Implementation of slow coherency based controlled islanding using DIgSILENT powerfactory and MATLAB

Tyuryukanov, I.; Naglič, M.; Popov, M.; van der Meijden, M.A.M.M.

DOI
10.1007/978-3-319-50532-9_11

Publication date
2018

Document Version
Final published version

Published in
Advanced Smart Grid Functionalities Based on PowerFactory

Chapter 11
Implementation of Slow Coherency Based Controlled Islanding Using DIgSILENT PowerFactory and MATLAB

I. Tyuryukanov, M. Naglič, M. Popov and M. A. M. M. van der Meijden

Abstract Intentional controlled islanding is a novel emergency control technique to mitigate wide-area instabilities by intelligently separating the power network into a set of self-sustainable islands. During the last decades, it has gained an increased attention due to the recent severe blackouts all over the world. Moreover, the increasing uncertainties in power system operation and planning put more requirements on the performance of the emergency control and stimulate the development of advanced System Integrity Protection Schemes (SIPS). As compared to the traditional SIPS, such as out-of-step protection, ICI is an adaptive online emergency control algorithm that aims to consider multiple objectives when separating the network. This chapter illustrates a basic ICI algorithm implemented in PowerFactory. It utilises the slow coherency theory and constrained graph partitioning in order to promote transient stability and create islands with a reasonable power balance. The algorithm is also capable to exclude specified network branches from the search space. The implementation is based on the coupling of Python and MATLAB program codes. It relies on the PowerFactory support of the Python scripting language (introduced in version 15.1) and the MATLAB Engine for Python (introduced in release 8.4). The chapter also provides a

Electronic supplementary material
The online version of this chapter (https://doi.org/10.1007/978-3-319-50532-9_11) contains supplementary material, which is available to authorized users.

The original version of this chapter was revised: ESM files have been included. The erratum to this chapter is available at https://doi.org/10.1007/978-3-319-50532-9_15

I. Tyuryukanov (✉) · M. Naglič · M. Popov · M. A. M. M. van der Meijden
Intelligent Electrical Power Grids, Department of Electrical Sustainable Energy, Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands
e-mail: i.tyuryukanov@tudelft.nl

M. A. M. M. van der Meijden
TenneT TSO B.V., Utrechtseweg 310, 6812 AR Arnhem, The Netherlands

F. Gonzalez-Longatt and J. L. Rueda Torres (eds.), *Advanced Smart Grid Functionalities Based on PowerFactory*, Green Energy and Technology,
https://doi.org/10.1007/978-3-319-50532-9_11

case study to illustrate the application of the presented ICI algorithm for wide-area instability mitigation in the PST 16 benchmark system.

Keywords Intentional controlled islanding · Controlled network separation Generator coherency · Graph partitioning · Python scripting · DIgSILENT PowerFactory to MATLAB interface via Python

11.1 Introduction

Due to the electrical industry deregulation and massive grid integration of renewable energy sources, electric power systems (EPSs) are expected to operate close to their stability limits. Several large-scale blackouts during the recent decades [1, 2] have demonstrated the increased vulnerability of existing electric power grids. Due to these reasons, blackout prevention has become a topic of great importance for the operation of future EPS.

Intentional controlled islanding (ICI) is an adaptive wide-area protection algorithm belonging to the class of System Integrity Protection Schemes (SIPS) [3]. The basic idea is to define in an existing network a set of islands so that the initial disturbance, which could lead to a system collapse, remains confined within one of the islands. Studies of historical blackouts (e.g. [4]) show that a proper system islanding combined with load shedding and generator dropping has the potential to prevent wide-area blackouts. Compared to traditional SIPS, such as out-of-step protection, ICI is a real-time control technique which in general aims to consider multiple objectives, e.g. load-generation balance, generator coherency, transmission line availability, thermal limits, voltage stability and transient stability [5, 6]. Due to this highly adaptive and sophisticated nature, the design of ICI algorithms is currently an active research area. To enable a near real-time situational awareness, an ICI algorithm requires access to Wide-Area Measurement Systems (WAMSs) data.

An ICI method should be used as the last measure to rescue the power grid from a dangerous instability. Consequently, the overall ICI problem is commonly subdivided into two stages: *when to island* and *how to island* [7]. The first stage aims to promptly determine the "point of no return" after which only ICI can save the grid. The second stage aims to split the network in a way that results in a stable islanded operation with all restoration constraints satisfied. In this chapter, some important considerations regarding the second global challenge are presented, together with the implementation of a simple ICI algorithm using PowerFactory and MATLAB.

Among the multiple ICI objectives listed above, only the following ones will be considered in the simplified algorithm presented in this chapter, namely: generator coherency, transmission line availability, load-generation balance (in an indirect way) and transient stability (in an indirect way). The resulting solution promotes transient stability of islands by putting only coherent generators into each island and by reducing the changes in generators' electric power through cutting transmission lines with a small active power flow. At the same time, the opening of transmission

lines with a small active power flow turns out to reduce the MW interdependency between islands, thus promoting load-generation balance. Finally, the algorithm is capable of restricting the splitting cutset only to the lines equipped with synchro-check relays, as only these lines can be reconnected during the restoration process (see the transmission line availability constraint in [5]). However, it should be noted that a practical ICI algorithm requires many additional factors to be taken into account. Including all relevant aspects would require significantly more space. Therefore, the presented ICI implementation in PowerFactory is a basic algorithm which may serve as an illustration of the concept.

The rest of the chapter is organised as follows. Section 11.2 provides a brief review of the slow coherency theory which is used to determine the coherent groups of generators (CGG). Section 11.3 explains the graph partitioning approach utilised to find the lines with a small MW power flow while excluding some unavailable branches (e.g. lines without synchro-check relays or transformers) and satisfying the generator coherency constraint. Section 11.4 gives an overview of the ICI program structure written in Python scripting language for PowerFactory and MATLAB. Section 11.5 provides a case study of the illustrated ICI method on the PST 16 benchmark system. Finally, Sect. 11.6 summarises the approach and gives some concluding remarks.

11.2 Slow Coherency

An important task in the design of an ICI algorithm is to identify the areas in a power system which should be separated from each other in case of instability. While the actual borders of the ICI areas may change depending on the loading condition, it is important to ensure that generators in each area synchronise after the network is separated in controlled manner. One approach to meet this requirement is to utilise the slow coherency theory developed in [8–10]. It has been pointed out in [9] that generators forming a slow coherent group, i.e. generators swinging together at oscillatory frequencies of slow inter-area modes, have a relatively strong dynamic coupling between each other. In other words, weak connections in a power network manifest themselves through slow coherency [9].

Therefore, it is prudent to utilise slow coherency identification approaches in order to find generators which should be grouped together for the purpose of ICI. The load buses corresponding to each CGG can be identified by using a graph partitioning algorithm like one described in Sect. 11.3.

Slow coherency identification approaches [8–10] are model-based methods suitable for offline computations. They are based on calculation of right eigenvectors of the electromechanical model of the power system (see Sect. 11.2.1) corresponding to the dominant slow modes. The motivations underlying this approach can be found in [8, 11]. It should be noted that significant changes in the power system operating condition, such as topology changes or large load steps, may cause the *weakly coherent* generators to change their CGG [12]. Therefore,

signal-based slow coherency approaches, such as [12], are preferable for a practical ICI implementation in a physical power system.

Given the above considerations, the overall slow coherency grouping approach can be summarised as follows:

- Formulate the electromechanical model of the studied EPS (see Sect. 11.2.1).
- Linearise the model and find its state-space representation (see Sect. 11.2.2).
- Identify the r slowest electromechanical modes of the linearised model (see Sect. 11.2.3).
- Compute the right eigenvectors corresponding to the r slowest modes. Combine the eigenvector columns into a matrix and group its rows based on the grouping algorithm as described in Sect. 11.2.4. As each row corresponds to a generator, the resulting grouping will reveal the CGGs.

11.2.1 Electromechanical Modelling

The well-known electromechanical model of an n-machine power system can be derived given the following assumptions [13]:

- The mechanical power input of synchronous generators is constant.
- Generator mechanical damping and asynchronous power are negligible.
- Synchronous generators can be represented in the network by the constant-emf-behind-the-transient-reactance model.
- The generator rotor angle coincides with the angle of the emf behind the transient reactance.
- Loads can be represented by constant impedances.

Given the assumptions above, the electrical network can be represented as shown in Fig. 11.1.

The network shown in Fig. 11.1 can be reduced to contain only the generator internal buses (i.e. the buses behind the transient reactances) by using the procedure called Kron reduction [13]. The main idea of this procedure is to eliminate the nodal equations of the original network which have zero current injections. As only the

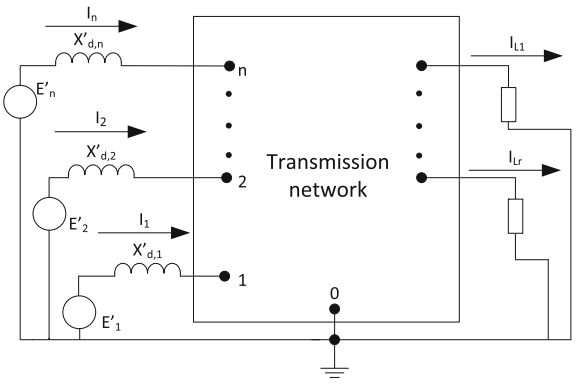


Fig. 11.1 Electromechanical model of a multi-machine power system

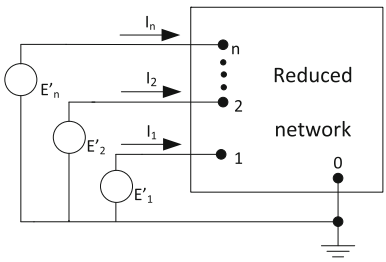


Fig. 11.2 Electromechanical model reduced to internal nodes of generators

generator internal buses' nodal equations have nonzero current injections, the voltages of the remaining network nodes can be represented as a linear combination of the internal generator voltages by exploiting the fact the left-hand sides of the network equations for the remaining nodes are zero. The reduced network obtained from the Kron reduction contains the equivalent admittances between every pair of internal generator nodes (i.e. the reduced network represents a full graph). Its graphical representation is shown in Fig. 11.2.

The reduced electromechanical model can be described by (11.1a) and (11.1b); see [9, 13].

$$\dot{\delta}_i = \omega_0(\omega_i - 1), \quad i = 1, \ldots, n \tag{11.1a}$$

$$\begin{aligned} 2H_i\dot{\omega}_i = &- \sum_{\substack{j=1 \\ j \neq i}}^{n} E_i' E_j' B_{ij} \sin(\delta_i - \delta_j) - \sum_{\substack{j=1 \\ j \neq i}}^{n} E_i' E_j' G_{ij} \cos(\delta_i - \delta_j) \\ &- D_i(\omega_i - 1) + P_{m,i} - E_i^{'} G_{ii}^2, \quad i = 1, \ldots, n \end{aligned} \tag{11.1b}$$

where δ_i is the rotor angle of generator i in rad, ω_i is the rotor speed of generator i per unit, H_i is the inertia constant of generator i in s, D_i is the damping coefficient of generator i per unit, $P_{m,i}$ is the mechanical power of generator i per unit, E' is the constant voltage behind the transient reactance per unit, G_{ij} and B_{ij} are the real and imaginary components of the $(i,j)^{th}$ entry of the admittance matrix of the reduced network (see Fig. 11.2) per unit and ω_0 is the base frequency in rad s^{-1}.

11.2.2 Linearised Model

The coherency behaviour of the generators can be more easily understood from the linearised electromechanical model. Equations (11.1a) and (11.1b) can be linearised about an equilibrium $\delta_i = \delta_{i,0}$ and $\omega_i = 1$, where $\delta_{i,0}$ is the equilibrium rotor angle of the ith generator. The equilibrium rotor angles of all generators can be obtained in a convenient fashion by calculating the power flow solution for the original

electromechanical model in Fig. 11.1 for the loading condition of interest. The details of this procedure in PowerFactory are given in Sect. 11.4.

The resulting linearised model derived from (11.1a) and (11.1b) is described by (11.2a) and (11.2b); see [8, 9].

$$\Delta\dot{\delta}_i = \omega_0 \Delta\omega_i, \quad i = 1, \ldots, n \tag{11.2a}$$

$$2H_i \Delta\dot{\omega}_i = -D_i \Delta\omega_i - \sum_{j=1}^{n} k_{ij} \Delta\delta_j, \quad i = 1, \ldots, n \tag{11.2b}$$

where $\Delta\delta_i = \delta_i - \delta_{i,0}$, $\Delta\omega_i = \omega_i - 1$ are the small perturbations of the rotor angles and speeds around their equilibrium values and the terms k_{ij} are according to (11.3a) and (11.3b).

$$k_{ij} = -E'_i E'_j [B_{ij} \cos(\delta_{i,0} - \delta_{j,0}) - G_{ij} \sin(\delta_{i,0} - \delta_{j,0})], \quad j \neq i \tag{11.3a}$$

$$k_{ii} = -\sum_{\substack{j=1 \\ j \neq i}}^{n} k_{ij} \tag{11.3b}$$

Equations (11.2a) and (11.2b) can be written in the matrix form as (11.4).

$$\begin{bmatrix} \Delta\dot{\delta}_1 \\ \Delta\dot{\delta}_2 \\ \cdots \\ \Delta\dot{\delta}_n \\ \Delta\dot{\omega}_1 \\ \Delta\dot{\omega}_2 \\ \cdots \\ \Delta\dot{\omega}_n \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \begin{matrix} \omega_0 & 0 & \cdots & 0 \\ 0 & \omega_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \omega_0 \end{matrix} \\ -\frac{1}{2}\mathbf{H}^{-1}\mathbf{K} & -\frac{1}{2}\mathbf{H}^{-1}\mathbf{D} \end{bmatrix} \begin{bmatrix} \Delta\delta_1 \\ \Delta\delta_2 \\ \cdots \\ \Delta\delta_n \\ \Delta\omega_1 \\ \Delta\omega_2 \\ \cdots \\ \Delta\omega_n \end{bmatrix} \tag{11.4}$$

where

$$\mathbf{H} = \mathrm{diag}(H_1, H_2, \ldots, H_n)$$
$$\mathbf{D} = \mathrm{diag}(D_1, D_2, \ldots, D_n)$$
$$\mathbf{K} = [k_{ij}]$$

By expressing $\Delta\omega_i = \frac{\Delta\dot{\delta}_i}{\omega_0}$ and neglecting damping, it is possible to reduce (11.4) and (11.5), which is the common form of the power system electromechanical model used for slow coherency analysis [8, 9]. The motivation to represent synchronous generators by simplified 2nd order models and to neglect damping is based on the observation that the CGGs do not depend significantly on the level of detail used in modelling the generating units [8, 11].

$$\Delta\ddot{\delta} = -\frac{1}{2}\mathbf{H}^{-1}\omega_0\mathbf{K}\Delta\delta \tag{11.5}$$

The properties of the linearised dynamic models (11.4) and (11.5) can be analysed by studying the eigenvalues and eigenvectors of their state matrices. In particular, the standard small-signal stability analysis [14, 15] can be performed for the model described by (11.4) in order to extract mode shapes corresponding to the electromechanical state variables $\Delta\delta$ and $\Delta\omega$. It should be noted that Eq. (11.5) fully describes the properties of the complete state-space model (11.4), given that the damping is neglected in (11.4). In particular, if λ_i is an eigenvalue of the state matrix of (11.5), then $\pm\sqrt{\lambda_i}$ are the eigenvalues of the state matrix of (11.4) with all damping constants set to zero [9].

Despite the fact that the model (11.5) is commonly used in the literature to analyse slow coherency, the standard state-space model (11.4) is better suitable for the slow coherency analysis with PowerFactory, as it is the model that is readily available through the Modal Analysis Toolbox of PowerFactory.

As the complete state-space model (11.4) is being utilised, it may be useful to review the difference between the terms *mode* and *eigenvalue*. A real eigenvalue corresponds to a non-oscillatory mode, while a complex conjugate eigenvalue *pair* corresponds to an oscillatory mode in the time-domain response of the linearised power system model [14].

11.2.3 Selection of Number of Slowest Modes

Slow coherency is defined as coherency with respect to the slowest modes of a system [9]. Although there are algorithms available to group generators in a power system with respect to the r slowest modes, where r is a predefined integer, the optimal value of r is not always known. Several references, e.g. [9], use the *eigengap* heuristic (11.6) in order to determine the point of separation between slow and fast electromechanical modes.

$$\varepsilon_i = |\mathrm{Im}(\lambda_{k+2})| - |\mathrm{Im}(\lambda_k)|, \quad k = 3, 5, \ldots, 2n-3, \quad i = 1, 2, \ldots, n-1 \tag{11.6}$$

where λ_k is the kth complex conjugate eigenvalue of the state matrix of Eq. (11.4), and all λ_k have been sorted in the increasing order or their imaginary parts. Given such ordering of eigenvalues, their counting starts at 3, because the state matrix of (11.4) has one zero eigenvalue and one small negative eigenvalue as the only real eigenvalues. All further eigenvalues come in complex conjugate pairs, and only the eigengaps between the slowest oscillatory modes are of interest.

Given Eq. (11.6), the number of slowest electromechanical modes to be considered for generator grouping can be expressed as follows:

$$r = \arg\max_i \varepsilon_i + 1 \tag{11.7}$$

where the increment of one in (11.7) is necessary and can be related to the two real eigenvalues of the state matrix in (11.4) which correspond to the common motion of the rotor angles and speeds of all generators [9]. In other words, the minimal number of slowest modes to separate the network is two, whereby the minimal value of $\arg\max_i \varepsilon_i$ is one.

11.2.4 Generator Grouping Algorithm

The r right eigenvectors corresponding to the r slowest modes of the power system model (11.4) serve as an input for the coherency grouping algorithm. As pairs of complex conjugate eigenvalues correspond to one oscillation mode, it is only necessary to take eigenvectors corresponding to one of two complex conjugate eigenvalues. It is possible to use both eigenvector entries related to rotor angles and to rotor speeds (i.e. both rotor angle and rotor speed mode shapes), as they show the same pattern. The rotor speed eigenvector entries are finally adopted for the analysis.

Among several slow coherency identification algorithms available in the literature, the so-called tight coherency grouping algorithm [10] is chosen to find CGGs for the purpose of ICI. It is capable of automatic detection of the number of CGGs. Moreover, CGGs identified by using this algorithm usually consist of generators which are electrically close. The generator grouping procedure [10] has several presteps that are given below.

- Combine the rotor speed mode shapes obtained from the r slowest eigenvectors into a matrix $\mathbf{V_s}$ consisting of n rows and r columns.
- Normalise the columns of the matrix $\mathbf{V_s}$ to have the length one.
- Define the slow coherency similarity between machines i and j as the cosine of the angle between w_i and w_j, which are the respective rows of the matrix $\mathbf{V_s}$.

$$d_{ij} = \frac{w_i w_j^T}{\|w_i\| \|w_j\|} \tag{11.8}$$

where $\|\cdot\|$ represents the vector length, i.e. the Euclidean norm of a vector.

- Define a tolerance γ usually ranging from 0.9 to 0.95. If the slow coherency similarity (11.8) between machines i and j is larger than γ, the machines are said to be coherent.

- Define a coherency matrix **C** as

$$[C_{ij}] = d_{ij} - \gamma \tag{11.9}$$

Only the extraction of *loose* coherent areas from the algorithm [10] is described below and implemented in MATLAB. The complete algorithm is available in the MATLAB-based Power System Toolbox (PST) [16], which is available online. Based on the given presteps, the set of rules to decide on *loosely* coherent generator groups can be summarised as follows.

- Machines i and j are coherent if $[C_{ij}] > 0$.
- If machines i and j are coherent and machines j and k are coherent, then machines i and k are also coherent.

It was observed empirically that the above algorithm usually performs well at identifying generator slow coherency. Higher values of the tolerance parameter γ correspond to smaller and tighter CGGs which tend to be robust even to significant changes in the power network (e.g. topological changes). Therefore, the value of γ of 0.95 is assumed for the coherency estimation in the subsequent sections.

11.3 Graph Partitioning

The slow coherency method presented in Sect. 11.2 solves the problem of identifying which generators can be grouped together for the purpose of ICI. For the complete islanding solution, a set of lines to be opened needs to be determined based on the multiple constraints or a subset of constraints (see Sect. 11.1).

The splitting cutset determination procedure is based on graph partitioning and aims at identifying the lines carrying the least amount of active power flow (shortly referred to as MW-flow). Its main steps are summarised below:

- Construct a weighted undirected graph $G = (V, E, W)$ representing the MW-flows in an electric power network consisting of m buses. The nodes and edges of G can be denoted as $v_i \in V$, $i = 1, 2, \ldots, m$ and $e_{ij} \in E \subset V \times V$, $i = 1, 2, \ldots, m$, respectively. The weight $w_{ij} = W(e_{ij})$, $i = 1, 2, \ldots, m$ of the edge e_{ij} represents the averaged active power flow through the respective power network branch.
- Reduce the graph G by following the guidelines presented in Sect. 11.3.1, which are largely based on [17] and [18]. The graph reduction procedure primarily serves the purpose of incorporation of generator coherency and transmission line availability constraints into the graph partitioning, but it also increases the computational efficiency by reducing the size of the problem.
- Apply the spectral clustering method briefly described in Sect. 11.3.2 to the reduced graph.

- Post-process the output of spectral clustering as illustrated in Sect. 11.3.3 in order to identify the resulting islands in the power network.

11.3.1 Graph Reduction

The MW-flow graph G can be reduced in two steps by following the corresponding steps of the procedure outlined in [17]:

- Collapse the edges of G corresponding to network elements which cannot be included into an islanding cutset (e.g. transformers and lines without synchro-check relays; see [5]) to single nodes. Such graph edges are further referred to as *unavailable edges*, as the corresponding power network branches are referred to as *unavailable network branches* [5].
- In the graph obtained after the reduction of unavailable edges, find subnetworks corresponding to the previously identified CGGs, e.g. using a shortest path algorithm as in [17]. Merge the found subnetworks into single nodes. Obtain the connectivity and weights of the final reduced graph (further referred to as G^R) by following the guidelines presented in [17, 18]. The number of nodes in G^R is m_R.

All cuts of the final reduced graph inherently satisfy the generator coherency and transmission line availability constraints. It is worth to note that, depending on the utilised subnetwork construction algorithm, the search for the CGG subnetworks may require multiple initialisations. However, as the utilised coherency algorithm is an offline model-based technique, the subnetworks do not need to be produced in an online fashion. In other words, the graph reduction is essentially an offline procedure; see [17].

11.3.2 Spectral Clustering

Spectral clustering is a well-established clustering technique based on graph representation of the input dataset [19]. Since electric power networks can be naturally represented as graphs, it is appealing to use spectral clustering for the identification of power network buses which are tightly coupled in terms of active power flow [5, 20]. In order to produce islands of balanced size, the *normalised* graph Laplacian matrix $\mathbf{L_n}$ is preferable for spectral clustering [19, 20]. Given the aforementioned definition of graph G, it can be computed according to (11.10); see [5, 21].

$$[\mathbf{L_n}_{ij}] = \begin{cases} 1, & \text{if } i = j \\ \frac{-w_{ij}}{\sqrt{d_i}\sqrt{d_j}}, & \text{if } i \neq j \text{ and } (i,j) \in E \\ 0, & \text{otherwise} \end{cases} \tag{11.10}$$

where $d_i = \sum_{j=1}^{m} w_{ij}$ is the weighted degree of the node v_i. For the power flow graphs other than G, the definition (11.10) should be adjusted accordingly. The normalised Laplacian of the reduced graph G^R described in Sect. 11.3.1 is of interest for the purpose of ICI. It is further referred to as $\mathbf{L_{n,R}}$.

With having $\mathbf{L_{n,R}}$ computed, the next step is to calculate its first r smallest eigenvalues and the corresponding eigenvectors, where r previously denoted the number of CGGs in the EPS; see Sect. 11.2. The number of computed eigenvectors usually corresponds to the desired number of islands [19]. That is, the goal is to identify a separate island for each CGG. The computed eigenvectors are combined into the matrix $\mathbf{X} = \mathbb{R}^{m_R \times r}$. According to [19], each row of the matrix $\mathbf{X}$ should be normalised to have length 1. The row normalisation process results in the matrix $\mathbf{Y} = \mathbb{R}^{m_R \times r}$, the rows of which represent the m_R coordinates of points in the r-dimensional Euclidian space. This so-called *spectral r-embedding* [5] reveals the clustering structure of the m_R reduced power network nodes with respect to active power flows between them.

11.3.3 Identification of Islands

Spectral embedding does not take the actual interconnections between the nodes in the input graph into account. In order to overcome this issue, it was recommended in [21] to define a new metric in spectral embedding which measures the distances between the points according to their connectivity in the underlying graph G^R. This is essentially equivalent to the creation of a new graph G^R_{SC} which has the same sets of nodes and edges as G^R, but the edge weights are redefined according to the Euclidean distances between the respective points in the spectral embedding. Then, the distance between any two points in the spectral embedding is defined as the shortest path distance between the respective nodes of G^R_{SC}.

As each show-coherent generator group is represented by a single node in G^R and in the resulting spectral embedding, it is possible to determine the boundaries of the islands by assigning the remaining load buses to the nearest (in the sense of the distance metric defined above) CGG. In this way, each identified CGG becomes assigned to its own partition.

After clusters of nodes have been identified by following the above procedure, the nodes of each cluster are mapped back to the nodes of the original graph, and the potential cutset is defined as the edges between the buses belonging to different clusters. By using the identified cutset, it is possible to separate the CGG which goes out-of-step with the rest of the network.

11.4 ICI Program Implementation

The PowerFactory implementation of the presented simplified ICI algorithm is based on the coupling of the DIgSILENT PowerFactory [22] and MATLAB [23] software tools through Python. In this way, the advantages of both software environments can be combined in order to implement more sophisticated algorithms. The mentioned coupling has become possible since releases 15.1 of PowerFactory and 8.4 (R2014b) of MATLAB. Release 15.1 of PowerFactory has introduced a dynamic Python module `powerfactory.pyd` as a means to interface PowerFactory with Python. Release 8.4 of MATLAB has introduced MATLAB Engine for Python, which allows to start or connect to MATLAB from Python.

Python is a non-proprietary high-level general-purpose interpreted programming language, which supports both object-oriented (OOP) and procedural programming paradigms. Python has been introduced to PowerFactory as an alternative to the built-in DIgSILENT programming language (DPL). The *PowerFactory* Python module provides the equivalents for the majority of functionalities available via DPL. By importing the *PowerFactory* Python module inside of a Python script, it is possible to control PowerFactory from the Python environment in the same way as it is possible with DPL. Moreover, the rich programming capabilities of Python become available for processing of data obtained from PowerFactory. Lastly, it becomes relatively easy to send data obtained from PowerFactory to other applications which also have an interface with Python (e.g. to MATLAB) and to receive data back from those applications to PowerFactory. An important additional advantage of Python scripting over DPL is the possibility to use debugger tools included into many Python Integrated Development Environments (IDEs).

In order to control PowerFactory via Python, a Python script file needs to be created externally on the hard drive and linked to PowerFactory via a **.ComPython* object. An important difference between **.ComDPL* objects for DPL scripts and **.ComPython* objects for Python scripts is that the actual script is not stored inside of the latter ones. It is also possible to control PowerFactory via Python without creating a **.ComPython* object. This can be accomplished by running PowerFactory in engine mode (see the User Manual [22] for more information).

11.4.1 ICI Program Components

The capabilities of Python as a mainstream programming language facilitate more structured and sophisticated program designs for PowerFactory (as compared to DPL). In particular, the support of OOP by Python is useful for writing larger programming projects related to PowerFactory.

The ICI program is a small-scale project written in Python and MATLAB languages. Its components are designed with the idea of code modularity in mind,

which is done in order to promote the code reuse. Although the program codes are too extensive to put them inside of the chapter, it is still possible to describe the functionalities of the main components of the project.

`ICI_PST16.py`: This file contains the top-level Python script which is started from PowerFactory and calls all other Python and MATLAB functions and class methods.

`pypf.py`: This file contains a Python class whose attributes link to the relevant data of the investigated PowerFactory project (e.g. to all network branches, terminals and generators). The class also contains several methods to manipulate the data contained in its attributes (`rename_buses_branches`, `extract_flows`, `insert_ici_events`, `extract_eig`) as well as auxiliary methods to maintain the class contents.

`pyUtils.py`: This file contains a Python module with auxiliary functions that are used inside of the methods of the `pypf` class.

`formEigVecMatr.m`: This MATLAB function selects the r eigenvectors corresponding to the r slowest system modes and returns the matrix $\mathbf{V_s}$ introduced in Sect. 11.2.4. The number r can be either predefined or estimated based on (11.6) and (11.7).

`coh_loose.m`: This MATLAB function implements the generator coherency grouping algorithm described in Sect. 11.2.4.

`digsi2graph.m`: This MATLAB function converts branch power flows extracted from PowerFactory to the MATLAB representation of the power flow graph G from Sect. 11.3. The graph G is modelled in MATLAB by its adjacency and incidence matrices. The function can also convert a list of unavailable edges extracted from PowerFactory to a MATLAB-compatible representation.

`COSC.m`: This MATLAB function implements the constrained graph partitioning algorithm described in Sect. 11.3 and returns the resulting cutset to Python. In order to accomplish this, it makes use of `digsi2graph.m` as well as about 10 other dedicated MATLAB functions.

11.4.2 Interaction Between Program Components

As the program is comprised of pieces which are largely independent, it is useful to illustrate the capabilities and behaviour of the separate components on an example that involves their interaction. The top-level Python script used for the simulation of the ICI case study in Sect. 11.5 is chosen as such an example. Its flow chart is depicted in Fig. 11.3. Although the original Python script implements a particular ICI test case, its flow chart in Fig. 11.3 is more generic and may correspond to a variety of system instability scenarios followed by ICI. Some comments about the flow chart steps are given below.

1. A link between the electric network model in PowerFactory and the graph G defined in Sect. 11.3 needs to be established in order to map the ICI solutions

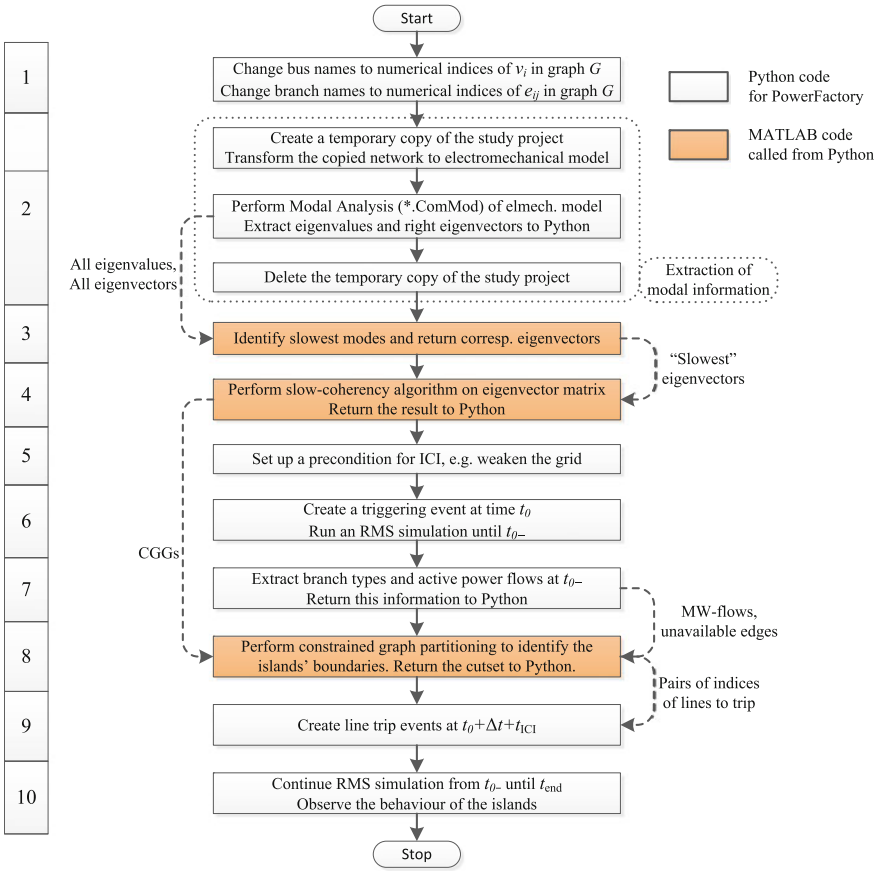


Fig. 11.3 Top-level ICI program structure

obtained for the graph G back to the network. As the renaming step is irreversible, it is recommended for the purpose of ICI study to create a separate copy of the investigated project.

2. Slow coherency is identified based on the electromechanical model of power system, which implies 2nd order generator models with zero damping and all generator controls disabled. As actual power networks are rarely modelled with such assumptions, it is convenient to make a temporary copy of the active project and to modify it accordingly. Then, the output of the modal analysis command (*.*ComMod*) performed on the modified project copy is returned back to Python. This output is comprised of all eigenvalues and right eigenvectors of the electromechanical model (11.4). Finally, the temporary copy of the project is deleted.

3. The r right eigenvectors corresponding to the r slowest modes are returned combined to the matrix $\mathbf{V_s}$. The number r can be either predefined or identified based on (11.6) and (11.7).
4. This step implements the identification of *loosely* coherent generator groups following the description in Sect. 11.2.4.
5. Usually, some sequence of adverse events precedes a network instability that causes ICI to operate. This sequence of events can often be reproduced by a script that changes the default network condition accordingly and, if necessary, rolls it back upon completion of the main program.
6. For a given instability precondition, a simulation event needs to be created that actually triggers the instability during the time-domain simulation run. The time instant of this event is denoted as t_0. After the time Δt following the triggering event, the instability is detected, which initiates the execution of the ICI algorithm.
7. Programmatically, it is easier to extract power flows in the network at t_{0-} (i.e. nearly at the time of the triggering event, but not including the triggering event) by first running the RMS-type time-domain simulation until t_{0-} and then extracting the resulting power flow variables for each network branch after the simulation has stopped.
8. This step implements the constrained graph partitioning algorithm outlined in Sect. 11.3.
9. This step inserts transmission line trip events which are used to separate the network after the RMS-type time-domain simulation resumes at Step 10. The time t_{ICI} represents an additional time delay related to the calculation of the islanding cutset and the actual network separation.
10. Resume the RMS simulation at t_{0-} in order to simulate the network separation and the resulting post-islanding transients.

In the algorithm flow chart in Fig. 11.3, slow coherent generator groups for the default network configuration are taken as input for islanding. Although it makes the algorithm less adaptive to the actual operating condition, the offline calculation and analysis of CGGs are easier to implement. This assumption allows to keep the ICI implementation at a manageable level. The assumption can be justified by the fact that slow coherent generator groups represent a fundamental property of the network related to its topological structure [9, 11] and usually do not experience significant changes.

11.5 ICI Case Study

This section presents simulation results for the sample ICI algorithm. The high-level structure of the case study has already been mentioned in Sect. 11.4 in the form of a flow chart. This section presents a particular scenario, in which the sample ICI algorithm is applied to mitigate a wide-area instability in the PST 16 benchmark system.

11.5.1 Test System

The ICI case study is based on the PST 16 benchmark system [24]. It consists of three meshed areas, 66 buses, 16 generators, 28 transformers and 51 transmission lines. Due to the unbalanced load and generation in the areas and the presence of long inter-area tie lines, the PST 16 test system is useful for studies of various stability problems [24]. The slightly modified version of the PST 16 test system model is used for the simulations. In particular, the thyristor-controlled series compensator (TCSC) has been removed, and the tie line between areas A and C is modelled to have a half of its original impedance (i.e. as a double-circuit line). The test system is shown in Fig. 11.4 together with the final network separation result.

In the nominal operating condition, all network elements are in service. The loads were slightly adjusted in order to produce a more distinct and realistic sequence of events leading to instability. The resulting active load and generation for each area are summarised in Table 11.1.

As it can be seen, Area A has a significant excess of generation, and it, in fact, supplies power to the neighbouring areas. At the same time, Area C has a significant excess of load, while Area B is more balanced. Area C is the main power consumer, and it is largely supplied from Area A, which leads to a significant power flow through tie line A–C. This power flow may become especially large if tie line A–B is disconnected, and power cannot be sent from Area A to Area B directly (cf. Sect. 11.5.2). Thus, the separation of the areas according to the tie lines can lead to a poor power balance and large changes in electric power outputs of the generators.

11.5.2 Wide-Area Instability Scenario

The PST 16 test system represents a relatively small power network with a limited number of realistic scenarios of a wide-area instability. The following instability scenario has been finally chosen:

- Transmission line C4–C6 is out of service due to maintenance.
- Tie line A–B trips due to a sustained short-circuit.
- As the consequence of the power flow redistribution, transformer station C8 3T is disconnected due to a 75% overload.
- The disconnection of transformer C8 3T causes wide-area rotor angle instability accompanied by low voltages in Area C.

Controlled islanding should be applied after the disconnection of transformer C8 3T as soon as the instability has been detected. Without loss of generality, an advanced instability detection mechanism capable of identifying the out-of-step condition before the angular differences reach 180° is assumed for this case study. However, as a controlled islanding algorithm may in practice require an additional

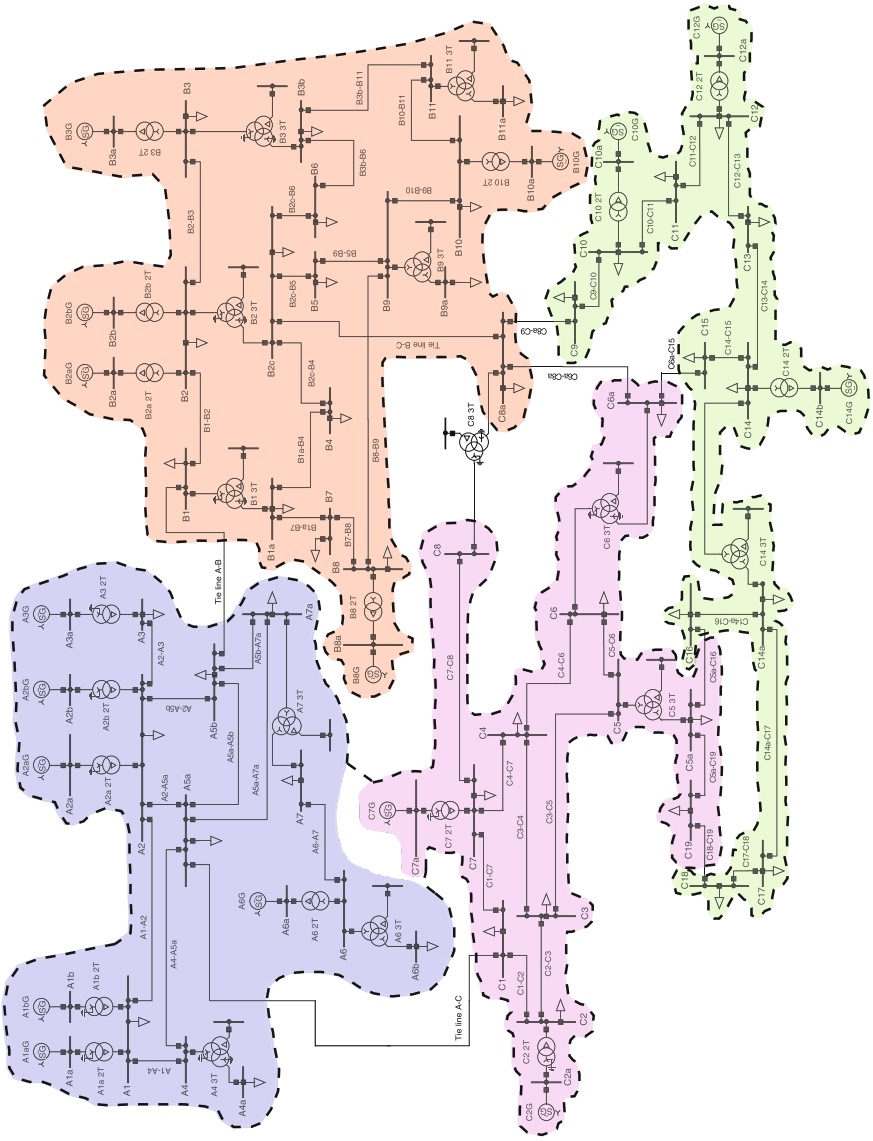


Fig. 11.4 PST 16 test system [24]. Blue: area of CGG-1; magenta: area of CGG-2; green: area of CGG-3; red: area of CGG-4. The areas have been determined by using the graph partitioning algorithm. The boundaries of the resulting islands are shown with dashed lines

Table 11.1 Load and generation in PST 16 system

	Active load (MW)	Generation (MW)
Area A	2535	5082
Area B	7215	6627
Area C	7833	6051
Total	17,583	17,761

non-negligible amount of time to calculate a solution, it is desirable to predict the emerging instability straight after the triggering event.

11.5.3 Controlled Islanding

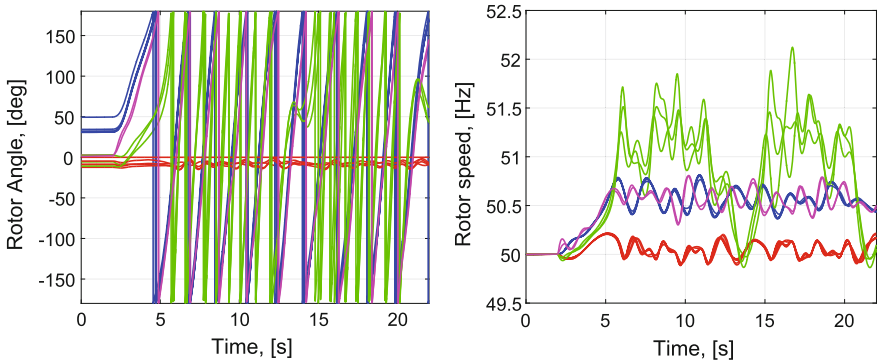
The first step of the ICI algorithm is to perform slow coherency analysis of the PST 16 network for the nominal operating condition. The eigenvalues and right eigenvectors of the electromechanical model (11.4) are obtained with the built-in Modal Analysis command of PowerFactory. The 16 smallest eigenvalues are {0, −0.0548, ±3.142j, ±3.718j, ±5.021j, ±5.265j, ±5.716j, ±5.894j, ±6.099j}. As it can be seen, there is a large gap between the complex eigenvalues ±3.718j and ±5.021j. Therefore, the number of eigenvectors for slow coherency analysis is three [8–10], and the generator grouping algorithm of Sect. 11.2.4 is performed with the eigenvectors corresponding to the eigenvalues {0, 3.142j, 3.718j}. The resulting grouping for $\gamma = 0.95$ is presented in Table 11.2.

As it can be seen, four coherent generator groups have been identified. The tolerance value γ for the coherency grouping algorithm has been taken at the highest recommended value of 0.95, which has resulted in four identified coherent groups. It is worth noting that the groups would be the same for γ equal to 0.9, and in general the CGGs in Table 11.2 are tight. The boundaries of each coherent generator group in terms of predisturbance active power flow are determined by the graph partitioning algorithm described in Sect. 11.3 and represented in Fig. 11.4.

After the disconnection of transformer C8 3T, the resulting unstable transient is shown in Fig. 11.5. As it can be seen, the actual out-of-step condition involves the loss of synchronism between CGG-3 and the rest of the network. Therefore, the

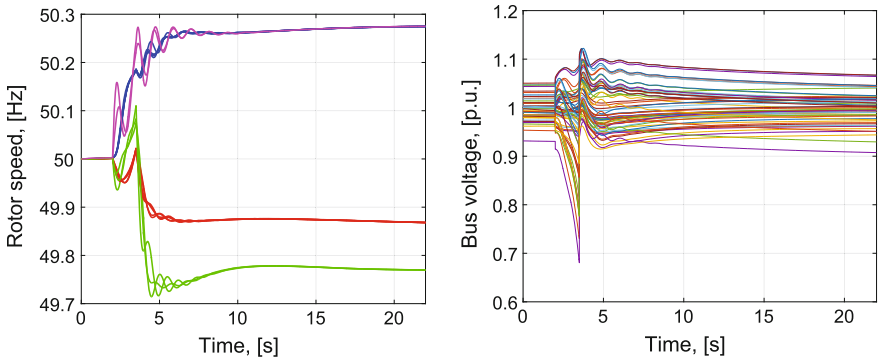
Table 11.2 Slow coherent groups of generators in the PST-16 network

Generator group	Generator buses
CGG-1	A1aG, A1bG, A2aG, A2bG, A3G, A6G
CGG-2	C2G, C7G
CGG-3	C10G, C12G, C14G
CGG-4	B2aG, B2bG, B3G, B10G, B8G



(a) Generator rotor angles (b) Generator rotor speeds

Fig. 11.5 Time-domain simulation of wide-area instability. Blue: signals of CGG-1; magenta: signals of CGG-2; green: signals of CGG-3; red: signals of CGG-4



(a) Generator rotor speeds (b) Bus voltage magnitudes

Fig. 11.6 Time-domain simulation of ICI

cutset between the area of CGG-3 and the rest of the network needs to be tripped. The resulting transient responses can be observed in Fig. 11.6. The time period Δt from the disconnection of transformer C8 3T to the initiation of the controlled islanding algorithm is 1 s. The time period t_{ICI} to compute and implement islanding is assumed 0.5 s, which is around the expected time for a controlled islanding algorithm to return a solution [5].

The resulting islands have a good power balance with their steady-state frequencies close to the nominal frequency of 50 Hz and the maximal voltage deviation not exceeding 0.1 p.u. In certain cases, the initial splitting boundary may also require a post-processing in order to improve the power balance of the islands. However, the multiple additional questions arising in the design of adaptive power network separation schemes are beyond the scope of the chapter.

11.6 Conclusion

A simplified ICI procedure to adaptively separate the network following a wide-area instability has been presented in this chapter. The procedure focuses on grouping the coherent generators together and finding the minimal active power flow cut. The methodology is applied to the PST 16 benchmark system in order to show the feasibility of the approach. The algorithm has been implemented by using the Python scripting language, which is available in PowerFactory since version 15.1. Through Python, PowerFactory could gain access to MATLAB and call the MATLAB components of the ICI code. Moreover, the Python part of the ICI code takes advantage of the object-oriented capabilities of the Python programming language.

It has been mentioned throughout this chapter that the original ICI problem is a very comprehensive one and includes many additional questions. Among others, the following issues are not considered: adaptive generator coherency estimation, voltage stability, equipment thermal limits. However, the demonstrated ICI methodology in PowerFactory can serve as the basis for the development of more advanced controlled network separation algorithms. Many of the unaddressed issues are currently being pursued as future work.

Acknowledgements This study was financially supported by the Dutch Scientific Council NWO-STW, under the project 408-13-025 within the program of Uncertainty Reduction of Sustainable Energy Systems (URSES) in collaboration with TenneT TSO and the Dutch National Metrology Institute, van Swinden laboratory.

References

1. G. Andersson, P. Donalek, R. Farmer, N. Hatziargyriou, I. Kamwa, P. Kundur, N. Martins, J. Paserba, P. Pourbeik, J. Sanchez-Gasca, R. Schulz, A. Stankovic, C. Taylor, V. Vittal, 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 (2005)
2. J. Romero, Blackouts illuminate India's power problems. IEEE Spectr. **49**(10), 11–12 (2012)
3. V. Madani, D. Novosel, S. Horowitz, M. Adamiak, J. Amantegui, D. Karlsson, S. Imai, A. Apostolov, IEEE psrc report on global industry experiences with system integrity protection schemes (sips). IEEE Trans. Power Deliv. **25**(4), 2143–2155 (2010)
4. B. Yang, V. Vittal, G.T. Heydt, Slow-coherency-based controlled islanding-a demonstration of the approach on the August 14, 2003 blackout scenario. IEEE Trans. Power Syst. **21**(4), 1840–1847 (2006)
5. J. Quirós-Tortós, R. Sánchez-García, J. Brodzki, J. Bialek, V. Terzija, Constrained spectral clustering-based methodology for intentional controlled islanding of large-scale power systems. IET Gener. Transm. Distrib. **9**(1), 31–42 (2015)
6. Q. Zhao, K. Sun, D.Z. Zheng, J. Ma, Q. Lu, A study of system splitting strategies for island operation of power system: a two-phase method based on obdds. IEEE Trans. Power Syst. **18**(4), 1556–1565 (2003)

7. H. Shao, S. Norris, Z. Lin, J. Bialek, Determination of when to Island by Analysing Dynamic Characteristics in Cascading Outages, in *2013 IEEE Grenoble PowerTech* (2013), pp. 1–6
8. B. Avramovic, P.V. Kokotovic, J.R. Winkelman, J.H. Chow, Area decomposition for electromechanical models of power systems. Automatica **16**(6), 637–648 (1980)
9. J.H. Chow, *Time-Scale Modeling of Dynamic Networks with Applications to Power Systems, Lecture Notes in Control and Information Sciences,* vol 46 (Springer-Verlag, Berlin and New York, 1982)
10. J.H. Chow, New Algorithms for Slow Coherency Aggregation of Large Power Systems, in *Systems and Control Theory for Power Systems, IMA Volumes in Mathematics and its Applications*, vol. 64, ed. by J.H. Chow, P.V. Kokotović, R.J. Thomas (Springer-Verlag, New York, 1995)
11. H. You, V. Vittal, X. Wang, Slow coherency-based islanding. IEEE Trans. Power Syst. **19**(1), 483–491 (2004)
12. M.A.M. Ariff, B.C. Pal, Coherency identification in interconnected power system-an independent component analysis approach. IEEE Trans. Power Syst. **28**(2), 1747–1755 (2013)
13. P.M. Anderson, A.A. Fouad, *Power System Control and Stability,* 2nd edn. (IEEE Press Power Engineering Series, IEEE Press and Wiley-Interscience, Piscataway, N.J, 2003)
14. P. Kundur, N.J. Balu, M.G. Lauby, *Power System Stability and Control* (The EPRI power system engineering series, McGraw-Hill, New York, 1994)
15. G. Rogers, *Power System Oscillations The Springer International Series in Engineering and Computer Science, Power Electronics and Power Systems* (Springer, US, Boston, MA, 2000)
16. C. Jh, C. Kw, A toolbox for power system dynamics and control engineering education and research. IEEE Trans. Power Syst. **7**(4), 1559–1564 (1992)
17. G. Xu, V. Vittal, Slow coherency based cutset determination algorithm for large power systems. IEEE Trans. Power Syst. **25**(2), 877–884 (2010)
18. S. Rangapuram, M. Hein, Constrained 1-Spectral Clustering, in *International Conference on Artificial Intelligence and Statistics (AISTATS*) (2012)
19. U. von Luxburg, A tutorial on spectral clustering. Stat. Comput. **17**(4), 395–416 (2007)
20. L. Ding, F.M. Gonzalez-Longatt, P. Wall, V. Terzija, Two-step spectral clustering controlled islanding algorithm. IEEE Trans. Power Syst. **28**(1), 75–84 (2013)
21. R.J. Sanchez-Garcia, M. Fennelly, S. Norris, N. Wright, G. Niblo, J. Brodzki, J.W. Bialek, Hierarchical spectral clustering of power grids. IEEE Trans. Power Syst. **29**(5), 2229–2237 (2014)
22. DIgSILENT GmbH, *PowerFactory v15.2.5 User Manual* (Gomaringen, Germany, 2015) online Edition. Available at http://www.digsilent.de
23. MATLAB, version 8.6.0 (R2015b) (The MathWorks Inc., Natick, Massachusetts, 2015). Available at http://www.mathworks.com (Online)
24. S.P. Teeuwsen, *Oscillatory Stability Assessment of Power Systems Using Computational Intelligence* (Ph.d. dissertation, University Duisburg-Essen, 2005)