Dynamic System Equivalents using Integrated PSS/E and Python for Transient Stability Studies

by

Rong Guo

A thesis submitted to the Faculty of Graduate Studies of

The University of Manitoba

in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

University of Manitoba

Winnipeg, Manitoba

Abstract

Transient stability studies are required to be carried out for an efficient and secure operation of power systems. However, due to the limitations of computer memory and processing speed, handling a complete set of DAEs (differential-algebraic system of equations) that describe a large scale interconnected power system is difficult and uneconomical. Hence, the transient stability of large power system is generally studied by dividing the system into study and external areas, and the external areas are replaced with a dynamic equivalent circuit to reduce the calculation time. The research on dynamic equivalent circuit is still worth exploring. Therefore, this thesis attempts to propose an approach to obtain a dynamic equivalent circuit for the external system.

In the proposed method, a dynamic equivalent circuit is obtained by adding equivalent generators to boundary buses of a static equivalent circuit. The static equivalent circuit of the external system can be constructed utilizing the static network reduction features available in PSS/E. Coherent generator groups within the external system are identified using the non-linear time domain simulation combined with Prony analysis both available in PSS/E. If a complete set of dynamic parameters are available, the parameters of equivalent models are calculated by aggregation methods. If not, the optimization techniques based on minimizing the cost function are utilized to determine the model parameters of equivalent machines, where the cost function is defined as the sum of squares of the difference between equivalent system transient voltage results and the full system transient voltage results. The proposed method is validated with the

New York and New England IEEE 68-bus system. The simulation has shown that the developed equivalent system is good at mimicking the dynamic features of the original system.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my academic advisor Prof. U. D. Annakkage for his support, guidance, advice and patience throughout my graduate studies. As my supervisor, what he has taught me not only academic and professional knowledge but also attitude to life. It is a great privilege to study and work under his supervision.

I would like to thank Mitacs Accelerate and Manitoba HVDC Research Centre for financially supporting my research and for offering the internship opportunity, which provided me a valuable work experience. A special thanks to Dr. Dharshana Muthumuni for his suggestions and assistances during my internship in Manitoba HVDC Research Centre.

I like to thank Prof. Aniruddha Gole, Prof. Shaahin Filizadeh, Prof. Athula Rajapakse and Prof. Behzad Kordi for the invaluable knowledge they shared during the courses.

I would like to thank my research team for sharing experience, interesting discussions we had and their encouragement. It is my pleasure to have you all as my colleagues and friends. I would like to thank all my friends in Winnipeg for their support during these years.

Last but not least, my deepest gratitude goes to my loving parents, for their unconditional love, support and understanding. Without them, I would not have been the person that I am today.

Dedication

To everyone who works hard to make the world better

Table of Contents

Abstract ..... **ii**

Acknowledgements ..... **iv**

Dedication ..... **v**

List of Figures ..... **ix**

List of Tables ..... **xi**

Chapter 1 Introduction **1**

1.1 Background and Motivation 1

1.2 Research Objectives 3

1.3 Thesis Organization 4

Chapter 2 Developing a Static Equivalent Circuit for External System **6**

2.1 Internal System, External System, and Boundary Buses 6

2.2 Change of Bus Type Code 7

2.3 Generation Netting 8

2.4 Static Reduction 9

2.4.1 Method Used in Static Reduction 9

2.4.2 Static Reduction in PSS/E 11

2.5 Equivalencing IEEE 68-Bus System 12

2.6 Summary 20

Chapter 3 Coherency Identification, Generators and Exciters Aggregation **21**

3.1 Coherency Identification 21

3.1.1 Non-linear Time Domain Simulation 21
3.1.2 Non-linear Time Domain Simulation Results 22
3.2 Prony Analysis 26
3.3 Generator Aggregation 29
3.3.1 Zhukov's Method 29
3.3.2 Aggregated Detailed Generator Models Provided by DYNRED 30
3.4 Exciter Aggregation 32
3.5 Calculation Results 33
3.6 Summary 35
Chapter 4 Dynamic Equivalent Circuit for External System 37
4.1 Proposed Methodology 37
4.1.1 Function-1: Dynamic Simulation Setup 42
4.1.2 Function-2: Dynamic Simulation Procedures 44
4.1.3 Function-3: Data Evaluation 46
4.1.3.1 Monte Carlo Method 46
4.1.3.2 Nelder-Mead Method 47
4.1.3.3 Powell Method 48
4.1.4 Function-4: Plotting 49
4.2 Results and Discussion 50
4.2.2 Validation of the Monte Carlo Method 51
4.2.3 Validation of Nelder-Mead Method 53
4.2.4 Validation of Powell Method 56

4.3 Summary 57

Chapter 5 Conclusions and Future Work 59

5.1 Conclusions 59

5.2 Contributions 61

5.3 Future Work 61

References 63

Appendix A Test System Data 68

Appendix B Prony Analysis Result 70

Appendix C Python Code 73

C.1 Mento Carlo Method 73

C.2 Nelder-Mead Method 86

C.3 Powell Method 97

List of Figures

Figure 2-1. Interconnected power system 7

Figure 2-2. Static equivalent circuit of external system 11

Figure 2-3. Operation of activity EEQV 12

Figure 2-4. Single-line diagram of IEEE 68-bus power system 13

Figure 2-5. Redrawn single-line diagram of reduced IEEE 68-bus system 15

Figure 3-1. Swing curves for all generators in external system 23

Figure 3-2. Swing curves for generators 10 and 11 24

Figure 3-3. Swing curves for generators 12 and 13 24

Figure 3-4. Swing curves for generators 14, 15, and 16 25

Figure 4-1. Process of determining dynamic model parameters 39

Figure 4-2. Interface between PSS/E and Python 40

Figure 4-3. Flow chart of the fitting process 41

Figure 4-4. Flow chart of Function-1: Dynamic simulation setup 43

Figure 4-5. Flow chart of Function-2: Dynamic simulation procedures 45

Figure 4-6. Flow chart of Fuction-4: Plotting 49

Figure 4-7. Single-line diagram of reduced 68-bus system with external system represented by

dynamic equivalent circuit 51

Figure 4-8. Best five cases obtained by Monte Carlo method 52

Figure 4-9. Worst five cases obtained by Monte Carlo method 53

Figure 4-10. Comparison between the good starting scenario and its corresponding optimal scenario using Nelder-Mead method 55

Figure 4-11. Comparison between the bad starting scenario and its corresponding optimal scenario using Nelder-Mead method 55

Figure 4-12. Comparison between the good starting scenario and its corresponding optimal scenario using Powell method 56

Figure 4-13. Comparison between the bad starting scenario and its corresponding optimal scenario using Powell method 57

Figure B. 1. Prony analysis results 72

List of Tables

Table 2-1. Bus Type Codes 8

Table 2-2. Comparison between Original and Static Equivalent System 15

Table 2-3. Comparison of Bus Voltages at Retained Buses 16

Table 2-4. Comparison of Power Flows at Retained Lines 17

Table 2-5. Comparison of Fault Levels at Retained Buses 18

Table 3-1. Results from Prony Analysis of Fig.3-5 Results – 5 to 8 sec Tie Window 27

Table 3-2. Estimated Parameters of Equivalent Generators Eigenvector 34

Table 3-3. Estimated Parameters of Equivalent Exciter 35

Table A. 1. Generator Dynamic Data on 100 MVA Base 68

Table A. 2. IEEE Type AC4 Excitation System Data 69

Chapter 1 Introduction

In this chapter, the background and motivation of this research are discussed. The research objectives and the outline of the thesis are listed at the end of this chapter.

1.1 Background and Motivation

Transient stability studies are required to be carried out by the utilities for successful planning, operation and control, and post-disturbance analysis of large interconnected power systems [1]. The complete power system model for transient stability analysis can be mathematically described by a very large set of differential equations modelling generation stations which are coupled by the algebraic equations describing the transmission network and loads [2]. With the increasing scale of power systems, handling a complete set of differential and algebraic equations that describe the entire interconnected system becomes difficult and uneconomical due to its huge computation burden. Therefore, it is necessary to divide the interconnected power system into a study system and one or more external systems. In transient studies, the study system is of specific interest and represented by a detailed model. In order to investigate the influence of the external system on the study system, it is necessary to generate a dynamic equivalent circuit for the external system, which can be achieved by reducing the number of generators and network nodes in the external system. The development of dynamic equivalent circuits has existed for decades, and there are three main approaches reported

in literature:

1. Model methods: describe the external system by an approximate linear model [3]-[5].
2. Coherency methods: identify coherent groups of generators and then replace the generators in coherent groups with equivalent generators [6]-[9].
3. Measurement or simulation based methods: obtain the external system response to applied disturbance by measurements or simulations, and identify model parameters through curve fitting techniques [10]-[13].

Due to the fact that the linear state equations of the equivalent model cannot reflect the characteristics of real physical power system components, the dynamic equivalent circuit generated by the Model methods cannot be directly used for transient stability studies. In comparison, Coherency methods and measurement or simulation based methods can be directly used for transient stability studies. However, the Coherency methods are proposed on the condition that the structure and parameters of the external system are available, which cannot always be satisfied, especially the dynamic parameters of the generators.

This project aims to develop a generalized method to obtain a dynamic equivalent circuit for external power system. Specifically, there are three motivations that are taken into account. First, to improve the accuracy of dynamic equivalent circuits, the simulation

based method is employed to reduce the errors resulted by coherency identification and aggregation methods. Second, when the parameters of external generators are not available, the simulation based method can be applied to this case to determine equivalent dynamic parameters. Finally, as the development of a good dynamic equivalent circuit is a time-consuming task, this thesis thus provides a much easier way to do this using integrated commercial software PSS/E and Python.

1.2 Research Objectives

The objective of this research is to develop a dynamic equivalent circuit to replace the external system, which is based on combining the simulation based method with the coherency method. In detail, the proposed scheme can be achieved following the steps given below.

- Obtain a static equivalent network for the external system
- Identify coherency among generators in the external system and aggregate the generators facilitated with their exciters in a coherent group to an equivalent generator and an equivalent exciter.
- Split the equivalent active power and reactive power in the static equivalent circuit into a generator and a load at a boundary bus.
- Capture the effect of external system on the dynamic performance of internal system by equivalent generator and associated controls.

- Tune basic parameters of the fictitious equivalent generator based on optimization algorithms so that the voltage recovery of the dynamic equivalent system can match well with that of the full system.
- Evaluate the proposed methods with New York and New England IEEE 68-bus system.

1.3 Thesis Organization

The rest of this thesis is organized as follows.

Chapter 2 introduces how to generate a static equivalent circuit for the external system in software package PSS/E, which is validated with New York and New England IEEE 68-bus system.

Chapter 3 presents the non-linear time domain simulation and Prony analysis that are available in PSS/E to identify coherency among generators. After the coherent generators are identified, Zhukov's [15],[16] and DYNEQU's [26] methods for aggregating generators in a coherent group are introduced. In addition, a simple strategy for computing aggregate exciter parameters is presented. Aggregation results of the generator and exciter aggregation of the IEEE 68-bus system are provided.

Chapter 4 proposes an optimization based fitting process for determining parameters of equivalent models. Particular attention will be given to how to construct a mathematical model to describe this problem and how to realize this in Python. Monte

Carlo, Nelder-Mead and Powell optimization algorithms are used to find optimal parameters via the fitting process. Moreover, the proposed approach is validated with IEEE 68-bus system.

Chapter 5 summarizes our main contributions in this thesis. In addition, some suggestions on future work are given.

Chapter 2 Developing a Static Equivalent Circuit for External System

In this chapter, the procedure of generating a static equivalent circuit to replace the external system is presented, which is based on the commercially available power system simulation software PSS/E. A static equivalent circuit for external system is constructed in PSS/E by the following procedures:

1. Distinguishing internal and external areas.
2. Changing bus type code.
3. Netting generation with loads.
4. Performing static reduction.

2.1 Internal System, External System, and Boundary Buses

A common practical approach to handle a large scale power system is to split it into two subsystems: the internal system and external systems. The internal system is also known as the study area in which transient studies are performed. Any system outside the internal system is known as external system. To generate a static equivalent circuit for the external system, the internal system remains unchanged while the external system is eliminated by static equivalent techniques. Boundary buses refer to those buses from which branches

connecting internal and external areas, which are required to be retained during the equivalence process. The relation among internal, external, and boundary buses are illustrated in Figure 2-1.

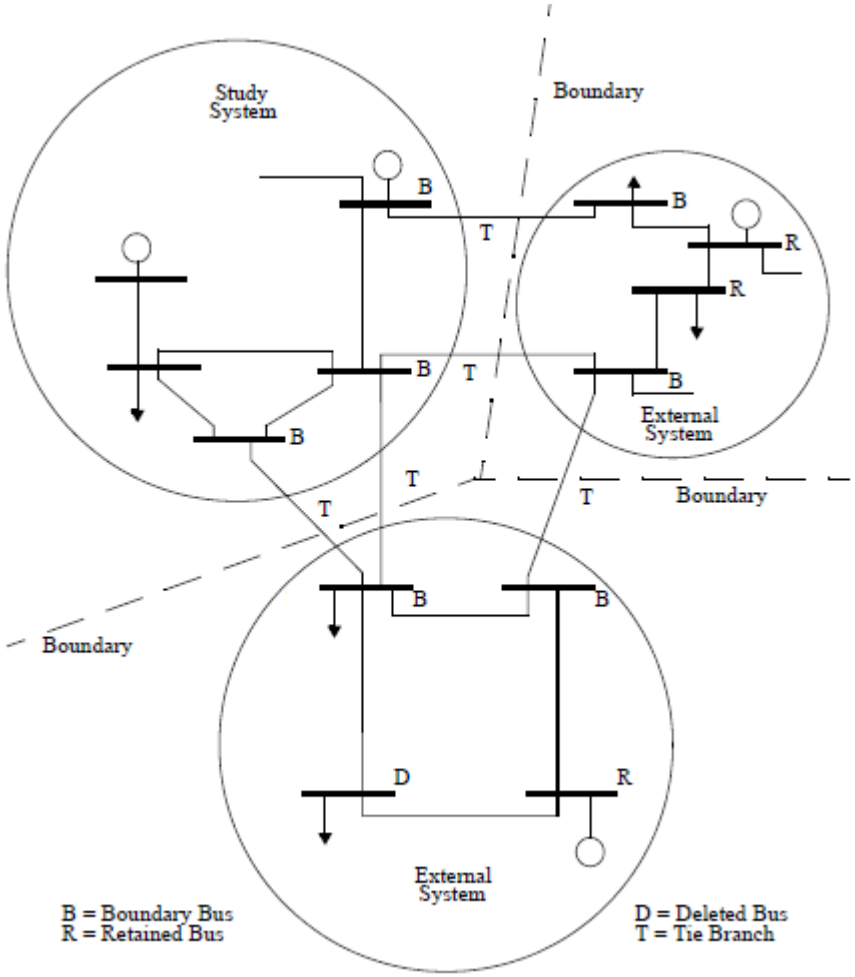

Figure 2-1. Interconnected power system

2.2 Change of Bus Type Code

PSS/E uses bus codes 1 to 7 to distinguish different types of buses [4]. This is briefly described in Table 2-1. Bus codes 1, 2, 3, and 4 are normal bus type codes, referring to

load bus, generator bus, swing bus, and off-line bus, respectively. Type codes 5, 6, 7 are special bus type codes only used for the equivalent construction. Bus type code 5 is assigned to the load bus, which is needed to be retained during the equivalencing process. In addition, the generator bus is retained by the equivalent construction process by changing bus type code 2 to 6. Bus type code 7 is used to preserve the slack bus that is not deleted by equivalencing. All bus type codes can be obtained and changed by accessing the *Bus* tab. When the equivalence process is complete, all special type codes are returned back to 1, 2, and 3, respectively.

Table 2-1. Bus Type Codes

Normal bus type	Description	Corresponding special bus type	Description
Type 1	PQ bus	Type 5	Retained PQ bus during the equivalent construction process
Type 2	PV bus	Type 6	Retained PV bus during the equivalent construction process
Type 3	Swing bus	Type 7	Retained swing bus during the equivalent construction process
Type 4	Inactive bus		

2.3 Generation Netting

Before reducing the network, the generation netting activity (NETG/GNET) [14] is used to eliminate large generation. The activity NETG/GENT removes a large number of generators by changing the in-service generation to an equivalent negative load. The

generators to be replaced by NETG/GNET may be specified by the area, owner, zone, base kV, and bus number options. Activity NETG/GNET processes all selected generators and swing buses (type code 2 or 3, respectively) except those designated as retained generators indicated by type codes 6 or 7. The following steps are involved in this netting process.

1. Convert bus type code 2 or 3 to 1.
2. Introduce a new load at the respective bus.
3. Set PL of the new load to –PG.
4. Set QL of the new load to –QG.

In short, activity NETG/GNET deletes the generation at a single bus with no other effects on the system by changing bus type code to 1. This bus will be further eliminated by implementing static network reduction. To distinguish equivalent loads from original power system loads, the equivalent loads are assigned the identifier 99.

2.4 Static Reduction

2.4.1 Method Used in Static Reduction

A static equivalent circuit of external system is constructed by performing a static reduction [14]-[17]. Static reduction involves eliminating the external buses and branches to reduce the scale and complexity of the external system, which can be achieved by employing the Gaussian elimination method to reduce the admittance matrix of the

external system. The admittance matrix of the external system is given by

$$\begin{bmatrix} I_R \\ I_D \end{bmatrix} = \begin{bmatrix} Y_{RR} & Y_{RD} \\ Y_{DR} & Y_{DD} \end{bmatrix} \begin{bmatrix} V_R \\ V_D \end{bmatrix} \quad (2.1)$$

where the subscripts R and D denote nodes in the external system to be retained and deleted, respectively. Hence, I_R and V_R represent node current and voltage at the nodes to be retained, and I_D and V_D are node current and voltage at the nodes to be deleted.

The desired form of an equivalent circuit only contains I_R and V_R, with variables I_D and V_D assumed to be linearly dependent on variables I_R and V_R. The second row of (2.1) is rearranged as:

$$V_D = Y_{DD}^{-1}(I_D - Y_{DR}V_R) \quad (2.2)$$

Substituting (2.2) into the first row of (2.1), the current equation of I_R can be written as

$$I_R = (Y_{RR} - Y_{RD}Y_{DD}^{-1}Y_{DR})V_R + Y_{RD}Y_{DD}^{-1}I_D \quad (2.3)$$

Due to the fact that all boundary buses are retained [18], the first term of (3) can be regarded as new equivalent transmission lines between the boundary buses and static shunts connecting the boundary buses. The second term represents a set of equivalent currents injected at boundary buses. These equivalent currents reproduce the effect of load currents at the deleted nodes, which can be transformed to equivalent constant real and reactive power loads at boundary buses. The static equivalent circuit is shown in Figure 2-2.

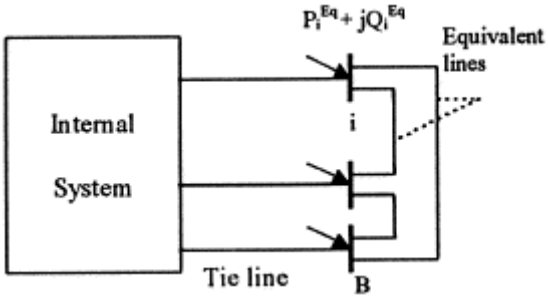

Figure 2-2. Static equivalent circuit of external system

2.4.2 Static Reduction in PSS/E

A static equivalent circuit obtained by the method discussed above can be implemented by activity EEQV in PSS/E [14]. This activity starts with a solved power flow model containing external buses and boundary buses, and leaves the resulting equivalent circuit in the working file. Activity EEQV effectively accomplishes the following tasks of constructing an electrical equivalent in a single execution:

1. Isolate the external system from the study area.
2. Build an electrical equivalent circuit for the external system.
3. Attach the equivalent circuit to the study system.

The equivalent circuit generated by activity EEQV represents the original network model with a reduced network model in the working file consisting of a unequivalenced internal system and an equivalenced external system (see Figure 2-3). In the equivalence process, activity EEQV performs all mathematical operations involved in the admittance matrix reduction automatically. All type 1 buses in the external system are eliminated, whereas boundary buses of the external system are automatically retained. Buses that are

indicated by type code 4 are not included in the equivalent circuit so that the relevant computation can be ignored. If the bus type code is 5, 6, or 7, activity EEQV will not delete it but change the type code back to 1, 2, or 3, respectively. When activity EEQV introduces equivalent branches, loads, and bus shunts to retained boundary buses, these new electrical elements are assigned with identifier 99.

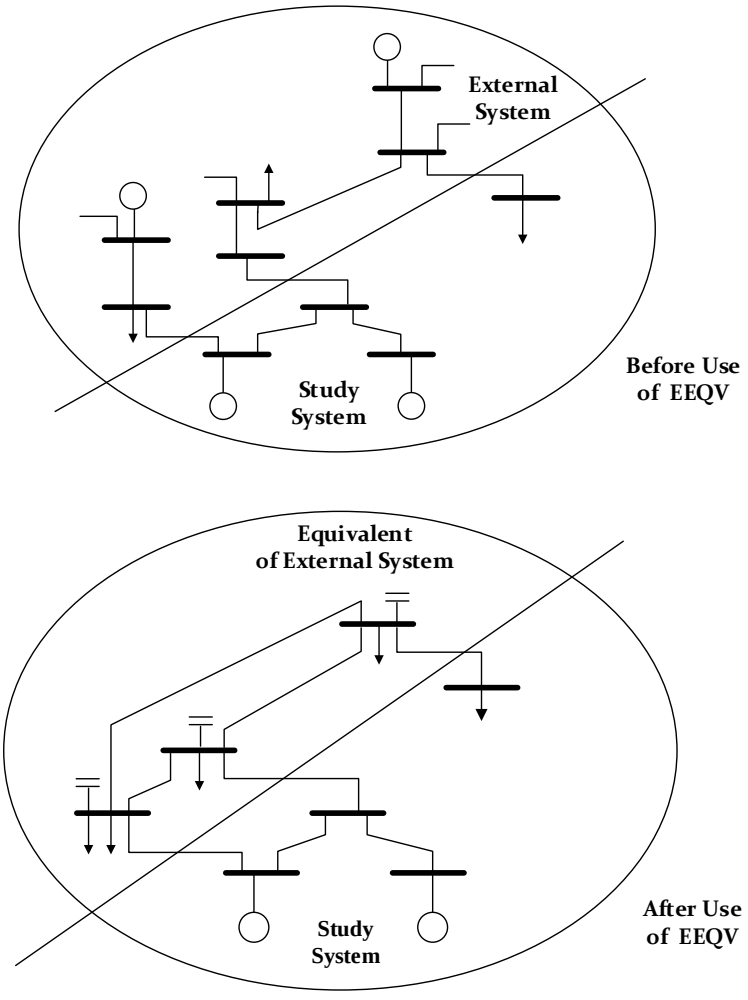

Figure 2-3. Operation of activity EEQV

2.5 Equivalencing IEEE 68-Bus System

Consider the IEEE 68-bus system shown in Figure 2-4, which is a reduced order

equivalent of the inter-connected New England test system (NETS) and New York power system (NYPS) [19]. This system contains five areas shown to be bounded by thin dashed lines in Figure 2-4. Now assume that Area-1 is interested in carrying out security analysis, while the remaining portion, including Area-2, 3, 4, 5, can be regarded as the external area. As can be seen from Figure 2-4, thick dashed lines separate the 68-bus system into two subsystems. The power model of Subsystem 2, which is of particular interest, is entirely retained. Subsystem 1 that will be replaced by its equivalent network has two boundary buses that connect the internal system through six tie lines, as shown in Figure 2-4. It is expected that only the boundary buses 53 and 61 are retained as the result of creating an equivalent circuit of the external area, and all of other buses will be removed in Subsystem 1.

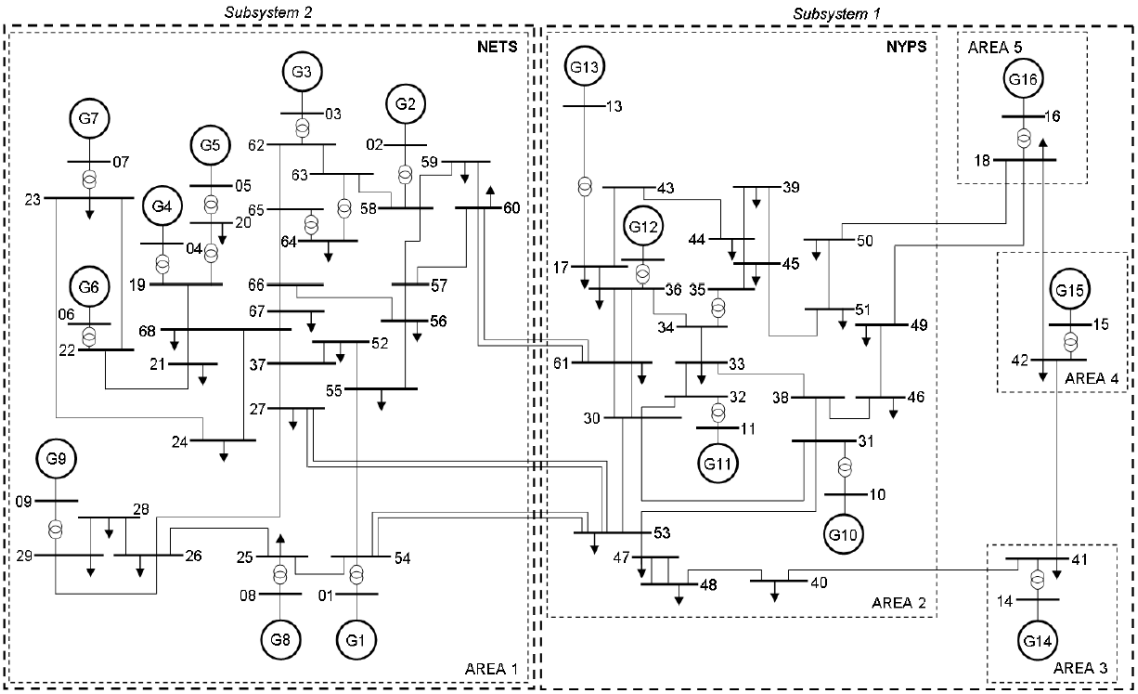

Figure 2-4. Single-line diagram of IEEE 68-bus power system

Since there are no other boundary buses (except buses 53 and 61) required to be retained after the equivalence process, there is no need to change any bus type codes. The generation netting with loads was implemented in Subsystem 1. Generators 10-16 in the external system were replaced by corresponding negative loads at this stage, and all bus type codes of these 7 generators were converted from 2 to 1 after performing NETG/GNET.

Subsequently, the network reduction was performed in the external system. After building an electrical equivalent circuit of the external system, the redrawn single line diagram of the IEEE 68-bus system with new branches, loads, and shunt are shown in Figure 2-5. As compared to Figure 2-4, it is apparent that there are some changes in topology due to the effect of network reduction. Table 2-2 presents a comparison between the original system and the reduced system. It is seen that Area 2, 3, 4, 5 were removed from the original system, the total number of buses was reduced to 38, and the number of branches decreased from 69 to 38. The equivalence process removed all generators and transformers in the external system. The boundary buses of the external system were retained so that loads at boundary buses 53 and 61 were retained in the reduced system. There were one equivalent fixed shunt and one equivalent load attached to boundary buses 53 and 61 respectively in the external area, each with a circuit identifier of 99.

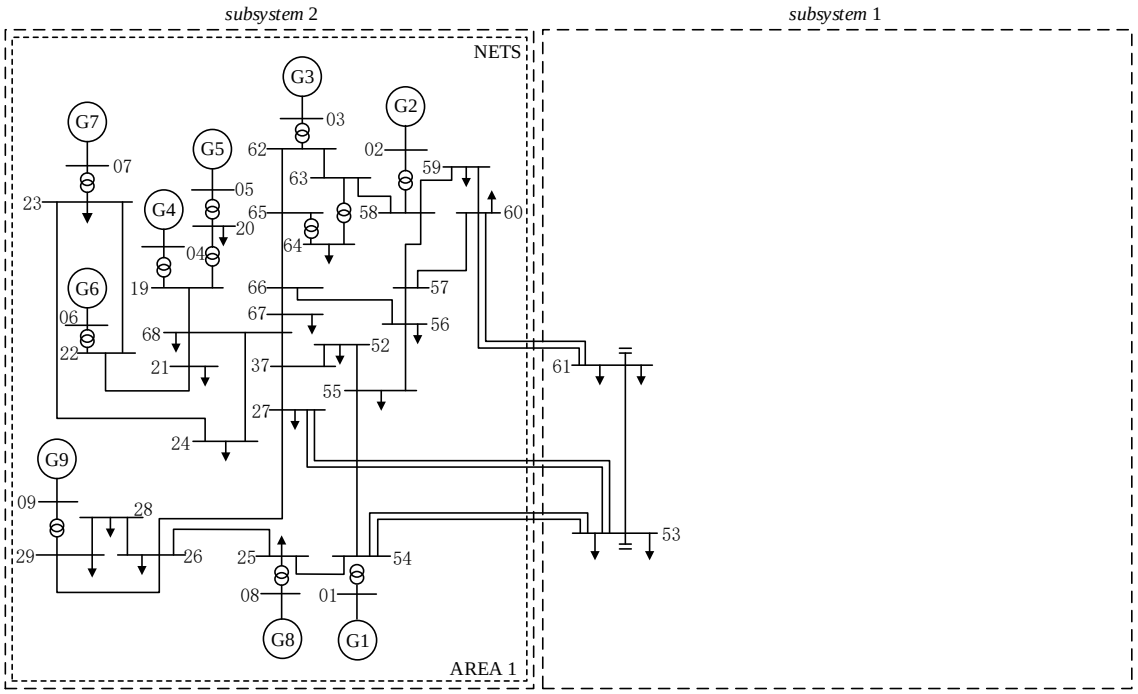

Figure 2-5. Redrawn single-line diagram of reduced IEEE 68-bus system

Table 2-2. Comparison between Original and Static Equivalent System

No.	Original system	Static equivalent system
Areas	5	1
Buses	68	38
Branches	69	38
Total generators	16	9
Generators in external system	7	0
Total transformers	19	11
Transformers in external system	8	0
Total loads	36	22
Loads in external system	18	4
Fixed shunts	0	2

To validate the effectiveness of the network equivalence scheme used in this chapter, bus voltages and power flow results of the original system and the reduced system are compared as below. It is seen from Table 2-3, the voltage values and angles at retained buses have no change before and after static equivalencing. Table 2-4 illustrates that the active and reactive power through retained lines in the equivalent circuit are the same as those in the original system. For simplicity, only few bus voltages and power flows are presented below to show the correctness of steady-state simulation results.

Table 2-3. Comparison of Bus Voltages at Retained Buses

Bus no.	Original system		Static equivalent system	
	Voltage (p.u.)	Angle (deg)	Voltage (p.u.)	Angle (deg)
1	1.045	10.54	1.045	10.54
6	1.05	20.44	1.05	20.44
22	1.0462	15.08	1.0462	15.08
37	1.0267	5.81	1.0267	5.81
53	1.0244	8.72	1.0244	8.72
61	1.0155	4.61	1.0155	4.61
64	0.9993	10.71	0.9993	10.71

Table 2-4. Comparison of Power Flows at Retained Lines

From bus no.	To bus no.	Original system		Static equivalent system	
		P_{line} (MW)	Q_{line} (MVAR)	P_{line} (MW)	Q_{line} (MVAR)
60	61	118.4	-35.6	118.4	-35.6
27	53	-32.4	-14.6	-32.4	-14.6
27	26	83.7	-88.9	83.7	-88.9
54	53	-24.5	5.8	-24.5	5.8
23	24	371.3	14.6	371.3	14.6
63	58	300.7	-31.7	300.7	-31.7
57	56	163.4	-27.3	163.4	-27.3

Fault level is another critical metric for evaluation after network reduction. Fault level is also known as short circuit capacity, which is used to measure the ability of an equipment or system to sustain under the worst short circuit condition. There are four main types of short circuit faults: 1) single line to earth; 2) line to line; 3) line to line to ground; and 4) three phase. Among these types, a three phase fault leads to the maximum fault current. The fault level of the bus in power system is defined as the product of the pre-fault voltage and the fault current of the bus. As the voltage at a bus prior to fault is close to nominal value 1 p.u., the maximum short circuit current is used to describe the fault level, which is the magnitude of three phase fault current [20]. Three phase fault current is accessible by PSS/E activity ASCC in a single execution [14]. By applying ASCC to both original and static equivalent systems, the fault levels at retained buses

before and after static equivalencing are compared, as shown in Table 2-5. It can be seen that the static equivalent model of the external system resulted in reduced fault levels at the buses in the internal subsystem as expected. Moreover, the fault levels can be improved by adding equivalent generators at boundary buses, which will be discussed in the Chapter 4.

Table 2-5. Comparison of Fault Levels at Retained Buses

Bus no.	Original system (p.u.)	Static equivalent system (p.u.)
1	74.4	69.53
2	45.11	42.2
3	48.25	45.77
4	57.49	57.01
5	38.84	38.67
6	54.45	53.93
7	47.13	46.88
8	46.5	44.51
9	37.58	37.1
19	62.18	60.91
20	42.98	42.57
21	58.12	56.14
22	61.47	60.13
23	57.71	56.56
24	64.15	60.89

25	70.16	60.92
26	51.44	47.65
27	53.74	49.11
28	30	29.26
29	34.85	34.06
37	75.22	67.83
52	64.94	58.06
53	97.6	45.17
54	94.46	72.61
55	75.07	62.94
56	70.28	57.79
57	76.23	57.83
58	77.46	59.19
59	66.78	49.07
60	73.95	50.57
61	119.37	41.86
62	65.37	55.63
63	65.81	54.77
64	27.54	25.63
65	61.96	53.28
66	66.08	56.83
67	63.21	58.39
68	86.19	79.61

2.6 Summary

This chapter presented how to develop a static equivalent circuit for the external system by using commercially available power system simulation software PSS/E, and New England & New York 68-bus system was used to demonstrate the effectiveness of the proposed scheme.

First, the power system was first divided into a study area and an external area. The study area was then kept unchanged, and then activity NETG/GNET is applied in the external system to net the generation with load. Second, activity EEQV was used to reduce the external network based on the impedance reduction process.

To validate the performance of the static equivalent circuit, bus voltages, power transfer and faults levels of the reduced system were compared with that of the original system. The comparison results have shown that the static equivalent circuit developed can replace the external system.

Chapter 3 Coherency Identification, Generators and Exciters Aggregation

Previous chapter presented how to reduce the scale and complexity of external system in PSS/E. However, the generated static equivalent circuit cannot reflect the dynamic features of external system. Therefore, it is necessary to reduce the number of synchronous generators in the external area while retaining their influence on the internal area, which can be achieved by aggregation of coherent generators. This thesis determines the coherent generators using non-linear time domain simulation, making use of the great capability of PSS/E in performing dynamic simulation of large scale power system. In addition, Prony analysis is employed to further group coherent generators. After identifying coherent generators, generators and their exciters in each coherent group are aggregated to form an equivalent generator and an equivalent exciter in each coherent group.

3.1 Coherency Identification

Section 3.1.1 introduces how to use non-linear time domain simulation to identify coherent generators. Section 3.1.2 shows simulation results of coherency identification that is carried out in the IEEE 68-bus system using the non-linear time domain simulation.

3.1.1 Non-linear Time Domain Simulation

Coherent generators are identified by means of non-linear time domain simulation. The

coherency identification technique is based on the comparison of time domain responses when the power system is disturbed. In this method, a three phase fault is applied in the study area and the rotor angles of generators in the external system are compared. Generators are defined as coherent if relative differences between their rotor angles remain constant within a specified tolerance over a certain period of time following a disturbance [21], which can be mathematically expressed as:

$$max\left|\Delta\theta_i(t) - \Delta\theta_j(t)\right| < \xi \tag{3.1}$$

where $\Delta\theta_i(t) = \theta_i(t) - \theta_i(0)$ and $\Delta\theta_j(t) = \theta_j(t) - \theta_j(0)$ are the angular deviations at time instant t for generator i and j. The absolute difference of rotor angles deviation between two coherent generators less than a specified tolerance ξ for the entire simulation.

In this research, dynamic responses of generators to the fault are simulated by PSS/E to identify coherent generators. Time domain responses of generators in the external area are monitored, and those generators with rotor angles swinging together are considered to be coherent.

3.1.2 Non-linear Time Domain Simulation Results

The adopted coherent method was applied to the New England & New York 68-bus case. In this case, a three phase fault at bus 55 is applied at 0.1 s and cleared at 0.15 s. The rotor angle plots obtained from the PSS/E of generators 10-16 with respect to the generator at bus 10 is given in Figure 3-1. For the purpose of clarity, the generator swing curves are

classified into different figures depending on the curve similarity, as shown in Figs. 3-2-3-4.

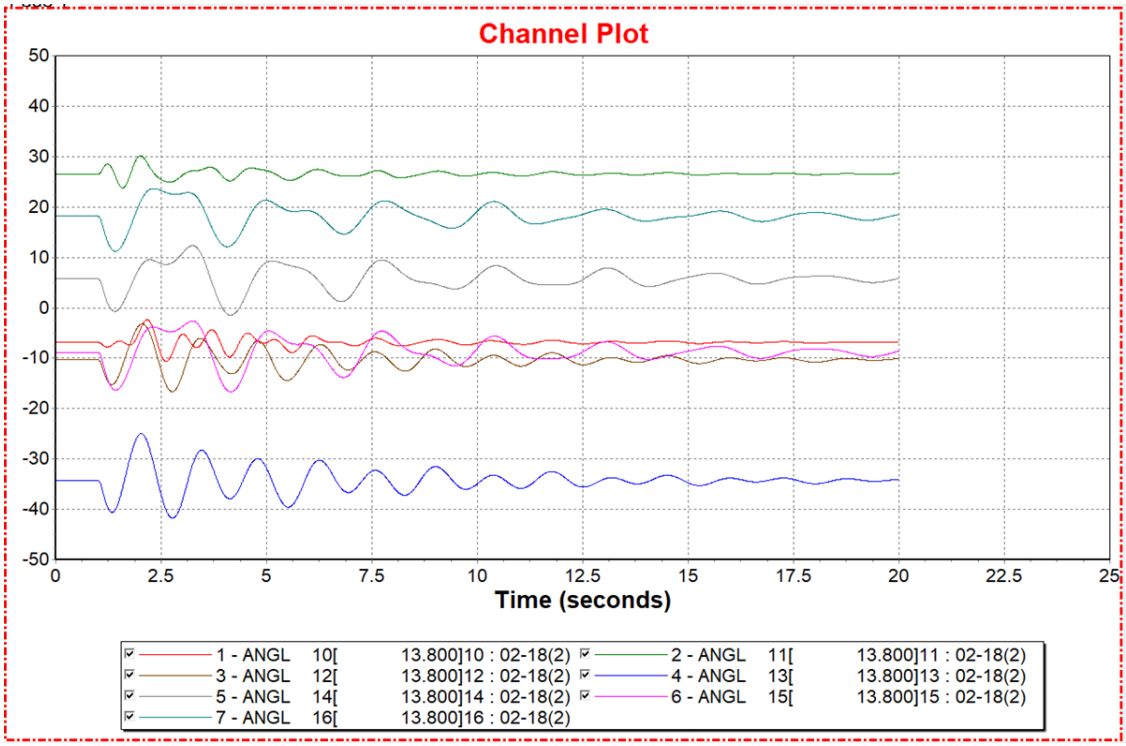

Figure 3-1. Swing curves for all generators in external system

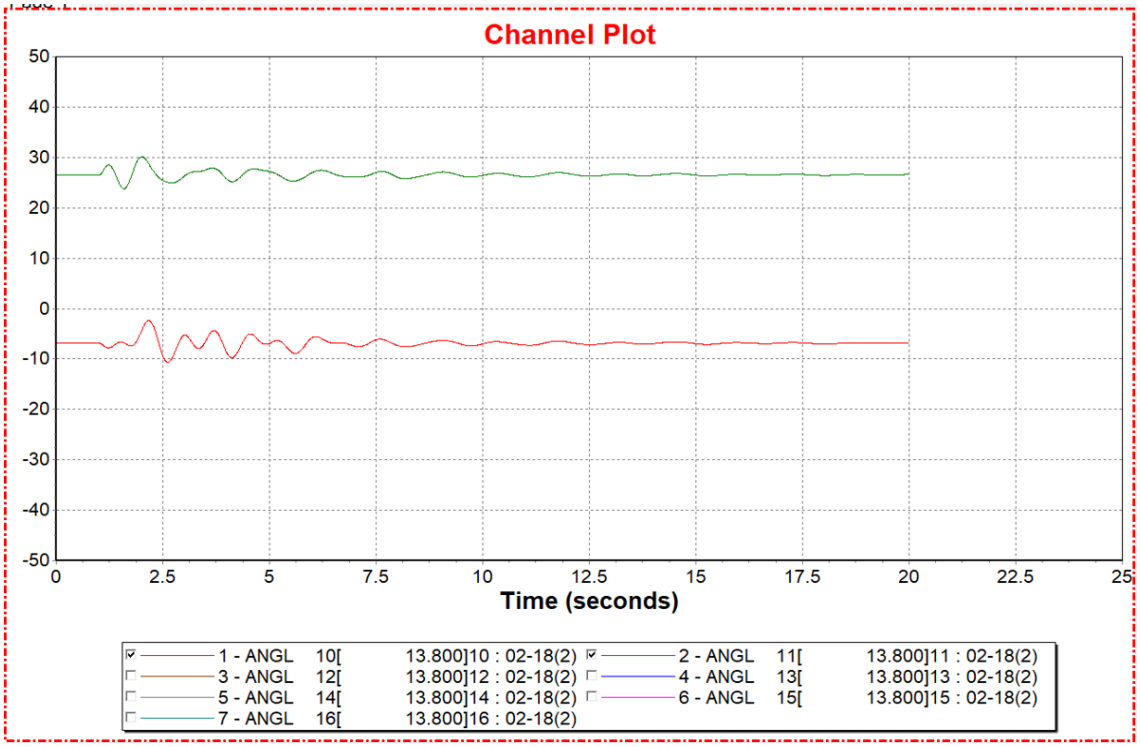

Figure 3-2. Swing curves for generators 10 and 11

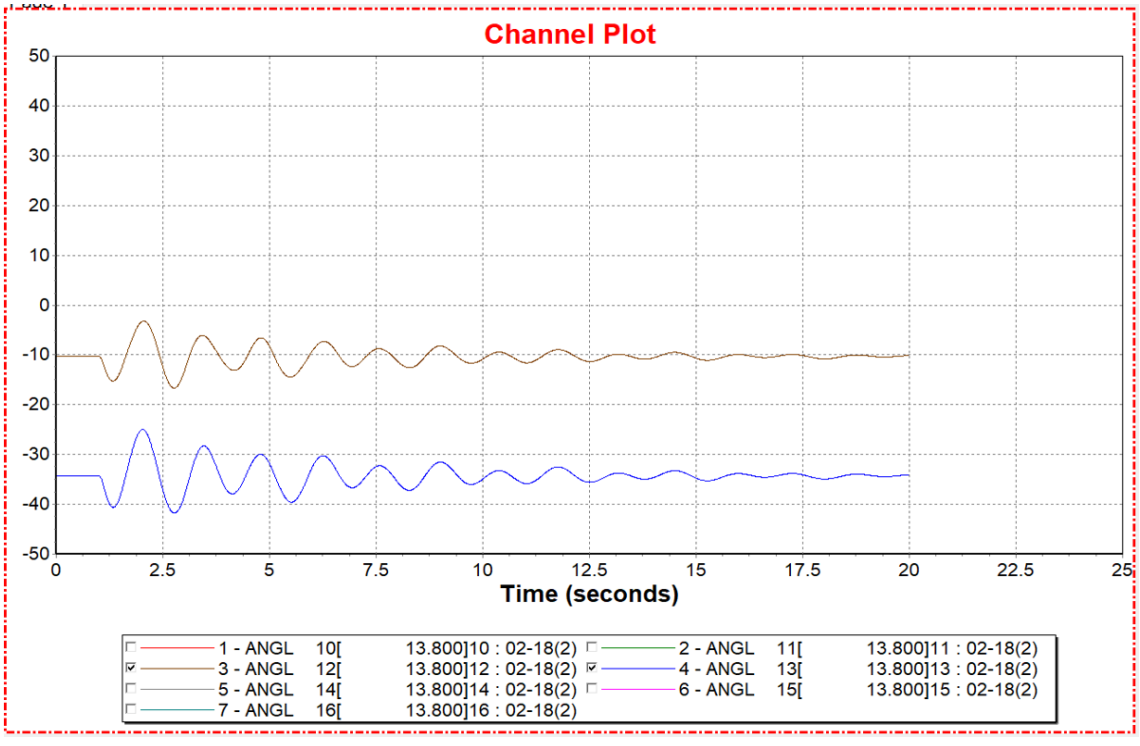

Figure 3-3. Swing curves for generators 12 and 13

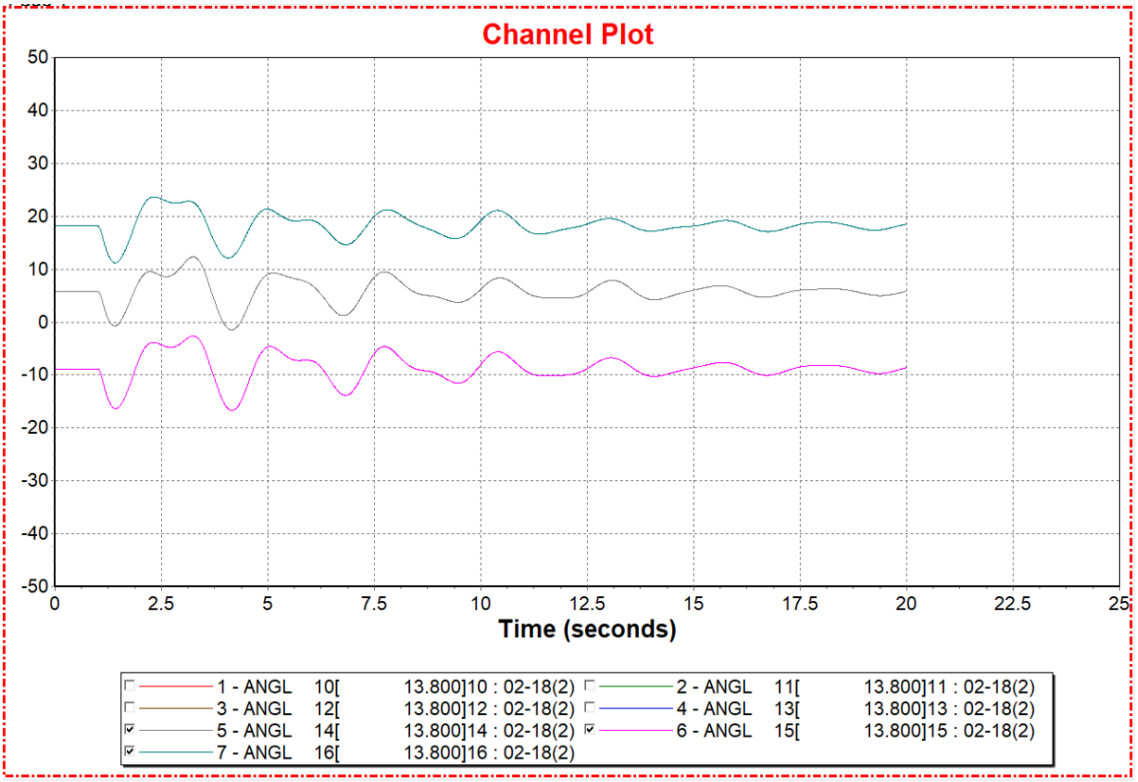

Figure 3-4. Swing curves for generators 14, 15, and 16

Dynamic simulation results showed that generators in the external system can be divided into three coherent groups. As shown in Figure 3-2, it is seen that generators 10 and 11 swinging together are coherent, Generators 12 and 13 are coherent due to the closeness of their rotor angle variations, as shown in Figure 3-3. Figure 3-4 depicts that generators 14, 15, and 16 have the most similar swing curves; therefore, they are considered as a coherent group. The results of coherent generators identification using the non-linear time domain simulation are similar to the results in [22] where the two-time scale method was used.

By using an equivalent generator to replace generators in a coherent group, three equivalent generators are obtained at this stage to represent the dynamic behaviors of

external system. However, it is noted that there are only two boundary buses (i.e., buses 53 and 61) retaining in the external system, limiting the number of equivalent generators to two. Therefore, in the next section, Prony analysis is employed to divide generators in the external system into two coherent groups.

3.2 Prony Analysis

Prony analysis is considered as an extension of Fourier analysis. It directly decomposes a given signal into its dominant modes of oscillation in terms of amplitude, frequency, damping, and relative phases [23]. In [24], the basic principle of Prony analysis is presented. In this research, Prony analysis is used for calculating complex eigenvalues and associated eigenvector components, which is valuable in identifying modal interaction mechanisms. Prony analysis is available in the PSS/E software, and one can find relevant instructions on how to use the Prony technique in [25].

The time window 5 to 8 sec in Figure 3-5 were selected for Prony analysis. The first 5 second traces were ignored, allowing well-damped modes to die out and the system to regain linearity. The analysis results using Prony techniques in the PSS/E are presented in Appendix B, and dominant components of each generator are listed in Table 3-1.

Table 3-1. Results from Prony Analysis of Fig.3-5 Results – 5 to 8 sec Tie Window

Generator no.	Comp no.	Magnitude	Angle (°)	Frequency (Hz)
10	1	29.499	--	--
	2	2.7432	-97.26	0.732
11	1	63.271	--	--
	2	2.4732	-84.37	0.710
12	1	25.474	--	--
	2	6.3019	-80.05	0.705
13	1	7.0352	-87.20	0.718
	2	2.0979	--	--
14	1	49.389	--	--
	2	14.838	113.49	0.292
	3	6.9844	-64.95	0.659
15	1	31.512	--	--
	2	11.864	96.44	0.332
	3	7.2445	-85.26	0.700
16	1	61.475	--	--
	2	12.585	115.17	0.286
	3	6.3555	-78.85	0.672

It can be seen from Table 3-1 that generators 10, 11, 12, and 13 have a roughly 0.7 Hz dominant component in addition to steady-state components. Meanwhile, except steady-state components, generators 14, 15 and 16 have an approximately 0.3 Hz dominant component and a 0.7 Hz component of about a half the size. Therefore, we

classify generators 10, 11, 12, and 13 as a coherent group, while generators 14, 15, and 16 as another coherent group.

It is observed from Figure 2-4. Single-line diagram of IEEE 68-bus power system that generator 14 is connected by several transmission links to boundary bus 53 and geographically far away from boundary bus 61. For this reason, the equivalent generator used to represent the coherent group containing generator 14 is attached to boundary bus 53. Therefore, the equivalent generator obtained by aggregating generators 10, 11, 12, and 13 connects to boundary bus 61.

3.3 Generator Aggregation

After coherent generators are identified, generators in each coherent group are aggregated into an equivalent generator that represents each individual coherent group. The parameters of dynamic equivalent models can be determined by two approaches, i.e. Zhukov's method [15], [16] and detailed aggregation method provided by DYNRED [26].

3.3.1 Zhukov's Method

Zhukov's method that has been reported in [15], [16] is used here. However, unlike the original Zhukov's method, the equivalent generator is directly connected to the boundary bus. For this reason, the terminal voltage of the equivalent generator is equal to the voltage of the boundary bus, where the equivalent generator is attached. The electrical active and reactive power outputs of the equivalent generator are scaled up to match the total electrical power of the coherent group, which is given by

$$P_{eq} + jQ_{eq} = \sum_{i=1}^{n} (P_i + jQ_i) \tag{3.2}$$

where P_{eq} and Q_{eq} are the real and reactive power of the equivalent generator, P_i and Q_i are real and reactive power of generator i, n denotes the number of coherent generators in the same coherent group. The swing equation of the rotor is given by

$$M_i \frac{d\omega_i}{dt} = P_{mi} - P_{ei} - D_i\omega_i \tag{3.3}$$

where M_i, P_{mi}, P_{ei}, D_i and ω_i denote respectively the generator inertia, mechanical power, electrical power, damping constants, and angular speed of generator i. Since

coherent generators have almost identical angular frequencies and thus assumed to be equal to ω, the swing equation of the equivalent generator is thus expressed as

$$\left(\sum_{i=1}^{n} M_i\right)\frac{d\omega}{dt} = \sum_{i=1}^{n} P_{mi} - \sum_{i=1}^{n} P_{ei} - \left(\sum_{i=1}^{n} D_i\right)\omega \tag{3.4}$$

The mechanical and electrical power of the equivalent generator is the sum of those of coherent generators, respectively. Therefore, inertia and damping constant of the equivalent generator are also defined as the sum of inertia and damping constant of all the generators in a coherent group, which take the form,

$$M_{eq} = \sum_{i=1}^{n} M_i \tag{3.5}$$

$$D_{eq} = \sum_{i=1}^{n} D_i \tag{3.6}$$

As all the generators in one coherent group are connected to the boundary bus, their transient, and sub-transient d- and q-axes reactance can be calculated by paralleling the reactance of all the individual generators in the same coherent group, which can be expressed as follows:

$$X_{eq} = \frac{1}{\sum_{i=1}^{n} X_i} \tag{3.7}$$

3.3.2 Aggregated Detailed Generator Models Provided by DYNRED

In this section, equations of the equivalent generator model are presented. The parameters of the GENROU dynamic model [27] are obtained using a weighted least square method

[26]. As mentioned earlier, since the equivalent generator has been moved to the boundary bus, the terminal voltage of the equivalent generator is equal to that of the boundary bus. The rating, inertia, and damping constants of the equivalent generator are the sum of the ratings, inertia, and damping constant of coherent generators being aggregated, respectively. The detailed aggregation method provided by DYNRED for calculating equivalent reactance can be summarized below. Under the generator initial conditions, the initial load angle is calculated by

$$\delta_i = \tan^{-1}\left(\frac{P}{\frac{E_t^2}{X_{qi}} + Q}\right) \tag{3.8}$$

The equivalent generator load angle is defined by

$$\delta_e = \frac{1}{2}\frac{\sum_{i=1}^{n}(Y_{qi} - Y_{di})\sin 2\delta_i}{\sum_{i=1}^{n}(Y_{qi} - Y_{di})\sin 2\delta_i} \tag{3.9}$$

where,

$$Y_{di} = \frac{1}{X_{di}} \tag{3.10}$$

$$Y_{qi} = \frac{1}{X_{qi}} \tag{3.11}$$

Thus, the equivalent operational impedances are given by

$$X_{de} = \frac{2}{\sum_{i=1}^{n}\left(\frac{1}{X_{di}} + \frac{1}{X_{qi}}\right) - \left(\frac{1}{X_{qi}} - \frac{1}{X_{di}}\right)\cos 2(\delta_i - \delta_e)} \tag{3.12}$$

$$X_{qe} = \frac{2}{\sum_{i=1}^{n}\left(\frac{1}{X_{di}} + \frac{1}{X_{qi}}\right) + \left(\frac{1}{X_{qi}} - \frac{1}{X_{di}}\right)\cos 2(\delta_i - \delta_e)} \tag{3.13}$$

$$X'_{de} = \frac{2}{\sum_{i=1}^{n}\left(\frac{1}{X'_{di}} + \frac{1}{X'_{qi}}\right) - \left(\frac{1}{X'_{qi}} - \frac{1}{X'_{di}}\right)\cos 2(\delta_i - \delta_e)} \tag{3.14}$$

$$X'_{qe} = \frac{2}{\sum_{i=1}^{n}\left(\frac{1}{X'_{di}} + \frac{1}{X'_{qi}}\right) + \left(\frac{1}{X'_{qi}} - \frac{1}{X'_{di}}\right)\cos 2(\delta_i - \delta_e)} \tag{3.15}$$

$$X''_{de} = \frac{2}{\sum_{i=1}^{n}\left(\frac{1}{X''_{di}} + \frac{1}{X''_{qi}}\right) - \left(\frac{1}{X''_{qi}} - \frac{1'}{X'_{di}}\right)\cos 2(\delta_i - \delta_e)} \tag{3.16}$$

$$X''_{qe} = \frac{2}{\sum_{i=1}^{n}\left(\frac{1}{X''_{di}} + \frac{1}{X''_{qi}}\right) + \left(\frac{1}{X''_{qi}} - \frac{1}{X''_{di}}\right)\cos 2(\delta_i - \delta_e)} \tag{3.17}$$

3.4 Exciter Aggregation

In the previous section, the identified coherent generators in a group are aggregated to an equivalent generator. The associated exciters are needed to be aggregated as well. The aggregation of exciter models in this research is based on the method described in [6]. In [6], a trajectory sensitivity method [28] was used to find the optimal parameters of the equivalent exciter model. It was reported that the equivalent regulator gain K_A has the greatest impact on improving the performance of the equivalent exciter. The value of K_A can be easily obtained by an MVA weighted average method. The mathematical definition is given by

$$K_{Ae} = \frac{\sum_{i=1}^{n} MAV_i \times H_i \times K_{Ai}}{\sum_{i=1}^{n} MVA_i \times H_i} \tag{3.18}$$

where K_{Ai}, MVA_i, and H_i are the regulator gain, generator MVA base, and inertia constant of the exciter for generator i, respectively; and n is the number of generators equipped with exciter models in the coherent group. It is noted that the generator inertia is included

in the formula (3.18). This is because in the system data, all MVA bases of the external generators have been converted to 100. Multiplying the MVA base of 100 by the corresponding inertia constant can restore the ratio of MAV base between generators. Other equivalent exciter parameters are set to those of the most dominant unit in the coherent group.

3.5 Calculation Results

The generator and exciter aggregation methods described above are applied to New England and New York system. The six generators in the external system that need to be aggregated are represented by the two-axis model and equipped with Type AC-4 [29] exciters. Table A.1 provides the detailed synchronous machine data for generators 10-16 on a 100 MVA base. Table A.2 gives the Type AC-4 exciter data for these six generators (The generator data and exciter data are given in Appendix A). The equivalent generators are modeled using the GENROU dynamic model. It is important to note that X_d, X_q, X'_d, X'_q, X''_d, X''_q, X_l, H, and D are in per unit based on generator MVA base, and X''_d must equal to X''_q for this type of generator model. The IEEE type AC4A excitation is chosen as the equivalent exciter model.

As investigated in previous sections, six generators in the external area can be divided into two coherent groups. Therefore, two equivalent generators and two associated exciters are required to be determined. The parameters of dynamic generator models calculated by two approaches (i.e., Zhukov's method and detailed aggregation

method provided by DYNRED) are summarized in Table 3-3. Table 3-4 shows the results of equivalent exciters using descried exciter aggregation method. For the sake of simplicity, we use the abbreviations for Zhukov's method and detailed aggregation method provided by DYNRED, namely Zhukov and DYNRED. As shown in Table 3-3, the parameter values of equivalent generators identified by Zhukov's method and DYNRED's method are almost the same.

Table 3-2. Estimated Parameters of Equivalent Generators Eigenvector

Parameters	First coherent group (G10, G11, G12 and G13)		Second coherent group (G14, G15 and G16)	
	Zhukov	DYNRED	Zhukov	DYNRED
Pe (MW)	6441	6441	6785	6785
He (s)	161.8750	161.8750	350	350
De (pu/pu)	24.6650	24.6650	100	100
Xle (pu)	0.0012	0.0012	6.0084×10^{-4}	6.0086×10^{-4}
Xde (pu)	0.011	0.0108	0.006	0.006
Xqe (pu)	0.0103	0.0104	0.0057	0.0057
X'de (pu)	0.0021	0.0021	0.001	0.001
X'qe (pu)	0.0019	0.0019	8.8235×10^{-4}	8.8735×10^{-4}
X''de (pu)	0.0015	0.0015	8.109×10^{-4}	8.109×10^{-4}
X''qe (pu)	0.0015	0.0015	8.109×10^{-4}	8.109×10^{-4}

Table 3-3. Estimated Parameters of Equivalent Exciter

	Tr (sec)	Tc (sec)	Tb (sec)	Ka	Ta (sec)	Kc
Equivalent exciter 1	0.01	1.0	10	100	10	0
Equivalent exciter 2	0.01	1.0	10	100	10	0

3.6 Summary

In this chapter, the non-linear time domain simulation was first employed to identify coherent groups of generators. Then, Prony analysis was used as an aide in the interpretation of non-linear time domain simulation results; particularly in the case that the number of coherent groups is greater than the number of boundary buses so that generators need to be further grouped. Both two methods can be implemented in the commercially available power system simulation software (PSS/E). After coherent generators were identified, generators with their exciters in a coherent group were replaced by an equivalent generator and an equivalent exciter. In addition, the aggregation methods used to determine the parameter values of equivalent generators and exciters have been presented. IEEE 68-bus system was used to validate the coherent identification and aggregation of the coherent generators and their exciters, indicating that dynamic features of the external area now can be represented by two equivalent generators with their equivalent exciter models.

The aggregation methods discussed in this chapter are only suitable for the case that

the complete set of modelling data is available. To improve the accuracy of dynamic models by reducing errors resulted by coherent identification method and dynamic aggregation, some crucial parameters of equivalent generators that may have a considerable effect on the study area will be further turned in the next chapter.

Chapter 4 Dynamic Equivalent Circuit for External System

The dynamic equivalent circuit for an external system is constructed by adding equivalent generator and exciter models, which represent the dynamics of the external system, to boundary buses of the static equivalent circuit. A static equivalent circuit of the external system can be generated in PSS/E as described in Chapter 2. When dynamic parameters of the external generator and their exciters are available, the parameters of equivalent models can be calculated by aggregation methods discussed in Chapter 3. This chapter presents a fitting process for determining model parameters, which can be used to improve the accuracy of the dynamic equivalent circuit by eliminating errors brought by coherent identification and dynamic aggregation. In particular, the proposed curve fitting techniques can also be used to determine the parameters of equivalent models when the complete set of modelling data are not available.

4.1 Proposed Methodology

In this subsection, the equivalent system refers to the reduced system, consisting of the unchanged internal system and the dynamic equivalent external system. The proposed fitting process determines the parameters of equivalent generators in an iterative way. The dynamic parameters of equivalent generators are identified by fitting the response of the equivalent system with the original system. The optimal parameters of equivalent

generators are obtained by minimizing the error between the dynamic response of the original system and the equivalent system. The transient voltage recovery waveforms are used to as the error criterion for the fitting process. To investigate the effect of the external system on the dynamic performance of the internal system, the voltage recovery characteristics at the buses in the internal system are studied. The transient voltage recovery waveform is obtained by performing transient simulation. Thus, voltage values at each simulation time instant are chosen to form an objective function for the fitting process, given by

$$minf(x) = \sum_{i=0}^{n} \left[V_f(i) - V_e(x,i)\right]^2 \tag{4.1}$$

subject to

$$x_{low} \leq x \leq x_{up}$$

where the subscripts f and e denote the full system and the equivalent system, respectively. The voltage values are given by points on the transient voltage recovery waveform and n is the total number of points over the simulation period. $V_f(i)$ is the voltage value (known in advance) at point i obtained from the full system, $V_e(x,i)$ is the voltage value at point i obtained from the equivalent system. Here, x denotes the parameter to be identified by the fitting process. The voltage value of the equivalent system at point i varies as a function of the parameter x. x_{low} and x_{up} are the parameter lower and upper bounds. The basic principle of the objective function is to search for

optimal parameters so that the dynamic responses of the reduced system can be roughly close to the dynamic responses of the full system. The fitting process is depicted in Figure 4-1.

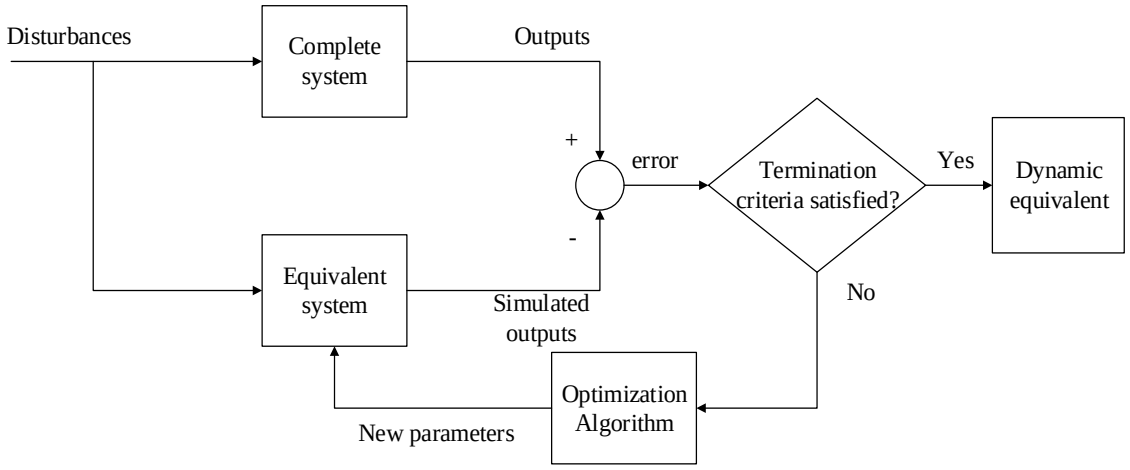

Figure 4-1. Process of determining dynamic model parameters

The proposed algorithm is iteratively executed in the Python environment based on the PSS/E Application Program Interface (API). Each iteration of the algorithm automatically:

- Run dynamic simulation of equivalent system in PSS/E
- Export dynamic response of equivalent system to Python
- Evaluate the objective function $f(x)$ for simulation time period
- Adjust parameters according to the optimization algorithms
- Feedback the adjusted values to PSS/E

API provides various functions that allow Python to read and write from and to

PSS/E by creating an interface between them. The algorithm runs PSS/E within Python environment. Simulation results will be analyzed in Python for next iteration. The iteration between Python and PSS/E is illustrated in the following flowchart (of Figure 4-2).

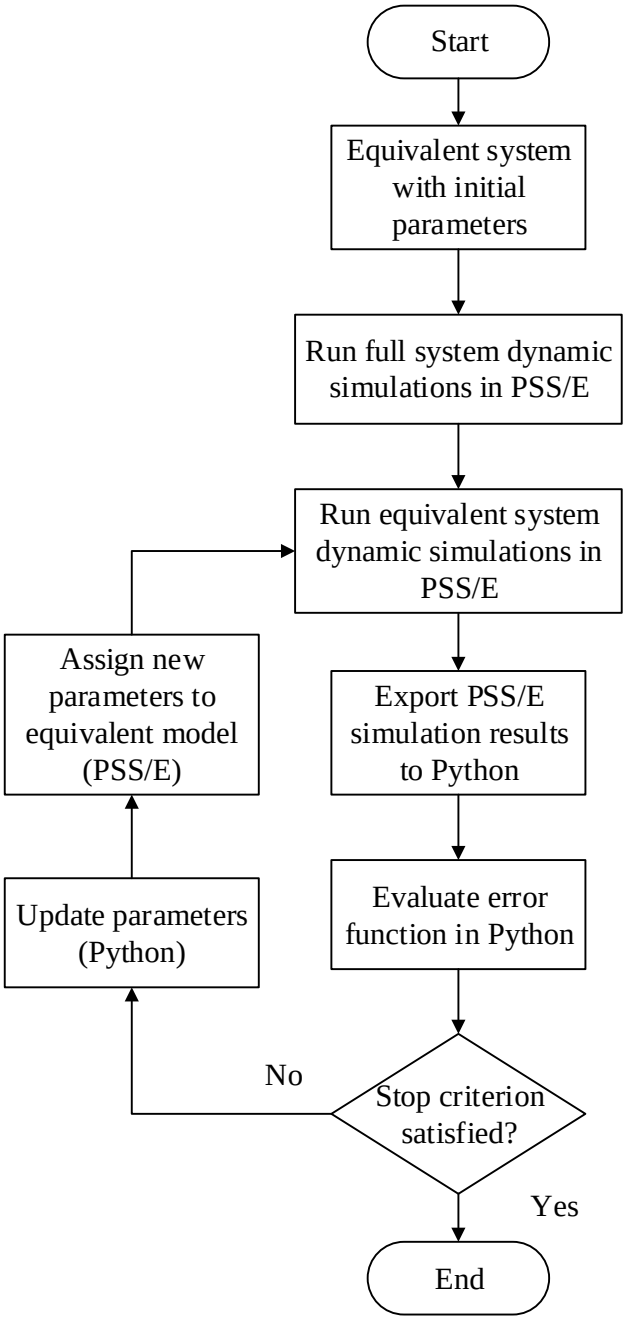

Figure 4-2. Interface between PSS/E and Python

The overall process of the proposed method is composed of four functions listed below:

- Function-1: Dynamic simulation setup
- Function-2: Dynamic simulation procedures
- Function-3: Data evaluation
- Function-4: Plotting

The fitting process is an iterative process, as shown in Figure 4-3, with each iteration consisting of Function-2 and Function-3: dynamic simulations and data evaluation. Function-1 is conducted to prepare the case prior to running a dynamic simulation. Function-4 is used to plot the desired results when the iteration process ends.

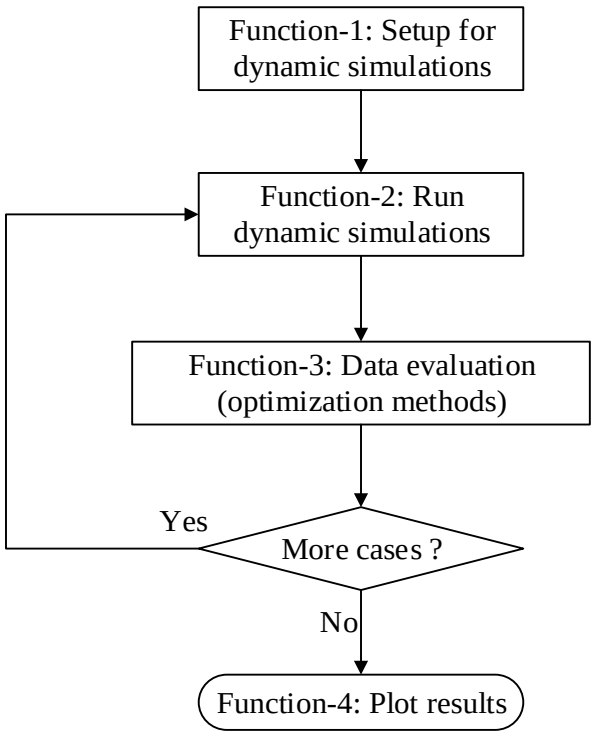

Figure 4-3. Flow chart of the fitting process

4.1.1 Function-1: Dynamic Simulation Setup

Dynamic simulation setup function aims to get the case prepared for performing a dynamic simulation in PSS/E. This function starts with a solved PSS/E load flow case, and the following activities will be executed:

1. Activity CONL: convert constant MVA loads to desired constant power, constant current, and constant admittance characteristics
2. Activity CONG: convert generators from power flow models to Norton equivalents that are required by the dynamic simulation activities
3. Activity ORDR: re-order buses to maintain sparsity
4. Activity FACT: factorize the admittance matrix (Y matrix) that will be used in activity TYSL
5. Activity TYSL: calculates triangularized Y matrix network solution

The flow chart of function-1 is shown in Figure 4-4. The first column provides a brief description of the activities described above, the second column indicates the corresponding PSS/E activity ID, and the corresponding python programming instructions are given in the last column. This function results in a converted power flow case which needs to be saved in its converted form for subsequent dynamic analyses. Note that this converted case should be saved using a different name because the conversion process is not reversible.

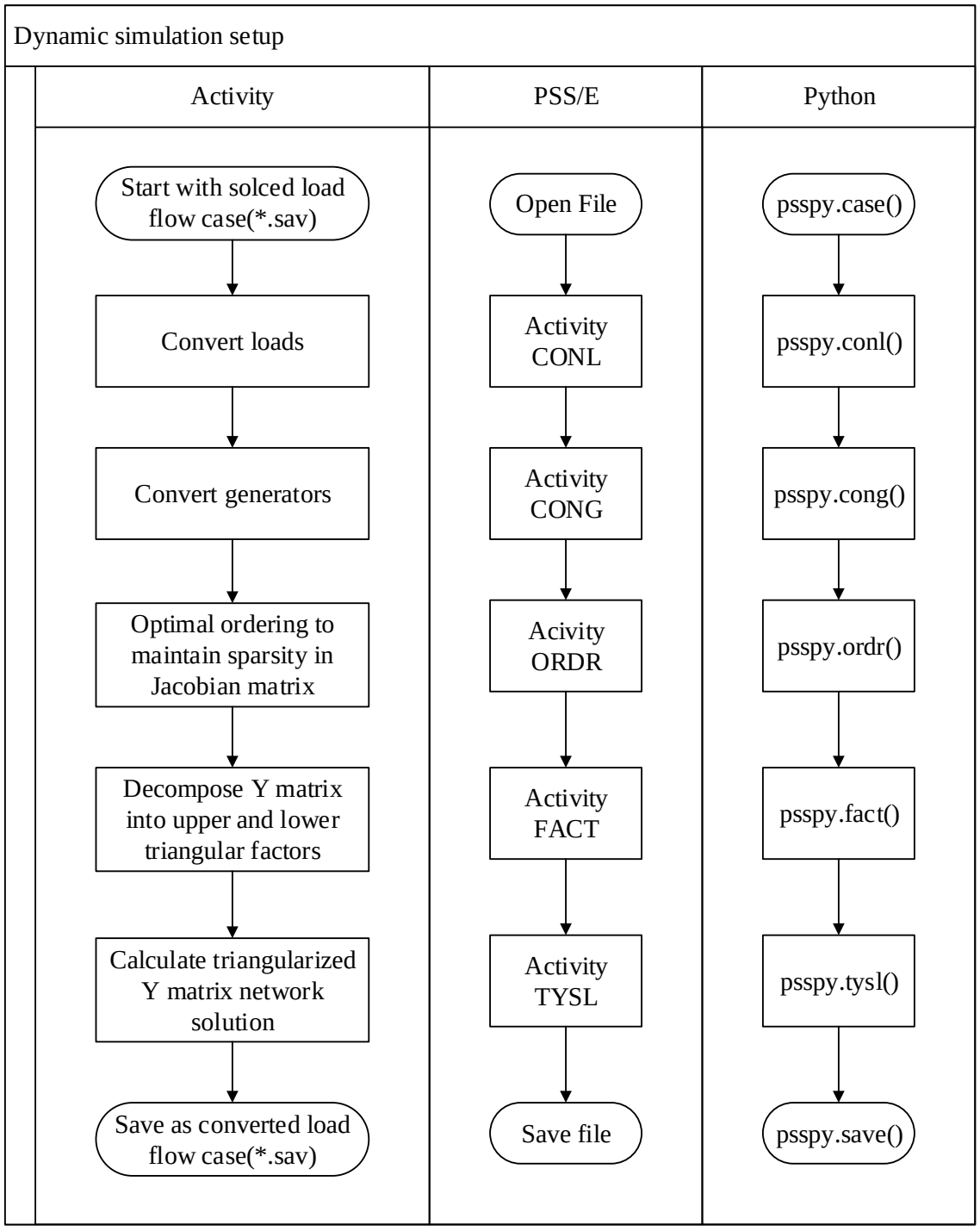

Figure 4-4. Flow chart of Function-1: Dynamic simulation setup

4.1.2 Function-2: Dynamic Simulation Procedures

After finishing function-1, the dynamic simulations in function-2 will be implemented. This function involves:

- Perform the dynamic simulations by applying disturbances as required
- Obtain the transient voltage recovery waveforms

The flow chart of the dynamic simulation process is shown in Figure 4-5. It is seen that the simulation procedures are described in the left column, and the right column provides the corresponding python programming instructions. The outputs of dynamic simulations (i.e., voltage recovery waveforms) will be analyzed by function-3.

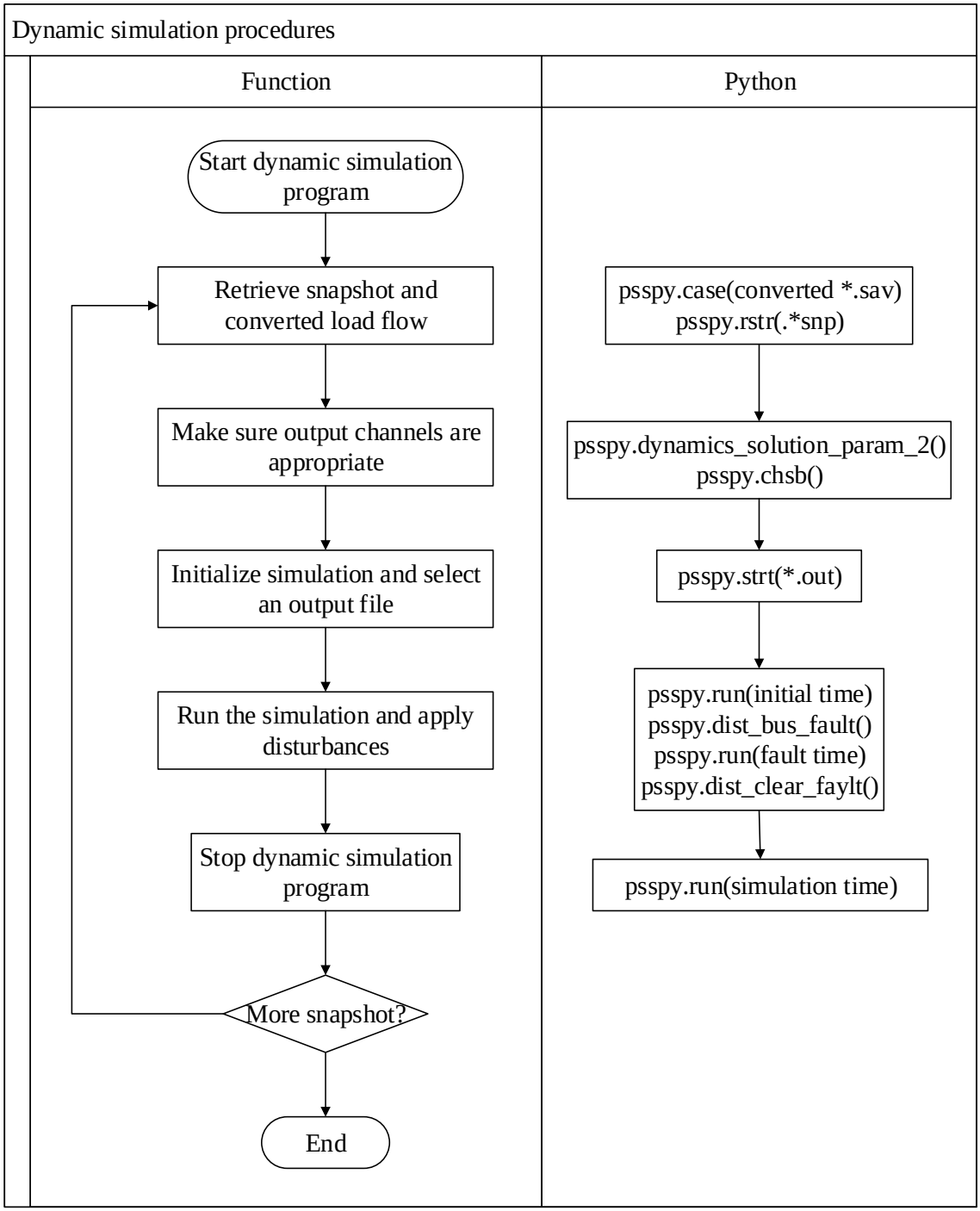

Figure 4-5. Flow chart of Function-2: Dynamic simulation procedures

4.1.3 Function-3: Data Evaluation

The results of dynamic simulations are saved in a binary file (*.out), which can be exported to excel spreadsheets using the dyntools module. To perform data evaluation in Python, we need to convert the desired column data in the excel file to an array of values. Then, voltage values at each time instant obtained from equivalent system are compared to that of the full system. In this optimization problem, there is no mathematical function that can be used to express the relationship between the inputs (i.e., equivalent model parameters) and the outputs (i.e., voltage responses). The system output corresponding to the specific set of system inputs can only be obtained by using time domain simulation tools. As such, the objective function which relates the outputs, i.e., the transient voltage responses, to the inputs, i.e., the parameters of equivalent models, cannot take derivatives. Three optimization algorithms, namely Monte Carlo method, Nelder-Mead method and Powell method, are adopted here to determine the equivalent parameters by minimizing the objective function. These three optimization methods are chosen because they do not require any derivative information. In the following, a brief introduction to three methods, as well as the working mechanism, will be given.

4.1.3.1 Monte Carlo Method

The Monte Carlo method aims to generate a large number of random samples and then use these samples to calculate the values of interest. In the Monte Carlo method, generating random variables with known probability distribution is the main task,

therefore the Monte Carlo method is also known as random sampling [30].

In this research, uniform distribution are used to generate random variables for dynamic models, which can be achieved by using python function: $random.uniform(x, y)$. The Python code for this method is presented in Appendix C.1.

4.1.3.2 Nelder-Mead Method

Nelder-Mead Method is used to find the maximum and minimum values of the objective function in a multidimensional space. It is a direct search method based on comparison, and is usually applied to nonlinear optimization problems with unknown derivatives [31].

The Nelder-Mead algorithm uses n+1 test points to construct a simplex in an n-dimensional space, and then calculates the corresponding objective function value of each point. The purpose is to obtain an optimal point by finding a new test point to replace the old test point. The worst point will be replaced by the reflection point of the centroid of the remaining n points. If the reflection point is better than the previous points, the search is continued in the direction of the reflection point; if not, all points are shrunk in a better direction. There are four potential operations for updating points at each iteration, namely, reflection, expansion, contraction and shrink, and one of them might be implemented by judging current situations. More details of the Nelder-Mead algorithm can be found in [32].

The spicy.optimize package provides several commonly used optimization algorithms, including the Nelder-Mead algorithm. The corresponding Python code is given in Appendix C.2.

4.1.3.3 Powell Method

Powell method is also known as the directional acceleration method, which was proposed by Powell in 1964 [33]. It is a search method that uses the properties of conjugate directions to accelerate the convergence rate. Therefore, Powell algorithm is a very effective direct search method. The principle is to establish a direction set, where the directions in this set are linearly dependent of each other. Starting from the initial point (x_0), a linear search is performed to find the optimal solution along this direction (using the Brent method), so that a new position (x_1) is obtained. Then, by adding a new direction of ($x_1 - x_0$) to the direction set, the first direction in the original set is removed. The algorithm keeps iterating until there is no significant improvements. More details of the Powell method can be found in [34], [35].

The Powell method is also available in scipy.optimize by setting method='Powell', and the relevant Python code can be found in Appendix C.3.

4.1.4 Function-4: Plotting

Last function is used to plot the optimal results so as to provide a visual comparison between the transient voltage recovery waveforms of the full system and that of the equivalent system. The flow chart of function-4 is shown below (Figure 4-6), where CHNF function under dyntools module provides methods to access and post process the data from PSSE dynamics simulation studies channel output files (.out). Optnchn is a dictionary under matplotlib module specifying channels to plot and plotting options.

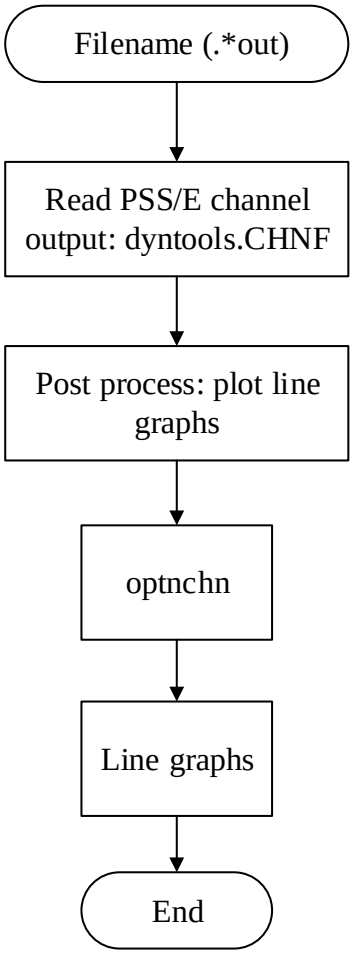

Figure 4-6. Flow chart of Fuction-4: Plotting

4.2 Results and Discussion

In this section, the proposed method is validated by identifying optimal parameters of equivalent generators. The original transient voltage response is obtained from the full New England & New York 68-bus system (Figure 2-4. Single-line diagram of IEEE 68-bus power system). The reduced 68-bus system for validating the fitting process is shown in Figure 4-7. It is seen from Figure 4-7, the reduced 68-bus system consists of an unchanged internal system and a dynamic equivalent external system, where generators in the external area (generators 10-16) are replaced by two equivalent generators at boundary buses 53 and 61. To investigate the influence of external system on the dynamic performance of internal system, a fault is applied in the internal system and the dynamic responses of buses in the internal system are analyzed. Here, we only take two main parameters of each equivalent generator as an example to validate the proposed fitting process.

- H: inertia
- D: damping coefficient

These two parameters for each equivalent generator must be determined such that the objective function is minimized. For both the full system and the reduced system, a three phase fault was applied at bus 37 at 1 second and cleared after six cycles (100ms). The transient voltage response of bus 27, which is one bus away from bus 37 and close to the external system, is monitored. Therefore, the accuracy of the proposed fitting

process using optimization algorithms is examined by how well the terminal voltage at bus 27 of the reduced system match with that of the full system.

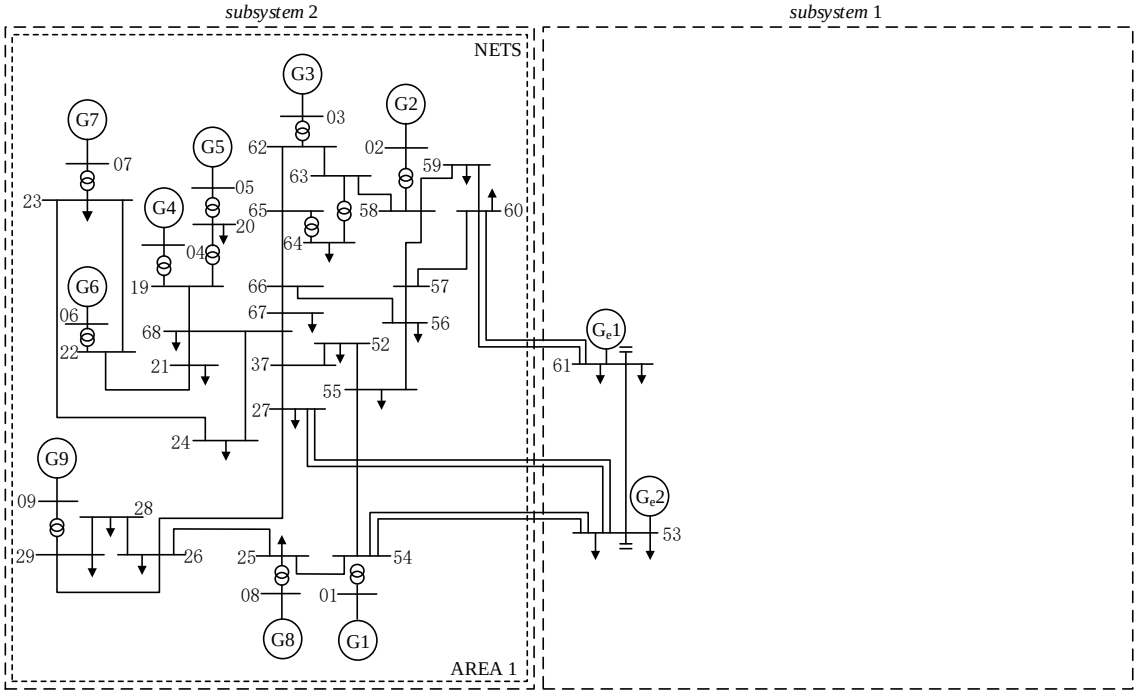

Figure 4-7. Single-line diagram of reduced 68-bus system with external system represented by dynamic equivalent circuit

4.2.2 Validation of the Monte Carlo Method

To validate the Monte Carlo method, the corresponding ranges of four parameters were given. The best five fitting curves are plotted, as shown in Figure 4-8. It is seen that the transient voltage recovery waveforms of the reduced system match well with that of the full system. In addition, the worst five cases are also plotted, as shown in Figure 4-9, as a comparison to the good matching in Figure 4-8. It turns out that the Monte Carlo method provides good performance in determining dynamic parameters.

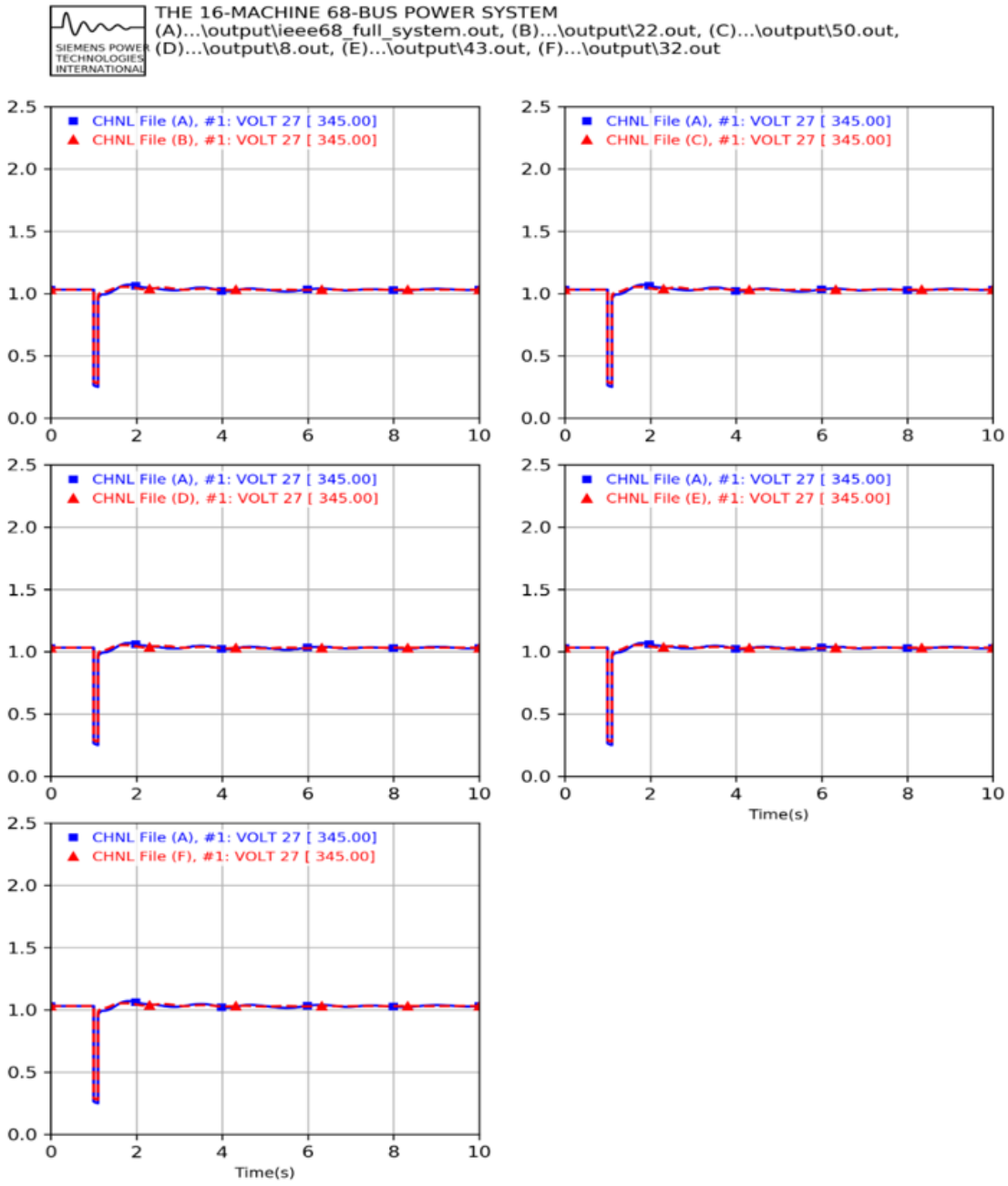

Figure 4-8. Best five cases obtained by Monte Carlo method

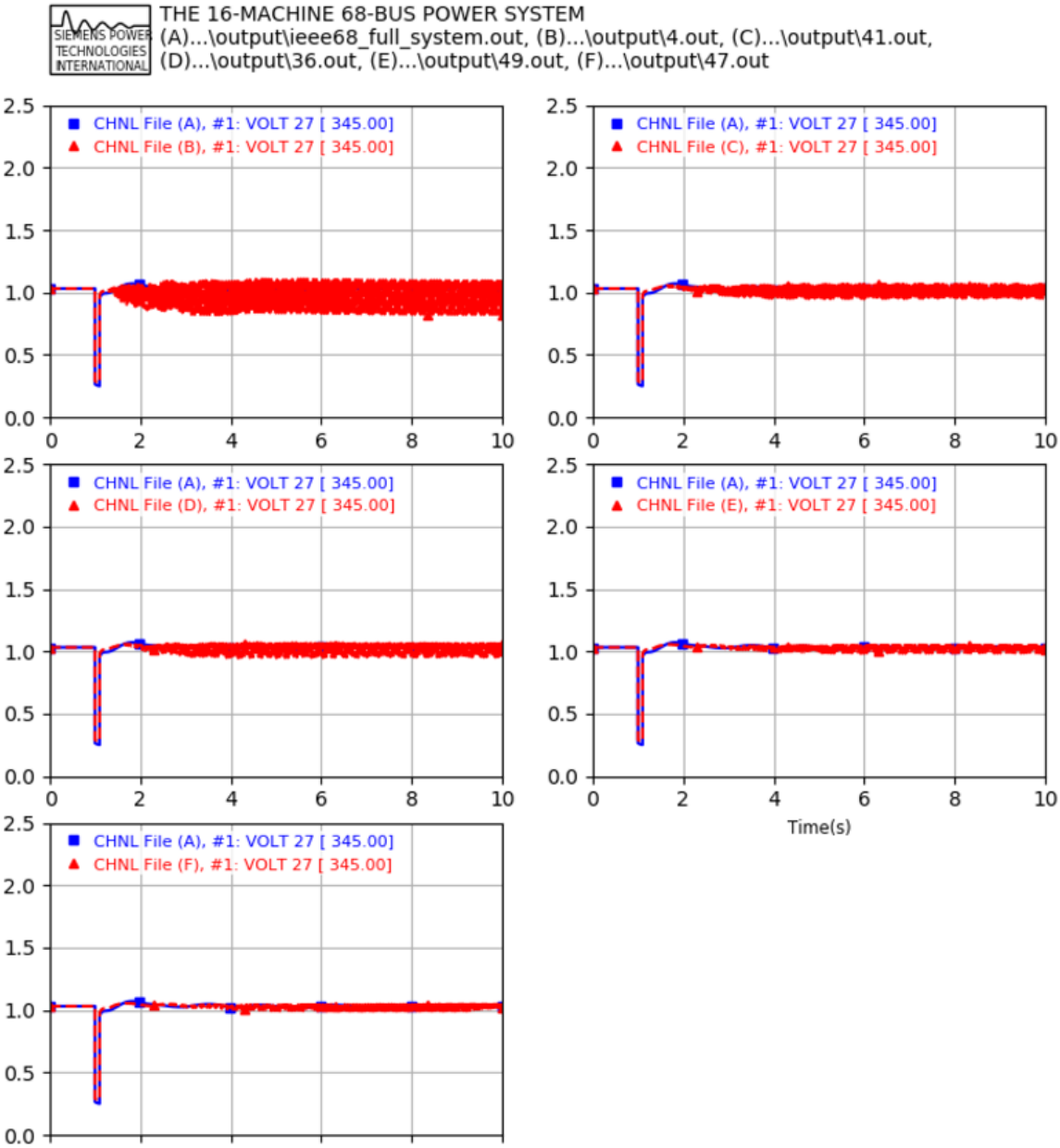

Figure 4-9. Worst five cases obtained by Monte Carlo method

4.2.3 Validation of Nelder-Mead Method

In the case that the complete dynamic parameters of generators are available, the parameters of equivalent generator can be calculated by aggregation methods described in Chapter 3. For IEEE 68-bus system, the aggregation values of parameters H and D

(H_1=161.875, D_1=24.665, H_2=350, D_2=100) can be used as a good starting point for optimization methods. When the dynamic parameters of generators are unknown, equivalent parameters H and D are set to random values (e.g., H_1=0.5, D_1=0.2, H_2=0.5, D_2=0.2), which can be regarded as bad initial values to begin the optimization process.

The fitting process utilizing Nelder-Mead method were validated under both situations. Simulation results are shown in Figs. 4-10-4-11. Figure 4-10 compares the scenario in which good initial values are used to its corresponding optimal result obtained by fitting process utilizing Nelder-Mead method. Figure 4-11 compares the scenario in which bad initial values are used to its corresponding optimal result obtained by fitting process using Nelder-Mead method. It is seen that the Nelder-Mead method fails to converge to a critical point of objective function with if the starting point is not good enough. That is, it fails to find optimal values for parameters H and D of equivalent generators when the optimization process starts with bad initial values.

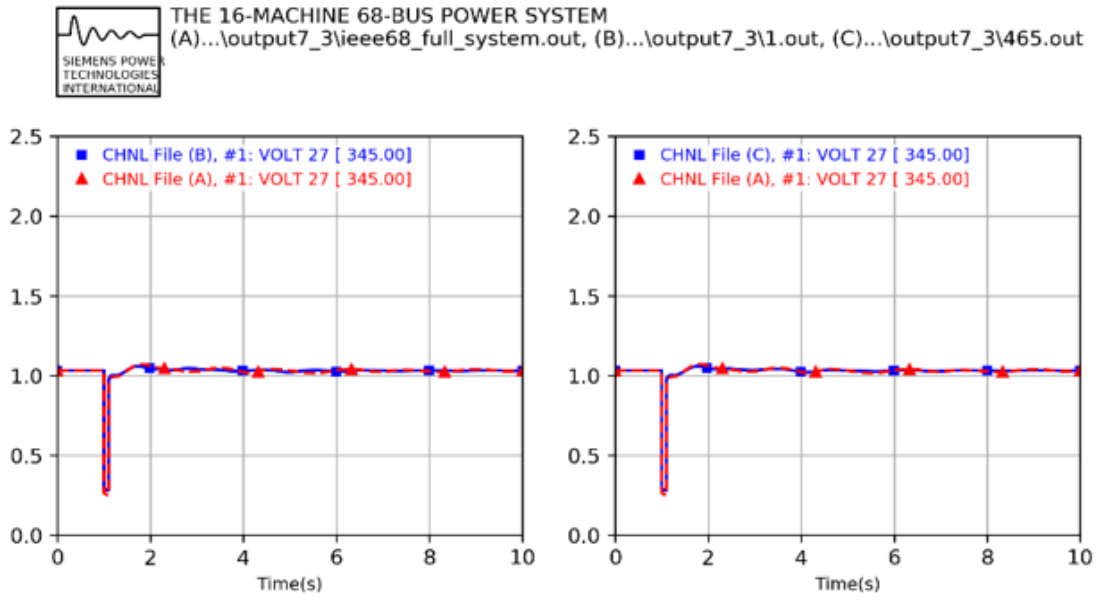

Figure 4-10. Comparison between the good starting scenario and its corresponding optimal scenario using Nelder-Mead method

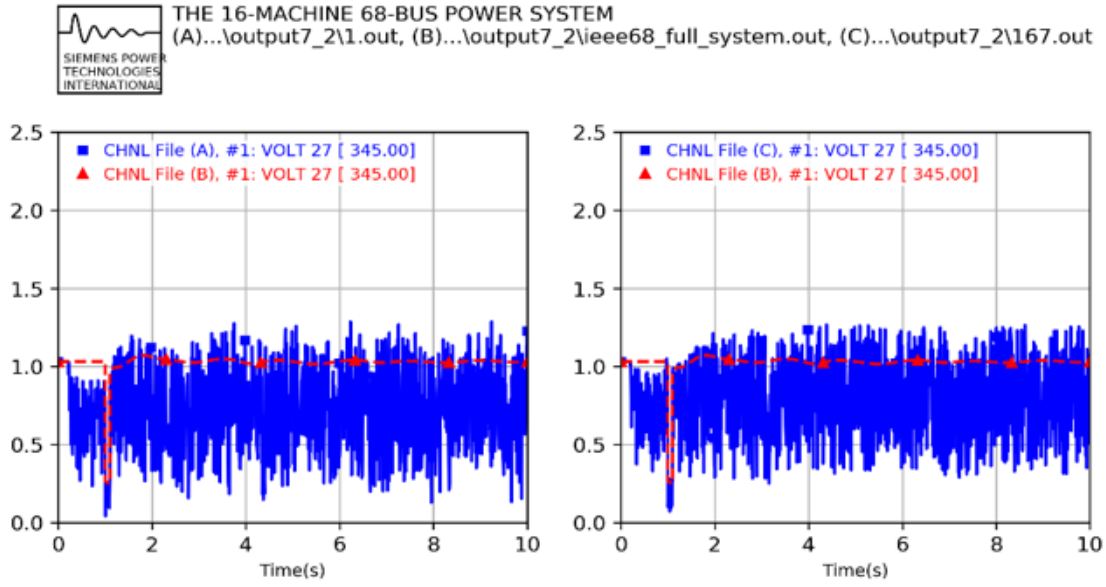

Figure 4-11. Comparison between the bad starting scenario and its corresponding optimal scenario using Nelder-Mead method

4.2.4 Validation of Powell Method

Similar to the procedures of validating the Nelder Mead method, the comparison of good initial scenario with its optimal scenario obtained by using Powell method is shown in Figure 4-12. Meanwhile, the optimal scenario resulted from Powell method starts with bad initial values are compared to the scenario in which its bad initial values are used, which are shown in Figure 4-13. By comparison, Powell method is efficient in finding optimal parameters for H and D to match the full system voltage responses, even if the starting values are bad.

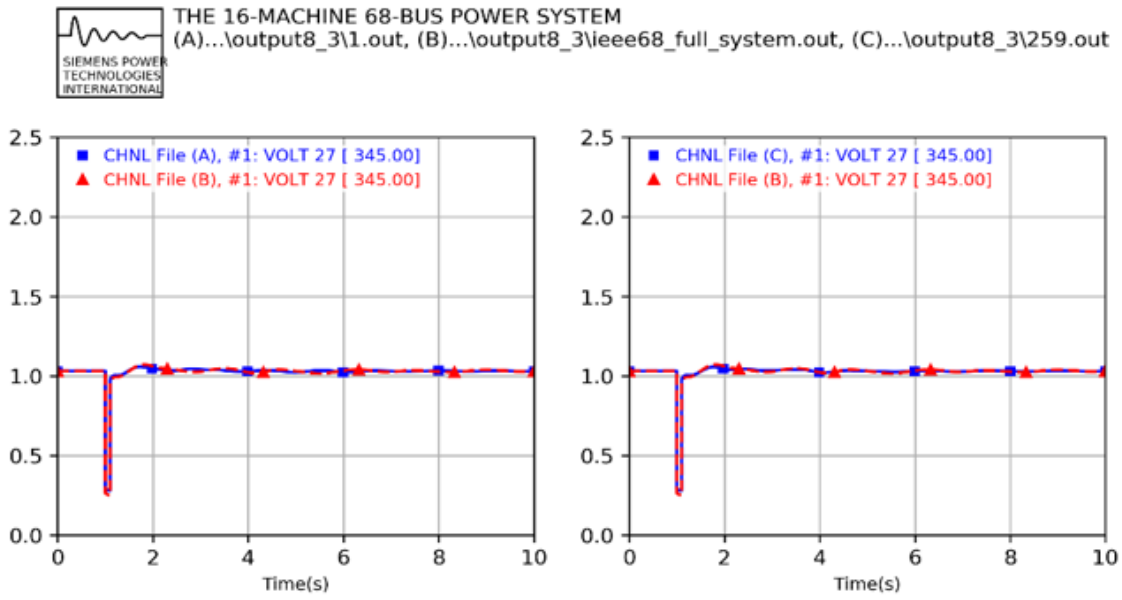

Figure 4-12. Comparison between the good starting scenario and its corresponding optimal scenario using Powell method

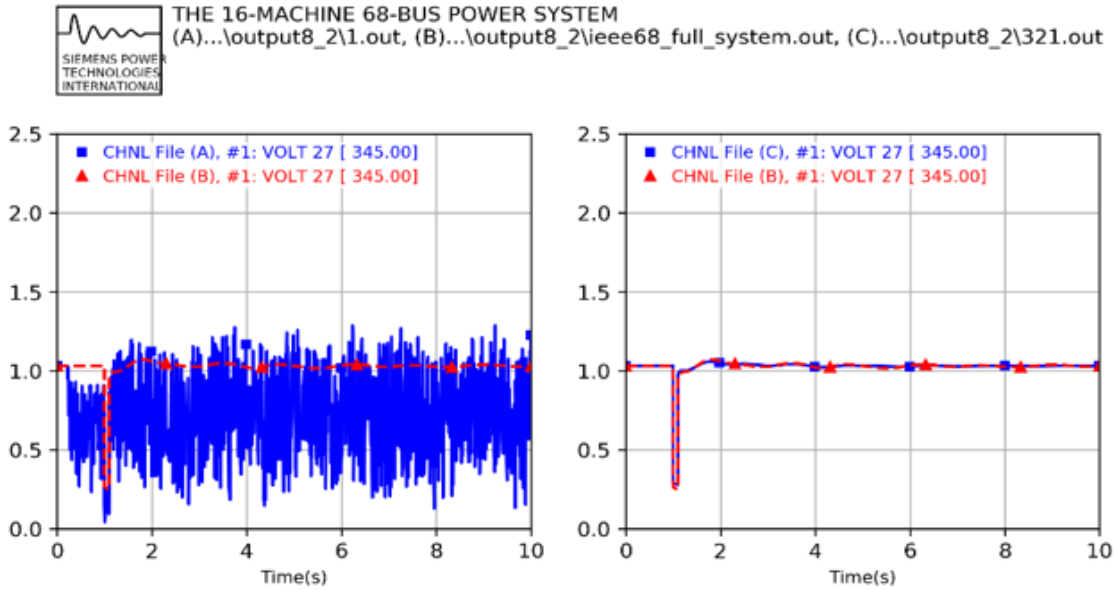

Figure 4-13. Comparison between the bad starting scenario and its corresponding optimal scenario using Powell method

4.3 Summary

Chapter 4 proposed a fitting process utilizing the optimization algorithms to determine the parameters of equivalent generators. The objective function is defined as the sum of square of the differences in voltage response obtained from the full system and the corresponding ones obtained from the dynamic equivalent system. The fitting process can be implemented using the following procedures in the Python environment. First, we set up the power flow case for dynamic simulation. Second, the dynamic simulations are performed in PSS/E, of which the simulation results are then exported to Python for data evaluation. Dynamic simulations and data evaluation are repeated until we get the minimum value of the objective function. The last step is to plot the desired results. For

data evaluation, three optimization algorithms including Mento Carlo method, Nelder-Mead Method and Powell method are used for this research. These methods are adopted for calculating the minimum of the objective function without an underlying mathematical definition.

Simulations were performed to validate the proposed fitting process using three optimization algorithms. Monte Carlo and Powell methods were proved to be effective in identifying the equivalent parameters even when we do not have the complete dynamic parameters of generators that need to be equivalenced. Despite that the Nelder-Mead method is easy to implement, it may fail to converge to a critical point of the objective function with bad initial values. In other words, we cannot use Nelder-Mead method to determine the parameters of equivalent models when complete dynamic data are not accessible.

Chapter 5 Conclusions and Future Work

This section presents the conclusions and contributions of the thesis and future work can be carried out in this research area.

5.1 Conclusions

The power system is not a static system. It is a dynamic system and vulnerable to various types of disturbances such as short circuit faults, loss of a transmission line, power transformers, or generators. When large power networks are to be simulated it requires a heavy computational burden. A common practical approach to handle a large system is to split the power system into two sub systems, internal and external. Internal system is the part of the network in which the simulations need to be performed. This thesis proposed to develop a dynamic equivalent circuit for external power system using integrated software PSS/E and Python.

The proposed approach starts with generating a static equivalent circuit in PSS/E for the external system to significantly reduce the size of external network, which can be established following four steps: 1) Distinguish study and external areas. 2) Change bus type code to retain the desired buses in the external system. 3) Net generation with equivalent negative loads to eliminate generations in the external system. 4) Build an equivalent circuit for external system using Gauss elimination method. This is described

in Chapter 3. After the static equivalent circuit was established, the dynamic equivalent circuit for external system can be constructed by adding equivalent generators and associated controls at the boundary buses, where the equivalent generators are obtained by analyzing the coherency among generators in the external system. In this research, the non-linear time domain simulation was conducted to identify coherent generators since it is easy to implement in PSS/E. Moreover, Prony analysis that is also available in PSS/E was used to further group the external generators, which is based on comparing the damping components, amplitudes, frequencies and phase of dynamic simulation results of each generator. After the coherent groups were identified, generators and equipped exciters in a coherent group were aggregated to an equivalent generator and an equivalent exciter. The equivalent generator parameters were calculated based on the Zhukov's method or DYNEQU's method, while the aggregate exciter parameters were calculated by using the MVA-base weighted Ka, with the time constants set to those from the exciter of the most dominant unit in a coherent group. This part is presented in Chapter 4. Chapter 5 focused on tuning the equivalent generator parameters to generate optimal transient voltage recovery close to that of the full system. An optimization based fitting process was proposed to determine the values of equivalent model parameters, in which Monte Carlo, Nelder-Mead and Powell optimization methods were employed. New York and New England IEEE 68 bus was used for validating the effectiveness of the proposed method in constructing a dynamic equivalent circuit. It was shown that the proposed

approach based on Monte Carlo and Powell methods are effective to search optimal equivalent generator parameters, whereas, the Nelder-Mead algorithm failed to provide an optimal solution in the case that the choice of initial values is not good enough.

5.2 Contributions

This thesis presented an approach to develop a dynamic equivalent circuit for the external system while remaining its effects on the study system. The main feature of this approach is the use of a nonlinear optimization based fitting process to determine the model parameters of equivalent generators. The optimization technique has been used to supplement the coherency method, where coherent generators were identified using non-linear time domain simulation combined with Prony analysis features available in the PSS/E software. The developed approach using integrated commercial software PSS/E and Python is easier and more automated, thus repeatable. Validations proved that Monte Carlo and Powell methods were effective in determining the parameters of equivalent generators even when the parameters of external generators are not available.

5.3 Future Work

The future work of this thesis are proposed from the following perspectives

- The proposed optimization method for determining the equivalent parameters can be used to investigate the influence of the parameters of fictitious equivalent generators (rating, reactance, inertia, damping and time constants) on the transient

voltage recovery properties of the equivalent system. Based on the understanding gained in the above task, some rules to tune the basic parameters of the equivalent generators may be developed.

- Only the parameters of equivalent generators were investigated in this thesis. The research on the equivalent parameters of aggregate exciters and aggregate governor models could be considered in the future.
- In this research, three optimization methods are used. However, the Powell method does not use any gradient evaluations, it may take longer to find the minimum. There might be other more efficient optimization algorithms that can be used to quickly produce satisfactory results.

References

[1] P. Dandeno, P. Kundur, S. Umans, I. Kamwa, H. Karmaker, S. Salon, M. Shah and A. El-Serafi, “IEEE guide for synchronous generator modeling practices and applications in power system stability analyses,” *IEEE Power Eng. Soc.*, vol. 11, no. 2, pp. 1110-2002, 2003

[2] B. Stott, “Power system dynamic response calculations,” *Proc. IEEE*, vol. 67, no. 2 pp. 219-241 Feb. 1979.

[3] S. Takeda and S. Nishida, “Derivation of Dynamic Equivalents for Stability Analysis,” *Elect. Power Energy Syst.*, vol. 2, no. 3, pp. 20-31, 1984.

[4] S. Nishida and S. Takeda, “Derivation of Equivalents for Dynamic Security Assessment,” *Elect. Power Energy Syst.*, vol. 6, no. 5, pp. 13-25, 1980.

[5] S. Olivera and J. Queiroz, “Modal Dynamic Equivalents for Electric Power Systems. I: Theory,” *IEEE Trans. Power Syst.*, vol. 3, no. 6, pp. 65-79, 1988.

[6] R. J. Galarza J. H. Chow W. W. Price A. W. Hargrave and P. M. Hirsch, “Aggregation of exciter models for constructing power system dynamic equivalents,” *IEEE Trans. Power Syst.*, vol. 13, no. 3, pp. 782-788, 1998.

[7] R. J. Newell, M. D. Risan, L. Allen, I. S. Rao and D. L. Stuehm, “Utility experience with coherency-based dynamic equivalents of very large systems,” *IEEE Trans. Power App. Syst.*, vol. PAS-104, no. 11, pp. 3056-3063, 1985.

[8] L. Wang, M. Klein, S. Yirga and P. Kundur, “Dynamic reduction of large power systems for stability studies,” *IEEE Trans. Power Syst.*, vol. 12, no. 2, pp. 889-895, 1997.

[9] X. Lei, D. Povh and O. Ruhle, “Industrial approaches for dynamic equivalents of large power systems,” in *Proc. IEEE Power Eng. Soc. Winter Meeting*, New York City, New York, 2002, pp. 1036-1042.

[10] Y.-N. Yu and M. A. El-Sharkawi, “Estimation of external dynamic equivalents of a thirteen-machine system,” *IEEE Trans. Power App. Syst.*, vol. PAS-100, no. 3, pp. 1324-1332, 1981.

[11] A. M. Stankovic and A. T. Saric, “Transient power system analysis with measurement-based gray box and hybrid dynamic equivalents,” *IEEE Trans. Power Syst.*, vol. 19, no. 1, pp. 455-462, 2004.

[12] P. Ju, L. Q. Ni and F. Wu, “Dynamic equivalents of power systems with online measurements. Part 1: Theory,” *Proc. Inst. Elect. Eng. Gen. Transm. Distrib.*, vol. 151, no. 2, pp. 175-178, 2004.

[13] P. Ju, F. Li, N. G. Yang, X. M. Wu and N. Q. He, “Dynamic equivalents of power systems with online measurements Part 2: Applications,” *Proc. Inst. Elect. Eng. Gen. Transm. Distrib.*, vol. 151, no. 2, pp. 179-182, 2004.

[14] PTI, *PSS/E Application Guide* (Vol. I). PSS/E Brochure, 2002.

[15] S. W. Peter and P. M. Mangalore, *Power System Dynamics and Stability*, London: John Wiley & Sons Ltd, 1997.

[16] A. J. Germond and R. Podmore, "Dynamic Aggregation of Generating Unit Models," *IEEE Trans. Power Apparatus Syst.*, vol. PAS-97, no. 4, pp. 1060-1069, 1978.

[17] T. L. Baldwin and L. M. Arun and G. Phadke, "Dynamic ward equivalents for transient stability analysis," *IEEE Trans. Power Syst.*, vol. 9, no. 1, pp. 59-67, 1994.

[18] S. Chittora and S. N. Singh, "Coherency based dynamic equivalencing of electric power system," in *2014 Eighteenth Nat. Power Syst. Conf. (NPSC)*, Guwahati, 2014, pp. 1-6.

[19] G. Rogers, *Power System Oscillations*, Norwell, MA: Kluwer, 2000.

[20] G.P. Deng, Y.Z. Sun and Jian Xu, "A new voltage stability analysis method by considering short circuit capacity", *Autom. Electr. Power Syst.*, vol. 33, no. 8, pp. 15-19, 2009.

[21] J. P. Yang, G. H. Cheng and Z. Xu, "Dynamic reduction of large power system in PSS/E," *IEEE/PES Tran. and Distrib. Conf. Exhib.*, Dalian, China, 2005, pp. 1-4.

[22] M. L. Ourari, L. -. Dessaint and Van-Que Do, "Dynamic equivalent modeling of large power systems using structure preservation technique," *IEEE Trans. Power Syst.,* vol. 21, no. 3, pp. 1284-1295, 2006.

[23] IEEE Task Force Identification of Electromechanical Modes in Power Systems June

2012.

[24] J. F. Hauer, C. J. Demeure and L. L. Scharf, “Initial results in prony analysis of power system response signals,” *IEEE Trans. Power Syst.*, vol. 5, no. 1, pp. 80-89, 1990.

[25] Modal Analysis Plotting, *PSSPLT Program Manual*, PSS/E.

[26] P. Robin and A. Germond, *Development of dynamic equivalents for transient stability studies*, EPRI Report EL-456, 1977.

[27] S. W. Peter and M. A. Pai, *Power system dynamics and stability*, New York: Willey, 2008.

[28] P. M. Frank, *An introduction to sensitivity theory*, New York: Academic Press 1978.

[29] I. C. Report, “Excitation Models for Power System Stability Studies,” *IEEE Trans. Power Apparatus Syst.*, vol. 2, pp. 494-509, 1981.

[30] J. R. Birge, *Quasi-monte carlo approaches to option pricing*, Technical Report, Department of Industrial and Operation Engineering, University of Michigan, 1994.

[31] Jr. J. E. Dennis and V. Torczon, “Direct search methods on parallel machines,” *SIAM J. Optim*., vol. 1, no. 4, pp. 448-474, 1991.

[32] F. Gao and L. Han, “Implementing the Nelder-Mead simplex algorithm with adaptive parameters,” *Comput. Optim. Appl*., vol. 51, no. 1, pp. 259-277, 2012.

[33] M. J. D. Powell, “An efficient method for finding the minimum of a function of several variables without calculating derivatives,” *Comput. J*., vol. 7, pp. 155-162,

1964.

[34] X. Du, J. Dang, Y. Wang, X. Liu and S. Li, “An algorithm multi-resolution medical image registration based on firefly algorithm and powell,” *IEEE Conf. Intell. Syst. Design Eng. Appl.*, Hong Kong, 2013, pp. 274-277.

[35] Y. Lei and V. Zhang, “An improved 2D-3D medical image registration algorithm based on modified mutual information and expanded Powell method,” in *IEEE Int. Conf. Medical Imaging Physics Eng. (ICMIPE)*, Shenyang, 2013, pp. 24-29.

Appendix A Test System Data

Table A. 1. Generator Dynamic Data on 100 MVA Base

Machine	Xl (pu)	Xd (pu)	X'd (pu)	X''d (pu)	T'do (sec)	T''do (sec)
1	0.0125	0.1	0.031	0.025	10.2	0.05
2	0.035	0.295	0.0697	0.05	6.56	0.05
3	0.0304	0.2495	0.0531	0.045	5.7	0.05
4	0.0295	0.101262	0.0436	0.035	7.69	0.05
5	0.027	0.33	0.066	0.05	5.4	0.05
6	0.0224	0.254	0.05	0.04	7.3	0.05
7	0.0322	0.295	0.049	0.04	5.66	0.05
8	0.028	0.29	0.057	0.045	6.7	0.05
9	0.0298	0.2106	0.057	0.045	4.79	0.05
10	0.0199	0.169	0.0457	0.04	9.37	0.05
11	0.0103	0.128	0.018	0.012	4.1	0.05
12	0.022	0.101	0.031	0.025	7.4	0.05
13	0.00015	0.0148	0.00275	0.002	5.9	0.05
14	0.0017	0.018	0.00285	0.0023	4.1	0.05
15	0.0017	0.018	0.00285	0.0023	4.1	0.05
16	0.0041/2	0.00205	0.0178	0.00275	7.8	0.05

Table A. 1. (continued) Generator Dynamic Data on 100 MVA Base

Machine	Xq (pu)	X'q (pu)	X''q (pu)	T'qo (sec)	T''qo (sec)	H (sec)	D
1	0.069	0.028	0.025	1.5	0.035	42.05	4.0
2	0.282	0.060	0.05	1.5	0.035	30.2	9.75
3	0.0237	0.050	0.045	1.5	0.035	35.8	10
4	0.258	0.040	0.035	1.5	0.035	28.6	10
5	0.31	0.060	0.05	0.44	0.035	26.0	3
6	0.241	0.045	0.04	0.4	0.035	34.8	10
7	0.292	0.045	0.04	1.5	0.035	26.4	8
8	0.280	0.050	0.045	0.41	0.035	24.3	9
9	0.205	0.050	0.045	1.96	0.035	34.5	14
10	0.115	0.045	0.04	1.5	0.035	31.6	5.56
11	0.123	0.015	0.012	1.5	0.035	28.2	13.6
12	0.095	0.028	0.025	1.5	0.035	92.3	13.5
13	0.0143	0.025	0.002	1.5	0.035	496	66
14	0.0173	0.0025	0.0023	1.5	0.035	300.0	100
15	0.0173	0.0025	0.0023	1.5	0.035	300.0	100
16	0.0167	0.003	0.00275	1.5	0.035	450	100

Table A. 2. IEEE Type AC4 Excitation System Data

Tr(sec)	Tc(sec)	Tb(sec)	Ka	Ta(sec)	Kc
0.01	1.0	10	100	10	0

Appendix B Prony Analysis Result

THE 16-MACHINE 68-BUS POWER SYSTEM

CHANNEL: CHNL# 1: [ANGL 10[13.800]10]
TIME INTERVAL: 5.0000 - 8.0000 SEC.

MODAL COMPONENTS

COMP. NO	EIGENVALUE REAL	EIGENVALUE IMAGINARY	EIGENVECTOR MAGNITUDE	EIGENVECTOR ANGLE	REMARKS	
1	-0.278956E-03	--	29.499	--	TCNST:	3584.796 SC.
2	-0.142548	4.60065	2.7432	-97.26	FREQ.:	0.732 HZ.
3	-0.138911	2.58799	1.5170	56.43	FREQ.:	0.412 HZ.
4	-0.474870	7.44802	1.3504	-28.15	FREQ.:	1.185 HZ.
5	-2.42641	14.7857	0.51683E-01	87.87	FREQ.:	2.353 HZ.
6	-0.351370E-01	9.56082	0.38298E-01	-89.65	FREQ.:	1.522 HZ.
7	-1.81245	18.8700	0.27576E-01	173.52	FREQ.:	3.003 HZ.
8	-1.33207	25.9890	0.10583E-01	-166.39	FREQ.:	4.136 HZ.

CHANNEL: CHNL# 2: [ANGL 11[13.800]11]
TIME INTERVAL: 5.0000 - 8.0000 SEC.

MODAL COMPONENTS

COMP. NO	EIGENVALUE REAL	EIGENVALUE IMAGINARY	EIGENVECTOR MAGNITUDE	EIGENVECTOR ANGLE	REMARKS	
1	-0.510289E-02	--	63.271	--	TCNST:	195.968 SC.
2	-0.417424E-01	4.46315	2.4732	-84.37	FREQ.:	0.710 HZ.
3	0.786150E-01	2.18944	1.1559	85.87	FREQ.:	0.348 HZ.
4	-0.201069	7.47134	0.77419	-31.74	FREQ.:	1.189 HZ.
5	-0.566341	9.36728	0.18727	27.50	FREQ.:	1.491 HZ.
6	-1.68539	22.1496	0.12866E-01	114.46	FREQ.:	3.525 HZ.
7	-1.43488	28.2231	0.97781E-02	153.05	FREQ.:	4.492 HZ.
8	-0.296489	16.6908	0.44472E-02	96.24	FREQ.:	2.656 HZ.
9	-0.569327	31.1198	0.40502E-02	-146.65	FREQ.:	4.953 HZ.

CHANNEL: CHNL# 3: [ANGL 12[13.800]12]
TIME INTERVAL: 5.0000 - 8.0000 SEC.

MODAL COMPONENTS

COMP. NO	EIGENVALUE REAL	EIGENVALUE IMAGINARY	EIGENVECTOR MAGNITUDE	EIGENVECTOR ANGLE	REMARKS	
1	0.120129E-01	--	25.474	--		
2	-0.224071	4.42810	6.3019	-80.05	FREQ.:	0.705 HZ.
3	-0.403782	6.56219	0.91640	53.97	FREQ.:	1.044 HZ.
4	-1.88989	9.44050	0.11416	-99.54	FREQ.:	1.503 HZ.
5	-0.451731	11.5178	0.82240E-02	88.67	FREQ.:	1.833 HZ.
6	-0.703488	24.2716	0.89870E-03	-33.14	FREQ.:	3.863 HZ.
7	0.405292	30.2366	0.15052E-03	-41.34	FREQ.:	4.812 HZ.
8	1.02107	16.4512	0.64234E-04	120.06	FREQ.:	2.618 HZ.
9	5.82311	--	-0.57121E-08	--		

CHANNEL: CHNL# 4: [ANGL 13[13.800]13]
TIME INTERVAL: 5.0000 - 8.0000 SEC.

MODAL COMPONENTS

COMP. NO	EIGENVALUE REAL	EIGENVALUE IMAGINARY	EIGENVECTOR MAGNITUDE	EIGENVECTOR ANGLE	REMARKS	
1	-0.186451	4.51113	7.0352	-87.20	FREQ.:	0.718 HZ.
2	-0.355887E-01	--	2.0979	--	TCNST:	28.099 SC.
3	-0.478384	7.10580	0.63788	-28.52	FREQ.:	1.131 HZ.
4	-4.48942	21.0699	0.45096E-02	-61.50	FREQ.:	3.353 HZ.
5	-0.678177E-01	11.1883	0.38739E-02	164.14	FREQ.:	1.781 HZ.
6	-2.46134	25.2680	0.18574E-02	57.89	FREQ.:	4.022 HZ.
7	2.27517	1.94711	0.76226E-03	82.38	FREQ.:	0.310 HZ.
8	0.185625	15.4756	0.13324E-03	-100.38	FREQ.:	2.463 HZ.

CHANNEL: CHNL# 5: [ANGL 14[13.800]14]
TIME INTERVAL: 5.0000 - 8.0000 SEC.

MODAL COMPONENTS

COMP. NO	EIGENVALUE REAL	EIGENVALUE IMAGINARY	EIGENVECTOR MAGNITUDE	EIGENVECTOR ANGLE	REMARKS	
1	-0.833470E-01	--	49.389	--	TCNST:	11.998 SC.
2	-0.439009	1.83724	14.838	113.49	FREQ.:	0.292 HZ.
3	-0.389071	4.14232	6.9844	-64.95	FREQ.:	0.659 HZ.
4	-6.56315	9.08114	1.1684	125.60	FREQ.:	1.445 HZ.
5	-0.381136	6.80856	0.88145	13.49	FREQ.:	1.084 HZ.
6	0.108472	11.2000	0.28153E-01	-28.58	FREQ.:	1.783 HZ.
7	-0.855451E-01	24.7264	0.15597E-01	-169.75	FREQ.:	3.935 HZ.
8	0.799616	16.2997	0.49901E-02	173.54	FREQ.:	2.594 HZ.

CHANNEL: CHNL# 6: [ANGL 15[13.800]15]
TIME INTERVAL: 5.0000 - 8.0000 SEC.

MODAL COMPONENTS

COMP. NO	EIGENVALUE REAL	EIGENVALUE IMAGINARY	EIGENVECTOR MAGNITUDE	EIGENVECTOR ANGLE	REMARKS	
1	-0.709167E-01	--	31.512	--	TCNST:	14.101 SC.
2	-0.332358	2.08838	11.864	96.44	FREQ.:	0.332 HZ.
3	-0.370764	4.39679	7.2445	-85.26	FREQ.:	0.700 HZ.
4	-0.411237	6.85270	0.85265	5.16	FREQ.:	1.091 HZ.
5	-14.2156	10.4801	0.33163E-01	101.12	FREQ.:	1.668 HZ.
6	-1.15587	25.6956	0.20866E-02	-137.02	FREQ.:	4.090 HZ.
7	-0.874750	30.5093	0.15074E-02	-106.46	FREQ.:	4.856 HZ.
8	0.143428	11.1140	0.13262E-02	-151.51	FREQ.:	1.769 HZ.
9	0.589773	16.2655	0.58466E-03	173.60	FREQ.:	2.589 HZ.

CHANNEL: CHNL# 7: [ANGL 16[13.800]16]
TIME INTERVAL: 5.0000 - 8.0000 SEC.

MODAL COMPONENTS

COMP. NO	EIGENVALUE REAL	EIGENVALUE IMAGINARY	EIGENVECTOR MAGNITUDE	EIGENVECTOR ANGLE	REMARKS	
1	-0.646919E-01	--	61.475	--	TCNST:	15.458 SC.
2	-0.355253	1.79951	12.585	115.17	FREQ.:	0.286 HZ.
3	-0.459509	4.22428	6.3555	-78.85	FREQ.:	0.672 HZ.
4	-9.95993	23.7225	0.77774	130.08	FREQ.:	3.776 HZ.
5	-0.354994	6.82971	0.77366	12.49	FREQ.:	1.087 HZ.
6	-1.64309	24.2887	0.12338	-33.45	FREQ.:	3.866 HZ.
7	-0.213469	15.8709	0.23892E-01	5.11	FREQ.:	2.526 HZ.
8	-0.880073E-02	11.0096	0.23432E-01	63.65	FREQ.:	1.752 HZ.

Figure B. 1. Prony analysis results

Real and imaginary parts of eigenvalues in sec^{-1} and rad/sec denote the mode damping and frequency, respectively. The last column in Figure B.1 provides the frequency in Hz or equivalent time constant in seconds if the eigenvalue is a complex conjugate or a negative real, respectively.

Appendix C Python Code

C.1 Mento Carlo Method

```
import os, sys, xlrd, random, re, os.path
import numpy as np
import matplotlib
import glob
import time
# Get installed location of latest PSS(R)E version
def latest_psse_location():
    import _winreg
    ptiloc = r"SOFTWARE\PTI"
    ptikey = _winreg.OpenKey(_winreg.HKEY_LOCAL_MACHINE, ptiloc, 0, _winreg.KEY_READ)
    ptikeyinfo = _winreg.QueryInfoKey(ptikey)
    numptisubkeys = ptikeyinfo[0]
    vdict = {}
    for i in range(numptisubkeys):
        vernum = _winreg.EnumKey(ptikey, i)
        try:
            n = int(vernum[-2:])
            vdict[n]=vernum
        except:
            pass
    vers = vdict.keys()
    vers.sort()
    k = vers[-1]
    lver = vdict[k]
```

```
    lverloc = ptiloc + "\\" + lver + "\\" + "Product Paths"
    lverkey = _winreg.OpenKey(_winreg.HKEY_LOCAL_MACHINE, lverloc, 0, _winreg.KEY_READ)
    lverdir, stype = _winreg.QueryValueEx(lverkey, 'PsseInstallPath')
    _winreg.CloseKey(ptikey)
    _winreg.CloseKey(lverkey)
    return lverdir
pssedir = latest_psse_location()
pssedir = str(pssedir)              # convert unicode to str
pssbindir  = os.path.join(pssedir,'PSSBIN')
# Check if running from Python Interpreter
exename = sys.executable
p, nx   = os.path.split(exename)
nx      = nx.lower()
if nx in ['python.exe', 'pythonw.exe']:
    os.environ['PATH'] = pssbindir + ';' + os.environ['PATH']
    sys.path.insert(0,pssbindir)
import redirect
redirect.psse2py()
import psspy
import dyntools

def prepare_dynamic_simulation():
    ierr = psspy.psseinit(buses=150000)
    psspy.case(savfile)
    # Convert loads
    psspy.conl(0,1,1,[0,0],[100.0,0.0,0.0,100.0])
    psspy.conl(0,1,2,[0,0],[100.0,0.0,0.0,100.0])
    psspy.conl(0,1,3,[0,0],[100.0,0.0,0.0,100.0])
```

```
    # Convert generators
    psspy.cong()

    # Solve for dynamics
    psspy.ordr(0)
    psspy.fact()
    psspy.tysl(0)

    # Save converted case
    psspy.save(savcfile)

def col2array(book):
    first_sheet = book.sheet_by_index(0)
    cells = first_sheet.col_slice(colx=1, start_rowx=4)
    col_list = []
    for cell in cells:
        col_list.append(cell.value)
    col_array = np.asarray(col_list)
    return col_array

def mc_sampling(n, n_range_pairs):
    # inputs:
    # n: number of variables, int
    # n_range_pairs: [x1_min, x1_max.....xn_min, xn_max], 2n length expected, list
    # return:
    # sample: list with a length of n

    if len(n_range_pairs) != 2 * n:
```

```
        print("mc_sampling error: n_range_paris length is not correct")
        exit()
    sample = []
    for i in range(n):
        sample.append(random.uniform(n_range_pairs[i * 2], n_range_pairs[i * 2 + 1]))
    return sample


def run_dynamic_simulation(snp):


    ierr = psspy.psseinit(buses=150000)  # choose here bus numbers you want
    print 'Simulation executed at time', time.ctime()
    print 'The snapshot file is', snp + '.'
    process_id = os.getpid()
    print("Process ID:{}".format(process_id))
    psspy.lines_per_page_one_device(1,90)
    psspy.progress_output(2,out_files + 'progress report ' + snp + '.txt',[0,0])
    psspy.case(savcfile)
    psspy.rstr(out_files + snp + '.snp')
    psspy.dynamics_solution_param_2()
    psspy.voltage_channel([1,17,1,27],"") # only add bus 154 voltage as channel 42 using
different API
    psspy.strt(1,out_files + snp + '.out')
    psspy.run(0, 1.0,1000,1,0)
    psspy.dist_bus_fault(37,1, 345.0,[0.0,-0.2E+10])
    psspy.run(0, 1.1,1000,1,0)
    psspy.dist_clear_fault(1)
    psspy.run(0, 10.0,1000,1,0)
    psspy.lines_per_page_one_device(2,10000000)
    psspy.progress_output(1,"",[0,0])
```

```
    print 'Simulation finished at time', time.ctime()
    print 'Dynamic simulation for SNP', snp + ' complete! \n\n'

# 1. Data extraction/information

def test_data_extraction(chnfobj):

    print '\n Testing call to xlsout'
    chnfobj.xlsout(channels=1,show=False)

# 6. One trace in each subplot, more than one trace in one plot and all trace in one plot for top five

def test_subplots_trace_all(chnfobj):
    chnfobj.set_plot_page_options(size='letter', orientation='portrait')
    chnfobj.set_plot_markers('square', 'triangle_up', 'thin_diamond', 'plus', 'x',
                             'circle', 'star', 'hexagon1')
    chnfobj.set_plot_line_styles('solid', 'dashed', 'dashdot', 'dotted')
    chnfobj.set_plot_line_colors('blue', 'red', 'black', 'green', 'cyan', 'magenta', 'pink', 'purple')

    optnfmt = {'rows':3,'columns':2,'dpi':300,'showttl':True, 'showoutfnam':True, 'showlogo':True,
               'legendtype':1, 'addmarker':True}

    optnchn1 = {1:{'chns':{a:1}},2:{'chns':{b:1}},3:{'chns':{c:1}},4:{'chns':{d:1}},5:{'chns':{e:1}}}
    pn,x = os.path.splitext(a)
    pltfile1 = pn+'.png'
```

```
figfiles1 = chnfobj.xyplots(optnchn1,optnfmt,pltfile1)

chnfobj.set_plot_legend_options(loc='upper left', borderpad=0.2, labelspacing=0.5,
                    handlelength=1.5, handletextpad=0.5, fontsize=8, frame=False)

optnchn2 = {1:{'chns':{a:1, base_out:1}},
          2:{'chns':{b:1, base_out:1}},
          3:{'chns':{c:1, base_out:1}},
          4:{'chns':{d:1, base_out:1}},
          5:{'chns':{e:1, base_out:1}},
         }
pn,x     = os.path.splitext(b)
pltfile2 = pn+'.png'

figfiles2 = chnfobj.xyplots(optnchn2,optnfmt,pltfile2)

optnchn3 = {1:{'chns':{a:1, b:1 ,c:1,d:1,e:1, base_out:1}},
         }
pn,x     = os.path.splitext(c)
pltfile3 = pn+'.png'

figfiles3 = chnfobj.xyplots(optnchn3,optnfmt,pltfile3)

figfiles = figfiles1[:]
figfiles.extend(figfiles2)
figfiles.extend(figfiles3)
if figfiles:
   print 'Plot fils saved:'
   for f in figfiles:
```

```
        print '    ', f

def test6():

    chnf = dyntools.CHNF(outlst)

    test_subplots_trace_all(chnf)
    chnf.plots_show()

def run_full_system():
    psspy.psseinit(buses=80000)

psspy.read(0,r"""C:\Users\guor\Desktop\PSSE_Python\IEEE68_full_system_raw_data.
raw""")
    psspy.fnsl([0,0,0,0,1,1,99,0])
    psspy.cong(0)
    psspy.conl(0,1,1,[0,0],[ 100.0,0.0,0.0, 100.0])
    psspy.conl(0,1,2,[0,0],[ 100.0,0.0,0.0, 100.0])
    psspy.conl(0,1,3,[0,0],[ 100.0,0.0,0.0, 100.0])
    psspy.ordr(0)
    psspy.fact()
    psspy.tysl(0)

psspy.save(r"""C:\Users\guor\Desktop\PSSE_Python\output\IEEE68_full_system_raw_
data.sav""")

    psspy.lines_per_page_one_device(1,90)
    psspy.progress_output(2,r"""C:\Users\guor\Desktop\PSSE_Python\output\progress
report 0.txt""",[0,0])
```

```
psspy.case(r"""C:\Users\guor\Desktop\PSSE_Python\output\IEEE68_full_system_raw_data.sav""")

psspy.dyre_new([1,1,1,1],r"""C:\Users\guor\Desktop\PSSE_Python\IEEE68_full_system_dynamic_data.dyr""","","","")
    psspy.dynamics_solution_param_2()
    psspy.bsys(1,0,[0.0,0.0],0,[],1,[27],0,[],0,[])
    psspy.chsb(1,0,[1,17,1,1,13,0])

psspy.snap([498,177,17,1,1],r"""C:\Users\guor\Desktop\PSSE_Python\output\IEEE68_full_system.snp""")



psspy.case(r"""C:\Users\guor\Desktop\PSSE_Python\output\IEEE68_full_system_raw_data.sav""")

psspy.rstr(r"""C:\Users\guor\Desktop\PSSE_Python\output\IEEE68_full_system.snp""")

psspy.strt(0,r"""C:\Users\guor\Desktop\PSSE_Python\output\IEEE68_full_system.out""")
    psspy.run(0, 1.0,1000,1,0)
    psspy.dist_bus_fault(37,1, 345.0,[0.0,-0.2E+10])
    psspy.run(0, 1.1,1000,1,0)
    psspy.dist_clear_fault(1)
    psspy.run(0, 10.0,1000,1,0)
    psspy.lines_per_page_one_device(2,10000000)
    psspy.progress_output(1,"",[0,0])
    chnf = dyntools.CHNF(r"""C:\Users\guor\Desktop\PSSE_Python\output\IEEE68_full_system.out""")
    test_data_extraction(chnf)
```

```
# Files Used

study_folder = r'C:\Users\guor\Desktop\PSSE_Python\\'
savfile = os.path.join(study_folder,'IEEE68_equivalent_system_data.sav')
out_files = r"""C:\Users\guor\Desktop\PSSE_Python\output\\"""
savcfile = os.path.join(out_files,'IEEE68_equivalent_system_data_cnv.sav')
outfilepath = os.path.join(study_folder,'*.out')

if __name__ == '__main__':

    FILE_PATH = r"C:\Users\guor\Desktop\PSSE_Python\output"
    # delete all files from last run
    for root, dirs, files in os.walk(FILE_PATH):
        for this_file in files:
            os.remove(os.path.join(root, this_file))
    # run base case
    run_full_system()
    # read base file
    BASE_FILE = "IEEE68_full_system.xlsx"
    book = xlrd.open_workbook(FILE_PATH + "\\" + BASE_FILE)
    base_array = col2array(book)
    book.release_resources()
    del book

    TOP_N = 5
    top_n_list = []
    WORST_N = 7
    worst_n_list = []
    best_evaluation_history = []
```

```
    saved_list = None
    K1 = 5
    K2 = 5
    prepare_dynamic_simulation()

    print 'Total Simulation using serial processing strated at time ' + time.ctime()
    start = time.time()

    for x in range(1, 50+1):
        [a, b, c, d] = mc_sampling(4,[1,10,0,3,1,10,0,3])
        ierr = psspy.psseinit(buses=80000)
        psspy.case(savfile)
        # creat snap file

psspy.dyre_new([1,1,1,1],r"""C:\Users\guor\Desktop\PSSE_Python\IEEE68_equivalent
_system_dynamic_data.dyr""","","","")
        psspy.change_plmod_con(53,r"""1""",r"""GENROU""",5, a)
        psspy.change_plmod_con(53,r"""1""",r"""GENROU""",6, b)
        psspy.change_plmod_con(61,r"""1""",r"""GENROU""",5, c)
        psspy.change_plmod_con(61,r"""1""",r"""GENROU""",6, d)
        psspy.snap([329,118,9,0,0],"C:\Users\guor\Desktop\PSSE_Python\output\\" + str(x)
+  '.snp')

        # run simulation
        run_dynamic_simulation(str(x))

        chnf = dyntools.CHNF("C:\Users\guor\Desktop\PSSE_Python\output\\" + str(x) +
'.out')
        test_data_extraction(chnf)
```

```
# ==============choose top 5================

# read compare file
book = xlrd.open_workbook("C:\Users\guor\Desktop\PSSE_Python\output\\" + str(x) + '.xlsx')
compare_array = col2array(book)
book.release_resources()
del book

# compare
if len(base_array) != len(compare_array):
    break
evaluation_value = sum((base_array - compare_array) ** 2)
top_n_list.append([evaluation_value, [a, b, c, d], x, "C:\Users\guor\Desktop\PSSE_Python\output\\" + str(x) + '.out'])
top_n_list.sort()
worst_n_list.append([evaluation_value, [a, b, c, d], x, "C:\Users\guor\Desktop\PSSE_Python\output\\" + str(x) + '.out'])
worst_n_list.sort(reverse=True)
best_evaluation_history.append(top_n_list[0][0])
# first N
if len(top_n_list) <= TOP_N:
    continue
# new minimum
if evaluation_value <= top_n_list[-1][0]:
    top_n_list.pop()
# worst N
if len(worst_n_list) <= WORST_N:
    continue
```

```
        # new maximum
        if evaluation_value >= worst_n_list[-1][0]:
            worst_n_list.pop()
    print '\n Top Three: \n'
    print(top_n_list)

    print '\n Worst Three: \n'
    print(worst_n_list)
    end = time.time()
    print '\nProcessing took ' + str(end - start) + ' seconds using serial processing\n'
    print 'Total Simulation finished at time ' + time.ctime()
    print '\nEnd of Dynamic Simulation \n'
    matplotlib.pyplot.figure(4)
    matplotlib.pyplot.plot(best_evaluation_history)
    # plot minimum
    outlst = []  # creat object
    for i in range(len(top_n_list)):
        outlst.append(top_n_list[i][3])
    print outlst
    [a,b,c,d,e] = outlst
    base_out                                                                    =
r"""C:\Users\guor\Desktop\PSSE_Python\output\IEEE68_full_system.out"""
    outlst.append(base_out)
    test6()

    # plot maximum
    outlst = []
    for i in range(len(worst_n_list)):
        outlst.append(worst_n_list[i][3])
```

```
    print outlst
    [a,b,c,d,e,g,h] = outlst
    base_out =
r"""C:\Users\guor\Desktop\PSSE_Python\output\IEEE68_full_system.out"""
    outlst.append(base_out)
        test6()
```

C.2 Nelder-Mead Method

```
import os, sys, xlrd, random, re, os.path
import numpy as np
import matplotlib
import glob
import time
from scipy.optimize import minimize
# Get installed location of latest PSS(R)E version
def latest_psse_location():
    import _winreg
    ptiloc = r"SOFTWARE\PTI"
    ptikey = _winreg.OpenKey(_winreg.HKEY_LOCAL_MACHINE, ptiloc, 0,
_winreg.KEY_READ)
    ptikeyinfo = _winreg.QueryInfoKey(ptikey)
    numptisubkeys = ptikeyinfo[0]
    vdict = {}
    for i in range(numptisubkeys):
        vernum = _winreg.EnumKey(ptikey, i)
        try:
            n = int(vernum[-2:])
            vdict[n]=vernum
        except:
            pass

    vers = vdict.keys()
    vers.sort()
    k = vers[-1]
    lver = vdict[k]
```

```
    lverloc = ptiloc + "\\" + lver + "\\" + "Product Paths"
    lverkey = _winreg.OpenKey(_winreg.HKEY_LOCAL_MACHINE, lverloc, 0, _winreg.KEY_READ)
    lverdir, stype = _winreg.QueryValueEx(lverkey, 'PsseInstallPath')
    _winreg.CloseKey(ptikey)
    _winreg.CloseKey(lverkey)
    return lverdir

pssedir = latest_psse_location()
pssedir = str(pssedir)              # convert unicode to str
pssbindir  = os.path.join(pssedir,'PSSBIN')

# Check if running from Python Interpreter
exename = sys.executable
p, nx   = os.path.split(exename)
nx      = nx.lower()
if nx in ['python.exe', 'pythonw.exe']:
    os.environ['PATH'] = pssbindir + ';' + os.environ['PATH']
    sys.path.insert(0,pssbindir)

import redirect
redirect.psse2py()
import psspy
import dyntools

def prepare_dynamic_simulation():
    ierr = psspy.psseinit(buses=150000)
    psspy.case(savfile)
    # Convert loads
```

```
    psspy.conl(0,1,1,[0,0],[100.0,0.0,0.0,100.0])
    psspy.conl(0,1,2,[0,0],[100.0,0.0,0.0,100.0])
    psspy.conl(0,1,3,[0,0],[100.0,0.0,0.0,100.0])

    # Convert generators
    psspy.cong()

    # Solve for dynamics
    psspy.ordr(0)
    psspy.fact()
    psspy.tysl(0)

    # Save converted case
    psspy.save(savcfile)

def col2array(book):
    first_sheet = book.sheet_by_index(0)
    cells = first_sheet.col_slice(colx=1, start_rowx=4)
    col_list = []
    for cell in cells:
        col_list.append(cell.value)
    col_array = np.asarray(col_list)
    return col_array

def run_dynamic_simulation(snp):

    ierr = psspy.psseinit(buses=150000)  # choose here bus numbers you want
    print 'Simulation executed at time', time.ctime()
    print 'The snapshot file is', snp + '.'
```

```
    process_id = os.getpid()
    print("Process ID:{}".format(process_id))
    psspy.lines_per_page_one_device(1,90)
    psspy.progress_output(2,out_files + 'progress report ' + snp + '.txt',[0,0])
    psspy.case(savcfile)
    psspy.rstr(out_files + snp + '.snp')
    psspy.dynamics_solution_param_2()
    psspy.voltage_channel([1,17,1,27],"") # only add bus 154 voltage as channel 42 using
different API
    psspy.strt(1,out_files + snp + '.out')
    psspy.run(0, 1.0,1000,1,0)
    psspy.dist_bus_fault(37,1, 345.0,[0.0,-0.2E+10])
    psspy.run(0, 1.1,1000,1,0)
    psspy.dist_clear_fault(1)
    psspy.run(0, 10.0,1000,1,0)
    psspy.lines_per_page_one_device(2,10000000)
    psspy.progress_output(1,"",[0,0])
    print 'Simulation finished at time', time.ctime()
    print 'Dynamic simulation for SNP', snp + ' complete! \n\n'

# 1. Data extraction/information

def test_data_extraction(chnfobj):

    print '\n Testing call to xlsout'
    chnfobj.xlsout(channels=1,show=False)

def run_full_system():
    psspy.psseinit(buses=80000)
```

```
psspy.read(0,r"""C:\Users\guor\Desktop\PSSE_Python\IEEE68_full_system_raw_data.raw""")
    psspy.fnsl([0,0,0,0,1,1,99,0])
    psspy.cong(0)
    psspy.conl(0,1,1,[0,0],[ 100.0,0.0,0.0, 100.0])
    psspy.conl(0,1,2,[0,0],[ 100.0,0.0,0.0, 100.0])
    psspy.conl(0,1,3,[0,0],[ 100.0,0.0,0.0, 100.0])
    psspy.ordr(0)
    psspy.fact()
    psspy.tysl(0)
    psspy.save(out_files+'IEEE68_full_system_raw_data.sav')

    psspy.lines_per_page_one_device(1,90)
    psspy.progress_output(2,out_files+'progress report 0.txt',[0,0])
    psspy.case(out_files+'IEEE68_full_system_raw_data.sav')

psspy.dyre_new([1,1,1,1],r"""C:\Users\guor\Desktop\PSSE_Python\IEEE68_full_system_dynamic_data.dyr""","","","")
    psspy.dynamics_solution_param_2()
    psspy.bsys(1,0,[0.0,0.0],0,[],1,[27],0,[],0,[])
    psspy.chsb(1,0,[1,17,1,1,13,0])
    psspy.snap([498,177,17,1,1],out_files+'IEEE68_full_system.snp')

    psspy.case(out_files+'IEEE68_full_system_raw_data.sav')
    psspy.rstr(out_files+'IEEE68_full_system.snp')
    psspy.strt(0,out_files+'IEEE68_full_system.out')
    psspy.run(0, 1.0,1000,1,0)
    psspy.dist_bus_fault(37,1, 345.0,[0.0,-0.2E+10])
    psspy.run(0, 1.1,1000,1,0)
```

```
    psspy.dist_clear_fault(1)
    psspy.run(0, 10.0,1000,1,0)
    psspy.lines_per_page_one_device(2,10000000)
    psspy.progress_output(1,"",[0,0])
    chnf = dyntools.CHNF(out_files+'IEEE68_full_system.out')
    test_data_extraction(chnf)

# Files Used

study_folder = r'C:\Users\guor\Desktop\PSSE_Python\\'
savfile = os.path.join(study_folder,'IEEE68_equivalent_system_data.sav')
out_files = r"""C:\Users\guor\Desktop\PSSE_Python\output7\\"""
savcfile = os.path.join(out_files,'IEEE68_equivalent_system_data_cnv.sav')
outfilepath = os.path.join(out_files,'*.out')

def losfunc(x):
    global counter
    # run base case
    run_full_system()
    # read base file
    book = xlrd.open_workbook(out_files+'IEEE68_full_system.xlsx')
    base_array = col2array(book)
    book.release_resources()
    del book
    # run compare case
    prepare_dynamic_simulation()
    ierr = psspy.psseinit(buses=80000)
    psspy.case(savfile)
```

```
    [a,b,c,d]=x
    if a < 0 or b < 0 or c <0 or d < 0:
        return 10000000000000
    0

    # creat snap file

psspy.dyre_new([1,1,1,1],r"""C:\Users\guor\Desktop\PSSE_Python\IEEE68_equivalent
_system_dynamic_data.dyr""","","","")
    psspy.change_plmod_con(53,r"""1""",r"""GENROU""",5, a)
    psspy.change_plmod_con(53,r"""1""",r"""GENROU""",6, b)
    psspy.change_plmod_con(61,r"""1""",r"""GENROU""",5, c)
    psspy.change_plmod_con(61,r"""1""",r"""GENROU""",6, d)
    psspy.snap([329,118,9,0,0],out_files + str(counter) +  '.snp')

    # run simulation
    run_dynamic_simulation(str(counter))
    chnf = dyntools.CHNF(out_files + str(counter) +  '.out')
    test_data_extraction(chnf)

    # read compare file
    book = xlrd.open_workbook(out_files + str(counter) +  '.xlsx')
    compare_array = col2array(book)
    book.release_resources()
    del book
    counter += 1
    se = sum((base_array - compare_array) ** 2)
    for _ in range(5):
        print("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!")
```

```
    print(a, b, c, d, se)
    for _ in range(5):
        print("##############################################")

    return se
# 6. One trace in each subplot, more than one trace in one plot and all trace in one plot for top five

def test_subplots_trace_all(chnfobj):
    chnfobj.set_plot_page_options(size='letter', orientation='portrait')
    chnfobj.set_plot_markers('square', 'triangle_up', 'thin_diamond', 'plus', 'x',
                             'circle', 'star', 'hexagon1')
    chnfobj.set_plot_line_styles('solid', 'dashed', 'dashdot', 'dotted')
    chnfobj.set_plot_line_colors('blue', 'red', 'black', 'green', 'cyan', 'magenta', 'pink', 'purple')

    optnfmt = {'rows':3,'columns':2,'dpi':300,'showttl':True, 'showoutfnam':True, 'showlogo':True,
               'legendtype':1, 'addmarker':True}

    optnchn1 = {1:{'chns':{a:1}},2:{'chns':{b:1}}}
    pn,x     = os.path.splitext(a)
    pltfile1 = pn+'.png'

    figfiles1 = chnfobj.xyplots(optnchn1,optnfmt,pltfile1)

    chnfobj.set_plot_legend_options(loc='upper left', borderpad=0.2, labelspacing=0.5,
                                    handlelength=1.5, handletextpad=0.5, fontsize=8, frame=False)
```

```
    optnchn2 = {1:{'chns':{a:1, base_out:1}},
                2:{'chns':{b:1, base_out:1}},
               }
    pn,x     = os.path.splitext(b)
    pltfile2 = pn+'.png'

    figfiles2 = chnfobj.xyplots(optnchn2,optnfmt,pltfile2)



    optnchn3 = {1:{'chns':{a:1, b:1 , base_out:1}},
               }
    pn,x     = os.path.splitext(base_out)
    pltfile3 = pn+'.png'

    figfiles3 = chnfobj.xyplots(optnchn3,optnfmt,pltfile3)

    figfiles = figfiles1[:]
    figfiles.extend(figfiles2)
    figfiles.extend(figfiles3)
    if figfiles:
       print 'Plot fils saved:'
       for f in figfiles:
          print '    ', f

def test6():

    chnf = dyntools.CHNF(outlst)

    test_subplots_trace_all(chnf)
```

```
    chnf.plots_show()

if __name__ == '__main__':
    FILE_PATH = r"C:\Users\guor\Desktop\PSSE_Python\output7"

    counter = 1
    # delete all files from last run
    for root, dirs, files in os.walk(FILE_PATH):
        for this_file in files:
            os.remove(os.path.join(root, this_file))

    print 'Simulation using serial processing strated at time ' + time.ctime()
    start = time.time()
    x0=np.array([0.5, 0.2, 0.5, 0.2])
    res=minimize(losfunc,x0,method='nelder-mead',options={'xatol':1e-4, 'disp':True})
    end = time.time()
    print '\nProcessing took ' + str(end - start) + ' seconds using serial processing\n'
    print 'Total Simulation finished at time ' + time.ctime()
    print '\nEnd of Dynamic Simulation \n'
    print(res.x)

    # plot
    file_list = glob.glob(r"C:\Users\guor\Desktop\PSSE_Python\output7\*.out")
    biggest_number = 0
    for file_path in file_list:
        try:
            file_number = int(file_path.split("\\")[-1][:-4])
            if file_number > biggest_number:
                biggest_number = file_number
```

```
        except ValueError:
            continue
    solution_file_path = FILE_PATH + "\\" + str(biggest_number) + ".out"
    print(solution_file_path)
    # plot
    outlst =[] # creat object
    outlst.extend([r"C:\Users\guor\Desktop\PSSE_Python\output7\1.out",
solution_file_path])
    print(outlst)
    [a,b]=outlst
    base_out                                                                =
r"""C:\Users\guor\Desktop\PSSE_Python\output7\IEEE68_full_system.out"""
    outlst.append(base_out)
            test6()
```

C.3 Powell Method

```
import os, sys, xlrd, random, re, os.path
import numpy as np
import matplotlib
import glob
import time
from scipy.optimize import minimize

# Get installed location of latest PSS(R)E version
def latest_psse_location():
    import _winreg
    ptiloc = r"SOFTWARE\PTI"
    ptikey = _winreg.OpenKey(_winreg.HKEY_LOCAL_MACHINE, ptiloc, 0,
_winreg.KEY_READ)
    ptikeyinfo = _winreg.QueryInfoKey(ptikey)
    numptisubkeys = ptikeyinfo[0]
    vdict = {}
    for i in range(numptisubkeys):
        vernum = _winreg.EnumKey(ptikey, i)
        try:
            n = int(vernum[-2:])
            vdict[n]=vernum
        except:
            pass

    vers = vdict.keys()
    vers.sort()
    k = vers[-1]
```

```
    lver = vdict[k]
    lverloc = ptiloc + "\\" + lver + "\\" + "Product Paths"
    lverkey = _winreg.OpenKey(_winreg.HKEY_LOCAL_MACHINE, lverloc, 0,
_winreg.KEY_READ)
    lverdir, stype = _winreg.QueryValueEx(lverkey, 'PsseInstallPath')
    _winreg.CloseKey(ptikey)
    _winreg.CloseKey(lverkey)
    return lverdir

pssedir = latest_psse_location()
pssedir = str(pssedir)              # convert unicode to str
pssbindir  = os.path.join(pssedir,'PSSBIN')

# Check if running from Python Interpreter
exename = sys.executable
p, nx   = os.path.split(exename)
nx      = nx.lower()
if nx in ['python.exe', 'pythonw.exe']:
    os.environ['PATH'] = pssbindir + ';' + os.environ['PATH']
    sys.path.insert(0,pssbindir)

import redirect
redirect.psse2py()
import psspy
import dyntools

def prepare_dynamic_simulation():
    ierr = psspy.psseinit(buses=150000)
    psspy.case(savfile)
```

```
    # Convert loads
    psspy.conl(0,1,1,[0,0],[100.0,0.0,0.0,100.0])
    psspy.conl(0,1,2,[0,0],[100.0,0.0,0.0,100.0])
    psspy.conl(0,1,3,[0,0],[100.0,0.0,0.0,100.0])

    # Convert generators
    psspy.cong()

    # Solve for dynamics
    psspy.ordr(0)
    psspy.fact()
    psspy.tysl(0)

    # Save converted case
    psspy.save(savcfile)

def col2array(book):
    first_sheet = book.sheet_by_index(0)
    cells = first_sheet.col_slice(colx=1, start_rowx=4)
    col_list = []
    for cell in cells:
        col_list.append(cell.value)
    col_array = np.asarray(col_list)
    return col_array

def run_dynamic_simulation(snp):

    ierr = psspy.psseinit(buses=150000)  # choose here bus numbers you want
    print 'Simulation executed at time', time.ctime()
```

```
    print 'The snapshot file is', snp + '.'
    process_id = os.getpid()
    print("Process ID:{}".format(process_id))
    psspy.lines_per_page_one_device(1,90)
    psspy.progress_output(2,out_files + 'progress report ' + snp + '.txt',[0,0])
    psspy.case(savcfile)
    psspy.rstr(out_files + snp + '.snp')
    psspy.dynamics_solution_param_2()
    psspy.voltage_channel([1,17,1,27],"") # only add bus 154 voltage as channel 42 using
different API
    psspy.strt(1,out_files + snp + '.out')
    psspy.run(0, 1.0,1000,1,0)
    psspy.dist_bus_fault(37,1, 345.0,[0.0,-0.2E+10])
    psspy.run(0, 1.1,1000,1,0)
    psspy.dist_clear_fault(1)
    psspy.run(0, 10.0,1000,1,0)
    psspy.lines_per_page_one_device(2,10000000)
    psspy.progress_output(1,"",[0,0])
    print 'Simulation finished at time', time.ctime()
    print 'Dynamic simulation for SNP', snp + ' complete! \n\n'

# 1. Data extraction/information

def test_data_extraction(chnfobj):

    print '\n Testing call to xlsout'
    chnfobj.xlsout(channels=1,show=False)

def run_full_system():
```

```
    psspy.psseinit(buses=80000)

psspy.read(0,r"""C:\Users\guor\Desktop\PSSE_Python\IEEE68_full_system_raw_data.raw""")
    psspy.fnsl([0,0,0,0,1,1,99,0])
    psspy.cong(0)
    psspy.conl(0,1,1,[0,0],[ 100.0,0.0,0.0, 100.0])
    psspy.conl(0,1,2,[0,0],[ 100.0,0.0,0.0, 100.0])
    psspy.conl(0,1,3,[0,0],[ 100.0,0.0,0.0, 100.0])
    psspy.ordr(0)
    psspy.fact()
    psspy.tysl(0)
    psspy.save(out_files+'IEEE68_full_system_raw_data.sav')

    psspy.lines_per_page_one_device(1,90)
    psspy.progress_output(2,out_files+'progress report 0.txt',[0,0])
    psspy.case(out_files+'IEEE68_full_system_raw_data.sav')

psspy.dyre_new([1,1,1,1],r"""C:\Users\guor\Desktop\PSSE_Python\IEEE68_full_system_dynamic_data.dyr""","","","")
    psspy.dynamics_solution_param_2()
    psspy.bsys(1,0,[0.0,0.0],0,[],1,[27],0,[],0,[])
    psspy.chsb(1,0,[1,17,1,1,13,0])
    psspy.snap([498,177,17,1,1],out_files+'IEEE68_full_system.snp')

    psspy.case(out_files+'IEEE68_full_system_raw_data.sav')
    psspy.rstr(out_files+'IEEE68_full_system.snp')
    psspy.strt(0,out_files+'IEEE68_full_system.out')
    psspy.run(0, 1.0,1000,1,0)
    psspy.dist_bus_fault(37,1, 345.0,[0.0,-0.2E+10])
```

```
    psspy.run(0, 1.1,1000,1,0)
    psspy.dist_clear_fault(1)
    psspy.run(0, 10.0,1000,1,0)
    psspy.lines_per_page_one_device(2,10000000)
    psspy.progress_output(1,"",[0,0])
    chnf = dyntools.CHNF(out_files+'IEEE68_full_system.out')
    test_data_extraction(chnf)

# Files Used

study_folder = r'C:\Users\guor\Desktop\PSSE_Python\\'
savfile = os.path.join(study_folder,'IEEE68_equivalent_system_data.sav')
out_files = r"""C:\Users\guor\Desktop\PSSE_Python\output8_3\\"""
savcfile = os.path.join(out_files,'IEEE68_equivalent_system_data_cnv.sav')
outfilepath = os.path.join(out_files,'*.out')

def losfunc(x):
    global counter
    # run base case
    run_full_system()
    # read base file
    book = xlrd.open_workbook(out_files+'IEEE68_full_system.xlsx')
    base_array = col2array(book)
    book.release_resources()
    del book

    # run compare case
    prepare_dynamic_simulation()
    ierr = psspy.psseinit(buses=80000)
```

```
    psspy.case(savfile)

    [a,b,c,d]=x
    if a < 0 or b < 0 or c <0 or d < 0:
        return 10000000000000
    0

    # creat snap file

psspy.dyre_new([1,1,1,1],r"""C:\Users\guor\Desktop\PSSE_Python\IEEE68_equivalent
_system_dynamic_data.dyr""","","","")
    psspy.change_plmod_con(53,r"""1""",r"""GENROU""",5, a)
    psspy.change_plmod_con(53,r"""1""",r"""GENROU""",6, b)
    psspy.change_plmod_con(61,r"""1""",r"""GENROU""",5, c)
    psspy.change_plmod_con(61,r"""1""",r"""GENROU""",6, d)
    psspy.snap([329,118,9,0,0],out_files + str(counter) +  '.snp')

    # run simulation
    run_dynamic_simulation(str(counter))
    chnf = dyntools.CHNF(out_files + str(counter) +  '.out')
    test_data_extraction(chnf)

    # read compare file
    book = xlrd.open_workbook(out_files + str(counter) +  '.xlsx')
    compare_array = col2array(book)
    book.release_resources()
    del book
    counter += 1
    se = sum((base_array - compare_array) ** 2)
```

```
    for _ in range(5):
        print("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!")
    print(a, b, c, d, se)
    for _ in range(5):
        print("############################################")

    return se
# 6. One trace in each subplot, more than one trace in one plot and all trace in one plot for top five

def test_subplots_trace_all(chnfobj):
    chnfobj.set_plot_page_options(size='letter', orientation='portrait')
    chnfobj.set_plot_markers('square', 'triangle_up', 'thin_diamond', 'plus', 'x',
                             'circle', 'star', 'hexagon1')
    chnfobj.set_plot_line_styles('solid', 'dashed', 'dashdot', 'dotted')
    chnfobj.set_plot_line_colors('blue', 'red', 'black', 'green', 'cyan', 'magenta', 'pink', 'purple')

    optnfmt = {'rows':3,'columns':2,'dpi':300,'showttl':True, 'showoutfnam':True, 'showlogo':True,
               'legendtype':1, 'addmarker':True}

    optnchn1 = {1:{'chns':{a:1}},2:{'chns':{b:1}}}
    pn,x = os.path.splitext(a)
    pltfile1 = pn+'.png'

    figfiles1 = chnfobj.xyplots(optnchn1,optnfmt,pltfile1)

    chnfobj.set_plot_legend_options(loc='upper left', borderpad=0.2, labelspacing=0.5,
```

```
                    handlelength=1.5, handletextpad=0.5, fontsize=8, frame=False)

    optnchn2 = {1:{'chns':{a:1, base_out:1}},
                2:{'chns':{b:1, base_out:1}},
               }
    pn,x     = os.path.splitext(b)
    pltfile2 = pn+'.png'

    figfiles2 = chnfobj.xyplots(optnchn2,optnfmt,pltfile2)


    optnchn3 = {1:{'chns':{a:1, b:1 , base_out:1}},
               }
    pn,x     = os.path.splitext(base_out)
    pltfile3 = pn+'.png'

    figfiles3 = chnfobj.xyplots(optnchn3,optnfmt,pltfile3)

    figfiles = figfiles1[:]
    figfiles.extend(figfiles2)
    figfiles.extend(figfiles3)
    if figfiles:
       print 'Plot fils saved:'
       for f in figfiles:
          print '    ', f

def test6():

    chnf = dyntools.CHNF(outlst)
```

```
    test_subplots_trace_all(chnf)
    chnf.plots_show()

if __name__ == '__main__':
    FILE_PATH = r"C:\Users\guor\Desktop\PSSE_Python\output8_3"

    counter = 1
    # delete all files from last run
    for root, dirs, files in os.walk(FILE_PATH):
        for this_file in files:
            os.remove(os.path.join(root, this_file))

    print 'Simulation using serial processing strated at time ' + time.ctime()
    start = time.time()
    x0=np.array([161.875, 24.665, 350, 100])
    res=minimize(losfunc,x0,method='powell',options={'xtol':1e-1, 'disp':True})
    end = time.time()
    print '\nProcessing took ' + str(end - start) + ' seconds using serial processing\n'
    print 'Total Simulation finished at time ' + time.ctime()
    print '\nEnd of Dynamic Simulation \n'
    print(res.x)

    # plot
    file_list = glob.glob(r"C:\Users\guor\Desktop\PSSE_Python\output8_3\*.out")
    biggest_number = 0
    for file_path in file_list:
        try:
            file_number = int(file_path.split("\\")[-1][:-4])
```

```
            if file_number > biggest_number:
                biggest_number = file_number
        except ValueError:
            continue
    solution_file_path = FILE_PATH + "\\" + str(biggest_number) + ".out"
    print(solution_file_path)
    # plot
    outlst =[] # creat object
    outlst.extend([r"C:\Users\guor\Desktop\PSSE_Python\output8_3\1.out",
solution_file_path])
    print(outlst)
    [a,b]=outlst
    base_out                                                                    =
r"""C:\Users\guor\Desktop\PSSE_Python\output8_3\IEEE68_full_system.out"""
    outlst.append(base_out)
            test6()
```