ===
        else if(U>Uqdbpos .and. U<Uqmax)      ! II. Q(U) droop ind.
            Q = k_Q_ov*(U-Uqdbpos);

        else if(U>=Uqmax .and. U<Uqdbpos)      ! III. Q(U) max ind.
            Q = -sin(acos(PFmin))*Snom;

        else if({U>=Uqdbpos}.and.{U<=Uqmax}){ ! IV. P(U) droop
            P_set = k_P*U-d_P;
            if(P_set<P)
                P = P_set;
            Q = -tan(acos(PFmin))*P;
        }

        else if(U>Uqmax)                       ! V. P(U) cut-off
            Q = 0; P = 0;

        ! === undervoltage
        else if(U<Uqdbneg .and. U>Uqmin)      ! VI. Q(U) droop cap.
            Q = k_Q_uv*(U-Uqdbneg);

        else if(U<=Uqmin)                     ! VII. Q(U) max cap.
            Q = sin(acos(PFmin))*Snom;

        CheckApparentPowerConstraints.Execute(Snom,P,Q);
        CheckPowerFactor.Execute(P,Q,PFmin);
        PV_obj:pgini = P;                     ! assign to generator
        PV_obj:qgini = Q;                     ! assign to generator

        ! Check if changes in setpoints < epsilon (DeltaS)
        if({sqrt(sqr(Q_old-Q)+sqr(P_old-P)) < DeltaS})
            converged = converged .and. 1;    ! check if all converged
        else
            converged = 0;
    }
    i_iter+=1; ! Iteration counter
}
while( .not. converged);

```

Fig. 14.15 DPL script (part) for reactive power control Q(U)

where the first quotient determines whether it is positive (capacitive) or negative (inductive) Fig. 14.18. Figure 14.19 shows the input for the script.

Figure 14.20 shows how the power factor is limited for the reactive power set point. Note that according to the control algorithm, the maximum set point is between the limits of $Q_{\max,\text{ind}}$ and $Q_{\max,\text{cap}}$.

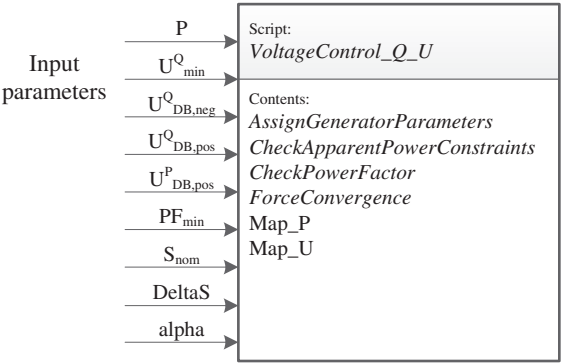


Fig. 14.16 DPL command *VoltageControl_Q_U*

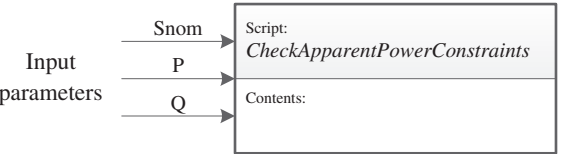


Fig. 14.17 DPL script for running voltage control with external parameters

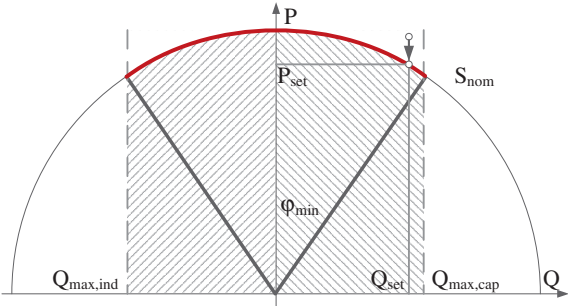


Fig. 14.18 DDPL command *VoltageControl_Q_U*

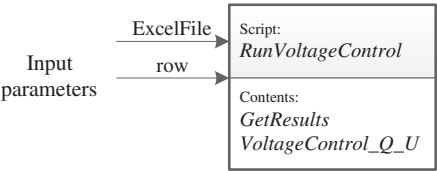


Fig. 14.19 DPL command for checking the power factor limit of the generator

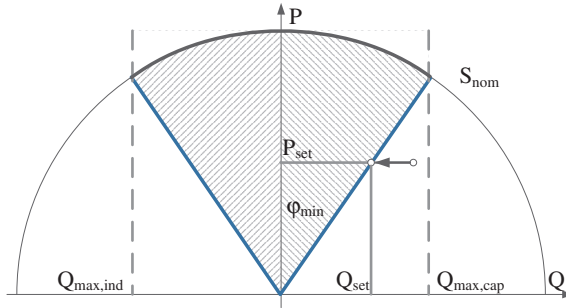


Fig. 14.20 Generator capability curve: power factor constraints

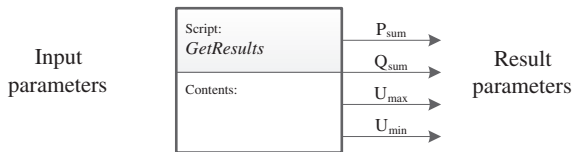


Fig. 14.21 DPL command for simulation results analysis

14.4.7 Analysing Simulation Results

As an example, the sum of the active and reactive power of all generators and the maximum and minimum voltages are evaluated. Figure 14.21 shows the result parameters of the script and Fig. 14.22 shows the DPL code. The script *GetResults*

```
! ### GetResults ###
set sGens, sTerms; ! set of generators and terminals
object oGen,oTerm; ! generator and terminals objects

sGens = AllRelevant('*.ElmGenStat'); ! get set of all generators
sTerms = AllRelevant('*.ElmTerm'); ! get set of all terminals

!Get highest and lowest voltage in network
sTerms.SortToVar(1,'m:u'); !Sort terminals by highest voltage
oTerm = sTerms.First(); !Get first terminal in list
umax = oTerm:m:u; !Get voltage of terminal
sTerms.SortToVar(0,'m:u'); !Sort terminals by lowest voltage
oTerm = sTerms.First(); !Get first terminal in list
umin = oTerm:m:u; !Get voltage of terminal

!Sum active and reactive power
Psum = 0; !Set Ptot to zero
Qsum = 0; !Set Qtot to zero
for(oGen = sGens.First();oGen;oGen = sGens.Next()){
    Psum = Psum + oGen:pgini; !Cumulate active power
    Qsum = Qsum + oGen:qgini; !Cumulate reactive power
}
```

Fig. 14.22 DPL script for simulation result analysis

Table 14.2 Example results for the voltage control scenarios (Excel)

Input Parameter	Results			
Gen.	All generators		Network	
P_{act} [MW]	P_{tot} [MW]	Q_{tot} [Mvar]	U_{max} [p.u.]	U_{min} [p.u.]
0.0005	0.0045	−0.00279	1.04717963	0.99999664
0.0007	0.0054	−0.003304	1.04715858	0.99999667
0.0008	0.0054	−0.003348	1.0471555	0.99999667
0.001	0.0063	−0.003589	1.04715356	0.99999672

is called every time after the *Voltage_Control_Q_U* script. For every row of input parameters from the external Excel file, the maximum and minimum voltage are obtained, and the results of the total achieved active and reactive power are summed up and then exported to the result columns (see Fig. 14.22) (Table 14.2).

14.5 Conclusions

Automation of simulation tasks by means of scripting with DPL (and RCOM) has been presented. The example demonstrated the implementation of a control algorithm used in multiple models and the use of iterative convergence to a steady state of the system in DPL. Depending on the requirements of the simulation and analysis task, the right way of implementation in terms of efficiency and other criteria, such as time saving, needs to be chosen. Sometimes a combination of the introduced concepts may be the most suitable.

14.5.1 Outlook

With the introduction of Python in PowerFactory 15.1.1, the API functionalities can be used in a scripting environment, as opposed to compile C++ code and run PowerFactory within an application. Moreover, the possibility to connect a debugger to it will boost the workflow for users and programmers.

References

1. E-Control (2013) TOR Part D4, Technical and organizational rules for network operators and users—part D4; Parallel operation of generators connected to the distribution network
2. Technical Guideline Generating Plants Connected to the Medium-Voltage Network (2008), BDEW Bundesverband der Energie- und Wasserwirtschaft e.V., Jun. 2008
3. Fawzy T, Premm D, Bletterie B, Goršek A (2011) Active contribution of PV inverters to voltage control—from a smart grid vision to full-scale implementation. *e & i Elektrotechnik und Informationstechnik* 128(4):110–115

4. Bletterie B, Stojanovic A, Kadam S, Laus G, Heidl M, Winter C, Hanek D, Pamer A, Abart A (2013) Local voltage control by PV inverters: first operating experience from simulation, laboratory tests and field tests presented at the 27th European Photovoltaic Solar Energy Conference and Exhibition (EU PVSEC). Paris, pp. 4574–4581
5. Mende D, Schwarz J, Schmidt S, Premm D, Sakschewski V, Pfalzgraf M, Homeyer H, Schmiesing J, Brantl J (2013) Reactive power control of PV plants to increase the grid hosting capacity presented at the 27th European Photovoltaic Solar Energy Conference and Exhibition (EU PVSEC). Paris, pp 4225 - 4230
6. Stetz T, Kraiczy M, Braun M, Schmidt S (2013) Technical and economical assessment of voltage control strategies in distribution grids. *Prog Photovoltaics Res Appl* 21(6):1292–1307
7. Stetz T, Marten F, Braun M (2013) Improved low voltage grid-integration of photovoltaic systems in germany. *IEEE Trans Sustain Energy* 4(2):534–542
8. Einfalt A, Zeilinger F, Schwalbe R, Bletterie B, Kadam S (2013) Controlling active low voltage distribution grids with minimum efforts on costs and engineering. In *IECON 2013—39th annual conference of the IEEE Industrial Electronics Society*, pp 7456–7461

Chapter 15

Interfacing PowerFactory: Co-simulation, Real-Time Simulation and Controller Hardware-in-the-Loop Applications

Matthias Stifter, Filip Andrén, Roman Schwalbe
and Werner Tremmel

Abstract Various possibilities exist to interface the power system model and simulation engine of PowerFactory. These interfaces open applications for co-simulation, real-time simulation for controller hardware-in-the-loop (HIL) set-ups or automation and control of simulation. This chapter gives a comprehensive overview of the various interfaces PowerFactory provides—namely to MATLAB, external C++ functions, OPC, RCOM, API and DGS—as well as typical applications for their use. An example for connecting to an external controller for automated voltage control of an on load tap changer using common TCP/IP connections is elaborated in detail.

Keywords PowerFactory interfaces · Co-simulation · MATLAB · DSL/DLL · RCOM · Power system analysis · Smart grids

15.1 Introduction

The coupling with other applications is needed for instance to handle the increasing complexity of power system applications interacting with other domains like information and communication technologies. Another motivation could be for re-using existing validated models of other simulation applications or components which are real, such as protection devices or other controls.

Electronic supplementary material The online version of this chapter (doi:10.1007/978-3-319-12958-7_15) contains supplementary material, which is available to authorized users.

M. Stifter (✉) · F. Andrén · R. Schwalbe · W. Tremmel
Energy Department, AIT Austrian Institute of Technology, Giefinggasse 2, 1210 Vienna,
Austria
e-mail: matthias.stifter@ait.ac.at

The different possibilities which exist to couple PowerFactory with other models and simulators are explained in this chapter and examples of applications will be discussed in detail. These include among other topics:

- **MATLAB:** PowerFactory build-in interface (DSL) for co-simulation
- **DLL:** Using external DLL in DSL components (e.g.: TCP/IP sockets) and DPL scripts
- **OPC:** Industrial standard interface—OPC client—in use with multi-agent systems and controller hardware in the loop
- **RCOM:** Remote communication—remote procedure call interface for using PowerFactory in engine mode (e.g. automated simulation)
- **API:** Direct control of PowerFactory internal data model and advanced functionality (e.g. co-simulation)
- **DGS:** file format for exchanging data models and geographical information

An overview of the interface mechanism provided by PowerFactory is given in Fig. 15.1.

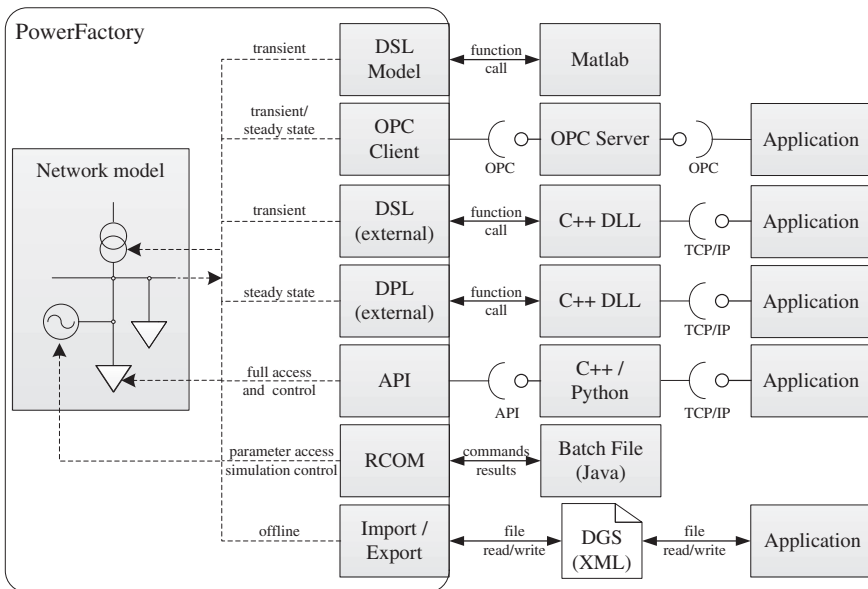


Fig. 15.1 Overview of interfaces provided by PowerFactory

15.2 Co-simulation

15.2.1 Motivation for Co-simulation

Coupled simulation is motivated when external models or controls need to be interfaced with the simulation in PowerFactory. Examples are controls developed in other applications (e.g. MATLAB/Simulink, Visual Studio) which may be deployed to production systems or embedded systems. Also coupling simulation applications with models from other physical domains, like thermal systems, are conceivable.

15.2.2 Real Time

When controllers or other hardware are connected to simulated environments for development, testing and validation, constraints regarding the time for solving a simulation step are imposed on the hardware-in-the-loop (HIL) set-up. PowerFactory is able to run dynamic simulation in real-time, synchronised to a clock (e.g. PC clock). Depending on the application, controller-HIL requires simulation steps typically in the timescale from milliseconds up to minutes. For the interaction with amplifiers in power-HIL set-ups, hard time constraints in the order of microseconds to milliseconds have to be fulfilled. This can be achieved by dedicated real-time simulators.

15.3 Interfaces for Co-simulation

This chapter introduces the interfacing mechanisms provided by PowerFactory.

15.3.1 MATLAB/Simulink

PowerFactory provides a direct interface to MATLAB, which can be used as a DSL model. The basic idea is to call a MATLAB function instead of executing the model or equations usually defined in a DSL block. Figure 15.2 shows the main functionality.

In order to execute, the MATLAB function PowerFactory opens an instance of MATLAB, wherein the actual MATLAB function (M-File) is executed. PowerFactory uses global variables to communicate with MATLAB. Therefore, the MATLAB function does not have any input parameters. Thus, the first thing the MATLAB function should do is to define the global variables used by PowerFactory. This is seen in Fig. 15.2. The remaining part of the MATLAB function represents the actual behaviour of the DSL model. Applications could be to

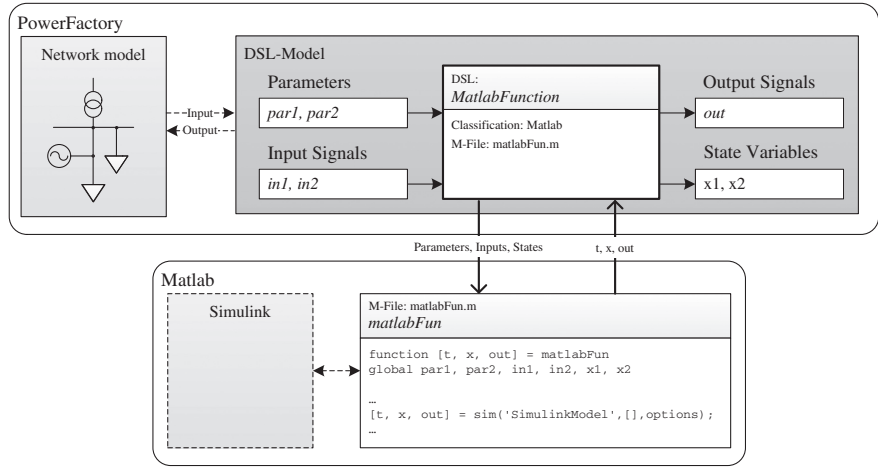


Fig. 15.2 MATLAB interface in PowerFactory

simulate a Simulink model or execute an optimisation algorithm in pure MATLAB code. Once the MATLAB function finishes, three output parameters must be set and returned to PowerFactory. These are the timestamps, the states variables and the outputs signals for the DSL model.

The interface to MATLAB/Simulink provides a rather simple way to outsource control code and algorithms into MATLAB. However, the interface has some issues to consider: (i) the mixed use of global variables and function parameters is confusing and difficult to set up, (ii) the interface is rather slow since an instance of MATLAB is running alongside PowerFactory and (iii) the MATLAB function is executed anew in each simulation step of PowerFactory. Since PowerFactory executes the MATLAB function anew in each simulation step, internal states of the function need to be saved in MATLAB between each execution. This may lead to problems, if, for example, a Simulink model is simulated in MATLAB. A co-simulation setting with a detailed battery modelled in Simulink has been published in [1].

15.3.2 DSL Functions Linked to External DLLs

External DSL functions written in C/C++ can be included into DSL blocks in two different ways, respective two different external dynamic linking libraries (.dll). Figure 15.3 illustrates the access to 'event triggered external functions' in *digexdyn.dll* and the access by 'function call' to *digexfun.dll*.

The libraries are placed in the PowerFactory installation folder and automatically loaded (only) at start-up. Note that multiple libraries can be provided if they use the *digexdyn*.dll* and *digexfun*.dll* prefix format.

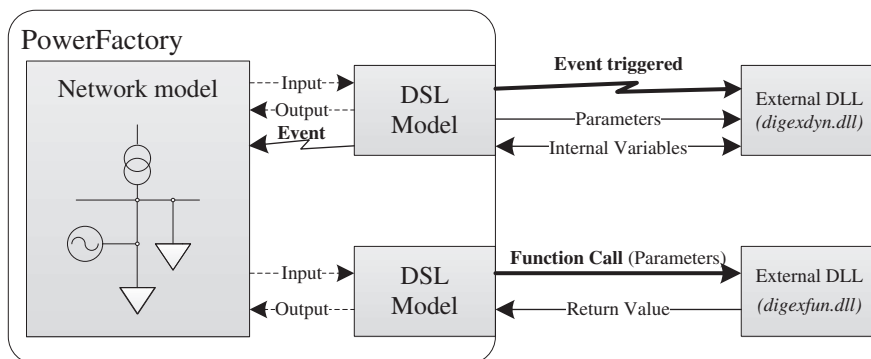


Fig. 15.3 Implementation of external event driven function and function call in external dynamic linking library

Documentation regarding the external event driven C/C++ function can be requested from DigSILENT support. An example for the user-defined DSL function can be found in the installation directory of PowerFactory under the folder *digexfun*.

15.3.3 DPL Functions Linked to External DLLs

Similar to the external implemented DSL functions for dynamic/transient analysis, the DPL interpreter can be extended by user-defined functions to be used for steady-state analysis (see technical documentation: External DPL Functions). The functions are placed in *digdplfun*.dll* in the installation folder, where multiple libraries are possible.

The function call has to provide arguments of the following signature: list of arguments passed to the function, flag for 'checking only' (DPL syntax check), result of the function and status (success and type of result value).

A simple example which checks if the reactive power is within the allowed power factor limits is shown in the listing in Fig. 15.4.

In combination with a steady-state simulation (e.g. a time series or '*timesweep*'), the DPL function call can be blocking, waiting for the external application to return values as an input for the ongoing simulation. This is important in a co-simulation set-up when the simulation step has to be synchronised with the external application.

15.3.4 OPC—OLE for Process Communication

OPC server connects client processes which run normally in real time such as automation and control components. If the value of a data point changes at the OPC server, its value is propagated to the subscribed OPC clients. Depending on the

```
// The DPL function CheckPowerFactor limits Q to the minimum allowed power factor.
DIGDPLFUN_API void CheckPowerFactor(DplArgList* argList, int checkonly, ExtProcRes
&result, ProcResTp &status)
{
    status = ISDBL;           // set return type to string
    if (checkonly){           // doing the argument check here
        if (argList->nArg != 3) // three arguments are expected
            status = ISARGERR;
        return;
    }

    double p = argList->Arg[0].d; // active power
    double q = argList->Arg[1].d; // reactive power
    double pfmin = argList->Arg[2].d; // power factor limit
    if (abs(q)>tan(acos(pfmin))*p){ // limit if below limit
        result.d=q/abs(q)*tan(acos(pfmin))*p;
    }
    else
        result.d=q;
}
```

Fig. 15.4 External DPL function for checking the allowed power factor

implementation, there is a minimum cycle time of the server which starts from milliseconds up to seconds. Since OPC is intended for automation applications, the connected processes run normally asynchronous with system time and time steps in simulation applications are not synchronised.

External data link (*ComLink*) is a built-in OPC client. In the object's properties, the *Link to* option '*OPC TDS*' can be used to communicate with an OPC server¹ in transient/dynamic simulation and the '*OPC OSE*' for communication within DPL. While using the '*OPC TDS*' link in RMS simulation, the data variables of the 'External DAT Measurements' (*StaExtDatmea*) are updated automatically according to the 'Dead-band' option of the data link object. Changes below this threshold are ignored. In contrast to this, receiving and sending needs to be explicitly executed when using the '*OPC OSE*' link. In the property *Prog ID*, the exact name of the server's instance has to be provided (e.g. '*Matrikon.OPC.Universal.1*'). Note that multi-threading has to be enabled in *Configuration* → *Advanced* → *Tab: Advanced* → *Enable Multi-Threading*. Object 'External DAT measurements' represent the OPC data point in PowerFactory and can be linked to object variables (*Tab: Post-processing* → *Object/Variable Name*). They also can be set and read individually with DPL.

If applications are coupled for co-simulation via OPC, a synchronisation mechanism can be used to sync the simulation time step between the simulation components. The listing in Fig. 15.6 shows a mechanism for synchronising the simulation with an external application. The OPC interface has been extensively used to develop and validate a coordinated voltage controller [2–4] as well as in combination with multi-agent systems to validate intelligent charging algorithms [5, 6].

¹ Matrikon provides a free OPC server, which has been used in this example. The 32-bit version cannot be used with 64-bit version of PowerFactory.

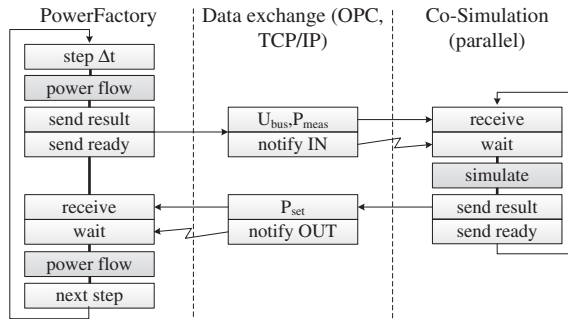


Fig. 15.5 Synchronisation mechanism between applications using ‘notifies’ [7]

Figure 15.5 shows the principle of the synchronisation between two applications when they have to wait on each other. When the system clock is used for synchronisation, real-time capabilities are necessary.

The DPL code in Fig. 15.6 shows the usage of the OPC link object. The script *LDFCharacteristicsOPC* can be found in the project under *Study Case* → *Study Case OPC*. If the script is started, it waits for the OPC data point *SIMULATOR_NOTIFICATION_IN* to be incremented to advance the simulation by one step. It performs the power flow operation and increments the *SIMULATOR_NOTIFICATION_OUT* to indicate that the simulation step has been performed. The script can be tested with an OPC client, by incrementing the value of the data point *SIMULATOR_NOTIFICATION_IN*. For demonstration purposes, also one file characteristic is set for *Load1*. The characteristic’s file trigger *TriFile* is passed as the parameter *oTriFile* to the script (see *Edit* → *Basic Options*). Other characteristics can be added to model, for example other loads.

Many applications and systems support the industry de facto standard of OPC connectivity, reaching from protection and automation devices to software applications. A drawback of the OPC interface may be the slow exchange of data, which has a minimum update rate of several milliseconds.

15.3.5 RCOM—Remote Procedure Call

Until version 14.x, the so-called *ENGINE* folder was part of the installation. Via the included *digrcom* application, a number of commands have been available to control the simulation from outside. It is still possible to use that interface in version 15.x, but it is expected that this interface will be replaced in future by the Python interface. RCOM is well suited for simulation automation since it is possible to change for instance DPL script parameters or other model attributes. Therefore, it is possible to use this interface in combination with an external application for a coupled simulation. It is also possible to use this remote procedure call interface to connect to an instance running on a remote machine. To enable RCOM, the line *rcom/start* (and

```

! ### LDFCharacteristicsOPC ###

int nDataNum;                ! number of received data objects
int status,status_new, status_OPC; ! status variables
int sleepTime;               ! time to check for new values
sleepTime=1;                 ! in milliseconds

status_OPC = Link:isLinkStarted; ! check if running otherwise start
if(status_OPC = 0)
{
    Info('Link is not started. Starting...');
    Link.Execute();
    Sleep(2000); ! wait for communication layer
    status_OPC = Link:isLinkStarted;

    if(status_OPC = 0)
    {
        Error('Link could not be started');
        exit();
    }
    Info('Link successfully started.');
```

```

}

SIMULATOR_NOTIFICATION_IN.InitTmp(); ! initialise data objects
SIMULATOR_NOTIFICATION_OUT.InitTmp(); ! initialise data objects

nDataNum = Link.ReceiveData(1); ! receive OPC data
SIMULATOR_NOTIFICATION_IN.GetMeaValue(status); ! read value
status_new=status; ! store status

for( dTriVal = TriValMin ; dTriVal <= TriValMax ; dTriVal += TriValDelta )
{
    oTriFile:ftrigger = dTriVal; ! advance file trigger
    EchoOff();
    Ldf.Execute(); ! power flow
    EchoOn();

    !Communication Part between PowerFactory and OPC server
    !=====Send Values to OPC
    oTriPFready:ftrigger=status; ! remember trigger
    SIMULATOR_NOTIFICATION_OUT.SetMeaValue(status); ! write value
    nDataNum = Link.SendData(); ! send OCP data
    if(nDataNum = 0) { ! check if successful
        CloseOPC.Execute(renewOPC); exit(); ! otherwise close and exit
    }
    !=====Ask every sleepTime for OPC-Updates
    do
    {
        Sleep(sleepTime); ! sleep between the checks
        nDataNum = Link.ReceiveData(); ! receive OPC data
        if(nDataNum>0){ ! number of changed data objects
            SIMULATOR_NOTIFICATION_IN.GetMeaValue(status_new); ! read value
        }
    }while(status=status_new) ! status value not changed
    !===== all data have been received...
    status=status_new; ! advance / increment status
    oTriPFready:ftrigger = status_new; ! OPC updates only when trigger is set
    ResetCalculation(); ! reset calculations
}
CloseOPC.Execute(renewOPC); ! close OPC

```

Fig. 15.6 DPL script for synchronising simulation execution via OPC

```

List<String> command = new ArrayList<String>();
command.add("C:\\DigSILENT\\pf140\\ENGINE\\digrcm"); // RCOM
command.add("-d");
command.add("-p");
command.add("ncacn_ip_tcp");
command.add("-n");
command.add("127.0.0.1");
command.add("-e");
command.add("2001");

List<String> args = new LinkedList<String>(); // arguments
args.add("ac \\user\\project\\n"); // activate project
args.add("cd \\user\\project\\Network Model\\Network Data\\n"); // change folder
args.add("man/que/one obj=Load1.ElmLod variable=plinir\\n"); // set variable
ProcessBuilder pb = new ProcessBuilder(command); // shell command process
pb.directory(new File("C:\\DigSILENT\\pf150\\Engine")); // path
java.lang.Process process = null;
try {
    process = pb.start(); // execute
} catch (IOException ex) { }
OutputStream os = process.getOutputStream(); // input for RCOM
final InputStream is = process.getInputStream(); // get results while sending
PrintWriter pw = new PrintWriter(new BufferedWriter(new OutputStreamWriter(os)));
for (String arg : args) {
    pw.println(arg); // send arguments
}

```

Fig. 15.7 Java code snippet for connecting via RCOM

rcom/erhoff) has to be added to the *Configuration* → *Advanced* → *Startup Commands*. The listing in Fig. 15.7 sketches how to start and pass arguments to the RCOM interface from Java. This interface has been utilised, for example in [8].

15.3.6 API—Application Programming Interface

Basically, everything you can do in PowerFactory can be done via the API. It exposes internal model and objects of the network data as well as analysis functions and results to be dynamically linked into any C++ application environment. In engine mode, the graphical user interface is hidden. Figure 15.8 illustrates this concept. The classes provided by the API are as follows: the *API* instance itself, the *Application* instance, the output window and the *DataObject* as well as *Values*.

Analysis functions like power flow or dynamic simulation (RMS) can be started via the API. It has to be mentioned that the setting of calculation relevant objects

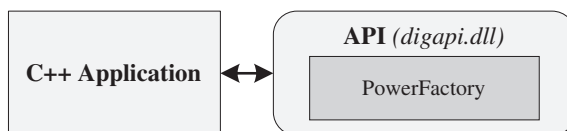


Fig. 15.8 Use of the API in a stand-alone C++ application

during a transient simulation is not possible, thus the simulation has to be stopped and initial conditions recalculated (or a combination of API and *digexdyn.dll/digexfun.dll* can be used, see Sect. 15.4).

15.3.7 API Python

Since version 15.1, a Python wrapper for the API has been provided. Since Python is an interpreter based programming language no compiler is necessary to use the C++ API. Python scripts can be directly used within PowerFactory or used for running the application in engine mode and controlling it from an external application. The code in Fig. 15.9 shows an example of a script executed directly within PowerFactory.

15.3.8 DGS

DGS stands for *DigSILENT Interface for Geographical Information Systems*. It is a file format to exchange data with other data sources, e.g. geographical information systems (GIS) and other simulation applications. The format can be pure ASCII based, but also XML or even in Microsoft Excel or Access format. It is possible to export and import full network models together with element type definitions and graphic representations. Additionally, results can be exported or load characteristics imported. It is possible to use this format to make changes in the network (e.g. breaker and switch position) and perform simulation and analysis automated from external applications (e.g. via RCOM commands).

```

1 if __name__ == '__main__':
2     #connect to PowerFactory
3     import powerfactory as pf
4     app = pf.GetApplication()
5     if app is None:
6         raise Exception("getting PowerFactory application failed")
7
8     #print to PowerFactory output window
9     app.PrintInfo("Hello World!")
10
11     #get active project
12     prj = app.GetActiveProject()
13     if prj is None:
14         raise Exception("no project activated")

```

Fig. 15.9 Python code for activating and accessing PowerFactory

Table 15.1 Comparison of interfaces for coupled simulation [7]

Analysis function	MATLAB	DLL	OPC	RCOM	API, Python	API+DLL	DGS
Steady state		(×) ^c	×	×	×	×	
Dynamic RMS	×	×	(×) ^{a,b}	×	(×) ^c	(×) ^d	
Transient EMT	×	×	—	—	(×) ^c	(×) ^d	
Simulation control				×	×	×	

^a Timing constraints apply, when it comes to real-time simulation

^b Synchronisation mechanism (simulation master) necessary when running parallel applications

^c Changing variables is not possible during RMS simulation

^d Multi-threading and queued synchronisation of signals are required to have full read/write access

^e Using external defined DPL function in blocking mode for synchronising with other applications

15.3.9 Summary Interfaces for Co-simulation

The overview makes no claim to be complete with respect to applications or possible ways of interaction. The simplest way of exchange—to write to and read from file system—has not mentioned here, since it is not considered to be any option regarding to performance and flexibility. An analysis of the different synchronisation and latency issues introduced with co-simulation has been given by the authors in [7]. Table 15.1 summarises the possibilities to use a specific interface for the required analysis function.

Efforts have been undertaken to standardise the way time continuous simulators can exchange models and which can be used in a co-simulation set-up by the so-called Functional Mock-up Interface (FMI). The authors have investigated and used the FMI for co-simulation in various set-ups [9]. Developments for implantation of a PowerFactory FMI interface (both for model exchange and co-simulation) are ongoing. Results are expected to be released under open source.

15.4 Customised External DSL Functions

Externally implemented DSL functions (event driven and user defined) provide versatile possibilities for extensions of DSL. But one has to bear in mind that DSL works only in RMS/EMS dynamic/transient simulation. Two mechanisms exists, which will be explained briefly and afterwards in more detail. Working in DSL blocks means that all available network and model variables have to be wired in the composite model prior to the simulation. Arbitrary signals cannot be accessed the way the API provides them.

15.4.1 External Event Driven C/C++ Function

Event triggered external function. The first mechanism is based on an event call with a specific function name. The externally implemented method is invoked with

this specific DSL block variables passed as arguments (Fig. 15.10). Essentially, these are as follows: simulation time, parameters (including parameters and internal variables) and output variables (constant). DSL block input variables are not passed. The implemented method is expected to set up a string which represents a DSL event call (up to 5 events). This event is emitted when the external function is left again. Since the parameters, in particular, the internal variables are changeable; this can be used to pass any number of results from the external function back to the DSL block. DSL syntax is provided in Fig. 15.11.

Some issues have to be considered when using this feature: the parameter array includes the parameter and the internal variables defined for this specific DSL block in alphabetical order. Therefore, the order must be known to access the right variable in the parameter list. Practically, it is possible to assign input variables to internal parameters and access them at the external function. To return results from the external function back to the DSL function, the results of the external function are assigned to the passed parameter list within the DLL, and then, the internal parameters are assigned to the DSL output variables. Recall that the list of parameters is in alphabetical order; therefore, using prefixes is recommended. For instance, parameters starting with prefix ‘arg’ and internal parameters with ‘internal’ (‘a’ is before ‘i’) holding the input variables and output variables, respectively (‘in’ for ‘out’). Since the size of the parameter list and the number of outputs is passed to the external function, the number of input variables can only be determined if the number of DSL parameters is known. It has turned out to be practicable to design only the block which communicates with the simple external function with a simple and distinct functionality and fixed number of parameters.

Depending on the general behaviour of the DSL block, the output of the function might be delayed to the next simulation time step, while issuing an event would immediately affect the network model. Events in DSL are triggered when a positive

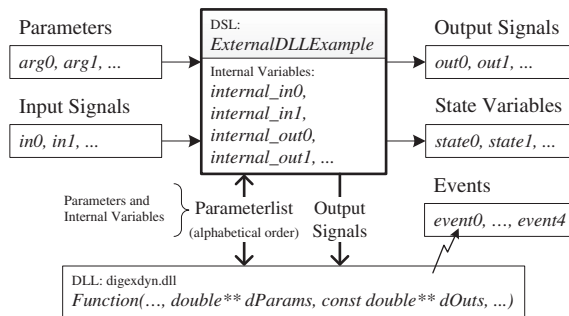


Fig. 15.10 Exchanging variables with the external event driven function

```
event(0,clock,'function=FunctionName')
event(1,modulo(time())/step_size,2)-1,'function=Controller')
```

Fig. 15.11 DSL syntax for event triggered external functions

```

! DSL Code ExternalDLLExample
inc(internal_out0)=0      !initial condition
inc(internal_out1)=0      !initial condition

inc(out0)=internal_out0   !initial condition
inc(out1)=internal_out1   !initial condition

! assign 'in' variables to 'internal_in'
internal_in0=in0
internal_in1=in1

! external function reads 'internal_in' and writes to 'internal_out'
event(1,module(time()),2)-1,'function=IPChannelEvent')

! assign 'internal_out' parameter to 'out'
out0=internal_out0
out1=internal_out1

```

Fig. 15.12 DSL equations for *ExternalDLLExample*

zero passing of the clock argument occurs, e.g. when it changes from -1 to $+1$. Therefore, the event can be triggered only every second simulation time step as the maximum trigger rate or a second event call is added with an inverted clock signal. The listings in Figs. 15.12 and 15.13 show the DSL code for the event driven external DSL function and the implementation in C++.

```

//extract of userdyn.cpp
void __cdecl eventdrivenFunction(
    double tEvent,           // simulation time in seconds
    double** dParams,        // parameter array
    const double** dOuts,    // output signals array
    const double** dIntSigs, // not used (NULL)
    char** eventstr,         // char*[5] max. length char[100]
    char* msg,               // message to output window
    int nParams,             // no of parameters
    int nOuts,               // no of outputs
    int nIntSigs)            // not used
{
    double *param0, *internal_in0, *internal_out0;
    double result0, evntDelay = 0;
    param0 = dParams[0];     // parameter of the DSL block
    internal_in0 = dParams[2]; // input signal in0 in DSL
    internal_out0 = dParams[3]; // output signal out0 in DSL

    /// do calculations here ///

    *internal_out0 = result0; // will be assigned to out0
    sprintf(msg,"Result = %d", result0); // put output message

    // event e.g. "create=EvtParam name=Test Target=this ...."
    sprintf(eventstr[0], "create=EvtParam target=Object name=Name dtype=%f
variable=Variable value=%d",evntDelay,result0);
}
}

```

Fig. 15.13 Implementation of the event driven external DSL function in C/C++

15.4.2 External C/C++ User Function

Function call with parameters. An external DSL function can be implemented and called from within the DSL block like a common build-in function. Though easier to use and implement, only the variables passed as arguments can be accessed externally and only one result value can be returned (Fig. 15.14). The result can be assigned directly with an output signal. DSL syntax to call the user function is shown in Fig. 15.15. With function calls, it is also possible to exchange data with external applications.

To realise the external function in C/C++, it has to be implemented in *userdyn.dll* and registered in the same file within *RegisterFunctions*. After that the user function has to be added to the module-definition file (*userdyn.def*). In the *digexdyn.dll* example provided by DIgSILENT support of how to implement an external user-defined function (*‘Example shunt controller’*), this will be explained in detail with more documentation and the source code.

The code in Fig. 15.16 shows an example implementation of an external DSL user function.

15.4.3 External Communication with API and External DLL

One of the main drawbacks when running RMS dynamic simulation is the circumstance that changing a calculation relevant model means that the initial conditions for the RMS simulation have to be re-calculated (power flow). Since co-simulation needs to exchange and set model variables, e.g. reactive power output, a mechanism is needed to initiate or trigger changes in the model. DSL has been

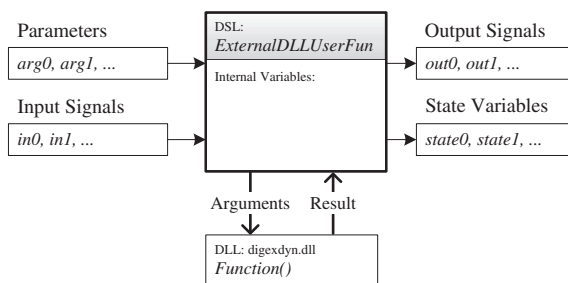


Fig. 15.14 Calling external user-defined functions with arguments and receiving a result

```

inc(result)= 0           ! initial condition
result = functionName(arg0,arg1,...) ! function call with arguments
  
```

Fig. 15.15 DSL call to external user function


```

//extract of userfun.cpp
void __stdcall functionName(void)
{ // example for DSL external user function
  // arguments: arg0, arg1, ...
  double result;

  // pop arguments from stack
  double arg0 = pop(); // 1. argument
  double arg1 = pop(); // 2. argument

  /// do calculations here ///

  // push output signal to stack
  push(result);
}

```

Fig. 15.16 Implementation of external DSL user function in C/C++

developed in order to interact with the network equation system to reflect changes of controller set points and so on. Since there is no possibility to create events with the API dynamically during simulation, it has to be connected to the external defined functions. Both the function call and the event driven external defined functions have the possibilities to interact with the model through the configured signals, wired in the DSL frame/composite model. So the only way to write variables during the RMS simulation is to communicate with the DSL block via the external defined functions. A simple but effective possibility is to define a map (or any other synchronised, thread safe queue) where signals (variables) can be written and read from the DSL and the API. Since this is a very generic way to interact with the running simulation, the API could also receive and send values to other applications via, e.g. TCP/IP sockets, shared memory. This and different variants of the mechanism have been successfully used for various co-simulation applications in laboratory settings [10], teaching [11] as well possibility to co-simulate grid components as if they were in a field test environment [12].

Figure 15.17 shows the concept of using a share map between the DSL block and the API. Only values of type double are allowed for arguments, and therefore, the keys of the nested maps are the *id* and the *variable* (*var*) argument.

The listing in Fig. 15.18 shows the (atomic) operation to read and write to the map. The functions are called from the DSL block and from an application using the API.

15.5 Example Project: Co-simulation via TCP Sockets

15.5.1 Description

A versatile mechanism for connecting to other applications is achieved by opening TCP/IP-based network socket connections and sending data encoded by a protocol and decoded by the receiving side. It can be accessed in the project by activating *Study Case* → *Study Case CoSim*.

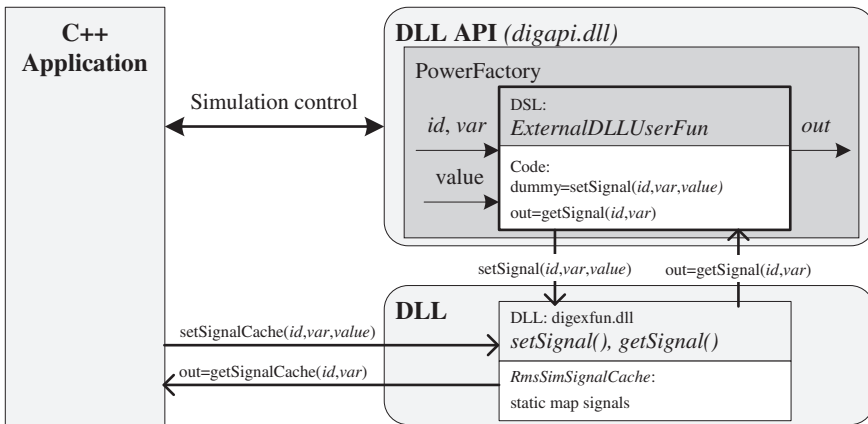


Fig. 15.17 Exchanging signals (values) between an application and an external user-defined function during RMS dynamic simulation

```

//extract of userfun.cpp
// Gets the value for the given variable at the specified id
void __stdcall getSignal(void) {
    double signal_id=pop() // identification of the map (int)
    double signal_var=pop(); // identification of the var (int)
    double d=RmsSimSignalCache::getSignalCache(
        (unsigned int)signal_id,(unsigned int)signal_var);
    push(d); // value of the variable (double)
}

// Set the value for the given variable at the specified id
void __stdcall setSignal(void) {
    double signal_id=pop(); // identification of the map (int)
    double signal_var=pop(); // identification of the var (int)
    double signal_value=pop(); // value of the variable (double)
    RmsSimSignalCache::setSignalCache((unsigned int)signal_id,
        (unsigned int)signal_var,signal_value);
    push(signal_value); // value of the variable (double)
}

```

Fig. 15.18 Implementation of external DSL user function in C/C++ for setting and getting variable/value pairs

In this example, the use of C++ implemented, external user-defined DSL functions are demonstrated. The external DLL opens a socket connection to a specified server and transmits the values encoded in Abstract Syntax Notation (ASN.1). At the server, a simple tap changer controller decides on the received voltage level whether to increase or decrease the voltage by setting a new tap position. This simple example demonstrates the possibilities of communication with external applications by using the provided DSL functions for communicating with external applications. The server side is implemented in Java, but any programming language or application can be used, which supports socket connections. The following chapter will guide you step by step through the process of setting up this example.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- simple test config -->
<ipconnection logging="all" logfile="D:\PFIPConnection.log">
  <channel id="1">
    <host>127.0.0.1</host>
    <port>1598</port>
    <writeonchanged>no</writeonchanged>
    <blocking>no</blocking>
    <inputs>2</inputs> <!--number of vars to write (e.g., inTap, inVolt) -->
    <outputs>1</outputs> <!--number of vars to read (e.g., outTap) -->
  </channel>
</ipconnection>
```

Fig. 15.19 XML configuration for the TCP socket connections

15.5.2 Preparation

The external user-defined C++ functions have been compiled into dynamic linking libraries. These files have to be located in the PowerFactory installation folder (e.g. *D:\Program Files\DIgSILENT\PowerFactory 15.1*) in order to be loaded automatically at start-up:

- *PFIPConnection.xml*—XML-based configuration of the TCP sockets
- *DigExFunPFIP.dll*—external defined DSL user function
- *DigExDynPFIP.dll*—external defined DSL event driven function
- *PFIPConnection.dll*—TCP/IP connection class
- *Toolbox.dll*—Helper class

The libraries are provided for the 32 and 64 bit versions.

The XML code in Fig. 15.19 shows the necessary configuration file for the TCP connection. When data is written to the socket connection, it does not wait until an answer is received unless this is configured with the blocking attribute. This enables synchronisation with the external application.

15.5.2.1 Folder Structure and References

References to external files (e.g. profiles) are generally a problem when exchanging or sharing the same project. It has turned out to be very practically to use references in your project to a dedicated network drive (e.g. A:²). The mapping can be done to any folder on your PC (even another network drive). The advantage is clear that you can share your project without having to change the path references. Your colleagues only have to map their data folder to the dedicated network drive. In windows, mapping of network drives to local folders is done via the

² In case you might still use your A: floppy drive (e.g. doing the PowerFactory database backup regularly on 1.44-MB floppy disks), you might consider to dismount the 5.25" floppy drive, backup the 360 kB data to your 3.5" floppy disks in drive A: and use the drive letter B: for the network mapped folder.

Explorer → *Extras* → *Map network drive ...* and map for drive letter A: for instance to \\127.0.0.1\\d\$anyfolder.

In order to have the right references to the external measurement data files, the batch file *configure.bat* has to be executed. It will map the network drive A: to the data folder contained in that folder automatically for you. You can start the batch file on the CD drive or in the folder on your hard disk, if you have made a copy from the CD. All external profiles in the example project are mapped to, e.g. A: \\data\\PV22_DSL.txt.

15.5.3 DSL Block Definition

The necessary DSL blocks can be found in the project under *Library* → *User-Defined Models* → *Files* and → *PF TCP Com*.

15.5.3.1 Profiles

The block definition for accessing external profiles to be included into the network model is simple and links the file columns to the parameters '*Pext*' and '*Qext*' of the load object (refer to the graphic page of the block or → *Show Graphic* in the context menu in the library).

15.5.3.2 Communication with the External Controller

The block definition of the DSL *TapChangerCtrl*, frame is shown in Fig. 15.20. The reference voltage could also be taken from the transformer if it is re-wired in the block definition (connecting transformer output port 1 to the '*TcpCom TapCtrl*'). The '*TcpCom TapCtrl*' is the block (or also called 'slot') for communication with

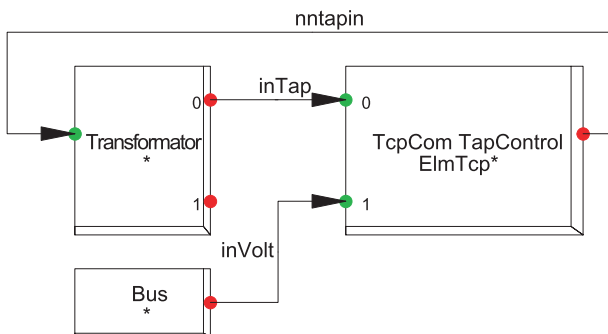


Fig. 15.20 DSL frame for the tap changer control

the external control algorithm. It takes the tap position and the voltage as an input and sets the tap position according to the controller parameters.

The DSL code for communicating with the external control is shown in Fig. 15.21. The function call is simple to use, and the event driven function call could be used alternatively (not shown here). The tap position and the bus voltages are written to the external application, and the new tap position will be read.

The function call and TCP connection to a Java server, where a simple tap changer control algorithm is implemented, are shown in Fig. 15.22.

15.5.4 Composite Models in the Network

DSL composite model (*ElmComp*) is the ‘instance’ of the block definition in the network model. In the *Data Manager*, add a new composite model to the grid: *New Object* → *Composite Model*. This has to be done for the profiles (*EV*, *Loads*, *PV*) as well as for the TCP/IP Connector *TapChangerCtrl* with the external controller. You can put them for instance under the folder *Files* and *Controller*.

```
! Define channelID
vardef(channelID) = ;'ID for TCP config in XML file'
inc(outTap)=IPChannelStart(channelID,60,0)
inc(inTap)=0      ! input tap position
inc(inVolt)=1     ! input voltage

! write and read data over TCP channel
dummy1 = IPChannelWrite(time(),channelID,0,inTap) ! send tap pos
dummy2 = IPChannelWrite(time(),channelID,1,inVolt) ! send voltage
outTap = IPChannelRead(time(),channelID,0)         ! receive new tap
```

Fig. 15.21 DSL code for ‘TcpCom TapCtrl’

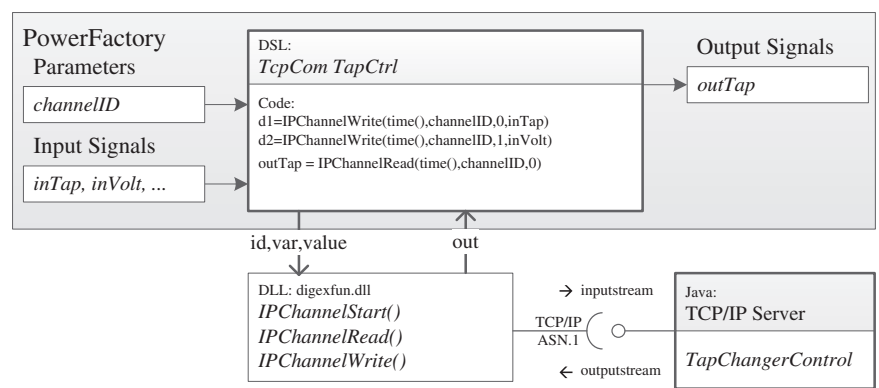


Fig. 15.22 DSL block definition for TcpCom TapCtrl

15.5.4.1 Profiles

To include the profile, a measurement file has to be created for each type of profile (two columns, for the active and reactive power of the load model). In the data manager add to the content of the composite model a new object: *New Object* → *Others* (*Filter: Net elements (Elm*)*) → *Element: Measurement File (ElmFile)*. Name it for instance ‘Measurement File EV’ and select as filename *A:\data\EV_3Ph_11kW_1 day DSL.txt*. Profiles are provided in kW, but since the internal unit are MWs, the scaling factor ‘a’ should be scaled, e.g. 0.01 for EV, 0.001 for active and reactive loads and −0.025 for a 25 kWp PV system (negative load means generation).

After adding a measurement file object and linking it to the right file, it has to be referenced. Edit the composite model and click the right mouse button in the *Net Elements* field of the slot definition: *Select Element/Type ...* and then click on the previous created measurement file for the respective profile. Add the load objects on which the profiles will be applied (e.g. for *EV1* select load objects *Load_EV1* from the *demogrid* network model). Table 15.2 shows an example for the composite model slot definition for EV profiles.

15.5.4.2 Communication with the External Controller

An instance of the block definition of the TCP communication has to be created under the composite model *TapChangerCtrl*. Similar to the measurement file, add a new object (or click the right mouse button): *New Object* → *Common Model (ElmDsl)* and select the block definition ‘*TcpCom TapCtrl*’ for the communication from: *Library* → *User-Defined Models* → *PF TCP Com*.

A dialog will appear which asks for the name of this object (same as block definition, e.g. ‘*TcpCom TapCtrl*’) and the parameter channel ID. This ID refers to the configured channel in the *PFIPConnection.xml* where the host and port address are defined (see Fig. 15.19). An arbitrary number of connections could be established with other instances of this DSL block.

Now, the DSL model has to be assigned to the slot ‘*TcpCom TapCtrl*’ (the name of this block in the block definition). Click the right mouse button into the empty slot: *Select Element/Type ...* and select the DSL model which has been created in the previous step in the content of the composite model in the network model.

The others slots have to be referenced to objects from the network model according to their functionality in the definition. For example, the network’s

Table 15.2 Composite model: slot definition for profiles (e.g. EV)

	Slot BlkSlot	Net elements Elm*, Sta*, IntRef
1	EV profile file	✓ Measurement File EV
2	EV1	✓ Load_EV1
3	EV2	✓ Load_EV2
4

transformer will be assigned to the slot *Transformer* in the composite model, and the voltage is taken from any bus in the network, configured in the slot *Bus* (e.g. *Bus81*).

15.5.5 External Controller (Tap Changer Control)

After setting up the DSL models for profiles and the communication, the external controller (implemented as a server) has to be started. The provided external user function acts as a client and tries to connect to a server at the given host and port provided by the XML configuration in *PFIPConnection.xml*. After establishing a connection, the server will wait for receiving tap position and a voltage and calculates the new tap position according to the given allowed voltage limits.

A very simple server and tap controller for demonstration purposes is provided in Fig. 15.23. The tap change controller is just a conditional statement checking if the voltage exceeds or is below a given limit.

The Java program can be started with the batch command file provided (*start_tapChangerController.bat*) or via the shell command line in Fig. 15.24. Note that *java.exe* must be in the path or the path has to be specified in the command.

15.5.6 Simulation Set-up

To start the simulation, the server with the tap change controller has to be started first in order to accept the incoming connection. The RMS simulation is initialised by: *Calculation* → *RMS/EMS Simulation* → *Initial Conditions* ...

15.5.6.1 Basic Options

The *Simulation Method* should be set by default to *RMS values (Electromechanical Transients)* and the *Network Representation* to *Balanced, Positive Sequence*.

15.5.6.2 Step Sizes

The profiles of the simulation are based on per minute based averaged measurements for one day, thus having 1,440 entries. The unit for the time index in the measurement files is seconds; therefore, the simulation end time is 1,440 s (one second corresponds to one minute). The step size of the simulation can be set to one second. The smaller the simulation time step, the faster is the update of the simulation with the controller response from the external application, since at least one simulation time step is passing before the new set point is applied to the network

```

class ASN1_Server {
    private static final int _DOUBLESIZE = 8; // double has 8 byte
    private static final int _TYPESIZE = 1; // first byte is the type

    public static void main(String argv[]) throws Exception {
        int port = Integer.parseInt(argv[0]); // first argument: port
        double upperLimit = Double.parseDouble(argv[1]); // upper limit
        double lowerLimit = Double.parseDouble(argv[2]); // lower limit
        byte[] bIn = new byte[18]; // incoming byte array
        byte[] bOut = new byte[9]; // outgoing byte array

        ServerSocket serverSocket = new ServerSocket(port);
        Socket socket = serverSocket.accept();
        InputStream in = new BufferedInputStream(socket.getInputStream());
        DataOutputStream out = new DataOutputStream(socket.getOutputStream());

        while (true) {
            String action = ""; // assume no action
            int numBytes = in.read(bIn);
            ByteBuffer bb = ByteBuffer.wrap(bIn); // using ByteBuffer class
            double tap = bb.getDouble(_TYPESIZE); // starting after type byte
            double value = bb.getDouble(_TYPESIZE+_DOUBLESIZE+1); // next double

            // CONTROLLER START
            if (value > upperLimit) {
                tap = tap + 1; // bring voltage down
                action = "UP"; // tap one position up
            } else if (value < lowerLimit) {
                tap = tap - 1; // increase voltage
                action = "DOWN"; // tap one position down
            } // CONTROLLER FINISH

            if (!action.isEmpty()) { // send new position
                bOut[0] = bIn[0]; // encode double type (75)
                ByteBuffer bbOut = ByteBuffer.wrap(bOut);
                bbOut.putDouble(_TYPESIZE, tap); // after the first byte
                out.write(bbOut.array()); // get byte array
                out.flush(); // send immediately
            }
        }
    }
}

```

Fig. 15.23 Java server and tap changer control

```

java -jar path_to\ASN1_Server.jar port upperLimit lowerLimit
java -jar tapChangerCtrl\ASN1_Server.jar 1598 1.03 0.96

```

Fig. 15.24 Shell command to start the server

model. The error due to latency correlates with the synchronisation points (simulation time steps) with the external application.

15.5.6.3 Real Time

The option *Real-Time Simulation* is used and should be set to *Synchronised by system time*, since the external controller needs time to process the signals and calculate the

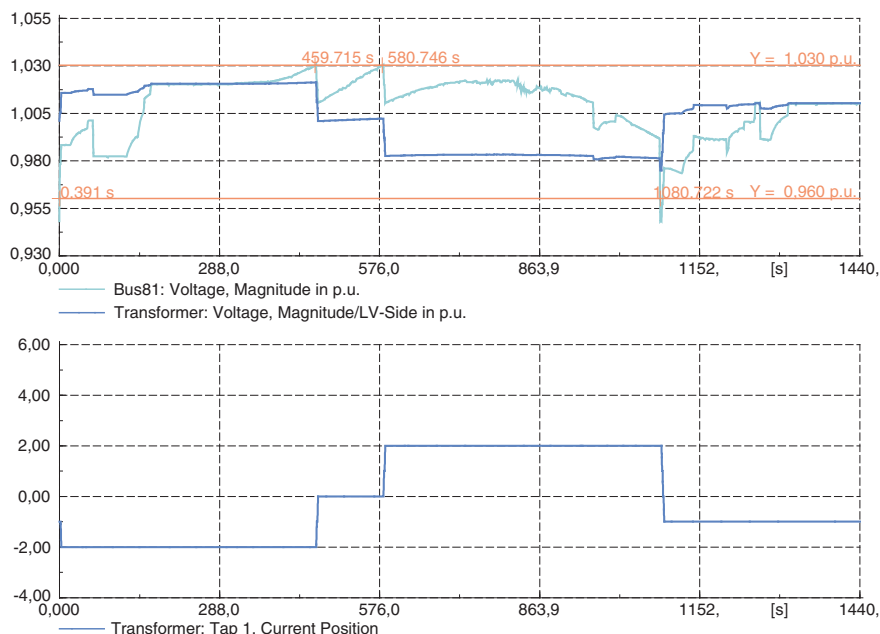


Fig. 15.25 Simulation results: voltages and tap position

external signal. If the simulation is not synchronised, either by time or a dedicated simulation control, the simulation will run-out of synchronisation and result in errors. At least an up scaling of the simulation (*Ratio between real time and Calculation time*: 1,000 %) should not be too fast for the control loop, thus resulting in a total simulation time of 144 s (10 times faster of 1,440 simulation steps).

15.5.7 Simulation Run

With *Calculation* → *RMS/EMS Simulation* → *Start Simulations* ... and simulation stop time set to 1,440, the results in the diagram *Voltages and Tap Position* should look like Fig. 15.25. The voltage in the bus will reach the upper limit twice and once the lower limit, so the tap control reacts accordingly.

15.6 Conclusions

This chapter has shown different ways of interacting with PowerFactory. This includes interaction with external models or interaction from external applications with the network model during simulation and the control of the simulation. The

interfaces are suited differently for specific applications (e.g. model exchange, real time, co-simulation with or without standardized clients) and the various combinations make them powerful to achieve almost any requirement for interchanging all sorts of data and signals.

One of the mayor advantages of interfacing with external applications is the coupling with external applications for validation within simulated environments. This decreases the time for prototyping and enables the direct deployment of the developed application (e.g. controller).

References

1. Andren F, Stifter M, Strasser T, Burnier de Castro D (2011) Framework for co-ordinated simulation of power networks and components in smart grids using common communication protocols. In: IECON 2011—37th annual conference on IEEE Industrial Electronics Society, pp 2700–2705
2. Andren F, Henein S, Stifter M (2011) Development and validation of a coordinated voltage controller using real-time simulation. In: IECON 2011—37th annual conference on IEEE Industrial Electronics Society, pp 3713–3718
3. Stifter M et al (2011) DG DemoNet validation: voltage control from simulation to field test. IEEE PES Innovative Smart Grids Technologies Europe, Manchester
4. Stifter M, Schwalbe R, Tremmel W, Henein S, Brunner H, Bletterie B, Abart A, Herb F, Pointner R (2012) DG DemoNet: experiences from volt/var control field trials and control algorithm advancements. In: 2012 3rd IEEE PES Innovative Smart Grid Technologies (ISGT Europe), pp 1–7
5. Burnier de Castro D, Ubermasser S, Henein S, Stifter M, Stockl J, Hoglinger S (2013) Dynamic co-simulation of agent-based controlled charging electric vehicles and their impacts on low-voltage networks. In: 2013 IEEE International Workshop on Intelligent Energy Systems (IWIES), pp 82–88
6. Stifter M, Ubermasser S (2013) Dynamic simulation of power system interaction with large electric vehicle fleet activities. In: PowerTech (POWERTECH), 2013 IEEE Grenoble, pp 1–6
7. Stifter M, Bletterie B, Burnier D, Brunner H, Abart A (2011) Analysis environment for low voltage networks. In: 2011 IEEE first International Workshop on Smart Grid Modeling and Simulation (SGMS), pp 61–66
8. Stifter M, Schwalbe R, Andren F, Strasser T (2013) Steady-state co-simulation with PowerFactory. In: 2013 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES), pp 1–6
9. Palensky P, Widl E, Stifter M, Elsheikh A (2013) Modeling intelligent energy systems: co-simulation platform for validating flexible-demand EV charging management. IEEE Transaction on Smart Grid 4(4):1939–1947
10. Andren F, Stifter M, Strasser T (2013) An environment for the coordinated simulation of power grids together with automation systems. In: PowerTech (POWERTECH), 2013 IEEE Grenoble, pp 1–6
11. Strasser T, Stifter M, Andren F, Palensky P (2014) Co-simulation training platform for smart grids. IEEE Transactions on Power Systems 29(4):1989–1997. doi:10.1109/TPWRS.2014.2305740
12. Mosshammer R, Kupzog F, Faschang M, Stifter M (2013) Loose coupling architecture for co-simulation of heterogeneous components. In: IECON 2013—39th annual conference of the IEEE Industrial Electronics Society, pp 7570–7575

Chapter 16

PowerFactory as a Software Stand-in for Hardware in Hardware-In-Loop Testing

Radhakrishnan Srinivasan

Abstract Rapid prototyping and validation of power system controllers have always been a challenge. With a rapid increase in distributed energy resources (DERs) and power-electronics-based power system devices, the systems that control today's electrical grid infrastructure are set to become more complex than ever, especially on the distribution front. Thus, rapid prototyping and validation of new complex control systems require testing with extensive set of test cases involving a multitude of power system components. The numerous types of variable involved, such as topology, energy flow, fault conditions, autonomous control systems, and protective device operation, most of which are stochastic in nature, naturally compound the problem of prototyping and validation. In such a scenario, using a hardware-in-loop (HIL) to test all possible test conditions is quite unrealistic. This chapter is about introducing DlgSILENT's PowerFactory as a good software stand-in for Power Systems in HIL testing.

Keywords Real-time simulation • Hardware-in-loop testing • Rapid prototyping • Validation of power system controllers

16.1 Introduction

In recent years, power system control is becoming increasingly complex with the introduction of new types of energy resources and the increase in interconnections using HVDC technology. Moreover, increasing calls for smarter grids are also demanding research and development of more advanced and complex control

Electronic supplementary material The online version of this chapter (doi:10.1007/978-3-319-12958-7_16) contains supplementary material, which is available to authorized users.

R. Srinivasan (✉)

Electrical Power Systems Engineer, Alstom Grid, Redmond, WA 98052, USA
e-mail: radhakrishnan1982.s@gmail.com

schemes. Rapid prototyping and validation of power system controllers has always been a challenge [1]: Setting up even a modest power system laboratory involves significant real estate and huge capital and operational cost. DIgSILENT's PowerFactory, equipped with strong modeling (both static and dynamic), simulation, and input/output capabilities proves to be a good software stand-in for more demanding HIL testing. PowerFactory has a vast library of static and dynamic power system models, including a spectrum of models for protection devices. Power system operation variables such as load profiles and switch statuses can be managed through specialized data organization mechanisms called study cases and scenarios. PowerFactory also allows users to develop their own physical systems using a built-in dynamic simulation language (DSL). As modeling power systems involves large number of parameters, PowerFactory provides a handy tool to estimate parameters.

PowerFactory features two types of simulations: The RMS simulation uses larger step sizes of integration, ranging from μs to ms or minutes, while the EMT simulation is for simulating electromagnetic transients at even smaller step sizes. Real-time simulation mode is another feature that proves to be very valuable. PowerFactory also supports an A-stable numerical integration algorithm where the steps are adjusted by PowerFactory such that smaller step size is used during transient condition, while the step size is increased as the simulation reaches a steady state.

PowerFactory supports simulation of virtually all types of events that occur in a power system such as switching, different varieties of fault, tap changing, and loading to name a few. Another important event supported by PowerFactory is the parameter event, through which specific parameters of individual models can be modified and used to communicate with the outside world: for instance, to provide measurements and receive control signals from a control system. Because OPC is one of the most common protocols used by the process and control systems industry, it is leveraged by PowerFactory as the preferred protocol for input and output. It also supports the scaling of signals while passing to and fro through these communication objects.

Yet another significant feature supported by PowerFactory is remote/engine mode operation, whereby all the functions can be executed through a remote procedure call. This is a pronounced advantage in control system development and automatic validation cycle. This chapter is intended to explain how to use DIgSILENT's PowerFactory as a software substitute for the hardware in HIL testing and control system prototypes.

16.2 Test System

The given demonstration system is a three-phase, 0.4 kV, 0.5 MVA photovoltaic (PV) system. The model assumes that the PV system is capable of dispatching reactive power and is scalable to any voltage and power ratings. To keep the model

simple, the power electronics portion of the system is not modeled in detail; rather, it is mimicked by controlling the real and reactive current of the source. Irradiance and temperature values of the model are fed as a time series from an *ElmFile* object in PowerFactory. This feature in PowerFactory allows models to consume the measured or simulated time series value of any variable, making the models more realistic. The demonstration system itself serves as a testimony to the modeling and simulation capabilities of PowerFactory (see Figs. 16.1 and 16.2).

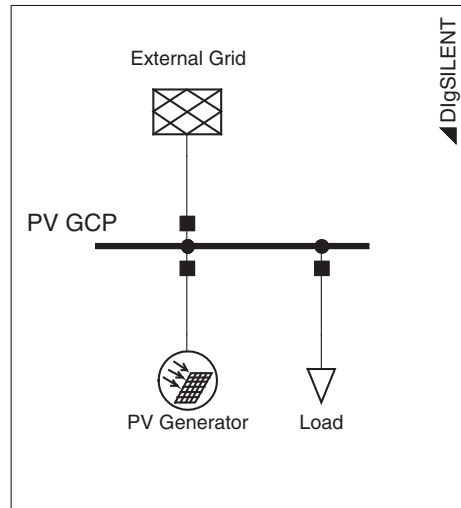
Figure 16.2 shows the dynamic portion of the model. Main components of the model are as follows:

Irradiance and Temperature Profile: These two slots in the model take an *ElmFile* object which feeds in the time series data of the irradiance and temperature profile.

Photovoltaic Model: Marked ⑤ in Fig. 16.2 is the block that models the current and voltage output of the PV cell at the maximum power point (MPP) on the V–I characteristics of the PV module. The given model does not have MPP tracking. It assumes the panel is operating at MPP voltage and is equal to the reference DC voltage feedback from the controller block. The readers can define the V–I characteristics of the PV cell as required through the “user-defined parameters” in the *ElmDsl* object. The *ElmDsl* object also takes the number of cells connected in series and parallel combinations to scale up/down the model to a desired voltage and power rating. The voltage and current calculations are implemented using DSL code, and equations are summarized as follows [2, 3];

$$I_{\text{set}} = I_{\text{sc1}} + \left[1 + \frac{K_I}{100} (T - T_{\text{ref}}) \right] \quad (16.1)$$

Fig. 16.1 Test system used for demonstration (0.4 kV, 0.5 MVA three-phase PV system)



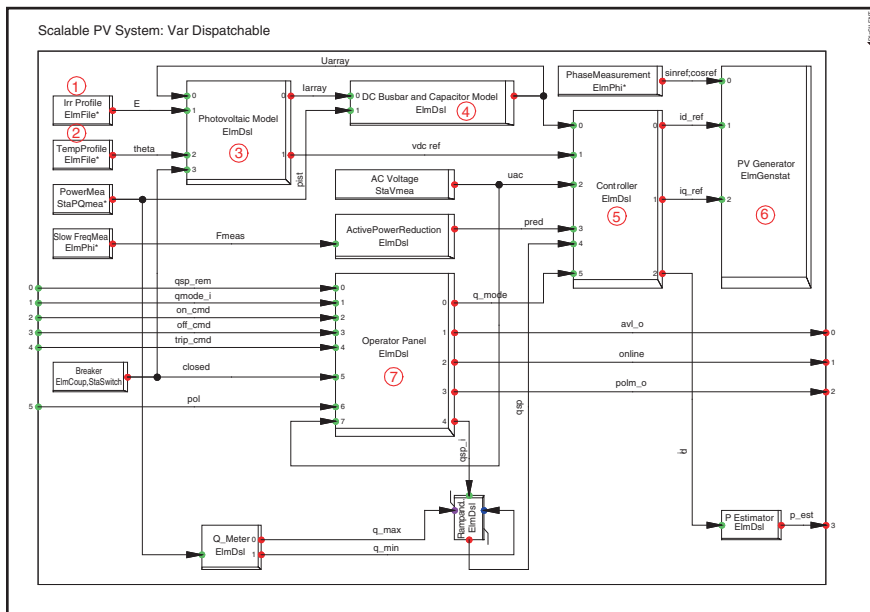


Fig. 16.2 PV system controller—DSL model

$$I_{sc} = I_{sct} \left(\frac{E}{E_{STC}} \right) \quad (16.2)$$

$$V_{oct} = V_{oc1} + [K_V(T - T_{ref})] \quad (16.3)$$

$$V_{oc} = V_{oct} \left(\frac{\ln E}{\ln E_{STC}} \right) \quad (16.4)$$

where

- I_{sct} Short-circuit current with only temperature dependence
- K_I Temperature correction factor for current given by PV panel manufacturers
- I_{sc} Short-circuit current with both temperature and irradiance dependence
- V_{oct} Open-circuit voltage with only temperature dependence
- K_V Temperature correction factor for voltage given by panel manufacturers
- V_{oc} Open-circuit voltage with both temperature and irradiance dependence
- T_{ref} Temperature at standard testing condition (STC)
- I_{sc1} Short-circuit current at STC, given by panel manufacturers
- V_{oc1} Open-circuit voltage at STC, given by panel manufacturers
- E_{STC} Irradiance at STC
- E Measured irradiance. In reality, it depends on the sunshine. But in the model, it comes from the profile
- T Measured temperature. Same as irradiance, it comes from the profile in the model

DC Bus bar and Capacitor Model: This block defines the dynamics of the DC bus bar and the capacitor in the PV system. Output from this block is the DC voltage across the capacitor in a real system.

Controller: Controller block is the control component of the given PV system model. It has two parts: active power control and reactive power control. The active power and reactive power are controlled using direct axis (I_d) component and quadrature axis (I_q) component of the current vector, respectively.

Active power control is designed to comply with the German grid code [2]. Thus, it implements the following control schemes: over-frequency power reduction and power off under low voltages.

Reactive power control is also designed to comply with the German grid code—Transmission Code 2007 and the System Service Ordinance SDLWindV [2]. In addition to that, the control is modified to make it a reactive power dispatchable DER.

PV Generator: This slot in the model takes static generator (*ElmGenstat*) object. *ElmGenstat* object in PowerFactory includes the DC side and the converter portions of an energy resource. Though *ElmGenstat* can be used both as a voltage source and as a current source, the given project uses it as a current source. As explained in the controller section, the active power output and reactive power output are controlled through I_d and I_q components of the current vector.

Operator Panel: Operator panel takes in a custom DSL model responsible for switch-in/cutoff operations. Additionally, it also receives and passes on all other remote and local commands and set points to appropriate blocks in the model as summarized below:

- Enable/disable remote control—all the following remote commands can be executed when the asset/energy resource is set to be available for remote control. When disabled, the asset goes to local control mode and takes command from the asset level control signals
- On/off from a remote system through OPC link¹ (on and off commands are executed after a delay to simulate communication and mechanical delays as in any physical system)
- Immediate tripping through OPC link (by this command, the asset is tripped without any delays)
- Change reactive power dispatch mode—remotely via OPC Link
- Change reactive power set point—remotely via OPC Link
- Change on/off delay—via model parameters
- Change local reactive power mode and set point—via model parameters
- Active power production is controlled via irradiance and temperature profiles.

¹ A brief introduction to OPC technology is given in the next section of this chapter.

16.3 Introduction to OPC Technology

Before moving ahead to run the simulation using the given test system, it is good to have a brief understanding about the OPC technology. PowerFactory uses OPC technology to communicate between the simulation and the power system control on the other side. OLE for process control (OPC), which stands for object linking and embedding (OLE) for process control, is the name of the standard specification developed by industrial automation task force.

OPC² standard specifies the communication of real-time plant data between control devices from different manufacturers. Since it is one of the widely used standards in the control system and SCADA industry, PowerFactory chooses to support OPC standard to communicate with other systems interfacing with simulations running in PowerFactory.

16.4 Making the Demonstration System Using the Given Template

Demonstration project file (*HIL_PV_DEMO.pfd*) provided with this chapter includes a working version of the model described above. However this section of the chapter explains how readers can make their own copy of the given model. PowerFactory users can make their own user-defined dynamic models and connect them with the built-in power system component models to model a real-world system. Having made such useful models, users can bundle them into templates so that the same system can be reproduced any number of times without having to rewire the control blocks again and again. The following procedure explains how readers can make their own versions of the model discussed above from the template given. (For instructions on how to make a template, refer to the PowerFactory user manual or contact PowerFactory support.)

1. Import the demonstration project (*HIL_PV_Demo.pfd*) supplied with this chapter
2. Position yourself on the single-line graphics board
3. Click on the “General Templates” button from the drawings toolbar to the right of the graphics board (see ❶ in Fig. 16.3)
4. Click “General Templates” to open a window that will display all templates in the <Project> → *Library* → *Templates* folder and other templates built into PowerFactory (see ❷ in Fig. 16.3)

² OPC standards are being maintained by the OPC foundation, and more details can be obtained from the following link: <https://opcfoundation.org/>.

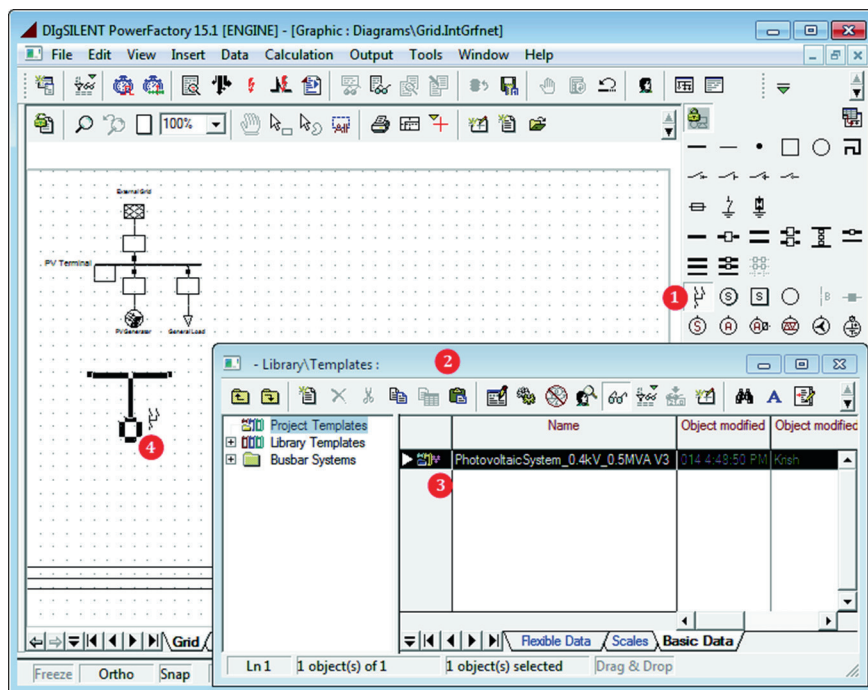


Fig. 16.3 Using template to create the demonstration system

5. Click to choose the template of your choice (“PVSystem_0.4kV_0.5MVA V3” in our case) and click twice on the desired location of the graphic board to get your own copy of the system (see ❹ in Fig. 16.3).³

16.5 Running Simulation Using the Test System

The block diagram in Fig. 16.4 shows all the essential components required in order to use PowerFactory as a software stand-in for hardware in a HIL testing of power system controls. The block diagram is quite self-explanatory: Commands or the control actions are sent by the *power system control* from one end, while the *dynamic power system* model and the simulation engine on the other end represent a physical power system that sends feedback and receives control signals to act on.

The centerpiece of this setup is the *OPC server* and the communication objects on both sides of it. In the given demonstration application, readers will be shown

³ Readers will need to adjust the parameters in “PV Array.ElmDs1” to achieve the desired MW output at a desired voltage.

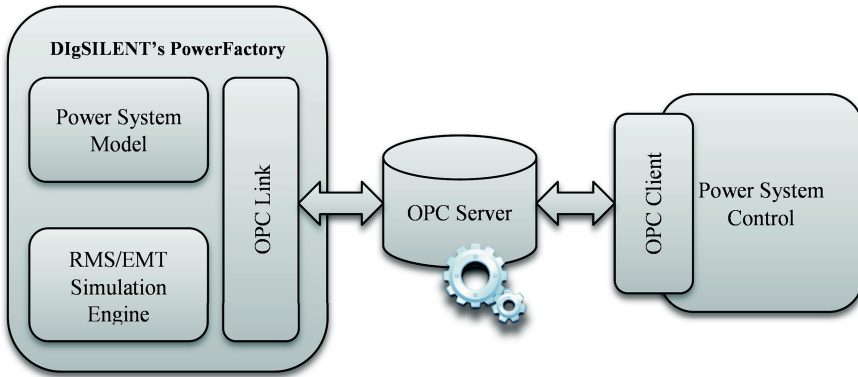


Fig. 16.4 HIL testing/simulation schematic

how to use a free but restricted version of MatrikonOPC server⁴ and an OPC server explorer that acts as the control system. The OPC server explorer will be used to send commands and set points. However, in real life, readers or other PowerFactory users will have an actual control system (the system being tested or prototyped) that will send the set points and control signals.

16.5.1 Prerequisites

The following discussions and instructions assume that the reader has PowerFactory Version 15.0 or later and that the demonstration system has been imported and is ready for simulation. In order for the simulation to communicate with any external system, we need an OPC server (running locally or in the network) with which PowerFactory OPC link object can communicate. The following section of the chapter explains in brief all these prerequisites.

Readers must also have the following software applications installed to simulate and control the test system provided with this chapter:

- MatrikonOPC Simulation Server⁵ V1.2.4 or newer, as recommended by DIgSILENT (for the example explained in this chapter, MatrikonOPC Simulation Server V 1.5.0.0, the latest version at the time, was used.)
- For x64 systems, use OPC Core Components Redistributable⁶

⁴ Free version of MatrikonOPC simulation server and server explorer can be downloaded from the following link: <https://www.matrikonopc.com/products/opc-desktop-tools/index.aspx>.

⁵ "MatrikonOPC Simulation Server" is freeware and can be downloaded from Matrikon's Web site. It is recommended that you use *all default options* for installation.

⁶ If running on a 64-bit system (x64), the OPC core components from OPC foundation need to be installed in addition. They are not required for 32-bit systems. These components can be downloaded from the OPC foundation Web site for free.

- On Microsoft Windows XP SP2, some DCOM settings must be changed. Please refer to the instructions provided in “Using OPC via DCOM with Microsoft Windows XP Service Pack 2.”

16.5.2 Configuring OPC Server

- OPC Server must be configured before it can transfer data to and from the control system on the other end of PowerFactory in a HIL test setup. The following section offers a step-by-step procedure for configuring the OPC server.
- Start OPC server as follows: *Start Menu* → *MatrikonOPC* → *Simulation* → *MatrikonOPC Server for Simulation*.⁷
- Successful starting of the OPC server is represented by the server window as shown in Fig. 16.5. Closing the server window stops the server, so make sure this window is open always during throughout the running of this demo.
- OPC server needs data points to which clients (PowerFactory and the control system in this case) read from and write to. OPC standard supports server data types (refer to OPC standard documents for more details on supported data types). The demonstration application also needs such data points, popularly known as OPC tags. Readers can manually create them one by one or import a preconfigured list from the CSV file provided, “*HIL_PV_Demo_OPC-STag_Config.csv*.” Execute the “*Import Aliases*” command either from the menu bar (*File* → *Import Aliases*) or the toolbar icon to import the OPC tags into server.
- Observe that under alias configuration node, there is now a new group named “*PVAsset*,” which in turn has all the tags defined as shown in Fig. 16.6.

16.5.3 Configuring PowerFactory

In order for PowerFactory to connect to an OPC server, following options must be enabled in its configuration settings.

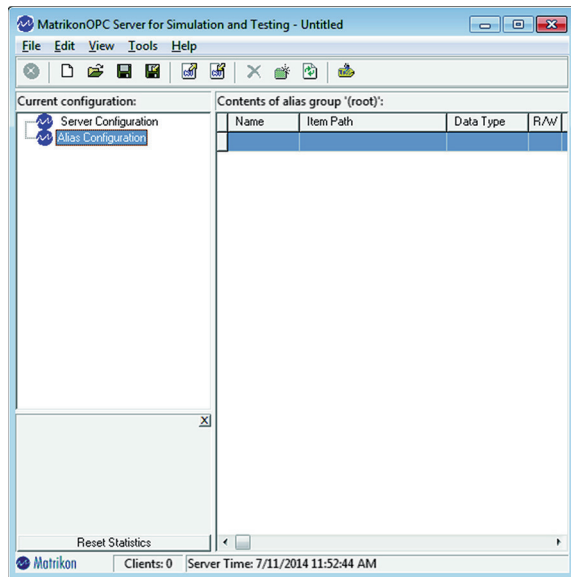
- Runtime Engine Mode
- Enable Multi-Threading

The above-mentioned configuration settings can be enabled using PowerFactory log-on screen as explained below:

- Open PowerFactory using the desktop shortcut or from the Windows start menu.
- When the log-on screen appears, click “*Advanced*” from the list as shown in the Fig. 16.7.

⁷ This is the program location if the server was installed accepting all default options. If not, start the server from the custom location.

Fig. 16.5 MatrikonOPC server screen



- From the contents of advanced configuration, once again choose “*Advanced Tab.*”
- Check “*Runtime Engine Mode*” and “*Multi-threading*” options to enable them as depicted in Fig. 16.7.
- Click *OK* to log on to PowerFactory (assuming that the correct username and password were entered via the log-on screen).

16.5.4 Configuring OPC Tags in PowerFactory Model

PowerFactory needs one external data object for every data point it needs to communicate with the OPC server. These data objects are represented by a class of external measurement objects such as real and reactive power measurement objects, voltage and current measurements, and so forth. These data objects are bidirectional; they can be read from and written to OPC servers. Though the demonstration model has all necessary external measurement objects preconfigured, the following discussion sheds some light on a few important parameters to be configured while creating those external measurement objects.

- **Tag ID:** Tag ID is one of the important configurations as this is the one that maps an external data measurement object with an OPC tag in the OPC server. This tag name must adhere to the following naming convention:

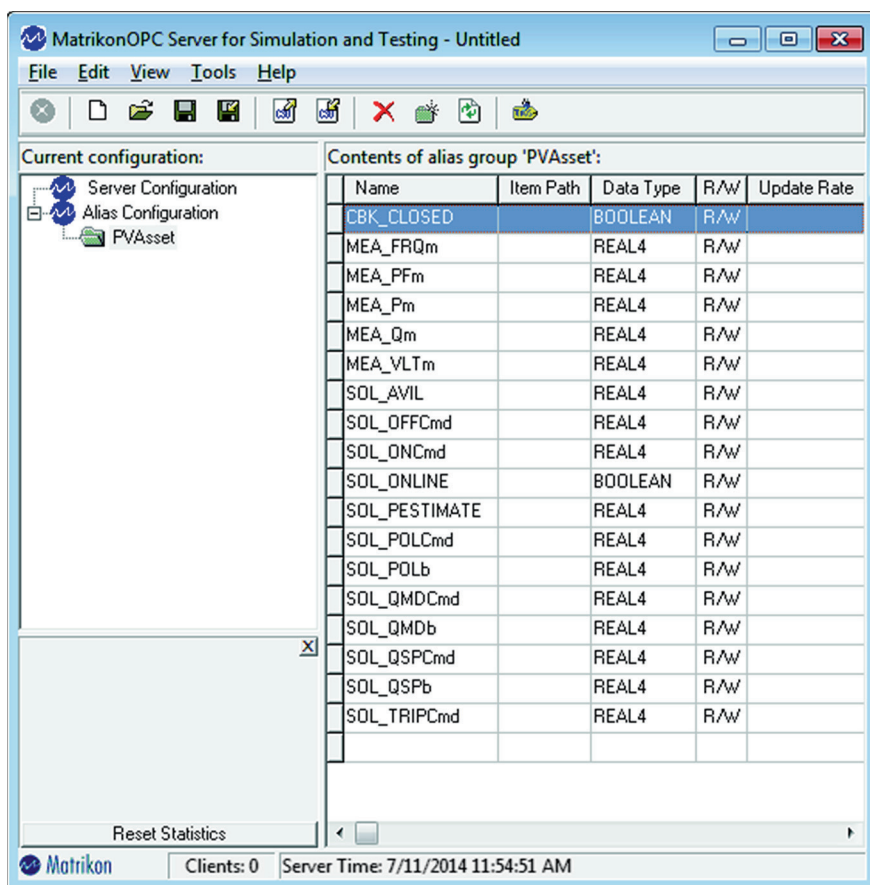


Fig. 16.6 OPC server screen showing the imported group and tag names

$$\text{Tag ID} = \langle \text{Alias Group Name} \rangle . \langle \text{Tag Name} \rangle$$

In the given sample, alias group name is “PVAsset” (see Fig. 16.6). So in order to create an external measurement object that reads in reactive power set point, the tag should have an ID as “PVAsset.SOL_QSPCmd” (marked ❶ in Fig. 16.8). Note that tag ID is configured in the *description* tab of the external measurement object’s edit window. Also in the given sample, each external measurement object is named by its tag ID for easy identification.

- **Read/Write Status:** The read/write status of the measurement object tells whether the tag is configured to read in data from the server or to write data to the server. The example shown in Fig. 16.8 (marked ❷) is reading from the server (MVar set point)

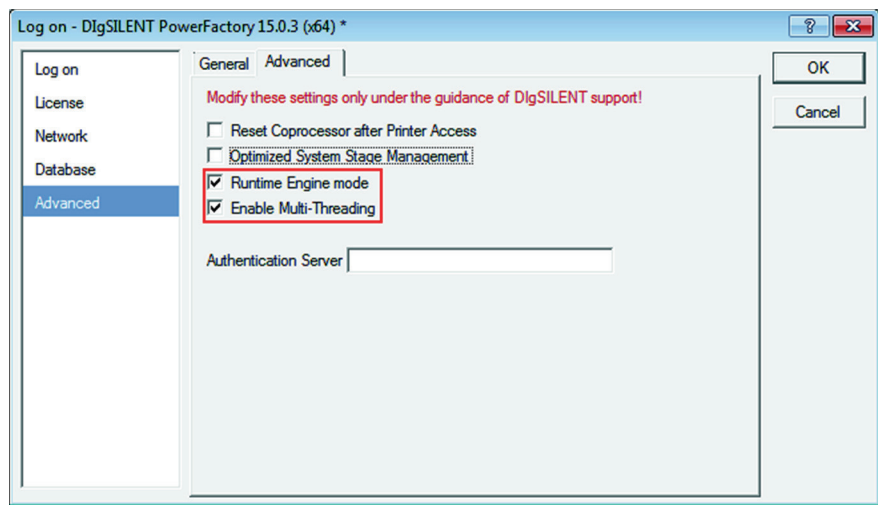


Fig. 16.7 Configuring PowerFactory using its log-on screen

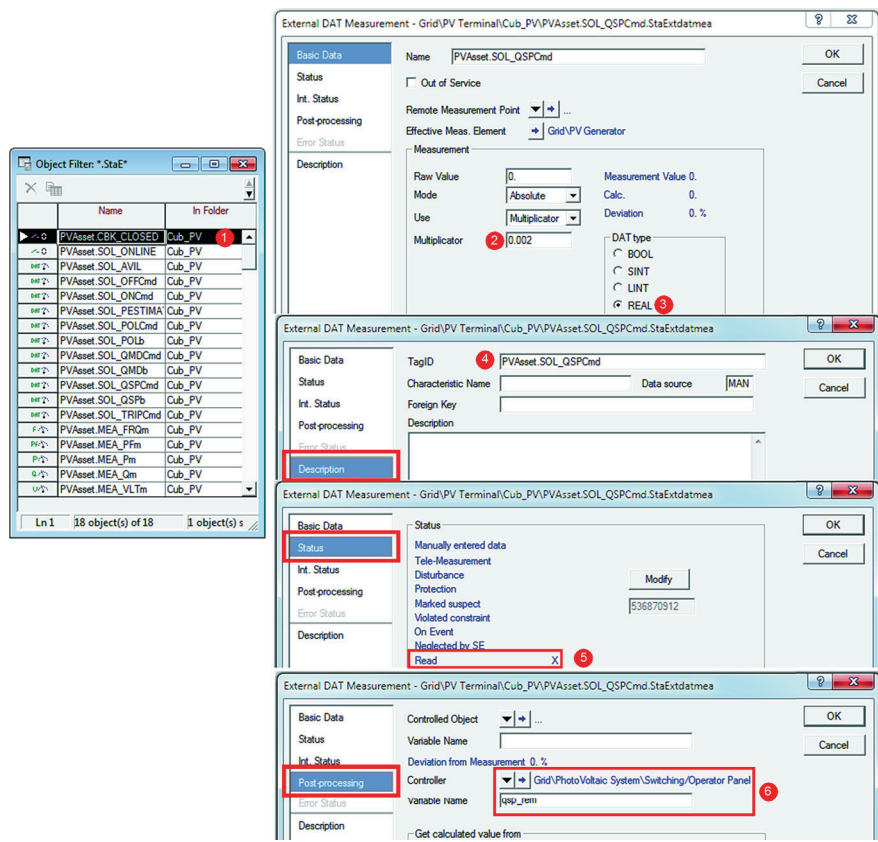


Fig. 16.8 Configuring an external data measurement object

- **Multiplicator:** This is a constant (marked ② in Fig. 16.8) with which the data get multiplied when the data are read from the server. (In the case where data are written into the server, the data are divided by this constant.) In the demo, the PV source has a power rating of 500 kVA. The dynamic model understands the set point in terms of per unit value. This multiplicator ($1/500 = 0.002$) ensures that the user set kVAr is converted into equivalent per unit value before being consumed by the model
- **Post-processing:** This configuration (marked ⑥ in Fig. 16.8) tells the measurement object where to deliver the data if it is reading from server or where to gather data if it is writing data to the server.

As mentioned earlier, the external measurements required to run the sample have been precreated. Readers are advised to refer to the PowerFactory User Manual or technical documentation [4] to learn about other types of measurement object and their parameters.

16.5.5 Connecting PowerFactory Model to OPC Server

The next step in running the demonstration is to import the project provided and establish the data connection between the OPC tags in the model and the data points defined in the OPC server. Establishing this data connection enables the PowerFactory model to read commands from and write measurements to the OPC server. The procedure to establish the data link is as follows:

1. If the sample project provided has not already been imported, import it now using the menu option: “File → Import → Data (*.pfd, *.dz, *.dle)”. From the pop-up dialog, choose the file “HIL_PV_Demo.pfd” to import the demonstration project
2. Activate the project (right click on the project name in the data manager and choose “Activate” from the context-sensitive menu)
3. Locate the OPC connectivity command object (*.ComLink) named link inside the active study case (marked ① in Fig. 16.9)
4. Double click the link object to open its editor
5. Configure ComLink command object’s parameters as explained below;
 - “**Link To**” parameter as “OPC TDS” (marked as ② in Fig. 16.9). TDS stands for time domain simulation. Another useful type of link is “OPC OSE,” where OSE stands for online state estimation.⁸
 - “**Computer Name**” is the computer in which the OPC server is installed. By default, it is assumed that the OPC server is installed and running on the same computer on which PowerFactory is installed

⁸ OPC OSE is not discussed in this chapter. Contact DlgSILENT for their OPC OSE example.

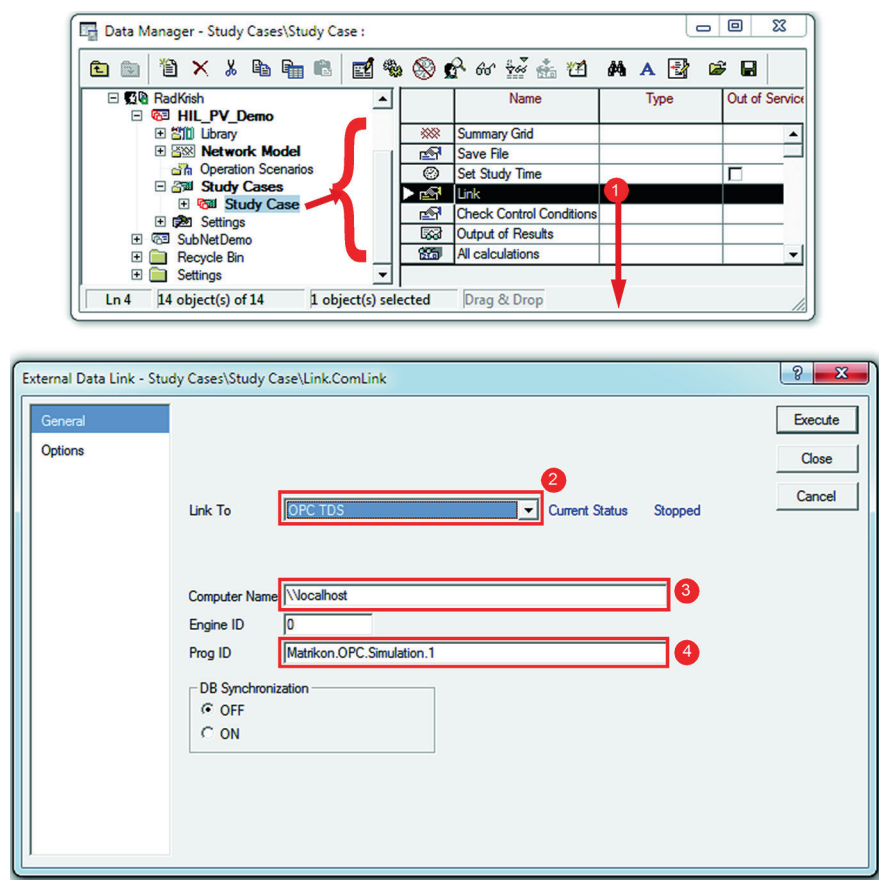


Fig. 16.9 PowerFactory screenshot showing data manger with an active study case and the OPC link command object editor

- “**Prog ID**” is the OPC server program ID which uniquely identifies the OPC server. OPC program ID is mapped against a DCOM CLSID in the registry. The Matrikon simulation server used in the example has the program ID “Matrikon.OPC.Simulation.1.” If you are using a different OPC server, you must use the corresponding program ID.
6. Observe the current status showing as “Stopped.”
 7. Make sure the OPC server is running and the server window is open.
 8. Click the “*Execute*” button in the OPC command object to connect to the OPC server. Note the connection success message in the output window as shown below in Fig. 16.10.
 9. Once again open the OPC command object to observe connection status as “Started”

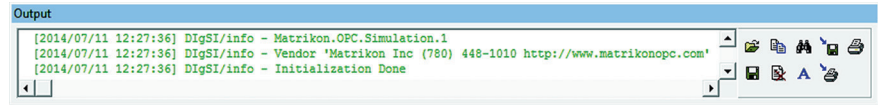


Fig. 16.10 OPC server connection (success) message

16.5.6 Using the Matrikon OPC Server Explorer to Monitor and Control the Demonstration System

As depicted in the HIL schematic in Fig. 16.4, in this demonstration, we have PowerFactory on one side of the HIL testing setup. On the other side of the setup, the Matrikon OPC server explorer has been substituted for the control system to monitor the system and to send the control signals. Matrikon OPC server explorer is another free application that is installed along with the Matrikon simulation server. The procedure to set up the OPC server explorer is as follows:

1. Ensure that OPC server is running and that PowerFactory is connected to the server.
2. Open OPC server explorer from Windows *Start* → *MatrikonOPC* → *Explorer* → *MatrikonOPC Explorer*.
3. Observe the server explorer window opening up with following details:
 - OPC server(s) running on the local computer (marked ❶ in Fig. 16.11)
 - Status of the connectivity between the OPC server and the server explorer (marked ❷ in Fig. 16.11)

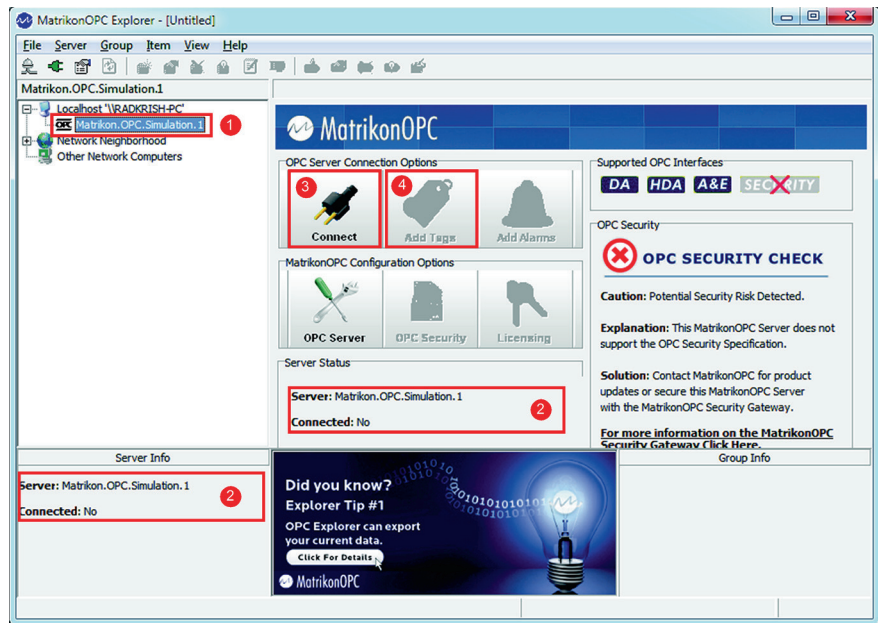


Fig. 16.11 Matrikon OPC server explorer

- Button to connect and disconnect from the OPC server (marked ③ in Fig. 16.11)
4. Click “Connect” to connect to the running OPC server if it is not already connected.
 5. Note the change in connection status and also check that the “Add Tags” button is enabled.
 6. Click the “Add Tags” button to bring up the tag configuration screen as shown in Fig. 16.12.
 7. Choose the “PVAsset” group from the “Available Items” tree view (marked ① in Fig. 16.12) and note that the tags imported into the OPC server are showing in the “Available Tags” list (marked ② in Fig. 16.12)
 8. Right click in the “Available Tags” list view and choose “Add all items to tag list” from the context menu (marked ③ in Fig. 16.12)
 9. Note all the tags showing up in the “Tags to be added list” (marked ④ in Fig. 16.12)
 10. Click OK to add all the tags to the observation list in the OPC server explorer as shown in Fig. 16.13. Check that the qualities of tags are reading good and the values for the data tags are all zero. The values are zero because the simulation has not started; the OPC server has not received any values from the Power-Factory simulation engine.

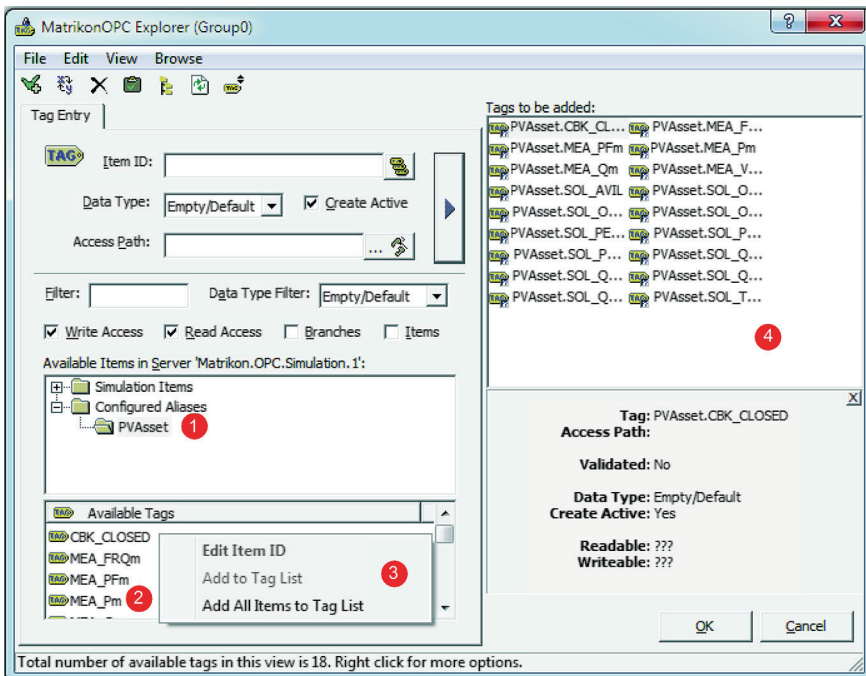


Fig. 16.12 MatrikonOPC server explorer—data tag configuration

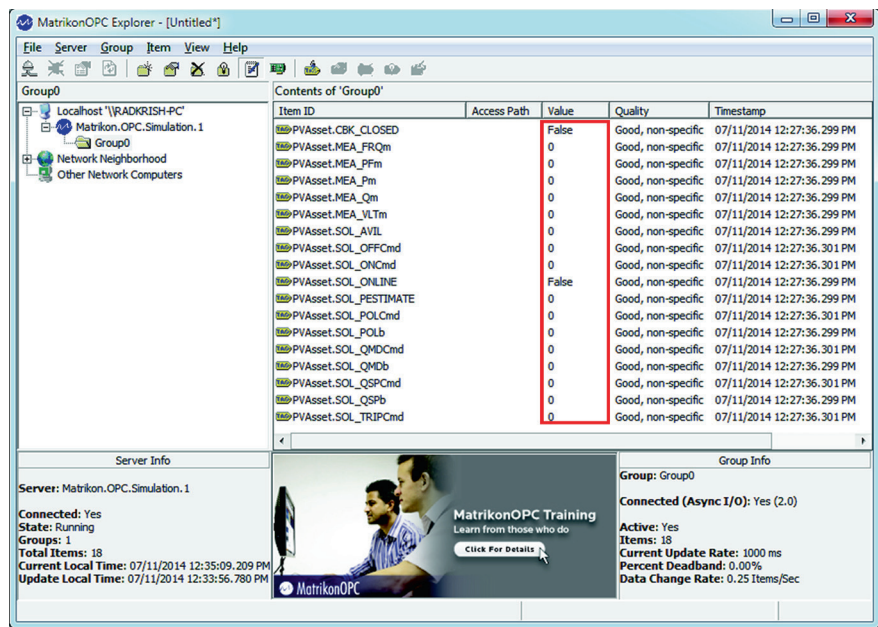


Fig. 16.13 MatrikonOPC server explorer—list of tags for observation and control

16.5.7 Configuring Solar Irradiance and Temperature Profile

As mentioned in the introductory discussion, *ElmFile* objects are often used to feed in system variables that change with respect to time. In the demo, solar irradiance and the temperature are such variables. The package provided with this chapter has two files containing time series data for irradiance and temperature; the names of the files are “*MeasuredSolarIrradianceProfile.txt*” and “*MockedUpTempratureProfile.txt*.” The screenshot in Fig. 16.14 shows the list of *ElmFile* objects used in the sample project and one of its edits showing the parameter to choose the appropriate file.

16.5.8 Initializing the System

This demonstration utilizes the RMS simulation feature in PowerFactory to simulate a real-world PV system. Initialization object and its configuration play an important role in the time domain simulation. The following section of the chapter briefly explains the initialization of the demonstration model and other interesting options available via the initialization object.

1. Ensure that the RMS/EMT simulation toolbox is selected in the toolbar. If it is not selected, use the “*Change toolbox*” drop-down button (marked ❶ in Fig. 16.15) and choose “*RMS/EMT Simulation*” (marked ❷ in Fig. 16.15).

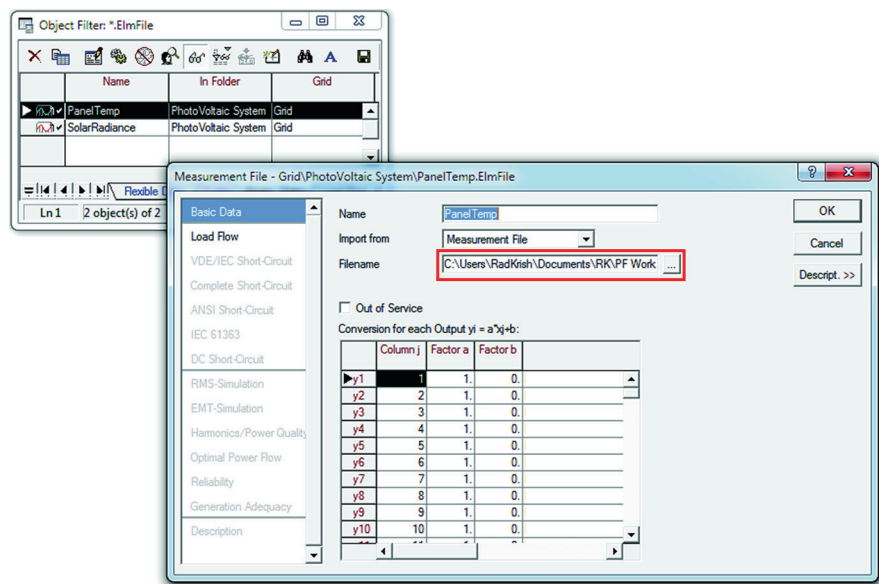
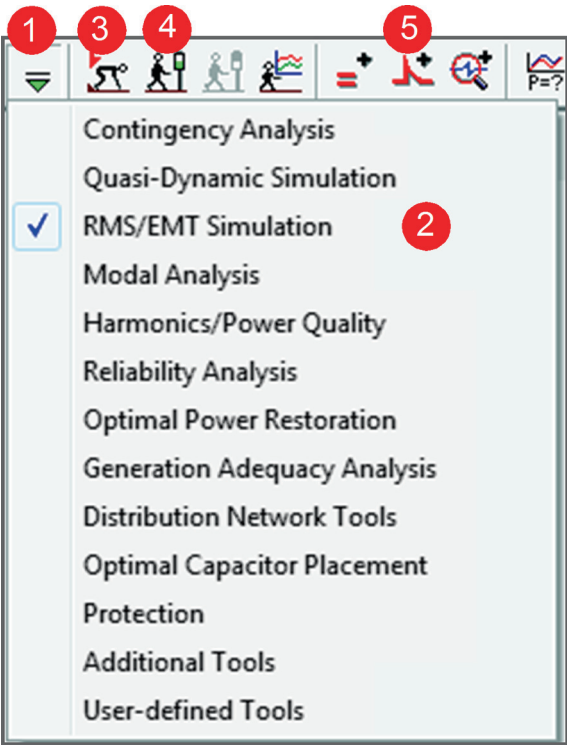


Fig. 16.14 Screenshot showing required *ElmFile* objects and the parameter to configure the time-series.csv file

Fig. 16.15 PowerFactory toolbar showing RMS simulation toolbox



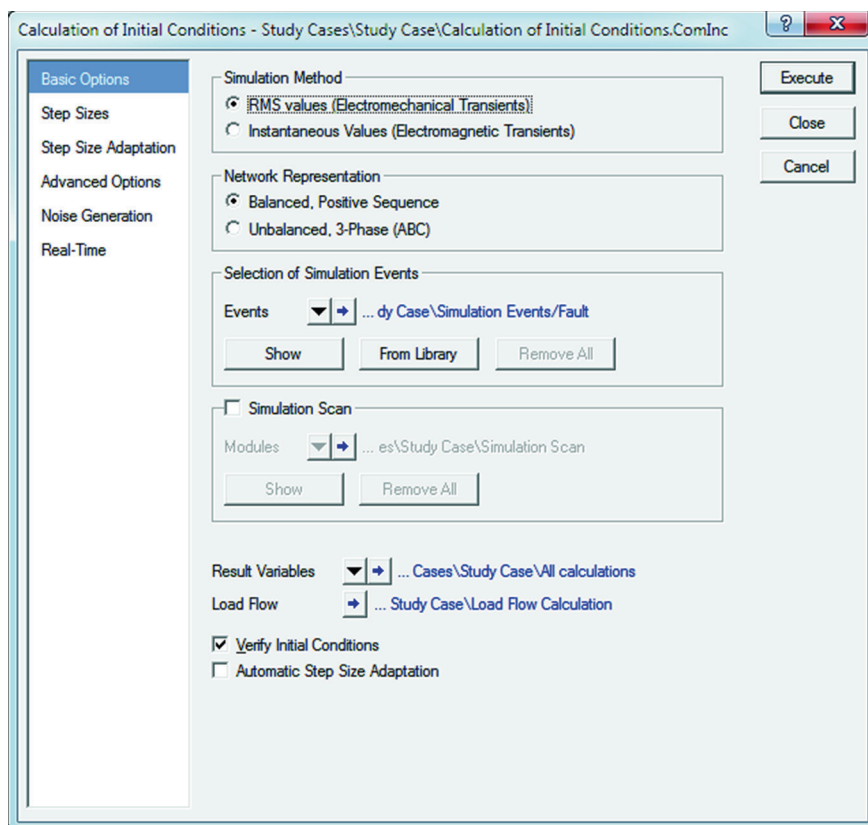


Fig. 16.16 Initialization command editor

2. Ensure that the OPC link object in PowerFactory is linked to the OPC server as explained in earlier sections of this chapter.
3. Click the “Events” button (marked ⑤ in Fig. 16.15) to open the events list created in the previous simulation session and delete all of them (if any).
4. Click the “Initialization” button (marked ⑥ in Fig. 16.15) to bring up the initialization command object editor as shown in Fig. 16.16. Readers are highly encouraged to spend some time investigating various important options or user configurable parameters offered by the initialization objects such as the following:⁹
 - Simulation method (RMS or EMT)
 - Network representation (balanced or unbalanced)
 - Load flows options

⁹ Refer to the PowerFactory user manual for more details about configuration parameters in the initialization screen.

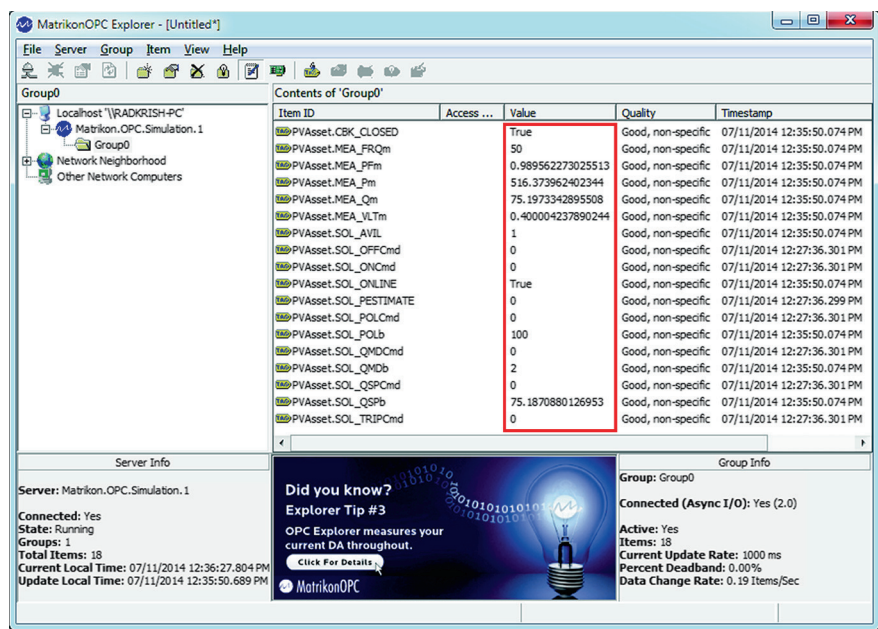


Fig. 16.17 Matrikon OPC server explorer—list of tags after initialization

- Integration step size configuration (setting a bigger step size speeds up the simulation)
 - Simulation start time
 - Real-time synchronization
 - Iteration control parameters
 - OPC read/write intervals, etc.
- Manipulating these parameters opens up numerous possibilities for RMS/EMT simulation in PowerFactory. Technical reference [4] supplied by DIgSILENT explains all the options in detail.
5. Click “Execute” to calculate initial condition.
 6. Note that the “Value” column in the OPC server explorer has started showing the values as communicated by the external measurement objects (see Fig. 16.17).

16.5.9 Running the Simulation

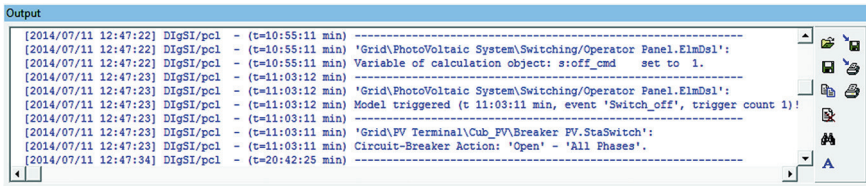
1. Click “Start Simulation” button (marked 4 in Fig. 16.15 to bring up the simulation object editor.
2. Enter the number of seconds for which you need to simulate the system in the text box next to the “Absolute” label in the command editor.
3. Click “Execute” to run the simulation.

4. Switch to “*Virtual Instrument Panel*” in the graphic board to observe the P and Q production plot against time in the x-axis as shown in Fig. 16.21.
5. At the same time, note that the values of the OPC are rapidly changing in real time as the simulation is running (refer Fig. 16.17)

16.6 Controlling the Model Using OPC Server Explorer

Following procedure explains how to control the model using the OPC server explorer (substituting for a real-world power system controller). For the demonstration, the following control actions should be executed from the MatrikonOPC server explorer.

1. Make sure the simulation is running.
2. Look at the OPC server and ensure that the value for “*PVAsset-SOL_OFFCmd*” is reading zero. Right click on the tag point and select “*Write Values*” from the context menu.
3. In the pop-up window, enter 1 (signal—high) for the “*New Value*” and click *OK*. Note the messages in PowerFactory output window, which reads as Fig. 16.18.
4. Also check that both P and Q output go to zero (see marker ❶ in Fig. 16.21).
5. Make sure to reset the value of “*PVAsset-SOL_OFFCmd*” to zero. If you do not, the switch-on control will not close the breakers.

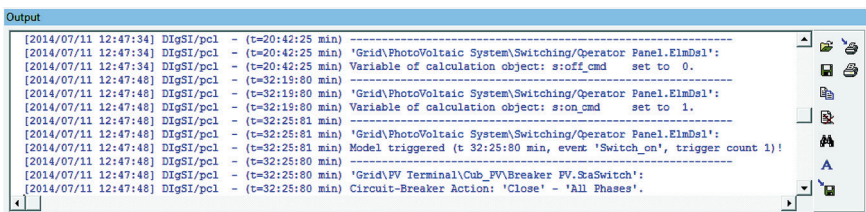


```

[2014/07/11 12:47:22] DigSI/pci - (t=10:55:11 min) -----
[2014/07/11 12:47:22] DigSI/pci - (t=10:55:11 min) 'Grid\PhotoVoltaic System\Switching/Operator Panel.ElmDel':
[2014/07/11 12:47:22] DigSI/pci - (t=10:55:11 min) Variable of calculation object: s:off_cmd set to 1.
[2014/07/11 12:47:23] DigSI/pci - (t=11:03:12 min) -----
[2014/07/11 12:47:23] DigSI/pci - (t=11:03:12 min) 'Grid\PhotoVoltaic System\Switching/Operator Panel.ElmDel':
[2014/07/11 12:47:23] DigSI/pci - (t=11:03:12 min) Model triggered (t 11:03:11 min, event 'Switch_off', trigger count 1)!
[2014/07/11 12:47:23] DigSI/pci - (t=11:03:11 min) -----
[2014/07/11 12:47:23] DigSI/pci - (t=11:03:11 min) 'Grid\FV Terminal\Cub_FV\Breaker FV.StaSwitch':
[2014/07/11 12:47:23] DigSI/pci - (t=11:03:11 min) Circuit-Breaker Action: 'Open' - 'All Phases'.
[2014/07/11 12:47:34] DigSI/pci - (t=20:42:25 min) -----

```

Fig. 16.18 PowerFactory output showing switching-off sequence



```

[2014/07/11 12:47:34] DigSI/pci - (t=20:42:25 min) -----
[2014/07/11 12:47:34] DigSI/pci - (t=20:42:25 min) 'Grid\PhotoVoltaic System\Switching/Operator Panel.ElmDel':
[2014/07/11 12:47:34] DigSI/pci - (t=20:42:25 min) Variable of calculation object: s:off_cmd set to 0.
[2014/07/11 12:47:48] DigSI/pci - (t=32:15:80 min) -----
[2014/07/11 12:47:48] DigSI/pci - (t=32:15:80 min) 'Grid\PhotoVoltaic System\Switching/Operator Panel.ElmDel':
[2014/07/11 12:47:48] DigSI/pci - (t=32:15:80 min) Variable of calculation object: s:on_cmd set to 1.
[2014/07/11 12:47:48] DigSI/pci - (t=32:25:81 min) -----
[2014/07/11 12:47:48] DigSI/pci - (t=32:25:81 min) 'Grid\PhotoVoltaic System\Switching/Operator Panel.ElmDel':
[2014/07/11 12:47:48] DigSI/pci - (t=32:25:81 min) Model triggered (t 32:25:80 min, event 'Switch_on', trigger count 1)!
[2014/07/11 12:47:48] DigSI/pci - (t=32:25:80 min) -----
[2014/07/11 12:47:48] DigSI/pci - (t=32:25:80 min) 'Grid\FV Terminal\Cub_FV\Breaker FV.StaSwitch':
[2014/07/11 12:47:48] DigSI/pci - (t=32:25:80 min) Circuit-Breaker Action: 'Close' - 'All Phases'.

```

Fig. 16.19 PowerFactory output showing switching-on sequence

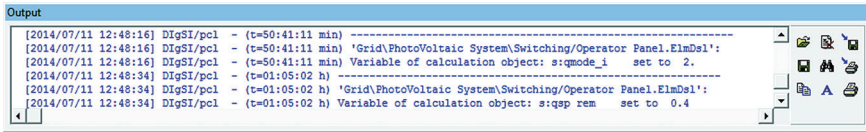


Fig. 16.20 PowerFactory output showing change in Q mode and set point

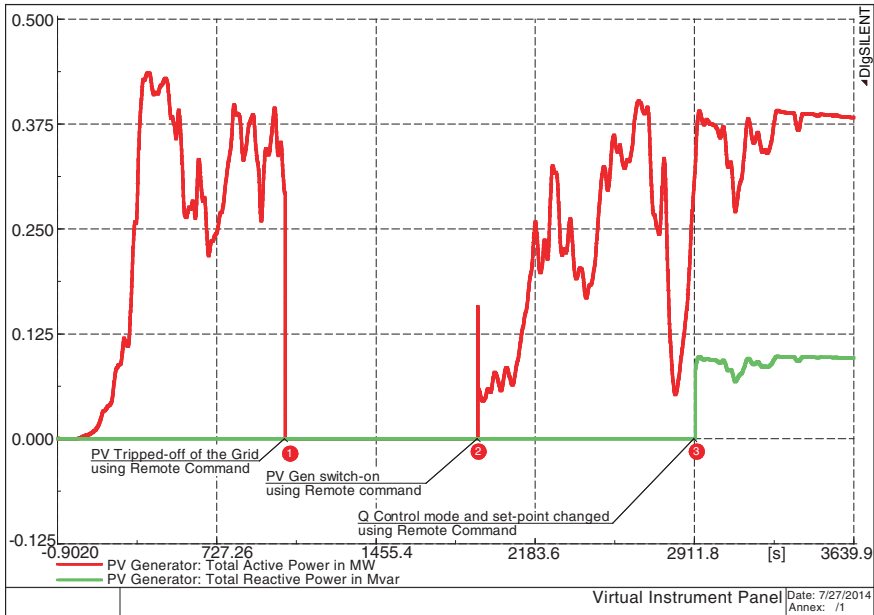


Fig. 16.21 Plots showing P and Q produced by the PV plant in the demonstration system

6. To switch the PV system back into the grid, change the value of “PVAsset_SOL_ONCmd” to 1 (high).
7. Observe the following messages in the output window (see Fig. 16.19) and change in P and Q output of the PV generator (see marker ② in Fig. 16.21).
8. Readers can exercise the Q dispatch control by following the steps explained below:
 - To change the Q dispatch mode, set the value of “PVAsset_SOL_QMDCmd” to 2 using the OPC server explorer as explained. The model understands the number 2 as a command for base-load mode.
 - Next, change the set point (PVAsset_SOL_QSPCmd) to, say, 200 kVAr. The model is capable of converting the number 200 to its P.U. value.

- Check the output window for messages about the change in Q mode and Q set point (see Fig. 16.20).
- P and Q dispatch in the virtual instrument panel also reflects the change in Q mode and Q set point as shown under marker ⑨ in Fig. 16.21.

16.7 Conclusion

Equipped with great features and wide range of calculation and event simulating capabilities, DigSILENT's PowerFactory comes in as a handy substitute to costly hardware in prototype developments and HIL testing. It also stands as a good substitute to super costly real-time digital simulators in simulating relatively smaller power systems. However, users need to take into account the following factors before using PowerFactory for prototyping and HIL testing.

- Modeling system dynamics—accuracy of simulation results compared to its measured values greatly depends on the precision with which the dynamics of the system is represented. However, the more the details in the model, simulations take longer time to solve. Thus, users have to make conscious effort on what is important in terms of system modeling for a given problem.
- The size of the network—size of the network also dictates the simulation speed. In many cases, it is possible to reduce large portions of the given power system and represents them by its simplified equivalents. However users have to be aware of the fact that; replacing a portion of the network with its equivalent can have significant effect on the simulation results.
- Integration step size or time step—step size must be chosen such that the simulation accurately represents the system frequency response up to the fastest transient of interest. Thus, the step size has to be worked out on a case-by-case basis.
- Synchronization between simulation time step and real time step. However, both in the case of controller prototyping and HIL testing, the setup is an off-line setup and thus time synchronization is not an issue.
- PowerFactory's inability to connect to more than one OPC server—a shortcoming which can be overcome by employing third-party OPC tunnelers.

In short, besides many other usages—PowerFactory can go a long way in model-based control system development, prototyping, and testing of power system controllers.

References

1. Bélanger J, Venne P, Paquin J-N (2010) White paper—the what, where and why of real-time simulation. Opal-RT Technologies, Montréal, Oct 2010. Available from http://www.opal-rt.com/sites/default/files/technical_papers/PES-GM-Tutorial_04%20-%20Real%20Time%20Simulation.pdf. Accessed 08 Dec 2014

2. Theologitis IT (2011) Comparison of existing PV models and possible integration under EU grid specifications, Master of Science thesis, KTH School of Electrical Engineering, Division of Electrical Power Systems EPS-2011, SE-100 44 Stockholm, Jun 2011, XR-EE-ES 2011:011
3. Mahmood F Improving Photo-voltaic Model in PowerFactory, Degree Project, KTH School of Electrical Engineering, Division of Electrical Power Systems EPS-2012, SE-100 44 Stockholm, Nov. 2012, XR-EE-ES 2012:017
4. DIgSILENT PowerFactory 15.0 user manual and technical references (Licensed users can download latest user manual and technical references from <http://www.digsilent.de/index.php/downloads.html>)

Chapter 17

Programming of Simplified Models of Flexible Alternating Current Transmission System (FACTS) Devices Using DIgSILENT Simulation Language

Jaime C. Cepeda, Esteban D. Agüero and Delia G. Colomé

Abstract An electric power system (EPS) must meet certain requirements that allow a continuous supply of demand at minimum cost and with the least environmental impact. For this purpose, several system controllers, such as the flexible alternating current transmission system (FACTS) devices, might be integrated into the grid, and so they should be included as part of the power system modelling in simulation programs. The addition of a new device in the transmission grid requires several studies that consider the analysis of the system for a wide variety of normal and stressed operating conditions, including maintenance situations. Thus, the integration of FACTS devices in the transmission network requires the development of static studies and dynamic analysis concerning stability simulation. These types of studies are important for proper system planning and operation. Although DIgSILENT PowerFactory has several dynamic models to represent to a number of power system components, its model library does not include a sufficient variety of dynamic models of FACTS devices. Only the model of the static var system (SVS) is already included in its library. However, in addition to typical functions for power system analysis, DIgSILENT PowerFactory offers versatility to model new

Electronic supplementary material The online version of this chapter (doi:10.1007/978-3-319-12958-7_17) contains supplementary material, which is available to authorized users.

J.C. Cepeda (✉)

Research and Development Department, Corporación Centro Nacional de Control de Energía—CENACE, Av. Atacazo and Panamericana Sur km 0, Quito, Ecuador
e-mail: cepedajaime@ieec.org; jcepeda@cenace.org.ec

E.D. Agüero · D.G. Colomé

Instituto de Energía Eléctrica, Universidad Nacional de San Juan - Consejo Nacional de Investigaciones Científicas y Técnicas, Av. Libertador 1109 (O), San Juan, Argentina
e-mail: eaguero@iee.unsj.edu.ar

D.G. Colomé

e-mail: colome@iee.unsj.edu.ar

components using its programming language DIgSILENT simulation language (DSL). The present chapter describes the methodology for including the following models of FACTS devices into DIgSILENT PowerFactory: thyristor controlled series compensator (TCSC), static series synchronous compensator (SSSC), static synchronous compensator (STATCOM) and unified power flow controller (UPFC). Additionally, power oscillation damper (POD) is also incorporated into the models in order to improve oscillatory conditions. The models that will be described correspond to the so-called simplified models that are designed to be used in stability studies of EPSs (i.e. electromechanical transients). Finally, simulations are performed in a test power system that show the dynamic behaviour of the implemented FACTS devices and allow their corresponding validation.

Keywords DIgSILENT simulation language · FACTS · TCSC · STATCOM · SSSC · UPFC

17.1 Introduction

FACTS technology comprises a set of controllers that offer the possibility of controlling one or more parameters that modify the operation of the transmission systems. These devices can operate stand-alone or coordinated with other control equipment. Among the parameters to be controlled are the series impedances of transmission lines, the parallel or shunt impedance, the line current, the amplitude of the bar voltages and the corresponding angular difference [16]. FACTS devices use power electronics-based converters [16] that allow the control of power flows and the improvement of effectiveness in the usage of transmission lines [14, 16, 18]. Additionally, FACTS devices offer some advantages to control the non-conventional variable energy sources, such as wind farms, further facilitating their integration into the system.

FACTS devices could be shunt, series or a combination of these ones [14]. Shunt controllers can be an adjustable impedance, a variable source or a combination of this equipment. All shunt devices inject current into the system at the connection point, and provided that the injected current phase is in quadrature with the line voltage, this device only supplies or consumes reactive power. Among the shunt controllers are the static var compensator (SVC) and the static synchronous compensator (STATCOM) [14, 18]. Series controllers, as well, can be a variable impedance or a variable source based on power electronics. All of these controllers inject voltage in series with the transmission line, and provided that the injected voltage phase is in quadrature with the line current, these devices only supply or consume reactive power. Among this category are the thyristor controlled series capacitor (TCSC) and the static synchronous series compensator (SSSC) [14, 18].

Combined controllers, such as the unified power flow controller (UPFC), are structured by a combination of series and shunt controllers, commonly linked by

continuous current circuit. These devices allow reactive power compensation between the shunt and series controllers, as well as active power flow interchange [14, 18].

The actual power system behaviour can be predicted using computer-based simulations; the several physical components of the system have to be adequately modelled in order to accurately represent the performance of the power system [3]. One of the key tasks for analysing the effect of including FACTS devices into the power system is their effective simulation. Thus, the implementation of FACTS devices in any type of power system simulation software is of particular concern of researchers, even more when the applications of these devices remain being a challenge for improving the power system steady-state and dynamic performance [9, 10]. Typically, FACTS devices are simulated in applications designed for electromagnetic transient analysis, such as SIMULINK–MATLAB [9, 11, 12]; nevertheless, this type of applications do not allow simulating FACTS immersed in a complex power system. An attractive option is the use of power system software (such as PSAT–MATLAB: [10]) in order to replicate the performance of FACTS devices considering their interaction with power systems. This type of simulation commonly uses simplified models [13]. Although the use of some of these applications can offer a good basis for understanding and research, they are not useful for performing commercial simulations. Therefore, it is extremely necessary to also have FACTS devices modelled in commercial software (such as PSS/E or DIgSILENT PowerFactory).

This chapter describes how TCSC, STATCOM, SSSC and UPFC can be modelled in DIgSILENT PowerFactory in order to allow electromechanical transient simulations (i.e. simplified models) that are useful to carry out stability studies. For this purpose, series capacitors *ElmScap* [5] and reactors *ElmSind* [6], as well as the static generator object *ElmGenstat* [7], are adequately controlled. It is important to clarify that the modelling of SVC is not considered by this chapter, since this model is actually available in the PowerFactory library [8], for which its simplified modelling is not required.

The static generator *ElmGenstat* object of PowerFactory is an easy-to-use model of any kind of static (no rotating) generator [7]. Therefore, it can be used in order to implement FACTS devices in DIgSILENT PowerFactory that are based on “static generators”, such as SSSC, STATCOM or UPFC. On the other hand, TCSC is implemented using an adequate control of series capacitors *ElmScap* and reactors *ElmSind* connected in parallel.

This chapter presents the FACTS devices’ implementation methodology that comprises several stages, including all the computations that allow linking the outputs of the controllers with the inputs of the static generator or the series reactors. Additionally, all the DSL aspects that permit including the FACTS models are also presented.

All the programmed FACTS devices will be tested using the WSCC 3-machine, 9-bus test system [1]. Additionally, in order to show the general guidelines as simple as possible, the simplified controllers presented by Milano [13] are implemented. However, the same procedure might be easily followed in order to implement even more sophisticated controllers.

Following this introduction, the outline of the chapter is as follows: Sect. 2 presents a general introduction to DSL. Section 3 overviews the implementation of FACTS devices in PowerFactory, that is TCSC, STATCOM, SSSC and UPFC. Section 4 shows the implementation behaviour via time domain simulation examples in a test power system. Finally, the main concluding remarks are summarized in Sect. 5.

17.2 DIgSILENT Simulation Language Implementation Steps

DIgSILENT simulation language (DSL) allows programming control models of any power system element, including protection devices, as well as it permits developing other components or routines oriented to run together with the time domain simulations [8]. In order to structure a DSL model, some hierarchical levels of models are used by PowerFactory [8]: *DSL block definitions, built-in models and common models, and composite models*.

To generate the model of a controller of some device or element of a power system capable of running inside the time domain simulation, it is vital to follow the hierarchical relationships between the above-mentioned DSL elements [8].

A *composite model* is a “mask” used to administrate the models associated with a machine or a system which selects all of the models and elements wanting to be related. The *composite model* is represented by means of a *composite frame*, which is a block diagram that interrelates the distinct objects of the control system defined by the composite model. On the other hand, the composite model can be formed by a *common model*, measuring devices and the network elements requiring control.

A *common model* combines general models in the time domain or equations with a set of parameters and creates an integrated model in the time domain. The *common model* is graphically represented by means of a *block diagram* that includes the transfer functions and control system equations to be implemented. The *block diagram* must also be composed by *macros* representing the distinct control system transfer functions.

These *macros* allow for definition of input signals, output signals, parameters, internal variables, state equations, state variables and minimum/maximum limits.

In the *macros*, contrary to the *composite block diagram*, the initial conditions remain undefined.

The DSL FACTS devices implementation methodology includes four stages, detailed as follows:

Stage 1: *DSL programming*

First, the appropriate *composite frame* that relates the interaction between all the objects (measurements, controllers and objects to be controlled) has to be depicted in a *block/frame diagram* page. After, all the necessary controllers have also to be structured in a *block/frame*

diagram page (one per each controller). These controllers can be easily implemented using the PowerFactory graphical interface [8].

Stage 2: *Model initialization*

Initializing each model is a fundamental task in order to allow the correct link between the controllers and the active grid. For proper initialization of models, it is necessary to represent the system state space from the block diagrams of the control system (i.e. from the transfer functions to obtain the state equations and output). After obtaining the state equations, the initial value of the state variables has to be determined considering that the derivatives of all state variables are zero in steady state.

Stage 3: *Linking the outputs of FACTS controllers to physical grid objects*

This step consists on determining those mathematical expressions that relate the physical magnitudes of the system, which are modified by the control signal from the FACTS model. These expressions are determined in order to link the outputs of the different FACTS controllers with the input of the static generator *ElmGenstat* object (in case of SSSC, STATCOM or UPFC) or to control the series reactor *ElmSind* (in case of TCSC). Details of this stage will be given in Sect. 3.

Stage 4: *Interfacing controllers and active grid objects*

The final step of DSL programming is to properly link the programmed controllers into the active grid. For this purpose, the block diagrams of controllers have to be masked as a *common models* and the composite frame as *composite model*. Every object related to the control strategy has to be adequately included in each slot of the *composite model*. The detailed explanation of these stages can be found in [8].

17.3 FACTS Implementation

Implementation of FACTS models requires determining the mathematical expressions that relate the physical magnitudes of the system objects that are modified by the control signals of each FACTS model.

The FACTS controllers for stability studies are represented by simplified models that capture the corresponding controller response to the fundamental frequency. This type of modelling assumes simplifications, such as not modelling the power electronics converter and not modelling the converter firing control systems [2, 17]. These types of simplified models are implemented via DSL.

The selection of the models representing FACTS in stability studies is firstly necessary to be specified in order to study the main features of each device and analyse the improvements introduced to the system [14–16]. Therefore, the simple controllers used by Milano [13] are employed to illustrate the general guidelines for FACTS implementation depicted in this section.

The following guidelines describe the implementation of TCSC, STATCOM, SSSC and UPFC. Implementation of SVC is not described since its simplified model is already available in the PowerFactory library [8].

17.3.1 Thyristor Controlled Series Capacitor (TCSC)

The TCSC operates as a series controlled reactance, whose objective is to compensate the transmission line impedance. The high switching speed of the TCSC provides a mechanism to control the power flow in the transmission lines, which allows increasing the load with the existing grid and the damping in the interconnection of large power systems, and also provides the possibility of quick power flow adjustment in response to several contingencies that may occur in the system. The TCSC can also regulate the steady-state power flow to keep it within the system physical limits [14, 18].

The TCSC can be modelled as a variable reactance (X_{TCSC}) in series with the transmission line. The variation of X_{TCSC} allows regulating the active power flow through the line, increasing the damping of power oscillations, among other benefits.

The modelling of a TCSC can be composed by different control types which allow the following functions [18]:

- Control of active power flow through the line and enhance the small signal stability of the system.
- Improvement of transient stability.
- Limiting short-circuit currents.

From these types of control strategies, the controller that allows the power flow control is modelled in this section [13]. Figure 17.1a illustrates a block diagram of a control system designed for controlling power flows and increasing the damping, which is presented by Milano [13]. This diagram consists of a PI (Proportional-integral) controller and a first-order delay. The purpose of this control is to modify the reactance of the transmission line in order to maintain a constant power flow.

The output signal of the block diagram of the TCSC is its variable susceptance (b_{TCSC}), whose expression, as stated by Milano [13], is:

$$b_{TCSC}(x_c) = -\frac{x_c/x_{km}}{x_{km} \cdot (1 - x_c/x_{km})} \quad (17.1)$$

where P_{ref} is the active power reference, P_{km} is the active power transmitted by the line between nodes k and m, u_{POD} is the power oscillation damper (POD) signal, b_{TCSC} is the susceptance of the TCSC, x_c is the control signal (output of the first-order delay), x_{km} is the line reactance between nodes k and m, K_p is the proportional gain of the PI controller, K_I is the integral gain of the PI controller and T_r is the first-order delay time constant.

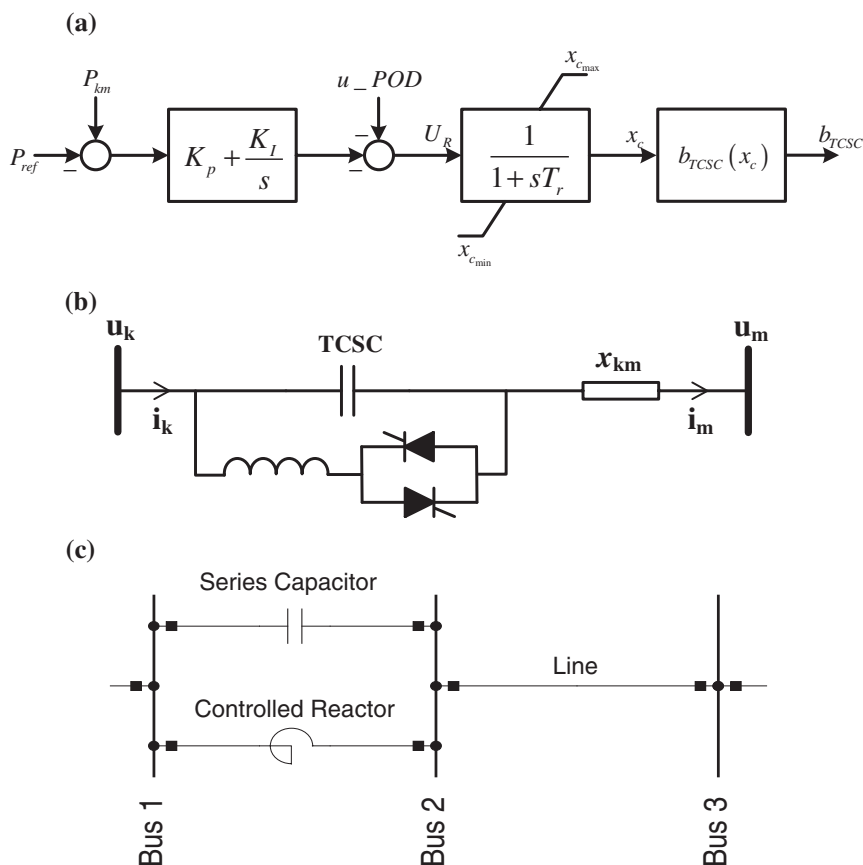


Fig. 17.1 Implemented TCSC: **a** control block diagram, **b** connectivity scheme, **c** implementation in PowerFactory

The operation of the TCSC control system permits measuring the active power flow through the line (P_{km}) and comparing it with the active power reference (P_{ref}). Thus, when P_{km} is higher than P_{ref} , the control reactance x_c is negative, which means that b_{TCSC} is positive. This means that the equivalent reactance of the line increases, causing a decrease of the transmitted active power (simulating the effect of increasing the line length). On the contrary, if P_{km} is lower than P_{ref} , then x_c is positive, which means that b_{TCSC} is negative; therefore, the equivalent reactance of the line decreases causing an increase in the transmitted power (behaves as if the line length be shortened).

TCSC implementation in DIgSILENT PowerFactory is performed through a fixed series capacitor in parallel with a controlled series reactor (thyristor controlled reactor—TCR). For this configuration, the variable to be controlled is the value of

the input reactance of the series reactor *ElmSind* (x_{in}) given in ohm [6]. Figure 17.1b shows the TCSC scheme in series with the line, whereas Fig. 17.1c depicts the implementation in PowerFactory.

Considering the scheme of Fig. 17.1b, the total branch reactance (x_T) is as follows:

$$x_T = x_{km} + x_{TCSC} \quad (17.2)$$

Besides, the x_{TCSC} reactance is the parallel equivalent between the capacitor and reactor reactances, and its expression is as follows:

$$j \cdot x_{TCSC} = -j \frac{1}{b_{TCSC}} = \frac{(-j \cdot x_{cap}) \cdot (j \cdot x_{in})}{j \cdot x_{in} - j \cdot x_{cap}} = \frac{x_{cap} \cdot x_{in}}{j(x_{in} - x_{cap})} = -j \frac{x_{cap} \cdot x_{in}}{(x_{in} - x_{cap})} \quad (17.3)$$

where x_{cap} is the reactance of the fixed capacitor *ElmScap*, x_{in} is the controlled reactance of variable reactor *ElmSind* and b_{TCSC} is the total TCSC susceptance ($x_{TCSC} = 1/b_{TCSC}$).

From (17.3), and considering that x_{cap} is a priori defined (i.e. the fixed capacitor does not change its reactance, which is appropriately chosen during the design), it is possible to determine the expression that relates x_{in} as a function of b_{TCSC} .

$$x_{in} = - \frac{x_{cap}}{b_{TCSC} \cdot x_{cap} - 1} \quad (17.4)$$

Since the reactor in parallel with the fixed capacitor might eventually make up together a resonance circuit, it is necessary to ensure that the x_{TCSC} does not surpass determined minimum and maximum thresholds. Thus, there is an operating zone where x_{TCSC} might acquire extremely high values (i.e. parallel resonance), which necessarily has to be avoided (i.e. not-allowed operating zone). Consequently, the requirement of limiting the TCSC operating zone, so that $x_{TCSC} \in [x_{TCSCmin}, x_{TCSCmax}]$, has to be considered in the TCSC controller design. This fact highlights the requirement of specifying a limit of line compensation.

Therefore, in order to ensure the system stability, it is recommended to compensate up to a maximum compensating fraction (CF) of the line reactance nominal value (x_{km}), both in the capacitive (CF_{cap}) and inductive (CF_{ind}) TCSC operating zones. Therefore, x_{TCSC} will be in the range given by (17.5).

$$-CF_{cap} \cdot x_{km} \leq x_{TCSC} \leq CF_{ind} \cdot x_{km} \quad (17.5)$$

Based on system experiences, a rational CF might be 0.3 (30 % of x_{km}), which is used in the example shown in Sect. 4.

From (17.5), it is easy to determine that the TCSC susceptance b_{TCSC} will satisfy the following relations:

$$b_{\text{TCSC}} \leq -\frac{1}{CF_{\text{ind}} \cdot x_{km}} \quad \vee \quad b_{\text{TCSC}} \geq \frac{1}{CF_{\text{cap}} \cdot x_{km}} \quad (17.6)$$

Replacing (17.1) in (17.6), and after solving the inequations, the range of x_c (output of the first-order delay) can be determined as follows:

$$\frac{x_{km}}{1 + CF_{\text{ind}}} \leq x_c \leq \frac{x_{km}}{1 - CF_{\text{cap}}} \quad (17.7)$$

Therefore, the limits of the first-order delay $x_{c\text{max}}$ and $x_{c\text{min}}$ are as follows:

$$x_{c\text{min}} = \frac{x_{km}}{1 + CF_{\text{ind}}} \quad \wedge \quad x_{c\text{max}} = \frac{x_{km}}{1 - CF_{\text{cap}}} \quad (17.8)$$

These last presented limits allow ensuring the line compensation does not exceed the pre-defined thresholds, and also, the TCSC actually operates outside the not-allowed zone.

As explained beforehand, in DIGSILENT PowerFactory, the TCSC is implemented via the adequate control of the input reactance of the series reactor *ElmSind* (x_{in}) [6]. Thus, x_{in} is the signal to be controlled (i.e. the required output of the DSL model of the controllers). Then, it is necessary to replace the last block of the controllers (i.e. the block that computes b_{TCSC} from the control signal x_c , shown in Fig. 17.1a) by a block that allows establishing the relationship between the control signal x_c and x_{in} (i.e. the available signal to be controlled in the PowerFactory object *ElmSind*). For this purpose, the existence equivalency between (17.1) and the inverse of (17.3) can be used, as shown by (17.9).

$$-\frac{x_c/x_{km}}{x_{km} \cdot (1 - x_c/x_{km})} = \frac{x_{\text{in}} - x_{\text{cap}}}{x_{\text{cap}} \cdot x_{\text{in}}} \quad (17.9)$$

After several simplifications, it is possible to obtain from (17.9), the expression that relates x_{in} with the control signal x_c , as follows:

$$x_{\text{in}} = \frac{x_{\text{cap}} \cdot x_{km} - x_{\text{cap}} \cdot x_c}{\frac{x_{\text{cap}} \cdot x_c}{x_{km}} + x_{km} - x_c} \quad (17.10)$$

This last expression is introduced in a block called *TCSC-Xin Interface*, as can be seen in Fig. 17.2a, that replaces the last block presented in Fig. 17.1a, i.e. $b_{\text{TCSC}}(x_c)$. It is worth mentioning that, while x_{cap} , x_{km} and x_c are in per unit, x_{in} has to be in ohm (since it is the signal specification of the reactor *ElmSind* reactance in PowerFactory); thus, (17.10) has to be additionally multiplied by the base impedance (z_{base}).

Using the previous presented relationships, the block diagram *TCSC_controller.BlkDef* (Fig. 17.2a), representing the controller of TCSC, included in the *User Defined Models* library of *FACTS Project*, is built via DSL. This block diagram

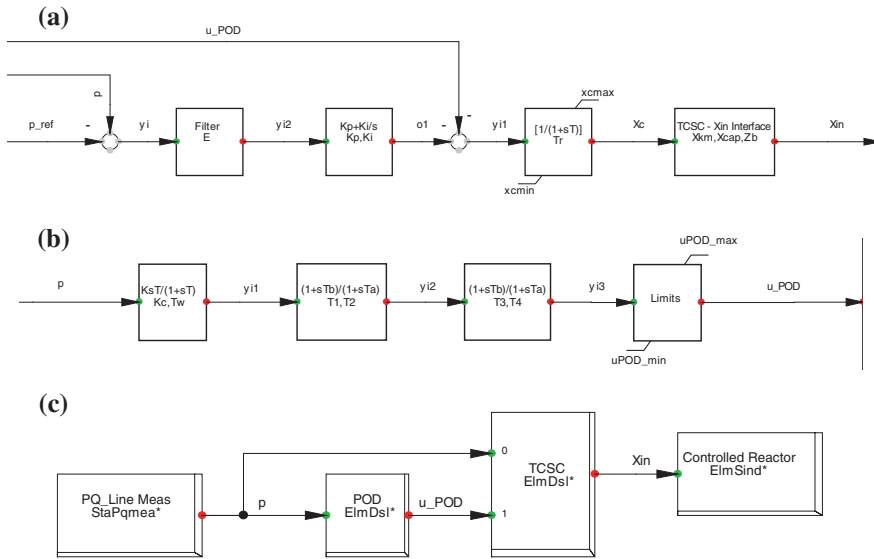


Fig. 17.2 DSL implementation of TCSC: **a** TCSC controller, **b** POD controller, **c** TCSC composite frame

includes an additional *filter* whose function is to prevent the problems of numerical integration errors occur during the search for steady-state value.

Initialization equations representing the TCSC model (included in the DSL block) are determined by considering that all the state variables (x) satisfy $dx/dt = 0$. By applying this consideration to each transfer function, a set of algebraic equations can be determined and solved. The TCSC initialization equations are as follows:

```
inc(x1)=-((Xin*Xkm)/Zb)+(Xcap*Xkm)/(Xcap+((Xin*Xcap)/(Zb*Xkm)))-(Xin/Zb)
inc(x2)=-x1
inc(p_ref)=p
inc(u_POD)=0

vardef(Xkm)='pu';'Line reactance'
vardef(Xcap)='pu';'Reactance of the fixed capacitor'
vardef(Zb)='ohm';'Base impedance'
vardef(E)='pu';'Series filter error'
vardef(Kp)='pu';'PI controller proportional gain'
vardef(Ki)='pu';'PI controller integral gain'
vardef(Tr)='s';'First order delay time constant'
vardef(xcmin)='pu';'Minimum limit of the first order delay'
vardef(xcmax)='pu';'Maximum limit of the first order delay'
```

Furthermore, a control loop belonging to the POD is also depicted in *TCSC_controller.BlkDef* (i.e. u_{POD} input). The POD function is to allow the power oscillations' damping increase during the dynamic response, whose parameters have to be adequately determined depending on the system

characteristics (this task is out of the present chapter scope, so typical parameter values are selected).

The POD employed in this chapter is a simple linear controller which consists of a washout filter, a dynamic lead–lag compensator of one or more stages, and a static limiter.

The washout filter serves to prevent the controller to response in the face of steady-state variations of input signal, i.e. to allow the controller only acts when there are transient perturbations. The washout filter used in this controller is a high-pass filter. The dynamic compensator consists of one or more lead–lag blocks providing the necessary phase lead to increase the synchronizing and damping torques. Finally, the static limiter restricts the control signal within the permissible range of the equipment operation. Figure 17.2b presents the DSL implementation of the POD (*POD.BlkDef*). The block diagram *POD.BlkDef* is included in the *FACTS Project*. For the TCSC case, the POD input signal is the active power transmitted by the line.

Initialization equations representing the POD model are as follows:

```
inc(x1)=p
inc(x2)=0
inc(x3)=0
inc(u_POD)=0

vardef(Kc)='pu'; 'Stabilizer gain'
vardef(Tw)='s'; 'Washout time constant'
vardef(T1)='s'; 'Phase compensation time constant 1'
vardef(T2)='s'; 'Phase compensation time constant 2'
vardef(T3)='s'; 'Phase compensation time constant 3'
vardef(T4)='s'; 'Phase compensation time constant 4'
vardef(uPOD_min)='pu'; 'Output minimum limit'
vardef(uPOD_max)='pu'; 'Output maximum limit'
```

The composite frame that allows the TCSC control (*TCSC_Frame.BlkDef*) is composed by slots representing the line power measurements, the DSL block diagrams of TCSC and POD controllers, and the controlled reactor (note that the fixed capacitor is not included in the frame since it does not change its capacitance). Figure 17.2c presents the DSL implementation of the TCSC composite frame.

Finally, the corresponding TCSC composite model is structured, and each EPS object, power measurement devices (*Stapqmea*) and DSL common models *ElmDsl* (belonging to TCSC and POD block diagrams) are adequately addressed.

17.3.2 Static Synchronous Compensator (STATCOM)

The STATCOM operates as a controlled source of reactive power (it is also possible to exchange active power using an energy storage device, but this operation is not considered in the presented implementation). The main feature of STATCOM is to provide voltage support without using large banks of capacitors and reactors in

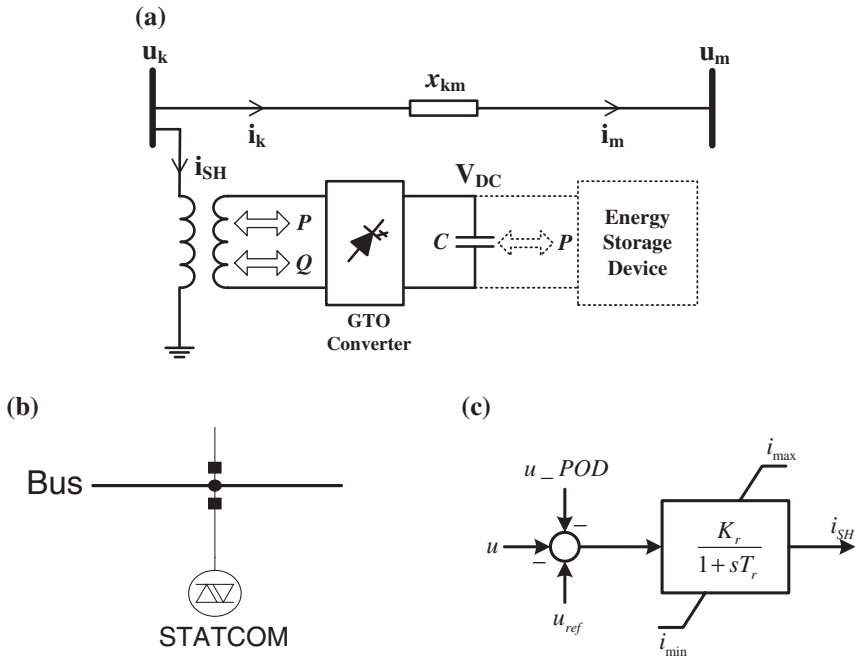


Fig. 17.3 Implemented STATCOM: **a** connectivity scheme, **b** implementation in PowerFactory, **c** control block diagram

order to supply or absorb reactive power (which is the limitation of SVC). The delivered or received reactive power is developed within the STATCOM, which is able to generate a synchronous voltage at its terminals using GTO converters and direct current (DC) energy storage. Thus, a STATCOM has the ability to control its capacitive or inductive current independently of system voltage [14, 18]. Figure 17.3a depicts an illustrative scheme of the STATCOM basis, whereas Fig. 17.3b shows the implementation in PowerFactory.

The STATCOM can be modelled as a controlled current source that injects or consumes reactive power in the bar to which it is connected.

The modelling presented in this chapter is oriented to implement a STATCOM without active power exchange with the network, i.e. there is only reactive power exchange with the EPS. According to this precept, the source must be capable of injecting a current that is in quadrature with the bus voltage.

When the bus voltage is below the reference voltage, the STATCOM will absorb a current that leads 90° voltage. In this way, the STATCOM injects reactive power into the network to increase the voltage, presenting a capacitive behaviour.

On the other hand, when the bus voltage is above the reference voltage, the STATCOM must absorb a current that lags 90° voltage. Thus, the STATCOM consumes reactive power from the grid in order to decrease the bus voltage. In this case, the STATCOM presents an inductive behaviour.

In this chapter, a simple STATCOM controller is used to illustrate the procedure of including the model in DIgSILENT PowerFactory. Figure 17.3c shows the controller employed to model the dynamic performance of the STATCOM. This simple controller is proposed by Milano [13].

Where K_r is the controller gain, T_r is the delay time constant of the controller, u_{POD} is the output signal of the POD, u_{ref} is the controller reference voltage, u is the controlled bus voltage, i_{SH} is the current injected by the STATCOM in pu, and i_{max} and i_{min} are the maximum and minimum values of current in pu.

In the case where u and u_{ref} are equal, the output signal i_{SH} is zero. This means that the STATCOM does not inject, neither consume current from the network, because the system does not need to be compensated.

The STATCOM is modelled in DIgSILENT PowerFactory through an element of the library called *Static Generator ElmGenstat* [7]. The static generator is an item very easy to use, whose applications are photovoltaic generators, fuel cells, storage devices, reactive power compensators, HVDC terminals and wind generators [7]. The basic data required by a static generator as well as further explanation about this object can be found in [7].

The basic data to be set in the *static generator ElmGenstat* are the rated apparent power, which depends on the reactive power to compensate in the bus.

For time domain simulations, the *static generator* supports two types of modelling: (i) current source model and (ii) voltage source model [7].

Since the simplified model of a STATCOM is represented by a current source, the first type of modelling is chosen. The equivalent circuit model of current source can be observed in [7], where the control signals are i_{d_ref} (axis d reference current in pu) and i_{q_ref} (axis q reference current in pu).

The following equations allow representing the behaviour of the *static generator ElmGenstat* when operates as a current source, as shown by [7].

$$\bar{i}_1 = (i_{d_ref} \cdot \cos(u) - i_{q_ref} \cdot \sin(u)) + j \cdot (i_{d_ref} \cdot \sin(u) + i_{q_ref} \cdot \cos(u)) \quad (17.11)$$

$$\begin{aligned} \cos(u) &= \frac{u_r}{u} \\ \sin(u) &= \frac{u_i}{u} \end{aligned} \quad (17.12)$$

where $\bar{u}_1 = u \cdot (\cos(u) + j \cdot \sin(u)) = u_r + j \cdot u_i$ is the complex voltage at controlled bus, \bar{i}_1 is the complex current that STATCOM injects into the network, u_r is the real component of bus voltage, u_i is the imaginary component of bus voltage, u is the module of bus voltage, i_r is the real component of current and i_i is the imaginary component of current.

Thus, (17.11) can be expressed by the following equation:

$$\bar{i}_1 = \left(i_{d_ref} \cdot \frac{u_r}{u} - i_{q_ref} \cdot \frac{u_i}{u} \right) + j \cdot \left(i_{d_ref} \cdot \frac{u_i}{u} + i_{q_ref} \cdot \frac{u_r}{u} \right) \quad (17.13)$$

Based on the apparent power flow relationship at the STATCOM connection bus, it is possible to determine the following expression:

$$\begin{aligned} \bar{S} &= \bar{U} \cdot \bar{I}^* = (u_r + j \cdot u_i) \cdot (i_r - j \cdot i_i) \\ &= (u_r \cdot i_r + u_i \cdot i_i) + j \cdot (u_i \cdot i_r - u_r \cdot i_i) = P + j \cdot Q \end{aligned} \quad (17.14)$$

Thus, the active and reactive powers can be represented as follows:

$$P = u_r \cdot i_r + u_i \cdot i_i \quad (17.15)$$

$$Q = u_i \cdot i_r - u_r \cdot i_i \quad (17.16)$$

As the STATCOM modelled in this section does not allow the active power exchange with the network, this means that $P = 0$. Then, solving i_i from (17.15) is as follows:

$$i_i = -\frac{u_r \cdot i_r}{u_i} \quad (17.17)$$

Also, the magnitude of the current that STATCOM injects ($i_{SH} = i_1$) is as follows:

$$\begin{aligned} i_{SH}^2 &= i_r^2 + i_i^2 = i_r^2 + \left(-\frac{u_r \cdot i_r}{u_i} \right)^2 = i_r^2 + \frac{u_r^2 \cdot i_r^2}{u_i^2} = i_r^2 \cdot \left(1 + \frac{u_r^2}{u_i^2} \right) = i_r^2 \cdot \left(\frac{u_i^2 + u_r^2}{u_i^2} \right) \\ i_{SH}^2 &= \frac{i_r^2 \cdot u^2}{u_i^2} \Rightarrow i_{SH} = \frac{i_r \cdot u}{u_i} \end{aligned} \quad (17.18)$$

Therefore, it is possible to determine expressions that relate i_r and i_i with i_{SH} as follows:

$$i_r = i_{SH} \cdot \frac{u_i}{u} \quad (17.19)$$

$$i_i = -i_{SH} \cdot \frac{u_r}{u} \quad (17.20)$$

Using the equality between (17.19) and (17.20), respectively, with the real and imaginary part of (17.13), a system of two variable equations is structured, whose solution determines $\{i_{d_ref}, i_{q_ref}\}$ as functions of i_{SH} . These relations allow structuring the interface between the *static generator* *ElmGenstat* control signals and the output of the STATCOM controller.

$$i_{q_ref} = -i_{SH} \quad (17.21)$$

$$i_{d_ref} = 0 \quad (17.22)$$

These relations have to be adequately incorporated in an extra block of the control diagram. Thus, the complete PowerFactory Block Diagram *STATCOM_controller.BlkDef* is in the User Defined Models of *FACTS Project*. This block diagram is structured by two blocks and has two input signals (u , and u_{POD}) and two output signals (i_{q_ref} and i_{d_ref}). Figure 17.4a shows the STATCOM controller implemented in DSL.

Initialization equations representing the STATCOM controller are as follows:

```
inc(x)=-iq_ref
inc(u_ref)=- (iq_ref/Kr)+u
inc(u_POD)=0

vardef(Kr)='pu'; 'Controller gain'
vardef(Tr)='s'; 'Delay time constant'
vardef(i_min)='pu'; 'Minimum current'
vardef(i_max)='pu'; 'Maximum current'
```

The composite frame of the STATCOM (*STATCOM_Frame.BlkDef*) consists of two slots that represent the voltage measurement devices, whose signals are connected to the POD and STATCOM DSL controllers, which finally connects to the static generator. Figure 17.4b depicts the STATCOM composite frame in DSL.

Finally, the corresponding STATCOM composite model is structured, and each EPS object, voltage measurement devices (*StaVmea*) and DSL common models *ElmDsl* (belonging to STATCOM and POD block diagrams) are adequately addressed.

17.3.3 Static Synchronous Series Compensator (SSSC)

As STATCOM is an improvement of SVC, the SSSC is the evolution of the TCSC. In this context, SSSC also operates as a controlled source of reactive power (this device can also exchange active power using an energy storage system, but this operation is not considered in the presented implementation). SSSC is also based on GTO technology and has the ability to control its capacitive or inductive voltage independently of the line current [14, 18]. Figure 17.5a depicts an illustrative scheme of the SSSC basis, where its series connection with the transmission line through a coupling transformer (i.e. booster transformer) [4] is also shown. Figure 17.5b, instead, shows the connection of the grid objects in PowerFactory.

The SSSC can be modelled as a controlled voltage source in series with the transmission line, whose output is a variable magnitude voltage that is in quadrature of phase with the line current, so that it only can inject or absorb reactive power

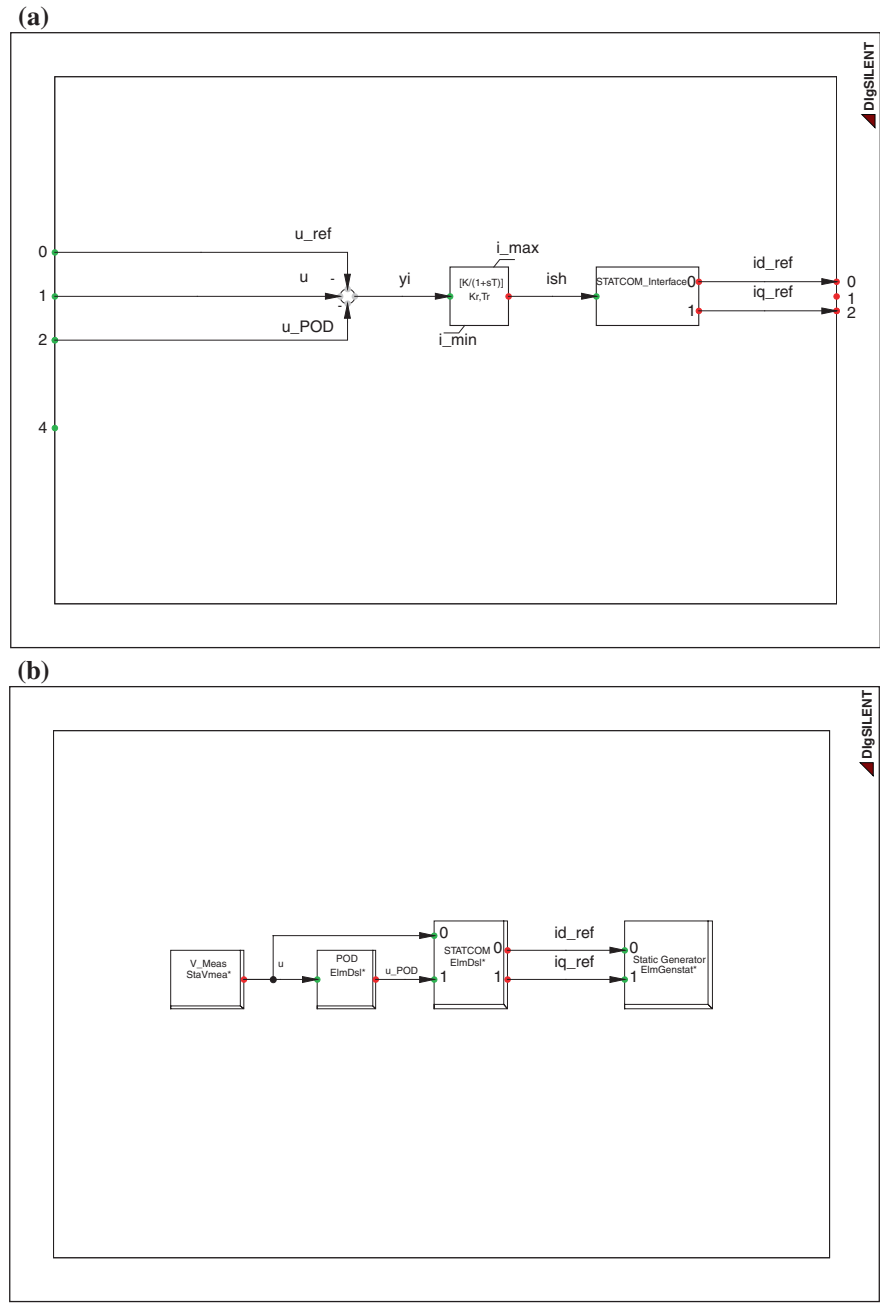
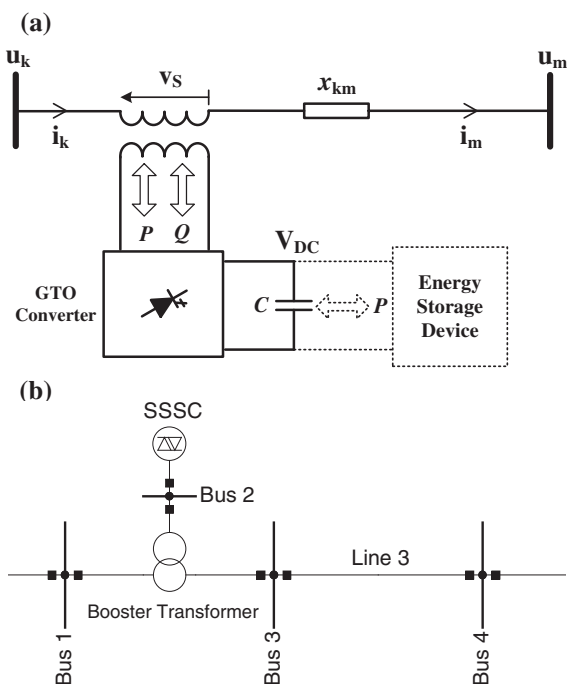


Fig. 17.4 DSL implementation of STATCOM: **a** STATCOM controller, **b** STATCOM composite frame

Fig. 17.5 Implemented SSSC: **a** connectivity scheme, **b** implementation in PowerFactory



from the grid. Unlike the TCSC, the injected voltage is independent of the magnitude of the line current. Thus, the *static generator* *ElmGenstat* is also employed in order to model the simplified SSSC in DigSILENT PowerFactory.

The control scheme is the same as the TCSC controller presented in Fig. 17.1a, whose function is to control the line power flow. For this purpose, the controller measures the power transmitted by the line (P_{km}) and compares it with P_{ref} . If $P_{km} > P_{ref}$, the output v_s is negative; therefore, the SSSC absorbs reactive power from the line, allowing the line congestion and the correspondent decrease of transmitted active power. On the contrary, if the output v_s is positive, so the SSSC produces reactive power that allows the line decongestion, increasing the transmitted active power.

To implement the SSSC in DigSILENT PowerFactory, the *booster transformer* *ElmTrb* object, which allows the connection of the variable voltage source in series with the transmission line, is used. This device is a two-winding transformer, where one of them is connected in series with the transmission line and the other is connected to the voltage source. In order to simplify the modelling of the booster transformer, the values of its reactance and resistance are equal to zero since they are taken at the voltage source. Furthermore, the transformer ratio equals one.

As stated before, the simplified model of a SSSC is represented by a voltage source; then, the second type of modelling the *static generator* *ElmGenstat* is chosen (i.e. as a controlled voltage source). The equivalent circuit model of voltage

source can be found in [7], where the control signals are $u1r_in$ (real part of voltage control signal in pu) and $u1i_in$ (imaginary part of voltage control signal in pu).

The following equations allow representing the behaviour of the static generator when operates as a voltage source as shown by [7].

$$u1r_in + j \cdot u1i_in = \bar{u}1 + \bar{i}1 \cdot \bar{z} \quad (17.23)$$

$$\bar{z} = R + j \cdot X$$

$$\bar{u}1 = u_r + j \cdot u_i \quad (17.24)$$

$$\bar{i}1 = i_r + j \cdot i_i$$

where R and X are the resistance and reactance of the voltage source, $\bar{u}1$ is the complex voltage at the terminals of the controlled source, $\bar{i}1$ is the injected current phasor, u_r is the real component of terminal voltage, u_i is the imaginary component of terminal voltage, v_s is the amplitude of the injected voltage, i_r is the real component of the current, i_i is the imaginary component of the current and $i1$ is the current amplitude.

Considering that the voltage source does not present active power losses (i.e. $R = 0$), (17.23) is simplified as follows:

$$u1r_in + j \cdot u1i_in = (u_r + j \cdot u_i) + (-i_i \cdot X + j \cdot i_r \cdot X) \quad (17.25)$$

Based on the apparent power flow relationship at the SSSC connection bus (i.e. LV terminal of booster transformer *ElmTrb*), it is possible to determine the following expression:

$$\begin{aligned} \bar{S} &= \bar{U} \cdot \bar{I}^* = (u_r + j \cdot u_i) \cdot (i_r - j \cdot i_i) \\ &= (u_r \cdot i_r + u_i \cdot i_i) + j \cdot (u_i \cdot i_r - u_r \cdot i_i) = P + j \cdot Q \end{aligned} \quad (17.26)$$

Thus, the active and reactive powers can be represented as follows:

$$P = u_r \cdot i_r + u_i \cdot i_i \quad (17.27)$$

$$Q = u_i \cdot i_r - u_r \cdot i_i \quad (17.28)$$

As the considered SSSC mode does not allow the active power exchange with the grid, thus, $P = 0$. Then, solving u_r from (17.27) is as follows:

$$u_r = -\frac{u_i \cdot i_i}{i_r} \quad (17.29)$$

Also, the magnitude of the voltage that STATCOM injects (v_S) is as follows:

$$\begin{aligned} v_S^2 &= u_r^2 + u_i^2 = \frac{u_i^2 \cdot i_i^2}{i_r^2} + u_i^2 = u_i^2 \cdot \left(\frac{i_i^2}{i_r^2} + 1 \right) = u_i^2 \cdot \left(\frac{i_i^2 + i_r^2}{i_r^2} \right) = u_i^2 \cdot \frac{i^2}{i_r^2} \\ \Rightarrow v_S &= u_i \cdot \frac{i}{i_r} \end{aligned} \quad (17.30)$$

Therefore, it is possible to determine expressions that relate u_r and u_i with v_S as follows:

$$u_r = -v_S \cdot \frac{i_i}{i} \quad (17.31)$$

$$u_i = v_S \cdot \frac{i_r}{i} \quad (17.32)$$

By replacing (17.31) and (17.32) in (17.25), it is possible to determine the inputs of the static generator functioning as a controlled voltage source $\{u_{lr_in}, u_{li_in}\}$ as functions of v_S . These relations allow structuring the interface between the static generator control signals and the output of the SSSC controller.

$$u_{lr_in} = -v_S \cdot \frac{i_i}{i} - i_i \cdot X \quad (17.33)$$

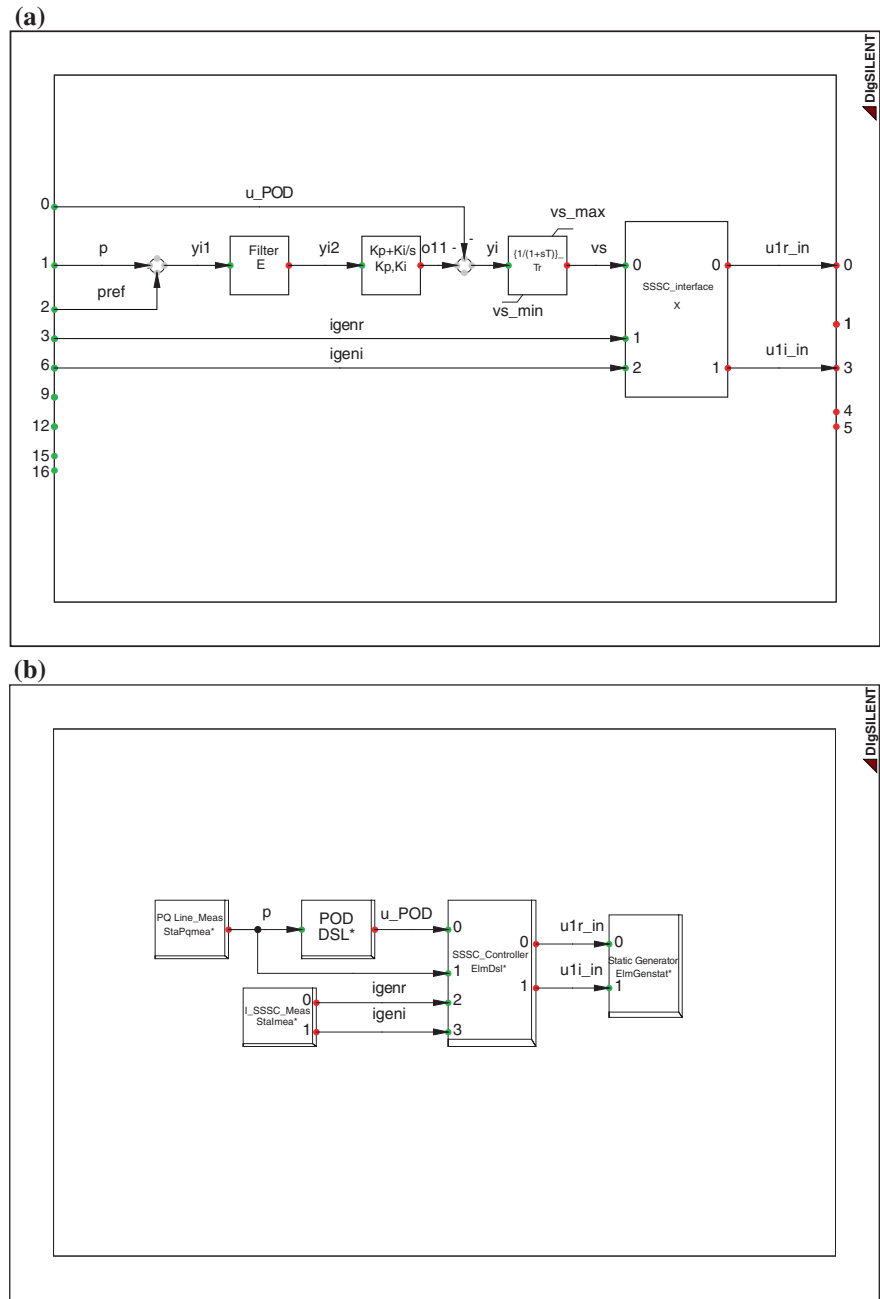
$$u_{li_in} = v_S \cdot \frac{i_r}{i} + i_r \cdot X \quad (17.34)$$

These relations have to be adequately incorporated in an extra block of the control diagram. Thus, the complete PowerFactory Block Diagram *SSSC_controller.BlkDef* has four input signals (p , u_{POD} , $igenr$ and $igeni$) and two output signals (u_{lr_in} and u_{li_in}). The signals $igenr$ and $igeni$ constitute the real and imaginary currents measured at the static generator. Figure 17.6a shows the DSL implementation of the SSSC controller.

Initialization equations representing the SSSC controller are as follows:

```
inc(vs)=(u1i_in-igenr*X)*sqrt(sqr(igenr)+sqr(igeni))/igenr
inc(x1)=vs
inc(x2)=-x1
inc(pref)=p
inc(u_POD)=0

vardef(E)='pu';'Series filter error'
vardef(Kp)='pu';'PI controller proportional gain'
vardef(Ki)='pu';'PI controller integral gain'
vardef(Tr)='s';'First order delay time constant'
vardef(X)='pu';'Short circuit reactance'
vardef(vs_min)='pu';'Minimum limit of the first order delay'
vardef(vs_max)='pu';'Maximum limit of the first order delay'
```



The composite frame of the SSSC, namely *SSSC_Frame.BlkDef*, consists of two slots that represent the measurement devices, whose signals are connected to the POD and SSSC DSL controllers, which finally connects to the static generator *ElmGenstat*. The DSL SSSC Frame is depicted in Fig. 17.6b.

Finally, the corresponding SSSC composite model is structured, and each EPS object, current and power measurement devices (*StaImea*, *StaPqmea*) and DSL common models *ElmDsl* (belonging to SSSC and POD block diagrams) are adequately addressed.

17.3.4 Unified Power Flow Controller (UPFC)

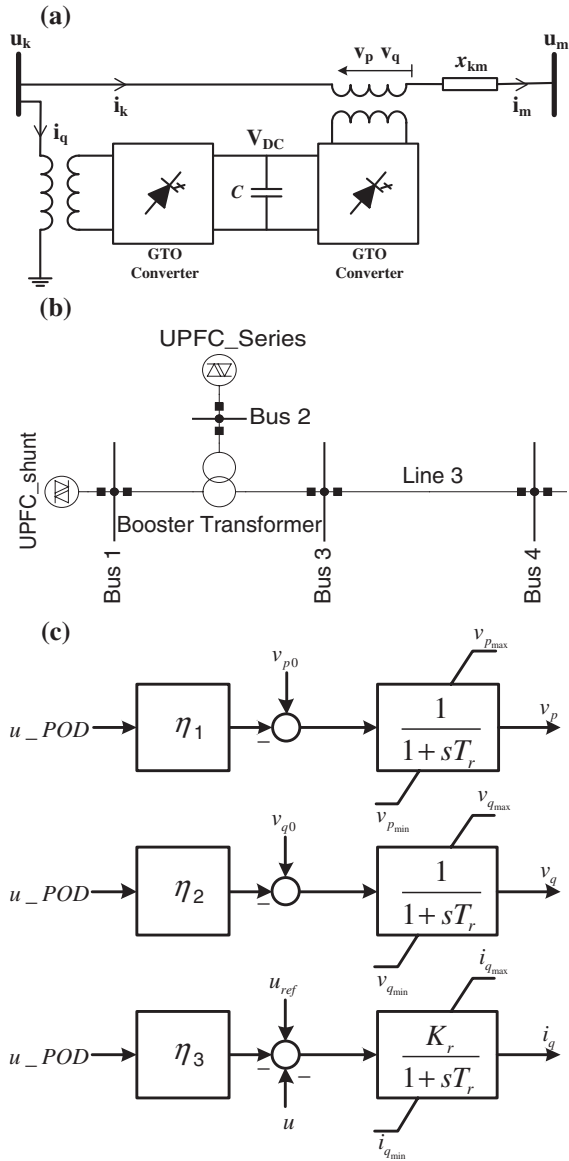
The UPFC results from the combination of a STATCOM and a SSSC, which are coupled on the DC site via a common link. This link allows bidirectional exchange of active and reactive power between these two devices in order to control the active and reactive power compensation to the system. The particular operation of UPFC allows the active power interchange without the need of employing an external power source (i.e. energy storage device). With this configuration, the UPFC is able to control, simultaneously or selectively, the bus voltage, the line impedance, the transmission angle, and the active and reactive power flow. In addition, the UPFC can provide shunt reactive compensation, which is controlled independently of the control of the other parameters [14, 18]. The basic scheme of a UPFC is shown in Fig. 17.7a, which depicts its series and shunt connection to the system, whereas the connection of the PowerFactory objects is shown in Fig. 17.7b.

Based on the previous presented modelling of STATCOM and SSSC, the UPFC can be modelled by a controlled voltage source v_s and by one controlled shunt current source i_{SH} [13]. Thus, two *static generators* are employed in order to model DiGSILENT PowerFactory the simplified UPFC presented by Milano [13].

To implement the UPFC in DiGSILENT PowerFactory, the shunt *static generator ElmGenstat* is connected to the line sending bus and the series *static generator ElmGenstat* is connected via a *booster transformer ElmTrb* (similarly to the SSSC connection).

Three output signals are used in order to control the SSSC, as shown by Milano [13]: i_q , v_p and v_q . Signal i_q represents the component of shunt current (i_{SH}) which is in quadrature with the bus voltage (in this implementation, the component that is in phase with the voltage is not considered, so that only reactive power can be exchanged by the shunt compensator $\Rightarrow i_{SH} = i_q$). Signal v_p represents the component of the series voltage v_s that is in phase with the line current. In steady state, the v_p initial condition v_{p0} is set to zero so that the exchange of active power between the UPFC and the system only takes place when this variable is modulated by the POD controller (i.e. during transients) [13]. Signal v_q represents the component of series voltage v_s that is in quadrature with line current. In steady state, the v_q initial condition v_{q0} might depend on two different control modes, that is constant voltage or constant reactance [13]. The presented implementation uses the constant

Fig. 17.7 Implemented UPFC: **a** connectivity scheme, **b** implementation in PowerFactory, **c** control block diagram



voltage mode, which considers that the magnitude of voltage v_q is constant independently of the line current, that is $v_q = v_{q0}$.

In order to control the above-mentioned signals, three controllers are used. The corresponding UPFC control block diagrams are shown in Fig. 17.7c.

Based on the apparent power flow relationship at the UPFC series component, it is possible to determine the following expression:

$$\begin{aligned}
\bar{S}_{\text{series}} &= \bar{U}_{\text{series}} \cdot \bar{I}_{\text{series}}^* = (u_r + j \cdot u_i) \cdot (i_r - j \cdot i_i) \\
&= (u_r \cdot i_r + u_i \cdot i_i) + j \cdot (u_i \cdot i_r - u_r \cdot i_i) \\
&= P_{\text{series}} + j \cdot Q_{\text{series}} = v_p i_{\text{series}} + j v_q i_{\text{series}}
\end{aligned} \tag{17.35}$$

Thus, the active and reactive powers can be represented as follows:

$$P_{\text{series}} = u_r \cdot i_r + u_i \cdot i_i = v_p i_{\text{series}} \tag{17.36}$$

$$Q_{\text{series}} = u_i \cdot i_r - u_r \cdot i_i = v_q i_{\text{series}} \tag{17.37}$$

The ratio between (17.36) and (17.37) allows relating v_p and v_q with the UPFC series component voltage and current as follows:

$$\frac{u_r \cdot i_r + u_i \cdot i_i}{u_i \cdot i_r - u_r \cdot i_i} = \frac{v_p}{v_q} \tag{17.38}$$

Considering the constant voltage control mode, so that $v_q = v_{q0}$, and the relation presented by (17.37), the following expression can be determined:

$$v_q = v_{q0} = \frac{Q_{\text{series}0}}{i_{\text{series}0}} \tag{17.39}$$

Replacing (17.39) in (17.37), and solving u_i , it is determined:

$$u_i = \frac{Q_{\text{series}0}}{i_{\text{series}0} \cdot i_r} i_{\text{series}} + u_r \cdot \frac{i_i}{i_r} \tag{17.40}$$

Replacing (17.40) in (17.38), and solving u_r , it is determined:

$$u_r = \frac{Q_{\text{series}0} (i_r v_p - i_i v_q)}{i_{\text{series}0} \cdot i_{\text{series}} \cdot v_q} \tag{17.41}$$

Replacing (17.41) in (17.40), and after several simplifications, it is possible to determine:

$$u_i = \frac{Q_{\text{series}0} (i_r v_q + i_i v_p)}{i_{\text{series}0} \cdot i_{\text{series}} \cdot v_q} \tag{17.42}$$

By replacing (17.41) and (17.42) in (17.25), it is possible to determine the inputs of the series static generator functioning as a controlled voltage source $\{u_{lr_in}, u_{li_in}\}$ as functions of v_p and v_q . These relations allow structuring the interface between the series *static generator* control signals and the output of the UPFC series component controller.

$$u1r_in = \frac{Q_{series0} (i_r v_p - i_i v_q)}{i_{series0} \cdot i_{series} \cdot v_q} - i_i \cdot X \quad (17.43)$$

$$u1i_in = \frac{Q_{series0} (i_r v_q + i_i v_p)}{i_{series0} \cdot i_{series} \cdot v_q} + i_r \cdot X \quad (17.44)$$

These relations have to be adequately incorporated in an extra block of the control diagrams related to the UPFC series component. The complete PowerFactory Block Diagram *UPFC_series_controller.BlkDef*, including the mentioned additional block, can be found in the *FACTS Project*. Figure 17.8a shows the DSL UPFC series controller.

Initialization equations representing the UPFC series component controller are as follows:

```
inc(Q0)=u1i_in*igenr-u1r_in*igeni-(sqr(igenr)+sqr(igeni))*X
inc(i0)=sqr(sqr(igenr)+sqr(igeni))
inc(u_POD)=0
inc(vp)=0
inc(vq)=Q0/i0
inc(x1)=vp
inc(x2)=vq
inc(vp0)=vp
inc(vq0)=vq

vardef(u1)='{0,1}'; 'Connection status of POD for vp'
vardef(u2)='{0,1}'; 'Connection status of POD for vq'
vardef(Trp)='s'; 'Vp delay time constant'
vardef(Trq)='s'; 'Vq delay time constant'
vardef(X)='pu'; 'Short circuit reactance'
vardef(vp_min)='pu'; 'Vp minimum limit'
vardef(vp_max)='pu'; 'Vp maximum limit'
vardef(vq_min)='pu'; 'Vq minimum limit'
vardef(vq_max)='pu'; 'Vq maximum limit'
```

On the other hand, the UPFC shunt component operates just like the STATCOM controller. Thus, the following equations are included in an extra block of the corresponding shunt controller:

$$i_{q_ref} = -i_q \quad (17.45)$$

$$i_{d_ref} = 0 \quad (17.46)$$

The complete PowerFactory Block Diagram *UPFC_shunt_controller.BlkDef* is also included in the *FACTS Project*. The DSL implementation of this shunt controller is depicted in Fig. 17.8b.

Initialization equations representing the UPFC shunt component controller are as follows:

The composite frame that allows the UPFC control is *UPFC_Frame.BlkDef*.

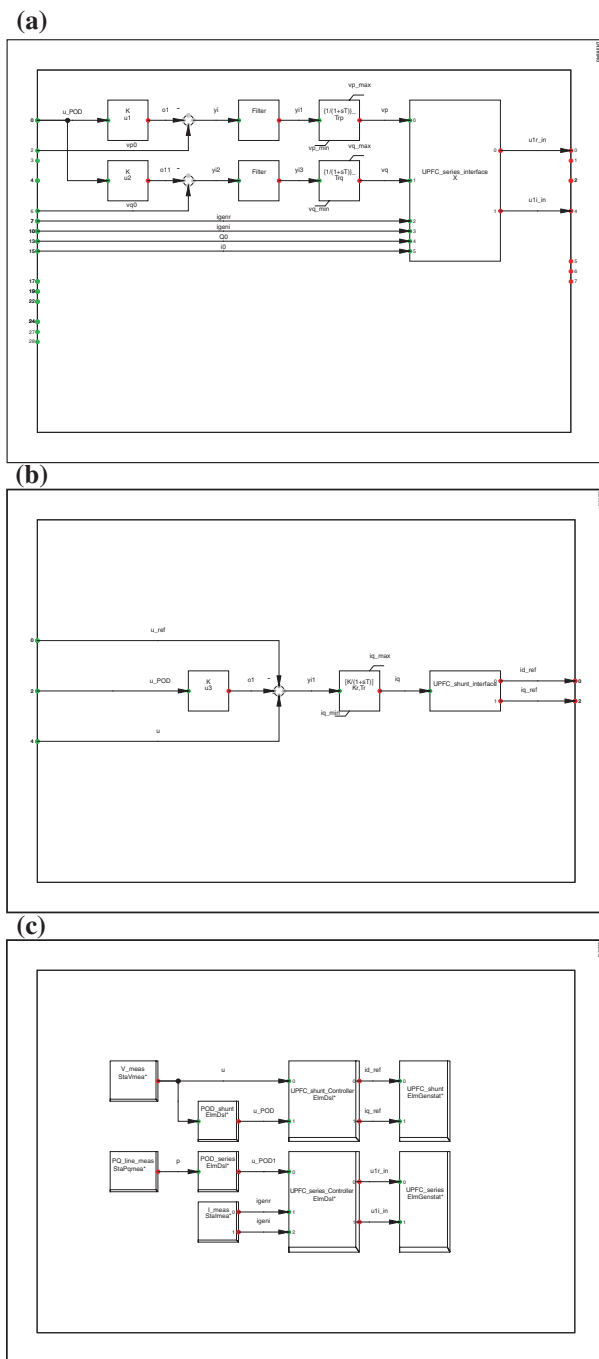


Fig. 17.8 DSL implementation of UPFC: **a** UPFC series controller, **b** UPFC shunt controller, **c** UPFC composite frame

```
inc(x)=-iq_ref
inc(u_ref)=-iq_ref/Kr+u
inc(u_POD)=0

vardef(u3)='{0,1}';'Connection status of POD for ip'
vardef(Kr)='pu';'Iq controller gain'
vardef(Tr)='s';'Iq delay time constant'
vardef(iq_min)='pu';'Iq minimum limit'
vardef(iq_max)='pu';'Iq maximum limit'
```

Finally, the corresponding UPFC composite model is structured, and each EPS object, voltage, current and power measurement devices (*StaVmea*, *StaImea*, *StaPqmea*) and DSL common models *ElmDsl* (belonging to UPFC series and shunt component, and POD block diagrams) are adequately addressed.

17.4 Application Examples

In this section, some results of time domain simulations concerning to the FACTS implementation into a test power system that show their effect on power system dynamic performance are presented.

17.4.1 Test System

For illustrative purposes, the FACTS devices are included in the WSCC 3-machine, 9-bus, 60-Hz test system [1], which is already implemented in PowerFactory as a demo project. The test system, as well as the incorporated FACTS devices, can be found in *FACTS Project*. It consists of 3 synchronous generators equipped with *automatic voltage regulators* (AVRs) and *governors* (GOVs).

17.4.2 Simulation Results

Case I: TCSC implementation

First, a TCSC has been implemented in series with Line 7–8 at bus 7, as can be seen in the *FACTS Project, Single Line Grid*. Simulations consist of time domain analysis. The considered event is a load event consisting of step increase of 3 % of load C at bus 8, applied at 1.0 s.

Figure 17.9 shows the TCSC responses after the load event. It can be shown how the TCSC allows controlling the power flow through the line in order to keep the same pre-disturbance power flow. Moreover, the action of the POD is also highlighted provoking more damped oscillations and with lower amplitudes.

Case II: SSSC implementation

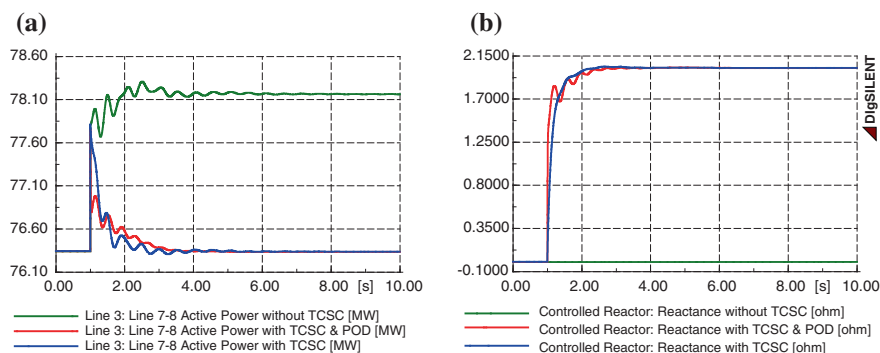


Fig. 17.9 TCSC simulation results -3 % of load C increasing: **a** active power in Line 7-8, **b** reactance of controlled reactor

The same event described in *Case I* is also simulated, but considering a SSSC to be installed instead of the TCSC. Figure 17.10a depicts the dynamic behaviour of the active power flow in Line 7-8, highlighting the quick response of the SSSC to keep the power flow in the initial values. In Fig. 17.10b, the signal control v_s is additionally shown. In this case, the influence of POD also allows a damping increasing in the power flow.

In order to compare the TCSC and SSSC behaviour, the dynamic response of both devices is presented in Fig. 17.10c. It can be noticed that SSSC shows a faster response than TCSC to compensate the power transfer in the transmission line.

This quick response is due to the fact that SSSC is based on rapid response converters and DC energy storage, which give a quicker response than the commutation of reactors or capacitors (which are the base of a TCSC).

Case III: STATCOM implementation

Subsequently, a STATCOM is considered to be installed at bus 5. In this condition, three-phase short circuit is applied at 10 % of Line 4-5 from bus 7 at 0.1 s and considering a fault impedance $Z_f = 0$, followed by the opening of the line at 0.2 s. In this event, the STATCOM injects reactive power in order to compensate the voltage drop at bus 5.

Figure 17.11 presents the STATCOM compensation after the occurrence of the short circuit. Note how the STATCOM controls the voltage of the connection bus in order to recover the initial voltage level. In addition, the POD allows controlling the oscillations that actually occur after the contingency. The variation of the injected reactive power is also presented in the figure.

Case IV: UPFC implementation

Afterwards, the installation of an UPFC is considered in Line 7-8 at bus 7. This device has the objective of compensating both active and reactive

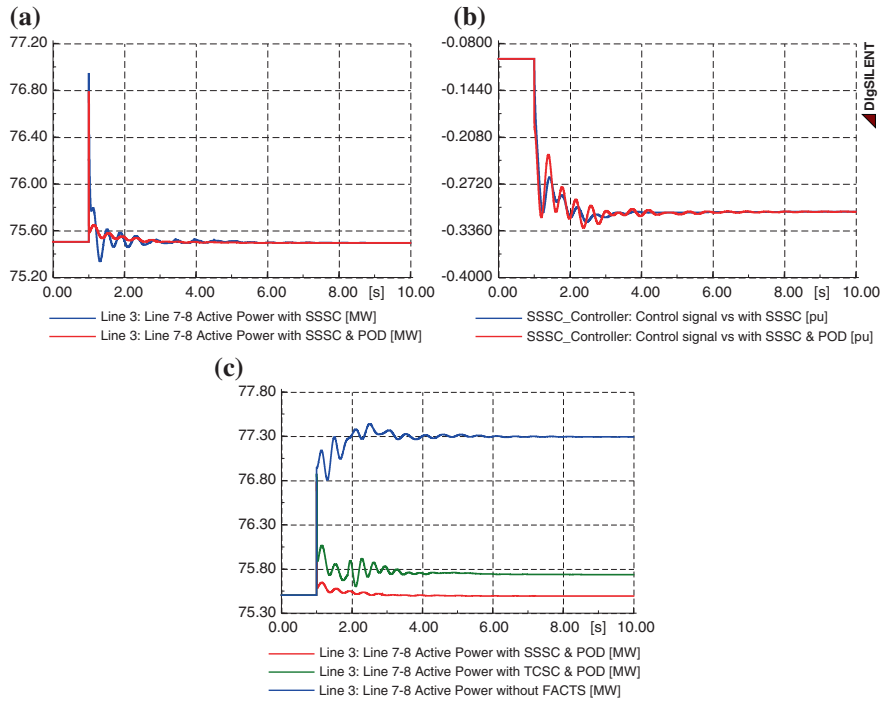


Fig. 17.10 SSSC simulation results -3% of load C increasing: **a** active power in Line 7–8, **b** SSSC control signal v_s , **c** TCSC and SSSC dynamic response

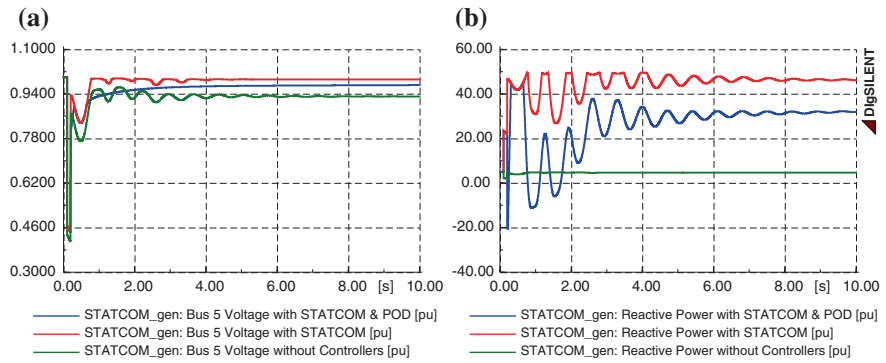


Fig. 17.11 STATCOM simulation results—short circuit at Line 4–5: **a** bus 5 V, **b** STATCOM injected reactive power

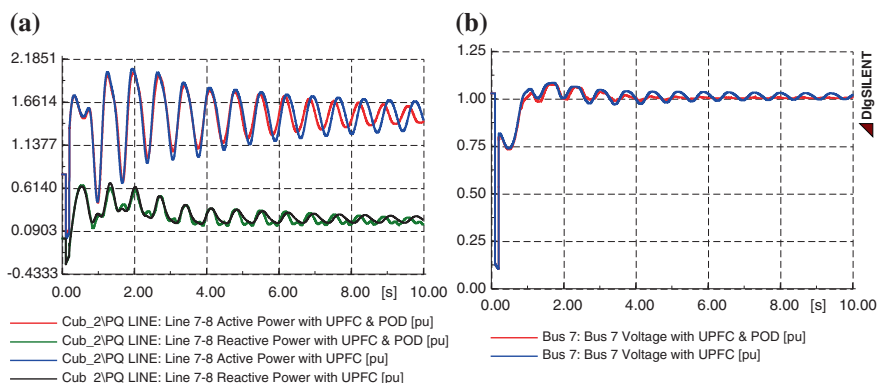


Fig. 17.12 UPFC simulation results—short circuit at Line 4–5: **a** active and reactive power in Line 7–8, **b** bus 7 V

power in the line (series compensation) and bus (shunt compensation) of its connection. In order to test the implementation in PowerFactory, the short-circuit event previously specified is also simulated (i.e. three-phase short circuit in Line 4–5, followed by the line opening). Figure 17.12a and b shows the electric signals compensated by the UPFC, where the line active and reactive powers, as well as the bus voltage, are depicted. The influence of the UPFC allows a less oscillatory response of the signals, mainly when PODs are considered in the control. Note that the actuation of the implemented controllers considerably changes if it is not considered a POD. This response is more noticed in the series compensation, where the injection of active power is directly depended on the POD action.

17.5 Concluding Remarks

Methodology to incorporate new control system models in DIgSILENT PowerFactory offers great versatility and simplicity. This type of programming offers the option of controlling different available library objects in order to model other components of the system that are not actually included in PowerFactory, such as FACTS devices, wind generation or photovoltaic generators (as in the case of the static generator).

In this chapter, guidelines to implement several simplified models of FACTS devices have been described in detail. These devices correspond to thyristor-based controllers or power converter-based controllers, such as TCSC, SSSC, STATCOM and UPFC. For illustrative purposes, the implemented controllers correspond to those simple models presented by Milano [13]. Nevertheless, the presented guidelines can be used in order to incorporate any type of controllers, even those of commercial devices.

Some time domain simulations have also been presented in order to show the dynamic behaviour of the implemented controllers with the objective of improving the system security considering two types of contingencies, that is a load event and a three-phase short circuit. The results obtained confirm that the models have been correctly implemented within the PowerFactory program and clearly demonstrate how the FACTS devices contribute to improve the security levels of the system.

References

1. Anderson PM, Fouad AA (2003) Power system control and stability, 2nd edn. IEEE press
2. Cañizares CA (2000) Power flow and transient stability models of FACTS controllers for voltage and angle stability studies. IEEE Power Eng Soc Winter Meet 2:1447–1454
3. Cepeda JC (2013) Real time vulnerability assessment of electric power systems using synchronized phasor measurement technology, Doctoral thesis, Instituto de Energía Eléctrica, Universidad Nacional de San Juan, San Juan, Argentina, ISBN: 978-987-33-4328-5
4. Chokhawala R, Danielsson B, Angquist L (2001) Power semiconductors in transmission and distribution applications, in Power Semiconductor Devices and ICs, 2001. ISPSD'01. In: Proceedings of the IEEE 13th international symposium, pp 3–10
5. DIgSILENT GmbH (2007a) Series capacitance TechRef ElmScap V2, Gomaringen, Germany
6. DIgSILENT GmbH (2007b) Series reactor TechRef ElmSind V1, Gomaringen, Germany
7. DIgSILENT GmbH (2010) Static generator TechRef ElmSind V1, Gomaringen, Germany
8. DIgSILENT GmbH (2011) DIgSILENT PowerFactory Version 14.1 User's Manual, Gomaringen, Germany
9. Djilani KYI, Samir H, Ahmed ZS, Mohamed D (2013) Modelling a unified power flow controller for the study of power system steady state and dynamic characteristics. In: 5th international conference on modeling, simulation and applied optimization (ICMSAO), pp 1–6, 28–30 Apr
10. Gupta A, Sharma PR (2013) Static and transient stability enhancement of power system by optimally placing UPFC (Unified Power Flow Controller). In: Third international conference on advanced computing and communication technologies (ACCT), pp 121–125, 6–7 Apr
11. Jena MK, Tripathy LN, Samantray SR (2013) Intelligent relaying of UPFC based transmission lines using decision tree. In: 1st international conference on emerging trends and applications in computer science (ICETACS), pp 224–229, 13–14 Sept
12. Komoni V, Krasniqi I, Kabashi G, Alidemaj A (2012) Control active and reactive power flow with UPFC connected in transmission line. In: 8th mediterranean conference on power generation, transmission, distribution and energy conversion (MEDPOWER 2012), pp 1, 6, 1–3 Oct
13. Milano F (2008) Power system analysis toolbox—PSAT, documentation for PSAT version 2.0.0
14. Molina M, Sarasua A, Suvire G (2011) Electrónica de potencia – Vinculación de generación dispersa a la red, Modulo II: Aplicaciones de electrónica de potencia, Universidad Nacional de San Juan – Instituto de Energía Eléctrica
15. Padiyar KR (2007) FACTS controllers in power transmission and distribution. New Age International Publishers, New Delhi
16. Rüberg S et al. (2010) Improving network controllability by flexible alternating current transmission system (FACTS) and by high voltage direct current (HVDC) transmission systems. REALISEGRID
17. Jiang Shan, Annakkage UD, Gole AM (2006) A platform for validation of FACTS models. IEEE Trans Power Deliv 21(1):484–491
18. Zhang X, Rehtanz C, Pal B (2006) Flexible AC transmission systems: modelling and control

Chapter 18

Active and Reactive Power Control of Wind Farm Based on Integrated Platform of PowerFactory and MATLAB

Min-hui Qian, Da-wei Zhao, Da-jun Jiang, Ling-zhi Zhu and Jin Ma

Abstract The integration of large-capacity wind energies into a power system creates big challenges to the power system operation and control due to the intrinsic intermittence of wind energies, which constrains further developments of wind power generation. Countries around the world including Europe, USA and China have all made Grid Codes with clear requirements on active and reactive power controls in a wind farm. By effective active and reactive power controls, wind power curtailments can be substantially reduced. Furthermore, there have been no mature simulation and analysis platforms in China on testing correctness and rationality of wind farm control strategies so far. This chapter reports our research as well as engineering practices on building an integrated active and reactive power control system for wind farms using PowerFactory and MATLAB. The highlights that make this work unique are as follows: (i) the integration of PowerFactory and MATLAB makes the active and reactive power control modelling efficient and flexible; (ii) both the real measurements of wind speeds and the active power forecasted from history data are inputted in text file format to each wind generator in PowerFactory; (iii) the reactive power control capabilities of wind generators are fully utilized to reduce the scale of the reactive power compensations installed in a wind farm or from the grid.

Keywords Interfacing PowerFactory with MATLAB • Active and reactive power control • Wind power

Electronic supplementary material The online version of this chapter (doi:10.1007/978-3-319-12958-7_18) contains supplementary material, which is available to authorized users.

M. Qian · D. Zhao · D. Jiang · L. Zhu
Renewable Energy Department, China Electric Power Research Institute, Nanjing, China

J. Ma (✉)
School of Electrical and Information Engineering, University of Sydney, Sydney, Australia
e-mail: j.ma@sydney.edu.au

18.1 Introduction

The integration of wind power in large scales will have significant impacts on power system operation and control [1–4]. The random nature of wind leads to high fluctuations of wind power generated, which in turn challenges the active power and reactive power control in a power grid containing large amounts of wind power [5–10]. On the other hand, the wind curtailments in wind farms that waste wind energy resources and reduce economic efficiency have become a major concern along the developments of the wind power. The wind farms in north, northeast and northwest regions of China are often operated in restricted modes, leading to serious wind power curtailments up to 20 % in 2012. Nevertheless, a well-managed active power and reactive power control system will minimize wind power curtailments and enhance the investment return rate.

Countries around the world including Europe, USA and China have all developed Grid Codes with clear requirements on active and reactive power controls in a wind farm. To test the active power and reactive power controllers and validate their consistency with the stipulated codes, there is an urgent need to develop an integrated simulation and analysis platform. To the authors' best knowledge, nonetheless, there has not been one so far in China.

In this chapter, we construct a wind farm power control testing platform by integrating DIGSILENT PowerFactory and MATLAB. Case studies on a 200 MW real wind farm validate the practicability and effectiveness of this platform. This chapter will be organized as follows: Sect. 18.1 presents a brief introduction on the active power and reactive power control requirements on a wind farm. Section 18.2 illustrates fully how PowerFactory is used to build a wind farm model from an individual wind generator level to a wind farm level. Section 18.3 demonstrates the models developed in MATLAB for the wind farm controller design. The controller structure and its realization by Simulink and *S* functions are discussed in detail. The architecture of frame and block in PowerFactory, used for communicating with MATLAB, will also be elaborated fully in this section. Case studies in Sect. 18.4 on a real power system containing a large wind farm verify the correctness and practicability of the developed platform. Section 18.5 concludes the whole chapter.

18.1.1 Requirements on Active Power and Reactive Power Control of Wind Farms in China

In 2011, China issued its national standard on wind power integration, namely “Technical rules for connecting wind farms to a power system”. The standard clearly prescribed that a wind farm should be equipped with an active and reactive power control system, thus having the active and reactive power regulation ability.

1. Control modes on active power

According to the standard, a wind farm should have the ability to participate in power system frequency regulation, peak load supply and reserve management. When the total wind power outputs of a wind farm are more than 20 % of its rated value, all running wind turbines will take part in power system active power controls. Wind farms should be able to receive and automatically execute control commands from a system operator on active power changes under specified ramp rates and meet all requirements. In normal operation conditions, typical recommended limit values of active power control ramp rates are shown in Table 18.1. Meanwhile, there are different active power control requirements from the system operator, i.e. limit mode, balance mode and delta mode.

- **Limit mode:** The active power output of a wind farm should be controlled within a preset limit value or a command value given by a system operator. These limit values may differ from time to time according to system operation situations.
- **Balance mode:** The active power output of a wind farm will be controlled to reach a specified value under a given slope (if the designated value is higher than the available power, it will be replaced by the available value). Upon the exit of this operation mode, the active power control system returns to the maximum power point tracking (MPPT) mode following a given slope required by the national standard.
- **Delta mode:** The active power output of a wind farm will be controlled at a value ΔP less than the maximum power that it can generate. ΔP is a pre-determined value or a time-changing value from the system operator.

2. Control modes on reactive power

According to the standard, wind farms should be equipped with reactive power control systems to participate in system reactive power regulation and voltage control. A wind farm automatically regulates its reactive power to control the voltage at the point of common coupling (PCC). The regulation rate and control precision should also meet the requirements on power system voltage regulation. When the grid voltage is in a normal range, a wind farm should be able to control the voltage of PCC within the range of 97–107 % of the rated voltage. Meanwhile, the requirement stipulates different reactive power control types, i.e. voltage control mode, reactive power control mode and power factor control mode, which will be explained briefly as follows:

Table 18.1 Active power control parameters

Installed capacity (MW)	10-min ramp rate(MW)	1-min ramp rate (MW)
<30	10	3
30–150	Installed capacity/3	Installed capacity/10
>150	50	15

- **Voltage control mode:** The reactive power in a wind farm is managed so that the voltage target at the connection point given by the system operator can be achieved.
- **Reactive power control mode:** The reactive power control system is operated according to the reactive power control target given by the system operator.
- **Power factor control mode:** The reactive power control system is operated according to the power factor control target given by the system operator.

18.1.2 Active and Reactive Power Control Principles

1. Active power control system

Based on the aerodynamics theory, the mechanical power produced by wind turbine can be calculated as:

$$P_t = \frac{1}{2} C_p(\theta, \lambda) \rho A V_w^3 \quad (18.1)$$

The physical interpretations of all variables can be found in [9].

With the fast developments of wind power technologies, variable-speed wind generators, including Doubly Fed Induction Generator (DFIG) and Permanent Magnet Synchronous Generator (PMSG), have become dominant wind generator types [11]. Equation (18.1) shows that the wind turbine power can be controlled by w_t or θ , which consequently regulates electrical power outputs of a wind generator. Due to the randomness and volatility of wind speeds, relying only on the pitch angle to regulate wind turbine's power will drive a pitch control system into too frequent operations, leading to wind turbine fatigue and causing damages. Meanwhile, a pitch control system has a large inertia that will limit the power regulation rate of a wind generator. So many methods have been proposed in practising active power control systems in order to make full use of fast regulation characteristics of the converter and reduce the regulation frequency of mechanical parts. Herein, an efficient active power regulation mode is adopted, as described below:

- When the rotor speed of a wind generator is below its upper limit, the active power of a wind generator will be regulated by rotor speed control;
- When the rotor speed of a wind generator exceeds its upper limit, the active power of the wind generator will be regulated by the pitch angle.

2. Reactive power control system

DFIG is equipped with a back-to-back (AC–DC–AC) converter. The excitation of DFIG is provided by the rotor side converter. In general, the active power and reactive power are controlled separately. The DFIG's reactive power capacity is provided by the stator and the grid-side converter.

In China, the national standard, “Technical rules for connecting wind farms to a power system”, requests that a wind farm should make full use of wind generators’ reactive power capacities and regulation capabilities. When the total wind generators’ reactive power capacities are insufficient for system voltage regulation, reactive power compensation devices should be added. According to the national standard, reactive power controls in a wind farm should meet the following criteria: (i) respond fast in order to adapt to rapid wind power fluctuations; (ii) ensure both safety and economy during the operation; (iii) have capabilities to distinguish and adapt to various operation conditions; and (iv) have strong robustness.

18.2 Building a Wind Farm Model Based on PowerFactory

18.2.1 Wind Farm Model

It has always been a big issue to make a balance between the model accuracy and the model complexity [11–14]. A dynamic model other than a static one of a wind farm is essential in stability simulations if its dynamics has significant impacts on the transients being studied and also has matchable transient timescales with other dynamic components. With regard to modelling the active power and reactive power controls in a wind farm, the timescale of a simulation is from seconds to minutes. So the fast electromagnetic and electromechanic transients can be neglected and some simplifications will be made in modelling wind generators for enhancing the simulation efficiency without losing its accuracy at the same time. Modelling a wind farm requires building models of a large number of wind generators and turbines, unit transformers of wind generators, transmission lines, reactive power compensation devices and power substation transformers et al.

Modelling a wind farm includes modelling wind turbines/generators, various transformers and feeders. An abundant model library is provided in PowerFactory [15], which includes power transformers (2-winding transformer “ElmTr2”, 3-winding transformer “ElmTr3”) and lines (transmission lines “ElmLne”). Models of major components in a wind farm are shown in Table 18.2. Lines and transformers can be easily modelled based on the templates in the model library of PowerFactory. However, control strategies vary from different wind turbine and equipment manufacturers. So DiGSILENT Simulation Language (DSL) has to be employed to build different controller models, which will be shown in the following sections.

Table 18.2 Main elements of wind farm

Description	PowerFactory elements
Wind turbines	ElmGenStat/Composite model/Common model
Transformer	ElmTr2/ElmTr3
Line	ElmLne

18.2.2 Modelling Wind Generator and Its Control Strategies [13]

A variety of methods could be taken for modelling a wind generator and control strategies in PowerFactory, which can be classified by using detailed models and simplified models. A detailed wind turbine model of a DFIG (2.5 MW) is provided in PowerFactory model library while the simplified one is based on a controlled current source interface. As stated in the previous section, for the purpose of the active power and reactive power control, the simulation focuses on the slow dynamics compared with the electromagnetic and electromechanic transients. So wind generators are modelled as controlled current sources. Based on our experiences on many real engineering projects in China, this simplified wind generator model is good enough to simulate active and reactive power controls in a wind farm. However, the computation burden is significantly relived by using controlled current source model considering the bulk scale of a wind farm with hundreds of wind generators. Herein, only the implementation of the simplified wind generator model with controlled current source interface will be described in detail.

The wind power generator and its converter are represented by a controlled current source [13]. More complex electromagnetic transients and electromechanic transients of the generator and the converter are neglected. The nature of this model is a voltage-source inverter regulated by current references. So the external characteristics of a wind power generator and its inverter as a whole are modelled as a controlled current source behind the reactance. According to the d–q decoupled control strategies, the active and reactive power injecting into a grid from a wind power generator can be controlled separately through the respective active and reactive currents.

A DFIG model structure based on a controlled current source has been developed according to [13] and is shown in Fig. 18.1. The DFIG's control system can be seen clearly from this figure, which consists of an excitation system and a pitch control system. Various control blocks fall into three categories, namely generator/converter controllers, electrical controllers and wind turbine controllers. It should be noted that both the detailed and the simplified wind generator models apply the same models for wind turbines and related controllers. A composite model named "control frame" is created by using the "new object" icon in the toolbar of the data manager and then selecting "composite model" from the available options, as shown in Fig. 18.1. This composite model contains the following slots: generator control and converter control system, wind power system, pitch control system and rotor shaft system. Among them, the first component is an electrical control system while the rest three components are mechanical systems with slower responses compared to an electric system.

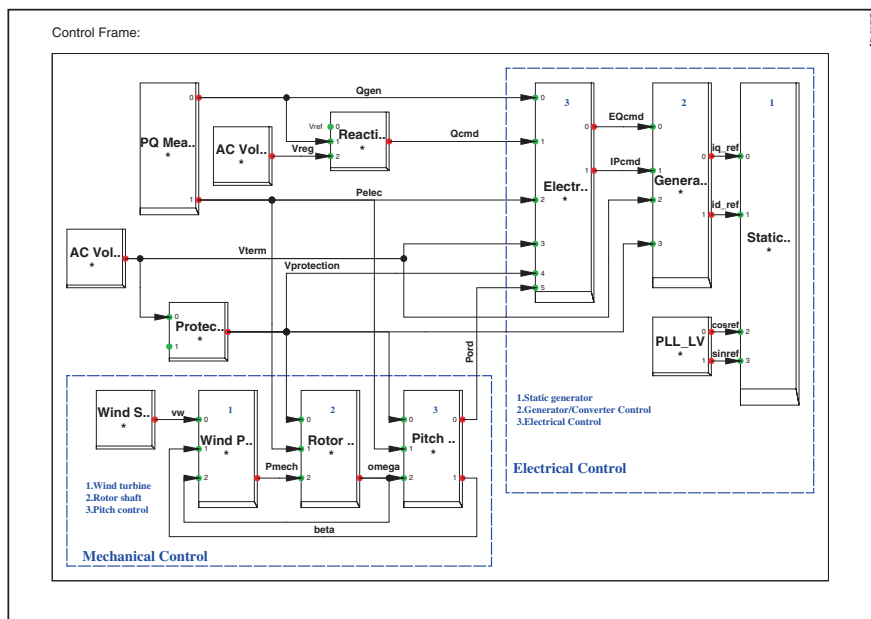
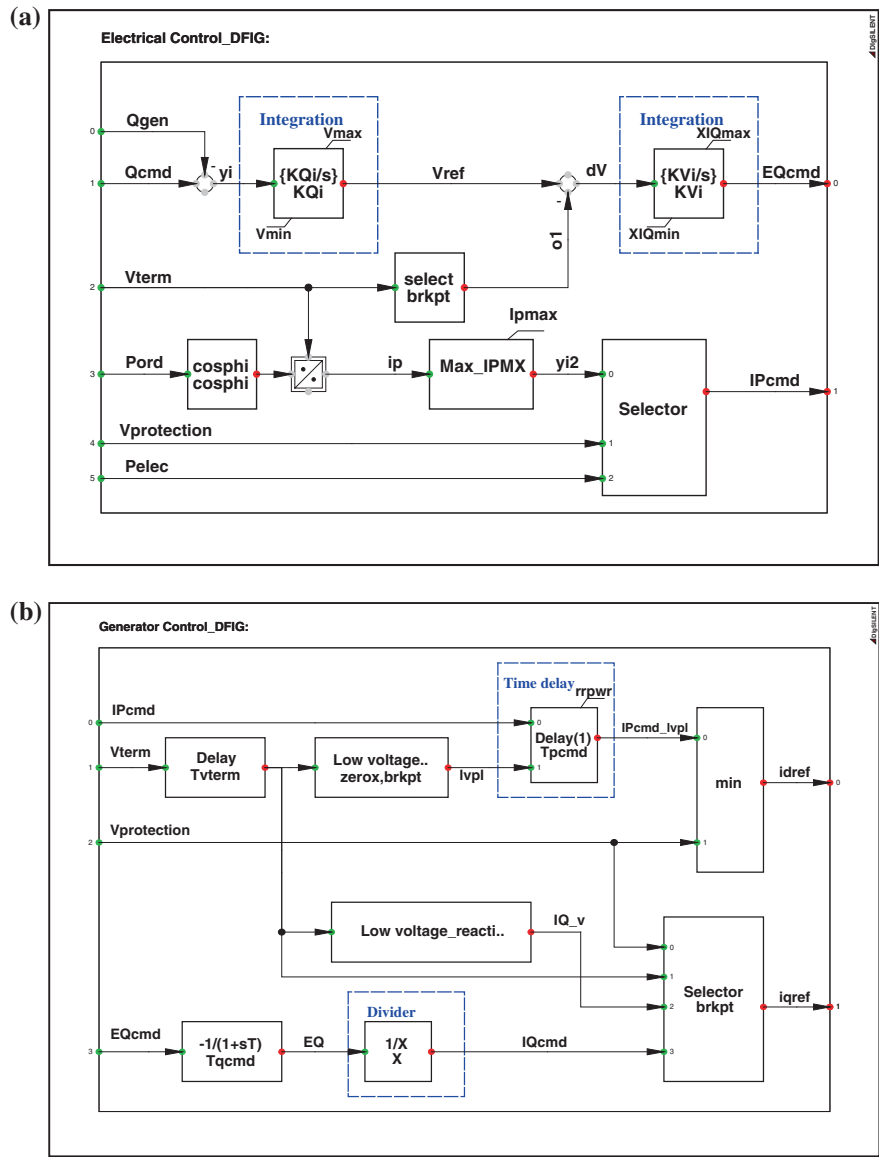


Fig. 18.1 DFIG Model structure based on controlled current source interface

18.2.2.1 Modelling Electrical System

1. Modelling electrical controllers

The objective of this electrical control model is to obtain the reference signals for active power and reactive power, namely the active current reference IP_{cmd} and the excitation voltage EQ_{cmd} . The control error of the reactive power is fed into an integration block with a gain K_{Qi} to generate a voltage reference. The measured terminal voltage is compared with this voltage reference, and the resulting voltage error is again inputted to an integration block with a gain K_{Vi} to compute the excitation voltage command E''_{qcmd} . The gain coefficients K_{Qi} and K_{Vi} , tuned to improve the performance of the control system, are usually finalized in field tests and further calibrated in simulation models accordingly. Figure 18.2a shows the electrical controller model established in PowerFactory, which is built by the following steps: first, click the “new object” icon in the toolbar of the data manager; second, select “common model” from the available options. Then, respective components in this model such as a block reference, a summation point, a multiplier and a divisor could be built by dragging the corresponding icons from the right pane of the window in edit mode. In addition to that, two critical blocks have to be added in this model, i.e. a PI controller with non-wind-up limits and a signal select element. The latter can be built using



the DSL special function “select”, and the former can be implemented using the function “lim” as follows:

```
x.=Ki*yi           ! state variable equation
yo=lim(x,y_min,y_max) !yo limit realized by the special function “lim”
```

The initialization of state variables is necessary and important for using a DSL model. In a user-defined model, two principles can be utilized for the initialization: (1) derivatives of state variables should be equal to zero at steady states; (2) an output variable will be considered as a known quantity. Thus, in this model, based on Fig. 18.2, initializations of the state variable and the input variable in the PI controller can be done as (other variables and signals are defined in similar manner):

```
inc(xVi)=EQcmd ! state variable initialization
inc(Qcmd)=Qgen ! input signal initialization
```

2. Modelling generator/converter

A static generator model is provided in PowerFactory Library [15]. Since the input signals of a static generator are defined as: I_{dref} , I_{qref} , $\cos\theta$ and $\sin\theta$, the excitation voltage command E_{Qcmd} has to be transformed to the required input signal. The following method is taken: the signal E_{Qcmd} is divided by the equivalent reactance X (the recommended value is 0.8 p.u. [13]). The quotient is the reactive current reference and will be used as the input signal to the static generator. A detailed representation of a generator/converter model in PowerFactory is shown in Fig. 18.2b, which includes mainly an inertial block and a divider. The inertial block is defined as follows:

```
x.=(yi-x)/T !state variable equation
yo=x        !output equation
```

Similarly, the state variable must be initialized before simulations as follows:

```
inc(x)=yi !state variable initialization
```

18.2.2.2 Modelling Mechanic System

1. Wind turbine model

A wind turbine model, shown in Fig. 18.3a, is developed in PowerFactory based on its mechanisms explained in Sect. 18.1.2. From Eq. (18.1), the input variables to a wind turbine are as follows: wind speed (V_w), pitch angle (β) and turbine velocity (ω) while the output variable is mechanic power (P_{mech}).

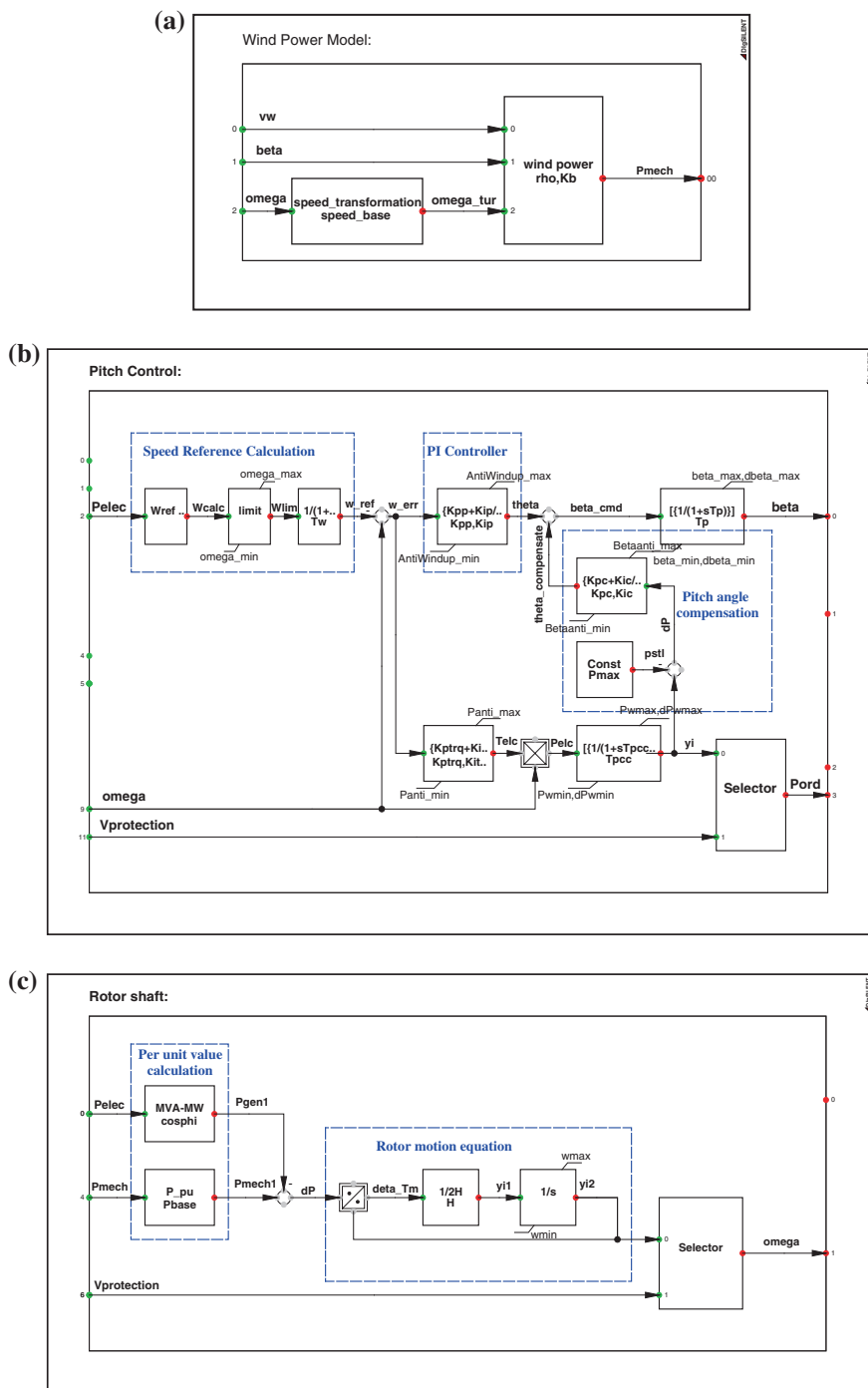


Fig. 18.3 Mechanic system model. **a** Wind turbine model. **b** Pitch control model. **c** Rotor shaft system

2. Pitch control model

According to input signals, there are two models of a pitch control system. One is the limiting speed model while the other is limiting power model. When one control model is chosen, the other control model is bypassed. Figure 18.3b shows details of the limiting speed model with an auxiliary limiting power model developed in our project. A PI controller is the main block in this model, and its implementation and initialization are as follows:

```

x.=Ki*yi                ! state variable equation
yo=lim(Kp*yi+x,y_min,y_max) ! yo limit
inc(yi) = yo            ! input signal initialization

```

3. Rotor shaft model

The rotor shaft model implements the rotor inertia equation. This equation uses the mechanical power from the wind power model and the electrical power from the generator/converter model to calculate the rotor speed. A one-mass rotor model is established in PowerFactory as shown in Fig. 18.3c.

18.3 Modelling Power Controllers for Wind Farm Based on MATLAB

18.3.1 Modelling Active/Reactive Power Control of Wind Farm

According to Chinese National Standard on wind farm active/reactive power control requirements mentioned in Sect. 18.1, the structure of the controller [5] is illustrated in Fig. 18.4.

The wind farm control system contains typically a power reference calculation and a dispatch function block. At the present stage, time delays due to the signal communication between the wind farm control level and the wind turbines are neglected.

18.3.1.1 Active Power Control Model

The active power controller manages total active power outputs of a wind farm to meet the active power requirements from a system operator at PCC of the wind farm. The controller ensures the required power at PCC by setting an appropriate active power reference for each wind turbine, and then, the wind turbine controller adjusts its power generation to the requested reference value. When losses in a wind farm grid are considered, the total power produced by wind turbines is higher than the power measured at PCC [5].

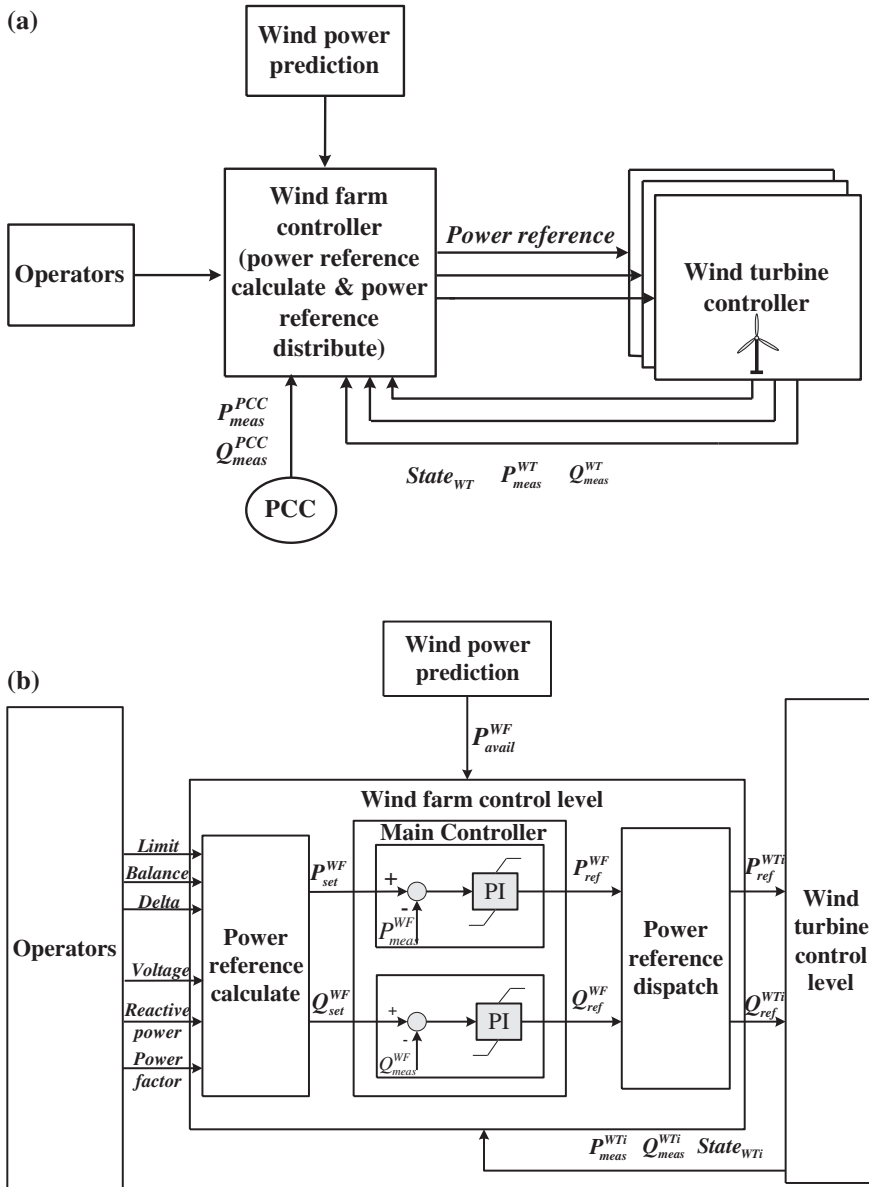


Fig. 18.4 Control structure of wind farm and wind turbine controllers. **a** Scheme of wind farm and wind turbine controllers. **b** Wind farm and wind turbine controllers

The wind farm controller will be able to control the active power delivered at PCC and will compensate dropouts of individual wind turbines by controlling active power from remaining connected turbines whenever this is possible. If the

communication between a certain wind turbine and the wind farm controller is not working, indicated by the shake-hand signal, this wind turbine will still work autonomously, and the wind farm active power controller will control the remaining wind turbines so that total wind farm power delivered at PCC still meets requirements from the system operation centre.

1. Calculation of active power references

According to the definition of active power control modes stated in Sect. 18.1 and by denoting P_{limit} as the dispatch order from a system operator in the limit mode, P_{balance} as the power order from a system operator in the balance mode, ΔP as the preset power difference from the system operator and $P_{\text{avail}}^{\text{WF}}$ as the predicted wind power that can be generated, the active power reference can be determined as follows:

- Limit mode

$$P_{\text{ref}}^{\text{WF}} = \min(P_{\text{limit}}, P_{\text{avail}}^{\text{WF}}) \quad (18.2)$$

- Balance mode

$$P_{\text{ref}}^{\text{WF}} = \min(P_{\text{balance}}, P_{\text{avail}}^{\text{WF}}) \quad (18.3)$$

- Delta mode

$$P_{\text{ref}}^{\text{WF}} = P_{\text{avail}}^{\text{WF}} - \Delta P. \quad (18.4)$$

2. Distribution of active power reference

For the purpose of real-time dynamic active power adjustments on each wind turbine in a wind farm, a zone regulation method is taken here. Wind turbines in a wind farm are partitioned into several zones, and the total active power is allocated firstly into different zones according to the wind power availability in each zone. Then, wind turbines in the same zone will share the allocated active power of that zone equally. The active power allocation scheme can be represented by the following equation:

$$\Delta P_{\text{ref}}^{\text{WT}i} = \frac{P_{\text{avail}}^i}{\sum_{i=1}^m \sum_{j=1}^{n_i} P_{\text{predict}}^j} * \Delta P_{\text{ref}}^{\text{WF}} \quad (18.5)$$

where i represents the i -th zone of wind generators, m represents the total number of wind generator zones, n_i denotes the total number of dispatchable generators in the i -th zone, P_{predict}^j is the predicted wind power of the j -th generator in the i -th zone, $\Delta P_{\text{ref}}^{\text{WF}}$ is the total active power changes to be made in that wind farm and $\Delta P_{\text{ref}}^{\text{WT}i}$ is the active power allocated to the generators in the i -th zone.

So the active power reference of each wind generator in the wind farm can be calculated as:

$$P_{\text{ref}}^{\text{WT}j} = P_{\text{measure}}^{\text{WT}j} + \Delta P_{\text{ref}}^{\text{WT}j} \quad (18.6)$$

where j represents the j -th wind generator in the wind farm, $P_{\text{ref}}^{\text{WT}j}$ represents the active power reference of the j -th wind generator and $P_{\text{measure}}^{\text{WT}j}$ represents the measured active power of the j -th wind generator, which is its real active power output before being adjusted, $\Delta P_{\text{ref}}^{\text{WT}j}$ represents required active power changes allocated to the j -th wind generator.

18.3.1.2 Reactive Power Control Model

Reactive power outputs of a large wind farm can be controlled into a specified range. The target of this control is often to achieve a unity power factor. However, with higher and higher wind power penetration, more advanced reactive power control is required in the national standard so that reactive power control in a wind farm can be more accurate and more flexible, which can be utilized as an effective means to provide reactive power supports to the grid.

The reactive power control objective in a wind farm can be set to achieve constant reactive power outputs, a specified power factor or an expected voltage output. In the voltage control mode, it is required that the voltage of PCC traces the set value from a system operator automatically. The reactive power outputs of each wind generator are coordinated by a wind farm reactive power controller in order to avoid instability as well as unnecessary reactive power flows among wind generators if each wind generator attempts to control the voltage of its own connection point individually.

1. Calculation of reactive power reference

According to the definition of reactive power control modes stated in Sect. 18.1 and by denoting U_{set} as the voltage reference from the system operator and k as the reactive power-voltage regulation ratio in the voltage control mode, Q_{set} as the reactive power reference from a system operator in the reactive power control mode and PF_{set} as the power factor reference from a system operator in the power factor control mode, the reactive power reference can be determined as follows:

- Voltage control mode

$$Q_{\text{ref}}^{\text{WF}} = Q_{\text{meas}}^{\text{WF}} + k * (U_{\text{set}} - U_{\text{meas}}^{\text{WF}}) \quad (18.7)$$

- Reactive power control mode

$$Q_{\text{ref}}^{\text{WF}} = Q_{\text{set}} \quad (18.8)$$

- Power factor control mode

$$Q_{\text{ref}}^{\text{WF}} = \frac{P_{\text{meas}}^{\text{WF}}}{\text{PF}_{\text{set}}} * \sqrt{1 - \text{PF}_{\text{set}}^2} \quad (18.9)$$

2. Distribution of reactive power reference

The total reactive power reference of a wind farm is allocated to each wind generator based on the sensitivity analysis of node voltages with respect to reactive power outputs of each generator [16], which can be derived from the PQ decoupled power flow analysis as:

$$\Delta V = B^{-1} \Delta Q \quad (18.10)$$

where ΔQ is the node reactive power change vector, B^{-1} is the inverse of the susceptance matrix reflecting sensitivities of node voltages with respect to node reactive power changes and ΔV is a vector of node voltage changes.

18.3.2 Realization of Wind Farm Active/Reactive Power Control

Since MATLAB has its advantage in controller modelling with abundant in-built control algorithms, a combined platform using PowerFactory and MATLAB would be more powerful in implementing wind farm active and reactive power controls. The development of an interface between PowerFactory and MATLAB is crucial in building this hybrid simulation platform.

When a MATLAB model is linked to PowerFactory [17–20], the following modules and models interact with each other during the simulation.

- PowerFactory,
- MATLAB data and model in the form of m-file (.m),
- MATLAB/Simulink model (.mdl).

In order to implement a MATLAB model, i.e. the wind farm active and reactive power controller, into a wind farm project in PowerFactory, a frame similar to a DSL model as shown in Fig. 18.5a has to be built. The first step is to create a “slot” inside a “composite frame” in the PowerFactory where the model should be inserted. Meanwhile, a block must be defined including inputs from PowerFactory, outputs from MATLAB and all state variables. Accordingly, all the defined inputs, outputs and state variables of this block must be declared properly in the m-file. Figure 18.5a illustrates the components of a wind farm control realized in PowerFactory, which include the following:

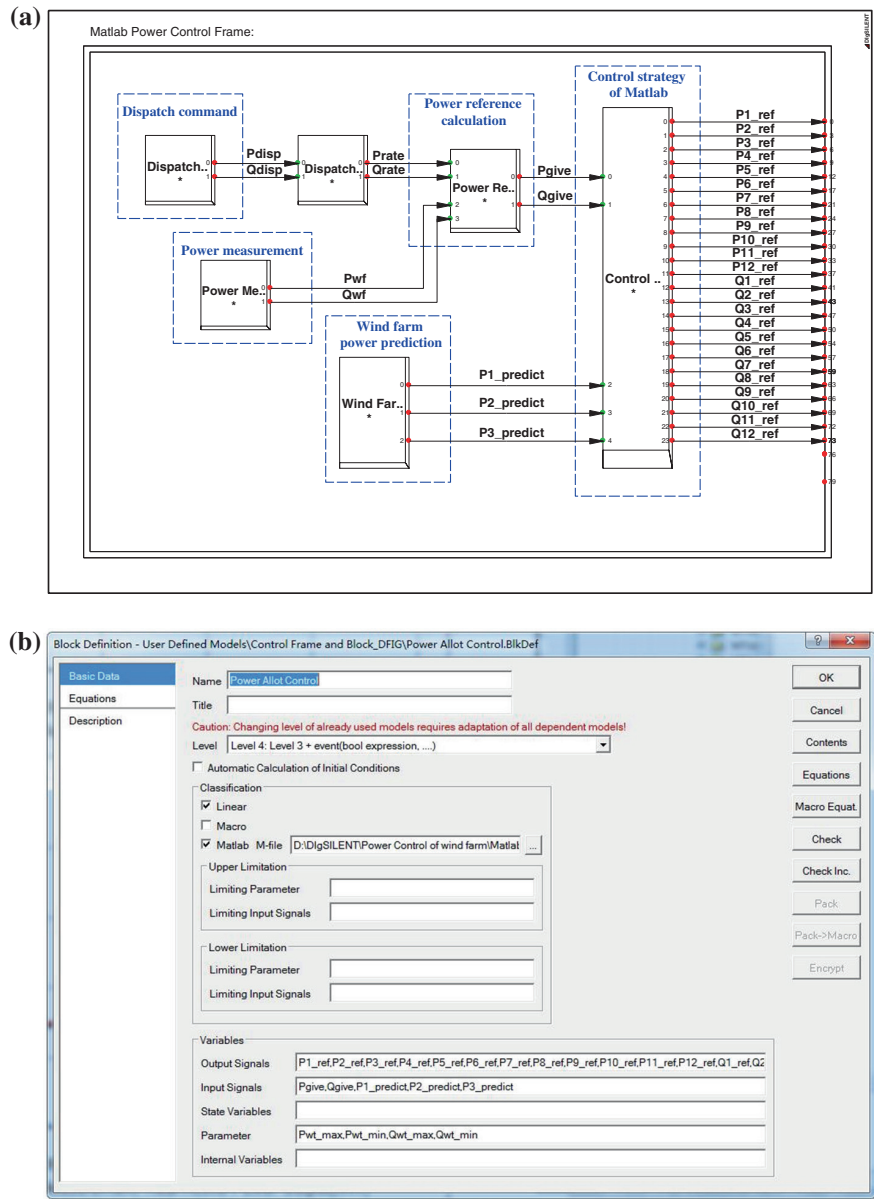


Fig. 18.5 Detailed model using a special frame in PowerFactory. **a** Frame of a wind farm controller. **b** Block definition

- Dispatch command slot: receives dispatch signals from a system operator;
- Power measurement slot: measures actual power outputs of a wind farm;
- Power reference calculating slot: calculates the active power reference of different control modes;
- Wind farm power prediction system slot: provides all power predictions of a wind farm;
- Control strategy with MATLAB slot: calculates the dispatch value for each wind turbine. Since the controllers are designed in MATLAB, a PowerFactory–MATLAB interface is necessary as a link of the two platforms.

A DSL model (object class `BlkDef`) as a connection of the two platforms should be set up in PowerFactory, and a block (named “Power Allot Control”) is defined to realize the control strategies based on the model libraries in PowerFactory. Input and output variables, parameters, state variables and respective limits are all defined in this block (see Fig. 18.5a). It is worth noting that the checkbox “MATLAB m-file” must be ticked and the MATLAB (m-file) file can be designated with which the block in PowerFactory can communicate during a simulation, as shown in Fig. 18.5b. Here, the MATLAB file “PowerAllotControl_interface.m” is defined as the connection file between the PowerFactory platform and the MATLAB platform.

The active and reactive power control strategies of a wind farm are implemented in simulations using MATLAB/Simulink package. The structure of this model is illustrated in Fig. 18.5a. The critical components of this model mainly include two PI controllers and one *S* function. One PI controller (defined as a block named “power reference calculating”), in middle top of Fig. 18.5a, uses both active power control commands from a system operator and the actual power generated in that wind farm as its inputs, and then derives the active power reference of this wind farm as its output signal. Similarly, another PI controller calculates the reactive power reference of the wind farm.

The active power and reactive power allocation strategies to each wind generator are realized by an *S* function in MATLAB. As illustrated in Fig. 18.6, the model receives the active and reactive power references, power predictions of a wind farm, power control modes from a system operator, etc., and then, the dispatch value of each wind generator will be calculated based on the active power and reactive power references of the whole wind farm through respective control strategies presented in previous subsections.

Based on Fig. 18.5, a *.m file is generated as an interface connecting the two platforms. The configuration of this interface file can be written as follows:

Function `[t, x, y] = PowerAllotControl_interface`

```
% "PowerAllotControl_interface" is name of the interface which connects
% Power Factory and Matlab
```

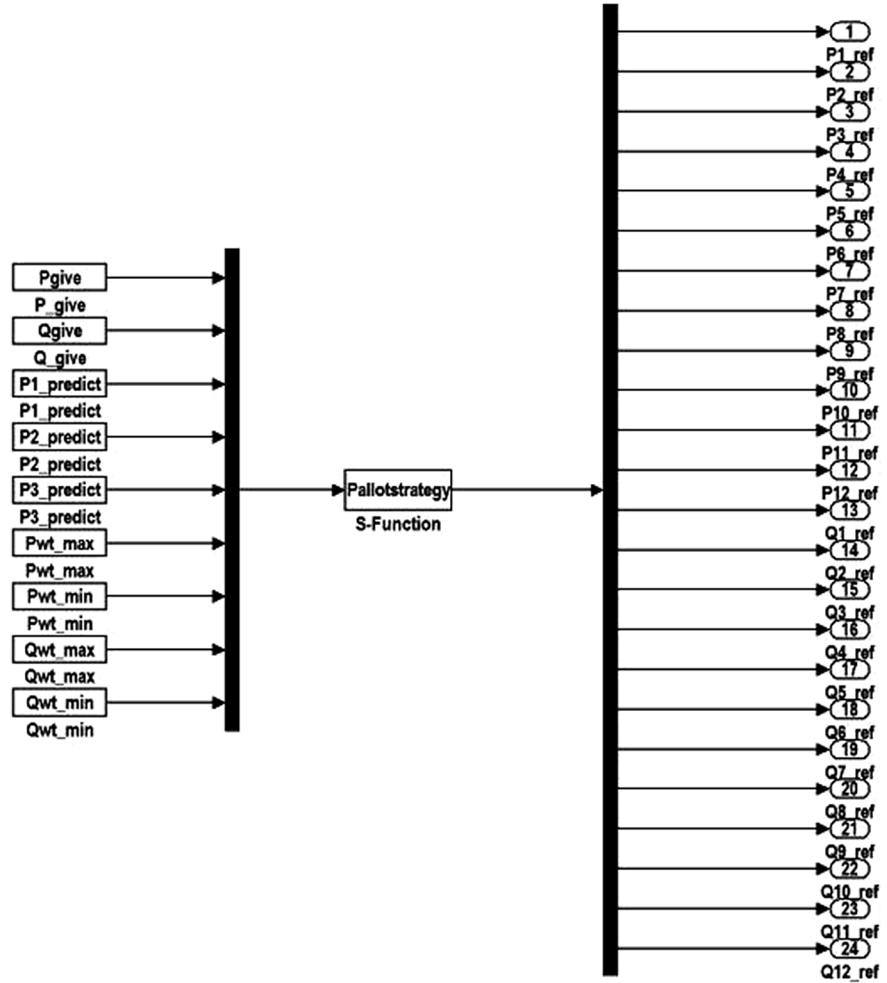


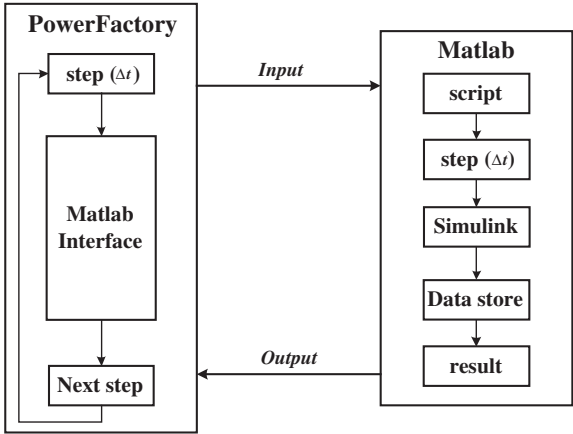
Fig. 18.6 Control strategy model implemented in MATLAB

```
global Pgive Qgive P1_predict P2_predict P3_predict Pwt_max Pwt_min Qwt_max
Qwt_min
% Definition of global variables which have also been defined in Power Factory

[t, x, y] = sim('PowerAllot_Control', [0.01]);
% "PowerAllot_Control" is the project name of Matlab/Simulink model
```

The initial conditions of the Simulink model in MATLAB are calculated from the power flow solutions in PowerFactory. In addition, the power flow also provides initial conditions for dynamic simulations in PowerFactory. It should be noted that the simulation is done interactively between PowerFactory and MATLAB rather

Fig. 18.7 Co-simulation of MATLAB/Simulink using PowerFactory interface



than in parallel. In each simulation step, PowerFactory inserts all input signals as “global variables” into the “workspace” that can be used by the external models in MATLAB. The configuration of the co-simulation between PowerFactory and MATLAB [17] is shown in Fig. 18.7. MATLAB simulates one time step and then returns the results for each of t , x and y , where t is a vector of time, x represents a matrix of state variable values and y represents a matrix of output values. Since only one time step is simulated each time in MATLAB, the starting and ending values will be returned to MATLAB *.m file that will be used in the DSL model in PowerFactory to calculate the derivatives of the state variables and outputs [15].

18.4 Case Studies

18.4.1 Testing Wind Farm

Among our engineering practices on active as well as reactive power controls in real wind farms in China, a wind farm in Gansu province is chosen here as an example for modelling a wind farm and controlling its active power and reactive power. This wind farm has an installed capacity of 200 MW, consisting of 134 wind turbines. All wind generators are DFIGs. The wind generator model described in Sect. 18.2 is applied here. Wind generators are connected to the PCC by 12 feeders, and the schematic diagram of this wind farm is shown in Fig. 18.8.

Since a wind farm always contains lots of wind turbines distributed in a large geographical area, wind speeds experienced by each wind turbine are different due to the wake effect and time delays. In order to reflect real operation states of wind turbines, the measured actual wind speed data are employed in simulations. In PowerFactory, the data can be fed into a wind turbine model using measurement file (ElmFile) format. Data of different wind speeds of each feeder in the wind farm are stored in ElmFile. Then, simulations in PowerFactory can apply measured real wind speeds in order to reflect wind farm’s practical operation characteristics.

18.4.2 Realization of Control Strategies

18.4.2.1 Realization of Active Power Control

The wind farm model established in Sect. 18.4.1 is employed to validate the functionalities of the active power control system. In order to operate wind turbines in accordance with instructions from the active power control system in the wind farm, some guides are listed below:

1. Deactivate the active power control of an individual wind turbine, which is defined in control frame shown in Fig. 18.1, so each control model can receive the active power control instructions from the wind farm power controller shown in Fig. 18.9a.

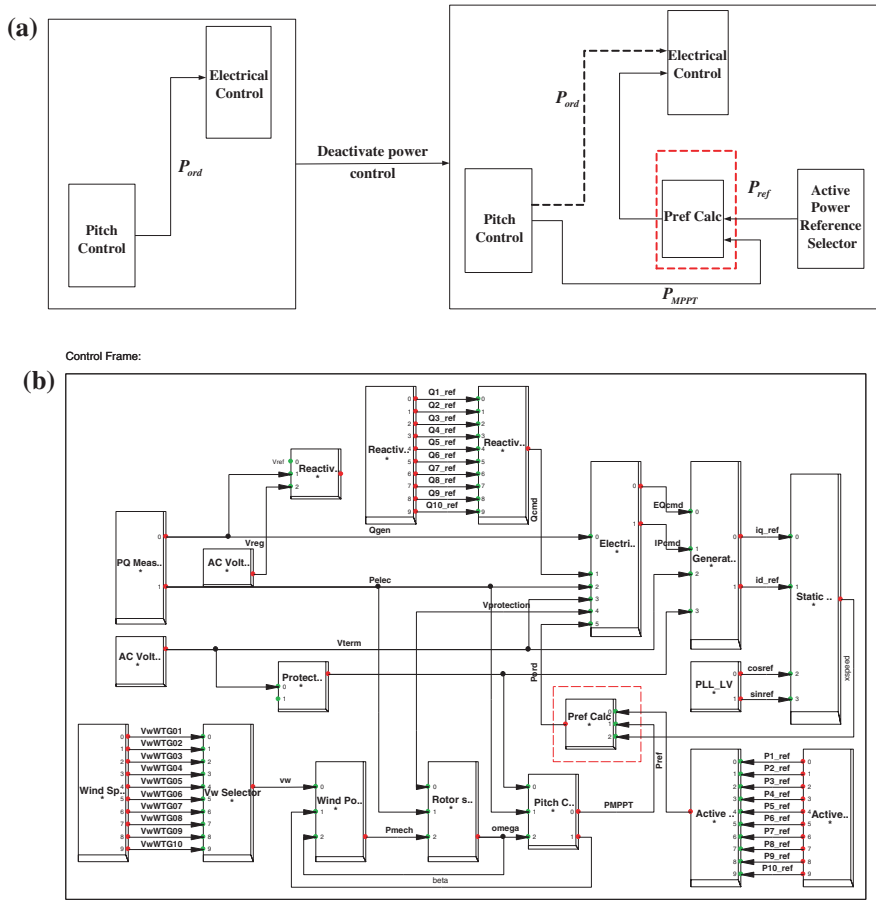


Fig. 18.9 Realization of active power control system in PowerFactory. **a** Schematic of active power control deactivation. **b** Active power control realization of DFIG in PowerFactory. **c** Parameters setting of feeders for active power control system. **d** Parameters setting and calling page

- 2. Set the active power reference of each wind generator to the calculated control command from MATLAB, so the actual active power output of each wind generator can be regulated to the reference value.
- 3. The actual active power output of a wind generator is influenced by the wind speed and the wind farm power controller: when the MPPT value is higher than the active power reference set by the wind farm power controller, the actual active power output will trace the power reference; otherwise, the active power output will trace the MPPT value.

Thus, a slot named “Pref Calc” should be added to decide the actual active power reference set for each wind generator in the “control frame” (see Fig. 18.9a, b, marked in dotted box), according to the following:

$$P_{ord} = \min(P_{ref}, P_{MPPT}) \tag{18.11}$$

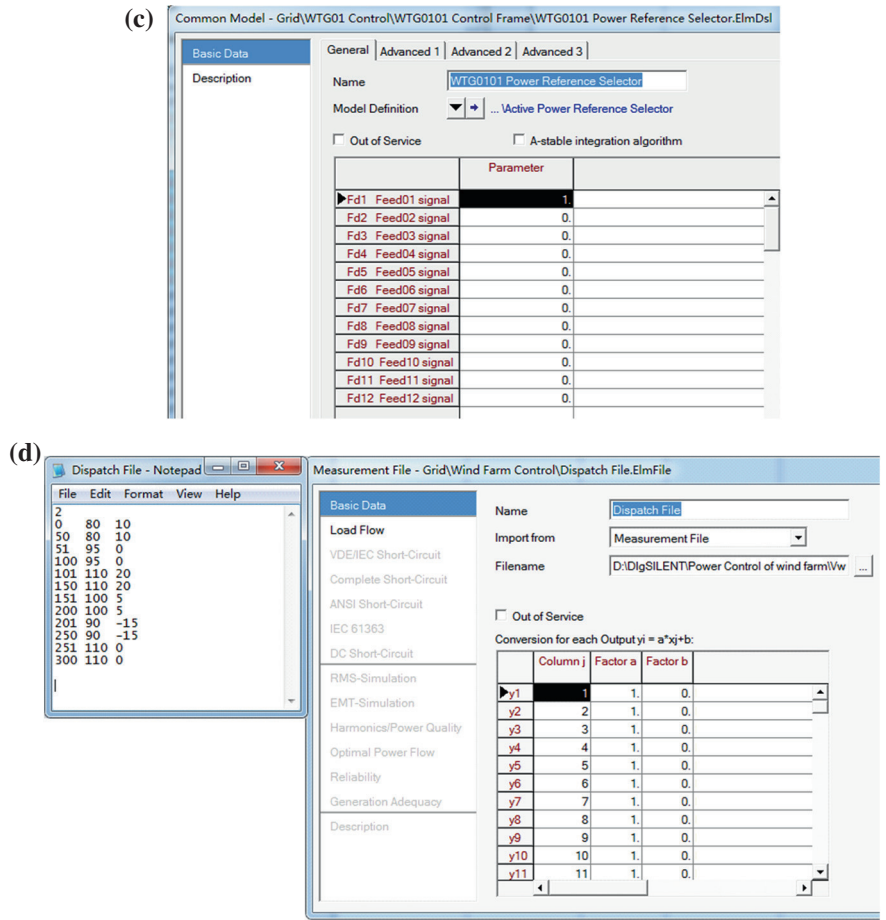


Fig. 18.9 (continued)

where P_{ord} is the actual active power reference of a wind generator, P_{ref} is the active power reference from wind farm power controller and P_{MPPT} is the MPPT value of a wind turbine.

As discussed in Sect. 18.1, to fully utilize the characteristics of wind generator configurations in a wind farm, a zone-based control strategy is taken. In a wind farm, wind turbines connected to the same feeder can be treated as a zone since they often are completely identical. For this purpose, a slot named “active power reference selector” is added to set the power references for different feeders (zones). The input of this slot is active power control instructions for wind turbines in that zone based on MATLAB calculation, while the slot output is active power control instructions for each wind generator in that zone. To reduce the model complexity, each generator is identified in the common model by a unique ID, such as Fd1, Fd2, ..., Fd12 in this example, shown in Fig. 18.9c. These ID parameters will be set when the respective “block” is called by “common model”. Take wind turbine WTG0101 as an example. Suppose it is connected to feeder #1. Then, the parameter Fd1 of the “common model” (WTG0101 active power reference selector) should be set to 1 and the others equal to zero. Similarly, parameter Fd2 should be set to 1 and the rest to zero for wind turbine WTG0201. The wind farm active power control system receives power references from a system operator, and these power references are stored in a text file. Different power references, including both active power and reactive power, are stored in corresponding columns of this text file as seen in Fig. 18.9d.

18.4.2.2 Realization of Reactive Power Control

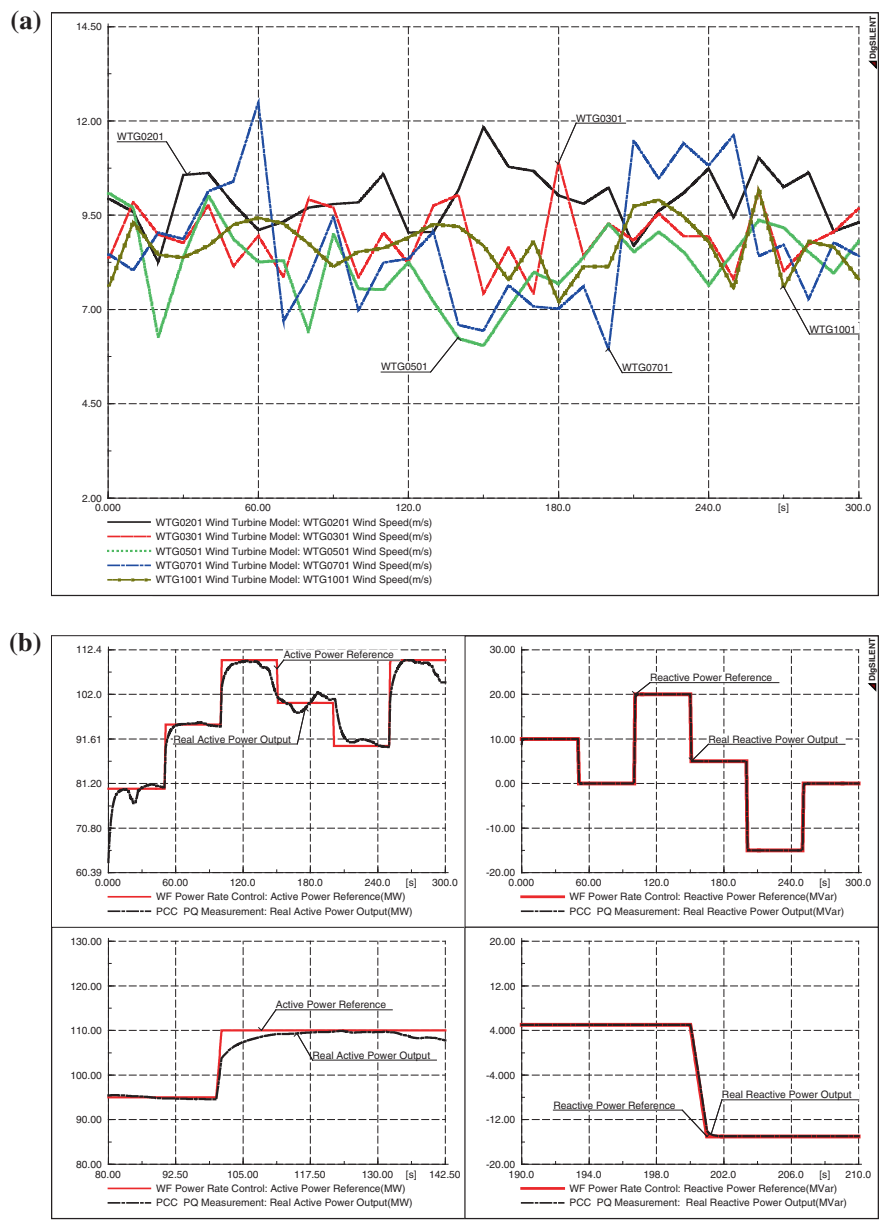
The wind farm model established in Sect. 18.4.1 is also applied to validate the functions of the reactive power control system. In order to operate the wind turbine correctly under instructions from the reactive power control system, some steps need to be taken as follows:

1. Deactivate the reactive power control signal in the user-defined control frame for each individual generator, so the control model can receive the reactive power control instructions from the wind farm reactive power controller.
2. The actual reactive power output will be controlled to follow references from a system operator by regulating the reactive power output of each wind generator. The reactive power control commands are calculated in MATLAB.

Similar to the active power control in Sect. 18.4.2.1, a slot named “reactive power reference selector” is created in order to choose the appropriate reactive power reference.

18.4.3 Simulation Analysis

This section describes the simulation results of a wind farm active and reactive power control system based on the integrated platform of PowerFactory and MATLAB.



Different control instructions from system operators are simulated, and control strategies of the wind farm are tested through comparisons between the responses of the wind farm and assumed power references. The control sequences are set as follows

(assume that reactive power regulation capacity of wind generators in the researched wind farm can meet demands from a system operation centre):

Settings of active power: During the first 50 s, the power reference is set to 80 MW; at $t = 51.0$ s, the power reference is stepped up to 95 MW and lasts 50 s; at $t = 101.0$ s, the power reference is again stepped up to 110 MW and is maintained at this value for 50 s; at $t = 151.0$ s, the power reference is stepped down to 100 MW and lasts 50 s; at $t = 201.0$ s, the power reference is again stepped down to 90 MW and lasts 50 s; and at $t = 251.0$ s, the power reference is stepped up to 110 MW.

Settings of reactive power: The initial reactive power reference is 10 MVar and lasts 50 s; at $t = 51.0$ s, the power reference is stepped down to 0 MVar and is maintained at this value for 50 s; at $t = 101.0$ s, the power reference is stepped up to 20 MVar and lasts 50 s; at $t = 151.0$ s, the power reference is stepped down to 5 MVar; and at $t = 201.0$ s, the power reference is stepped down again to -15 MVar and after 50 s it is stepped up to 0 MVar.

The simulation lasts for 300 s. Actual wind speeds shown in Fig. 18.10a are used as inputs to wind turbines; the active and reactive power outputs of the researched wind farm are also shown in Fig. 18.10b.

It can be seen clearly from Fig. 18.10 that the developed wind farm active power and reactive power controllers are able to control power outputs of wind generators in the wind farm to reach the power references imposed by the system operator with good dynamic responses in terms of the response speed and overshoots. Specially, since the active power control will be affected by wind speeds, if dispatch command from the operator is higher than the maximum output of a wind farm, the wind turbine will work in MPPT mode, and the active power output of the wind farm will be less than the dispatched value (as shown in Fig. 18.10b). Furthermore, the active and reactive power controls are decoupled in DFIG. In this case, the response speed of the reactive power control is obviously faster than that of the active power control as shown in Fig. 18.10b. In summary, the simulation results show that total power outputs at PCC of this wind farm can meet the demands from system operators and the wind farm operation performances could satisfy the Grid Codes.

18.5 Conclusion

This chapter presents an integrated simulation platform using PowerFactory and MATLAB for active power and reactive power control in wind farms. The main conclusions of this work are as follows: (1) An integrated platform combining PowerFactory and MATLAB could take full advantages of both simulation softwares; (2) A wind farm model in PowerFactory and respective controller models in MATLAB are developed and integrated successfully; and (3) Applications to a real wind farm in China validate the feasibility and correctness of the integrated platform.

This integrated platform has been widely applied in testing and analysing performances of wind farm power controllers all over China. Further investigations on using PowerFactory and MATLAB to build an integrated simulation platform for mid- and long-term stability analysis are undergoing.

References

1. Su C et al (2013) Mitigation of power system oscillation caused by wind power fluctuation. *IET Renew Power Gener* 7:639–651
2. Suriyaarachchi DHR et al (2013) A procedure to study sub-synchronous interactions in wind integrated power systems. *IEEE Trans Power Syst* 28:377–384
3. Gonzalez-Longatt F et al (2012) Probabilistic assessment of operational risk considering different wind turbine technologies. In: 3rd IEEE PES international conference and exhibition, innovative smart grid technologies (ISGT Europe), pp 1–6
4. Rueda JL, Erlich I (2011) Impacts of large scale integration of wind power on power system small-signal stability. In: 4th International conference on electric utility deregulation and restructuring and power technologies (DRPT) 2011, pp 673–681
5. Sørensen P et al (2005) Wind farm models and control strategies. *Risø-R-1464*
6. Sørensen P et al (2005) Operation and control of large wind turbines and wind farms. *Risø-R-1532*
7. Kayikçi M, Milanović JV (2007) Reactive power control strategies for DFIG-based plants. *IEEE Trans Energy Convers* 22:389–396
8. Karthikeya BR, Schütt RJ (2014) Overview of wind park control strategies. *IEEE Trans Sustain Energy* 5:416–422
9. Chang-Chien L-R et al (2014) Modeling of wind farm participation in AGC. *IEEE Trans Power Syst* 29:1204–1211
10. Pham HV et al (2014) Online optimal control of reactive sources in wind power plants. *IEEE Trans Sustain Energy* 5:608–616
11. Future IEC 61400-27-1 (2013) Wind Turbines—Part 27-1: Electrical simulation models for wind power generation. PNW 88-464 Ed. 1.0, Committee Draft for Voting 2013-11-06
12. Kayikçi M, Milanović JV (2008) Assessing transient response of DFIG-based wind plants—the influence of model simplifications and parameters. *IEEE Trans Power Syst* 23:545–554
13. Clark K et al (2010) Modeling of GE wind turbine-generators for grid studies, Version 4.5. General Electric International, Inc., Schenectady
14. Asmine M et al (2011) Model validation for wind turbine generator models. *IEEE Trans Power Syst* 26:1769–1782
15. DIgSILENT (2013) PowerFactory User's Manual 15.0. DIgSILENT GmbH Technol
16. Chen N et al (2009) Strategy for reactive power control of wind farm for improving voltage stability in wind power integrated region. *Proc CSEE* 29:102–108
17. Stifter M et al (2013) Steady-state co-simulation with PowerFactory. In: Workshop on modeling and simulation of cyber-physical energy systems (MSCPES) 2013, pp 1–6
18. Kerahroudi SK et al (2013) Initial development of a novel stability control system for the future GB transmission system operation. In: 48th International universities power engineering conference (UPEC) 2013, pp 1–6
19. Kong X et al (2013) Co-simulation of a marine electrical power system using PowerFactory and MATLAB/Simulink. In: IEEE electric ship technologies symposium (ESTS) 2013, pp 62–65
20. Eriksson R (2014) Coordinated control of multiterminal DC grid power injections for improved rotor-angle stability based on Lyapunov theory. *IEEE Trans Power Delivery*. published on line

Chapter 19

Implementation of Simplified Models of Local Controller for Multi-terminal HVDC Systems in DIgSILENT PowerFactory

Francisco M. Gonzalez-Longatt, J.M. Roldan, José Luis Rueda,
C.A. Charalambous and B.S. Rajpurohit

Abstract The North Sea has a vast potential for renewable energy generation: offshore wind power, tidal and wave energy. The voltage source converter (VSC) and high voltage direct current (HVDC) systems are more flexible than their AC counterparts. This offers distinct advantages for integrating offshore wind farms to inland grid system. It seems that advances on technologies open the door for VSC-HVDC systems at higher voltage and at higher power range, which is making multi-terminal HVDC (MTDC) system technically feasible. The control system for MTDC consists of a central master controller and local terminal controllers at the site of each converter station. The terminal controllers (outer controllers) are mainly responsible

Electronic supplementary material The online version of this chapter (doi:10.1007/978-3-319-12958-7_19) contains supplementary material, which is available to authorized users.

F.M. Gonzalez-Longatt (✉)
School of Electronic, Electrical and Systems Engineering, Loughborough University,
LE11 3TU Loughborough, UK
e-mail: fglongatt@fglongatt.org

J.M. Roldan
Escuela Superior de Ingenieria, Universidad de Sevilla, 20134 Seville, Spain
e-mail: jmroldan@us.es

J.L. Rueda
Department of Electrical Sustainable Energy, Delft University of Technology,
Mekelweg 4, 2628 CD Delft, The Netherlands
e-mail: j.l.ruedatorres@tudelft.nl

C.A. Charalambous
Department of Electrical and Computer Engineering, University of Cyprus,
75 Kallipoleos Avenue, 1678 Nicosia, Cyprus
e-mail: cchara@ucy.ac.cy

B.S. Rajpurohit
School of Computing and Electrical Engineering, Indian Institute of Technology Mandi,
175001 Mandi, Himachal Pradesh, India
e-mail: bsr@iitmandi.ac.in

for active power control, reactive power control, DC voltage regulation and AC voltage regulation. Typical MTDC consists of several VSC-HVDC terminals connected together, and different operation mode and controllers allows them interact together. DC voltage controllers play a very important role on the DC network performance. There are several DC voltage control strategies possible: voltage margin, two-stage direct voltage controller, three-stage direct voltage controller, voltage droop, etc. The contribution of this book's chapter is to present some of the main aspects regarding the modelling and simulation of two control strategies: voltage margin method (VMM) and standard voltage droop (SVD). To this end, theoretical aspects of controllers are presented and are used to develop DIGSILENT simulation language (DSL) models. The developed models are used to evaluate the performance a simple 3-terminal HVDC system.

Keywords HVDC transmission • HVDC converter • Load flow analysis • VSC-HVDC

19.1 Introduction

Electrical power systems have been developed, over more than 50 years, to deliver electricity to end-users; this approach requires using a vital infrastructure to link the energy producers and the consumers. That approach of power systems design and operation has served their purpose with great success for many decades mainly because they were developed to meet the needs of large and predominantly carbon-based energy producers located remotely from the load centres. Nowadays, power systems must cope with three driving forces of change [1]:

1. **Environmental constraints based on climate change.** The climate conference in Kyoto was the first time internationally binding targets for the reduction of greenhouse until 2012 by 5.2 % compared to 1990. The *United Nations Climate Change Conference* was held in Cancun in 2010 [2] and has agreed to continue the implementation of the Kyoto only without setting new targets for the period after 2012. However, the European Union (EU) has been seriously committed to CO₂ reduction itself. In 2007, it was agreed that the target triple of supply, competitiveness and environment should reduce the CO₂ emissions by 2020 by at least 20 % compared to 1990. Several stakeholders argue that this would not be sufficient to limit the effect of warming process of the atmosphere within 2 °C and for this reason, the EU considered to increase the reduction target to 30 % by 2020. By 2040, emissions are to be reduced by 60 %. With the use of appropriate technologies, no CO₂ should be emitted by the power generation industry by 2050 [3]. Reducing the greenhouse gas emissions by 80 % is the specific target of the UK government by 2050 [4]. This target is defined on the Climate Change Act 2008 [5]. De-carbonising the power sector is the key factor to reach this objective, and this will enable further low-carbon choices in the transport sector (e.g. plug-in hybrid and electric vehicles) and in buildings (electric heat pumps).

2. **Security of Energy Supply.** Over the coming decades, governments around the world face a daunting challenge in meeting the energy needs of a growing and developing world population while mitigating the impacts of global climate change. Security of supply is an important goal of energy policy in many countries around the world. The importance of energy security derives from the critical role that energy plays in all aspects of everyday's and business' life [6]. As demand for resources rises within today's turbulent global markets, supply chain vulnerability is becoming a significant issue. Global sourcing has created more complex and increasingly risky supply chains. Severe energy security has serious implications for social, environmental and economic well-being. The conversion of the centralised power generation structures that are currently using imported primary fuels such as coal, oil, gas and uranium to more decentralised renewable power plant systems opens the chance of reducing the import dependence from fossil energy sources. Europe as a whole is a major importer of natural gas. Apart from Norway, Russia remains one of Europe's most important natural gas suppliers. Europe's natural gas consumption is projected to grow while its own domestic natural gas production continues to decline. Increasing energy efficiency is clearly the most cost-effective part of the energy revolution. The UK's government has been working on energy security for years, making sure consumers can access the energy they need at prices that are not excessively volatile. It has been reached by a combination of its liberalised energy markets, firm regulation and extensive North Sea resources. The Department of Energy and Climate Change of UK is actively working in several aspects in order to guarantee the energy system has adequate capacity and is diverse and reliable [7].
3. **Economic development.** Development in electric power systems must contribute to growth and in parallel minimise the costs attributed to consumers. It is necessary to maintain a right balance between investing in generation, non-generation balancing technologies (i.e. storage, demand-side response and interconnection) and network assets. In addition, the efficient operation of power systems is critical to maximising the efficient use of assets across the system. When conventional power is substituted by wind power, the avoided cost depends on the degree to which wind power substitutes each of three components—fuel cost, O&M costs and capital. The economic competitiveness of wind power generation will depend on short-term prediction, and specific conditions for budding into short-term forward and spot markets at the power exchange. Some calculations demonstrate that although wind power might be more expensive than conventional power today, it may nevertheless take up a significant share in investors' power plant portfolios as a hedge against volatile fossil fuel prices [8]. Continuing research and development work is needed in order to ensure wind power is to continue reducing its generation costs and sustainable economy growth.

While current networks presently fulfil their function, they will not be sufficient to meet the future challenges described above. These challenges require technical, economic and policy developments in order to move towards lower-carbon generation technologies as well as higher efficiency devices and systems [1].

The radical changes that power systems are undergoing will change the landscape of future power networks. One of the technical challenges is the development of a Pan-European transmission network to facilitate the integration of large-scale renewable energy sources and the balancing and transportation of electricity based on underwater multi-terminal high voltage direct current (MTDC) transmission [1, 9, 10].

This kind of interconnection will facilitate markets to import and export electricity according to the market prices on either side of the interconnector using larger distances and lower losses. Increased amounts of interconnection have the potential to bring savings to the system where connected markets have different generation and/or demand profiles to trade. In such circumstances, interconnection could result in generation capacity being dispatched more efficiently and reducing the total generation capacity required. The existing power grid in Europe is a highly interconnected system, spanning the whole of Continental Europe with connections to neighbouring systems, e.g. in Scandinavia (Nordel), the UK and Russia. The current structure of this meshed, supra-national system was largely influenced by available generation technologies. The UK electricity network is connected to the systems in France (National Grid and Réseau de Transport d'Electricité, 2 GW), Northern Ireland (IFA, 2 GW) and the Netherlands (BritNed, 1GW) through "interconnectors", with others under construction or planned. Potential future interconnector opportunities include interconnectors between UK and Belgium (Nemo Link), Norway (2 GW), France, Denmark and Iceland.

Supergrid is the name of this future electricity system that will enable Europe to undertake a once-off transition to sustainability [11]. The electricity transmission system involved on supergrid should be mainly based on direct current (DC), designed to facilitate large-scale sustainable power generation in remote areas for transmission to centres of consumption, one of the fundamental attributes being the enhancement of the markets in electricity trading [12]. The North Sea has a vast amount of wind energy with largest energy per area densities located about 100–300 km of distance from shore [13]. MTDC transmission would be the more feasible solution at such distances of subsea transmission. There are several advantages of use of MTDC system, but two of those make it suitable for a massive deployment in future power systems: it allows a higher efficiency on the bulk power transmission over long distance and it provides a very high controllability in terms of power flows maximising the integration of variable power coming from renewable energy resources.

There are two kinds of HVDC transmission technology [14]: Line commutated converter (LCC)-based HVDC and voltage-sourced converter (VSC)-based HVDC. LCC-HVDC has several disadvantages [15]: it cannot perform self-restoration upon disconnection from the connected AC grid nor provide black start to the connected AC grid, in order to reverse the power transmitted; the DC voltage must be reversed. VSC-HVDC is superior to LCC-HVDC: risk of commutation failure is reduced using self-commutated switches, communication is not needed, it has black start compatibility and it has superior controllability: it is capable of independent control of active and reactive power flow.

Supergrid will probably grow in stages from connecting one offshore wind farm to one onshore grid towards linking several far offshore wind farms to multiple onshore grids [16]. It will include many HVDC cables to integrate all offshore wind power systems. When this kind of DC grid is built, connections must be made at the DC bus, multiple undersea cables and multiple converters at the same bus creating a MTDC configuration. For this, VSC-HVDC is the most appropriate technology as it uses a common DC voltage and injects a variable current [17, 18].

This book's chapter presents a discussion of the main modelling and simulation aspects of two control strategies used for MTDC: voltage margin method (VMM) and standard voltage droop (SVD). The organisation of the chapter is as follows:

Section 19.2 discusses the theoretical aspects of controllers and it is used to develop DIgSILENT simulation language (DSL) models in Sect. 19.3. Developed models are tested and validated using a simple 3-terminal HVDC system in Sect. 19.4.

19.2 Control Strategies for MTDC Network Operation

The control schemes have a large impact on system dynamics. It is an important task to determine the modelling requirements of the control schemes. The control system for a MTDC is composed of two different layers of controllers [19–22]: (i) *terminal controllers* and (ii) a *master controller* as illustrated in Fig. 19.1.

19.3 Master Controller

The *master controller* is provided with the minimum set of functions necessary for coordinated operation of the terminals in the DC circuits [20] i.e. start and stop, minimisation of losses, oscillation damping and power flow reversal, black start, AC frequency and AC voltage support. This controller optimises the overall performance

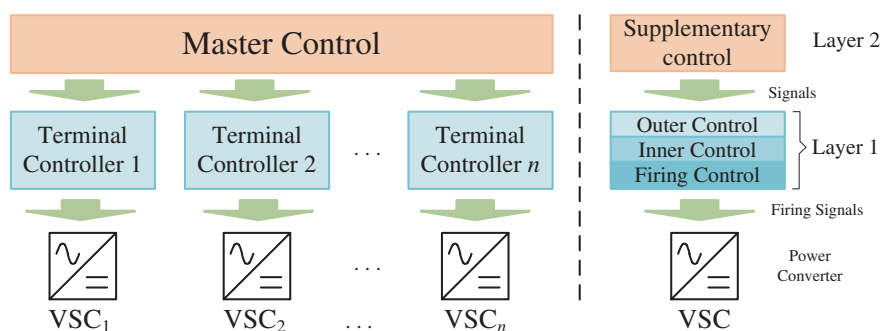


Fig. 19.1 Schematic representation of MTDC control system hierarchy

of the MTDC by regulating the DC side voltage. They are not necessary for the operation of the MTDC system, but greatly enhance their functionality.

- **Frequency control:** Frequency control is indispensable when a converter is located in a passive system, but it can also be used in an active power system. Frequency is regulated by modulating active power.
- **Damping control:** A converter can damp oscillations occurring in the AC power system by an additional controller. Input signals can be local, or not local, such as generator speeds, which may require communication. The output signal modulates active power, so that the active power swing is counteracted.
- **AC voltage control:** Instead of directly controlling AC voltage in the outer controller, an additional loop can be created around the reactive power control. The reactive power set point is determined from the desired AC voltage.

In this chapter, no models for supplementary controllers are developed. Apart from the fact that MTDC systems can operate perfectly well without supplementary controllers, the reason is that it is not desirable to come up with generic models of master controllers.

19.4 Terminal Controller

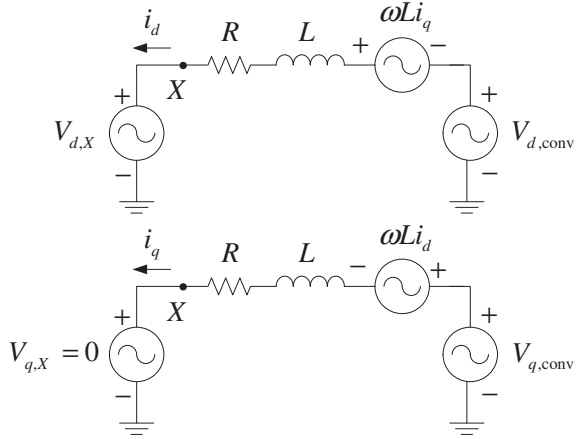
The *terminal controller* controls the specific converter by calculating the pulse-width modulation (PWM) pulses for the converter bridges. The firing controller is the fastest controller and inner control, outer control and supplementary control are used for increasingly higher level functions and have increasingly higher cycle times. This MTDC control system is implemented on a hierarchical way. It is a cascaded system, where every level accepts the input of the previous one and feeds its output signal to the next level. This is schematically represented in Fig. 19.1.

19.5 Firing Control

Firing control is the lowest control level inside the terminal control system and it acts very fast. Firing control, also known as valve control, takes the desired converter waveform as an input and determines by means of the valve firing logic the pulses that need to be generated [23]. The firing logic is communicated to the controllable switches (e.g. IGBTs GTO's), and pulses are generated that switch on/off the switched at the appropriate instants. The firing instances are synchronised using a phase-locked loop (PLL).

The pattern of the pulses depends on the topology of the bridge and the switching method. As converters are considered to be a black box, the details of the converter topology are not known and modelling the firing control is impossible. However, firing control has cycle times in the few micro-seconds range. The use of

Fig. 19.2 Equivalent circuits in d and q axes of VSC-HVDC for d -axis aligned with voltage phasor of phase- a



the *space vector* in the control design and implementation enables to make a fully decoupled linear control of active and reactive currents.

The d - q reference frame is selected in such a way that the d -axis is aligned to the voltage phasor of phase- a of point X . This means that the PLL should be phase locked to phase- a voltage phasor of the reference point, X . This results in

$$\begin{aligned} V_{q,X} &= 0 \\ V_{d,X} &= V_X \end{aligned} \quad (19.1)$$

The simplified equivalent model of VSC-HVDC in d - q reference is shown in Fig. 19.2. From the d - q equivalent circuit as observing from the reference point X , the apparent power (S_{conv}) injected by the VSC converter into the AC network is given by:

$$S_{\text{conv}} = \frac{1}{2} (V_{d,X} + j0) (i_d - ji_q) \quad (19.2)$$

The active and reactive powers (P_{dq} , Q_{dq}) provided by the VSC-HVDC converter to the AC become:

$$P_{dq} = \frac{3}{2} V_{d,X} i_d \quad (19.3)$$

$$Q_{dq} = -\frac{3}{2} V_{d,X} i_q \quad (19.4)$$

19.6 Inner Controller

The *inner control or current control loop* is designed to be much faster than the outer controllers. It is not fast enough, however, to warrant neglecting its dynamics. This means current controllers and all relevant controllers higher in the hierarchy must be modelled. This control system controls the current through the phase reactor. Decoupled control is used, which means that voltages and currents are decomposed in dq -components, controlled independently [23]. The output of the current control is the desired converter voltage.

The inner current controller is developed based on the following equation.

$$L \frac{d}{dt} \begin{bmatrix} i_d \\ i_q \end{bmatrix} = \begin{bmatrix} V_{d,X} \\ V_{q,X} \end{bmatrix} - \begin{bmatrix} V_{d,\text{conv}} \\ V_{q,\text{conv}} \end{bmatrix} - r \begin{bmatrix} i_d \\ i_q \end{bmatrix} - \omega L \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} i_d \\ i_q \end{bmatrix} \quad (19.5)$$

An implementation of Eq. (19.5) is presented in Fig. 19.3, and it shows the d - and q -components of current controllers of the inner current loop.

The power converter has a time delay caused by the sinusoidal pulse-width modulator and it can be approximated as:

$$e^{-T_w s} \approx \frac{1}{(1 + T_w s)} \quad (19.6)$$

where time delay is defined by $T_w = 1/2 f_s$, where f_s is the switching frequency of the converter. Proportional integral (PI) controllers are used for closed loop control and the zeroes of the PI controllers are selected to cancel the dominant pole in the external circuit [13]. The time constant $\tau = L/r$ is much higher than T_w for a typical VSC, and hence will be the dominant pole to be cancelled. The cross-coupling currents in Eq. (19.4) are compensated by feed-forward terms in the controllers as shown in Fig. 19.3.

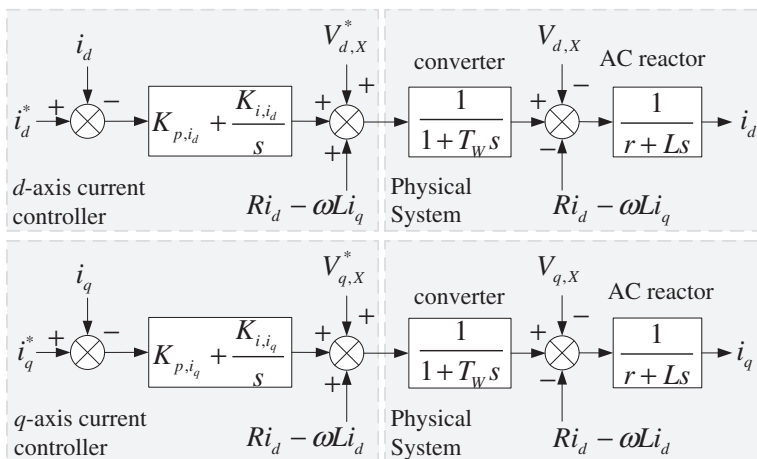


Fig. 19.3 Inner current controllers

19.7 Outer Controller

The *outer controllers* are the ones responsible for providing the current references signals for the inner current controller. The terminal controller determines the behaviour of the converter at the system bus. Several targets can be set [4, 12]:

- **Active power control:** determines the active power exchanged with the AC grid.
- **Reactive power control:** determines the reactive power exchanged with the AC grid.
- **AC voltage control:** instead of controlling reactive power, AC voltage can be directly controlled, determining the voltage of the system bus.
- **DC voltage control:** used to keep the DC voltage control constant.

The outer controllers have in common their provision of a current set point in the dq -frame for the inner current controller. Their behaviour directly influences the dynamics of the AC system and are therefore of paramount importance in modelling MTDC systems.

Active current (i_d) is used to control either of active power flow or DC voltage level. Similarly, the reactive current (i_q) is used to control either of reactive power flow into stiff grid connection or AC voltage support in weak grid connection.

19.7.1 Active Power Controller

The active power flow (P) of the VSC-HVDC terminal is given by Eq. (19.3):

$$P_{dq} = \frac{3}{2} V_{d,x} i_d \quad (19.3)$$

where V_x is resultant voltage in dq -reference frame and is desired to have a constant value. Hence, active power flow can be controlled by active current (i_d).

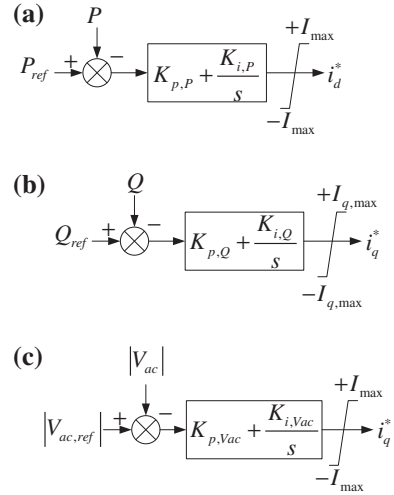
This controller allows an excellent control on the active power exchanged with the AC grid. An implementation of the active power controller is presented in Fig. 19.4a.

The output of the active power controller (i_d^*) provides the reference input to the d -axis current controller of the inner current loop in Fig. 19.3. The maximum current through the VSC-HVDC converter must be controlled in order to avoid potentially dangerous over currents on the commutation devices. In order to limit the magnitude of current in the VSC-HVDC to a maximum limit, the output of the active power controller is followed by a limiter function of $\pm I_{\max}$ limits, where:

$$-I_{\max} \leq i_d^* \leq +I_{\max} \quad (19.7)$$

where $I_{\max} = I_{\text{rated}}$.

Fig. 19.4 Outer controllers
a Controllers PI controller for active power control.
b Controllers PI controller for reactive power control.
c Basic scheme for AC voltage controller



19.7.2 Reactive Power Controller

The reactive power (Q) exchanged by the VSC-HVDC converter and the AC grid is given by Eq. (19.4):

$$Q_{dq} = -\frac{3}{2} V_{d,X} i_q \quad (19.4)$$

The reference of reactive current (i_d^*) is used to control the reactive power flow provided by the VSC-HVDC converter and an implementation of this controller is presented on Fig. 19.4b. As in the case for active power control, i_q^* will be the reference input for the reactive current controller of the inner current loop in Fig. 19.3.

The reference of reactive current (i_d^*) must be limited in order to avoid any damage on the commutation devices; as a consequence, i_d^* is limited to $\pm I_{q,max}$ in such a way that the total converter current should not exceed the rated current ($I_{max} = I_{rated}$). This takes the assumption that that priority is given to the transfer of active power. Hence:

$$I_{q,max} = \sqrt{I_{max}^2 - (i_d^*)^2} \quad (19.8)$$

19.7.3 AC Voltage Controller

A VSC-HVDC converter connected to a power system has the capability to control the AC voltage at the connection point; this feature is especially important on a weak grid, where there is significant line resistance and inductance creating a considerable amount of voltage fluctuations with changing active power flow.

This controller is designed to regulate the amplitude of the AC voltage (V_{ac}) at the common bus to be equal to the given reference value by modifying i_d^* . This implies that the controller governs the converter to generate an amount of reactive power so that the voltage at the common bus matches the given reference value ($V_{ac,ref}$). An implementation of this controller is presented on Fig. 19.4c.

19.8 DC Voltage Controller

DC voltage control is certainly one of the most important tasks given to the VSC-HVDC stations inside a MTDC network. A well-controlled DC voltage on a MTDC system is a guarantee of the power balance between all the interconnected nodes. Considering the operational requirements for DC voltage on MTDC, the literature provides two main control strategies which possibly can be applied in future transnational networks [24]: (i) the *direct voltage droop method* and the (ii) *voltage margin method*. These methods enable sharing of load among two or more DC voltage regulating terminals operating in parallel and provide controls in MTDC. Figure 19.7 shows a general scheme for VSC-HVDC system considering only two converter substations (Fig. 19.5).

19.8.1 Principle of Voltage Margin Method (VMM)

The *voltage margin* is defined as the difference between the DC reference voltages of the two terminals (ΔU_{dc}) [20]. Figure 19.6a shows the $U_{dc}-P$ characteristics of both terminals at Terminal A, and the intersection $U_{dc}-P$ of the characteristics of each terminal is the operating point “a”.

When the active power is to be transmitted from Terminal B to Terminal A ($P_A < 0$, $P_B > 0$), the voltage margin (ΔU_{dc}) is subtracted from the DC reference voltage for Terminal A. Terminal B (rectifier) determines the DC system voltage

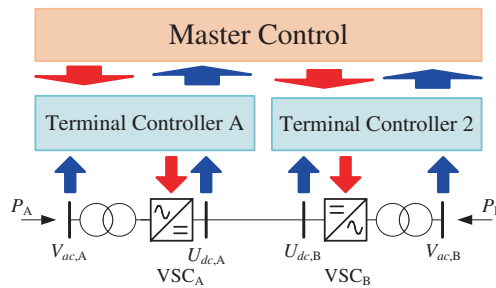


Fig. 19.5 General scheme for two converter stations VSC-HVDC system. VSC_i operates as inverter ($P_i < 0$) or rectifier ($P_i > 0$) depending in power direction

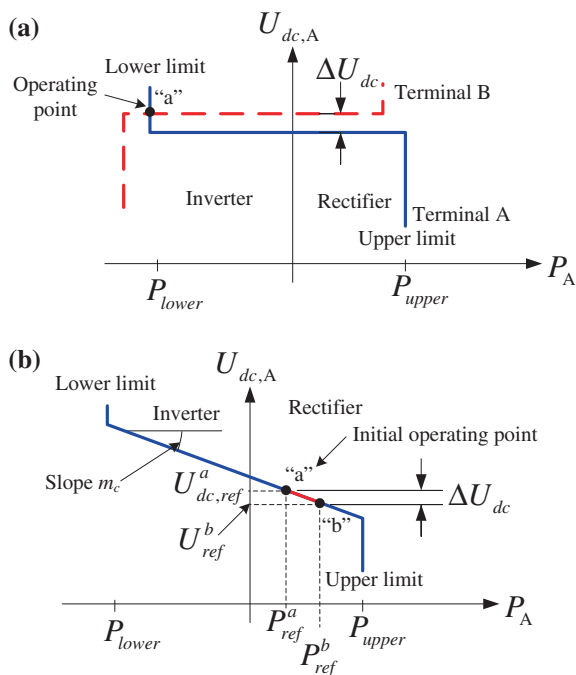
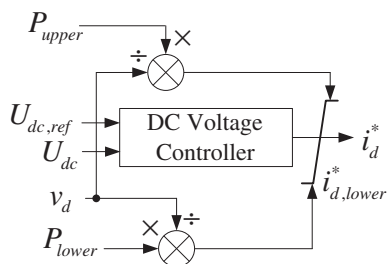


Fig. 19.6 U_{dc} - P characteristics of DC voltage controllers. **a** U_{dc} - P characteristic showing the operating point "a" in VMM for one terminal **b** U_{dc} - P characteristic showing the operating point "a" in VMM for one terminal

and Terminal A (inverter) controls the active power (P_A) determined by the lower limit of the DC voltage regulator. The DC voltage controller tries to keep the DC voltage to the reference value $U_{dc,ref}$ by adjusting P_A , until P_A reaches the upper limit or the lower limit (see Fig. 19.7).

The VMM gives reliable way of controlling MTDC without the need for communication between terminals and is capable of keeping the steady-state voltage with in pre-set limits even after load switching and disconnection of some converter terminals. But on the other hand, this method implies allocation of only one terminal at a time for the regulation of DC voltage and the other terminals do not experience significant change during changes in power flow of the DC network.

Fig. 19.7 Basic scheme for VMM controller with adjustable limits



19.8.2 Principle of Voltage Droop Method (VDM)

Frequency droop control is a well-established method and the basis for stable operation in all AC grids. The system's frequency is used as a global measure for the instantaneous balance between power generation and demand [25]. The *DC voltage droop method* is a coordinated control to maintain a power balance and a desired power exchange in the MTDC. This control is a modification of the *VMM* control where the horizontal line sections ($P_{\text{lower}} < P_A < P_{\text{upper}}$) of the $U_{\text{dc}}-P$ characteristic curves is replaced by a line with small slope (m_c) [26]. The DC voltage droop, m_c , indicates the degree of compensation of power unbalance in the DC grid at a cost of reduction in the DC bus voltage. This principle of *VMM* control is shown in Fig. 19.6b. When $U_{\text{dc},A}$ drops (e.g. due to large withdrawal of power someplace else in the DC network, operation point moves from “a” to “b”), the slack converter station (VSC_A) will increase the active power injection in the DC grid P_A until a new equilibrium point ($U_{\text{dc,ref}}^b, P_{A,\text{ref}}^b$), at a lower DC voltage, is reached ($U_{\text{dc,ref}}^b = U_{\text{dc,ref}}^a - \Delta U_{\text{dc}}$). The use of a proportional DC voltage controller allows multiple converters to regulate the voltage at the same time and the concept of distributed slack bus is possible.

Figure 19.8 shows how the droop characteristic is implemented based on the power active controller. When the voltage droop control is used in the absence of a PI controller, the voltage controller's active power P will change when the value of the DC bus voltage changes.

19.9 Dynamic Modelling in DIgSILENT Power Factory

DIgSILENT PowerFactory has developed highly flexible and accurate features for time-domain modelling. The DSL provided to PowerFactory the capability to define new dynamic controllers which receive input signals from the simulated power system and subsequently react by changing some other signals. DSL itself can be looked upon as an add-on to the transient analysis functionality of PowerFactory. The DSL language is used to programme models for the electrical controllers and other components used in electrical power systems.

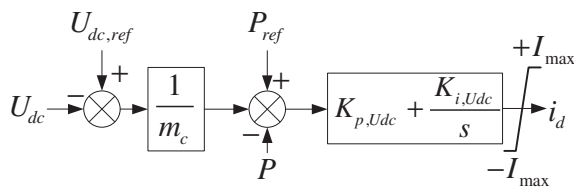


Fig. 19.8 Voltage droop controller

This simulation tool falls into the category of continuous system simulation language (CSSL), originally designed by the Simulations Council Inc (SCI) in 1967 in an attempt to unify the continuous simulations field. This programming approach allows the modelling and simulation of systems characterised by ordinary and partial differential equations.

DSL allows the definition of every linear or nonlinear system of differential equations, dead times (e.g. ideal wave equations), arithmetic or logic expression (e.g. digital controllers), and event (e.g. open breaker if $x > y$).

PowerFactory uses a *partitioned solution* with *explicit integration* method on the solution of the differential algebraic model (DAE) for power system dynamic analysis. During the time-domain simulation, the model equations of the DSL models are combined with those describing the dynamic behaviour of the power system components. These equations are then evaluated together, leading to an integrated transient simulation of the combination of the power system and its controllers.

PowerFactory modelling philosophy is targeted towards a strictly *hierarchical system modelling approach* as depicted in Fig. 19.9.

DSL is a very flexible language and it can be used for: (i) writing a DSL-programme, (ii) drawing a block diagram, or (iii) combination of both approaches. In this book’s chapter, a combination of drawing a block diagram and writing DSL-commands is used. Creating a general methodology to develop dynamic models using DSL is beyond the scope of this chapter, however, a simple 3-step procedure can be followed as general rule: *Step 1*: Create the Frame Diagram showing how the slots are interconnected, *Step 2*: Create each of the model definitions and set appropriate initial conditions, *Step 3*: Create a composite model and fill the slots with the relevant elements. The development of those steps on the case of inner and outer controller of a MTDC system is shown in the next sections.

19.10 Modelling a MTDC in DIgSILENT Power Factory

One of the main components on a MTDC system is the VSC-HVDC power converter. A PWM converter model (*ElmVscmono*) is used for VSC-HVDC converter stations; it represents a self-commutated, voltage source AC/DC converter (with a capacitive DC-circuit included). This built-in model (*ElmVscmono*) is shipped by default without any controls, for this reason DSL model for controllers must be

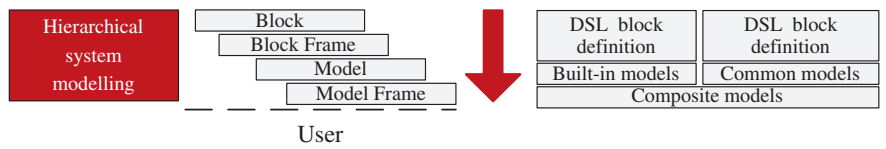


Fig. 19.9 Schematic representation of the DSL hierarchical system modelling approach

developed. A set of four input variables can be defined of the PWM converter model. It allows specifying a pulse-width modulation index-vector, with reference to a reference-system that is defined by \cosref and \sinref . Pmr and Pmi are the real and imaginary components of pulse-width modulation index based on the output of a dq -current controller.

The MTDC system is expected to consist of several VSC-HVDC terminals connected to each other through the DC network. Local control at each terminal should be able to adopt a control strategy depending on the specific needs described by the basic features required by MTDC system. The terminal controller must monitor DC and AC side and should control DC side parameters as well as AC side parameters. Using the bus parameter control on DC and AC side, several operation modes can be defined, a discussion about the all possible different control modes of VSC-HVDC terminal can be found in [13].

Two reactive power control functions are included into VSC-HVDC stations from the AC network side [15]: (i) Q -mode: the reactive power injected ($Q_{ac,i}$) into the AC network is kept constant, as consequence the AC voltage ($V_{ac,i}$) might change. (ii) V -mode: the reactive power converter injection ($Q_{ac,i}$) is enough to keep is AC node voltage magnitude ($V_{ac,i}$) constant. On the DC network side, there are two different control functions for each converter: (i) P_{ac} -control: The active power ($P_{ac,i}$) injected in the AC network is kept constant and the AC voltage ($V_{ac,i}$) is allowed to vary, (ii) U_{dc} -control: The converter controls its active power injection ($P_{ac,i}$) to keep its DC node voltage constant ($U_{dc,i}$).

Consider the schematic representation of the MTDC control system's hierarchy which is shown in Fig. 19.1. This section deals with the implementation of the inner and outer controller as presented in Sect. 19.2. Next sections show the DSL implementation of outer controllers for a MTDC system. The implementation presented in this chapter considers two possible operation modes for the terminals controllers: PQ -control and Q - U_{dc} -control. The components of pulse-width modulation index (Pmi and Pmr) are calculated using a dq -current controller.

19.10.1 Composite Frame

The composite model provides an overview diagram showing the interconnections between slots. Each block on those frames (slots) is placeholders for the models that describe their dynamic behaviour. A composites frame is a block definition object (*BlkDef*) which contains only slots and connectors, showing how the network elements and common models are connected together. Figure 19.10a and b shows the composite frame for PQ -control and Q - U_{dc} -control, respectively. Those composite frames contain the definitions of each slot, indicating the type of object assigned to the slot.

The composite frame for PQ -control consists of several slots: active and reactive measurement blocks, PLL measure system, model for active power control (P -controller), model of reactive power control (Q -controller), model of frequency

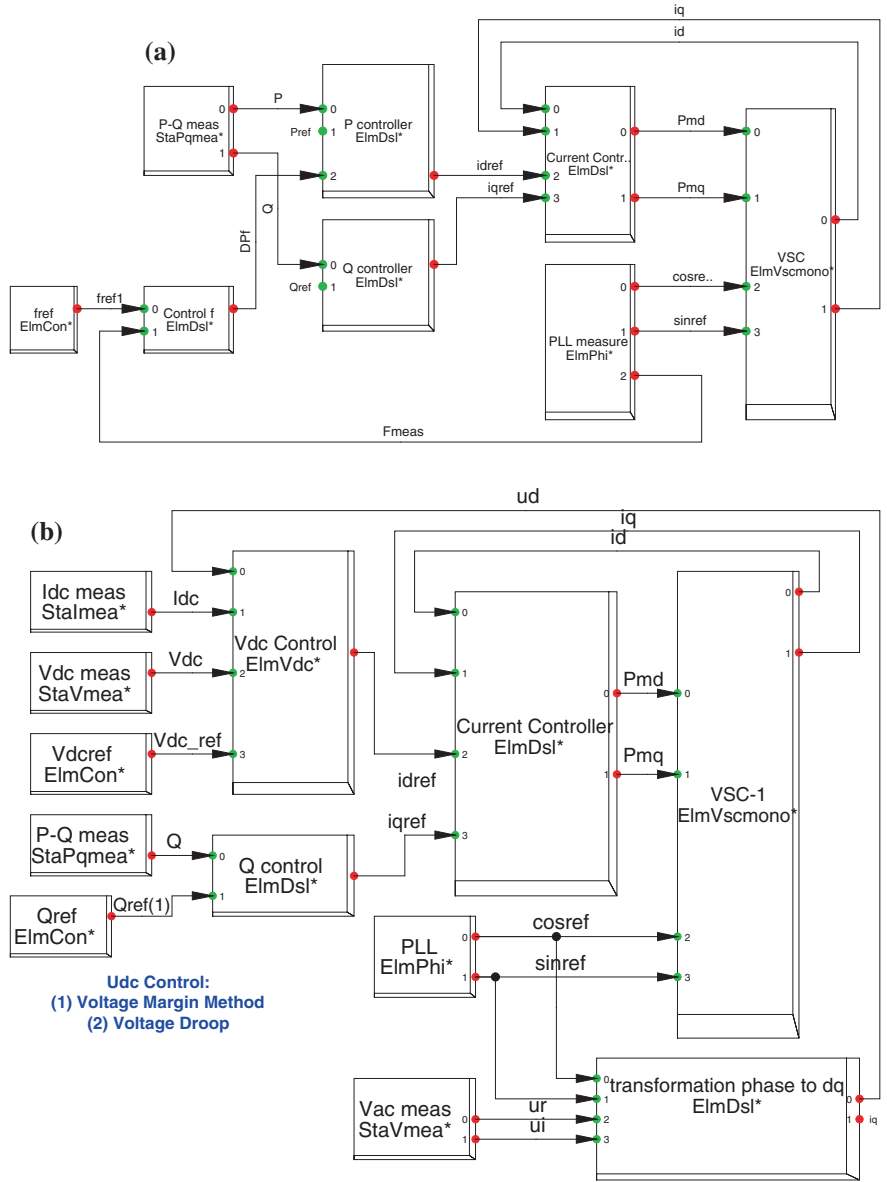


Fig. 19.10 Composite frames of terminal controllers in MTDC system. **a** Composite frame for *PQ*-control. **b** Composite frame for *Q-U_{dc}*-control

control (Control f), dq -current control frame (Current Controller) and finally the modulation indexes (Pmd and Pmq) which are fed into the power converter ($ElmVscmono$). The composite frame for Q - U_{dc} -control consists of several slots: active and reactive measurement blocks, PLL measure system, DC current and voltage measurements, AC voltage measurements, model for reactive power control (Q -controller), model DC voltage control (Vdc controller), dq -transformation used for the AC voltage, dq -current control frame (Current Controller) and finally the modulation indexes (Pmd and Pmq) which are fed into the power converter ($ElmVscmono$).

19.10.2 Model Definitions

The model that describes the dynamic behaviour of each controller on the frames must be defined by a block diagram, called in DSL as *model definition*. This defines the transfer function of a dynamic model, in the form of equations and/or graphical block diagrams. Figure 19.11 shows the model definitions created based on the controllers presented in Fig. 19.4a, b for active and reactive power controllers. A limited first-order transfer function $\{K(1 + 1/sT)\}$ is used to model the PI block. This is an equivalent version of the classical PI transfer function $(k_p + k_i/s)$ where $k_p = K$ and $k_i = K/T$.

The dq -current controller used in the DSL implementation of the inner controller is a slightly different version of the block diagram presented in Fig. 19.3.

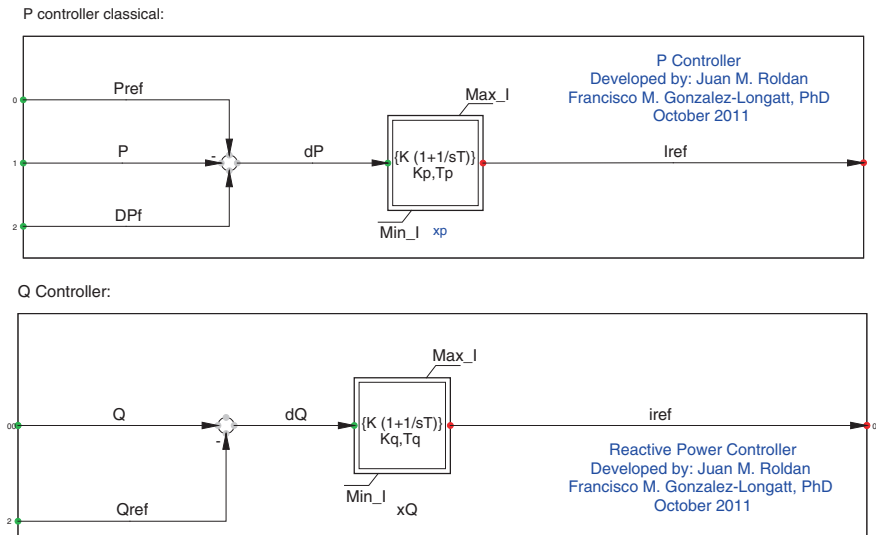


Fig. 19.11 Model definition for active and reactive power controllers

The DC voltage controllers are depicted in Fig. 19.13. A limited proportional integral function $\{K(1 + 1/sT)\}$ is used on the model definition and the DC control method for DC VMM and droop controller are presented in Fig. 19.13a, b, respectively.

19.10.3 Model Initialization

All DSL models must be initialized according to the steady-state conditions. In PowerFactory, the steady-state conditions are obtained from the load flow calculations (*ComLfd*) prior to a time-domain simulation. New user-defined DSL models require a proper initialization in order to reach correct results in time-domain simulations. In a DSL model, all variables or signals that cannot be determined directly from the load flow solution must be manually initialised. However, it must be noticed, not all variables and signals in a model need to be manually initialised. PowerFactory will try to use the model equations to compute its initial value. However, the classical error message will appear if the model equations have undefined variables or signals (e.g. an unknown input). The initialization process typically starts from the grid elements and goes backwards through the other models, fully initializing each block at a time.

The best procedure for composite block initialization is as follows: *Step 1*: clearly define the signals flow, *Step 2*: determine which signals or variables are known and unknown, *Step 3*: use the final value theorem (FVT) to calculate output steady-state of common primitive blocks. An alternative method to FVT is using the state-state model representation and set all derivative terms to zero. *Step 4*: calculate the unknown signals (and variables) in terms of the known quantities.

The developed DSL model of the MTDC in this chapter in Digsilent Power Factory requires two composite frames and at least seven model definitions. A detailed illustrative example of model initialisation is presented in this chapter for space constraints.

The active power controller presented in Fig. 19.11a is based on limited proportional integral function $\{K(1 + 1/sT)\}$ which is already built-in the global library for macros in PowerFactory using a DSL model where the state variable is x . The DSL code representation for the built-in model of this function in terms of dynamic equations is as follows:

```
b1 = K/T                                ! Constant definition
x. = yi                                 ! State variable equation
yo=lim(b1*limstate(x,Ymin/b1,Ymax/b1)+K*yi,Ymin,Ymax) ! yo Limit
limits(T)=(0,]                          ! safety check
```

The initialization of this common primitive block is very simple. In steady state, the derivative terms are zero and there is nothing to integrate, as a consequence, the

input $y_i = 0$. The output is already at its steady-state value $y_o = y_{o,ss}$, which is known or calculated from the steady-state values obtained from the load flow quantities.

The state variable, x , must be initialized to the steady-state output $y_{o,ss}$. Now, this initialization must be implemented in the block definition of the model of P-controller. The DSL implementation of active power controller has three inputs: DPf : changes on active power caused by frequency, $Pref$: reference of active power and P : actual active power obtained from circuit measurement. The active power in the MTDC system is obtained from the steady-state conditions obtained from the load flow calculation, as consequence the reference of this controller is initialized at such value, ($Pref = P$). The system frequency in steady state is equal to the nominal frequency, and there are no changes on the active power caused by frequency deviation; as a consequence, the changes on active power caused by frequency must be initialized at zero ($DPf = 0$). DSL code representation for the model initialization is as follows:

```
1: inc(DPf) = 0                                ! P changes caused by f
2: inc(Pref) = P                                ! P reference
3: inc(xP) = idref*Tp/Kp                        ! State variable, named xP
4:
5: ! Variable Definition
6: vardef(Kp) = 'p.u.'; 'P Controller Gain'
7: ardef(Tp) = 'sec'; 'P Controller Time constant'
8: vardef(Min_I) = 'p.u.'; 'Min d-q axis I'
9: vardef(Max_I) = 'p.u.'; 'Max d-q axis I'
```

The same procedure for initialization is followed for the Q -controller shown in Fig. 19.11b, and DSL code for the model initialization is as follows:

```
1: inc(Qref) = Q                                ! Q reference
2: inc(xQ) = idref*Tq/Kq                        ! State variable, named xQ
3: ! Variable Definition
4: vardef(Kq) = 'p.u.'; 'Q Controller Gain'
5: vardef(Tq) = 'sec'; 'Q Controller Time constant'
6: vardef(Min_I) = 'p.u.'; 'Min d-q axis I'
7: vardef(Max_I) = 'p.u.'; 'Max d-q axis I'
```

DSL code for the model initialization of the dq-current controller is shown in Fig. 19.12.

```
inc(xq)=Pmq*T/K                                ! State variable, named xq
inc(xd)=Pmd*T/K                                ! State variable, named xd
inc(idref)=id                                  ! d-axis current reference
inc(iqref)=iq                                  ! q-axis current reference
! Variable Definition
vardef(T)='s'; 'dq-axis Current Measurement Time Constant'
vardef(K)='p.u.'; 'dq-axis Gain'
vardef(Min_Pm)='p.u.'; 'Min. dq-axis modulation index'
vardef(Max_Pm)='p.u.'; 'Max. dq-axis modulation index'
vardef(Max)='p.u.'; 'Max. modulation index'
vardef(Max_I)='p.u.'; 'Max. module current'
```

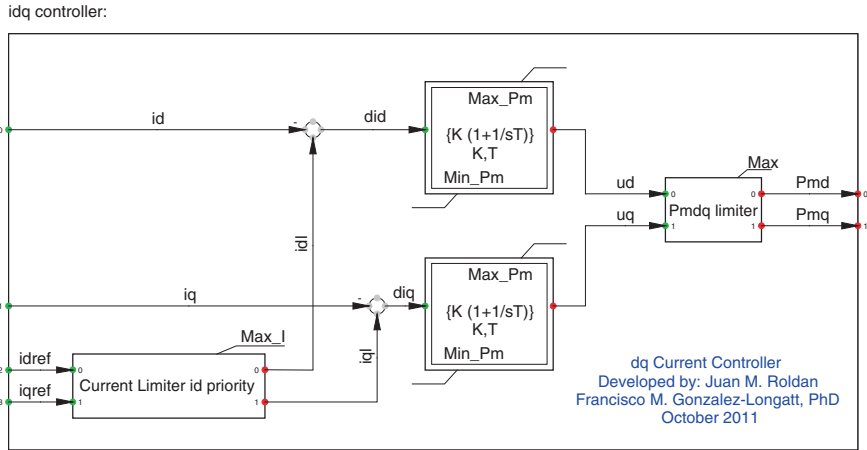


Fig. 19.12 Model definition for dq-current controllers

Finally, the DSL codes for the model initialization of the DC voltage controllers are presented in (Fig. 19.13):

```
! Voltage Margin Method: Model initialization
inc(o1)=idref-(Vdc/ud)*2*Idc/3           ! State variable, named o1
inc(xu)=o1*Tu/Ku                         ! State variable, named xu
inc(Vdc_ref)=Vdc                        ! Reference of DC voltage
inc(Pref)=P                             ! Reference of active power
inc(xP)=idref*Tp/Kp                     ! State variable, named xP
! Variable Definition
vardef(Kp)='p.u.';'P Controller Gain'
vardef(Tp)='p.u.';'P Controller Time Constant'
vardef(Ku)='p.u.';'Controller Gain u'
vardef(Tu)='sec';'Controller Time Constant u'
vardef(Min_I)='p.u.';'Min. q-d axis I'
vardef(Max_I)='p.u.';'Max. q-d axis I'
```

```
! Voltage Droop Method: Model initialization
inc(Pref)=P
inc(Vdcref)=Vdc
inc(xP)=idref*Tp/Kp
! Variable Definition
vardef(Kv)='p.u.';'P-V slope (negative)'
vardef(Kp)='p.u.';'P Controller Gain'
vardef(Tp)='p.u.';'P Controller Time Constant'
vardef(Min_I)='p.u.';'Min. q-d axis I'
vardef(Max_I)='p.u.';'Max. q-d axis I'
```

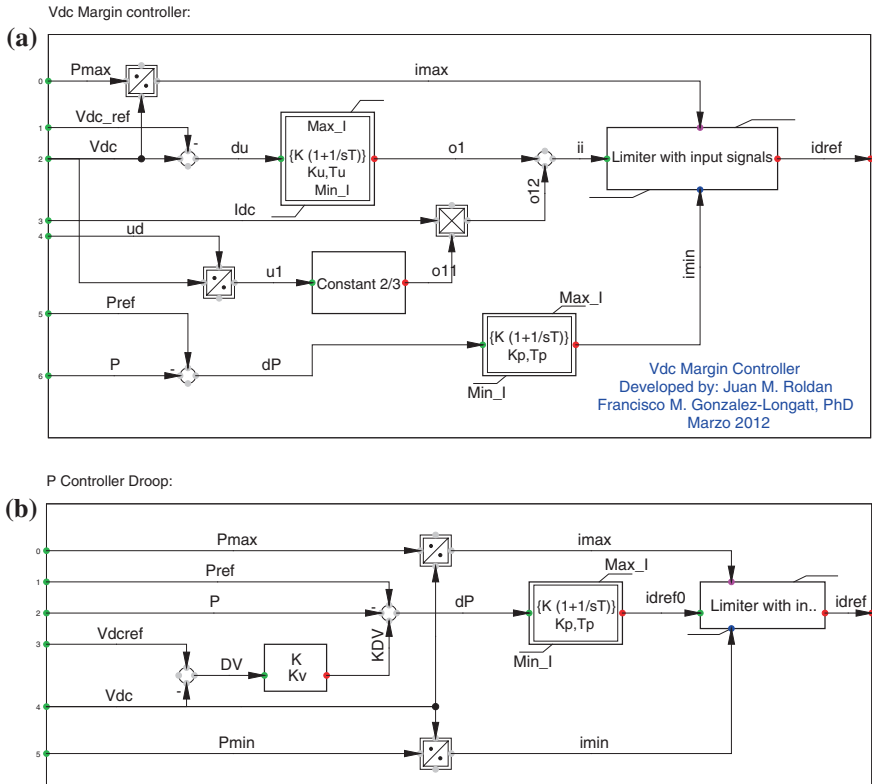


Fig. 19.13 Model definition of DC voltage controllers. **a** DC voltage margin controller method. **b** DC voltage droop controller method

19.11 Application Example

In this Section, the dynamic behaviour of the implementation of a MTDC system in DIgSILENT PowerFactory is demonstrated. Time-domain simulations on a AC/DC test system are used to analyse the performance of the controller following a converter outage when considering two DC voltage control strategies: VMM and voltage droop method. The test network used in this paper is the classical 5-bus test network which is taken from G.W. Stagg and A.H. El-Abiad [27] and 3-node VSC-MTDC network with is connected to the AC test system (Fig. 19.14).

The converter at bus 3 (VSC37) is chosen as a DC slack bus, thereby controlling the voltage on the DC network. This converter station is also used to control the voltage at bus 3 and it is the main target to evaluate two different DC voltage control strategies. The other converter stations (VSC26 and VSC58) are directly controlling their reactive power injections (constant Q -mode). Data on the converter station phase reactors and line resistances can be obtained from [14, 28].

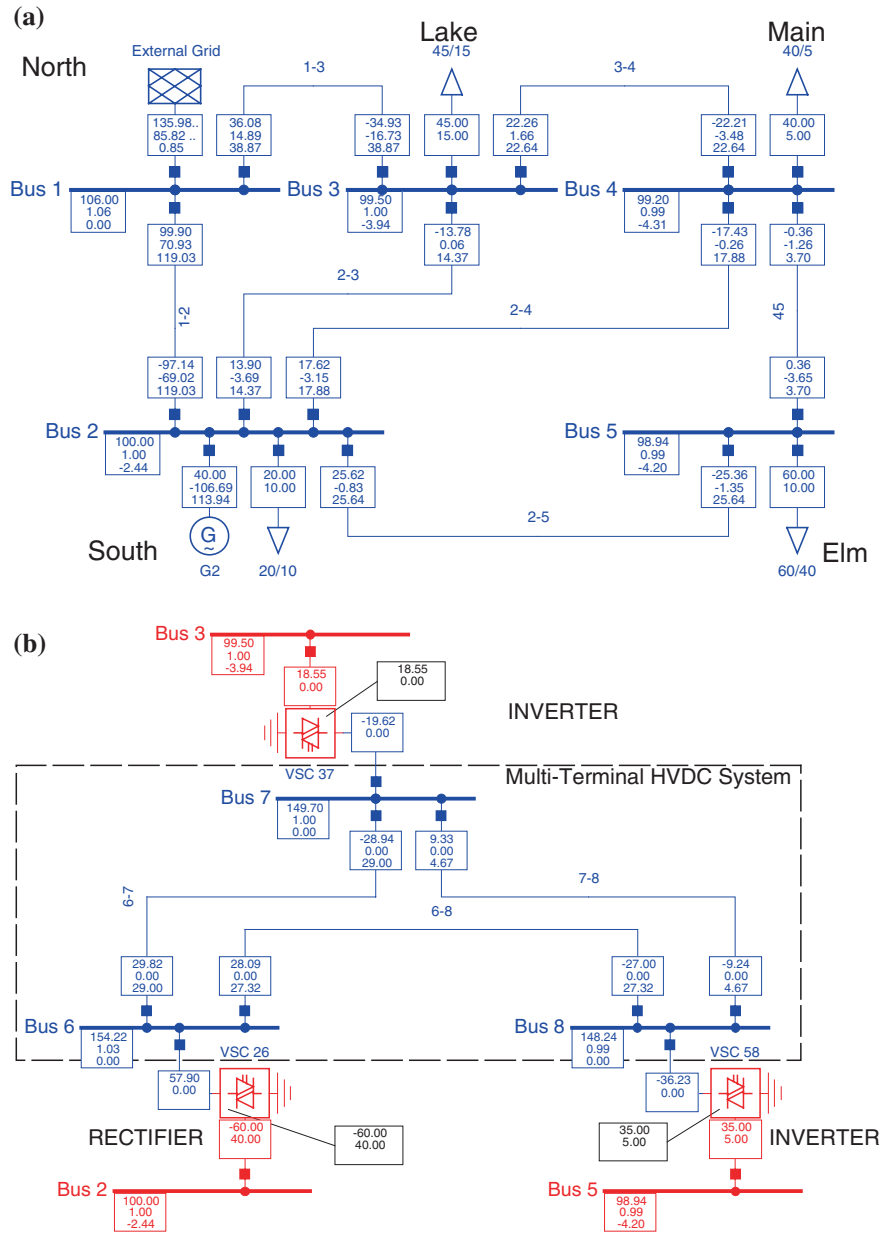


Fig. 19.14 AC/DC test system **a** AC test system: 5-node AC network. **b** DC test system: 3-node VSC-MTDC system

19.11.1 Case I: Sudden Load Increase

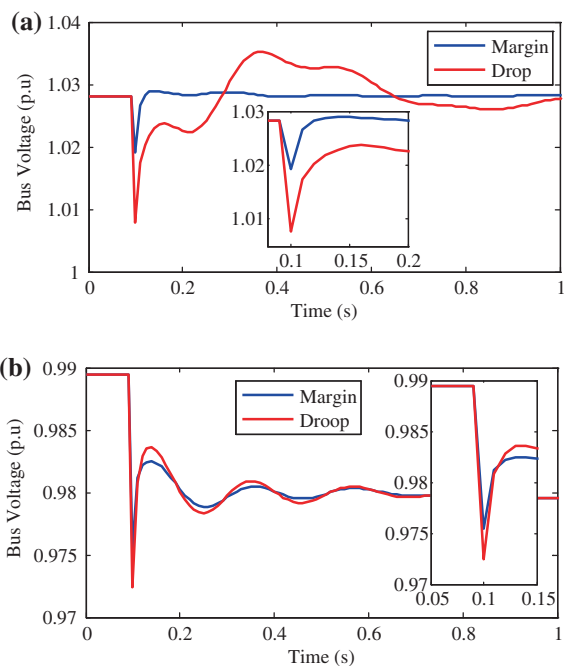
A sudden load increase of 40 MW on bus 3 is considered in this scenario. The dynamic behaviour of DC voltage at bus 6 and AC bus 5 is depicted in Fig. 19.15a and b.

The blue line shows the bus voltage's response having only one voltage controller operating, VMM. The red line represents the dynamic response when a voltage droop controller ($m_c = -0.1$) is operating on converter station VSC26. The transient over-voltages and under-voltages are reduced as expected using the droop control. The slopes of the voltage droop controller considered in this simulation are $1/m_c = -10.0$, -8.0 and -2.0 p.u. for converters VSC26, VSC37 and VSC58, respectively.

19.11.2 Case II: One Converter Outage

This simulation results are used to investigate the effect of a distributed voltage droop control on bus 2 (VSC26). Two different values of voltage droop slope (m_c) have been tested. Result shows the dynamic response of bus voltages is clearly influenced by the voltage droop characteristic. Figure 19.16a shows response of bus voltage at bus 6 considering a perturbation based on the outage of VSC58.

Fig. 19.15 Dynamic response of V_{ac} at Bus 5 (bottom) and U_{dc} at Bus 6 (top) after sudden load increase event considering two control strategies: voltage margin (blue) and droop (red). Case I. **a** Bus 6. **b** Bus 5



As shown, an incorrect selection slope value may causes transient responses with greater over-voltages on the DC bus (Droop B, green line). However, if voltage droop slope is appropriately selected, it can mainly assist the voltage at slack bus 3 and the system can handle transients caused by one converter station outage.

The AC voltage transient response is not significantly influenced by the voltage droop slope as shown in Fig. 19.16b. When the droop control is implemented in a larger DC network, the contribution of each converter to the DC voltage control can be adapted by modifying its droop characteristic. The values of the voltage droop slope used in this simulation are shown in Table 19.1.

Regarding the power load flow, the slack station-converter at bus 3 (VSC37) adapts its power, whereas the power injected by the converter at bus 2 (VSC26) remains unaltered. After a sudden converter-station disconnection, bus voltage at the remaining converter in operation exhibits a voltage droop which is previously defined by the Droop B characteristic. As shown by the results, the remaining converter powers are both lowered, dictated by their droop characteristics. Simulation shows the voltage margin control is capable to survive a converter outage just if this converter is operating on constant power mode.

Different values of voltage droop slope have been tested showing that the transient response is clearly influenced by the voltage droop characteristic. When two converters on the MTDC operate with DC voltage droop characteristic, it

Fig. 19.16 Dynamic response on case II. **a** Bus 6, DC voltage transient with margin and droop control strategies. **b** Bus 5, AC voltage transient with margin and droop control strategies

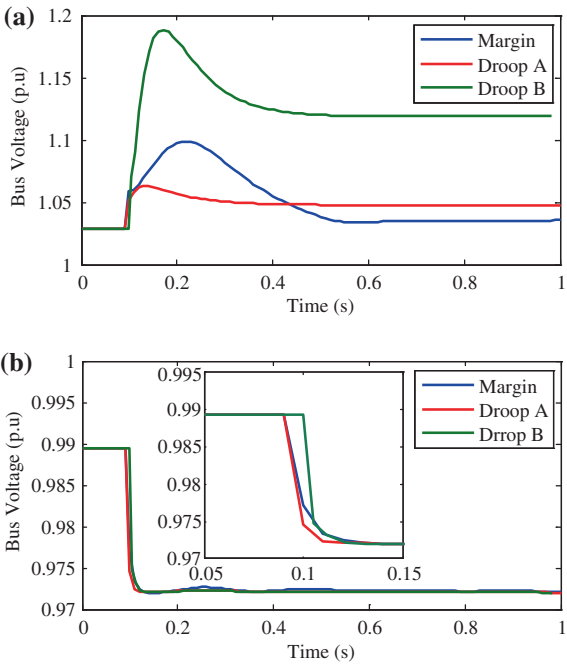


Table 19.1 Slope of voltage droop characteristic (p.u.)

Droop A ($-1/m_{c,i}$)		Droop B ($-1/m_{c,i}$)	
VSC23	VSC37	VSC23	VSC37
-10.0	-8.0	-2.0	-2.0

appears a “collaborative scheme” for the DC voltage support, sharing the task of controlling the DC voltage. Simulation results demonstrate the voltage margin control is capable to survive a converter outage just if this converter is operating on constant power mode.

References

1. Gonzalez-Longatt F (2014) Frequency Control and Inertial Response Schemes for the Future Power Networks. In: Hossain J, Mahmud A (eds) *Advances in technologies for generation, transmission and storage, green energy and technology series*, vol VIII. Springer, Singapore, p 363

2. UN (2011) The conference of the parties (15 Mar 2011). The Cancun agreement. FCCC/CP/2010/7/Add.1, Decision 1/CP.16, The Cancun agreements: outcome of the work of the Ad Hoc working group on long-term cooperative action under the convention. Decision (1/CP.16). Available from <http://unfccc.int/resource/docs/2010/cop16/eng/07a01.pdf>

3. EUREL Electrical Power Vision 2040 for Europe (Online). Available from www.eurel.org/home/.../EUREL-PV2040-Full_Version_Web.pdf

4. GOV. UK (2011) Policy reducing the UK’s greenhouse gas emissions by 80 % by 2050. Available from <https://www.gov.uk/government/policies/reducing-the-uk-s-greenhouse-gas-emissions-by-80-by-2050>

5. legislation. gov. UK (2008) Climate Change Act 2008. Available from <http://www.legislation.gov.uk/ukpga/2008/27/contents>

6. Winzer C (2012) Conceptualizing energy security, *Energy Policy*, 46:36–48

7. DECC (2013) Department of energy and climate change. Available from <https://www.gov.uk/government/organisations/department-of-energy-climate-change>

8. EWEA (2011) European wind energy association: policy/project—offshore wind. Available from <http://www.ewea.org/index.php?id=203>

9. Cornago AA (2011) Can the European supergrid become reality? Available from <http://www.publicserviceeurope.com/article/406/can-the-european-supergrid-become-reality>

10. Chauhan RK, Rajpurohit BS, Singh SN, Gonzalez-Longatt FM (2014) DC grid interconnection for conversion losses and cost optimization. In: Mahmud A, Hossain J (eds) *Renewable energy integration*. Springer, Singapore, pp 327–345

11. FOSG (2011) Friends of the Supergrid. Available from <http://www.friendsofthesupergrid.eu/>

12. FOSG (2010) Friends of Supergrind. Position paper on the EC Communication for a European Infrastructure Package. Available from <http://www.friendsofthesupergrid.eu/documentation.aspx>

13. Haileselassie TM (2008) Control of multi-terminal VSC-HVDC systems, master of science in energy and environment. Department of Electrical Power Engineering, Norwegian University of Science and Technology, Trondheim

14. Gonzalez-Longatt F, Roldan J (2012) Effects of DC voltage control strategies of voltage response on multi-terminal HVDC following a disturbance. In: 47th International Universities Power Engineering Conference (UPEC 2012), pp 1–6

15. Gonzalez-Longatt F, Roldan J, Charalambous CA (2012) Power flow solution on multi-terminal HVDC systems: supergrid case. Presented at the international conference on renewable energies and power quality (ICREPQ'12), Santiago de Compostela
16. Vrana TK, Torres-Olguin RE, Liu B, Haileselassie TM (2010) The North Sea super grid—a technical perspective. In: ACDC. 9th IET international conference on AC and DC power transmission, 2010, pp 1–5
17. Van Hertem D, Ghandhari M, Delimar M (2010) Technical limitations towards a Supergrid: a European perspective. In: 2010 IEEE International, Energy Conference and Exhibition (EnergyCon), pp 302–309
18. Van Hertem D, Ghandhari M (2010) Multi-terminal VSC HVDC for the European supergrid: Obstacles. *Renew Sustain Energy Rev* 14:3156–3163
19. Zhu J, Booth C (2010) Future multi-terminal HVDC transmission systems using voltage source converters. Presented at the 45th international universities power engineering conference (UPEC), 2010 Cardiff, Wales
20. Nakajima T, Irokawa S (1999) A control system for HVDC transmission by voltage sourced converters. In: Power engineering society summer meeting, 1999. IEEE, vol. 2, pp 1113–1119
21. Seki N (2000) Field testing of 53 MVA three-terminal DC link between power system using GTO converters. In: IEEE power engineering society winter meeting 2000, pp 2504–2508
22. Wang J, Li X, Qiu X (2005) Power system research on distributed generation penetration. *Autom Electr Power Syst* 29:97–99
23. Cole S (2010) Steady-State and Dynamic Modelling of VSC HVDC Systems for Power Systems Simulation, Doctor in de Ingenieurswetenschappen Doctor in de ingenieurswetenschappen, Faculteit Ingenieurswetenschappen, Departement Elektrotechniek. Universiteit Leuven, Belgium
24. Wang W, Barnes M (2014) Power flow algorithms for multi-terminal VSC-HVDC with droop control. *IEEE Trans Power Syst* 29(4):1721–7930. doi:10.1109/TPWRS.2013.2294198
25. Haileselassie TM, Uhlen K (2010) Primary frequency control of remote grids connected by multi-terminal HVDC. In: General Meeting, 2010 IEEE on power and energy society, pp 1–6
26. Hendriks RL, Paap GC, Kling WL (2007) Control of a multiterminal VSC transmission scheme for connecting offshore wind farms. In: European Wind Energy Conference, Milan, Italy
27. Stagg GW, El-Abiad AH (1968) Computer methods in power system analysis. McGraw-Hill, New York
28. Gonzalez-Longatt F, Roldan J, Burgos-Payán M, Terzija V (2012) Implications of the DC Voltage Control Strategy on the Dynamic Behavior of Multi-terminal HVDC following a Converter Outage, presented at the CIGRE-UK and European T&D network solutions to the challenge of increasing levels of renewable generation. Newcastle-under-Lyme College, Staffordshire, United Kingdom

Chapter 20

Estimation of Equivalent Model for Cluster of Induction Generator Based on PMU Measurements

Francisco M. Gonzalez-Longatt, José Luis Rueda,
C.A. Charalambous and P. De Oliveira

Abstract The induction generator (IG) is widely used in many applications due to its simplicity and ease of operation. Typical applications involve the following: split-shaft micro-turbines (SSMT), mini-hydro (MH), and fixed-speed wind turbines (FSWT). The accurate knowledge of the machine parameters is especially important in order to establish the performance of the IG as well as to directly affect its operational and control characteristics. The problem of IG parameter estimation results specially complicated to solve when a cluster of IG is interconnected to create of virtual power plant (VPP). Then, it is desirable to have an effective method to estimate the parameters of an equivalent model for a cluster of IG (which does not require detailed definition of the power plant structure and parameters) by using novel digital measurement equipment such as phasor measurement units (PMU) in transmission and distribution networks. This chapter presents a method for the estimation of an equivalent model (named as EqMCIG App) for a cluster of IG, based on the response to a system frequency disturbance. The performance and

Electronic supplementary material The online version of this chapter (doi:10.1007/978-3-319-12958-7_20) contains supplementary material, which is available to authorized users.

F.M. Gonzalez-Longatt (✉)
School of Electronic, Electrical and Systems Engineering, Loughborough University,
Loughborough LE11 3TU, UK
e-mail: fglongatt@fglongatt.org

J.L. Rueda
Department of Electrical Sustainable Energy, Delft University of Technology,
Mekelweg 4, 2628 CD Delft, The Netherlands
e-mail: J.L.RuedaTorres@tudelft.nl

C.A. Charalambous
Department of Electrical and Computer Engineering, University of Cyprus,
75 Kallipoleos Avenue, 1678 Nicosia, Cyprus
e-mail: cchara@ucy.ac.cy

P. De Oliveira
The Energy Institute of Simon Bolivar University, Caracas 2122, Venezuela
e-mail: pdeoliveira@usb.ve

robustness of the method are evaluated using two different test systems where the EqMCIG is identified using the variable metric method (VMM). Numerical results demonstrate the viewpoint and effectiveness of the proposed methodology. This chapter have three main contributions: (i) Performing “parameter estimation” using *ComIdent* command (ii) The use of DSL model on model parameter identification (composite Frame, block definition—*BlkDef*) (iii) Use of a “Measurement File” object (*ElmFile*) as inputs variables.

Keywords Fixed-speed wind turbines • Generator modeling • Induction machine • Parameter estimation

20.1 Introduction

Future power systems will face several challenges; one of them is the anticipated massive integration of highly variable power generation coming from renewable energy resources and supported by so many different technologies pertaining to energy storage that will be integrated to the ac grid using power converters [1, 2]. The highly fluctuating generation systems will be primarily based on a very wide variety of technologies and designs. This change in the generation mix of a system will introduce new challenges in many fields, e.g. frequency control. Several renewable technologies provide little, or no, inertia to the system [1]. Thus, they do not contribute to the primary frequency response of a system. The increased use of a generation technology with this characteristic will cause a considerable reduction in the ability of an operator to ensure the security of system frequency after a disturbance. However, there are several power generation technologies based on single-cage induction generators (SCIG) and they provide natural damping of power system oscillations during a system frequency disturbance. Control of the output power provided by a cluster of SCIG during a disturbance could effectively modify the system inertia providing a natural frequency response [1]. These two features of a cluster of SCIG offer the opportunity to use them as a tool during frequency control.

An induction generator (IG) is, in principle, an induction motor with torque applied to the shaft, although there may be some modifications made to the machine design to optimize its performance as a generator [3]. This type of machine is widely used in many applications due to its simple construction and ease of operation. The suitability of using IG for power systems application and renewable energy has been reported in several publications [4–7]. Typical applications involve the following: split-shaft micro-turbines (SSMT) [8], mini-hydro (MH) [9], and fixed-speed wind turbines (FSWT) [10].

The accurate knowledge of the machine parameters is especially important in order to establish the performance of an IG and directly affect the operational and control characteristics of the IG [3]. The problem of parameters estimation for

induction machines (IM) has receiving great attention from various research workers for years [11, 12]. These schemes use the measured response of the IM as a change in voltage to estimate the induction machine parameters. This problem has been resolved using several optimization techniques such as: genetic algorithm (GA) [13–15], a local search algorithm (LSA), a simulated annealing (SA) approach, an evolution strategy (ES), particle swarm optimization (PSO) [16, 17], and improved particle swarm optimization (IPSO) [3], Kalman filter [18]. Most of these techniques are applied to data gathered during the start-up of the machine [14, 19–21] and comparison between the results of the estimated model and data gathered by applying mechanical tests to the machine is used for others [15, 16].

The problem of IG parameters estimation can be particularly complicated to calculate when a cluster of IGs is interconnected to create a *virtual power plant* (VPP). The VPP concept enables the aggregation of distributed generation, controllable loads, and storage devices supported on information and communication system [22]. It is desirable to have an effective method to estimate the parameters of an equivalent model for a cluster of IG which does not require detailed definition of the power plant structure and parameters and using the novel digital measurement equipment such as phasor measurement units (PMU) in transmission and distribution networks.

This chapter presents a method for the estimation of an equivalent model for a cluster of IG based on online response to a system frequency disturbance. All of the parameter estimation schemes introduced above deal exclusively with the response of an induction machine to a change in voltage, but the method developed in this chapter uses the measured response of an IG to system frequency response. The response of the IG is viewed in terms of the active and reactive power flow associated with the IG. Parameter estimation based on the IG response to a change in frequency is possible as both the active power generated and reactive power consumed are dependent on the frequency of the system that the IG is connected to. The method presented in this chapter represents an improvement in the performance of the parameter estimation, and it represents an important benefit to online applications dependent on accurate representations of power system components, e.g., intelligent controlled islanding or stability assessments. This chapter presents a method for the estimation of an equivalent model (named as *EqMCIG App*) for a cluster of IGs, based on the response to a system frequency disturbance. The proposed methods are developed using DIgSILENT PowerFactory, and the performance and robustness of the method are evaluated using two different tests system where the EqMCIG is identified using the variable metric method (VMM). Numerical results demonstrate the viewpoint and effectiveness of the proposed methodology. This chapter has three main contributions: (i) Performing “parameter estimation” using *ComIdent* command, (ii) The use of DSL model on model parameter identification (composite Frame, block definition—*BlkDef*), (iii) Use of a “Measurement File” object (ElmFile) as inputs variables. The paper is organized as follows: Section 20.2 describes the equivalent model for the cluster of IG, Section 20.3 presents the method for the equivalent model estimation, and Section 20.4 shows the results of simulations that confirm the validity of the proposed method.

20.2 Equivalent Model for a Cluster of Induction Generators

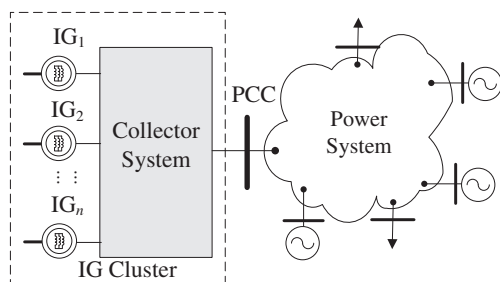
Several IGs are connected to the electrical power network through a collector system. This system starts from the step-up transformer connected to each IG and includes power cables, power factor compensation devices, and the main substation. Finally, the cluster of IG is connected to the power network at point of common coupling (PCC).

Figure 20.1 shows a schematic diagram of IG cluster. Representative examples of a cluster of IG concept include VPP based on SSMT and wind farm based on FSWT.

In order to develop an equivalent model for a cluster of induction valid for system disturbances and suitable for online response, the following consideration has been taken [3, 23]:

- *Dynamic of prime mover of each IG is neglected.* The mechanical power (P_m) on the shaft of each machine is considered constant and equal to the value previous the system disturbance. This assumption is valid for very small timescales (seconds) in most of the IG application. For example, the changes in the wind speed are negligible; during periods of several seconds as consequence, rotor of wind turbine works with a constant wind speed providing a constant torque in FSWT.
- *The electro-mechanical model of the IG is considered.* The electrical model of the IG is represented by a fourth-order state-space model and the mechanical part by a second-order system. The electrical part of the model is represented using Park transformation considering a rotating reference frame. Using this approach, all the effects of system frequency changes is included.
- *Equivalent series impedance is used to model the collector system.* Considering the changes in the system frequency changes are small, the collector system can be considered as linear system and modeled such a RLC electrical network. Topology of the collector system can be complex, but equivalent Thevenin impedance can be used to modeling the collector system in per unit values.

Fig. 20.1 Schematic diagram of a grid-connected cluster of IG



20.3 Equivalent Model Estimation Method

A cost-effective solution to improve grid planning, operation, maintenance, and energy trading in large power systems is the use of wide-area monitoring, protection, and control (WAMPAC) [24, 25]. It enables the development of several applications such as real-time network model parameters estimation producing more accurate results that can be used for other real-time applications. In this paper, an equivalent model for a cluster of IG, called EqMCIG App, based on measurement data, is presented. The general concept of EqMCIG App is depicted in Fig. 20.2.

A phasor measurement unit (PMU_i) is installed at the point of common connection (PCC) of the cluster of IG (Bus_i); each PMU provides a data set (Φ_i) which represents an accurate time-stamped measurement data suitable for observing power system dynamics. Measurements of voltages (V_i) and currents (I_i), including frequency (f_i), are included in the data set Φ_i , which is transmitted using a communication media to the data concentrator (DC). When a (credible) system frequency disturbance occurs, DC sends the measurement raw data to the EqMCIG application. Data set is preprocessed, and voltage and current measurements are used to compute real and reactive power at common connection, $\theta_i = [P_i \ Q_i]$.

The procedure to estimate the parameter of the EqMCIG based on measurement data is described as follows:

- Step 1: Obtain a set of input–output data, ψ_i and θ_i , derived from a set of measurement Φ_i .
- Step 2: Estimate the model parameters \mathbf{x} using a suitable method.
- Step 3: Evaluate the derived EqMCIG using the estimated set of parameter \mathbf{x} in order to obtain the estimated output $\theta_i = [P_i \ Q_i]$.
- Step 4: If any quality solution (previously defined, e.g., $\|\theta_i - \hat{\theta}_i\|^2 < \varepsilon$ criterion) is not met, go to Step 3.

Fig. 20.2 Schematic representation of EqMCIG application

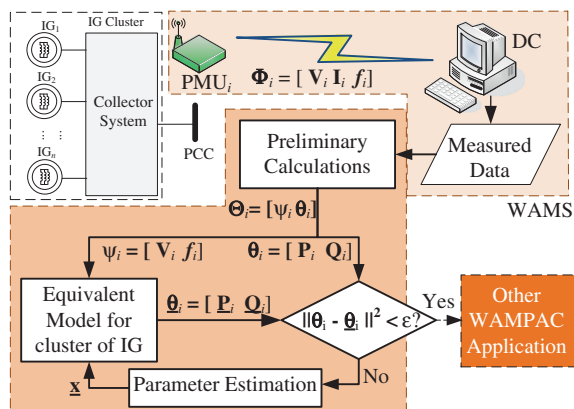
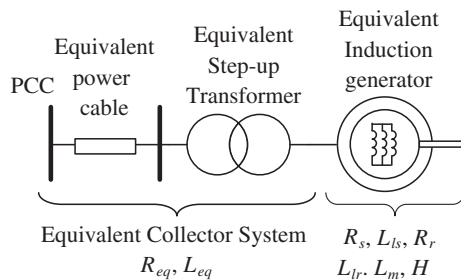


Fig. 20.3 Simplest representation of a cluster of IG



The EqMCIG is defined by a set of electrical and mechanical parameters included in the vector \mathbf{x} . The model proposed in this paper has a flexible structure and allows several levels of details.

The most simple model is defined by eight parameters: $\mathbf{x} = [R_{eq} \ L_{eq} \ R_s \ L_{ls} \ R_r \ L_{lr} \ L_m \ H]$, where R_{eq} and L_{eq} are the electrical parameters of an equivalent series element that represent the Thevenin equivalent model for the collector system, and the remaining parameters correspond to the equivalent IG model. The EqMCIG in this simple case is represented in Fig. 20.3.

More detailed models consider several other effects such as power factor capacitors, cable capacitances, and active power losses associated with the magnetizing current in the transformers. Under these situations, the method for the EqMCIG is in essence the same; the only difference is the number of parameter in the vector \mathbf{x} and its impact on the computer time required to reach the solution.

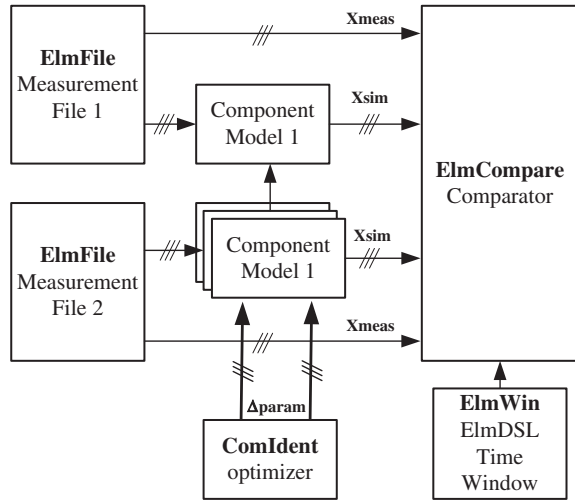
The vector \mathbf{x} is represented in per unit quantities in order to take advantages of several properties of this system: All quantities are referred to one side of the transformer (PCC), the results are independent of the tri-phases connections used in transformers, and the per unit quantities vary in narrow band as consequence errors are easily detected. A convenient selection of the base values is needed, and in this method, the active power immediately before the system frequency disturbance and the rated voltage at PCC are used as base values.

An important part of the EqMCIG application is a suitable parameter estimation method. There are a lot of parameter identification methods; however, all can be included in three main groups: artificial intelligence (AI) methods [3, 26], nonlinear least squares (NLS) methods [27], and hybrid methods [28]. The EqMCIG Application is suitable for all three groups of methods; an illustrative example is presented in the next sections to demonstrate the performance of the proposed application.

20.4 Model Parameter Identification in DIgSILENT Power Factory

PowerFactory includes a useful feature that allows model parameter identification or parameter estimation for power system elements for which certain measurements. The goal of model parameter identification is to approximate the parameter

Fig. 20.4 The identification principle. More details can be found on the user manual of power factory



of a mathematical model of a dynamic system based on experimental data. The model should be compact and adequate in respect to the purpose of its use. The parameter identification command (ComIdent) is a high-performance nonlinear optimization tool, which is capable of a multi-parameter identification for one or more models, given a set of measured input and output signals. The whole process of model parameter identification is depicted on Fig. 20.4.

PowerFactory performs the parameter identification process by minimizing objective functions. The core of the parameter identification command (ComIdent) is the use of the objective functions; PowerFactory allows the use of a DSL to perform this task. The objective functions are calculated by *ElmCompare* objects from the difference between measured responses and calculated responses of one or more power system elements.

The composite frame is used as a graphic diagram showing interconnections between the main slots used for the model parameter identification. The block diagram implemented in DSL for EqMCIG application is shown on Fig. 20.5, and it includes four slots: comparison slot: *ElmCompare*, two slots for measurements taken from the parametric dynamic model: power (StaPqmea: Pacsim and Qacsim) and voltage (StaVmea: Vacsim), and Measurement file slot: The measurement file object (ElmFile) is used for reading data taken during a tests and stored in a file (Vmeas, Pmeas, Qmeas).

This DSL implementation of the EqMCIG application is time-domain-based simulations using RMS quantities. Voltage, active power, and reactive power measurement on the PCC from a detailed situation are compared with the measured response of the same variables from the parametric dynamic model.

The parametric dynamic model implemented in PowerFactory based on the simplest representation of a cluster of IG is shown in Fig. 20.3. A variable voltage source is used to feed the measured signals into the parametric dynamic model; the composite frame of that variable voltage source is depicted in Fig. 20.6.

Fig. 20.5 Composite frame used on the model parameter identification

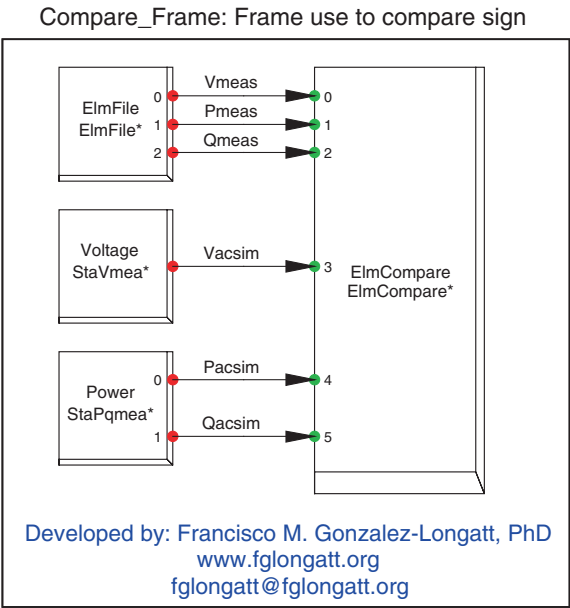
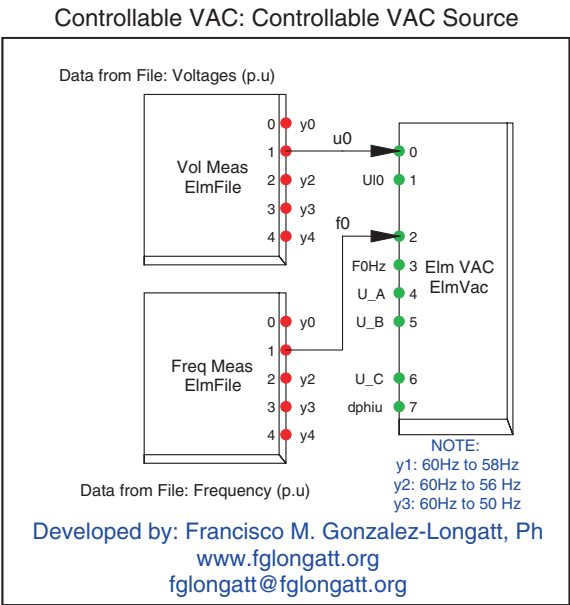


Fig. 20.6 Composite frame of variable AC voltage source



Two measurement file slots are used for reading the data measured on the full system during a test. The Measurement File (ElmFile) is used for reading data from a file during calculation. The measurement file is a simple ASCII file (.txt) with a

column for every variable. Two files are used in this implementation: RMS values of voltage measurements (VolMeas) and electrical frequency (FreMeas). Those signals are applied to the AC voltage source (ElmVac) to create the controllable voltage source ($u0$: amplitude of voltage in per unit and $f0$ electrical frequency in per unit).

20.5 Demonstrative Example

This section presents a demonstrative example which is used to demonstrate the effectiveness and robustness of the method for EqMCIG presented in this chapter. A cluster of IG which consists of 9×500 kW induction machines is considered for simulation purposes, the complete set of parameters for these machines are summarized in Table 20.1.

Two different topologies are considered for the collector system and these are shown in Fig. 20.7. A PMU is installed in the PCC at 0.69 kV providing the set of measurements Φ : voltages, currents, and frequency.

The set of measurements that the raw data are transmitted from the DC is installed to EqMCIG application. The measurements of voltage (V) and frequency (f) used for test proposed in this paper are shown in Fig. 20.8.

The technique used for the parameter identification is not the most important part of the method for estimating EqMCIG, for this reason any method can be used. In this paper, the power system software DlgSILENT® PowerFactory™ has used both the time-domain simulations for the cluster of IG to the model parameter identification. Measured data set is preprocessed, and voltages and currents measurements are used to compute $\theta_i = [P_i \ Q_i]$. The parameter identification process is performed by minimizing an objective function which is calculated from the difference between the measured $\theta = [P_e \ Q_e]$ and simulated responses $\underline{\theta} = [\underline{P}_e \ \underline{Q}_e]$ as follows:

$$\|\theta - \underline{\theta}\|^2 = \|\mathbf{P}_e - \underline{\mathbf{P}}_e\|^2 + \|\mathbf{Q}_e - \underline{\mathbf{Q}}_e\|^2 = fp \quad (20.1)$$

DlgSILENT® PowerFactory™ uses a powerful iterative multi-variable optimization procedure for finding a local minimum. It is based on quasi-Newton method, also known as VVM [29, 30]. It is one of the identification tools for nonlinear systems. The VVM finds the stationary point of a function, where the gradient is 0. It assumes that the function can be locally approximated as a quadratic function in

Table 20.1 Parameter valued for the IG

Parameter	Variable	Value
Stator resistance	R_s (p.u)	0.00599
Rotor resistance	R_r (p.u)	0.01691
Stator inductance	L_s (p.u)	0.08213
Rotor inductance	L_r (p.u)	0.10723
Magnetizing reactance	L_m (p.u)	2.55619
Inertia	J (kgm ²)	330.00

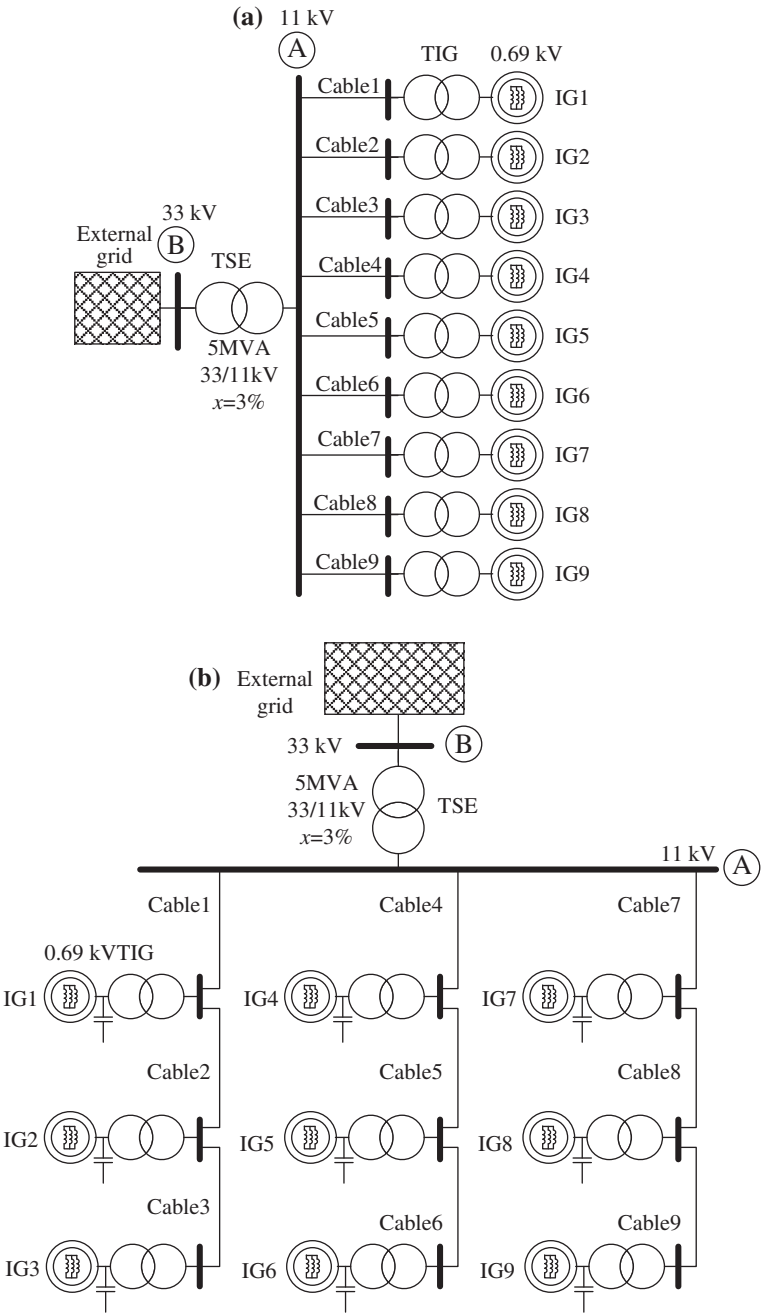


Fig. 20.7 Test Systems. **a** Test System I. **b** Test System II

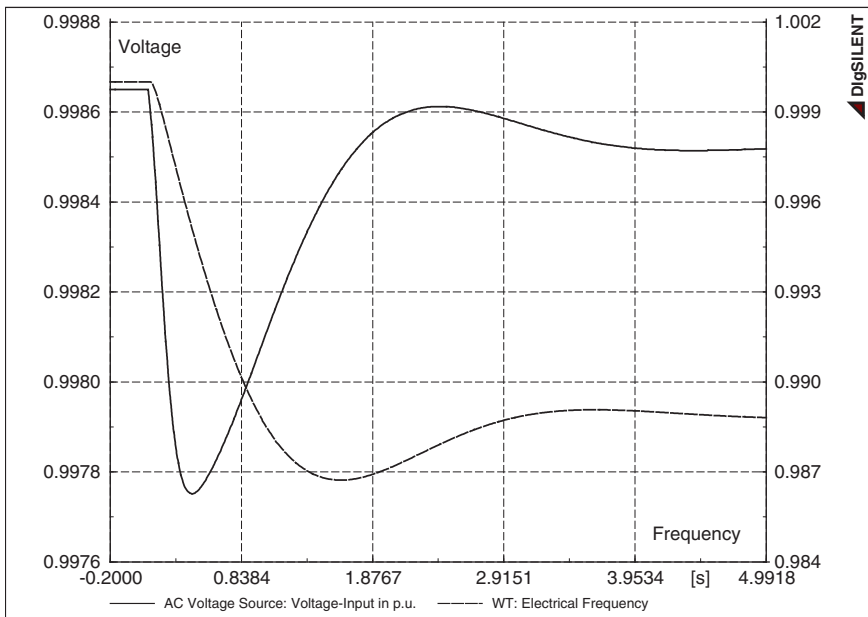


Fig. 20.8 Measurements of frequency and voltage at PCC for the cluster of induction generator considered

the region around the optimum, and it uses the first and second derivatives (gradient and Hessian) to find the stationary point [31]. The model parameter identification process is configured, for testing purposes in this paper, considering a tolerance for the error $\varepsilon < 10^{-4}$.

20.5.1 Test System I

Test System I consists of a cluster of 9×500 kW inductor generators as presented in Fig. 20.7a, and each machine has a step-up transformer (0.75 MVA, 0.69/11 kV, $x_{tr} = 2\%$) and it is directly connected to the 11 kV by a power cable ($r_{line} = 0.253 \Omega/\text{km}$, $x_{line} = 0.1036726 \Omega/\text{km}$, $b_{line} = 87.96459 \mu\text{S}/\text{km}$, long = 0.1 km).

The active power produced by cluster of inductor generators is 2.52 MW before the system disturbance (shown in Fig. 20.8), and each inductor generators is operating at 56 % of rated power.

The result of the parameters identification for the EqMCIG considered on the Test System I is shown in Fig. 20.9.

The data sets of active and reactive measurement and the simulated response for the set of parameters obtained from the model parameter identification are shown in Fig. 20.10.

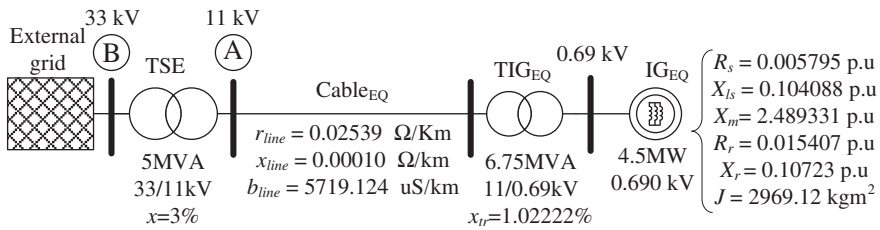


Fig. 20.9 Results of the model parameter identification for Test System I

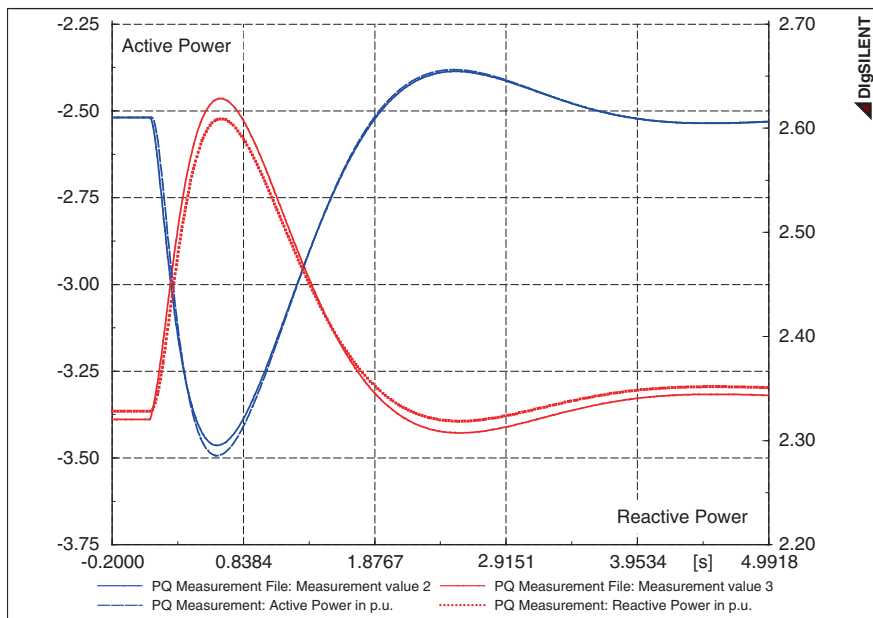


Fig. 20.10 Results of the EqMCIG estimation for the Test System I: measured versus simulated response

The error between the simulated response of the EqMCIG estimated, and the measurement result is below 2 % both for active and reactive power. This is a very good indicator of the robustness of the proposed method.

20.5.2 Test System II

Test System II consists of a cluster of $9 \times 500 \text{ kW}$ inductor generators as presented in Fig. 20.7b, and each machine has a capacitor for reactive power compensation at terminals (p_{capn}) and a step-up transformer (0.75 MVA, 0.69/11 kV, $x_{tr} = 2 \%$).

Topology used in the collector system is shown in Fig. 20.4b, and it is based on 11-kV power cables ($r_{\text{line}} = 0.253 \text{ } \Omega/\text{km}$, $x_{\text{line}} = 0.1036726 \text{ } \Omega/\text{km}$, $b_{\text{line}} = 87.96459 \text{ } \mu\text{S}/\text{km}$). Cables have different lengths in the collector system: Cable 1, 4, 7: 100 m, Cable 2, 5, 8: 200 m, and Cable 3, 6, 9: 300 m. A system disturbance as shown in Fig. 20.5 is applied to the Test System II when the cluster of inductor generators is exporting 2.52 MW to the external grid. The general EqMCIG and the parameters to be estimated by the EqMCIG application are depicted in Fig. 20.11.

The reactive power compensation is an important factor that influences the performance of the cluster of inductor generator. It is especially true during system disturbances involving changes in the voltage and frequency.

Several scenarios of reactive power compensation levels have been considered for the Test System I: 0, 25, 50, 75, and 100 % of the full-load reactive power consumption. Table 20.2 shows the results of the model parameters identification based on the EqMCIG application for the different scenarios considered.

The only difference between the scenarios considered is the reactive power compensation on terminals of each inductor generator. It is expected that the

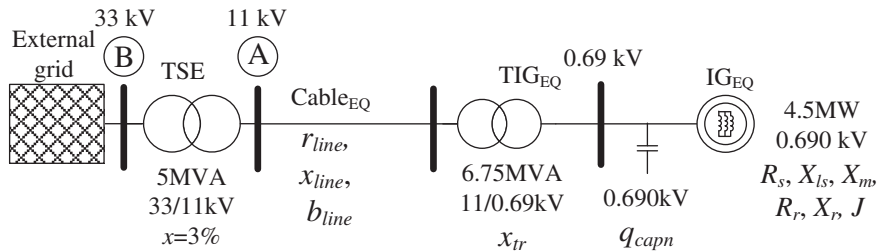


Fig. 20.11 General EqMICG for Test System II showing the parameters to be estimated

Table 20.2 Results of the parameter identification on Test System II for several reactive compensation levels

Parameter	Reactive power compensation				
	0 %	25 %	50 %	75 %	100 %
R_{line} (p.u)	0.03156	0.02531	0.02350	0.02331	0.02531
X_{line} (p.u)	0.10370	0.10370	0.10366	0.10365	0.10368
B_{line} (p.u)	879.118	879.118	879.118	879.118	879.118
q_{capn} (p.u)	–	0.0191	1.3950	1.4189	2.7897
x_{tr} (p.u)	0.019825	0.013875	0.019999	0.010602	0.010603
R_s (p.u)	0.0053	0.0052	0.0063	0.0052	0.0050
X_s (p.u)	0.0946	0.0914	0.0859	0.0913	0.1068
R_r (p.u)	0.0140	0.0139	0.0136	0.0140	0.0138
X_r (p.u)	0.1201	0.1138	0.1222	0.1137	0.0914
X_m (p.u)	2.4886	25597	2.5392	3.5830	2.5054
J (kgm ²)	1,278.525	1,281.388	1,273.587	1,278.007	1,286.244
fp (p.u)	0.000614	0.000413	0.000596	0.000457	0.000488

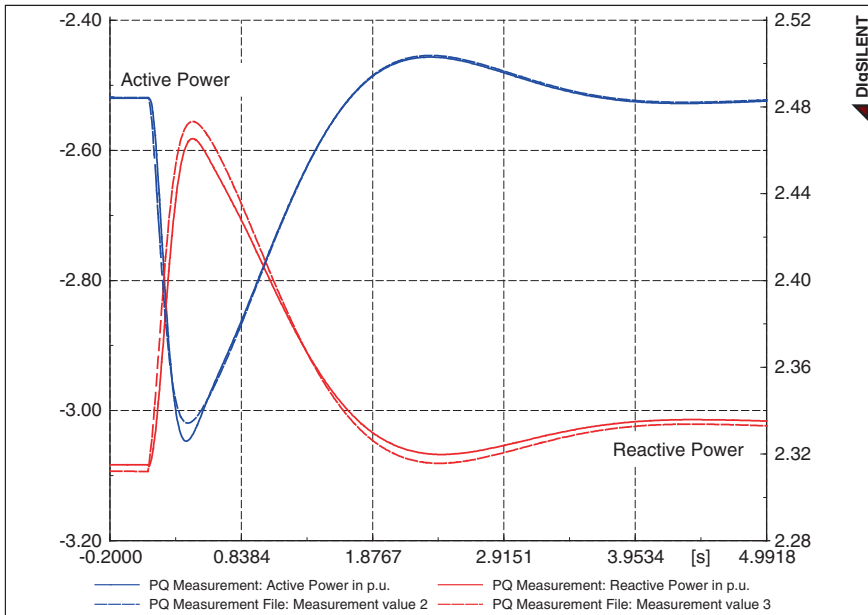


Fig. 20.12 Results of the EqMCIG estimation for the Test System II: measured versus simulated response. Scenario I: 0 % reactive power compensation

parameters of the equivalent step-up transformer, cable, and induction machine would be the same for all parameters identifications. However, results in Table 20.1 show some differences for a parameter between the scenarios. These differences are due to the changes in the voltage profile in the collector system and changes in steady-state operation point of every induction machine; it is caused by reactive power compensation. It is noticed that the maximum differences is below 10 % and it is evident on the reactance of the step-up transformer.

The last row of Table 20.2 shows the parameter fp that represents the value of the objective function reached during the parameter identification process. The maximum value of fp is reached in the case where reactive compensation is not considered in the Test System II.

Figures 20.12, 20.13, and 20.14 show plots of data sets of active and reactive measurements and the simulated responses for the set of parameters obtained from method proposed in this paper considering several reactive power compensation levels.

The changes in the reactive power compensation are evident on each plotting, whereas changes in the active power are insignificant. Difference between the measurements and the simulated response for active power is less than reactive power; this difference is almost the same for all reactive power compensation. This situation is shown in Fig. 20.15.

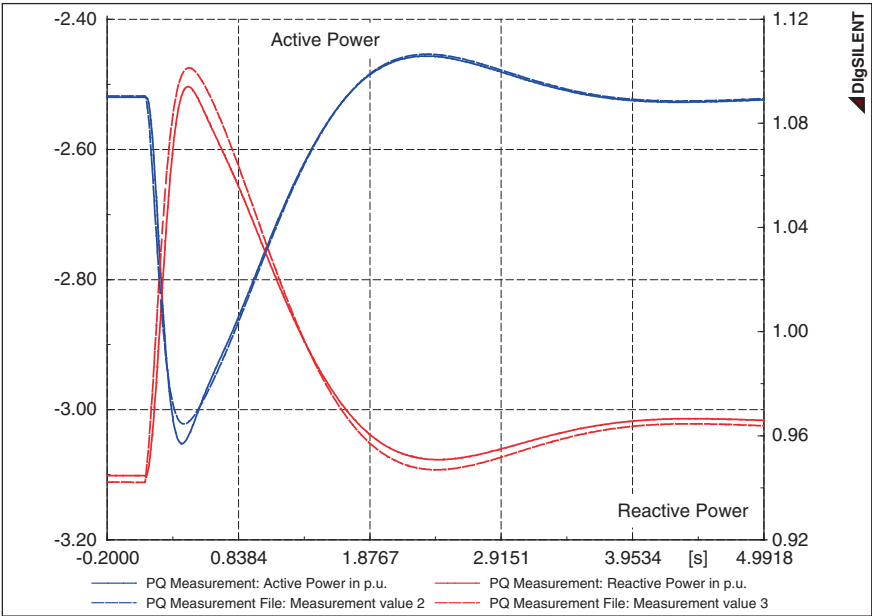


Fig. 20.13 Results of the EqMCIG estimation for the Test System II: measured versus simulated response. Scenario III: 50 % reactive power compensation

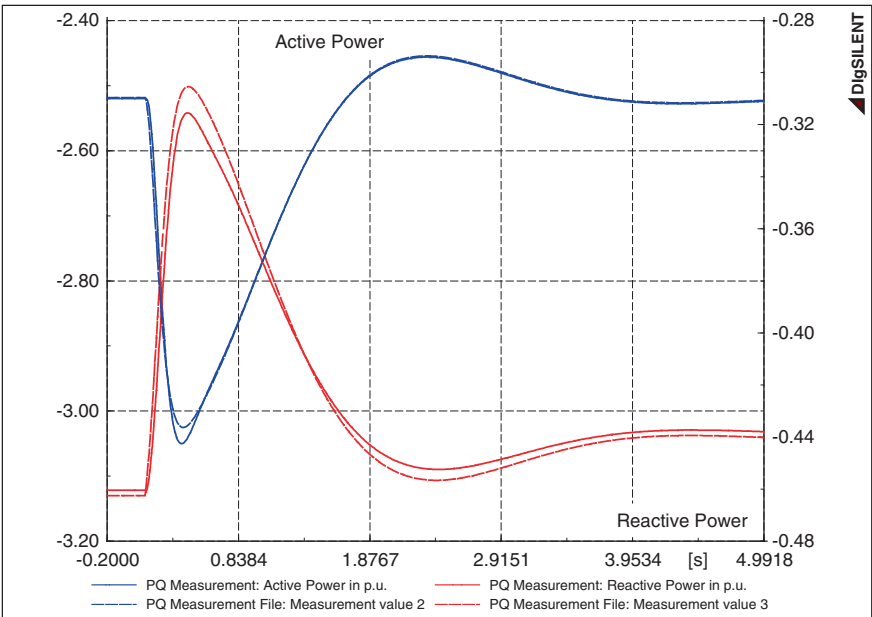


Fig. 20.14 Results of the EqMCIG estimation for the Test System II: measured versus simulated response. Scenario III: 100 % reactive power compensation

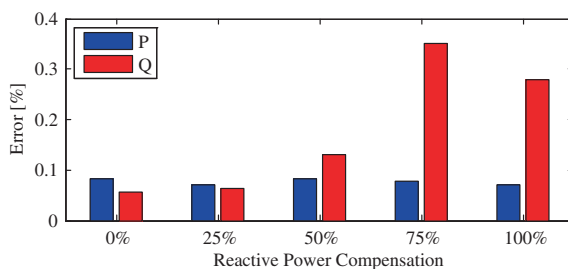


Fig. 20.15 Errors in active a reactive power simulated response using the EqMCIG estimation for the Test System II compared with the measured response

References

1. Gonzalez-Longatt F (2014) Frequency control and inertial response schemes for the future power networks. In: Hossain J, Mahmud A (eds) Large scale renewable power generation. Springer, Singapore, pp 193–231
2. Chauhan RK, Rajpurohit BS, Singh SN, Gonzalez-Longatt FM (2014) DC grid interconnection for conversion losses and cost optimization. In: Hossain J, Mahmud A (eds) Renewable energy integration. Springer, Singapore, pp 327–345
3. Gonzalez-Longatt F, Regulski P, Wall P, Terzija V (2011) Induction generator model parameter estimation using improved particle swarm optimization and on-line response to a change in frequency. Presented at the IEEE on the power and energy society general meeting, Detroit, 2011
4. de Mello FP, Feltes JW, Hannett LN, White JC (1982) Application of induction generators in power systems. IEEE Trans Power Apparatus Syst PAS-101:3385–3393
5. Simões MG, Farret FA (2006) Renewable energy systems: design and analysis with induction generators. CRC Press, Boca Raton
6. Simões MG, Farret FA (2007) Alternative energy systems: design and analysis with induction generators (Power electronics and applications series). CRC Press, Boca Raton
7. Lai LL, Chan TF (2007) Distributed generation—induction and permanent magnet generators. Wiley, New Jersey
8. Al-Hinai A, Schoder K, Feliachi A (2003) Control of grid-connected split-shaft microturbine distributed generator. In: Proceedings of the 35th southeastern symposium on system theory, pp 84–88
9. Murthy SS, Jha CS, Ghorashi AH, Nagendra Rao PS (1989) Performance analysis of grid connected induction generators driven by hydro/wind turbines including grid abnormalities. In: Proceedings of the 24th intersociety conference energy conversion engineering (IECEC-89), vol 4. pp 2045–2050
10. Abdin ES, Xu W (1998) Control design and dynamic performance analysis of a wind turbine-induction generator unit. In: Proceedings of international conference on power system technology (POWERCON '98), vol 2. pp 1198–1202
11. Ansuji S, Shokoooh F, Schinzinger R (1988) Parameter estimation for induction machines based on sensitivity analysis. In: Industrial applications society 35th annual petroleum and chemical industry conference, 1988, record of conference papers, pp 35–40
12. Velez-Reyes M, Minami K, Verghese GC (1989) Recursive speed and parameter estimation for induction machines. In: Conference record of the 1989 IEEE industry applications society annual meeting, 1989, vol 1. pp 607–611
13. Bishop RR, Richards GG (1990) Identifying induction machine parameters using a genetic optimization algorithm. In: *Proceedings of Southeastcon '90, IEEE*, 1990, vol 2. pp. 476–479

14. Huang KS, Wu QH, Turner DR (2002) Effective identification of induction motor parameters based on fewer measurements. *IEEE Trans Energy Convers* 17:55–60
15. Sag T, Cunkas M (2007) Multiobjective genetic estimation to induction motor parameters. In: International Aegean conference on electrical machines and power electronics (ACEMP '07), pp 628–631
16. Huynh DC, Dunnigan MW (2010) Parameter estimation of an induction machine using advanced particle swarm optimisation algorithms. *Electr Power Appl IET* 4:748–760
17. Huynh DC, Dunnigan MW (2010) Parameter estimation of an induction machine using a dynamic particle swarm optimization algorithm. In: IEEE international symposium on industrial electronics (ISIE), 2010, pp 1414–1419
18. Marino P, Mungiguerra V, Russo F, Vasca F, (1996) Parameter and state estimation for induction motors via interlaced least squares algorithm and Kalman filter. In: IEEE 27th annual conference on power electronics specialists, PESC '96, 1996, vol 2. pp 1235–1241
19. Ursem RK, Vadstrup P (2003) Parameter identification of induction motors using differential evolution. In: The 2003 Congress on Evolutionary Computation, CEC '03, 2003, vol 2. pp 790–796
20. Karimi A, Choudhry MA, Feliachi A (2007) PSO-based evolutionary optimization for parameter identification of an induction motor. In: Proceedings of 39th North American power symposium, 2007, NAPS '07, pp 659–664
21. Chen G, Guo W, Huang K (2007) On line parameter identification of an induction motor using improved particle swarm optimization. In: Chinese control conference, 2007, CCC 2007, pp 745–749
22. Bakari KE, Kling WL (2010) Virtual power plants: an answer to increasing distributed generation. In: Conference on innovative smart grid technologies (ISGT Europe), 2010 IEEE PES, pp 1–6
23. González-Longatt F, Regulski P, Wall P, Terzija V (2011) Fixed speed wind generator model parameter estimation using improved particle swarm optimization and system frequency disturbances. In: The 1st IET conference on renewable power generation, RPG 2011, Edinburgh, pp 1–5
24. Terzija V (2007) Wide area monitoring protection and control—WAMPAC. In: International conference on information and communication technology in electrical sciences (ICTES 2007), pp I-1
25. Terzija V, Valverde G, Deyu C, Regulski P, Madani V, Fitch J et al (2011) Wide-area monitoring, protection, and control of future electric power networks. In: Proceedings of the IEEE, vol 99. pp 80–93
26. Kampisios K, Zanchetta P, Gerada C, Trentin A (2008) Identification of induction machine electrical parameters using genetic algorithms optimization. In: Industry applications society annual meeting, IAS '08, IEEE, 2008, pp 1–7
27. Filho EBS, Lima AMN, Jacobina CB (1991) Parameter estimation for induction machines via non-linear least squares method. In: Proceedings of international conference on industrial electronics, control and instrumentation, IECON '91, 1991, vol 1. pp 639–643
28. de Oliveira PJR, Seixas PF, Aguirre LA, Peixoto ZMA (1998) Parameter estimation of a induction machine using a continuous time model. In: Proceedings of the 24th annual conference of the IEEE on industrial electronics society, IECON '98, 1998, vol 1. pp 292–296
29. Fletcher R (1970) A new approach to variable metric algorithms. *Comput J* 13:317–322
30. Fletcher R, Powell MJD (1963) A rapidly convergent descent method for minimization. *Comput J* 6:163–168
31. Fletcher R, Powell MJD (1987) Practical methods of optimization, 2nd edn. Wiley, New York

