Steady-State Analysis in Transmission System Planning per Latest TPL-001-4 NERC Standard

Marcos Alexander Ayala Zelaya

Thesis submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Master of Science

in

Electrical Engineering

Virgilio A. Centeno, Chair

Jaime De La Reelopez, Co-Chair

Robert P. Broadwater

December 9, 2015

Blacksburg, Virginia

Keywords: electric power systems, transmission system, planning, steady-state, contingencies, NERC, TPL-001-4

Steady-State Analysis in Transmission System Planning per Latest TPL-001-4 NERC Standard

Marcos Alexander Ayala Zelaya

Abstract

Several cascading outages that have occurred in the past have shown the importance of performing appropriate and coordinated transmission system planning between the different Planning Coordinators and Transmission Planners in a power system. In very large, interconnected systems, this coordinated planning is necessary due to the inherent complexity that characterizes non-lineal electric power systems. A major task in transmission system planning is to conduct contingency studies that would determine the consequences in the system when single or multiple of its elements trip due to failure or are disconnected during scheduled maintenance outages. This work develops general programs aimed at facilitating the performance of such contingency studies based on the latest Transmission System Planning Performance Requirement standard issued by NERC, TPL-001-4. The focus of this work is on the requirements for the steady state analysis described in the standard, which includes specific steady state performance planning events. The developed programs have been tested in a few base cases and have demonstrated their ability to facilitate contingency studies with any system regardless of its size.

Dedicado a mi papá, mamá y hermano;

quienes siempre me han demostrado cariño

incondicional a lo largo de mi vida.

Acknowledgements

I would like to express my most sincere gratitude to Dr. Virgilio Centeno, who has helped me not only during my research but also throughout my graduate studies at Virginia Tech. His personal and academic advice has been an invaluable guidance to complete my studies after I had been outside the academia for several years.

To Dr. Jaime De La Ree López, whose life story and passion for teaching, power systems and electrical engineering have been an inspiration to me. I am very thankful because both Dr. De La Ree and Dr. Centeno have supported me in different invaluable ways throughout my studies.

To my new friends from Virginia and old ones from California and El Salvador. True friends are indeed hard to come by, and I really appreciate all the time we have spent together either studying, catching up, encouraging each other, or simply making silly jokes.

I have so many people to be thankful for that it is difficult to mention all of them. From the professors who started my academic formation in El Salvador to the ones who taught me ESL in the US. To the managers and supervisors that despite my personal circumstances believed in giving me an opportunity to achieve my academic and professional goals in the US. None of this would have been possible without countless people that have helped me either directly or indirectly.

Finally, I am very grateful for all the support, advice and love I have received from my family. Without them, I would not be here today, and accomplishing my dream of coming to graduate school would have been an impossible endeavor.

Table of Contents

Abstract ..... **ii**

Acknowledgements ..... **iv**

Table of Contents ..... **v**

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

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

1. Introduction **1**

1.1 North American Reliability Corporation (NERC) 2

1.2 Need for Automation in Transmission Systems Planning 4

1.3 Thesis Overview 5

2. NERC Standards for Transmission Planning **6**

2.1 Ongoing Work on the Transmission Planning Standards 6

2.1.1 Standard TPL-001-0.1: *System Performance under Normal (No Contingency) Conditions (Category A)* 8

2.1.2 Standard TPL-002-0b: *System Performance Following Loss of a Single Bulk Electric System Element (Category B)* 9

2.1.3 Standard TPL-003-0b: *System Performance Following Loss of Two or More Bulk Electric System Elements (Category C)* 10

2.1.4 Standard TPL-004-0a: *System Performance Following Extreme Events Resulting in the Loss of Two or More Bulk Electric System Elements (Category D)* 11

2.1.5 Common Requirements among Standards TPL-001-0.1, TPL-002-0b, TPL-003-0b, and TPL-004-0a 13

2.1.6 Standard TPL-001-4: *Transmission System Planning Performance Requirements* 14

3. Automation in Transmission Systems Planning Simulations 23

3.1 Introduction to Python 23

3.1.1 Python Standard Library 25

3.1.2 Built-in-Functions 25

3.1.3 Modules 26

3.1.4 Important Tools of Python Language 27

3.1.4.1 Functions 27

3.1.4.2 Lists 27

3.1.4.3 File Objects 27

3.2 Description of Developed Application 28

3.2.1 Algorithm for Categories P1, P2, P4 and P7 29

3.2.1.1 Python Initialization 30

3.2.1.2 Base Case 30

3.2.1.3 PSS/E Initialization 31

3.2.1.4 Events Area 31

3.2.1.5 Bus Subsystem Definition 31

3.2.1.6 P1_E1 con Generation 32

3.2.1.7 P1_E1 dfx Generation 33

3.2.1.8 P1_E1 acc Generation 33

3.2.1.9 P1_E1 Results 36

3.2.2 Algorithm for Categories P3 and P6 37

3.2.2.1 Gen Bus & Mach Identification 38

3.2.2.2 Initial Condition Base Case - Loss 39

3.2.2.3 Perform System Adjustments 39

3.2.3 Events and Criteria for Future Research 40

4. Analysis of Results 42

4.1 Power System Element Contingency, System Adjustments, and Overload and Voltage Reports 42

4.1.1 Performance during P1 events 44

4.1.2 Performance during P2 events 46

4.1.3 Performance during P3 events 48

4.1.4 Performance during P4 events 50

4.1.5 Performance during P6 events 51

4.1.6 Performance during P7 events 52

4.2 Coordination between Adjacent Planning Coordinators and Transmission Planners 54

5. Conclusions 59

5.1 Future Research 60

6. References 62

Appendix A Python Codes 65

A.1 Category P1 65
A.2 Category P2 71
A.3 Category P3 75
A.4 Category P4 83
A.5 Category P6_I1 91
A.6 Category P6_I2 98
A.7 Category P6_I3 105
A.8 Category P6_I4 118
A.9 Category P7 125
A.10 Home Script 128

List of Figures

Figure 1-1 NERC Regions Map [2] 3

Figure 3-1 High-Level Language Processed through an Interpreter [15] 24

Figure 3-2 Algorithm for Category P1 Planning Events 29

Figure 3-3 Outline of Evaluation Procedure Using AC 35

Figure 3-4 Algorithm for Category P3 Planning Events 38

Figure 4-1 Modifications Made to Original savnw.sav Case 42

Figure 4-2 Monitored Generation Step-Up Transformers with Unit at Bus 101 Tripped 45

Figure 4-3 Monitored Generation Step-Up Transformers with Unit at Bus 102 Tripped 45

Figure 4-4 Overloads and Voltage Violations Report when Planning Events Trip each Generation Unit 46

Figure 4-5 Contingencies for E1 Events in Category P2 47

Figure 4-6 Results for Events 1 in Category P2 47

Figure 4-7 Generation Step-Up Transformers Flows with Units at Buses 101 and 102 Tripped. 48

Figure 4-8 Overloads and Voltage Violations Report with Unit at Bus 101 Tripped as Initial Condition and Transmission Line between Bus 151 and 152 Tripped 49

Figure 4-9 Local Backup-Breaker Failure Example 50

Figure 4-10 Overloads and Voltage Violations Report with Stuck Breaker at Bus 155 Trying to Clear a Fault in Transmission Line between Buses 155 and 156 51

Figure 4-11 Automatic Switched Shunt Device Operation with Fixed Shunt Reactor at Bus 151 Tripped as Initial Condition 52

Figure 4-12 Overloads and Voltage Violations Report with Fixed Shunt Reactor at Bus 151 Tripped as Initial Condition and A Few Transmission Lines Tripped 52
Figure 4-13 Contingencies for E1 Events in Category P7 53
Figure 4-14 Results for E1 Events in Category P7 53
Figure 4-15 Contingencies for E2 Events in Category P7 54
Figure 4-16 Results for E2 Events in Category P7 54
Figure 4-17 IEEE 300-Bus System Used as Base Case 56
Figure 4-18 Input from the User in the Home.py Program 57
Figure 4-19 Low Voltage Violations in CAESS for E3 Events within VTES - Category P1 57
Figure 4-20 Low Voltage Violations in CAESS for E1 Events within VTES - Category P4 58
Figure 4-21 Low Voltage Violation in CAESS for E1 Event within VTES - Category P6 with Transmission Circuit 192 - 225 Tripped as Initial Condition 58

List of Tables

Table 2-1 Mandatory Standards Subject to Enforcement [5] 6

Table 2-2 Mandatory Transmission Planning (TPL) Standards [5] 8

Table 2-3 System Performance Conditions - Category A [8] 9

Table 2-4 System Performance Conditions - Category B [9] 10

Table 2-5 System Performance Conditions - Category C [10] 11

Table 2-6 System Performance Conditions - Category D [11] 12

Table 2-7 Steady State & Stability Performance Planning Events [12] 22

Table 3-1 Built-in-Functions in Python [17] 25

Table 3-2 Summary of Contingency Events Included in this Work per TPL-001-4 [12] 41

Table 4-1 Steady State Voltages and Post-Contingency Voltage Deviations Limits 44

1. Introduction

Before the advent of the Internet at the end of 20th century, the electric power grid was considered as the largest machine that had ever been invented in the world. Nowadays, it continues to be so important and works so well that usually we only realize how much our lives, economies, and societies rely on it during blackouts or power outages.

One of the principal components in an electric power grid is its transmission system. Among its functions are to deliver energy from generators to the rest of the power system, provide for energy interchange between different electric utilities, and supply energy to the subtransmission and distribution systems [1].

The transmission system should carry out all these functions in a reliable and continuous manner in order to guarantee that the electric demand in the system will be supplied by the available generation at all times. This task can be a challenge considering the dynamic behavior of the electric load for different days, months, and seasons within a year. Moreover, unusual weather patterns can affect the availability of resources such as water and wind, which may reduce the scheduled electric generation from those resources. Further, generation from fossil fuels might become unfeasible if there are unexpected oil price spikes in international markets. Finally, planned and unplanned equipment outages in the electric power system can ultimately affect how the transmission system is able to perform its functions.

It becomes very important, therefore, to conduct adequate transmission system planning studies in advance which take into account all the variables that might play a role during the real

operation of transmission systems. The goal is to ensure that even before such variables, bus voltages in the transmission system remain within a range close to rated values, and transmission lines and transformers are not overloaded. The guidelines and regulations that must be followed to conduct the transmission system planning are typically established by self-regulatory organizations.

1.1 North American Reliability Corporation (NERC)

The North American Electric Reliability Corporation (NERC) is a not-for-profit international regulatory authority whose mission is to assure the reliability of the bulk power system in North America. It is responsible for maintaining power system standards and reliability. Therefore, it is also in charge of developing and enforcing the standards to conduct the transmission system planning [2].

NERC works with eight regional entities to improve the reliability of the bulk power system. These entities account for virtually all the electricity supplied in the United States, Canada, and a portion of Baja California Norte, Mexico (Figure 1-1). The members of the regional entities come from all segments of the electric industry.

The role that NERC has played in the power industry has evolved overtime. It was first created in 1968 as a result of the blackout in the northeastern United States in 1965. This was the first blackout that showed the importance of the cooperation and interaction between electric utilities. Initially, NERC only issued recommendations and information that formed the basis for more reliable and secure planning and operation of the power system. After the 2003 blackout in the

northeastern United States and Canada, however, NERC recommendations and reliability standards have become mandatory [3].

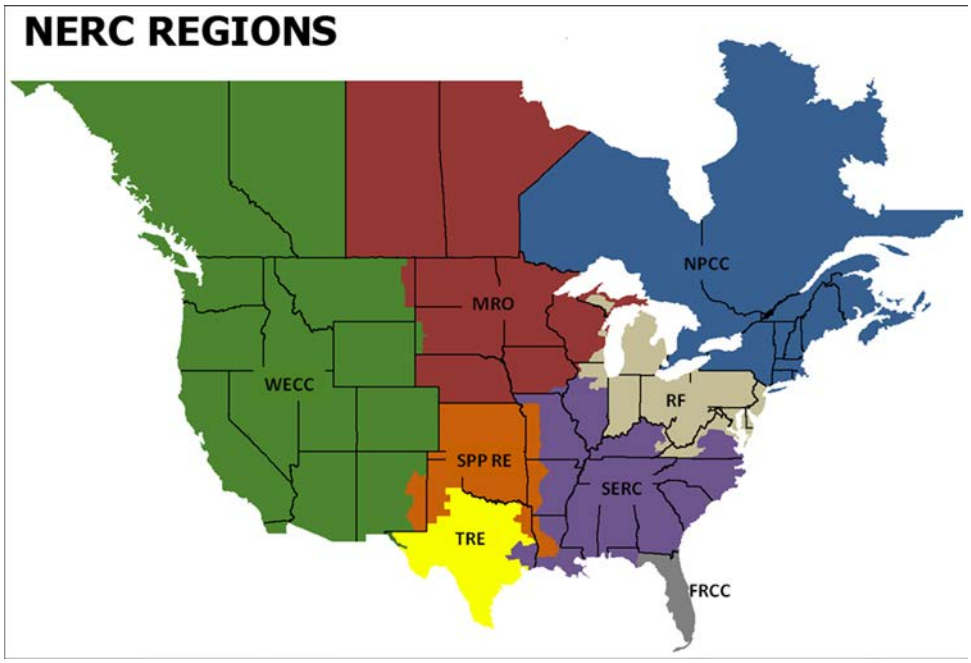

Figure 1-1 NERC Regions Map [2]

The way in which the power system planning is performed has also evolved overtime, along with the restructuring of the electric utility industry in the United States. In the 1990s, large, vertically integrated monopolies in the electric industry were replaced by a horizontal structure with generating, transmission, and distribution companies as separate business facilities [1].

Historically, a vertically integrated utility would plan for generation and transmission needs within its service territory. These projects were developed to satisfy the utility's reliability and economic needs. Under the new horizontal structure, however, the generation and transmission planning has become more complex due to the different parties that now have nondiscriminatory,

open access to the electric system at the generation, transmission and distribution levels, even within the same utility [4].

These changes in the power industry have brought about new challenges in the planning process, where it is still difficult to overcome the traditional mindset of planners. Traditionally, a utility transmission planner was primarily concerned with the transport of bulk generation to load centers without violation of local constraints. With the new, large interconnected systems, transmission planning requires a wide-area perspective, aging infrastructure awareness, a willingness to coordination extensively, an economic mindset, and an ability to effectively integrate new technologies with traditional approaches [4].

The goal of this work is to develop a group of general applications that perform the steady-state contingency studies based on the latest Transmission System Planning Performance Requirement standard issued by NERC, TPL-001-4. These general applications would ease conducting similar studies for any system or base case.

1.2 Need for Automation in Transmission Systems Planning

Due to the high complexity, non-linearity, and large size that characterize typical power systems, it can be daunting to study the performance of the transmission system in order to conduct its planning. This task has become even more challenging with the new large, interconnected transmission systems, which can have thousands of buses, transmission lines, transformers and generators. The North American Eastern Interconnected system alone would require more than 150000 major components in a power flow model [4]. Indeed, it would require a lot of time and

calculations to assess all the variables and contingencies that might affect the transmission system planning. Luckily, nowadays there are software tools that can help automate some of these calculations.

This work uses the capabilities of the software Power System Simulator for Engineering (PSS/E) to perform all the power flow calculations in the steady state analysis for the contingencies per TPL-001-4, and the programming language Python to automate most of the process.

1.3 Thesis Overview

The next chapter presents a brief introduction to the NERC reliability standards for transmission planning. It explains in more detail the latest TPL-001-4 standard. Chapter 3 presents a brief introduction to the Python programming language and its integration with PSS/E. Then the application that has been developed to perform the steady state performance required by TPL-001-4 in the transmission system planning process is described. Chapter 4 presents an analysis of the results obtained with the application developed, and Chapter 5 states the conclusions and proposed future research.

2. NERC Standards for Transmission Planning

To achieve the high reliability that must characterize electric power systems, common standards applicable across the power industry are required. NERC has been responsible for issuing these standards for the bulk power system in North America since its inception in 1968. The Transmission Planning standards are but a small sliver of a large spectrum of reliability standards issued by NERC, as it is shown on Table 2-1.

NERC Reliability Standards
Resource and Demand Balancing (BAL)
Communications (COM)
Critical Infrastructure Protection (CIP)
Emergency Preparedness and Operations (EOP)
Facilities Design, Connections, and Maintenance (FAC)
Interchange Scheduling and Coordination (INT)
Interconnection Reliability Operations and Coordination (IRO)
Modeling, Data, and Analysis (MOD)
Nuclear (NUC)
Personnel Performance, Training, and Qualifications (PER)
Protection and Control (PRC)
Transmission Operations (TOP)
Transmission Planning (TPL)
Voltage and Reactive (VAR)

Table 2-1 Mandatory Standards Subject to Enforcement [5]

2.1 Ongoing Work on the Transmission Planning Standards

On August 14, 2003, one of the worst power systems failures in the last few decades occurred due a cascading outage of transmission and generation facilities in the North American Eastern Interconnection. It led to a blackout that affected approximately 50 million people in eight U.S. states and two Canadian provinces [6]. As a result of the blackout, a U.S.-Canada Power System

Outage Task Force was established to investigate its causes and make recommendations in order to minimize the likelihood and scope of similar events in the future.

Based on the recommendations made by the Final Report of the task force [7], several NERC reliability standards have been either created or updated after the 2003 blackout. In the case of the transmission planning standards, the latest TPL-001-4 standard is the end result after a series of four standards that established different contingency conditions and criteria necessary to perform adequate transmission system planning.

Table 2-2 shows the four reliability standards, each one covering specific normal and emergency conditions categories. These four standards will be completely superseded by TPL-001-4 in January 2016.

As part of its role in the electric power industry, NERC continues to update existing and develop new transmission planning standards in order to include in the planning process the different criteria that might affect the transmission system performance. For instance, the Transmission System Planned Performance for Geomagnetic Disturbance Events standard, TPL-007-1, is currently pending regulatory approval, and it establishes the requirements for transmission system planned performance during geomagnetic disturbance (GMD) events.

Standard Number	Title	First Version Issued	Enforcement Date (Latest Version)	Notes
TPL-001-0.1	System Performance Under Normal (No Contingency) Conditions (Category A)	4/1/2005	5/13/2009	Valid until 12/31/2015
TPL-002-0b	System Performance Following Loss of a Single Bulk Electric System Element (Category B)	2/8/2005	10/24/2011	Valid until 12/31/2015
TPL-003-0b	System Performance Following Loss of Two or More Bulk Electric System Elements (Category C)	2/8/2005	6/20/2013	Valid until 12/31/2015
TPL-004-0a	System Performance Following Extreme Events Resulting in the Loss of Two or More Bulk Electric System Elements (Category D)	4/1/2005	6/20/2013	Valid until 12/31/2015
TPL-001-4	Transmission System Planning Performance Requirements	4/1/2005	1/1/2015	R1 and R7
			1/1/2016	R2 thru R6 and R8

Table 2-2 Mandatory Transmission Planning (TPL) Standards [5]

In the next sections, a brief overview about the existing transmission planning standards will be presented, proceeding then to explain the latest TPL-001-4 standard in more detail.

2.1.1 Standard TPL-001-0.1: *System Performance under Normal (No Contingency) Conditions (Category A)*

This standard has three requirements. The first requirement states that both the Planning Authority and Transmission Planner are responsible for demonstrating through valid assessments that, with all facilities in service in the portion of the interconnected transmission system that belongs to them and under normal (pre-contingency) operating procedures, the transmission system can be operated in such a way that it will be able to supply the projected customer demands and projected Firm Transmission Services at all demand levels over a range of forecast system demands [8].

To meet this requirement, the valid assessments can be studies and/or simulations that must show the system performance following Category A of the Table 2-3 shown below.

Category	Contingencies	System Limits or Impacts		
	Initiating Event(s) and Contingency Element(s)	System Stable and both Thermal and Voltage Limits within Applicable Rating [a]	Loss of Demand or Curtailed Firm Transfers	Cascading Outages
A No Contingencies	All Facilities in Service	Yes	No	No

Table 2-3 System Performance Conditions - Category A [8]

This requirement also mandates the establishment of normal (pre-contingency) operating procedures.

The second and third requirements are explained in the section 2.1.5 as they are common among the standards TPL-001-0.1, TPL-002-0b, TPL-003-0b, and TPL-004-0a.

2.1.2 Standard TPL-002-0b: *System Performance Following Loss of a Single Bulk Electric System Element (Category B)*

This standard has three requirements. The first requirement states that both the Planning Authority and Transmission Planner are responsible for demonstrating through valid assessments in the portion of the interconnected transmission system that belongs to them that the transmission system can be operated in such a way that it will be able to supply the projected customer demands and projected Firm Transmission Services at all demand levels over a range of forecast system demands, under the contingency conditions defined in Category B of the Table 2-4 shown below [9].

Category	Contingencies	System Limits or Impacts		
	Initiating Event(s) and Contingency Element(s)	System Stable and both Thermal and Voltage Limits within Applicable Rating [a]	Loss of Demand or Curtailed Firm Transfers	Cascading Outages
B Event resulting in the loss of a single element.	Single Line Ground (SLG) or 3-Phase (3Ø) Fault, with Normal Clearing: 1. Generator 2. Transmission Circuit 3. Transformer Loss of an Element without a Fault	Yes Yes Yes Yes	No [b] No [b] No [b] No [b]	No No No No
	Single Pole Block, Normal Clearing [e]: 4. Single Pole (dc) Line	Yes	No [b]	No

Table 2-4 System Performance Conditions - Category B [9]

The second and third requirements are explained in the section 2.1.5 as they are common among the standards TPL-001-0.1, TPL-002-0b, TPL-003-0b, and TPL-004-0a.

2.1.3 Standard TPL-003-0b: *System Performance Following Loss of Two or More Bulk Electric System Elements (Category C)*

This standard has three requirements. The first requirement states that both the Planning Authority and Transmission Planner are responsible for demonstrating through valid assessments in the portion of the interconnected transmission system that belongs to them that the transmission system can be operated in such a way that it will be able to supply the projected customer demands and projected Firm Transmission Services at all demand levels over a range of forecast system demands, under the contingency conditions defined in Category C of the Table 2-5 shown below [10].

Category	Contingencies	System Limits or Impacts		
	Initiating Event(s) and Contingency Element(s)	System Stable and both Thermal and Voltage Limits within Applicable Rating [a]	Loss of Demand or Curtailed Firm Transfers	Cascading Outages
C Event(s) resulting in the loss of two or more (multiple) elements.	SLG Fault, with Normal Clearing[e]: 1. Bus Section	Yes	Planned/ Controlled[c]	No
	2. Breaker (failure or internal Fault)	Yes	Planned/ Controlled[c]	No
	SLG or 3Ø Fault, with Normal Clearing[e], Manual System Adjustments, followed by another SLG or 3Ø Fault, with Normal Clearing[e]: 3. Category B (B1, B2, B3, or B4) contingency, manual system adjustments, followed by another Category B (B1, B2, B3, or B4) contingency	Yes	Planned/ Controlled[c]	No
	Bipolar Block, with Normal Clearing[e]: 4. Bipolar (dc) Line Fault (non 3Ø), with Normal Clearing[e]:	Yes	Planned/ Controlled[c]	No
	5. Any two circuits of a multiple circuit towerline[f]	Yes	Planned/ Controlled[c]	No
	SLG Fault, with Delayed Clearing[e] (stuck breaker or protection system failure): 6. Generator	Yes	Planned/ Controlled[c]	No
	7. Transformer	Yes	Planned/ Controlled[c]	No
	8. Transmission Circuit	Yes	Planned/ Controlled[c]	No
	9. Bus Section	Yes	Planned/ Controlled[c]	No

Table 2-5 System Performance Conditions - Category C [10]

The second and third requirements are explained in the section 2.1.5 as they are common among the standards TPL-001-0.1, TPL-002-0b, TPL-003-0b, and TPL-004-0a.

2.1.4 Standard TPL-004-0a: *System Performance Following Extreme Events Resulting in the Loss of Two or More Bulk Electric System Elements (Category D)*

This standard has two requirements. The first requirement states that both the Planning Authority and Transmission Planner are responsible for demonstrating through valid assessments

in the portion of the interconnected transmission system that belongs to them that the transmission system has been evaluated for the risks and consequences of a number of each of the extreme contingencies listed in the Category D of the Table 2-6 shown below [11].

D [d] Extreme event resulting in two or more (multiple) elements removed or Cascading out of service.	3Ø Fault, with Delayed Clearing [e] (stuck breaker or protection system failure): 1. Generator 2. Transmission Circuit 3. Transformer 4. Bus Section 3Ø Fault, with Normal Clearing [e]: 5. Breaker (failure or internal Fault) 6. Loss of towerline with three or more circuits 7. All transmission lines on a common right-of way 8. Loss of a substation (one voltage level plus transformers) 9. Loss of a switching station (one voltage level plus transformers) 10. Loss of all generating units at a station 11. Loss of a large Load or major Load center 12. Failure of a fully redundant Special Protection System (or remedial action scheme) to operate when required 13. Operation, partial operation, or misoperation of a fully redundant Special Protection System (or Remedial Action Scheme) in response to an event or abnormal system condition for which it was not intended to operate 14. Impact of severe power swings or oscillations from Disturbances in another Regional Reliability Organization.	Evaluate for risks and consequences. ▪ May involve substantial loss of customer Demand and generation in a widespread area or areas. ▪ Portions or all of the interconnected systems may or may not achieve a new, stable operating point. ▪ Evaluation of these events may require joint studies with neighboring systems.

Table 2-6 System Performance Conditions - Category D [11]

The valid assessment for severe contingencies should be conducted for near-term only; that is, for years one through five in the future starting from the assessment date.

The second requirement is explained in the section 2.1.5 as it is common among the standards TPL-001-0.1, TPL-002-0b, TPL-003-0b, and TPL-004-0a.

2.1.5 Common Requirements among Standards TPL-001-0.1, TPL-002-0b, TPL-003-0b, and TPL-004-0a

The requirement 1 in all these standards indicates that the Planning Authority's and Transmission Planner's assessment shall:

- Be made annually, unless there are no significant changes to the system conditions.
- Cover critical system conditions and study years.
- Have all projected firm transfers modeled.
- Include existing and planned facilities.
- Include Reactive Power resources needed to meet system performance.
- Include the effects of existing and planned protection systems, including backup or redundant systems.
- Include the effects of existing and planned control devices.
- Include planned outages at those demand levels for which planned outages are performed.

The requirement 1 in the standards TPL-001-0.1, TPL-002-0b, and TPL-003-0b indicates that the valid assessments shall:

- Be conducted for both near-term (years one through five) and long-term (years six through ten) planning horizons.
- Be performed and evaluated for selected demand levels over the range of forecast system demands.

The requirement 2 in the standards TPL-001-0.1, TPL-002-0b, and TPL-003-0b indicates that when the Planning Authority and Transmission Planner find out that they are unable to comply

with the first requirement in its respective standards, they should each provide a detailed summary of their plans, called corrective plans, in order to meet the required system performance per its respective first requirement. They should also review in subsequent annual assessments the continuing need for the corrective plans.

The requirement 3 in the standards TPL-001-0.1, TPL-002-0b, and TPL-003-0b and the requirement 2 in the standard TPL-004-0a mandate the Planning Authority and Transmission Planner to document the results of their Reliability Assessments and corrective plans, as well as to report them annually to their respective Regional Reliability Organization.

2.1.6 Standard TPL-001-4: *Transmission System Planning Performance Requirements*

This standard has eight requirements, R1 thru R8 [12]. Requirements R1 and R7, as well as all the definitions included in the standard were enforced on January 1st, 2015, whereas the remaining requirements will be enforced until January 1st, 2016. By this date, TPL-001-4 will have completely superseded the standards TPL-001-0.1, TPL-002-0b, TPL-003-0b, and TPL-004-0a. Next, a brief description for all the requirements in this standard is presented.

R1. Each Transmission Planner (TP) and Planning Coordinator (PC) shall maintain System models within its respective area for performing studies needed to complete its Planning Assessment. These System models shall reflect as much as possible the projected System conditions in the transmission system by including all the existing and planned Bulk Electric

System (BES) facilities, real and reactive load forecasts, and Firm Transmission Services and Interchanges.

R2. Each TP and PC shall prepare an annual Planning Assessment of its portion of the BES. This requirement can be further subdivided in subareas depending on the analysis that is performed: steady state, near and long term; short circuit; and Stability, near and long term.

Steady State Analysis – Near Term:

The steady-state component in the Planning Assessment for the Near-Term Transmission Planning Horizon shall be supported by a current annual study. A past study may be used if it meets the requirement R2, part 2.6 per TPL-001-4.

Valid studies shall include the following:

- System peak Load for either year one or year two, and for year five.
- System off-peak Load for one of the five years.
- P1 events in Table 2-7 with known outages under those System peak or Off-Peak conditions when known outages are scheduled.
- Sensitivity cases shall be utilized for each of the System peak or Off-Peak conditions in order to demonstrate the impact of changes to the basic assumptions used in the model. The sensitivity analysis must vary conditions by a sufficient amount to stress the system within a range of credible conditions.

- The impact of possible spare equipment unavailability on System performance shall be studied when such unavailability corresponds to a major Transmission equipment that has a lead time of one year or more.

Steady State Analysis – Long Term:

The steady-state component in the Planning Assessment for the Long-Term Transmission Planning Horizon shall be supported by a current study that evaluates expected System peak Load conditions for one of the years in the Long-Term Transmission Planning Horizon. A past study may be used if they meet the requirement R2, part 2.6 per TPL-001-4.

Short Circuit Analysis:

This analysis shall be conducted for the Near-Term Transmission Planning Horizon in the Planning Assessment to determine the occurrence of overstressed circuit breakers; that is, circuit breakers that no longer have an interrupting capability for Faults that they will be expected to interrupt using the System short circuit model with any planned generation and Transmission Facilities in service.

Stability Analysis – Near Term:

The Stability analysis component in the Planning Assessment for the Near-Term Transmission Planning Horizon shall be supported by a current annual study. A past study may be used if it meets the requirement R2, part 2.6 per TPL-001-4.

Valid studies shall include the following:

- System peak Load for one of the five years, with Load levels that include a Load model which represents the expected dynamic behavior of Loads that could impact the study area, considering the behavior of induction motor Loads. An aggregate System Load model which represents the overall dynamic behavior of the Load is acceptable.
- System off-peak Load for one of the five years.
- Sensitivity cases shall be utilized for each of the System peak or Off-Peak conditions in order to demonstrate the impact of changes to the basic assumptions used in the model. The sensitivity analysis must vary conditions by a sufficient amount to stress the system within a range of credible conditions.

Stability Analysis – Long Term:

The Stability analysis component in the Planning Assessment for the Long-Term Transmission Planning Horizon shall be assessed to address the impact of proposed new generation or changes in that timeframe. A past study may be used if they meet the requirement R2, part 2.6 per TPL-001-4.

Corrective Actions Plans:

For planning events shown in Table 2-7, when the analysis indicates an inability of the System to meet the performance requirements in Table 2-7, the Planning Assessment shall include Corrective Action Plan(s) addressing how the performance requirements will be met.

The Corrective Action Plan(s) shall list System deficiencies and the associated actions needed to achieve required System performance, include actions to resolve performance deficiencies identified in multiple sensitivity studies, and be reviewed in subsequent annual Planning Assessments for continued validity.

If the PC or TP is unable to implement a Corrective Action Plan in the required timeframe due to circumstances beyond their control, then the PC or TP is allowed to use Non-Consequential Load Loss and curtailment of Firm Transmission Service to correct the situation that would normally not be permitted in Table 2-7.

If the short circuit current interrupting duty on circuit breakers exceeds their Equipment Rating, the Planning Assessment shall include Corrective Action Plan to address the Equipment Rating violations.

R3. For the steady state component of the Planning Assessment, each PC and TP shall perform studies based on computer simulation models using data provided in requirement R1 for the Near-Term and Long-Term Transmission Planning Horizons.

Studies shall be performed for planning events to determine whether the BES meets the performance requirements in Table 2-7 based on a Contingency list that includes all the events expected to produce more severe System impacts on its portion of the BES. The PC and TP shall coordinate with adjacent PCs and TPs to ensure that Contingencies on adjacent Systems which may impact their Systems are included in the Contingency list.

Studies shall be performed to assess the impact of the extreme events which are identified in a list created with those events in Table 2-7 expected to produce more severe System impacts on its portion of the BES. If the analysis concludes that there is Cascading caused by the occurrence of extreme events, an evaluation of possible actions designed to reduce the likelihood or mitigate the consequences and adverse impacts of the event(s) shall be conducted.

Every Contingency analysis performed shall simulate the removal of all elements that the Protection System and other automatic controls are expected to disconnect for each Contingency without operator intervention. Likewise, the Contingency analysis shall simulate the expected automatic operation of existing and planned devices designed to provide steady state control of electrical system quantities when such devices impact the study area.

R4. For the Stability component of the Planning Assessment, each PC and TP shall perform the Contingency analyses listed in Table 2-7 based on computer simulation models using data provided in requirement R1 for the Near-Term and Long-Term Transmission Planning Horizons.

Studies shall be performed for planning events to determine whether the BES meets the performance requirements in Table 2-7 based on a Contingency list that includes all the events expected to produce more severe System impacts on its portion of the BES. The PC and TP shall coordinate with adjacent PCs and TPs to ensure that Contingencies on adjacent Systems which may impact their Systems are included in the Contingency list.

For planning event P1, no generating unit shall pull out of synchronism. For planning events P2 through P7, when a generator pulls out of synchronism in the simulations, the resulting apparent impedance swings shall not result in the tripping of any Transmission system elements other than the generating unit and its directly connected Facilities. For planning events P1 through P7, power oscillations shall exhibit acceptable damping as established by the PC and TP.

Studies shall be performed to assess the impact of the extreme events which are identified in a list created with those events in Table 2-7 expected to produce more severe System impacts on its portion of the BES. If the analysis concludes that there is Cascading caused by the occurrence of extreme events, an evaluation of possible actions designed to reduce the likelihood or mitigate the consequences and adverse impacts of the event(s) shall be conducted.

Every Contingency analysis performed shall simulate the removal of all elements that the Protection System and other automatic controls are expected to disconnect for each Contingency without operator intervention. Likewise, the Contingency analysis shall simulate the expected automatic operation of existing and planned devices designed to provide steady state control of electrical system quantities when such devices impact the study area.

R5. Each PC and TP shall have criteria for acceptable System steady state voltage limits, post-Contingency voltage deviations, and the transient voltage response for its System. A low-voltage level and a maximum length of time that transient voltages may remain below that level shall be specified for transient voltage response.

R6. Each PC and TP shall define and document, within their Planning Assessment, the criteria or methodology used in the analysis to identify System instability for conditions such as Cascading, voltage instability, or uncontrolled islanding.

R7. Each PC, in conjunction with each of its TPs, shall determine and identify each entity's individual and joint responsibilities for performing the required studies for the Planning Assessment.

R8. Each PC and TP shall distribute its Planning Assessment results to adjacent PCs and TPs within 90 calendar days of completing its Planning Assessment. Documented comments that might be received from any such recipients shall be addressed by the PC or TP with a documented response within 90 calendar days of receipt of those comments.

Steady State & Stability:

a. The System shall remain stable. Cascading and uncontrolled islanding shall not occur.
b. Consequential Load Loss as well as generation loss is acceptable as a consequence of any event excluding P0.
c. Simulate the removal of all elements that Protection Systems and other controls are expected to automatically disconnect for each event.
d. Simulate Normal Clearing unless otherwise specified.
e. Planned System adjustments such as Transmission configuration changes and re-dispatch of generation are allowed if such adjustments are executable within the time duration applicable to the Facility Ratings.

Steady State Only:

f. Applicable Facility Ratings shall not be exceeded.
g. System steady state voltages and post-Contingency voltage deviations shall be within acceptable limits as established by the Planning Coordinator and the Transmission Planner.
h. Planning event P0 is applicable to steady state only.
i. The response of voltage sensitive Load that is disconnected from the System by end-user equipment associated with an event shall not be used to meet steady state performance requirements.

Stability Only:

j. Transient voltage response shall be within acceptable limits established by the Planning Coordinator and the Transmission Planner.

Category	Initial Condition	Event [1]	Fault Type [2]	BES Level [3]	Interruption of Firm Transmission Service Allowed [4]	Non-Consequential Load Loss Allowed
P0 No Contingency	Normal System	None	N/A	EHV, HV	No	No
P1 Single Contingency	Normal System	Loss of one of the following: 1. Generator 2. Transmission Circuit 3. Transformer [5] 4. Shunt Device [6]	3Ø	EHV, HV	No[9]	No[12]
		5. Single Pole of a DC line	SLG			
P2 Single Contingency	Normal System	1. Opening of a line section w/o a fault [7]	N/A	EHV, HV	No[9]	No[12]
		2. Bus Section Fault	SLG	EHV	No[9]	No
				HV	Yes	Yes
		3. Internal Breaker Fault [8] (non-Bus-tie Breaker)	SLG	EHV	No[9]	No
				HV	Yes	Yes
		4. Internal Breaker Fault (Bus-tie Breaker) [8]	SLG	EHV, HV	Yes	Yes
P3 Multiple Contingency	Loss of generator unit followed by System adjustments[9]	Loss of one of the following: 1. Generator 2. Transmission Circuit 3. Transformer [5] 4. Shunt Device [6]	3Ø	EHV, HV	No[9]	No[12]
		5. Single pole of a DC line	SLG			
P4 Multiple Contingency *(Fault plus stuck breaker[10])*	Normal System	Loss of multiple elements caused by a stuck breaker [10](non-Bus-tie Breaker) attempting to clear a Fault on one of the following: 1. Generator 2. Transmission Circuit 3. Transformer [5] 4. Shunt Device [6] 5. Bus Section	SLG	EHV	No[9]	No
				HV	Yes	Yes
		6. Loss of multiple elements caused by a stuck breaker[10] (Bus-tie Breaker) attempting to clear a Fault on the associated bus	SLG	EHV, HV	Yes	Yes
P5 Multiple Contingency *(Fault plus relay failure to operate)*	Normal System	Delayed Fault Clearing due to the failure of a non-redundant relay[13] protecting the Faulted element to operate as designed, for one of the following: 1. Generator 2. Transmission Circuit 3. Transformer [5] 4. Shunt Device [6] 5. Bus Section	SLG	EHV	No[9]	No
				HV	Yes	Yes
P6 Multiple Contingency *(Two overlapping singles)*	Loss of one of the following followed by System adjustments.[9] 1. Transmission Circuit 2. Transformer [5] 3. Shunt Device[6] 4. Single pole of a DC line	Loss of one of the following: 1. Transmission Circuit 2. Transformer [5] 3. Shunt Device [6]	3Ø	EHV, HV	Yes	Yes
		4. Single pole of a DC line	SLG	EHV, HV	Yes	Yes
P7 Multiple Contingency *(Common Structure)*	Normal System	The loss of: 1. Any two adjacent (vertically or horizontally) circuits on common structure [11] 2. Loss of a bipolar DC line	SLG	EHV, HV	Yes	Yes

Table 2-7 Steady State & Stability Performance Planning Events [12]

3. Automation in Transmission Systems Planning Simulations

As it is shown in Table 2-7, the standard TPL-001-4 requires to perform several contingency studies for the different categories in the steady state and stability analysis. Furthermore, TPL-001-4 requires to consider different cases with sensitivities for Near and Long-Term Transmission Planning Horizons. In large power systems, conducting these studies can take a lot of time and human resources. Hence, there is a need to develop general tools that could help simplify and streamline the procedures to conduct these analysis for either small or large power systems.

This work uses the capabilities of the software Power System Simulator for Engineering (PSS/E) to perform all the power flow calculations in the steady state analysis for the contingencies per TPL-001-4, and the programming language Python to automate most of the process.

3.1 Introduction to Python

This section presents only a brief introduction to the Python programming language. For tutorials and further details, refer to [13-17].

Python is a high-level programming language that is easy to learn and widely used in science in the fields of numeric programming, artificial intelligence, image processing, biology and others. It is known to be easy and can express with clarity complex ideas. Such complex ideas can easily lead to errors in other languages for solutions to the same problems.

High-level programming languages have the advantage of being much easier to program than low-level languages. Programs developed in a high-level language take less time to write, are shorter and easier to read, and are more likely to be correct due to their simplicity. Moreover, high-level programming languages are portable, thereby enabling them to run on different kinds of computers with few or no modifications. As a result of these advantages, almost all programs are written in high-level languages.

Python is also considered an interpreted language because its programs are executed by an interpreter. An interpreter reads a high-level program and executes it by processing the program a little at a time, alternately reading lines and performing computations.

Figure 3-1 High-Level Language Processed through an Interpreter [15]

There are two modes to use the interpreter in Python: the command-line mode and the script mode. In the command line mode, Python programs are typed directly in the command prompt, and the interpreter immediately prints the result. In the script mode, a program is written in a file, called a script, and then the interpreter is used to execute the contents of the file. To execute the program in this case, the interpreter should know the name of the script. By convention, these scripts have file names that end with the extension *.py.

3.1.1 Python Standard Library

Python comes with a library that includes different kinds of components. It contains data types that would normally be considered part of the “core” of a language, such as numbers, strings and lists. For these types, Python defines the form of literals and places some constraints on their semantics. The library also contains built-in functions that can be used by all Python code without the need of an import statement. The bulk of the library consists of a collection of modules.

3.1.2 Built-in-Functions

The Python interpreter has a number of functions and types built into it that are always available. They are listed in alphabetical order in Table 3-1.

abs	dict	help	min	setattr
all	dir	hex	next	slice
any	divmod	id	object	sorted
ascii	enumerate	input	oct	staticmethod
bin	eval	int	open	str
bool	exec	isinstance	ord	sum
bytearray	filter	issubclass	pow	super
bytes	float	iter	print	tuple
callable	format	len	property	type
chr	frozenset	list	range	vars
classmethod	getattr	locals	repr	zip
compile	globals	map	reversed	__import__
complex	hasattr	max	round	
delattr	hash	memoryview	set	

Table 3-1 Built-in-Functions in Python [17]

In addition to the standard library installed with Python, several other libraries can be imported into a program. Some of these libraries are:

- **csv:** allows to read and/or write files with comma separated values.
- **os:** a cross-platform interface to the functionality that most operating systems provide.
- **sys:** contains objects maintained by the interpreter that interact strongly with it, such as the module search path and the command-line arguments passed to the program.
- **psspy:** is one of the libraries or Python extension modules that comes with PSS/E. It allows to perform power systems simulations directly from Python and without having to open PSS/E [18]. This library is where the main interactions between Python and PSS/E occur.

3.1.3 Modules

Definitions made in the Python interpreter, such as functions and variables, are lost when the interpreter is closed. Thus, in order to write longer programs, it is better to use a text editor to prepare the input for the interpreter and run it with that file as input instead. This is known as creating a script. As programs get longer, they are usually split into several files for easier maintenance. A function can also be created when its code is used in several programs, thereby avoiding to copy its definition into each program.

To support this, Python has a way to put definitions in a file and use them in a script or in an interactive instance of the interpreter. Such a file is called a module. Definitions from a module can be imported into other modules or into the main module.

A module is a file containing Python definitions and statements. The file name is the module name with the suffix .py appended. Within a module, the module's name is available as a string as the value of the global variable __name__.

3.1.4 Important Tools of Python Language

A few of the most important tools used in Python programming are described in this section.

3.1.4.1 Functions

Besides offering simple syntax, Python also contributes to the construction of maintainable code by separating code into logical groups, such as modules and functions. A function is a named sequence of statements that performs a desired operation. A function can be thought of Python code grouped together into a single file, and it can be called as many times as programmers want. A function can also have functions embedded within it.

3.1.4.2 Lists

A list is an ordered set of values, where each value is identified by a number called index. The values that make up a list are called its elements. The simplest method to create a list is to enclose its elements in square brackets ([and]). Then elements in a list can be accessed starting from zero for the first element to m-1 for the last element, where m is the total number of elements in the list.

3.1.4.3 File Objects

While a program is running, its data is in memory. When the program ends, or the computer shuts down, data in memory disappears. To store data permanently, it must be put in a file.

In Python, opening a file creates what is called a file object. The file object can be assigned to any variable. Once a file is opened, methods can be invoked on the variable that has been assigned

to the file object. For instance, if a file has been opened for writing, then the *write* method can be invoked to put data into the file. If a file has been opened for reading, then the *read* method can be invoked to retrieve its content into a string.

3.2 Description of Developed Application

Nine separate Python programs have been developed to perform the contingencies indicated by the standard TPL-001-4 in steady state. The categories that are covered in this work are P0 thru P4 and P6 thru P7. Extreme events for steady-state are included as part of this work.

To facilitate running the simulations, a main program has also been created where the user inputs the name of the case to study, the name and number of the area where the contingencies will be simulated, and the specific category for steady-state planning events that the user would like to simulate. The complete codes for the programs are shown later in the Appendix A.

The algorithms used for the categories P1, P2, P4 and P7 are very similar, varying only in the number and type of events based on the TPL-001-4 standard and in the sections where their respective contingencies are generated. Likewise, the algorithms used for the categories P3 and P6 are very similar, varying only in the number and type of events based on the TPL-001-4 standard and in the sections where their respective initial conditions and contingencies are generated.

Taking advantage of the similarities between the algorithms, the next sections describe in detail each separate algorithm, using as examples the programs for categories P1 and P3 only. It is important to note that while some explanations are given in the next few sections about the code,

references [18, 19] for PSS/E and [13-17] for Python are very useful to gain a better understanding about all the functions and libraries used in the programs developed. In addition, the Python code of the programs has been well documented. Finally, the explanations try to focus on the simulation settings as they pertain to the requirements per TPL-001-4.

3.2.1 Algorithm for Categories P1, P2, P4 and P7

Figure 3-2 shows the algorithm used for the category P1 events. A similar algorithm has been used for the categories P2, P4 and P7. Explanations for each component and the main differences between the categories is given following the algorithm.

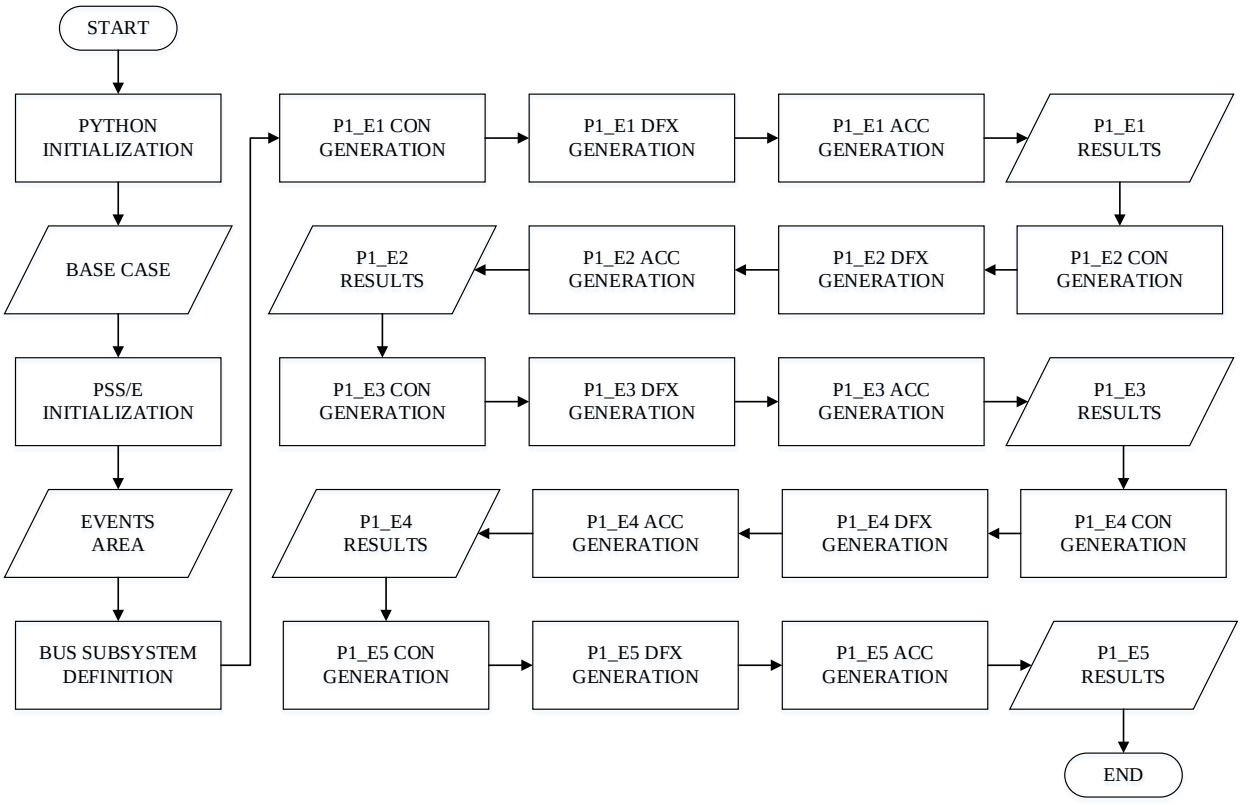

Figure 3-2 Algorithm for Category P1 Planning Events

As it can be inferred from Figure 3-2, the blocks where the files *.con, *.dfx, and *.acc and the results are generated do not vary much between the different events E1 through E5. In fact, the block with the largest differences corresponds to the one where the contingencies for the different events are created. Thus, in the next sub sections, most blocks in Figure 3-2 are explained, using only the string of blocks for P1_E1 events to describe in general what happens within the remaining E2 through E5 events.

3.2.1.1 Python Initialization

This block lets Python know where to look for the PSS/E tools and library files in the local disk drive where PSS/E has been installed. To accomplish this, the first parts of the program, beginning with *sys.path.append* and *os.environ['PATH']* are adding to the Python and system path variables. This effectively lets Python know where to look to find psspy, for example, which is one of several Python extension modules that come with PSS/E [20].

This section then proceeds to import the library psspy, and it also contains code that prints output information that PSS/E normally prints while running directly onto the command line of the interpreter.

3.2.1.2 Base Case

In this section, the user inputs to the program what base case file is going to be working with during the contingency studies for the events E1 thru E5. This base case should be in the format *.sav, and the program has been developed assuming it is located in the same folder where the program for each category is located.

3.2.1.3 PSS/E Initialization

This section initializes PSS/E by indicating the maximum number of buses that it might be working with. It also opens the base case.

3.2.1.4 Events Area

Here is where the user can specify the buses that PSS/E should consider while performing the AC Contingency studies. The user can identify the areas and zones of interest, as well as the voltage range.

It is important to note that the programs have been developed assuming that the names for the subsystem (*.sub), monitored (*.mon), and tripping (*.trp) files match that of the area name specified by the user in this block. In addition, the programs have been created assuming that the subsystem, monitored, and tripping files are located in the same folder where the program for each category is located.

3.2.1.5 Bus Subsystem Definition

Python takes the information specified by the user in the prior section, and it uses the Application Program Interface *bsys* to create a bus subsystem [18], thereby narrowing down the area where the contingencies occur in the base case.

3.2.1.6 P1_E1 con Generation

To lose one of the generators per contingency in the bus subsystem, the program first pairs each in-service generation unit with its respective bus. This pairing allows Python to later use a for loop to write in a contingency file with extension *.con the specific machine that is removed per each single contingency.

This block is where the main differences between the events E1 thru E5 can be found. For E2, the program pairs each in-service, non-transformer branch with the two buses to which it is connected. For E3, it pairs each in-service, transformer branch with the two buses where it is connected. For E4, it pairs each in-service fixed shunt device with the in-service bus to which it is connected. For E5, it pairs each in-service, two-terminal DC line with the buses where it is connected. Finally, after saving these pairings in lists, the program uses for loops to create the different contingency files accordingly.

Likewise, this block is where the main differences between the algorithms for the categories P1, P2, P4 and P7 can be found. It has to be unique for the different events because that is how the contingencies are defined based on the TPL-001-4 standard.

When the blocks generating the contingencies work with DC lines, it is assumed that their Line Names are such that a single pole in a bipolar DC line ends with 1_2, for instance, to indicate that this is the pole with negative voltage with respect to ground in the DC line number 1. Consequently, the respective pole with positive voltage would end with 1_1. Both monopolar and bipolar DC lines should be named ending at least with a number.

3.2.1.7 P1_E1 dfx Generation

A distribution factor data file is created in this block. This file includes information related to the network conditions contained in the working case. It also preserves line outage distribution factors [19] that are calculated based on the network. To create the file, it is needed to first have defined the subsystem, monitored, and contingency files.

Input for the process of creating the distribution factor file is contained in three data files [19]:

- Subsystem Description Data file (*.sub) indicates relevant subsystems of the working case.
- Monitored Element Data file (*.mon) specifies network elements to be monitored for problems such as overloaded lines or voltage violations.
- Contingency Description Data file (*.con) states the contingencies to be tested.

As stated earlier, the programs have been developed assuming that the names for the subsystem (*.sub), monitored (*.mon), and tripping (*.trp) files match that of the area name specified by the user in the section 3.2.1.4. Also, the programs have been created assuming that the subsystem, monitored, and tripping files are located in the same folder where the program for each category is located.

3.2.1.8 P1_E1 acc Generation

A multiple level AC contingency calculation is run in this block. It includes tripping events and corrective actions in order to meet the requirements included in the TPL-001-4 standard. For instance, a tripping file is used to simulate the removal that automatic controls are expected to

disconnect for each contingency without operator intervention. In the simulations performed in this work, this tripping file includes information for the Generator Step-Up transformers only.

Also, the expected automatic operation of existing devices to provide steady state control of electrical system quantities is simulated. This includes phase-shifting transformers, load tap changing transformers, capacitors and inductors. Moreover, area interchange controls are simulated at the tie lines to guarantee that expected power transfers will be met as much as possible. Finally, before the loss of a generation unit, a redispatch is conducted with participating machines connected to dispatch subsystem buses and positive active power generation. Each machine's participation factor is its maximum active power generation with positive values [19]. Figure 3-3 summarizes the procedure of evaluating a single contingency with multiple level contingency analysis in PSS/E.

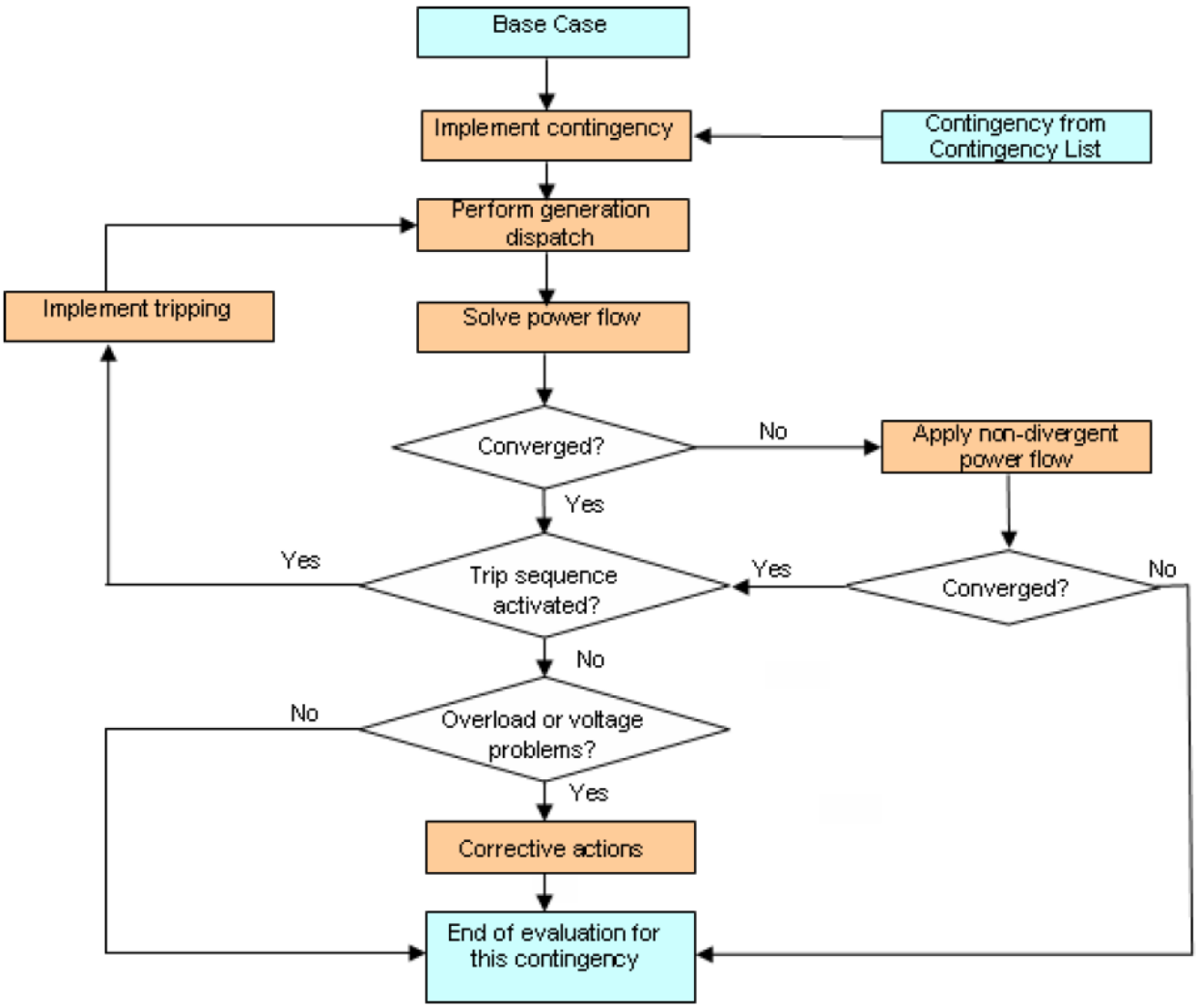

Figure 3-3 Outline of Evaluation Procedure Using AC Power Flows for a Single Contingency [19]

The multiple level AC contingency analysis is a very powerful tool for transmission planning engineers. It enables to model special relay actuation schemes. For example, tripping sequences can be used to model automatic actions such as remedial action schemes (RAS), special protection schemes (SPS), and other operating guides based on previous planning experience with a system. During contingency analysis, switching is performed and new load flows are calculated whenever relaying sequences are triggered. The tripping simulation can simulate cascading outages from

contingencies. The intent is to calculate post-contingency state variables that are as close as possible to real-time operation [19].

3.2.1.9 P1_E1 Results

Some of the results contained in the final *.acc file created are sent to two output files. The first one is an excel file *.xls where three spreadsheets are tabulated with information for all branch flows, transformer flows, bus voltages, and contingency events. The excel file is saved in the same location where all the other files are located.

The second output file is a text file *.txt that shows the results as a non-spreadsheet overload report. The settings used to create this report check for overload branches and transformers exceeding 90% of their Rate A capacity, whose loading has increased in at least 5% from the base case. In addition, the settings check for voltages that exceed specific maximum and minimum values, as well as voltage deviations. The criteria for these voltages is specified in the Monitored Element Data file for the respective areas. The text output file is also saved in the same location as the other files.

Alternatively, the extensive data stored in the final *.acc file can be accessed directly by using the application *ACCC Post Processor (AcccBrwsGrid)* that is installed with PSS/E. This allows for a more thorough analysis of the results.

3.2.2 Algorithm for Categories P3 and P6

Figure 3-4 shows the algorithm used for the category P3 events. A similar algorithm has been used for the category P6. Explanations for each component and the main differences between the categories is given following the algorithm. Most of the blocks in Figure 3-4 have already been explained in section 3.2.1. The main difference between this algorithm and the one explained earlier resides on the existence of initial conditions. These initial conditions lead to new base cases where a second contingency, either single or multiple, can occur. The following sub sections will focus on explaining the remaining new blocks.

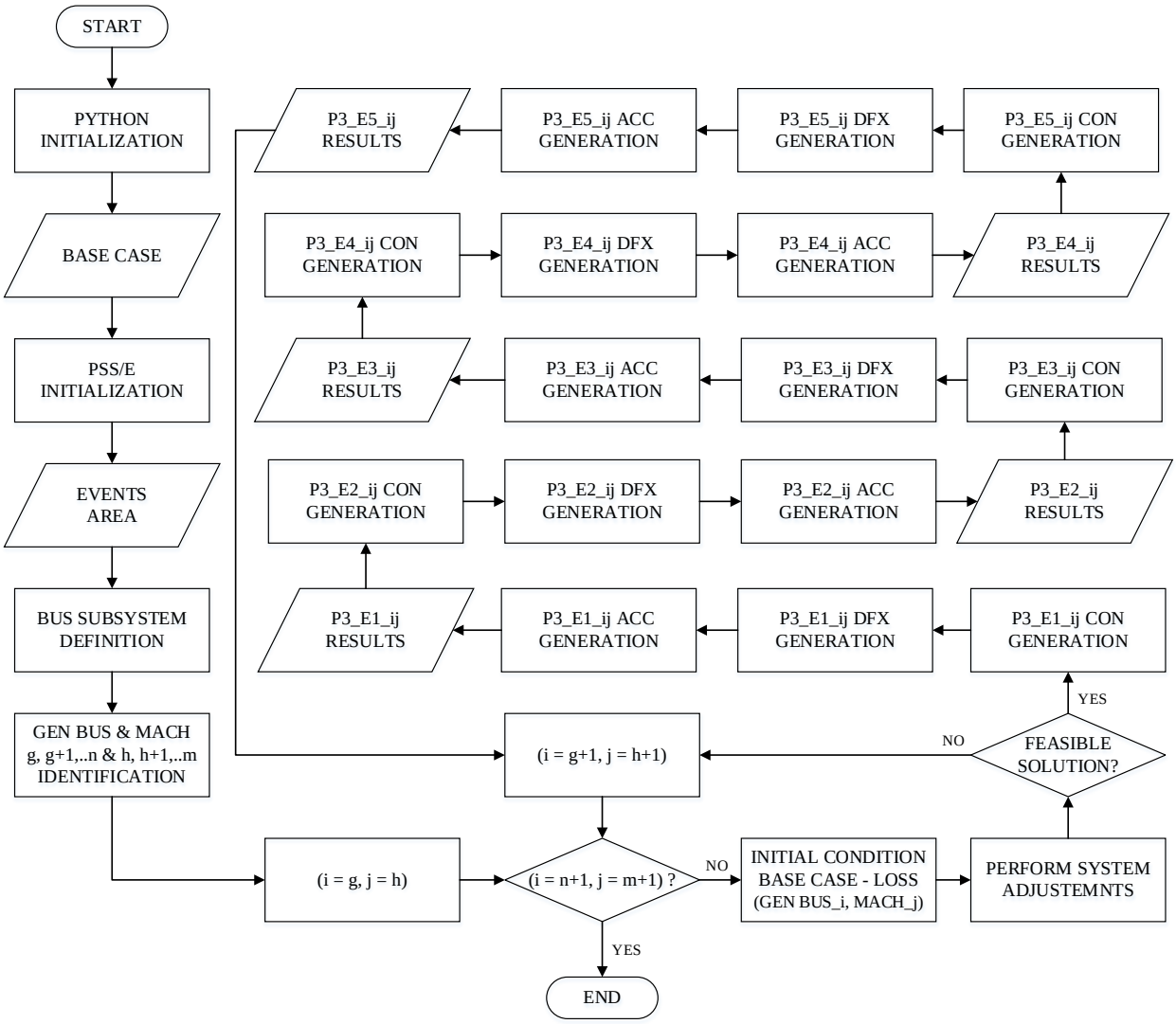

Figure 3-4 Algorithm for Category P3 Planning Events

3.2.2.1 Gen Bus & Mach Identification

The intent of this block is to pair each in-service generation unit with its respective bus. This pairing allows Python to later use a for loop to create new base cases, one for each generation unit removed from the system, that represent the initial conditions required by TPL-001-4. Then Python stays inside the same for loop to perform the contingencies for the events E1 thru E5 in a similar way as it was explained in section 3.2.1.

This block is where the main differences between the algorithms for the categories P3 and P6 can be found. The pairings that are performed between generators, transmission circuits, transformers, shunt devices, and single poles of DC lines with their respective buses represent the different initial conditions indicated in the TPL-001-4 standard.

3.2.2.2 Initial Condition Base Case - Loss

This block first creates a temporary contingency file that includes a single contingency for one of the machines connected to each bus. Then this temporary *.con file is used to generate a temporary distribution factor data file. Finally, the temporary *.dfx file is used to impose the single contingency in the original base case.

3.2.2.3 Perform System Adjustments

To reflect what occurs in real transmission system operation, TPL-001-4 specifies that system adjustments must follow the loss of each generation unit. When the contingency is imposed in the original base case, a redispatch is conducted with participating machines connected to dispatch subsystem buses and positive active power generation. Each machine's participation factor is its maximum active power generation with positive values [19]. Furthermore, new power flows are calculated in order to reach steady state variables with the new initial conditions. While calculating these power flows, the expected automatic operation of existing devices to provide steady state control of electrical system quantities is simulated. This includes phase-shifting transformers, load tap changing transformers, capacitors and inductors. Finally, area interchange controls are simulated at the tie lines to guarantee that expected transfers will be met as much as possible.

After calculating the new steady state variables, the new base case is saved. However, this new case is evaluated to determine whether or not a feasible solution has been reached. If so, then the procedure described in sections 3.2.1.6 to 3.2.1.9 is used to analyze the contingencies for the events E1 thru E5.

3.2.3 Events and Criteria for Future Research

After the 2003 blackout affected a large portion of the North American Eastern Interconnection, it became obvious the need to enforce the inclusion of protection scheme operations in the simulations or studies used to conduct power system planning. This inclusion was first mandated by the standards TPL-001-0.1, TPL-002-0b, TPL-003-0b, and TPL-004-0a, and now by the latest TPL-001-4 standard.

Even though this work has included most of the steady-state planning events indicated by TPL-001-4, there has been a limitation due to the lack of access to protection schemes information from real base cases and substations. As a result, not all the events have been included in this work. In addition, severe events have not been simulated. Including these events and the response of schemes such as remedial action schemes (RAS) and special protection schemes (SPS) should be considered material for future research.

Table 3-2 shows the specific steady-state performance planning events that have been simulated in this work based on the TPL-001-4 standard.

Steady State & Stability:

a. The System shall remain stable. Cascading and uncontrolled islanding shall not occur.
b. Consequential Load Loss as well as generation loss is acceptable as a consequence of any event excluding P0.
c. Simulate the removal of all elements that Protection Systems and other controls are expected to automatically disconnect for each event.
d. Simulate Normal Clearing unless otherwise specified.
e. Planned System adjustments such as Transmission configuration changes and re-dispatch of generation are allowed if such adjustments are executable within the time duration applicable to the Facility Ratings.

Steady State Only:

f. Applicable Facility Ratings shall not be exceeded.
g. System steady state voltages and post-Contingency voltage deviations shall be within acceptable limits as established by the Planning Coordinator and the Transmission Planner.
h. Planning event P0 is applicable to steady state only.
i. The response of voltage sensitive Load that is disconnected from the System by end-user equipment associated with an event shall not be used to meet steady state performance requirements.

Stability Only:

j. Transient voltage response shall be within acceptable limits established by the Planning Coordinator and the Transmission Planner.

Category	Initial Condition	Event [1]	Fault Type [2]	BES Level [3]	Interruption of Firm Transmission Service Allowed [4]	Non-Consequential Load Loss Allowed
P0 No Contingency	Normal System	None	N/A	EHV, HV	No	No
P1 Single Contingency	Normal System	Loss of one of the following: 1. Generator 2. Transmission Circuit 3. Transformer [5] 4. Shunt Device [6]	3Ø	EHV, HV	No[9]	No[12]
		5. Single Pole of a DC line	SLG			
P2 Single Contingency	Normal System	1. Opening of a line section w/o a fault [7]	N/A	EHV, HV	No[9]	No[12]
		2. Bus Section Fault	SLG	EHV	No[9]	No
				HV	Yes	Yes
P3 Multiple Contingency	Loss of generator unit followed by System adjustments[9]	Loss of one of the following: 1. Generator 2. Transmission Circuit 3. Transformer [5] 4. Shunt Device [6]	3Ø	EHV, HV	No[9]	No[12]
		5. Single pole of a DC line	SLG			
P4 Multiple Contingency *(Fault plus stuck breaker[10])*	Normal System	Loss of multiple elements caused by a stuck breaker [10](non-Bus-tie Breaker) attempting to clear a Fault on one of the following: 1. Generator 2. Transmission Circuit 3. Transformer [5] 4. Shunt Device [6] 5. Bus Section	SLG	EHV	No[9]	No
				HV	Yes	Yes
		6. Loss of multiple elements caused by a stuck breaker[10] (Bus-tie Breaker) attempting to clear a Fault on the associated bus	SLG	EHV, HV	Yes	Yes
P6 Multiple Contingency *(Two overlapping singles)*	Loss of one of the following followed by System adjustments.[9] 1. Transmission Circuit 2. Transformer [5] 3. Shunt Device[6] 4. Single pole of a DC line	Loss of one of the following: 1. Transmission Circuit 2. Transformer [5] 3. Shunt Device [6]	3Ø	EHV, HV	Yes	Yes
		4. Single pole of a DC line	SLG	EHV, HV	Yes	Yes
P7 Multiple Contingency *(Common Structure)*	Normal System	The loss of: 1. Any two adjacent (vertically or horizontally) circuits on common structure [11] 2. Loss of a bipolar DC line	SLG	EHV, HV	Yes	Yes

Table 3-2 Summary of Contingency Events Included in this Work per TPL-001-4 [12]

4. Analysis of Results

This section shows two examples applying the programs developed, describing for each one how they pertain to the compliance of the TPL-001-4 standard. The two examples use slightly modified versions of two general base cases *savnw.sav*, included in PSS/E, and the standard IEEE 300 bus system.

4.1 Power System Element Contingency, System Adjustments, and Overload and Voltage Reports

For the first example, a modified version of the *savnw.sav* is used. One of the reasons to modify the original case is so that all the power system elements specified in the TPL-001-4 standard for planning events are included. Figure 4-1 summarizes the changes made to the original file.

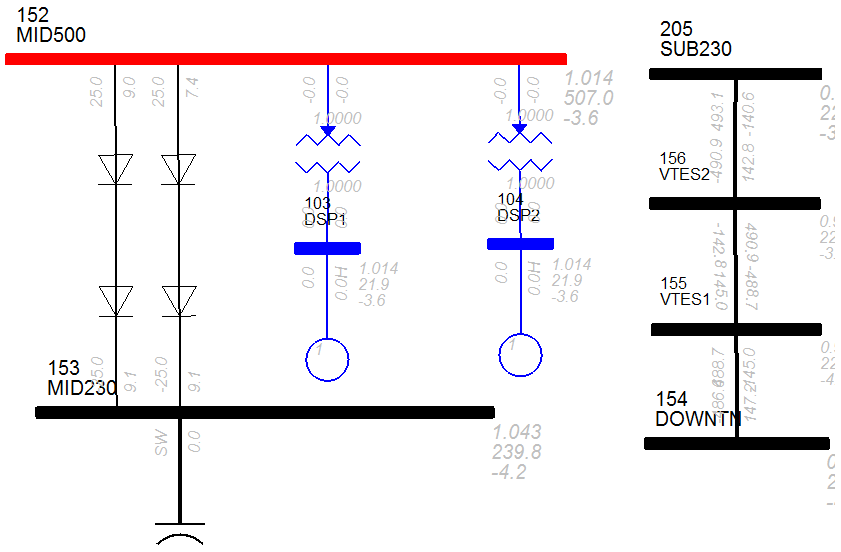

Figure 4-1 Modifications Made to Original savnw.sav Case

The following elements have been added:

- A bipolar DC line between buses 152 and 153 modeled as two single poles.
- A switched bank of capacitors at bus 153
- New generation units at new buses 103 and 104 which will be dispatched to make up for lost generation in the area of interest when any of the remaining two units in the base case are tripped. In this work, this redispatch is conducted with participating machines connected to dispatch subsystem buses and positive active power generation. The method to calculate this redispatch might vary among Planning Coordinators and Transmission Planners.
- Three short transmission lines between buses 154 and 205, with their respective new buses in between.

The standard establishes in its requirement R1 that for the category P0 normal system conditions should exist. This implies not only that all the in-service equipment must be modeled correctly, but also that no overloads or voltage violations are present. In order to meet this requirement in the simulations performed in this work, the short transmission lines between buses 154 and 205 were added to alleviate a 157% overload in the tie-line between buses 154 and 205 in the original base case.

In trying to show how the program works with simple examples, the next few sections describe one specific contingency event for each category included in this work based on the standard TPL-001-4. Unless otherwise noted, most contingencies take place in the area 1, FLAPCO. Table 4-1 shows the voltage criteria used in all the simulations in this work.

	Minimum Voltage (p.u.)	Maximum Voltage (p.u.)	Maximum Voltage Drop (p.u.)	Maximum Voltage Rise (p.u.)
HV (100kV to 300kV)	0.95	1.06	0.1	0.1
EHV (300kV to 765kV)	0.98	1.08		
Percentage Flow Threshold for Transmission Lines and Transformers:				90

Table 4-1 Steady State Voltages and Post-Contingency Voltage Deviations Limits

4.1.1 Performance during P1 events

The output *.acc, *.txt, and *.xls files contain specific information that can be used to assess the performance of the simulations. In addition, during the calculations messages are printed directly in the command line so that the user can see what is occurring in the simulations.

It is important to note that in the new base case that would correspond to category P0, unit 1 at bus 101 is generating 255 MW, while unit 1 at bus 102 is generating 200 MW. Therefore, upon losing either unit, a redispatch needs to be performed according to the TPL-001-4 standard.

When the unit at bus 101 is tripped, the following message is shown in the command line:

```
PROCESSING  CONTINGENCY 'M1B101':
REMOVE MACHINE 1 FROM BUS 101 [NUC-A       21.600]
GENERATION DISPATCH MW:    255.
LOAD SHEDDING MW:      0.
```

It effectively lets the user know that 255 MW of generation have been redispatched to make up for the lost unit. To verify that the new generation is coming from the units at buses 103 and 104, one can open the *.acc output file with the program *ACCC Post Processor (AcccBrwsGrid*) that comes with PSS/E. Figure 4-2 shows the flow report in the generating step-up transformers under this contingency. It clearly indicates that the units at buses 103 and 104 are injecting 255.16

MW into the system to make up for the lost generation, whereas the tripped unit at bus 101 is no longer producing power.

Monitored Element				%	Flow	Flow Change
101 NUC-A	21.600	151 NUCPANT	500.0	0.00	0.00	256.59
102 NUC-B	21.600	151 NUCPANT	500.0	16.14	201.79	0.06
103 DSP1	21.600	152 MID500	500.0	20.41	127.58	127.58
104 DSP2	21.600	152 MID500	500.0	20.41	127.58	127.58

Figure 4-2 Monitored Generation Step-Up Transformers with Unit at Bus 101 Tripped

It is relevant to note that in this base case bus 3011, located in area 5, WORLD, is the swing bus in the interconnected system. When generation is lost in FLAPCO, the lack of generation is not injected by the swing bus in the interconnected system. Instead, it comes from the redispatched units that were added to the original base case. This also allows the multiple AC contingency analysis to converge, reaching new steady state variables.

As expected, similar results are obtained when the tripped unit is the one at bus 102. This is shown in Figure 4-3.

Monitored Element				%	Flow	Flow Change
101 NUC-A	21.600	151 NUCPANT	500.0	21.57	269.69	13.10
102 NUC-B	21.600	151 NUCPANT	500.0	0.00	0.00	201.73
103 DSP1	21.600	152 MID500	500.0	16.00	100.01	100.01
104 DSP2	21.600	152 MID500	500.0	16.00	100.01	100.01

Figure 4-3 Monitored Generation Step-Up Transformers with Unit at Bus 102 Tripped

To find out whether overloads or voltage violations have occurred during each contingency or not, one can look at the *.txt output file. Figure 4-4 shows the report for all the contingencies. The results indicate that no overloads or violations have taken place.

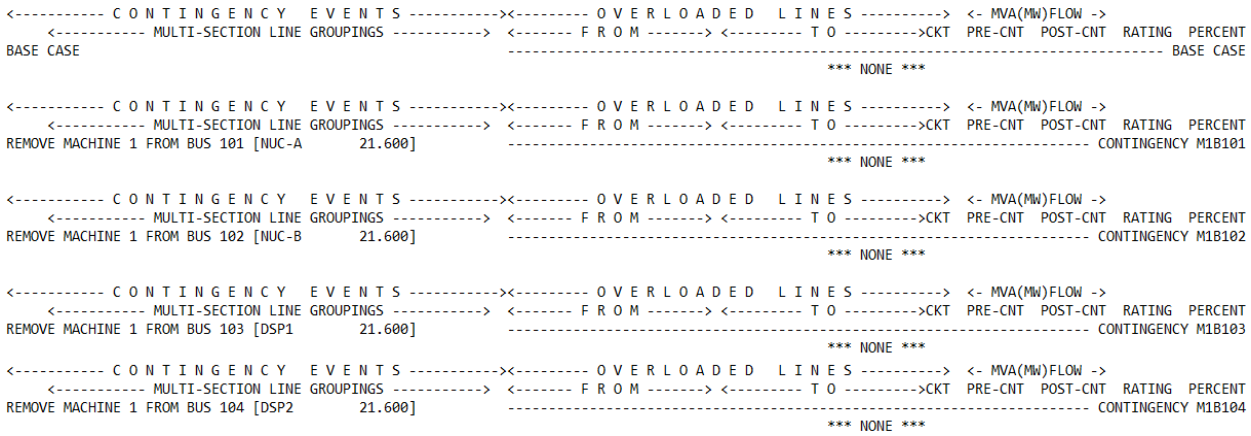

Figure 4-4 Overloads and Voltage Violations Report when Planning Events Trip each Generation Unit

4.1.2 Performance during P2 events

It happens that in FLAPCO, the area that was used in section 4.1.1, there are no multi-section lines. However, the remaining two areas in this base case include at least one. To show the event E1 in category P2, the simulations are run again but with the contingencies taking place in area 2, LIGHTCO. This change in areas can easily be achieved due to the way the programs have been developed. It is just a matter of making sure to have the *.mon, *.sub, and *.trp files for the desired area in the same folder as the base case *.sav and then running the desired programs.

For the multi-section line between buses 201, 204, and 205, Figure 4-5 shows the list of contingencies to be evaluated during the AC contingency analysis.

```
COM PSS(R)E 32
COM
COM OPENING OF A LINE SECTION W/O A FAULT 'LIGHTCO'

CONTINGENCY '201_204_1'
OPEN BRANCH FROM BUS 201 TO BUS 204 CIRCUIT 1
END

CONTINGENCY '204_205_1'
OPEN BRANCH FROM BUS 204 TO BUS 205 CIRCUIT 1
END

END
```

Figure 4-5 Contingencies for E1 Events in Category P2

Transmission lines with multi-sections are modeled in PSS/E with dummy buses in-between the starting and ending buses; therefore, there is no direct access to the dummy buses. This unique characteristic has been considered in the simulations so that appropriate software settings are automatically changed while generating the contingencies to be evaluated. By doing so, each separate section, rather than the entire multi-section line, is opened without the presence of a fault.

Label	Converged	Maximum Bus Mismatch	System Mismatch	Solution termination state
BASE CASE	☑	0.00	0.00	MET CONVERGENCE TOLERANCE
201_204_1	☐	574.39	2496.53	BLOWN UP
204_205_1	☐	837.32	3407.98	BLOWN UP

Figure 4-6 Results for Events 1 in Category P2

Figure 4-6 shows the mismatches obtained after the multiple level AC contingency calculations. Both contingencies made the case diverge due to the lack of robustness that the original *savnw.sav* case has. Observe that the standard TPL-001-4 specifically indicates that the simulation should be such that each line section is opened without any fault type. In fact, this type of events is the ones that Planning Coordinators and Transmission Planners use to simulate scheduled maintenance that is performed to the transmission lines in their respective portion of the interconnected transmission system.

4.1.3 Performance during P3 events

When two independent events trip the two generation units that are initially injecting all the power in FLAPCO in the base case, allowing for system adjustments between their trips, a similar situation as the one observed in section 4.1.1 should be observed. That is, system adjustments are such that the generation units connected at buses 103 and 104 pick up all the unbalance in generation. Figure 4-7 shows the results for that planning event.

Monitored Element				%	Flow	Flow Change
101 NUC-A	21.600	151 NUCPANT	500.0	0.00	0.00	0.00
102 NUC-B	21.600	151 NUCPANT	500.0	0.00	0.00	201.77
103 DSP1	21.600	152 MID500	500.0	36.42	227.60	100.02
104 DSP2	21.600	152 MID500	500.0	36.42	227.60	100.02

Figure 4-7 Generation Step-Up Transformers Flows with Units at Buses 101 and 102 Tripped

As expected, with an initial generation equal to 255 MW for unit 1 at bus 101 and 200 MW for unit 1 at 102, the total unbalance is indeed picked up by the unit 1 at bus 103 and unit 1 at bus 104. Note that, as it is indicated by the TPL-001-4 standard, these calculations are conducted in two stages to allow for planned system adjustments such as a re-dispatch of generation.

The first stage is basically what was explained in section 4.1.1. Then this new steady-state variables constitute the initial conditions for 5 new different planning events E1 thru E5.

So far, no flow or voltage violations have been reported in any simulations. However, if the results obtained for the events 2 in the category P3 are observed, the first violations in the monitored area, FLAPCO, start to occur. Figure 4-8 shows all the violations when the circuit 1 of

the line between buses 151 and 152 is tripped, after the generation unit at bus 101 had tripped. Note that the reports include transmission lines and buses that are specified in the *.mon file. In this instance, the monitored equipment is located in the area FLAPCO.

```
<----------- C O N T I N G E N C Y   E V E N T S ----------><--------- O V E R L O A D E D   L I N E S ---------->  <- MVA(MW)FLOW ->
     <----------- MULTI-SECTION LINE GROUPINGS ----------->  <------- F R O M -------> <--------- T O --------->CKT  PRE-CNT  POST-CNT  RATING  PERCENT
BASE CASE                                                   ------------------------------------------------------------------------------------ BASE CASE
                                                                                                                *** NONE ***

<----------- C O N T I N G E N C Y   E V E N T S ----------><--------- O V E R L O A D E D   L I N E S ---------->  <- MVA(MW)FLOW ->
     <----------- MULTI-SECTION LINE GROUPINGS ----------->  <------- F R O M -------> <--------- T O --------->CKT  PRE-CNT  POST-CNT  RATING  PERCENT
DISCONNECT BRANCH FROM BUS 151 [NUCPANT     500.00] TO BUS 152 [MID500      500.00] CIRCUIT 1 ---------------------------------- CONTINGENCY 151_152_1
                                                             154 DOWNTN      230.00    155*VTES1       230.00 1     489.8    505.7    600.0     91.7
                                                             154*DOWNTN      230.00    205 SUB230      230.00 1     480.7    495.9    600.0     90.2
                                                             155 VTES1       230.00    156*VTES2       230.00 1     491.2    507.3    600.0     91.7
                                                             156 VTES2       230.00    205*SUB230      230.00 1     492.5    509.0    600.0     91.7

                                                  X-------- B U S --------X  V-CONT  V-INIT  X-------- B U S --------X  V-CONT  V-INIT
    'FLAPCO_HV   ' BUSES WITH VOLTAGE LESS THAN 0.9500:    154 DOWNTN      230.00 0.91628 0.96613     155 VTES1       230.00 0.91917 0.96878
                                                           156 VTES2       230.00 0.92209 0.97145

    'FLAPCO_EHV  ' BUSES WITH VOLTAGE LESS THAN 0.9800:    152 MID500      500.00 0.97363 1.01866
```

Figure 4-8 Overloads and Voltage Violations Report with Unit at Bus 101 Tripped as Initial Condition and Transmission Line between Bus 151 and 152 Tripped

Although the standard TPL-001-4 states that the annual planning assessment shall be such that no equipment is allowed to operate above its rated capacity after the Corrective Action Plans have been implemented, the reports generated by the developed programs use a 90% threshold of their rated capacity to include the results in the reports. In doing so, transmission planners can flag contingencies that cause equipment to get very close to their rated capacity. These flagged pieces of equipment would require further analysis in order to prevent a new, unexpected contingency to worsen the power system operating conditions.

A further study as to why the lines and buses in Figure 4-8 have flow and voltage violations can determine that the net power imported from area 2, LIGHTCO, increases in order to supply the loads located at bus 154, DOWNTN, as well as other loads in the system. Further, these tie lines offer the best route for the power to travel from the generators that are left in the

interconnected system downstream to the loads. As a consequence, the flows in the tie-lines increase, while voltages along the same flow path drop.

4.1.4 Performance during P4 events

The simulations have assumed local backup-breaker failure when there is a loss of multiple elements caused by a stuck breaker attempting to clear a fault on a transmission circuit. Figure 4-9 illustrates a simple example to explain the logic behind it.

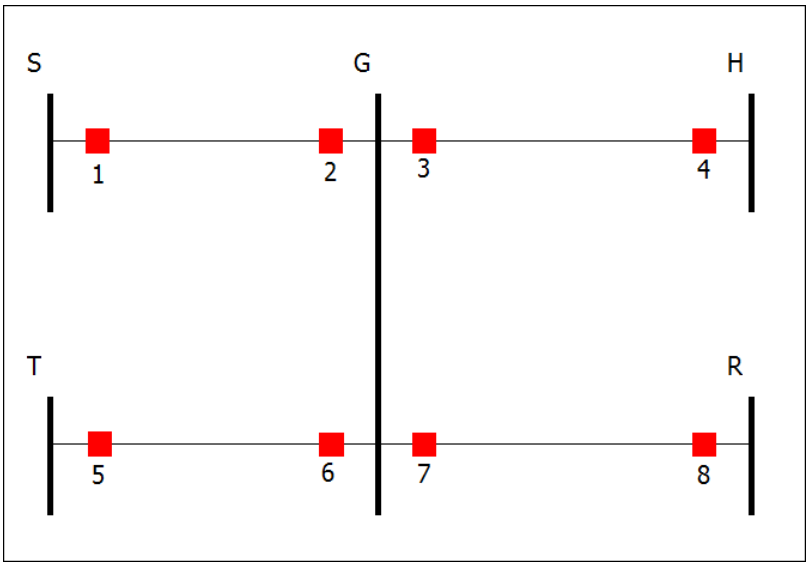

Figure 4-9 Local Backup-Breaker Failure Example

Before a fault in the line between buses G and H, both breakers 3 and 4 should trip. If only breaker 4 operates and breaker 3 remains stuck, then the backup protection assumed has been local. This means that breakers 2, 6, and 7 would operate in order to clear the fault in the transmission line between buses G and H. A similar logic has been used when there is a loss of multiple elements caused by a stuck breaker attempting to clear a fault on a transformer.

An example for the results can be seen by looking at the fault in the transmission line between buses 155 and 156, when the only breaker to operate is the one at bus 155. Figure 4-10 shows the output in the text file for that planning event.

```
<---------- C O N T I N G E N C Y   E V E N T S ----------><--------- O V E R L O A D E D   L I N E S ---------->  <- MVA(MW)FLOW ->
     <---------- MULTI-SECTION LINE GROUPINGS ----------->  <------- F R O M -------> <--------- T O --------->CKT  PRE-CNT  POST-CNT  RATING  PERCENT
OPEN LINE FROM BUS 156 [VTES2       230.00] TO BUS 205 [SUB230      230.00] CKT 1 ------------------------------------------------ CONTINGENCY 155_156_1
OPEN LINE FROM BUS 156 [VTES2       230.00] TO BUS 155 [VTES1       230.00] CKT 1
                                                                154*DOWNTN      230.00    205 SUB230      230.00 1     500.1    903.6    600.0    154.4

                                                          X-------- B U S --------X  V-CONT  V-INIT  X-------- B U S --------X  V-CONT  V-INIT
 'FLAPCO_HV    ' BUSES WITH VOLTAGE GREATER THAN 1.0600:    153 MID230      230.00 1.06076 1.04260
```

Figure 4-10 Overloads and Voltage Violations Report with Stuck Breaker at Bus 155 Trying to Clear a Fault in Transmission Line between Buses 155 and 156

4.1.5 Performance during P6 events

With the loss of a shunt device as an initial condition, it is expected that the reactive power support available in the respective portion of the interconnected system will be affected. However, TPL-001-4 states that simulations shall include expected automatic operation of existing and planned devices designed to provide steady state control of electrical system quantities such as switched capacitors and inductors [12]. This can be verified in the results of the simulation by opening the new base case that is created with the initial conditions.

In the modified *savnw.sav* case, a fixed inductor is connected to bus 151, and a switched shunt device is injecting a total of 0 MVAR at bus 153. When the initial condition trips the reactor at bus 151, the switched shunt device automatically changes its reactive power to absorb the maximum possible amount based on its settings. Figure 4-11 shows the results for this expected automatic

operation of existing switched shunt devices. XXX shows a few violations in the output *.txt file when transmission lines are lost after the initial condition.

Figure 4-11 Automatic Switched Shunt Device Operation with Fixed Shunt Reactor at Bus 151 Tripped as Initial Condition

```
<----------- C O N T I N G E N C Y   E V E N T S ----------><--------- O V E R L O A D E D   L I N E S ----------> <- MVA(MW)FLOW ->
     <----------- MULTI-SECTION LINE GROUPINGS -----------> <------- F R O M -------> <--------- T O --------->CKT  PRE-CNT  POST-CNT  RATING  PERCENT
DISCONNECT BRANCH FROM BUS 153 [MID230      230.00] TO BUS 154 [DOWNTN      230.00] CIRCUIT 1 ---------------------------------- CONTINGENCY 153_154_1
                                                       153 MID230      230.00    154*DOWNTN      230.00 2     200.6    275.2    300.0     94.2

                                                  X-------- B U S --------X  V-CONT  V-INIT  X-------- B U S --------X  V-CONT  V-INIT
 'FLAPCO_EHV  ' BUSES WITH VOLTAGE GREATER THAN 1.0800:    152 MID500      500.00 1.09354 1.06031

<----------- C O N T I N G E N C Y   E V E N T S ----------><--------- O V E R L O A D E D   L I N E S ----------> <- MVA(MW)FLOW ->
     <----------- MULTI-SECTION LINE GROUPINGS -----------> <------- F R O M -------> <--------- T O --------->CKT  PRE-CNT  POST-CNT  RATING  PERCENT
DISCONNECT BRANCH FROM BUS 153 [MID230      230.00] TO BUS 154 [DOWNTN      230.00] CIRCUIT 2 ---------------------------------- CONTINGENCY 153_154_2
                                                       153 MID230      230.00    154*DOWNTN      230.00 1     237.0    306.1    300.0     104.8

                                                  X-------- B U S --------X  V-CONT  V-INIT  X-------- B U S --------X  V-CONT  V-INIT
 'FLAPCO_EHV  ' BUSES WITH VOLTAGE GREATER THAN 1.0800:    152 MID500      500.00 1.08594 1.06031
```

Figure 4-12 Overloads and Voltage Violations Report with Fixed Shunt Reactor at Bus 151 Tripped as Initial Condition and A Few Transmission Lines Tripped

4.1.6 Performance during P7 events

Planning events for category P7 include contingencies that would take any two adjacent circuits on a common structure or a bipolar DC line, whose monopoles are usually physically close, too. The simulations in this work have assumed that all the parallel lines in the original *savnw.sav* case share a common structure for more than 1 mile, as indicated by footnote 11 on Table 11 in the standard TPL-001-4 [12]. In addition, the simulations have assumed that the location of the

bipolar DC line is such that the topography allows to install both monopoles on the same structures. Hence, the performance of the program will be evaluated based upon these assumptions.

First, for the transmission lines that share common structures, it is expected that both parallel lines should be tripped at the same time by the contingency. Figure 4-13 shows the automatically generated *.con file for the event E1. It confirms that both circuits are tripped simultaneously.

```
COM PSS(R)E 32
COM
COM LOSS OF MULTIPLE ADJACENT TRANSMISSION LINES ON COMMON STRUCTURE IN SUBSYSTEM 'FLAPCO'

CONTINGENCY 'L151_152'
DISCONNECT BRANCH FROM BUS 151 TO BUS 152 CIRCUIT 1
DISCONNECT BRANCH FROM BUS 151 TO BUS 152 CIRCUIT 2
END

CONTINGENCY 'L153_154'
DISCONNECT BRANCH FROM BUS 153 TO BUS 154 CIRCUIT 1
DISCONNECT BRANCH FROM BUS 153 TO BUS 154 CIRCUIT 2
END

END
```

Figure 4-13 Contingencies for E1 Events in Category P7

The importance that these parallel lines have in the modified base case is confirmed by looking at the results of the multiple AC contingency calculations in Figure 4-14. In both instances, a converged solution is not reached. The reason is similar to the results explained earlier when the multi-section line in LIGHTCO was removed; that is, the original *savnw.sav* case has a lack of robustness that makes it prone to diverge before single or multiple contingencies.

A	B	C	D	E	F	G
CONTINGENCY	EVENTS	CONVERGED	CONVERGENCE STATE	ISLANDS	MVAWORST	MVATOTAL
BASE CASE	BASE CASE	TRUE	Met convergence tolerance	0	0.0046	0.0140
L151_152	DISCONNECT BRANCH FROM BUS 151 [NUCPANT 500.00] TO BUS 152 [MID500 500.00] CIRCUIT 1	FALSE	Blown up	0	406.2475	1662.5591
	DISCONNECT BRANCH FROM BUS 151 [NUCPANT 500.00] TO BUS 152 [MID500 500.00] CIRCUIT 2					
L153_154	DISCONNECT BRANCH FROM BUS 153 [MID230 230.00] TO BUS 154 [DOWNTN 230.00] CIRCUIT 1	FALSE	Blown up	0	475.9708	1699.0116
	DISCONNECT BRANCH FROM BUS 153 [MID230 230.00] TO BUS 154 [DOWNTN 230.00] CIRCUIT 2					

Figure 4-14 Results for E1 Events in Category P7

As for the bipolar DC line, Figure 4-15 shows the *.con file automatically generated by the program, followed by Figure 4-16 with the results after the multiple AC contingency calculations. The results indicate that in this case a feasible solution is reached before a multiple contingency that trips both monopoles at the same time. Neither flow nor voltage violations were found for this planning event E2.

```
COM PSS(R)E 32
COM
COM LOSS OF A BIPOLAR DC LINE IN SUBSYSTEM 'FLAPCO'

CONTINGENCY 'D152_153'
BLOCK TWOTERMDC 1_1
BLOCK TWOTERMDC 1_2
END

END
```

Figure 4-15 Contingencies for E2 Events in Category P7

Label	Converged	Maximum Bus Mismatch	System Mismatch	Solution termination state
BASE CASE	☑	0.00	0.01	MET CONVERGENCE TOLERANCE
D152_153	☑	0.36	1.63	MET CONVERGENCE TOLERANCE

Figure 4-16 Results for E2 Events in Category P7

4.2 Coordination between Adjacent Planning Coordinators and Transmission Planners

Requirements R3 and R4 in the TPL-001-04 standard state that Planning Coordinators and Transmission Planners must coordinate with adjacent Planning Coordinators and Transmission Planners to make sure that any contingencies that could affect their systems are included in the contingency list used for their studies and planning assessments [12].

These are very important requirements considering that this lack of communication and coordination may have played an essential role in the 2003 cascading outage of transmission and generation facilities in the North American Eastern Interconnection.

To demonstrate how this type of coordination can be conducted with the programs developed, the IEEE 300 bus system has been used to perform the events for the categories included in this work. This time, a full description of the steps will be given.

First, all the *.sav, *.mon, *.sub, *.trp files and the ten Python scripts developed should be located inside the same folder. Then Windows PowerShell or any other terminal program could be used to run the Home.py main program.

For this study, the areas in the original IEEE 300-bus system have been renamed to CAESS for area 1, DELSUR for area 2, and VTES for area 3. A few generators have also been added to the original base case in different buses within area VTES to simulate re-dispatch of generation when needed. Finally, ratings Rate A and Rate B for transmission lines and transformers have been estimated as that information is not included in the original IEEE 300-bus system.

Because all contingencies will be run in VTES, the *.mon, *.sub, and *.trp files must be named after that area. Figure 4-17 shows the single line diagram with the areas. Based on the TPL-001-4 standard, CAESS shall coordinate with both DELSUR and VTES.

To perform the simulation correctly, the content of the *.mon file should be modified to specify the branches and buses that wish to be monitored in CAESS in order to determine whether contingencies occurring within VTES cause violations in CAESS. If available, therefore, the specific criteria used by the Planning Coordinator or Transmission Planner in CAESS can be indicated in the *.mon file.

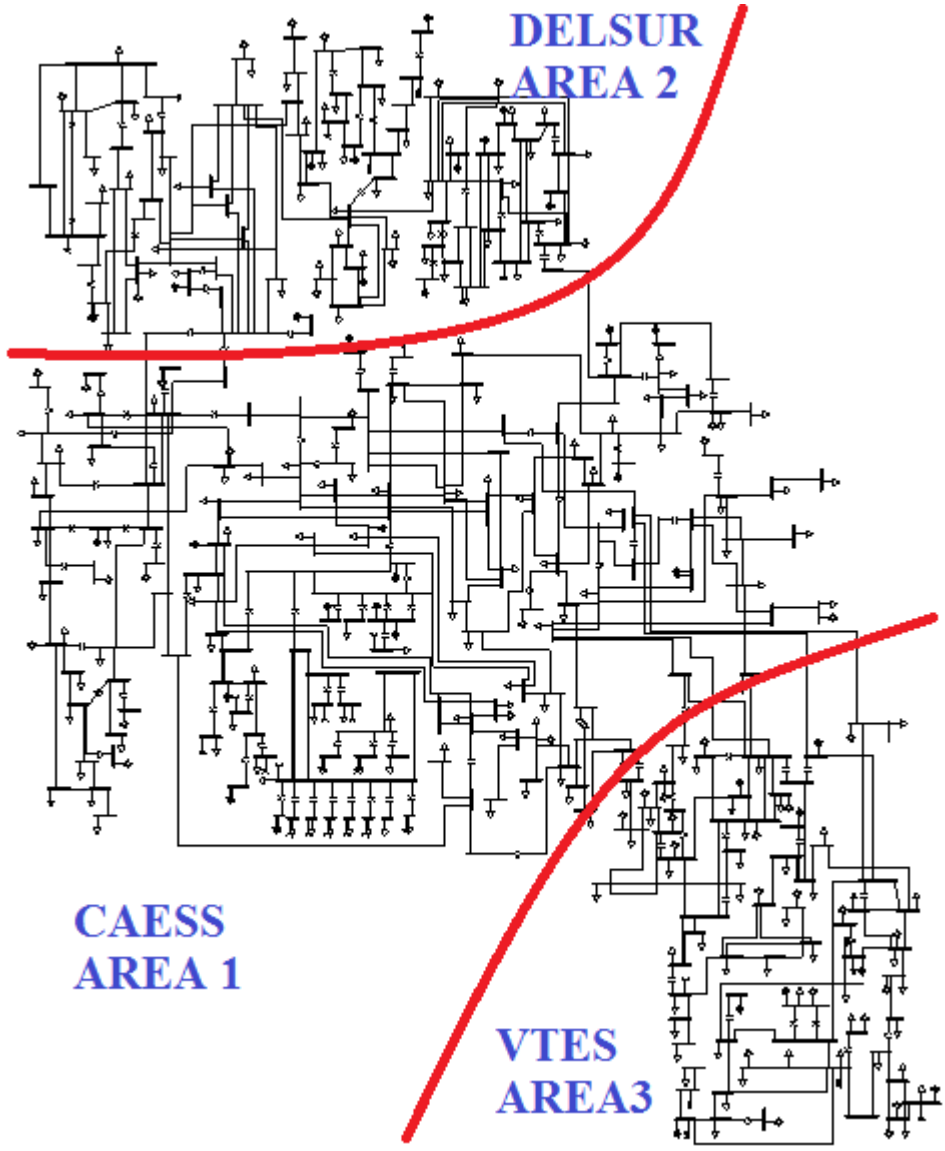

Figure 4-17 IEEE 300-Bus System Used as Base Case

After completing the initial settings described above, the user can open Windows PowerShell and navigate to the folder with the files. Figure 4-18 illustrates the inputs from the user in this case if the events for the category P1 would like to be studied.

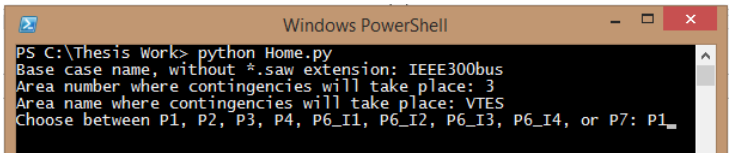

Figure 4-18 Input from the User in the Home.py Program

Note that any of the categories P1 through P4 and P6 through P7 can be input in this initial prompt. After pressing Enter, the program opens the respective contingency file script and proceeds to perform all the AC calculations for the planning events accordingly.

The program output the results in three files, *.acc, *.xls, and *.txt. Examples of what they look like and the information they contain have been shown in previous sections. The next three figures are just a few examples of the results for different contingencies in this particular study.

```
<---------- C O N T I N G E N C Y   E V E N T S ---------->  <-------- O V E R L O A D E D   L I N E S ---------->  <- MVA(MW)FLOW ->
     <---------- MULTI-SECTION LINE GROUPINGS ---------->  <------- F R O M ------->  <--------- T O --------->CKT  PRE-CNT  POST-CNT  RATING  PERCENT
DISCONNECT BRANCH FROM BUS 196 [3          115.00] TO BUS 2040 [3          115.00] CIRCUIT 1 ---------------------------------- CONTINGENCY 196_2040_1
                                                                                                                *** NONE ***

                                                X-------- B U S --------X  V-CONT  V-INIT  X-------- B U S --------X  V-CONT  V-INIT
    'CAESS_HV    ' BUSES WITH VOLTAGE LESS THAN 0.9500:     69 1          115.00 0.94603 0.96311      70 1          115.00 0.94989 0.95132
```

Figure 4-19 Low Voltage Violations in CAESS for E3 Events within VTES - Category P1

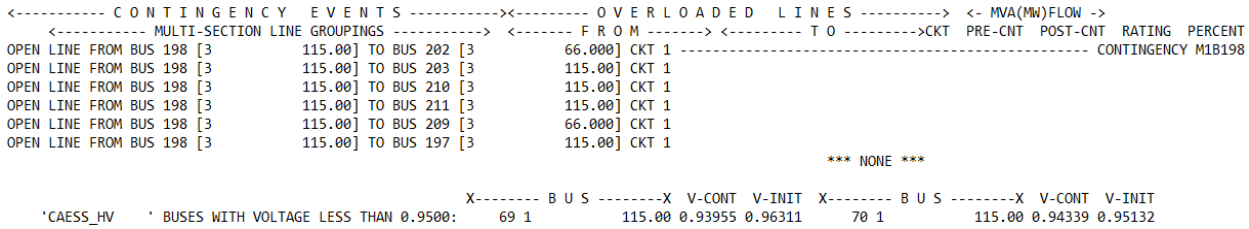

Figure 4-20 Low Voltage Violations in CAESS for E1 Events within VTES - Category P4

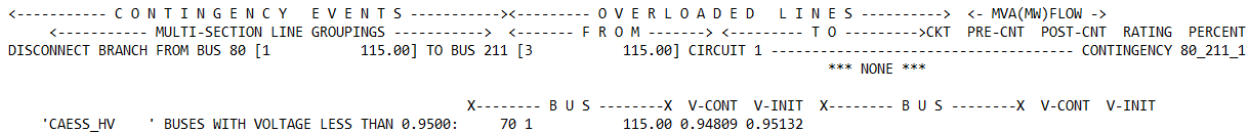

Figure 4-21 Low Voltage Violation in CAESS for E1 Event within VTES - Category P6 with Transmission Circuit 192 - 225 Tripped as Initial Condition

After looking at several of the reports and violations observed in the results, it became clear that voltage violations at buses 69 and 70 in CAESS were the most frequent violations for several contingencies occurring within VTES. Therefore, it is with this type of coordination between VTES and CAESS that the transmission planners in CAESS can identify such voltage violations and then determine what devices could be installed that would provide the needed reactive power support to avoid having the same violations when equipment operates or trips in VTES.

5. Conclusions

The results obtained from tests conducted with several base cases show that the programs developed accurately simulate most single and multiple contingency planning events based on the requirements specified in the TPL-001-4 standard for steady-state analysis. The algorithms developed for the programs greatly facilitate performing transmission system planning simulations to engineers, students and researchers as they automate most of the processes that generate and analyze the contingencies inherent to these simulations. The creation of output files with the simulation results has also been automated.

The programs and algorithms developed represent tools that effectively generalize the procedures needed to conduct steady state analysis in transmission system planning irrespective of the size of the system, imposing single or multiple contingencies while checking that applicable facility ratings are not exceeded and system steady state voltages and post-contingency voltage deviations are within acceptable limits. Still, the program codes could be further improved in the future in order to increase its performance and speed, as well as reduce its demand for computational resources during the AC contingency simulations.

This work also demonstrates how the Python programming language and software Power System Simulator for Engineering (PSS/E) can become powerful tools when they are combined, helping study very large sets of contingencies in a relatively short amount of time. Moreover, it shows how the tools and libraries available in Python and PSS/E can be applied to electric power transmission systems planning.

Steady-state planning events are simulated in this work following most of the criteria indicated in the standard TPL-001-4. Not only are single and multiple contingencies properly simulated, but also system adjustments are included for those categories where the standard allows for them, removing elements from the system that automatic controls are expected to disconnect for each contingency without operator intervention, redispatching generation units in order to meet the system demand before the loss of another generation unit, and including the expected automatic operation of devices that provide steady state control of the electrical system, such as phase-shifting transformers, load tap changing transformers, capacitors and inductors.

5.1 Future Research

Accurately simulating the steady-state planning events for categories P2, P5, and extreme events can be part of future research. Moreover, it would be interesting to incorporate in these simulations other protection schemes such as remedial action schemes (RAS) or special protection schemes (SPS), as well as those schemes based on wide area phasor measurement units or PMUs.

As this work only deals with the steady state analysis in the TPL-001-4 standard, future work can try to develop and implement similar programs for the stability and short-circuit analysis that are also required to be conducted annually by Planning Coordinators and Transmission Planners in the standard. This would imply looking in detail at the dynamic response of the power system, especially during the period of time when protection schemes are operating. Transient voltages, for instance, normally increase during switching operations.

TPL-001-4 also calls for the creation of a list where the contingencies expected to produce more severe system impacts are identified. Although in this work the programs have been developed to analyze all the possible contingencies in the Bulk Electric System, ranking single and multiple contingency events to focus on the most relevant ones in future transmission system planning research can also reduce considerably the time required to compute such simulations in large power systems.

6. References

[1] J. D. Glover, M. S. Sarma, and T. J. Overbye, *Power system analysis and design*, 5th ed. Stamford, CT: Cengage Learning, 2012.

[2] North American Electric Reliability Corporation. *About NERC.* Available: www.nerc.com

[3] S. Horowitz, A. G. Phadke, and B. Renz, "The Future of Power Transmission," *Power and Energy Magazine, IEEE,* vol. 8, pp. 34-40, 2010.

[4] D. J. Morrow and R. E. Brown, "Future vision - The Challenge of Effective Transmission Planning," *Power and Energy Magazine, IEEE,* vol. 5, pp. 36-45, 2007.

[5] North American Electric Reliability Corporation. *Reliability Standards - NERC.* Available: http://www.nerc.net/standardsreports/standardssummary.aspx

[6] G. Andersson, P. Donalek, R. Farmer, N. Hatziargyriou, I. Kamwa, P. Kundur, *et al.*, "Causes of the 2003 major grid blackouts in North America and Europe, and recommended means to improve system dynamic performance," *Power Systems, IEEE Transactions on,* vol. 20, pp. 1922-1928, 2005.

[7] U.S.-Canada Power System Outage Task Force, "Final Report on the August 14, 2003 Blackout in the United States and Canada: Causes and Recommendations," U.S. Department of Energy and Natural Resources Canada, Ed., ed, 2004.

[8] North American Electric Reliability Corporation, "System Performance Under Normal (No Contingency) Conditions (Category A)," in *TPL-001-0.1*, North American Electric Reliability Corporation, Ed., ed, 2009.

[9] North American Electric Reliability Corporation, "System Performance Following Loss of a Single Bulk Electric System Element (Category B)," in *TPL-002-0b*, North American Electric Reliability Corporation, Ed., ed, 2011.

[10] North American Electric Reliability Corporation, "System Performance Following Loss of Two or More Bulk Electric System Elements (Category C)," in *TPL-003-0b*, North American Electric Reliability Corporation, Ed., ed, 2013.

[11] North American Electric Reliability Corporation, "System Performance Following Extreme Events Resulting in the Loss of Two or More Bulk Electric System Elements (Category D)," in *TPL-004-0a*, North American Electric Reliability Corporation, Ed., ed, 2013.

[12] North American Electric Reliability Corporation, "Transmission System Planning Performance Requirements," in *TPL-001-4*, North American Electric Reliability Corporation, Ed., ed, 2013.

[13] J. Marini, "Up and Running with Python," ed: Lynda.com, 2013.

[14] Massachusetts Institute of Technology, "Python Tutorial," ed: MIT OpenCourseWare (OCW), 2011.

[15] A. B. Downey. (2012). *Think Python: How to Think Like a Computer Scientist* [Online Book]. Available: http://www.greenteapress.com/thinkpython/html/

[16] Z. A. Shaw. (2013). *Learn Python the Hard Way (Third ed.)* [Online Book]. Available: http://learnpythonthehardway.org/book/index.html

[17] F. R. Muhammad Usman Khalid, Waqar Akhtar, Muhammad Haroon Ghauri, *Python Based Power System Automation in PSS/E*, First Edition ed.: Lulu, 2014.

[18] Siemens Power Technologies International (Siemens PTI). (2009). *PSS®E Application Program Interface (API)*.

[19] Siemens Power Technologies International (Siemens PTI). (2009). *Program Operation Manual*.

[20] whit. (2011). *Run PSSE From Python and Not the Other Way Around*. Available: http://www.whit.com.au/blog/2011/07/run-psse-from-python-and-not-other-way/

Appendix A Python Codes

A.1 Category P1

```
"""
THIS PROGRAM PERFORMS P1 CONTINGENCIES PER TPL-001-4
EVENTS 1 THROUGH 5
"""

import os,sys

PSSE_LOCATION = r"C:\Program Files\PTI\PSSE32\PSSBIN"
sys.path.append(PSSE_LOCATION)
os.environ['PATH'] = os.environ['PATH'] + ';' +  PSSE_LOCATION

import psspy
import redirect
redirect.psse2py()

from sys import argv
if len(sys.argv) > 1:
    script, CASE1, areas, subSysLabel = argv
else:
    CASE1 = raw_input("Base case name, without *.sav extension: ")
    areas = raw_input("Area number where contingencies will take place: ")
    subSysLabel = raw_input("Area name where contingencies will take place: ")

areas = [int(areas)]


#---------------------------------
# PSS/E Saved case

CASE = CASE1 + '.sav'

psspy.psseinit(50000)
psspy.prompt_output(4,"",[0,0])
psspy.alert_output(4,"",[0,0])

# Defining a subsystem with buses in study area
sid = 0
usekv = 1
numarea = len(areas)
subsysID = sid


"""
Program to perform all P1_E1 contingencies.
"""
psspy.case(CASE)
basekv=[1,999]
psspy.bsys(sid, usekv, basekv, numarea, areas)

# CON file name with contingencies
conFileName1 = "P1_E1"
conFileName = conFileName1 + '.con'

# Returns a list with Machine Identifiers for all in-service machines
# at in-service plants
ierr, (allMachID,) = psspy.amachchar(subsysID, 1, 'ID')
print "machID =", allMachID

# Returns a list with Bus Numbers for all in-service machines
# at in-service plants
ierr, (busNums,) = psspy.amachint(subsysID, 1, 'NUMBER')
print "list of bus numbers ", busNums
```

```
# Opens a file object for both writing and reading. Overwrites the
# existing file if the file exists. If the file does not exist, creates
# a new file for reading and writing.
conFile = open(conFileName, 'w+')

# .con file header
conFile.write("COM PSS(R)E 32\n")
conFile.write("COM\n")
conFile.write("COM LOSS OF SINGLE GENERATOR IN SUBSYSTEM '%s' \n\n"
              % subSysLabel)

# Contingencies for removing each specified machine
for machid, busnum in zip(allMachID, busNums):
        conFile.write("CONTINGENCY 'M%dB%d'\n" %( int(machid), busnum))
        conFile.write('REMOVE MACHINE %s FROM BUS %d \n' %(machid, busnum))
        conFile.write('END \n\n')

conFile.write('END')
conFile.close()

# Run Multi-level AC Contingency Solution
psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
psspy.dfax([1,1],subSysLabel,subSysLabel,conFileName,
           conFileName1)

psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                       [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                       [subSysLabel + '_DSP',subSysLabel + '_HV',
                        subSysLabel + '_HV',subSysLabel + '_HV',
                        subSysLabel + '_HV',
                        subSysLabel + '_BULK',subSysLabel + '_BULK'],
                      conFileName1 + '.dfx',conFileName1,"","",subSysLabel + '.trp')

# Generate final excel file with results
import pssexcel

ACCFile=sys.path[0]

pssexcel.accc(conFileName1 + '.acc',['e','b','v'],colabel='',stype='contingency',
              busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
              xlsfile='ACCCResults_' + conFileName1 + '.xls',sheet='',
              overwritesheet=True,show=False)#,ratecon='a',baseflowvio=False,
              #basevoltvio=False,flowlimit=90.0,flowchange=5.0,voltchange=0.0)

# Generate final text file with results
psspy.report_output(islct=2, filarg="./" + 'ACCCResults_' + conFileName1 + '.txt',
                    options=[0])
psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                               [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                               conFileName1)
psspy.report_output(islct=2, options=[0])


"""
Program to perform all P1_E2 contingencies.
"""
psspy.case(CASE)
basekv=[100,999]
psspy.bsys(sid, usekv, basekv, numarea, areas)

# CON file name with contingencies
conFileName1 = "P1_E2"
conFileName = conFileName1 + '.con'

# Returns a list with Circuit Identifiers for only in-service
# non-transformer branches
ierr, (allBrnID,) = psspy.abrnchar(subsysID, 1, 3, 1, 1, 'ID')
print "BrnID =", allBrnID

# Returns a list with From Bus Numbers for only in-service
# non-transformer branches
```

```
ierr, (FrombusNums,) = psspy.abrnint(subsysID, 1, 3, 1, 1, 'FROMNUMBER')
print "list of FROM bus numbers Brn ", FrombusNums

# Returns a list with To Bus Numbers for only in-service
# non-transformer branches
ierr, (TobusNums,) = psspy.abrnint(subsysID, 1, 3, 1, 1, 'TONUMBER')
print "list of TO bus numbers Brn ", TobusNums

# Opens a file object for both writing and reading. Overwrites the
# existing file if the file exists. If the file does not exist, creates
# a new file for reading and writing.
conFile = open(conFileName, 'w+')

# .con file header
conFile.write("COM PSS(R)E 32\n")
conFile.write("COM\n")
conFile.write("COM LOSS OF SINGLE TRANSMISSION LINE IN SUBSYSTEM '%s' \n\n"
              % subSysLabel)

# Contingencies for removing each specified Transmission Circuit
for Brnid, frombusnum, tobusnum in zip(allBrnID, FrombusNums, TobusNums):
        conFile.write("CONTINGENCY '%d_%d_%d'\n"
                      %(frombusnum, tobusnum, int(Brnid)))
        conFile.write('DISCONNECT BRANCH FROM BUS %d TO BUS %d CIRCUIT %d\n'
                       %(frombusnum, tobusnum, int(Brnid)))
        conFile.write('END \n\n')

conFile.write('END')
conFile.close()

# Run Multi-level AC Contingency Solution
psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
psspy.dfax([1,1],subSysLabel,subSysLabel,conFileName,
           conFileName1)

psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                       [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                       [subSysLabel + '_DSP',subSysLabel + '_HV',
                        subSysLabel + '_HV',subSysLabel + '_HV',
                        subSysLabel + '_HV',
                        subSysLabel + '_BULK',subSysLabel + '_BULK'],
                      conFileName1 + '.dfx',conFileName1,"","",subSysLabel + '.trp')


# Generate final excel file with results
pssexcel.accc(conFileName1 + '.acc',['e','b','v'],colabel='',stype='contingency',
              busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
              xlsfile='ACCCResults_' + conFileName1 + '.xls',sheet='',
              overwritesheet=True,show=False)

# Generate final text file with results
psspy.report_output(islct=2, filarg="./" + 'ACCCResults_' + conFileName1 + '.txt',
                    options=[0])
psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                               [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                               conFileName1)
psspy.report_output(islct=2, options=[0])


"""
Program to perform all P1_E3 contingencies.
"""
psspy.case(CASE)
basekv=[100,999]
psspy.bsys(sid, usekv, basekv, numarea, areas)

# CON file name with contingencies
conFileName1 = "P1_E3"
conFileName = conFileName1 + '.con'

# Returns a list with Circuit Identifiers for only in-service
```

```
# two-winding transformers branches
ierr, (allTrnID,) = psspy.atrnchar(subsysID, 1, 3, 1, 1, 'ID')
print "TrnID =", allTrnID

# Returns a list with From Bus Numbers for only in-service
# two-winding transformers branches
ierr, (FrombusNums,) = psspy.atrnint(subsysID, 1, 3, 1, 1, 'FROMNUMBER')
print "list of FROM bus numbers Trn ", FrombusNums

# Returns a list with To Bus Numbers for only in-service
# two-winding transformers branches
ierr, (TobusNums,) = psspy.atrnint(subsysID, 1, 3, 1, 1, 'TONUMBER')
print "list of TO bus numbers Trn ", TobusNums

# Opens a file object for both writing and reading. Overwrites the
# existing file if the file exists. If the file does not exist, creates
# a new file for reading and writing.
conFile = open(conFileName, 'w+')

# .con file header
conFile.write("COM PSS(R)E 32\n")
conFile.write("COM\n")
conFile.write("COM LOSS OF SINGLE TRANSFORMER IN SUBSYSTEM '%s' \n\n"
              % subSysLabel)

# Contingencies for removing each specified Transformer
for Trnid, frombusnum, tobusnum in zip(allTrnID, FrombusNums, TobusNums):
        conFile.write("CONTINGENCY '%d_%d_%d'\n"
                      %(frombusnum, tobusnum, int(Trnid)))
        conFile.write('DISCONNECT BRANCH FROM BUS %d TO BUS %d CIRCUIT %d\n'
                      %(frombusnum, tobusnum, int(Trnid)))
        conFile.write('END \n\n')

conFile.write('END')
conFile.close()

# Run Multi-level AC Contingency Solution
psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
psspy.dfax([1,1],subSysLabel,subSysLabel,conFileName,
           conFileName1)

psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                       [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                       [subSysLabel + '_DSP',subSysLabel + '_HV',
                        subSysLabel + '_HV',subSysLabel + '_HV',
                        subSysLabel + '_HV',
                        subSysLabel + '_BULK',subSysLabel + '_BULK'],
                      conFileName1 + '.dfx',conFileName1,"","",subSysLabel + '.trp')


# Generate final excel file with results
pssexcel.accc(conFileName1 + '.acc',['e','b','v'],colabel='',stype='contingency',
              busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
              xlsfile='ACCCResults_' + conFileName1 + '.xls',sheet='',
              overwritesheet=True,show=False)

# Generate final text file with results
psspy.report_output(islct=2, filarg="./" + 'ACCCResults_' + conFileName1 + '.txt',
                    options=[0])
psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                               [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                               conFileName1)
psspy.report_output(islct=2, options=[0])


"""
Program to perform all P1_E4 contingencies.
"""
psspy.case(CASE)
basekv=[100,999]
psspy.bsys(sid, usekv, basekv, numarea, areas)
```

```
# CON file name with contingencies
conFileName1 = "P1_E4"
conFileName = conFileName1 + '.con'

### Part for FIXED Shunts Starts ###
# Returns a list with Shunt Identifiers for only in-service
# fixed shunts at in-service buses
ierr, (allFxShuntID,) = psspy.afxshuntchar(subsysID, 1, 'ID')
print "FxShuntID =", allFxShuntID

# Returns a list with Bus Numbers for only in-service
# fixed shunts at in-service buses
ierr, (busNums,) = psspy.afxshuntint(subsysID, 1, 'NUMBER')
print "list of bus numbers Fixed Shunts ", busNums

### Part for SWITCHED Shunts Starts ###
# Returns a list with Bus Numbers for only in-service
# switched shunts at in-service buses
ierr, (busNumsSw,) = psspy.aswshint(sid, 1, 'NUMBER')
print "list of bus numbers Switched Shunts ", busNumsSw

# Opens a file object for both writing and reading. Overwrites the
# existing file if the file exists. If the file does not exist, creates
# a new file for reading and writing.
conFile = open(conFileName, 'w+')

# .con file header
conFile.write("COM PSS(R)E 32\n")
conFile.write("COM\n")
conFile.write("COM LOSS OF SINGLE SHUNT DEVICE IN SUBSYSTEM '%s' \n\n"
              % subSysLabel)

# Contingencies for removing each specified fixed shunt
for FxShuntid, busnum in zip(allFxShuntID, busNums):
        conFile.write("CONTINGENCY 'F%dB%d'\n" %( int(FxShuntid), busnum))
        conFile.write('REMOVE SHUNT %s FROM BUS %d\n' %(FxShuntid, busnum))
        conFile.write('END\n\n')

# Contingencies for removing each specified switched shunt
for busnum in busNumsSw:
        conFile.write("CONTINGENCY 'SwB%d'\n" %busnum)
        conFile.write('REMOVE SWSHUNT FROM BUS %d\n' %busnum)
        conFile.write('END\n\n')

conFile.write('END')
conFile.close()

# Run Multi-level AC Contingency Solution
psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
psspy.dfax([1,1],subSysLabel,subSysLabel,conFileName,
           conFileName1)

psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                       [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                       [subSysLabel + '_DSP',subSysLabel + '_HV',
                        subSysLabel + '_HV',subSysLabel + '_HV',
                        subSysLabel + '_HV',
                        subSysLabel + '_BULK',subSysLabel + '_BULK'],
                      conFileName1 + '.dfx',conFileName1,"","",subSysLabel + '.trp')


# Generate final excel file with results
pssexcel.accc(conFileName1 + '.acc',['e','b','v'],colabel='',stype='contingency',
              busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
              xlsfile='ACCCResults_' + conFileName1 + '.xls',sheet='',
              overwritesheet=True,show=False)

# Generate final text file with results
psspy.report_output(islct=2, filarg="./" + 'ACCCResults_' + conFileName1 + '.txt',
                    options=[0])
```

```
psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                               [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                               conFileName1)
psspy.report_output(islct=2, options=[0])


"""
Program to perform all P1_E5 contingencies.
"""
psspy.case(CASE)
basekv=[1,999]
psspy.bsys(sid, usekv, basekv, numarea, areas)

# CON file name with contingencies
conFileName1 = "P1_E5"
conFileName = conFileName1 + '.con'

# Returns a list with DC Line names for only in-service
# two-terminal dc lines
ierr, (allDCBrnNames,) = psspy.a2trmdcchar(subsysID, 3, 1, 'DCNAME')
print "DCBrnID =", allDCBrnNames

# Returns a list with From or Rectifier Bus Numbers for only in-service
# two-terminal dc lines
ierr, (FrombusNums,) = psspy.a2trmdcint(subsysID, 3, 1, 'FROMNUMBER')
print "list of FROM bus numbers DCBrn ", FrombusNums

# Returns a list with To or Inverter Bus Numbers for only in-service
# two-terminal dc lines
ierr, (TobusNums,) = psspy.a2trmdcint(subsysID, 3, 1, 'TONUMBER')
print "list of TO bus numbers DCBrn ", TobusNums

# Opens a file object for both writing and reading. Overwrites the
# existing file if the file exists. If the file does not exist, creates
# a new file for reading and writing.
conFile = open(conFileName, 'w+')

# .con file header
conFile.write("COM PSS(R)E 32\n")
conFile.write("COM\n")
conFile.write("COM LOSS OF SINGLE POLE OF A DC LINE IN SUBSYSTEM '%s' \n\n"
              % subSysLabel)

# Contingencies for removing each specified Pole of a DC Line
for DCBrnname, frombusnum, tobusnum in zip(allDCBrnNames, FrombusNums, TobusNums):
        conFile.write("CONTINGENCY '%d_%d_%s'\n"
                      %(frombusnum, tobusnum, DCBrnname))
        conFile.write('BLOCK TWOTERMDC %s\n' %DCBrnname)
        conFile.write('END \n\n')

conFile.write('END')
conFile.close()

# Run Multi-level AC Contingency Solution
psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
psspy.dfax([1,1],subSysLabel,subSysLabel,conFileName,
           conFileName1)

psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                       [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                       [subSysLabel + '_DSP',subSysLabel + '_HV',
                        subSysLabel + '_HV',subSysLabel + '_HV',
                        subSysLabel + '_HV',
                        subSysLabel + '_BULK',subSysLabel + '_BULK'],
                      conFileName1 + '.dfx',conFileName1,"","",subSysLabel + '.trp')


# Generate final excel file with results
pssexcel.accc(conFileName1 + '.acc',['e','b','v'],colabel='',stype='contingency',
              busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
              xlsfile='ACCCResults_' + conFileName1 + '.xls',sheet='',
```

```
            overwritesheet=True,show=False)

# Generate final text file with results
psspy.report_output(islct=2, filarg="./" + 'ACCCResults_' + conFileName1 + '.txt',
                    options=[0])
psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                               [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                               conFileName1)
psspy.report_output(islct=2, options=[0])


print('\nP1 Category Complete!')
print('NOTE: It is highly recommended to create a new folder where')
print('      to move all the files created by this program.\n')
```

A.2 Category P2

```
"""
THIS PROGRAM PERFORMS P2 CONTINGENCIES PER TPL-001-4
EVENTS 1 THROUGH 2
"""

import os,sys

PSSE_LOCATION = r"C:\Program Files\PTI\PSSE32\PSSBIN"
sys.path.append(PSSE_LOCATION)
os.environ['PATH'] = os.environ['PATH'] + ';' +  PSSE_LOCATION

import psspy
import redirect
redirect.psse2py()

from sys import argv
if len(sys.argv) > 1:
    script, CASE1, areas, subSysLabel = argv
else:
    CASE1 = raw_input("Base case name, without *.sav extension: ")
    areas = raw_input("Area number where contingencies will take place: ")
    subSysLabel = raw_input("Area name where contingencies will take place: ")

areas = [int(areas)]


#---------------------------------
# PSS/E Saved case

CASE = CASE1 + '.sav'

psspy.psseinit(50000)
psspy.prompt_output(4,"",[0,0])
psspy.alert_output(4,"",[0,0])

# Defining a subsystem with buses in study area
sid = 0
usekv = 1
numarea = len(areas)
subsysID = sid


"""
Program to perform all P2_E1 contingencies.
"""
psspy.case(CASE)
basekv=[100,999]
psspy.bsys(sid, usekv, basekv, numarea, areas)
```

```
psspy.multisection_reporting(0)
# CON file name with contingencies
conFileName1 = "P2_E1"
conFileName = conFileName1 + '.con'

# Returns a list with Circuit Identifiers for only in-service
# non-transformer branches and two-winding transformers
ierr, (allBrnID,) = psspy.abrnchar(subsysID, 1, 3, 3, 1, 'ID')
print "BrnID =", allBrnID

# Returns a list with From Bus Numbers for only in-service
# non-transformer branches and two-winding transformers
ierr, (FrombusNums,) = psspy.abrnint(subsysID, 1, 3, 3, 1, 'FROMNUMBER')
print "list of FROM bus numbers Brn ", FrombusNums

# The next section is intended to identify dummy FROM buses
# used in multi-section lines
global FromDummy
FromDummy = []

for isdummy in FrombusNums:
    ierr, ival = psspy.busint(isdummy, 'DUMMY')
    FromDummy.append(ival)

print "list of FROM dummy buses ", FromDummy

# Returns a list with To Bus Numbers for only in-service
# non-transformer branches and two-winding transformers
ierr, (TobusNums,) = psspy.abrnint(subsysID, 1, 3, 3, 1, 'TONUMBER')
print "list of TO bus numbers Brn ", TobusNums

# The next section is intended to identify dummy TO buses
# used in multi-section lines
global ToDummy
ToDummy = []

for isdummy in TobusNums:
    ierr, ival = psspy.busint(isdummy, 'DUMMY')
    ToDummy.append(ival)

print "list of TO dummy buses ", ToDummy

# Opens a file object for both writing and reading. Overwrites the
# existing file if the file exists. If the file does not exist, creates
# a new file for reading and writing.
conFile = open(conFileName, 'w+')

# .con file header
conFile.write("COM PSS(R)E 32\n")
conFile.write("COM\n")
conFile.write("COM OPENING OF A LINE SECTION W/O A FAULT '%s' \n\n"
              % subSysLabel)

# Contingencies for opening each specified Line Section
for Brnid, frombusnum, tobusnum, fromdummy, todummy in zip(allBrnID, FrombusNums, TobusNums,
FromDummy, ToDummy):
    if fromdummy == 1 or todummy == 1:
        conFile.write("CONTINGENCY '%d_%d_%d'\n"
                      %(frombusnum, tobusnum, int(Brnid)))
        conFile.write('OPEN BRANCH FROM BUS %d TO BUS %d CIRCUIT %d\n'
                       %(frombusnum, tobusnum, int(Brnid)))
        conFile.write('END \n\n')

conFile.write('END')
conFile.close()

# Run Multi-level AC Contingency Solution
psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
psspy.dfax([1,1],subSysLabel,subSysLabel,conFileName,
           conFileName1)
```

```
psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                       [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                       [subSysLabel + '_DSP',subSysLabel + '_HV',
                        subSysLabel + '_HV',subSysLabel + '_HV',
                        subSysLabel + '_HV',
                        subSysLabel + '_BULK',subSysLabel + '_BULK'],
                      conFileName1 + '.dfx',conFileName1,"","",subSysLabel + '.trp')

# Generate final excel file with results
import pssexcel

ACCFile=sys.path[0]

pssexcel.accc(conFileName1 + '.acc',['e','b','v'],colabel='',stype='contingency',
              busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
              xlsfile='ACCCResults_' + conFileName1 + '.xls',sheet='',
              overwritesheet=True,show=False)

# Generate final text file with results
psspy.report_output(islct=2, filarg="./" + 'ACCCResults_' + conFileName1 + '.txt',
                    options=[0])
psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                               [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                               conFileName1)
psspy.report_output(islct=2, options=[0])

psspy.multisection_reporting(1)


"""
Program to perform all P2_E2 contingencies.
"""
psspy.case(CASE)
basekv=[100,999]
psspy.bsys(sid, usekv, basekv, numarea, areas)

# CON file name with contingencies
conFileName1 = "P2_E2"
conFileName = conFileName1 + '.con'

# Returns a list with Circuit Identifiers for only in-service
# non-transformer branches
ierr, (allBrnID,) = psspy.abrnchar(subsysID, 1, 3, 1, 1, 'ID')
print "BrnID =", allBrnID

# Returns a list with From Bus Numbers for only in-service
# non-transformer branches
ierr, (FrombusNums,) = psspy.abrnint(subsysID, 1, 3, 1, 1, 'FROMNUMBER')
print "list of FROM bus numbers Brn ", FrombusNums

# Returns a list with To Bus Numbers for only in-service
# non-transformer branches
ierr, (TobusNums,) = psspy.abrnint(subsysID, 1, 3, 1, 1, 'TONUMBER')
print "list of TO bus numbers Brn ", TobusNums

# Returns a list with Branch impedances for only in-service
# non-transformer branches
ierr, (RXarray,) = psspy.abrncplx(subsysID, 1, 3, 1, 1, 'RX')
print "list of Branch impedance in pu ", RXarray

# The next section is intended to identify branches
# with impedances lower than the Zero impedance line threshold
global IsSection
IsSection = []

for issection in RXarray:
    magnitude = abs(issection)

    if magnitude <= 0.000100:
        IsSection.append(1)
    else:
```

```
        IsSection.append(0)

print "list of Identified Sections ", IsSection

# Opens a file object for both writing and reading. Overwrites the
# existing file if the file exists. If the file does not exist, creates
# a new file for reading and writing.
conFile = open(conFileName, 'w+')

# .con file header
conFile.write("COM PSS(R)E 32\n")
conFile.write("COM\n")
conFile.write("COM LOSS OF BUS SECTION DUE TO A FAULT '%s' \n\n"
              % subSysLabel)

# Contingencies for removing each specified Bus Section
for Brnid, frombusnum, tobusnum, issection in zip(allBrnID, FrombusNums, TobusNums, IsSection):
    if  issection == 1:
        conFile.write("COM DISCONNECT BUS SECTION BETWEEN BUS %d AND BUS %d DUE TO A FAULT\n"
                      %(frombusnum, tobusnum))
        conFile.write("CONTINGENCY '%d_%d_%d'\n"
                     %(frombusnum, tobusnum, int(Brnid)))
        conFile.write('DISCONNECT BRANCH FROM BUS %d TO BUS %d CIRCUIT %d\n'
                      %(frombusnum, tobusnum, int(Brnid)))
        conFile.write('END \n\n')

conFile.write('END')
conFile.close()

# Run Multi-level AC Contingency Solution
psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
psspy.dfax([1,1],subSysLabel,subSysLabel,conFileName,
           conFileName1)

psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                       [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                       [subSysLabel + '_DSP',subSysLabel + '_HV',
                        subSysLabel + '_HV',subSysLabel + '_HV',
                        subSysLabel + '_HV',
                        subSysLabel + '_BULK',subSysLabel + '_BULK'],
                      conFileName1 + '.dfx',conFileName1,"","",subSysLabel + '.trp')

# Generate final excel file with results
pssexcel.accc(conFileName1 + '.acc',['e','b','v'],colabel='',stype='contingency',
              busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
              xlsfile='ACCCResults_' + conFileName1 + '.xls',sheet='',
              overwritesheet=True,show=False)

# Generate final text file with results
psspy.report_output(islct=2, filarg="./" + 'ACCCResults_' + conFileName1 + '.txt',
                    options=[0])
psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                               [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                               conFileName1)
psspy.report_output(islct=2, options=[0])


print('\nP2 Category Complete!')
print('NOTE: It is highly recommended to create a new folder where')
print('      to move all the files created by this program.\n')
```

A.3 Category P3

```
"""
THIS PROGRAM PERFORMS P3 CONTINGENCIES PER TPL-001-4
EVENTS 1 THROUGH 5
"""

"""
First Part is to calculate Initial Conditions P3.
That is, loss of generator unit followed by
System adjustments
"""

import os,sys

PSSE_LOCATION = r"C:\Program Files\PTI\PSSE32\PSSBIN"
sys.path.append(PSSE_LOCATION)
os.environ['PATH'] = os.environ['PATH'] + ';' +  PSSE_LOCATION

import psspy
import redirect
redirect.psse2py()

from sys import argv
if len(sys.argv) > 1:
    script, CASE1, areas, subSysLabel = argv
else:
    CASE1 = raw_input("Base case name, without *.sav extension: ")
    areas = raw_input("Area number where contingencies will take place: ")
    subSysLabel = raw_input("Area name where contingencies will take place: ")

areas = [int(areas)]


#---------------------------------
# PSS/E Saved case

CASE = CASE1 + '.sav'

psspy.psseinit(50000)
psspy.prompt_output(4,"",[0,0])
psspy.alert_output(4,"",[0,0])

# Defining a subsystem with buses in study area
sid = 0
usekv = 1
numarea = len(areas)
subsysID = sid

psspy.case(CASE)
basekv=[1,999]
psspy.bsys(sid, usekv, basekv, numarea, areas)

conFileName1 = "P6_P1_E1"

# Returns a list with Machine Identifiers for all in-service machines
# at in-service plants
ierr, (allMachID,) = psspy.amachchar(subsysID, 1, 'ID')
print "machID =", allMachID

# Returns a list with Bus Numbers for all in-service machines
# at in-service plants
ierr, (busNums,) = psspy.amachint(subsysID, 1, 'NUMBER')
print "list of bus numbers ", busNums

# Loop to lose each generator, perform system adjustments,
# and study the respective events
for machid, busnum in zip(allMachID, busNums):
    """
```

```
For Loop Part to create a temporary contingency file for each generator
lost in ORIGINAL case file
"""
psspy.case(CASE)
psspy.prompt_output(4,"",[0,0])
psspy.alert_output(4,"",[0,0])

# Opens a file object for both writing and reading. Overwrites the
# existing file if the file exists. If the file does not exist, creates
# a new file for reading and writing.
conFile = open(conFileName1 + '.con', 'w+')

# .con file header
conFile.write("COM PSS(R)E 32\n")
conFile.write("COM\n")
conFile.write("COM LOSS OF SINGLE GENERATOR IN SUBSYSTEM '%s' \n\n"
              % subSysLabel)

# Contingency for removing each specified machine, one at a time.
conFile.write("CONTINGENCY 'M%dB%d'\n" %( int(machid), busnum))
conFile.write('REMOVE MACHINE %s FROM BUS %d \n' %(machid, busnum))
conFile.write('END \n\n')
conFile.write('END')
conFile.close()

psspy.dfax([1,1], subSysLabel , subSysLabel , conFileName1 + '.con',
           conFileName1)

# System adjustment: Units Redispatch while Imposing Contingency
psspy.gendsp([2,0],[subSysLabel + '_DSP','M%dB%d'
                    %( int(machid), busnum)], conFileName1 + '.dfx',"","")

# The next piece of code is to avoid large islands in the solution
ierr,buses = psspy.tree(1,0)

while buses > 0:
    ierr,buses = psspy.tree(2,1)

# After redispatching, power flow Solution is solved a few times.
# Further System adjustments: enable stepping tap, phase shift,
# dc tap, and switched shunt adjustments.
psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
psspy.fdns([1,2,1,1,1,0,99,0])
psspy.fdns([1,2,1,1,1,0,99,0])
psspy.fdns([1,2,1,1,1,0,99,0])
psspy.fdns([1,2,1,1,1,0,99,0])
psspy.fdns([1,2,1,1,1,0,99,0])

# The new solved case is saved. This will be the Initial Condition
psspy.save(CASE1 + '_GENTRIP_' + 'MACH%d-BUS%d'
           %( int(machid), busnum) + '.sav')

# New working case to be used with the reamining contingecy events,
# E1 through E4.
CASE2 = CASE1 + '_GENTRIP_' + 'MACH%d-BUS%d' %( int(machid), busnum) + '.sav'

psspy.case(CASE2)
psspy.prompt_output(4,"",[0,0])
psspy.alert_output(4,"",[0,0])

ierr, bus, mismatch = psspy.maxmsm()
print bus

mva = psspy.sysmsm()
if mva > 0.7:
    continue

# Defining a subsystem with buses in study area
sid2 = sid
usekv2 = usekv
areas2 = areas
```

```
numarea2 = len(areas2)
subsysID2 = sid2
subSysLabel2 = subSysLabel


"""
For Loop Part to perform all P3_E1 contingencies.
"""
psspy.case(CASE2)
basekv2=[1,999]
psspy.bsys(sid2, usekv2, basekv2, numarea2, areas2)

# Temporary CON file name with contingencies
conFileName2 = "P3_E1"

# Returns a list with Machine Identifiers for all in-service machines
# at in-service plants
ierr, (allMachID2,) = psspy.amachchar(subsysID2, 1, 'ID')
print "machID =", allMachID2

# Returns a list with Bus Numbers for all in-service machines
# at in-service plants
ierr, (busNums2,) = psspy.amachint(subsysID2, 1, 'NUMBER')
print "list of bus numbers ", busNums2

# Opens a file object for both writing and reading. Overwrites the
# existing file if the file exists. If the file does not exist, creates
# a new file for reading and writing.
conFile = open(conFileName2 + '.con', 'w+')

# .con file header
conFile.write("COM PSS(R)E 32\n")
conFile.write("COM\n")
conFile.write("COM LOSS OF SINGLE GENERATOR IN SUBSYSTEM '%s' \n\n"
              % subSysLabel2)

# Contingencies for removing each specified machine
for machid2, busnum2 in zip(allMachID2, busNums2):
    conFile.write("CONTINGENCY 'M%dB%d'\n" %( int(machid2), busnum2))
    conFile.write('REMOVE MACHINE %s FROM BUS %d \n' %(machid2, busnum2))
    conFile.write('END \n\n')

conFile.write('END')
conFile.close()

# Run Multi-level AC Contingency Solution
psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
psspy.dfax([1,1], subSysLabel2, subSysLabel2, conFileName2 + '.con',
           conFileName2)

psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                   [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                   [subSysLabel + '_DSP',subSysLabel + '_HV',
                    subSysLabel + '_HV',subSysLabel + '_HV',
                    subSysLabel + '_HV',
                    subSysLabel + '_BULK',subSysLabel + '_BULK'],
                  conFileName2 + '.dfx',
                  conFileName2 + '_GENTRIP_' + 'MACH%d-BUS%d'
                  %( int(machid), busnum),
                  "","",subSysLabel + '.trp')


# Generate final excel file with results
import pssexcel

ACCFile=sys.path[0]

pssexcel.accc(conFileName2 + '_GENTRIP_' + 'MACH%d-BUS%d' %( int(machid), busnum) + ".acc",
              ['e','b','v'],colabel='',stype='contingency',
              busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
```

```
                xlsfile = conFileName2 + '_GENTRIP_' + 'MACH%d-BUS%d' %( int(machid), busnum) +
'.xls',
                sheet='',overwritesheet=True,show=False)

    # Generate final text file with results
    psspy.report_output(islct=2, filarg="./" + conFileName2 + '_GENTRIP_' + 'MACH%d-BUS%d' %(
int(machid), busnum) + '.txt',
                        options=[0])
    psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                                   [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                                   conFileName2 + '_GENTRIP_' + 'MACH%d-BUS%d'
                                   %( int(machid), busnum))
    psspy.report_output(islct=2, options=[0])


    """
    For Loop Part to perform all P3_E2 contingencies.
    """
    psspy.case(CASE2)
    basekv2=[100,999]
    psspy.bsys(sid2, usekv2, basekv2, numarea2, areas2)

    # Temporary CON file name with contingencies
    conFileName2 = "P3_E2"

    # Returns a list with Circuit Identifiers for only in-service
    # non-transformer branches
    ierr, (allBrnID2,) = psspy.abrnchar(subsysID2, 1, 3, 1, 1, 'ID')
    print "BrnID =", allBrnID2

    # Returns a list with From Bus Numbers for only in-service
    # non-transformer branches
    ierr, (FrombusNums2,) = psspy.abrnint(subsysID2, 1, 3, 1, 1, 'FROMNUMBER')
    print "list of FROM bus numbers Brn ", FrombusNums2

    # Returns a list with To Bus Numbers for only in-service
    # non-transformer branches
    ierr, (TobusNums2,) = psspy.abrnint(subsysID2, 1, 3, 1, 1, 'TONUMBER')
    print "list of TO bus numbers Brn ", TobusNums2

    # Opens a file object for both writing and reading. Overwrites the
    # existing file if the file exists. If the file does not exist, creates
    # a new file for reading and writing.
    conFile = open(conFileName2 + '.con', 'w+')

    # .con file header
    conFile.write("COM PSS(R)E 32\n")
    conFile.write("COM\n")
    conFile.write("COM LOSS OF SINGLE TRANSMISSION LINE IN SUBSYSTEM '%s' \n\n"
                  % subSysLabel2)

    # Contingencies for removing each specified Transmission Circuit
    for Brnid2, frombusnum2, tobusnum2 in zip(allBrnID2, FrombusNums2, TobusNums2):
        conFile.write("CONTINGENCY '%d_%d_%d'\n"
                      %(frombusnum2, tobusnum2, int(Brnid2)))
        conFile.write('DISCONNECT BRANCH FROM BUS %d TO BUS %d CIRCUIT %d\n'
                      %(frombusnum2, tobusnum2, int(Brnid2)))
        conFile.write('END \n\n')

    conFile.write('END')
    conFile.close()

    # Run Multi-level AC Contingency Solution
    psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
    psspy.dfax([1,1], subSysLabel2, subSysLabel2, conFileName2 + '.con',
               conFileName2)

    psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                       [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                       [subSysLabel + '_DSP',subSysLabel + '_HV',
                        subSysLabel + '_HV',subSysLabel + '_HV',
```

```
                      subSysLabel + '_HV',
                      subSysLabel + '_BULK',subSysLabel + '_BULK'],
                    conFileName2 + '.dfx',
                    conFileName2 + '_GENTRIP_' + 'MACH%d-BUS%d'
                    %( int(machid), busnum),
                    "","",subSysLabel + '.trp')


    # Generate final excel file with results
    pssexcel.accc(conFileName2 + '_GENTRIP_' + 'MACH%d-BUS%d' %( int(machid), busnum) + ".acc",
                  ['e','b','v'],colabel='',stype='contingency',
                  busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
                  xlsfile = conFileName2 + '_GENTRIP_' + 'MACH%d-BUS%d' %( int(machid), busnum) +
'.xls',
                  sheet='',overwritesheet=True,show=False)

    # Generate final text file with results
    psspy.report_output(islct=2, filarg="./" + conFileName2 + '_GENTRIP_' + 'MACH%d-BUS%d' %(
int(machid), busnum) + '.txt',
                        options=[0])
    psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                                   [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                                   conFileName2 + '_GENTRIP_' + 'MACH%d-BUS%d'
                                   %( int(machid), busnum))
    psspy.report_output(islct=2, options=[0])


    """
    For Loop Part to perform all P3_E3 contingencies.
    """
    psspy.case(CASE2)
    basekv2=[100,999]
    psspy.bsys(sid2, usekv2, basekv2, numarea2, areas2)

    # Temporary CON file name with contingencies
    conFileName2 = "P3_E3"

    # Returns a list with Circuit Identifiers for only in-service
    # two-winding transformers branches
    ierr, (allTrnID2,) = psspy.atrnchar(subsysID2, 1, 3, 1, 1, 'ID')
    print "TrnID =", allTrnID2

    # Returns a list with From Bus Numbers for only in-service
    # two-winding transformers branches
    ierr, (FrombusNums2,) = psspy.atrnint(subsysID2, 1, 3, 1, 1, 'FROMNUMBER')
    print "list of FROM bus numbers Trn ", FrombusNums2

    # Returns a list with To Bus Numbers for only in-service
    # two-winding transformers branches
    ierr, (TobusNums2,) = psspy.atrnint(subsysID2, 1, 3, 1, 1, 'TONUMBER')
    print "list of TO bus numbers Trn ", TobusNums2

    # Opens a file object for both writing and reading. Overwrites the
    # existing file if the file exists. If the file does not exist, creates
    # a new file for reading and writing.
    conFile = open(conFileName2 + '.con', 'w+')

    # .con file header
    conFile.write("COM PSS(R)E 32\n")
    conFile.write("COM\n")
    conFile.write("COM LOSS OF SINGLE TRANSFORMER IN SUBSYSTEM '%s' \n\n"
                  % subSysLabel2)

    # Contingencies for removing each specified Transformer
    for Trnid2, frombusnum2, tobusnum2 in zip(allTrnID2, FrombusNums2, TobusNums2):
        conFile.write("CONTINGENCY '%d_%d_%d'\n"
                      %(frombusnum2, tobusnum2, int(Trnid2)))
        conFile.write('DISCONNECT BRANCH FROM BUS %d TO BUS %d CIRCUIT %d\n'
                      %(frombusnum2, tobusnum2, int(Trnid2)))
        conFile.write('END \n\n')
```

```
    conFile.write('END')
    conFile.close()

    # Run Multi-level AC Contingency Solution
    psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
    psspy.dfax([1,1], subSysLabel2, subSysLabel2, conFileName2 + '.con',
               conFileName2)

    psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                        [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                        [subSysLabel + '_DSP',subSysLabel + '_HV',
                         subSysLabel + '_HV',subSysLabel + '_HV',
                         subSysLabel + '_HV',
                         subSysLabel + '_BULK',subSysLabel + '_BULK'],
                       conFileName2 + '.dfx',
                       conFileName2 + '_GENTRIP_' + 'MACH%d-BUS%d'
                       %( int(machid), busnum),
                       "","",subSysLabel + '.trp')


    # Generate final excel file with results
    pssexcel.accc(conFileName2 + '_GENTRIP_' + 'MACH%d-BUS%d' %( int(machid), busnum) + ".acc",
                  ['e','b','v'],colabel='',stype='contingency',
                  busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
                  xlsfile = conFileName2 + '_GENTRIP_' + 'MACH%d-BUS%d' %( int(machid), busnum) +
'.xls',
                  sheet='',overwritesheet=True,show=False)

    # Generate final text file with results
    psspy.report_output(islct=2, filarg="./" + conFileName2 + '_GENTRIP_' + 'MACH%d-BUS%d' %(
int(machid), busnum) + '.txt',
                        options=[0])
    psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                                   [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                                   conFileName2 + '_GENTRIP_' + 'MACH%d-BUS%d'
                                   %( int(machid), busnum))
    psspy.report_output(islct=2, options=[0])


    """
    For Loop Part to perform all P3_E4 contingencies.
    """
    psspy.case(CASE2)
    basekv2=[100,999]
    psspy.bsys(sid2, usekv2, basekv2, numarea2, areas2)

    # Temporary CON file name with contingencies
    conFileName2 = "P3_E4"

    ### Part for FIXED Shunts Starts ###
    # Returns a list with Shunt Identifiers for only in-service
    # fixed shunts at in-service buses
    ierr, (allFxShuntID2,) = psspy.afxshuntchar(subsysID2, 1, 'ID')
    print "FxShuntID =", allFxShuntID2

    # Returns a list with Bus Numbers for only in-service
    # fixed shunts at in-service buses
    ierr, (busNums2,) = psspy.afxshuntint(subsysID2, 1, 'NUMBER')
    print "list of bus numbers Fixed Shunts ", busNums2

    ### Part for SWITCHED Shunts Starts ###
    # Returns a list with Bus Numbers for only in-service
    # switched shunts at in-service buses
    ierr, (busNumsSw2,) = psspy.aswshint(sid2, 1, 'NUMBER')
    print "list of bus numbers Switched Shunts ", busNumsSw2

    # Opens a file object for both writing and reading. Overwrites the
    # existing file if the file exists. If the file does not exist, creates
    # a new file for reading and writing.
    conFile = open(conFileName2 + '.con', 'w+')
```

```
    # .con file header
    conFile.write("COM PSS(R)E 32\n")
    conFile.write("COM\n")
    conFile.write("COM LOSS OF SINGLE SHUNT DEVICE IN SUBSYSTEM '%s' \n\n"
                  % subSysLabel2)

    # Contingencies for removing each specified fixed shunt
    for FxShuntid2, busnum2 in zip(allFxShuntID2, busNums2):
        conFile.write("CONTINGENCY 'F%dB%d'\n"
                      %( int(FxShuntid2), busnum2))
        conFile.write('REMOVE SHUNT %s FROM BUS %d\n' %(FxShuntid2, busnum2))
        conFile.write('END \n\n')

    # Contingencies for removing each specified switched shunt
    for busnum2 in busNumsSw2:
        conFile.write("CONTINGENCY 'SwB%d'\n" %busnum2)
        conFile.write('REMOVE SWSHUNT FROM BUS %d\n' %busnum2)
        conFile.write('END\n\n')

    conFile.write('END')
    conFile.close()

    # Run Multi-level AC Contingency Solution
    psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
    psspy.dfax([1,1], subSysLabel2, subSysLabel2, conFileName2 + '.con',
               conFileName2)

    psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                        [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                        [subSysLabel + '_DSP',subSysLabel + '_HV',
                         subSysLabel + '_HV',subSysLabel + '_HV',
                         subSysLabel + '_HV',
                         subSysLabel + '_BULK',subSysLabel + '_BULK'],
                       conFileName2 + '.dfx',
                       conFileName2 + '_GENTRIP_' + 'MACH%d-BUS%d'
                       %( int(machid), busnum),
                       "","",subSysLabel + '.trp')


    # Generate final excel file with results
    pssexcel.accc(conFileName2 + '_GENTRIP_' + 'MACH%d-BUS%d' %( int(machid), busnum) + ".acc",
                  ['e','b','v'],colabel='',stype='contingency',
                  busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
                  xlsfile = conFileName2 + '_GENTRIP_' + 'MACH%d-BUS%d' %( int(machid), busnum) +
'.xls',
                  sheet='',overwritesheet=True,show=False)

    # Generate final text file with results
    psspy.report_output(islct=2, filarg="./" + conFileName2 + '_GENTRIP_' + 'MACH%d-BUS%d' %(
int(machid), busnum) + '.txt',
                        options=[0])
    psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                                   [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                                   conFileName2 + '_GENTRIP_' + 'MACH%d-BUS%d'
                                   %( int(machid), busnum))
    psspy.report_output(islct=2, options=[0])


    """
    For Loop Part to perform all P3_E5 contingencies.
    """
    psspy.case(CASE2)
    basekv2=[1,999]
    psspy.bsys(sid2, usekv2, basekv2, numarea2, areas2)

    # CON file name with contingencies
    conFileName2 = "P3_E5"

    # Returns a list with DC Line names for only in-service
    # two-terminal dc lines
    ierr, (allDCBrnNames2,) = psspy.a2trmdcchar(subsysID2, 3, 1, 'DCNAME')
```

```
    print "DCBrnID =", allDCBrnNames2

    # Returns a list with From or Rectifier Bus Numbers for only in-service
    # two-terminal dc lines
    ierr, (FrombusNums2,) = psspy.a2trmdcint(subsysID2, 3, 1, 'FROMNUMBER')
    print "list of FROM bus numbers DCBrn ", FrombusNums2

    # Returns a list with To or Inverter Bus Numbers for only in-service
    # two-terminal dc lines
    ierr, (TobusNums2,) = psspy.a2trmdcint(subsysID2, 3, 1, 'TONUMBER')
    print "list of TO bus numbers DCBrn ", TobusNums2

    # Opens a file object for both writing and reading. Overwrites the
    # existing file if the file exists. If the file does not exist, creates
    # a new file for reading and writing.
    conFile = open(conFileName2 + '.con', 'w+')

    # .con file header
    conFile.write("COM PSS(R)E 32\n")
    conFile.write("COM\n")
    conFile.write("COM LOSS OF SINGLE POLE OF A DC LINE IN SUBSYSTEM '%s' \n\n"
                  % subSysLabel)

    # Contingencies for removing each specified Pole of a DC Line
    for DCBrnname2, frombusnum2, tobusnum2 in zip(allDCBrnNames2, FrombusNums2, TobusNums2):
        conFile.write("CONTINGENCY '%d_%d_%s'\n"
                      %(frombusnum2, tobusnum2, DCBrnname2))
        conFile.write('BLOCK TWOTERMDC %s\n' %DCBrnname2)
        conFile.write('END \n\n')

    conFile.write('END')
    conFile.close()

    # Run Multi-level AC Contingency Solution
    psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
    psspy.dfax([1,1], subSysLabel2, subSysLabel2, conFileName2 + '.con',
               conFileName2)

    psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                      [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                      [subSysLabel + '_DSP',subSysLabel + '_HV',
                       subSysLabel + '_HV',subSysLabel + '_HV',
                       subSysLabel + '_HV',
                       subSysLabel + '_BULK',subSysLabel + '_BULK'],
                     conFileName2 + '.dfx',
                     conFileName2 + '_GENTRIP_' + 'MACH%d-BUS%d'
                     %( int(machid), busnum),
                     "","",subSysLabel + '.trp')


    # Generate final excel file with results
    pssexcel.accc(conFileName2 + '_GENTRIP_' + 'MACH%d-BUS%d' %( int(machid), busnum) + ".acc",
                  ['e','b','v'],colabel='',stype='contingency',
                  busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
                  xlsfile = conFileName2 + '_GENTRIP_' + 'MACH%d-BUS%d' %( int(machid), busnum) +
'.xls',
                  sheet='',overwritesheet=True,show=False)

    # Generate final text file with results
    psspy.report_output(islct=2, filarg="./" + conFileName2 + '_GENTRIP_' + 'MACH%d-BUS%d' %(
int(machid), busnum) + '.txt',
                        options=[0])
    psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                                   [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                                   conFileName2 + '_GENTRIP_' + 'MACH%d-BUS%d'
                                   %( int(machid), busnum))
    psspy.report_output(islct=2, options=[0])


print('\nP3 Category Complete!')
print('NOTE: It is highly recommended to create a new folder where')
```

```
print('      to move all the files created by this program.\n')
```

A.4 Category P4

```
"""
THIS PROGRAM PERFORMS P4 CONTINGENCIES PER TPL-001-4
EVENTS 1 THROUGH 6
"""

import os,sys

PSSE_LOCATION = r"C:\Program Files\PTI\PSSE32\PSSBIN"
sys.path.append(PSSE_LOCATION)
os.environ['PATH'] = os.environ['PATH'] + ';' +  PSSE_LOCATION

import psspy
import redirect
redirect.psse2py()

from sys import argv
if len(sys.argv) > 1:
    script, CASE1, areas, subSysLabel = argv
else:
    CASE1 = raw_input("Base case name, without *.sav extension: ")
    areas = raw_input("Area number where contingencies will take place: ")
    subSysLabel = raw_input("Area name where contingencies will take place: ")

areas = [int(areas)]


#---------------------------------
# PSS/E Saved case

CASE = CASE1 + '.sav'

psspy.psseinit(50000)
psspy.prompt_output(4,"",[0,0])
psspy.alert_output(4,"",[0,0])

# Defining a subsystem with buses in study area
sid = 0
usekv = 1
numarea = len(areas)
subsysID = sid


"""
Program to perform all P4_E1 contingencies.
"""
psspy.case(CASE)
basekv=[1,999]
psspy.bsys(sid, usekv, basekv, numarea, areas)

# CON file name with contingencies
conFileName1 = "P4_E1"
conFileName = conFileName1 + '.con'

# Returns a list with Machine Identifiers for all in-service machines
# at in-service plants
ierr, (allMachID,) = psspy.amachchar(subsysID, 1, 'ID')
print "machID =", allMachID

# Returns a list with Bus Numbers for all in-service machines
# at in-service plants
ierr, (busNums,) = psspy.amachint(subsysID, 1, 'NUMBER')
print "list of bus numbers ", busNums
```

```
# Opens a file object for both writing and reading. Overwrites the
# existing file if the file exists. If the file does not exist, creates
# a new file for reading and writing.
conFile = open(conFileName, 'w+')

# .con file header
conFile.write("COM PSS(R)E 32\n")
conFile.write("COM\n")
conFile.write("COM LOSS OF MULTIPLE ELEMENTS CAUSED BY A STUCK BREAKER\n")
conFile.write("COM ATTEMPTING TO CLEAR A FAULT ON A GENERATOR IN SUBSYSTEM '%s' \n\n"
              % subSysLabel)

# Contingencies for removing each specified machine
for machid, busnum in zip(allMachID, busNums):
        conFile.write("CONTINGENCY 'M%dB%d'\n" %( int(machid), busnum))
        conFile.write('DISCONNECT BUS %d \n' % busnum )
        conFile.write('END \n\n')

conFile.write('END')
conFile.close()

# Run Multi-level AC Contingency Solution
psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
psspy.dfax([1,1],subSysLabel,subSysLabel,conFileName,
           conFileName1)

psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                       [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                       [subSysLabel + '_DSP',subSysLabel + '_HV',
                        subSysLabel + '_HV',subSysLabel + '_HV',
                        subSysLabel + '_HV',
                        subSysLabel + '_BULK',subSysLabel + '_BULK'],
                      conFileName1 + '.dfx',conFileName1,"","",subSysLabel + '.trp')

# Generate final excel file with results
import pssexcel

ACCFile=sys.path[0]

pssexcel.accc(conFileName1 + '.acc',['e','b','v'],colabel='',stype='contingency',
              busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
              xlsfile='ACCCResults_' + conFileName1 + '.xls',sheet='',
              overwritesheet=True,show=False)#,ratecon='a',baseflowvio=False,
              #basevoltvio=False,flowlimit=90.0,flowchange=5.0,voltchange=0.0)

# Generate final text file with results
psspy.report_output(islct=2, filarg="./" + 'ACCCResults_' + conFileName1 + '.txt',
                    options=[0])
psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                               [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                               conFileName1)
psspy.report_output(islct=2, options=[0])


"""
Program to perform all P4_E2 contingencies.
"""
psspy.case(CASE)
basekv=[100,999]
psspy.bsys(sid, usekv, basekv, numarea, areas)

# CON file name with contingencies
conFileName1 = "P4_E2"
conFileName = conFileName1 + '.con'

# Returns a list with Circuit Identifiers for only in-service
# non-transformer branches
ierr, (allBrnID,) = psspy.abrnchar(subsysID, 1, 3, 1, 2, 'ID')
print "BrnID =", allBrnID
```

```
# Returns a list with From Bus Numbers for only in-service
# non-transformer branches
ierr, (FrombusNums,) = psspy.abrnint(subsysID, 1, 3, 1, 2, 'FROMNUMBER')
print "list of FROM bus numbers Brn ", FrombusNums

# Returns a list with To Bus Numbers for only in-service
# non-transformer branches
ierr, (TobusNums,) = psspy.abrnint(subsysID, 1, 3, 1, 2, 'TONUMBER')
print "list of TO bus numbers Brn ", TobusNums

# Opens a file object for both writing and reading. Overwrites the
# existing file if the file exists. If the file does not exist, creates
# a new file for reading and writing.
conFile = open(conFileName, 'w+')

# .con file header
conFile.write("COM PSS(R)E 32\n")
conFile.write("COM\n")
conFile.write("COM LOSS OF MULTIPLE ELEMENTS CAUSED BY A STUCK BREAKER\n")
conFile.write("COM ATTEMPTING TO CLEAR A FAULT ON A TRANSMISSION CIRCUIT IN SUBSYSTEM '%s' \n\n"
              % subSysLabel)

# Contingencies for removing each specified Transmission Circuit
for Brnid, frombusnum, tobusnum in zip(allBrnID, FrombusNums, TobusNums):
        conFile.write("CONTINGENCY '%d_%d_%d'\n"
                      %(frombusnum, tobusnum, int(Brnid)))
        conFile.write('DISCONNECT BUS %d \n' % tobusnum)
        conFile.write('END \n\n')

conFile.write('END')
conFile.close()

# Run Multi-level AC Contingency Solution
psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
psspy.dfax([1,1],subSysLabel,subSysLabel,conFileName,
           conFileName1)

psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                       [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                       [subSysLabel + '_DSP',subSysLabel + '_HV',
                        subSysLabel + '_HV',subSysLabel + '_HV',
                        subSysLabel + '_HV',
                        subSysLabel + '_BULK',subSysLabel + '_BULK'],
                      conFileName1 + '.dfx',conFileName1,"","",subSysLabel + '.trp')

# Generate final excel file with results
pssexcel.accc(conFileName1 + '.acc',['e','b','v'],colabel='',stype='contingency',
              busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
              xlsfile='ACCCResults_' + conFileName1 + '.xls',sheet='',
              overwritesheet=True,show=False)

# Generate final text file with results
psspy.report_output(islct=2, filarg="./" + 'ACCCResults_' + conFileName1 + '.txt',
                    options=[0])
psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                               [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                               conFileName1)
psspy.report_output(islct=2, options=[0])


"""
Program to perform all P4_E3 contingencies.
"""
psspy.case(CASE)
basekv=[100,999]
psspy.bsys(sid, usekv, basekv, numarea, areas)

# CON file name with contingencies
conFileName1 = "P4_E3"
conFileName = conFileName1 + '.con'
```

```
# Returns a list with Circuit Identifiers for only in-service
# two-winding transformers branches
ierr, (allTrnID,) = psspy.atrnchar(subsysID, 1, 3, 1, 2, 'ID')
print "TrnID =", allTrnID

# Returns a list with From Bus Numbers for only in-service
# two-winding transformers branches
ierr, (FrombusNums,) = psspy.atrnint(subsysID, 1, 3, 1, 2, 'FROMNUMBER')
print "list of FROM bus numbers Trn ", FrombusNums

# Returns a list with To Bus Numbers for only in-service
# two-winding transformers branches
ierr, (TobusNums,) = psspy.atrnint(subsysID, 1, 3, 1, 2, 'TONUMBER')
print "list of TO bus numbers Trn ", TobusNums

# Opens a file object for both writing and reading. Overwrites the
# existing file if the file exists. If the file does not exist, creates
# a new file for reading and writing.
conFile = open(conFileName, 'w+')

# .con file header
conFile.write("COM PSS(R)E 32\n")
conFile.write("COM\n")
conFile.write("COM LOSS OF MULTIPLE ELEMENTS CAUSED BY A STUCK BREAKER\n")
conFile.write("COM ATTEMPTING TO CLEAR A FAULT ON A TRANSFORMER IN SUBSYSTEM '%s' \n\n"
              % subSysLabel)

# Contingencies for removing each specified Transformer
for Trnid, frombusnum, tobusnum in zip(allTrnID, FrombusNums, TobusNums):
        conFile.write("CONTINGENCY '%d_%d_%d'\n"
                      %(frombusnum, tobusnum, int(Trnid)))
        conFile.write('DISCONNECT BUS %d \n' % tobusnum)
        conFile.write('END \n\n')

conFile.write('END')
conFile.close()

# Run Multi-level AC Contingency Solution
psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
psspy.dfax([1,1],subSysLabel,subSysLabel,conFileName,
           conFileName1)

psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                       [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                       [subSysLabel + '_DSP',subSysLabel + '_HV',
                        subSysLabel + '_HV',subSysLabel + '_HV',
                        subSysLabel + '_HV',
                        subSysLabel + '_BULK',subSysLabel + '_BULK'],
                      conFileName1 + '.dfx',conFileName1,"","",subSysLabel + '.trp')

# Generate final excel file with results
pssexcel.accc(conFileName1 + '.acc',['e','b','v'],colabel='',stype='contingency',
              busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
              xlsfile='ACCCResults_' + conFileName1 + '.xls',sheet='',
              overwritesheet=True,show=False)

# Generate final text file with results
psspy.report_output(islct=2, filarg="./" + 'ACCCResults_' + conFileName1 + '.txt',
                    options=[0])
psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                               [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                               conFileName1)
psspy.report_output(islct=2, options=[0])


"""
Program to perform all P4_E4 contingencies.
"""
psspy.case(CASE)
basekv=[100,999]
psspy.bsys(sid, usekv, basekv, numarea, areas)
```

```
# CON file name with contingencies
conFileName1 = "P4_E4"
conFileName = conFileName1 + '.con'

### Part for FIXED Shunts Starts ###
# Returns a list with Shunt Identifiers for only in-service
# fixed shunts at in-service buses
ierr, (allFxShuntID,) = psspy.afxshuntchar(subsysID, 1, 'ID')
print "FxShuntID =", allFxShuntID

# Returns a list with Bus Numbers for only in-service
# fixed shunts at in-service buses
ierr, (busNums,) = psspy.afxshuntint(subsysID, 1, 'NUMBER')
print "list of bus numbers Fixed Shunts ", busNums

### Part for SWITCHED Shunts Starts ###
# Returns a list with Bus Numbers for only in-service
# switched shunts at in-service buses
ierr, (busNumsSw,) = psspy.aswshint(sid, 1, 'NUMBER')
print "list of bus numbers Switched Shunts ", busNumsSw

# Opens a file object for both writing and reading. Overwrites the
# existing file if the file exists. If the file does not exist, creates
# a new file for reading and writing.
conFile = open(conFileName, 'w+')

# .con file header
conFile.write("COM PSS(R)E 32\n")
conFile.write("COM\n")
conFile.write("COM LOSS OF MULTIPLE ELEMENTS CAUSED BY A STUCK BREAKER\n")
conFile.write("COM ATTEMPTING TO CLEAR A FAULT ON A SHUNT DEVICE IN SUBSYSTEM '%s' \n\n"
              % subSysLabel)

# Contingencies for removing each specified fixed shunt
for FxShuntid, busnum in zip(allFxShuntID, busNums):
        conFile.write("CONTINGENCY 'F%dB%d'\n" %( int(FxShuntid), busnum))
        conFile.write('DISCONNECT BUS %d \n' % busnum)
        conFile.write('END\n\n')

# Contingencies for removing each specified switched shunt
for busnum in busNumsSw:
        conFile.write("CONTINGENCY 'SwB%d'\n" %busnum)
        conFile.write('DISCONNECT BUS %d \n' % busnum)
        conFile.write('END\n\n')

conFile.write('END')
conFile.close()

# Run Multi-level AC Contingency Solution
psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
psspy.dfax([1,1],subSysLabel,subSysLabel,conFileName,
           conFileName1)

psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                       [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                       [subSysLabel + '_DSP',subSysLabel + '_HV',
                        subSysLabel + '_HV',subSysLabel + '_HV',
                        subSysLabel + '_HV',
                        subSysLabel + '_BULK',subSysLabel + '_BULK'],
                      conFileName1 + '.dfx',conFileName1,"","",subSysLabel + '.trp')

# Generate final excel file with results
pssexcel.accc(conFileName1 + '.acc',['e','b','v'],colabel='',stype='contingency',
              busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
              xlsfile='ACCCResults_' + conFileName1 + '.xls',sheet='',
              overwritesheet=True,show=False)

# Generate final text file with results
psspy.report_output(islct=2, filarg="./" + 'ACCCResults_' + conFileName1 + '.txt',
                    options=[0])
```

```
psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                               [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                               conFileName1)
psspy.report_output(islct=2, options=[0])


"""
Program to perform all P4_E5 contingencies.
"""
psspy.case(CASE)
basekv=[100,999]
psspy.bsys(sid, usekv, basekv, numarea, areas)

# CON file name with contingencies
conFileName1 = "P4_E5"
conFileName = conFileName1 + '.con'

# Returns a list with Circuit Identifiers for only in-service
# non-transformer branches
ierr, (allBrnID,) = psspy.abrnchar(subsysID, 1, 3, 1, 1, 'ID')
print "BrnID =", allBrnID

# Returns a list with From Bus Numbers for only in-service
# non-transformer branches
ierr, (FrombusNums,) = psspy.abrnint(subsysID, 1, 3, 1, 1, 'FROMNUMBER')
print "list of FROM bus numbers Brn ", FrombusNums

# Returns a list with To Bus Numbers for only in-service
# non-transformer branches
ierr, (TobusNums,) = psspy.abrnint(subsysID, 1, 3, 1, 1, 'TONUMBER')
print "list of TO bus numbers Brn ", TobusNums

# Returns a list with Branch impedances for only in-service
# non-transformer branches
ierr, (RXarray,) = psspy.abrncplx(subsysID, 1, 3, 1, 1, 'RX')
print "list of Branch impedance in pu ", RXarray

# The next section is intended to identify branches
# with impedances lower than the Zero impedance line threshold
global IsSection
IsSection = []

for issection in RXarray:
    magnitude = abs(issection)

    if magnitude <= 0.000100:
        IsSection.append(1)
    else:
        IsSection.append(0)

print "list of Identified Sections ", IsSection

# Opens a file object for both writing and reading. Overwrites the
# existing file if the file exists. If the file does not exist, creates
# a new file for reading and writing.
conFile = open(conFileName, 'w+')

# .con file header
conFile.write("COM PSS(R)E 32\n")
conFile.write("COM\n")
conFile.write("COM LOSS OF MULTIPLE ELEMENTS CAUSED BY A STUCK NON-BUS-TIE BREAKER\n")
conFile.write("COM ATTEMPTING TO CLEAR A FAULT ON A BUS SECTION '%s' \n\n"
              % subSysLabel)

# Contingencies for removing multiple elements caused by a stuck Breaker
for Brnid, frombusnum, tobusnum, issection in zip(allBrnID, FrombusNums, TobusNums, IsSection):
    if  issection == 1:
        conFile.write("COM DISCONNECT MULTIPLE ELEMENTS DUE TO A SECTION %d FAULT IN BUS %d-BUS
%d\n"
                      % (frombusnum, frombusnum, tobusnum))
        conFile.write("CONTINGENCY '%d_%d_%d'\n" % (frombusnum, frombusnum, tobusnum))
```

```
        conFile.write('DISCONNECT BUS %d \n' % frombusnum)
        conFile.write('END \n\n')

        conFile.write("COM DISCONNECT MULTIPLE ELEMENTS DUE TO A SECTION %d FAULT IN BUS %d-BUS
%d\n"
                      % (tobusnum, frombusnum, tobusnum))
        conFile.write("CONTINGENCY '%d_%d_%d'\n" % (tobusnum, frombusnum, tobusnum))
        conFile.write('DISCONNECT BUS %d \n' % tobusnum)
        conFile.write('END \n\n')

conFile.write('END')
conFile.close()

# Run Multi-level AC Contingency Solution
psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
psspy.dfax([1,1],subSysLabel,subSysLabel,conFileName,
           conFileName1)

psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                       [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                       [subSysLabel + '_DSP',subSysLabel + '_HV',
                        subSysLabel + '_HV',subSysLabel + '_HV',
                        subSysLabel + '_HV',
                        subSysLabel + '_BULK',subSysLabel + '_BULK'],
                      conFileName1 + '.dfx',conFileName1,"","",subSysLabel + '.trp')

# Generate final excel file with results
pssexcel.accc(conFileName1 + '.acc',['e','b','v'],colabel='',stype='contingency',
              busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
              xlsfile='ACCCResults_' + conFileName1 + '.xls',sheet='',
              overwritesheet=True,show=False)

# Generate final text file with results
psspy.report_output(islct=2, filarg="./" + 'ACCCResults_' + conFileName1 + '.txt',
                    options=[0])
psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                               [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                               conFileName1)
psspy.report_output(islct=2, options=[0])


"""
Program to perform all P4_E6 contingencies.
"""
psspy.case(CASE)
basekv=[100,999]
psspy.bsys(sid, usekv, basekv, numarea, areas)

# CON file name with contingencies
conFileName1 = "P4_E6"
conFileName = conFileName1 + '.con'

# Returns a list with Circuit Identifiers for only in-service
# non-transformer branches
ierr, (allBrnID,) = psspy.abrnchar(subsysID, 1, 3, 1, 1, 'ID')
print "BrnID =", allBrnID

# Returns a list with From Bus Numbers for only in-service
# non-transformer branches
ierr, (FrombusNums,) = psspy.abrnint(subsysID, 1, 3, 1, 1, 'FROMNUMBER')
print "list of FROM bus numbers Brn ", FrombusNums

# Returns a list with To Bus Numbers for only in-service
# non-transformer branches
ierr, (TobusNums,) = psspy.abrnint(subsysID, 1, 3, 1, 1, 'TONUMBER')
print "list of TO bus numbers Brn ", TobusNums

# Returns a list with Branch impedances for only in-service
# non-transformer branches
ierr, (RXarray,) = psspy.abrncplx(subsysID, 1, 3, 1, 1, 'RX')
print "list of Branch impedance in pu ", RXarray
```

```
# The next section is intended to identify branches
# with impedances lower than the Zero impedance line threshold
global IsSection2
IsSection2 = []

for issection in RXarray:
    magnitude = abs(issection)

    if magnitude <= 0.000100:
        IsSection2.append(1)
    else:
        IsSection2.append(0)

print "list of Identified Sections ", IsSection2

# Opens a file object for both writing and reading. Overwrites the
# existing file if the file exists. If the file does not exist, creates
# a new file for reading and writing.
conFile = open(conFileName, 'w+')

# .con file header
conFile.write("COM PSS(R)E 32\n")
conFile.write("COM\n")
conFile.write("COM LOSS OF MULTIPLE ELEMENTS CAUSED BY A BUS-TIE STUCK BREAKER\n")
conFile.write("COM ATTEMPTING TO CLEAR A FAULT ON THE ASSOCIATED BUS '%s' \n\n"
              % subSysLabel)

# Contingencies for removing multiple elements caused by a stuck Bus-tie Breaker
for Brnid, frombusnum, tobusnum, issection in zip(allBrnID, FrombusNums, TobusNums, IsSection2):
    if  issection == 1:
        conFile.write("COM DISCONNECT MULTIPLE ELEMENTS DUE TO A STUCK TIE BREAKER BETWEEN
SECTION BUS %d AND BUS %d\n"
                      %(frombusnum, tobusnum))
        conFile.write("CONTINGENCY '%d_%d_%d'\n"
                     %(frombusnum, tobusnum, int(Brnid)))
        conFile.write('DISCONNECT BUS %d \n' % frombusnum)
        conFile.write('DISCONNECT BUS %d \n' % tobusnum)
        conFile.write('END \n\n')

conFile.write('END')
conFile.close()

# Run Multi-level AC Contingency Solution
psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
psspy.dfax([1,1],subSysLabel,subSysLabel,conFileName,
           conFileName1)

psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                       [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                       [subSysLabel + '_DSP',subSysLabel + '_HV',
                        subSysLabel + '_HV',subSysLabel + '_HV',
                        subSysLabel + '_HV',
                        subSysLabel + '_BULK',subSysLabel + '_BULK'],
                      conFileName1 + '.dfx',conFileName1,"","",subSysLabel + '.trp')

# Generate final excel file with results
pssexcel.accc(conFileName1 + '.acc',['e','b','v'],colabel='',stype='contingency',
              busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
              xlsfile='ACCCResults_' + conFileName1 + '.xls',sheet='',
              overwritesheet=True,show=False)

# Generate final text file with results
psspy.report_output(islct=2, filarg="./" + 'ACCCResults_' + conFileName1 + '.txt',
                    options=[0])
psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                               [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                               conFileName1)
psspy.report_output(islct=2, options=[0])
```

```
print('\nP4 Category Complete!')
print('NOTE: It is highly recommended to create a new folder where')
print('      to move all the files created by this program.\n')
```

A.5 Category P6_I1

```
"""
THIS PROGRAM PERFORMS P6 CONTINGENCIES PER TPL-001-4
INITIAL CONDITION 1 WITH EVENTS 1 THROUGH 4
"""

"""
First Part is to calculate Initial Condition I1.
That is, loss of a Transmission Circuit followed by
System adjustments
"""

import os,sys

PSSE_LOCATION = r"C:\Program Files\PTI\PSSE32\PSSBIN"
sys.path.append(PSSE_LOCATION)
os.environ['PATH'] = os.environ['PATH'] + ';' +  PSSE_LOCATION

import psspy
import redirect
redirect.psse2py()

from sys import argv
if len(sys.argv) > 1:
    script, CASE1, areas, subSysLabel = argv
else:
    CASE1 = raw_input("Base case name, without *.sav extension: ")
    areas = raw_input("Area number where contingencies will take place: ")
    subSysLabel = raw_input("Area name where contingencies will take place: ")

areas = [int(areas)]


#---------------------------------
# PSS/E Saved case

CASE = CASE1 + '.sav'

psspy.psseinit(50000)
psspy.prompt_output(4,"",[0,0])
psspy.alert_output(4,"",[0,0])

# Defining a subsystem with buses in study area
sid = 0
usekv = 1
numarea = len(areas)
subsysID = sid

psspy.case(CASE)
basekv=[100,999]
psspy.bsys(sid, usekv, basekv, numarea, areas)

conFileName1 = "P6_P1_E2"

# Returns a list with Circuit Identifiers for only in-service
# non-transformer branches
ierr, (allBrnID,) = psspy.abrnchar(subsysID, 1, 3, 1, 1, 'ID')
print "BrnID =", allBrnID

# Returns a list with From Bus Numbers for only in-service
# non-transformer branches
```

```
ierr, (FrombusNums,) = psspy.abrnint(subsysID, 1, 3, 1, 1, 'FROMNUMBER')
print "list of FROM bus numbers Brn ", FrombusNums

# Returns a list with To Bus Numbers for only in-service
# non-transformer branches
ierr, (TobusNums,) = psspy.abrnint(subsysID, 1, 3, 1, 1, 'TONUMBER')
print "list of TO bus numbers Brn ", TobusNums

# Loop to lose each Transmission Circuit, perform system adjustments,
# and study the respective events
for Brnid, frombusnum, tobusnum in zip(allBrnID, FrombusNums, TobusNums):
    """
    For Loop Part to create a temporary contingency file for each Transmission
    Circuit lost in ORIGINAL case file
    """
    psspy.case(CASE)
    psspy.prompt_output(4,"",[0,0])
    psspy.alert_output(4,"",[0,0])

    # Opens a file object for both writing and reading. Overwrites the
    # existing file if the file exists. If the file does not exist, creates
    # a new file for reading and writing.
    conFile = open(conFileName1 + '.con', 'w+')

    # .con file header
    conFile.write("COM PSS(R)E 32\n")
    conFile.write("COM\n")
    conFile.write("COM LOSS OF SINGLE TRANSMISSION LINE IN SUBSYSTEM '%s' \n\n"
                  % subSysLabel)

    # Contingency for removing each specified Transmission Circuit, one at a time.
    conFile.write("CONTINGENCY '%d_%d_%d'\n"
                  %(frombusnum, tobusnum, int(Brnid)))
    conFile.write('DISCONNECT BRANCH FROM BUS %d TO BUS %d CIRCUIT %d\n'
                  %(frombusnum, tobusnum, int(Brnid)))
    conFile.write('END \n\n')
    conFile.write('END')
    conFile.close()

    psspy.dfax([1,1], subSysLabel , subSysLabel , conFileName1 + '.con',
               conFileName1)

    # System adjustment: Units Redispatch while Imposing Contingency
    psspy.gendsp([2,0],[subSysLabel + '_DSP','%d_%d_%d'
                        %(frombusnum, tobusnum, int(Brnid))],
                 conFileName1 + '.dfx',"","")

    # The next piece of code is to avoid large islands in the solution
    ierr,buses = psspy.tree(1,0)

    while buses > 0:
        ierr,buses = psspy.tree(2,1)

    # After redispatching, power flow Solution is solved a few times.
    # Further System adjustments: enable stepping tap, phase shift,
    # dc tap, and switched shunt adjustments.
    psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
    psspy.fdns([1,2,1,1,1,0,99,0])
    psspy.fdns([1,2,1,1,1,0,99,0])
    psspy.fdns([1,2,1,1,1,0,99,0])
    psspy.fdns([1,2,1,1,1,0,99,0])
    psspy.fdns([1,2,1,1,1,0,99,0])

    # The new solved case is saved. This will be the Initial Condition
    psspy.save(CASE1 + '_TRANCKTTRIP_' + 'L%d_%d_%d'
               %(frombusnum, tobusnum, int(Brnid)) + '.sav')

    # New working case to be used with the reamining contingecy events,
    # E1 through E3.
    CASE2 = CASE1 + '_TRANCKTTRIP_' + 'L%d_%d_%d' %(frombusnum, tobusnum, int(Brnid)) + '.sav'
```

```
psspy.case(CASE2)
psspy.prompt_output(4,"",[0,0])
psspy.alert_output(4,"",[0,0])

ierr, bus, mismatch = psspy.maxmsm()
print bus

mva = psspy.sysmsm()
if mva > 0.7:
    continue

# Defining a subsystem with buses in study area 1
sid2 = sid
usekv2 = usekv
areas2 = areas
numarea2 = len(areas2)
subsysID2 = sid2
subSysLabel2 = subSysLabel


"""
For Loop Part to perform all P6_I1_E1 contingencies.
"""
psspy.case(CASE2)
basekv2=[100,999]
psspy.bsys(sid2, usekv2, basekv2, numarea2, areas2)

# Temporary CON file name with contingencies
conFileName2 = "P6_I1_E1"

# Returns a list with Circuit Identifiers for only in-service
# non-transformer branches
ierr, (allBrnID2,) = psspy.abrnchar(subsysID2, 1, 3, 1, 1, 'ID')
print "BrnID =", allBrnID2

# Returns a list with From Bus Numbers for only in-service
# non-transformer branches
ierr, (FrombusNums2,) = psspy.abrnint(subsysID2, 1, 3, 1, 1, 'FROMNUMBER')
print "list of FROM bus numbers Brn ", FrombusNums2

# Returns a list with To Bus Numbers for only in-service
# non-transformer branches
ierr, (TobusNums2,) = psspy.abrnint(subsysID2, 1, 3, 1, 1, 'TONUMBER')
print "list of TO bus numbers Brn ", TobusNums2

# Opens a file object for both writing and reading. Overwrites the
# existing file if the file exists. If the file does not exist, creates
# a new file for reading and writing.
conFile = open(conFileName2 + '.con', 'w+')

# .con file header
conFile.write("COM PSS(R)E 32\n")
conFile.write("COM\n")
conFile.write("COM LOSS OF SINGLE TRANSMISSION LINE IN SUBSYSTEM '%s' \n\n"
              % subSysLabel2)

# Contingencies for removing each specified Transmission Circuit
for Brnid2, frombusnum2, tobusnum2 in zip(allBrnID2, FrombusNums2, TobusNums2):
    conFile.write("CONTINGENCY '%d_%d_%d'\n"
                  %(frombusnum2, tobusnum2, int(Brnid2)))
    conFile.write('DISCONNECT BRANCH FROM BUS %d TO BUS %d CIRCUIT %d\n'
                  %(frombusnum2, tobusnum2, int(Brnid2)))
    conFile.write('END \n\n')

conFile.write('END')
conFile.close()

# Run Multi-level AC Contingency Solution
psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
psspy.dfax([1,1], subSysLabel2, subSysLabel2, conFileName2 + '.con',
           conFileName2)
```

```
    psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                        [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                        [subSysLabel + '_DSP',subSysLabel + '_HV',
                         subSysLabel + '_HV',subSysLabel + '_HV',
                         subSysLabel + '_HV',
                         subSysLabel + '_BULK',subSysLabel + '_BULK'],
                       conFileName2 + '.dfx',
                       conFileName2 + '_TRANCKTTRIP_' + 'L%d_%d_%d'
                       %(frombusnum, tobusnum, int(Brnid)),
                       "","",subSysLabel + '.trp')


    # Generate final excel file with results
    import pssexcel

    ACCFile=sys.path[0]

    pssexcel.accc(conFileName2 + '_TRANCKTTRIP_' + 'L%d_%d_%d' %(frombusnum, tobusnum,
int(Brnid)) + ".acc",
                  ['e','b','v'],colabel='',stype='contingency',
                  busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
                  xlsfile = conFileName2 + '_TRANCKTTRIP_' + 'L%d_%d_%d' %(frombusnum, tobusnum,
int(Brnid)) + '.xls',
                  sheet='',overwritesheet=True,show=False)

    # Generate final text file with results
    psspy.report_output(islct=2, filarg="./" + conFileName2 + '_TRANCKTTRIP_' + 'L%d_%d_%d'
%(frombusnum, tobusnum, int(Brnid)) + '.txt',
                        options=[0])
    psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                                   [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                                   conFileName2 + '_TRANCKTTRIP_' + 'L%d_%d_%d'
                                   %(frombusnum, tobusnum, int(Brnid)))
    psspy.report_output(islct=2, options=[0])


    """
    For Loop Part to perform all P6_I1_E2 contingencies.
    """
    psspy.case(CASE2)
    basekv2=[100,999]
    psspy.bsys(sid2, usekv2, basekv2, numarea2, areas2)

    # Temporary CON file name with contingencies
    conFileName2 = "P6_I1_E2"

    # Returns a list with Circuit Identifiers for only in-service
    # two-winding transformers branches
    ierr, (allTrnID2,) = psspy.atrnchar(subsysID2, 1, 3, 1, 1, 'ID')
    print "TrnID =", allTrnID2

    # Returns a list with From Bus Numbers for only in-service
    # two-winding transformers branches
    ierr, (FrombusNums2,) = psspy.atrnint(subsysID2, 1, 3, 1, 1, 'FROMNUMBER')
    print "list of FROM bus numbers Trn ", FrombusNums2

    # Returns a list with To Bus Numbers for only in-service
    # two-winding transformers branches
    ierr, (TobusNums2,) = psspy.atrnint(subsysID2, 1, 3, 1, 1, 'TONUMBER')
    print "list of TO bus numbers Trn ", TobusNums2

    # Opens a file object for both writing and reading. Overwrites the
    # existing file if the file exists. If the file does not exist, creates
    # a new file for reading and writing.
    conFile = open(conFileName2 + '.con', 'w+')

    # .con file header
    conFile.write("COM PSS(R)E 32\n")
    conFile.write("COM\n")
    conFile.write("COM LOSS OF SINGLE TRANSFORMER IN SUBSYSTEM '%s' \n\n"
```

```
                % subSysLabel2)

    # Contingencies for removing each specified Transformer
    for Trnid2, frombusnum2, tobusnum2 in zip(allTrnID2, FrombusNums2, TobusNums2):
        conFile.write("CONTINGENCY '%d_%d_%d'\n"
                      %(frombusnum2, tobusnum2, int(Trnid2)))
        conFile.write('DISCONNECT BRANCH FROM BUS %d TO BUS %d CIRCUIT %d\n'
                      %(frombusnum2, tobusnum2, int(Trnid2)))
        conFile.write('END \n\n')

    conFile.write('END')
    conFile.close()

    # Run Multi-level AC Contingency Solution
    psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
    psspy.dfax([1,1], subSysLabel2, subSysLabel2, conFileName2 + '.con',
               conFileName2)

    psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                        [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                        [subSysLabel + '_DSP',subSysLabel + '_HV',
                         subSysLabel + '_HV',subSysLabel + '_HV',
                         subSysLabel + '_HV',
                         subSysLabel + '_BULK',subSysLabel + '_BULK'],
                       conFileName2 + '.dfx',
                       conFileName2 + '_TRANCKTTRIP_' + 'L%d_%d_%d'
                       %(frombusnum, tobusnum, int(Brnid)),
                       "","",subSysLabel + '.trp')


    # Generate final excel file with results
    pssexcel.accc(conFileName2 + '_TRANCKTTRIP_' + 'L%d_%d_%d' %(frombusnum, tobusnum,
int(Brnid)) + ".acc",
                  ['e','b','v'],colabel='',stype='contingency',
                  busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
                  xlsfile = conFileName2 + '_TRANCKTTRIP_' + 'L%d_%d_%d' %(frombusnum, tobusnum,
int(Brnid)) + '.xls',
                  sheet='',overwritesheet=True,show=False)

    # Generate final text file with results
    psspy.report_output(islct=2, filarg="./" + conFileName2 + '_TRANCKTTRIP_' + 'L%d_%d_%d'
%(frombusnum, tobusnum, int(Brnid)) + '.txt',
                        options=[0])
    psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                                   [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                                   conFileName2 + '_TRANCKTTRIP_' + 'L%d_%d_%d'
                                   %(frombusnum, tobusnum, int(Brnid)))
    psspy.report_output(islct=2, options=[0])


    """
    For Loop Part to perform all P6_I1_E3 contingencies.
    """
    psspy.case(CASE2)
    basekv2=[100,999]
    psspy.bsys(sid2, usekv2, basekv2, numarea2, areas2)

    # Temporary CON file name with contingencies
    conFileName2 = "P6_I1_E3"

    ### Part for FIXED Shunts Starts ###
    # Returns a list with Shunt Identifiers for only in-service
    # fixed shunts at in-service buses
    ierr, (allFxShuntID2,) = psspy.afxshuntchar(subsysID2, 1, 'ID')
    print "FxShuntID =", allFxShuntID2

    # Returns a list with Bus Numbers for only in-service
    # fixed shunts at in-service buses
    ierr, (busNums2,) = psspy.afxshuntint(subsysID2, 1, 'NUMBER')
    print "list of bus numbers Fixed Shunts ", busNums2
```

```
    ### Part for SWITCHED Shunts Starts ###
    # Returns a list with Bus Numbers for only in-service
    # switched shunts at in-service buses
    ierr, (busNumsSw2,) = psspy.aswshint(sid2, 1, 'NUMBER')
    print "list of bus numbers Switched Shunts ", busNumsSw2

    # Opens a file object for both writing and reading. Overwrites the
    # existing file if the file exists. If the file does not exist, creates
    # a new file for reading and writing.
    conFile = open(conFileName2 + '.con', 'w+')

    # .con file header
    conFile.write("COM PSS(R)E 32\n")
    conFile.write("COM\n")
    conFile.write("COM LOSS OF SINGLE SHUNT DEVICE IN SUBSYSTEM '%s' \n\n"
                  % subSysLabel2)

    # Contingencies for removing each specified fixed shunt
    for FxShuntid2, busnum2 in zip(allFxShuntID2, busNums2):
        conFile.write("CONTINGENCY 'F%dB%d'\n"
                      %( int(FxShuntid2), busnum2))
        conFile.write('REMOVE SHUNT %s FROM BUS %d\n' %(FxShuntid2, busnum2))
        conFile.write('END \n\n')

    # Contingencies for removing each specified switched shunt
    for busnum2 in busNumsSw2:
        conFile.write("CONTINGENCY 'SwB%d'\n" %busnum2)
        conFile.write('REMOVE SWSHUNT FROM BUS %d\n' %busnum2)
        conFile.write('END\n\n')

    conFile.write('END')
    conFile.close()

    # Run Multi-level AC Contingency Solution
    psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
    psspy.dfax([1,1], subSysLabel2, subSysLabel2, conFileName2 + '.con',
               conFileName2)

    psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                      [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                      [subSysLabel + '_DSP',subSysLabel + '_HV',
                       subSysLabel + '_HV',subSysLabel + '_HV',
                       subSysLabel + '_HV',
                       subSysLabel + '_BULK',subSysLabel + '_BULK'],
                     conFileName2 + '.dfx',
                     conFileName2 + '_TRANCKTTRIP_' + 'L%d_%d_%d'
                     %(frombusnum, tobusnum, int(Brnid)),
                     "","",subSysLabel + '.trp')


    # Generate final excel file with results
    pssexcel.accc(conFileName2 + '_TRANCKTTRIP_' + 'L%d_%d_%d' %(frombusnum, tobusnum,
int(Brnid)) + ".acc",
                  ['e','b','v'],colabel='',stype='contingency',
                  busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
                  xlsfile = conFileName2 + '_TRANCKTTRIP_' + 'L%d_%d_%d' %(frombusnum, tobusnum,
int(Brnid)) + '.xls',
                  sheet='',overwritesheet=True,show=False)

    # Generate final text file with results
    psspy.report_output(islct=2, filarg="./" + conFileName2 + '_TRANCKTTRIP_' + 'L%d_%d_%d'
%(frombusnum, tobusnum, int(Brnid)) + '.txt',
                        options=[0])
    psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                                   [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                                   conFileName2 + '_TRANCKTTRIP_' + 'L%d_%d_%d'
                                   %(frombusnum, tobusnum, int(Brnid)))
    psspy.report_output(islct=2, options=[0])


    """
```

```
    For Loop Part to perform all P6_I1_E4 contingencies.
    """
    psspy.case(CASE2)
    basekv2=[1,999]
    psspy.bsys(sid2, usekv2, basekv2, numarea2, areas2)

    # CON file name with contingencies
    conFileName2 = "P6_I1_E4"

    # Returns a list with DC Line names for only in-service
    # two-terminal dc lines
    ierr, (allDCBrnNames2,) = psspy.a2trmdcchar(subsysID2, 3, 1, 'DCNAME')
    print "DCBrnID =", allDCBrnNames2

    # Returns a list with From or Rectifier Bus Numbers for only in-service
    # two-terminal dc lines
    ierr, (FrombusNums2,) = psspy.a2trmdcint(subsysID2, 3, 1, 'FROMNUMBER')
    print "list of FROM bus numbers DCBrn ", FrombusNums2

    # Returns a list with To or Inverter Bus Numbers for only in-service
    # two-terminal dc lines
    ierr, (TobusNums2,) = psspy.a2trmdcint(subsysID2, 3, 1, 'TONUMBER')
    print "list of TO bus numbers DCBrn ", TobusNums2

    # Opens a file object for both writing and reading. Overwrites the
    # existing file if the file exists. If the file does not exist, creates
    # a new file for reading and writing.
    conFile = open(conFileName2 + '.con', 'w+')

    # .con file header
    conFile.write("COM PSS(R)E 32\n")
    conFile.write("COM\n")
    conFile.write("COM LOSS OF SINGLE POLE OF A DC LINE IN SUBSYSTEM '%s' \n\n"
                  % subSysLabel)

    # Contingencies for removing each specified Pole of a DC Line
    for DCBrnname2, frombusnum2, tobusnum2 in zip(allDCBrnNames2, FrombusNums2, TobusNums2):
        conFile.write("CONTINGENCY '%d_%d_%s'\n"
                      %(frombusnum2, tobusnum2, DCBrnname2))
        conFile.write('BLOCK TWOTERMDC %s\n' %DCBrnname2)
        conFile.write('END \n\n')

    conFile.write('END')
    conFile.close()

    # Run Multi-level AC Contingency Solution
    psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
    psspy.dfax([1,1], subSysLabel2, subSysLabel2, conFileName2 + '.con',
               conFileName2)

    psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                       [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0],
                       [subSysLabel + '_DSP',subSysLabel + '_HV',
                        subSysLabel + '_HV',subSysLabel + '_HV',
                        subSysLabel + '_HV',
                        subSysLabel + '_BULK',subSysLabel + '_BULK'],
                      conFileName2 + '.dfx',
                      conFileName2 + '_TRANCKTTRIP_' + 'L%d_%d_%d'
                      %(frombusnum, tobusnum, int(Brnid)),
                      "","",subSysLabel + '.trp')


    # Generate final excel file with results
    pssexcel.accc(conFileName2 + '_TRANCKTTRIP_' + 'L%d_%d_%d' %(frombusnum, tobusnum,
int(Brnid)) + ".acc",
                  ['e','b','v'],colabel='',stype='contingency',
                  busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
                  xlsfile = conFileName2 + '_TRANCKTTRIP_' + 'L%d_%d_%d' %(frombusnum, tobusnum,
int(Brnid)) + '.xls',
                  sheet='',overwritesheet=True,show=False)
```

```
    # Generate final text file with results
    psspy.report_output(islct=2, filarg="./" + conFileName2 + '_TRANCKTTRIP_' + 'L%d_%d_%d'
%(frombusnum, tobusnum, int(Brnid)) + '.txt',
                        options=[0])
    psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                                   [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                                   conFileName2 + '_TRANCKTTRIP_' + 'L%d_%d_%d'
                                   %(frombusnum, tobusnum, int(Brnid)))
    psspy.report_output(islct=2, options=[0])


print('\nP6_I1 Category Complete!')
print('NOTE: It is highly recommended to create a new folder where')
print('      to move all the files created by this program.\n')
```

A.6 Category P6_I2

```
"""
THIS PROGRAM PERFORMS P6 CONTINGENCIES PER TPL-001-4
INITIAL CONDITION 2 WITH EVENTS 1 THROUGH 4
"""

"""
First Part is to calculate Initial Condition I2.
That is, loss of a Transformer followed by
System adjustments
"""

import os,sys

PSSE_LOCATION = r"C:\Program Files\PTI\PSSE32\PSSBIN"
sys.path.append(PSSE_LOCATION)
os.environ['PATH'] = os.environ['PATH'] + ';' +  PSSE_LOCATION

import psspy
import redirect
redirect.psse2py()

from sys import argv
if len(sys.argv) > 1:
    script, CASE1, areas, subSysLabel = argv
else:
    CASE1 = raw_input("Base case name, without *.sav extension: ")
    areas = raw_input("Area number where contingencies will take place: ")
    subSysLabel = raw_input("Area name where contingencies will take place: ")

areas = [int(areas)]


#---------------------------------
# PSS/E Saved case

CASE = CASE1 + '.sav'

psspy.psseinit(50000)
psspy.prompt_output(4,"",[0,0])
psspy.alert_output(4,"",[0,0])

# Defining a subsystem with buses in study area
sid = 0
usekv = 1
numarea = len(areas)
subsysID = sid

psspy.case(CASE)
basekv=[100,999]
```

```
psspy.bsys(sid, usekv, basekv, numarea, areas)

conFileName1 = "P6_P1_E3"

# Returns a list with Circuit Identifiers for only in-service
# two-winding transformers branches
ierr, (allTrnID,) = psspy.atrnchar(subsysID, 1, 3, 1, 1, 'ID')
print "TrnID =", allTrnID

# Returns a list with From Bus Numbers for only in-service
# two-winding transformers branches
ierr, (FrombusNums,) = psspy.atrnint(subsysID, 1, 3, 1, 1, 'FROMNUMBER')
print "list of FROM bus numbers Trn ", FrombusNums

# Returns a list with To Bus Numbers for only in-service
# two-winding transformers branches
ierr, (TobusNums,) = psspy.atrnint(subsysID, 1, 3, 1, 1, 'TONUMBER')
print "list of TO bus numbers Trn ", TobusNums

# Loop to lose each Transformer, perform system adjustments,
# and study the respective events
for Trnid, frombusnum, tobusnum in zip(allTrnID, FrombusNums, TobusNums):
    """
    For Loop Part to create a temporary contingency file for each Transformer
    lost in ORIGINAL case file
    """
    psspy.case(CASE)
    psspy.prompt_output(4,"",[0,0])
    psspy.alert_output(4,"",[0,0])

    # Opens a file object for both writing and reading. Overwrites the
    # existing file if the file exists. If the file does not exist, creates
    # a new file for reading and writing.
    conFile = open(conFileName1 + '.con', 'w+')

    # .con file header
    conFile.write("COM PSS(R)E 32\n")
    conFile.write("COM\n")
    conFile.write("COM LOSS OF SINGLE TRANSFORMER IN SUBSYSTEM '%s' \n\n"
                  % subSysLabel)

    # Contingency for removing each specified Transformer, one at a time.
    conFile.write("CONTINGENCY '%d_%d_%d'\n"
                  %(frombusnum, tobusnum, int(Trnid)))
    conFile.write('DISCONNECT BRANCH FROM BUS %d TO BUS %d CIRCUIT %d\n'
                  %(frombusnum, tobusnum, int(Trnid)))
    conFile.write('END \n\n')
    conFile.write('END')
    conFile.close()

    psspy.dfax([1,1], subSysLabel , subSysLabel , conFileName1 + '.con',
               conFileName1)

    # System adjustment: Units Redispatch while Imposing Contingency
    psspy.gendsp([2,0],[subSysLabel + '_DSP','%d_%d_%d'
                        %(frombusnum, tobusnum, int(Trnid))],
                 conFileName1 + '.dfx',"","")

    # The next piece of code is to avoid large islands in the solution
    ierr,buses = psspy.tree(1,0)

    while buses > 0:
        ierr,buses = psspy.tree(2,1)

    # After redispatching, power flow Solution is solved a few times.
    # Further System adjustments: enable stepping tap, phase shift,
    # dc tap, and switched shunt adjustments.
    psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
    psspy.fdns([1,2,1,1,1,0,99,0])
    psspy.fdns([1,2,1,1,1,0,99,0])
    psspy.fdns([1,2,1,1,1,0,99,0])
```

```
psspy.fdns([1,2,1,1,1,0,99,0])
psspy.fdns([1,2,1,1,1,0,99,0])

# The new solved case is saved. This will be the Initial Condition
psspy.save(CASE1 + '_TRANSFTRIP_' + 'L%d_%d_%d'
           %(frombusnum, tobusnum, int(Trnid)) + '.sav')

# New working case to be used with the reamining contingecy events,
# E1 through E3.
CASE2 = CASE1 + '_TRANSFTRIP_' + 'L%d_%d_%d' %(frombusnum, tobusnum, int(Trnid)) + '.sav'

psspy.case(CASE2)
psspy.prompt_output(4,"",[0,0])
psspy.alert_output(4,"",[0,0])

ierr, bus, mismatch = psspy.maxmsm()
print bus

mva = psspy.sysmsm()
if mva > 0.7:
    continue

# Defining a subsystem with buses in study area 1
sid2 = sid
usekv2 = usekv
areas2 = areas
numarea2 = len(areas2)
subsysID2 = sid2
subSysLabel2 = subSysLabel


"""
For Loop Part to perform all P6_I2_E1 contingencies.
"""
psspy.case(CASE2)
basekv2=[100,999]
psspy.bsys(sid2, usekv2, basekv2, numarea2, areas2)

# Temporary CON file name with contingencies
conFileName2 = "P6_I2_E1"

# Returns a list with Circuit Identifiers for only in-service
# non-transformer branches
ierr, (allBrnID2,) = psspy.abrnchar(subsysID2, 1, 3, 1, 1, 'ID')
print "BrnID =", allBrnID2

# Returns a list with From Bus Numbers for only in-service
# non-transformer branches
ierr, (FrombusNums2,) = psspy.abrnint(subsysID2, 1, 3, 1, 1, 'FROMNUMBER')
print "list of FROM bus numbers Brn ", FrombusNums2

# Returns a list with To Bus Numbers for only in-service
# non-transformer branches
ierr, (TobusNums2,) = psspy.abrnint(subsysID2, 1, 3, 1, 1, 'TONUMBER')
print "list of TO bus numbers Brn ", TobusNums2

# Opens a file object for both writing and reading. Overwrites the
# existing file if the file exists. If the file does not exist, creates
# a new file for reading and writing.
conFile = open(conFileName2 + '.con', 'w+')

# .con file header
conFile.write("COM PSS(R)E 32\n")
conFile.write("COM\n")
conFile.write("COM LOSS OF SINGLE TRANSMISSION LINE IN SUBSYSTEM '%s' \n\n"
              % subSysLabel2)

# Contingencies for removing each specified Transmission Circuit
for Brnid2, frombusnum2, tobusnum2 in zip(allBrnID2, FrombusNums2, TobusNums2):
    conFile.write("CONTINGENCY '%d_%d_%d'\n"
                  %(frombusnum2, tobusnum2, int(Brnid2)))
```

```
        conFile.write('DISCONNECT BRANCH FROM BUS %d TO BUS %d CIRCUIT %d\n'
                      %(frombusnum2, tobusnum2, int(Brnid2)))
        conFile.write('END \n\n')

    conFile.write('END')
    conFile.close()

    # Run Multi-level AC Contingency Solution
    psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
    psspy.dfax([1,1], subSysLabel2, subSysLabel2, conFileName2 + '.con',
               conFileName2)

    psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                        [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                        [subSysLabel + '_DSP',subSysLabel + '_HV',
                         subSysLabel + '_HV',subSysLabel + '_HV',
                         subSysLabel + '_HV',
                         subSysLabel + '_BULK',subSysLabel + '_BULK'],
                       conFileName2 + '.dfx',
                       conFileName2 + '_TRANSFTRIP_' + 'L%d_%d_%d'
                       %(frombusnum, tobusnum, int(Trnid)),
                       "","",subSysLabel + '.trp')


    # Generate final excel file with results
    import pssexcel

    ACCFile=sys.path[0]

    pssexcel.accc(conFileName2 + '_TRANSFTRIP_' + 'L%d_%d_%d' %(frombusnum, tobusnum, int(Trnid))
+ ".acc",
                  ['e','b','v'],colabel='',stype='contingency',
                  busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
                  xlsfile = conFileName2 + '_TRANSFTRIP_' + 'L%d_%d_%d' %(frombusnum, tobusnum,
int(Trnid)) + '.xls',
                  sheet='',overwritesheet=True,show=False)

    # Generate final text file with results
    psspy.report_output(islct=2, filarg="./" + conFileName2 + '_TRANSFTRIP_' + 'L%d_%d_%d'
%(frombusnum, tobusnum, int(Trnid)) + '.txt',
                        options=[0])
    psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                                   [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                                   conFileName2 + '_TRANSFTRIP_' + 'L%d_%d_%d'
                                   %(frombusnum, tobusnum, int(Trnid)))
    psspy.report_output(islct=2, options=[0])


    """
    For Loop Part to perform all P6_I2_E2 contingencies.
    """
    psspy.case(CASE2)
    basekv2=[100,999]
    psspy.bsys(sid2, usekv2, basekv2, numarea2, areas2)

    # Temporary CON file name with contingencies
    conFileName2 = "P6_I2_E2"

    # Returns a list with Circuit Identifiers for only in-service
    # two-winding transformers branches
    ierr, (allTrnID2,) = psspy.atrnchar(subsysID2, 1, 3, 1, 1, 'ID')
    print "TrnID =", allTrnID2

    # Returns a list with From Bus Numbers for only in-service
    # two-winding transformers branches
    ierr, (FrombusNums2,) = psspy.atrnint(subsysID2, 1, 3, 1, 1, 'FROMNUMBER')
    print "list of FROM bus numbers Trn ", FrombusNums2

    # Returns a list with To Bus Numbers for only in-service
    # two-winding transformers branches
    ierr, (TobusNums2,) = psspy.atrnint(subsysID2, 1, 3, 1, 1, 'TONUMBER')
```

```
    print "list of TO bus numbers Trn ", TobusNums2

    # Opens a file object for both writing and reading. Overwrites the
    # existing file if the file exists. If the file does not exist, creates
    # a new file for reading and writing.
    conFile = open(conFileName2 + '.con', 'w+')

    # .con file header
    conFile.write("COM PSS(R)E 32\n")
    conFile.write("COM\n")
    conFile.write("COM LOSS OF SINGLE TRANSFORMER IN SUBSYSTEM '%s' \n\n"
                  % subSysLabel2)

    # Contingencies for removing each specified Transformer
    for Trnid2, frombusnum2, tobusnum2 in zip(allTrnID2, FrombusNums2, TobusNums2):
        conFile.write("CONTINGENCY '%d_%d_%d'\n"
                      %(frombusnum2, tobusnum2, int(Trnid2)))
        conFile.write('DISCONNECT BRANCH FROM BUS %d TO BUS %d CIRCUIT %d\n'
                      %(frombusnum2, tobusnum2, int(Trnid2)))
        conFile.write('END \n\n')

    conFile.write('END')
    conFile.close()

    # Run Multi-level AC Contingency Solution
    psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
    psspy.dfax([1,1], subSysLabel2, subSysLabel2, conFileName2 + '.con',
               conFileName2)

    psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                       [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                       [subSysLabel + '_DSP',subSysLabel + '_HV',
                        subSysLabel + '_HV',subSysLabel + '_HV',
                        subSysLabel + '_HV',
                        subSysLabel + '_BULK',subSysLabel + '_BULK'],
                      conFileName2 + '.dfx',
                      conFileName2 + '_TRANSFTRIP_' + 'L%d_%d_%d'
                      %(frombusnum, tobusnum, int(Trnid)),
                      "","",subSysLabel + '.trp')

    # Generate final excel file with results

    pssexcel.accc(conFileName2 + '_TRANSFTRIP_' + 'L%d_%d_%d' %(frombusnum, tobusnum, int(Trnid))
+ ".acc",
                  ['e','b','v'],colabel='',stype='contingency',
                  busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
                  xlsfile = conFileName2 + '_TRANSFTRIP_' + 'L%d_%d_%d' %(frombusnum, tobusnum,
int(Trnid)) + '.xls',
                  sheet='',overwritesheet=True,show=False)

    # Generate final text file with results
    psspy.report_output(islct=2, filarg="./" + conFileName2 + '_TRANSFTRIP_' + 'L%d_%d_%d'
%(frombusnum, tobusnum, int(Trnid)) + '.txt',
                        options=[0])
    psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                                   [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                                   conFileName2 + '_TRANSFTRIP_' + 'L%d_%d_%d'
                                   %(frombusnum, tobusnum, int(Trnid)))
    psspy.report_output(islct=2, options=[0])


    """
    For Loop Part to perform all P6_I2_E3 contingencies.
    """
    psspy.case(CASE2)
    basekv2=[100,999]
    psspy.bsys(sid2, usekv2, basekv2, numarea2, areas2)

    # Temporary CON file name with contingencies
    conFileName2 = "P6_I2_E3"
```

```
    ### Part for FIXED Shunts Starts ###
    # Returns a list with Shunt Identifiers for only in-service
    # fixed shunts at in-service buses
    ierr, (allFxShuntID2,) = psspy.afxshuntchar(subsysID2, 1, 'ID')
    print "FxShuntID =", allFxShuntID2

    # Returns a list with Bus Numbers for only in-service
    # fixed shunts at in-service buses
    ierr, (busNums2,) = psspy.afxshuntint(subsysID2, 1, 'NUMBER')
    print "list of bus numbers Fixed Shunts ", busNums2

    ### Part for SWITCHED Shunts Starts ###
    # Returns a list with Bus Numbers for only in-service
    # switched shunts at in-service buses
    ierr, (busNumsSw2,) = psspy.aswshint(sid2, 1, 'NUMBER')
    print "list of bus numbers Switched Shunts ", busNumsSw2

    # Opens a file object for both writing and reading. Overwrites the
    # existing file if the file exists. If the file does not exist, creates
    # a new file for reading and writing.
    conFile = open(conFileName2 + '.con', 'w+')

    # .con file header
    conFile.write("COM PSS(R)E 32\n")
    conFile.write("COM\n")
    conFile.write("COM LOSS OF SINGLE SHUNT DEVICE IN SUBSYSTEM '%s' \n\n"
                  % subSysLabel2)

    # Contingencies for removing each specified fixed shunt
    for FxShuntid2, busnum2 in zip(allFxShuntID2, busNums2):
        conFile.write("CONTINGENCY 'F%dB%d'\n"
                      %( int(FxShuntid2), busnum2))
        conFile.write('REMOVE SHUNT %s FROM BUS %d\n' %(FxShuntid2, busnum2))
        conFile.write('END \n\n')

    # Contingencies for removing each specified switched shunt
    for busnum2 in busNumsSw2:
        conFile.write("CONTINGENCY 'SwB%d'\n" %busnum2)
        conFile.write('REMOVE SWSHUNT FROM BUS %d\n' %busnum2)
        conFile.write('END\n\n')

    conFile.write('END')
    conFile.close()

    # Run Multi-level AC Contingency Solution
    psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
    psspy.dfax([1,1], subSysLabel2, subSysLabel2, conFileName2 + '.con',
               conFileName2)

    psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                       [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                       [subSysLabel + '_DSP',subSysLabel + '_HV',
                        subSysLabel + '_HV',subSysLabel + '_HV',
                        subSysLabel + '_HV',
                        subSysLabel + '_BULK',subSysLabel + '_BULK'],
                      conFileName2 + '.dfx',
                      conFileName2 + '_TRANSFTRIP_' + 'L%d_%d_%d'
                      %(frombusnum, tobusnum, int(Trnid)),
                      "","",subSysLabel + '.trp')

    # Generate final excel file with results

    pssexcel.accc(conFileName2 + '_TRANSFTRIP_' + 'L%d_%d_%d' %(frombusnum, tobusnum, int(Trnid))
+ ".acc",
                  ['e','b','v'],colabel='',stype='contingency',
                  busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
                  xlsfile = conFileName2 + '_TRANSFTRIP_' + 'L%d_%d_%d' %(frombusnum, tobusnum,
int(Trnid)) + '.xls',
                  sheet='',overwritesheet=True,show=False)

    # Generate final text file with results
```

```
    psspy.report_output(islct=2, filarg="./" + conFileName2 + '_TRANSFTRIP_' + 'L%d_%d_%d'
%(frombusnum, tobusnum, int(Trnid)) + '.txt',
                        options=[0])
    psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                                   [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                                   conFileName2 + '_TRANSFTRIP_' + 'L%d_%d_%d'
                                   %(frombusnum, tobusnum, int(Trnid)))
    psspy.report_output(islct=2, options=[0])


    """
    For Loop Part to perform all P6_I2_E4 contingencies.
    """
    psspy.case(CASE2)
    basekv2=[1,999]
    psspy.bsys(sid2, usekv2, basekv2, numarea2, areas2)

    # CON file name with contingencies
    conFileName2 = "P6_I2_E4"

    # Returns a list with DC Line names for only in-service
    # two-terminal dc lines
    ierr, (allDCBrnNames2,) = psspy.a2trmdcchar(subsysID2, 3, 1, 'DCNAME')
    print "DCBrnID =", allDCBrnNames2

    # Returns a list with From or Rectifier Bus Numbers for only in-service
    # two-terminal dc lines
    ierr, (FrombusNums2,) = psspy.a2trmdcint(subsysID2, 3, 1, 'FROMNUMBER')
    print "list of FROM bus numbers DCBrn ", FrombusNums2

    # Returns a list with To or Inverter Bus Numbers for only in-service
    # two-terminal dc lines
    ierr, (TobusNums2,) = psspy.a2trmdcint(subsysID2, 3, 1, 'TONUMBER')
    print "list of TO bus numbers DCBrn ", TobusNums2

    # Opens a file object for both writing and reading. Overwrites the
    # existing file if the file exists. If the file does not exist, creates
    # a new file for reading and writing.
    conFile = open(conFileName2 + '.con', 'w+')

    # .con file header
    conFile.write("COM PSS(R)E 32\n")
    conFile.write("COM\n")
    conFile.write("COM LOSS OF SINGLE POLE OF A DC LINE IN SUBSYSTEM '%s' \n\n"
                  % subSysLabel)

    # Contingencies for removing each specified Pole of a DC Line
    for DCBrnname2, frombusnum2, tobusnum2 in zip(allDCBrnNames2, FrombusNums2, TobusNums2):
        conFile.write("CONTINGENCY '%d_%d_%s'\n"
                      %(frombusnum2, tobusnum2, DCBrnname2))
        conFile.write('BLOCK TWOTERMDC %s\n' %DCBrnname2)
        conFile.write('END \n\n')

    conFile.write('END')
    conFile.close()

    # Run Multi-level AC Contingency Solution
    psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
    psspy.dfax([1,1], subSysLabel2, subSysLabel2, conFileName2 + '.con',
               conFileName2)

    psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                       [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                       [subSysLabel + '_DSP',subSysLabel + '_HV',
                        subSysLabel + '_HV',subSysLabel + '_HV',
                        subSysLabel + '_HV',
                        subSysLabel + '_BULK',subSysLabel + '_BULK'],
                      conFileName2 + '.dfx',
                      conFileName2 + '_TRANSFTRIP_' + 'L%d_%d_%d'
                      %(frombusnum, tobusnum, int(Trnid)),
                      "","",subSysLabel + '.trp')
```

```
    # Generate final excel file with results
    pssexcel.accc(conFileName2 + '_TRANSFTRIP_' + 'L%d_%d_%d' %(frombusnum, tobusnum, int(Trnid))
+ ".acc",
                  ['e','b','v'],colabel='',stype='contingency',
                  busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
                  xlsfile = conFileName2 + '_TRANSFTRIP_' + 'L%d_%d_%d' %(frombusnum, tobusnum,
int(Trnid)) + '.xls',
                  sheet='',overwritesheet=True,show=False)

    # Generate final text file with results
    psspy.report_output(islct=2, filarg="./" + conFileName2 + '_TRANSFTRIP_' + 'L%d_%d_%d'
%(frombusnum, tobusnum, int(Trnid)) + '.txt',
                        options=[0])
    psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                                   [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                                   conFileName2 + '_TRANSFTRIP_' + 'L%d_%d_%d'
                                   %(frombusnum, tobusnum, int(Trnid)))
    psspy.report_output(islct=2, options=[0])


print('\nP6_I2 Category Complete!')
print('NOTE: It is highly recommended to create a new folder where')
print('      to move all the files created by this program.\n')
```

A.7 Category P6_I3

```
"""
THIS PROGRAM PERFORMS P6 CONTINGENCIES PER TPL-001-4
INITIAL CONDITION 3 WITH EVENTS 1 THROUGH 4
"""

"""
First Part is to calculate Initial Condition I3.
That is, loss of a Shunt Device followed by
System adjustments
"""

import os,sys

PSSE_LOCATION = r"C:\Program Files\PTI\PSSE32\PSSBIN"
sys.path.append(PSSE_LOCATION)
os.environ['PATH'] = os.environ['PATH'] + ';' +  PSSE_LOCATION

import psspy
import redirect
redirect.psse2py()

from sys import argv
if len(sys.argv) > 1:
    script, CASE1, areas, subSysLabel = argv
else:
    CASE1 = raw_input("Base case name, without *.sav extension: ")
    areas = raw_input("Area number where contingencies will take place: ")
    subSysLabel = raw_input("Area name where contingencies will take place: ")

areas = [int(areas)]


#---------------------------------
# PSS/E Saved case

CASE = CASE1 + '.sav'

psspy.psseinit(50000)
psspy.prompt_output(4,"",[0,0])
```

```
psspy.alert_output(4,"",[0,0])

# Defining a subsystem with buses in study area
sid = 0
usekv = 1
numarea = len(areas)
subsysID = sid

psspy.case(CASE)
basekv=[100,999]
psspy.bsys(sid, usekv, basekv, numarea, areas)

conFileName1 = "P6_P1_E4"

### Part for FIXED Shunts Starts ###
# Returns a list with Shunt Identifiers for only in-service
# fixed shunts at in-service buses
ierr, (allFxShuntID,) = psspy.afxshuntchar(subsysID, 1, 'ID')
print "FxShuntID =", allFxShuntID

# Returns a list with Bus Numbers for only in-service
# fixed shunts at in-service buses
ierr, (busNums,) = psspy.afxshuntint(subsysID, 1, 'NUMBER')
print "list of bus numbers Fixed Shunts ", busNums

### Part for SWITCHED Shunts Starts ###
# Returns a list with Bus Numbers for only in-service
# switched shunts at in-service buses
ierr, (busNumsSw,) = psspy.aswshint(sid, 1, 'NUMBER')
print "list of bus numbers Switched Shunts ", busNumsSw

# Loop to lose each Fixed Shunt Device, perform system adjustments,
# and study the respective events
for FxShuntid, busnum in zip(allFxShuntID, busNums):
    """
    For Loop Part to create a temporary contingency file for each Fixed Shunt
    Device lost in ORIGINAL case file
    """
    psspy.case(CASE)
    psspy.prompt_output(4,"",[0,0])
    psspy.alert_output(4,"",[0,0])

    # Opens a file object for both writing and reading. Overwrites the
    # existing file if the file exists. If the file does not exist, creates
    # a new file for reading and writing.
    conFile = open(conFileName1 + '.con', 'w+')

    # .con file header
    conFile.write("COM PSS(R)E 32\n")
    conFile.write("COM\n")
    conFile.write("COM LOSS OF SINGLE SHUNT DEVICE IN SUBSYSTEM '%s' \n\n"
                  % subSysLabel)

    # Contingency for removing each specified Fixed Shunt, one at a time.
    conFile.write("CONTINGENCY 'F%dB%d'\n" %( int(FxShuntid), busnum))
    conFile.write('REMOVE SHUNT %s FROM BUS %d\n' %(FxShuntid, busnum))
    conFile.write('END \n\n')
    conFile.write('END')
    conFile.close()

    psspy.dfax([1,1], subSysLabel , subSysLabel , conFileName1 + '.con',
              conFileName1)

    # System adjustment: Units Redispatch while Imposing Contingency
    psspy.gendsp([2,0],[subSysLabel + '_DSP','F%dB%d'
                        %( int(FxShuntid), busnum)],
                 conFileName1 + '.dfx',"","")

    # The next piece of code is to avoid large islands in the solution
    ierr,buses = psspy.tree(1,0)
```

```
while buses > 0:
    ierr,buses = psspy.tree(2,1)

# After redispatching, power flow Solution is solved a few times.
# Further System adjustments: enable stepping tap, phase shift,
# dc tap, and switched shunt adjustments.
psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
psspy.fdns([1,2,1,1,1,0,99,0])
psspy.fdns([1,2,1,1,1,0,99,0])
psspy.fdns([1,2,1,1,1,0,99,0])
psspy.fdns([1,2,1,1,1,0,99,0])
psspy.fdns([1,2,1,1,1,0,99,0])

# The new solved case is saved. This will be the Initial Condition
psspy.save(CASE1 + '_FXSHUNTTRIP_' + 'FxS%d-BUS%d'
           %( int(FxShuntid), busnum) + '.sav')

# New working case to be used with the reamining contingecy events,
# E1 through E3.
CASE2 = CASE1 + '_FXSHUNTTRIP_' + 'FxS%d-BUS%d' %( int(FxShuntid), busnum) + '.sav'

psspy.case(CASE2)
psspy.prompt_output(4,"",[0,0])
psspy.alert_output(4,"",[0,0])

ierr, bus, mismatch = psspy.maxmsm()
print bus

mva = psspy.sysmsm()
if mva > 0.7:
    continue

# Defining a subsystem with buses in study area 1
sid2 = sid
usekv2 = usekv
areas2 = areas
numarea2 = len(areas2)
subsysID2 = sid2
subSysLabel2 = subSysLabel


"""
For Loop Part to perform all P6_I3_E1 contingencies.
"""
psspy.case(CASE2)
basekv2=[100,999]
psspy.bsys(sid2, usekv2, basekv2, numarea2, areas2)

# Temporary CON file name with contingencies
conFileName2 = "P6_I3_E1"

# Returns a list with Circuit Identifiers for only in-service
# non-transformer branches
ierr, (allBrnID2,) = psspy.abrnchar(subsysID2, 1, 3, 1, 1, 'ID')
print "BrnID =", allBrnID2

# Returns a list with From Bus Numbers for only in-service
# non-transformer branches
ierr, (FrombusNums2,) = psspy.abrnint(subsysID2, 1, 3, 1, 1, 'FROMNUMBER')
print "list of FROM bus numbers Brn ", FrombusNums2

# Returns a list with To Bus Numbers for only in-service
# non-transformer branches
ierr, (TobusNums2,) = psspy.abrnint(subsysID2, 1, 3, 1, 1, 'TONUMBER')
print "list of TO bus numbers Brn ", TobusNums2

# Opens a file object for both writing and reading. Overwrites the
# existing file if the file exists. If the file does not exist, creates
# a new file for reading and writing.
conFile = open(conFileName2 + '.con', 'w+')
```

```
    # .con file header
    conFile.write("COM PSS(R)E 32\n")
    conFile.write("COM\n")
    conFile.write("COM LOSS OF SINGLE TRANSMISSION LINE IN SUBSYSTEM '%s' \n\n"
                  % subSysLabel2)

    # Contingencies for removing each specified Transmission Circuit
    for Brnid2, frombusnum2, tobusnum2 in zip(allBrnID2, FrombusNums2, TobusNums2):
        conFile.write("CONTINGENCY '%d_%d_%d'\n"
                      %(frombusnum2, tobusnum2, int(Brnid2)))
        conFile.write('DISCONNECT BRANCH FROM BUS %d TO BUS %d CIRCUIT %d\n'
                      %(frombusnum2, tobusnum2, int(Brnid2)))
        conFile.write('END \n\n')

    conFile.write('END')
    conFile.close()

    # Run Multi-level AC Contingency Solution
    psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
    psspy.dfax([1,1], subSysLabel2, subSysLabel2, conFileName2 + '.con',
               conFileName2)

    psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                        [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                        [subSysLabel + '_DSP',subSysLabel + '_HV',
                         subSysLabel + '_HV',subSysLabel + '_HV',
                         subSysLabel + '_HV',
                         subSysLabel + '_BULK',subSysLabel + '_BULK'],
                       conFileName2 + '.dfx',
                       conFileName2 + '_FXSHUNTTRIP_' + 'FxS%d-BUS%d'
                       %( int(FxShuntid), busnum),
                       "","",subSysLabel + '.trp')


    # Generate final excel file with results
    import pssexcel

    ACCFile=sys.path[0]

    pssexcel.accc(conFileName2 + '_FXSHUNTTRIP_' + 'FxS%d-BUS%d' %( int(FxShuntid), busnum) +
".acc",
                  ['e','b','v'],colabel='',stype='contingency',
                  busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
                  xlsfile = conFileName2 + '_FXSHUNTTRIP_' + 'FxS%d-BUS%d' %( int(FxShuntid),
busnum) + '.xls',
                  sheet='',overwritesheet=True,show=False)

    # Generate final text file with results
    psspy.report_output(islct=2, filarg="./" + conFileName2 + '_FXSHUNTTRIP_' + 'FxS%d-BUS%d' %(
int(FxShuntid), busnum) + '.txt',
                        options=[0])
    psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                                   [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                                   conFileName2 + '_FXSHUNTTRIP_' + 'FxS%d-BUS%d'
                                   %( int(FxShuntid), busnum))
    psspy.report_output(islct=2, options=[0])


    """
    For Loop Part to perform all P6_I3_E2 contingencies.
    """
    psspy.case(CASE2)
    basekv2=[100,999]
    psspy.bsys(sid2, usekv2, basekv2, numarea2, areas2)

    # Temporary CON file name with contingencies
    conFileName2 = "P6_I3_E2"

    # Returns a list with Circuit Identifiers for only in-service
    # two-winding transformers branches
    ierr, (allTrnID2,) = psspy.atrnchar(subsysID2, 1, 3, 1, 1, 'ID')
```

```
    print "TrnID =", allTrnID2

    # Returns a list with From Bus Numbers for only in-service
    # two-winding transformers branches
    ierr, (FrombusNums2,) = psspy.atrnint(subsysID2, 1, 3, 1, 1, 'FROMNUMBER')
    print "list of FROM bus numbers Trn ", FrombusNums2

    # Returns a list with To Bus Numbers for only in-service
    # two-winding transformers branches
    ierr, (TobusNums2,) = psspy.atrnint(subsysID2, 1, 3, 1, 1, 'TONUMBER')
    print "list of TO bus numbers Trn ", TobusNums2

    # Opens a file object for both writing and reading. Overwrites the
    # existing file if the file exists. If the file does not exist, creates
    # a new file for reading and writing.
    conFile = open(conFileName2 + '.con', 'w+')

    # .con file header
    conFile.write("COM PSS(R)E 32\n")
    conFile.write("COM\n")
    conFile.write("COM LOSS OF SINGLE TRANSFORMER IN SUBSYSTEM '%s' \n\n"
                  % subSysLabel2)

    # Contingencies for removing each specified Transformer
    for Trnid2, frombusnum2, tobusnum2 in zip(allTrnID2, FrombusNums2, TobusNums2):
        conFile.write("CONTINGENCY '%d_%d_%d'\n"
                      %(frombusnum2, tobusnum2, int(Trnid2)))
        conFile.write('DISCONNECT BRANCH FROM BUS %d TO BUS %d CIRCUIT %d\n'
                      %(frombusnum2, tobusnum2, int(Trnid2)))
        conFile.write('END \n\n')

    conFile.write('END')
    conFile.close()

    # Run Multi-level AC Contingency Solution
    psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
    psspy.dfax([1,1], subSysLabel2, subSysLabel2, conFileName2 + '.con',
               conFileName2)

    psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                       [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                       [subSysLabel + '_DSP',subSysLabel + '_HV',
                        subSysLabel + '_HV',subSysLabel + '_HV',
                        subSysLabel + '_HV',
                        subSysLabel + '_BULK',subSysLabel + '_BULK'],
                      conFileName2 + '.dfx',
                      conFileName2 + '_FXSHUNTTRIP_' + 'FxS%d-BUS%d'
                      %( int(FxShuntid), busnum),
                      "","",subSysLabel + '.trp')


    # Generate final excel file with results
    pssexcel.accc(conFileName2 + '_FXSHUNTTRIP_' + 'FxS%d-BUS%d' %( int(FxShuntid), busnum) +
".acc",
                  ['e','b','v'],colabel='',stype='contingency',
                  busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
                  xlsfile = conFileName2 + '_FXSHUNTTRIP_' + 'FxS%d-BUS%d' %( int(FxShuntid),
busnum) + '.xls',
                  sheet='',overwritesheet=True,show=False)

    # Generate final text file with results
    psspy.report_output(islct=2, filarg="./" + conFileName2 + '_FXSHUNTTRIP_' + 'FxS%d-BUS%d' %(
int(FxShuntid), busnum) + '.txt',
                        options=[0])
    psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                                   [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                                   conFileName2 + '_FXSHUNTTRIP_' + 'FxS%d-BUS%d'
                                   %( int(FxShuntid), busnum))
    psspy.report_output(islct=2, options=[0])
```

```
"""
For Loop Part to perform all P6_I3_E3 contingencies.
"""
psspy.case(CASE2)
basekv2=[100,999]
psspy.bsys(sid2, usekv2, basekv2, numarea2, areas2)

# Temporary CON file name with contingencies
conFileName2 = "P6_I3_E3"

### Part for FIXED Shunts Starts ###
# Returns a list with Shunt Identifiers for only in-service
# fixed shunts at in-service buses
ierr, (allFxShuntID2,) = psspy.afxshuntchar(subsysID2, 1, 'ID')
print "FxShuntID =", allFxShuntID2

# Returns a list with Bus Numbers for only in-service
# fixed shunts at in-service buses
ierr, (busNums2,) = psspy.afxshuntint(subsysID2, 1, 'NUMBER')
print "list of bus numbers Fixed Shunts ", busNums2

### Part for SWITCHED Shunts Starts ###
# Returns a list with Bus Numbers for only in-service
# switched shunts at in-service buses
ierr, (busNumsSw2,) = psspy.aswshint(sid2, 1, 'NUMBER')
print "list of bus numbers Switched Shunts ", busNumsSw2

# Opens a file object for both writing and reading. Overwrites the
# existing file if the file exists. If the file does not exist, creates
# a new file for reading and writing.
conFile = open(conFileName2 + '.con', 'w+')

# .con file header
conFile.write("COM PSS(R)E 32\n")
conFile.write("COM\n")
conFile.write("COM LOSS OF SINGLE SHUNT DEVICE IN SUBSYSTEM '%s' \n\n"
              % subSysLabel2)

# Contingencies for removing each specified fixed shunt
for FxShuntid2, busnum2 in zip(allFxShuntID2, busNums2):
    conFile.write("CONTINGENCY 'F%dB%d'\n"
                  %( int(FxShuntid2), busnum2))
    conFile.write('REMOVE SHUNT %s FROM BUS %d\n' %(FxShuntid2, busnum2))
    conFile.write('END \n\n')

# Contingencies for removing each specified switched shunt
for busnum2 in busNumsSw2:
    conFile.write("CONTINGENCY 'SwB%d'\n" %busnum2)
    conFile.write('REMOVE SWSHUNT FROM BUS %d\n' %busnum2)
    conFile.write('END\n\n')

conFile.write('END')
conFile.close()

# Run Multi-level AC Contingency Solution
psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
psspy.dfax([1,1], subSysLabel2, subSysLabel2, conFileName2 + '.con',
           conFileName2)

psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                   [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                   [subSysLabel + '_DSP',subSysLabel + '_HV',
                    subSysLabel + '_HV',subSysLabel + '_HV',
                    subSysLabel + '_HV',
                    subSysLabel + '_BULK',subSysLabel + '_BULK'],
                  conFileName2 + '.dfx',
                  conFileName2 + '_FXSHUNTTRIP_' + 'FxS%d-BUS%d'
                  %( int(FxShuntid), busnum),
                  "","",subSysLabel + '.trp')

# Generate final excel file with results
```

```
    pssexcel.accc(conFileName2 + '_FXSHUNTTRIP_' + 'FxS%d-BUS%d' %( int(FxShuntid), busnum) +
".acc",
                  ['e','b','v'],colabel='',stype='contingency',
                  busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
                  xlsfile = conFileName2 + '_FXSHUNTTRIP_' + 'FxS%d-BUS%d' %( int(FxShuntid),
busnum) + '.xls',
                  sheet='',overwritesheet=True,show=False)

    # Generate final text file with results
    psspy.report_output(islct=2, filarg="./" + conFileName2 + '_FXSHUNTTRIP_' + 'FxS%d-BUS%d' %(
int(FxShuntid), busnum) + '.txt',
                        options=[0])
    psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                                   [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                                   conFileName2 + '_FXSHUNTTRIP_' + 'FxS%d-BUS%d'
                                   %( int(FxShuntid), busnum))
    psspy.report_output(islct=2, options=[0])


    """
    For Loop Part to perform all P6_I3_E4 contingencies.
    """
    psspy.case(CASE2)
    basekv2=[1,999]
    psspy.bsys(sid2, usekv2, basekv2, numarea2, areas2)

    # CON file name with contingencies
    conFileName2 = "P6_I3_E4"

    # Returns a list with DC Line names for only in-service
    # two-terminal dc lines
    ierr, (allDCBrnNames2,) = psspy.a2trmdcchar(subsysID2, 3, 1, 'DCNAME')
    print "DCBrnID =", allDCBrnNames2

    # Returns a list with From or Rectifier Bus Numbers for only in-service
    # two-terminal dc lines
    ierr, (FrombusNums2,) = psspy.a2trmdcint(subsysID2, 3, 1, 'FROMNUMBER')
    print "list of FROM bus numbers DCBrn ", FrombusNums2

    # Returns a list with To or Inverter Bus Numbers for only in-service
    # two-terminal dc lines
    ierr, (TobusNums2,) = psspy.a2trmdcint(subsysID2, 3, 1, 'TONUMBER')
    print "list of TO bus numbers DCBrn ", TobusNums2

    # Opens a file object for both writing and reading. Overwrites the
    # existing file if the file exists. If the file does not exist, creates
    # a new file for reading and writing.
    conFile = open(conFileName2 + '.con', 'w+')

    # .con file header
    conFile.write("COM PSS(R)E 32\n")
    conFile.write("COM\n")
    conFile.write("COM LOSS OF SINGLE POLE OF A DC LINE IN SUBSYSTEM '%s' \n\n"
                  % subSysLabel)

    # Contingencies for removing each specified Pole of a DC Line
    for DCBrnname2, frombusnum2, tobusnum2 in zip(allDCBrnNames2, FrombusNums2, TobusNums2):
        conFile.write("CONTINGENCY '%d_%d_%s'\n"
                      %(frombusnum2, tobusnum2, DCBrnname2))
        conFile.write('BLOCK TWOTERMDC %s\n' %DCBrnname2)
        conFile.write('END \n\n')

    conFile.write('END')
    conFile.close()

    # Run Multi-level AC Contingency Solution
    psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
    psspy.dfax([1,1], subSysLabel2, subSysLabel2, conFileName2 + '.con',
               conFileName2)
```

```
    psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                         [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                         [subSysLabel + '_DSP',subSysLabel + '_HV',
                          subSysLabel + '_HV',subSysLabel + '_HV',
                          subSysLabel + '_HV',
                          subSysLabel + '_BULK',subSysLabel + '_BULK'],
                        conFileName2 + '.dfx',
                        conFileName2 + '_FXSHUNTTRIP_' + 'FxS%d-BUS%d'
                        %( int(FxShuntid), busnum),
                        "","",subSysLabel + '.trp')

    # Generate final excel file with results
    pssexcel.accc(conFileName2 + '_FXSHUNTTRIP_' + 'FxS%d-BUS%d' %( int(FxShuntid), busnum) +
".acc",
                  ['e','b','v'],colabel='',stype='contingency',
                  busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
                  xlsfile = conFileName2 + '_FXSHUNTTRIP_' + 'FxS%d-BUS%d' %( int(FxShuntid),
busnum) + '.xls',
                  sheet='',overwritesheet=True,show=False)

    # Generate final text file with results
    psspy.report_output(islct=2, filarg="./" + conFileName2 + '_FXSHUNTTRIP_' + 'FxS%d-BUS%d' %(
int(FxShuntid), busnum) + '.txt',
                        options=[0])
    psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                                   [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                                   conFileName2 + '_FXSHUNTTRIP_' + 'FxS%d-BUS%d'
                                   %( int(FxShuntid), busnum))
    psspy.report_output(islct=2, options=[0])


# Loop to lose each Switched Shunt Device, perform system adjustments,
# and study the respective events
for busnum in busNumsSw:
    """
    For Loop Part to create a temporary contingency file for each Switched Shunt
    Device lost in ORIGINAL case file
    """
    psspy.case(CASE)
    psspy.prompt_output(4,"",[0,0])
    psspy.alert_output(4,"",[0,0])

    # Opens a file object for both writing and reading. Overwrites the
    # existing file if the file exists. If the file does not exist, creates
    # a new file for reading and writing.
    conFile = open(conFileName1 + '.con', 'w+')

    # .con file header
    conFile.write("COM PSS(R)E 32\n")
    conFile.write("COM\n")
    conFile.write("COM LOSS OF SINGLE SHUNT DEVICE IN SUBSYSTEM '%s' \n\n"
                  % subSysLabel)

    # Contingency for removing each specified Switched Shunt, one at a time.
    conFile.write("CONTINGENCY 'SwB%d'\n" %busnum)
    conFile.write('REMOVE SWSHUNT FROM BUS %d\n' %busnum)
    conFile.write('END \n\n')
    conFile.write('END')
    conFile.close()

    psspy.dfax([1,1], subSysLabel , subSysLabel , conFileName1 + '.con',
               conFileName1)

    # System adjustment: Units Redispatch while Imposing Contingency
    psspy.gendsp([2,0],[subSysLabel + '_DSP','SwB%d'
                        %busnum],
                 conFileName1 + '.dfx',"","")

    # The next piece of code is to avoid large islands in the solution
    ierr,buses = psspy.tree(1,0)
```

```
while buses > 0:
    ierr,buses = psspy.tree(2,1)

# After redispatching, power flow Solution is solved a few times.
# Further System adjustments: enable stepping tap, phase shift,
# dc tap, and switched shunt adjustments.
psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
psspy.fdns([1,2,1,1,1,0,99,0])
psspy.fdns([1,2,1,1,1,0,99,0])
psspy.fdns([1,2,1,1,1,0,99,0])
psspy.fdns([1,2,1,1,1,0,99,0])
psspy.fdns([1,2,1,1,1,0,99,0])

# The new solved case is saved. This will be the Initial Condition
psspy.save(CASE1 + '_SWSHUNTTRIP_' + 'SwS-BUS%d'
           %busnum + '.sav')

# New working case to be used with the reamining contingecy events,
# E1 through E3.
CASE2 = CASE1 + '_SWSHUNTTRIP_' + 'SwS-BUS%d' %busnum + '.sav'

psspy.case(CASE2)
psspy.prompt_output(4,"",[0,0])
psspy.alert_output(4,"",[0,0])

ierr, bus, mismatch = psspy.maxmsm()
print bus

mva = psspy.sysmsm()
if mva > 0.7:
    continue

# Defining a subsystem with buses in study area 1
sid2 = sid
usekv2 = usekv
areas2 = areas
numarea2 = len(areas2)
subsysID2 = sid2
subSysLabel2 = subSysLabel


"""
For Loop Part to perform all P6_I3_E1 contingencies.
"""
psspy.case(CASE2)
basekv2=[100,999]
psspy.bsys(sid2, usekv2, basekv2, numarea2, areas2)

# Temporary CON file name with contingencies
conFileName2 = "P6_I3_E1"

# Returns a list with Circuit Identifiers for only in-service
# non-transformer branches
ierr, (allBrnID2,) = psspy.abrnchar(subsysID2, 1, 3, 1, 1, 'ID')
print "BrnID =", allBrnID2

# Returns a list with From Bus Numbers for only in-service
# non-transformer branches
ierr, (FrombusNums2,) = psspy.abrnint(subsysID2, 1, 3, 1, 1, 'FROMNUMBER')
print "list of FROM bus numbers Brn ", FrombusNums2

# Returns a list with To Bus Numbers for only in-service
# non-transformer branches
ierr, (TobusNums2,) = psspy.abrnint(subsysID2, 1, 3, 1, 1, 'TONUMBER')
print "list of TO bus numbers Brn ", TobusNums2

# Opens a file object for both writing and reading. Overwrites the
# existing file if the file exists. If the file does not exist, creates
# a new file for reading and writing.
conFile = open(conFileName2 + '.con', 'w+')
```

```
    # .con file header
    conFile.write("COM PSS(R)E 32\n")
    conFile.write("COM\n")
    conFile.write("COM LOSS OF SINGLE TRANSMISSION LINE IN SUBSYSTEM '%s' \n\n"
                  % subSysLabel2)

    # Contingencies for removing each specified Transmission Circuit
    for Brnid2, frombusnum2, tobusnum2 in zip(allBrnID2, FrombusNums2, TobusNums2):
        conFile.write("CONTINGENCY '%d_%d_%d'\n"
                      %(frombusnum2, tobusnum2, int(Brnid2)))
        conFile.write('DISCONNECT BRANCH FROM BUS %d TO BUS %d CIRCUIT %d\n'
                      %(frombusnum2, tobusnum2, int(Brnid2)))
        conFile.write('END \n\n')

    conFile.write('END')
    conFile.close()

    # Run Multi-level AC Contingency Solution
    psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
    psspy.dfax([1,1], subSysLabel2, subSysLabel2, conFileName2 + '.con',
               conFileName2)

    psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                       [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                       [subSysLabel + '_DSP',subSysLabel + '_HV',
                        subSysLabel + '_HV',subSysLabel + '_HV',
                        subSysLabel + '_HV',
                        subSysLabel + '_BULK',subSysLabel + '_BULK'],
                      conFileName2 + '.dfx',
                      conFileName2 + '_SWSHUNTTRIP_' + 'SwS-BUS%d'
                      %busnum,
                      "","",subSysLabel + '.trp')


    # Generate final excel file with results
    import pssexcel

    ACCFile=sys.path[0]

    pssexcel.accc(conFileName2 + '_SWSHUNTTRIP_' + 'SwS-BUS%d' %busnum + ".acc",
                  ['e','b','v'],colabel='',stype='contingency',
                  busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
                  xlsfile = conFileName2 + '_SWSHUNTTRIP_' + 'SwS-BUS%d' %busnum + '.xls',
                  sheet='',overwritesheet=True,show=False)

    # Generate final text file with results
    psspy.report_output(islct=2, filarg="./" + conFileName2 + '_SWSHUNTTRIP_' + 'SwS-BUS%d'
%busnum + '.txt',
                        options=[0])
    psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                                   [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                                   conFileName2 + '_SWSHUNTTRIP_' + 'SwS-BUS%d'
                                   %busnum)
    psspy.report_output(islct=2, options=[0])


    """
    For Loop Part to perform all P6_I3_E2 contingencies.
    """
    psspy.case(CASE2)
    basekv2=[100,999]
    psspy.bsys(sid2, usekv2, basekv2, numarea2, areas2)

    # Temporary CON file name with contingencies
    conFileName2 = "P6_I3_E2"

    # Returns a list with Circuit Identifiers for only in-service
    # two-winding transformers branches
    ierr, (allTrnID2,) = psspy.atrnchar(subsysID2, 1, 3, 1, 1, 'ID')
    print "TrnID =", allTrnID2
```

```
    # Returns a list with From Bus Numbers for only in-service
    # two-winding transformers branches
    ierr, (FrombusNums2,) = psspy.atrnint(subsysID2, 1, 3, 1, 1, 'FROMNUMBER')
    print "list of FROM bus numbers Trn ", FrombusNums2

    # Returns a list with To Bus Numbers for only in-service
    # two-winding transformers branches
    ierr, (TobusNums2,) = psspy.atrnint(subsysID2, 1, 3, 1, 1, 'TONUMBER')
    print "list of TO bus numbers Trn ", TobusNums2

    # Opens a file object for both writing and reading. Overwrites the
    # existing file if the file exists. If the file does not exist, creates
    # a new file for reading and writing.
    conFile = open(conFileName2 + '.con', 'w+')

    # .con file header
    conFile.write("COM PSS(R)E 32\n")
    conFile.write("COM\n")
    conFile.write("COM LOSS OF SINGLE TRANSFORMER IN SUBSYSTEM '%s' \n\n"
                  % subSysLabel2)

    # Contingencies for removing each specified Transformer
    for Trnid2, frombusnum2, tobusnum2 in zip(allTrnID2, FrombusNums2, TobusNums2):
        conFile.write("CONTINGENCY '%d_%d_%d'\n"
                      %(frombusnum2, tobusnum2, int(Trnid2)))
        conFile.write('DISCONNECT BRANCH FROM BUS %d TO BUS %d CIRCUIT %d\n'
                      %(frombusnum2, tobusnum2, int(Trnid2)))
        conFile.write('END \n\n')

    conFile.write('END')
    conFile.close()

    # Run Multi-level AC Contingency Solution
    psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
    psspy.dfax([1,1], subSysLabel2, subSysLabel2, conFileName2 + '.con',
               conFileName2)

    psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                       [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                       [subSysLabel + '_DSP',subSysLabel + '_HV',
                        subSysLabel + '_HV',subSysLabel + '_HV',
                        subSysLabel + '_HV',
                        subSysLabel + '_BULK',subSysLabel + '_BULK'],
                      conFileName2 + '.dfx',
                      conFileName2 + '_SWSHUNTTRIP_' + 'SwS-BUS%d'
                      %busnum,
                      "","",subSysLabel + '.trp')


    # Generate final excel file with results
    pssexcel.accc(conFileName2 + '_SWSHUNTTRIP_' + 'SwS-BUS%d' %busnum + ".acc",
                  ['e','b','v'],colabel='',stype='contingency',
                  busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
                  xlsfile = conFileName2 + '_SWSHUNTTRIP_' + 'SwS-BUS%d' %busnum + '.xls',
                  sheet='',overwritesheet=True,show=False)

    # Generate final text file with results
    psspy.report_output(islct=2, filarg="./" + conFileName2 + '_SWSHUNTTRIP_' + 'SwS-BUS%d'
%busnum + '.txt',
                        options=[0])
    psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                                   [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                                   conFileName2 + '_SWSHUNTTRIP_' + 'SwS-BUS%d'
                                   %busnum)
    psspy.report_output(islct=2, options=[0])


    """
    For Loop Part to perform all P6_I3_E3 contingencies.
    """
    psspy.case(CASE2)
```

```
basekv2=[100,999]
psspy.bsys(sid2, usekv2, basekv2, numarea2, areas2)

# Temporary CON file name with contingencies
conFileName2 = "P6_I3_E3"

### Part for FIXED Shunts Starts ###
# Returns a list with Shunt Identifiers for only in-service
# fixed shunts at in-service buses
ierr, (allFxShuntID2,) = psspy.afxshuntchar(subsysID2, 1, 'ID')
print "FxShuntID =", allFxShuntID2

# Returns a list with Bus Numbers for only in-service
# fixed shunts at in-service buses
ierr, (busNums2,) = psspy.afxshuntint(subsysID2, 1, 'NUMBER')
print "list of bus numbers Fixed Shunts ", busNums2

### Part for SWITCHED Shunts Starts ###
# Returns a list with Bus Numbers for only in-service
# switched shunts at in-service buses
ierr, (busNumsSw2,) = psspy.aswshint(sid2, 1, 'NUMBER')
print "list of bus numbers Switched Shunts ", busNumsSw2

# Opens a file object for both writing and reading. Overwrites the
# existing file if the file exists. If the file does not exist, creates
# a new file for reading and writing.
conFile = open(conFileName2 + '.con', 'w+')

# .con file header
conFile.write("COM PSS(R)E 32\n")
conFile.write("COM\n")
conFile.write("COM LOSS OF SINGLE SHUNT DEVICE IN SUBSYSTEM '%s' \n\n"
              % subSysLabel2)

# Contingencies for removing each specified fixed shunt
for FxShuntid2, busnum2 in zip(allFxShuntID2, busNums2):
    conFile.write("CONTINGENCY 'F%dB%d'\n"
                  %( int(FxShuntid2), busnum2))
    conFile.write('REMOVE SHUNT %s FROM BUS %d\n' %(FxShuntid2, busnum2))
    conFile.write('END \n\n')

# Contingencies for removing each specified switched shunt
for busnum2 in busNumsSw2:
    conFile.write("CONTINGENCY 'SwB%d'\n" %busnum2)
    conFile.write('REMOVE SWSHUNT FROM BUS %d\n' %busnum2)
    conFile.write('END\n\n')

conFile.write('END')
conFile.close()

# Run Multi-level AC Contingency Solution
psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
psspy.dfax([1,1], subSysLabel2, subSysLabel2, conFileName2 + '.con',
           conFileName2)

psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                    [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                    [subSysLabel + '_DSP',subSysLabel + '_HV',
                     subSysLabel + '_HV',subSysLabel + '_HV',
                     subSysLabel + '_HV',
                     subSysLabel + '_BULK',subSysLabel + '_BULK'],
                   conFileName2 + '.dfx',
                   conFileName2 + '_SWSHUNTTRIP_' + 'SwS-BUS%d'
                   %busnum,
                   "","",subSysLabel + '.trp')

# Generate final excel file with results

pssexcel.accc(conFileName2 + '_SWSHUNTTRIP_' + 'SwS-BUS%d' %busnum + ".acc",
              ['e','b','v'],colabel='',stype='contingency',
              busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
```

```
                xlsfile = conFileName2 + '_SWSHUNTTRIP_' + 'SwS-BUS%d' %busnum + '.xls',
                sheet='',overwritesheet=True,show=False)

    # Generate final text file with results
    psspy.report_output(islct=2, filarg="./" + conFileName2 + '_SWSHUNTTRIP_' + 'SwS-BUS%d'
%busnum + '.txt',
                        options=[0])
    psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                                   [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                                   conFileName2 + '_SWSHUNTTRIP_' + 'SwS-BUS%d'
                                   %busnum)
    psspy.report_output(islct=2, options=[0])


    """
    For Loop Part to perform all P6_I3_E4 contingencies.
    """
    psspy.case(CASE2)
    basekv2=[1,999]
    psspy.bsys(sid2, usekv2, basekv2, numarea2, areas2)

    # CON file name with contingencies
    conFileName2 = "P6_I3_E4"

    # Returns a list with DC Line names for only in-service
    # two-terminal dc lines
    ierr, (allDCBrnNames2,) = psspy.a2trmdcchar(subsysID2, 3, 1, 'DCNAME')
    print "DCBrnID =", allDCBrnNames2

    # Returns a list with From or Rectifier Bus Numbers for only in-service
    # two-terminal dc lines
    ierr, (FrombusNums2,) = psspy.a2trmdcint(subsysID2, 3, 1, 'FROMNUMBER')
    print "list of FROM bus numbers DCBrn ", FrombusNums2

    # Returns a list with To or Inverter Bus Numbers for only in-service
    # two-terminal dc lines
    ierr, (TobusNums2,) = psspy.a2trmdcint(subsysID2, 3, 1, 'TONUMBER')
    print "list of TO bus numbers DCBrn ", TobusNums2

    # Opens a file object for both writing and reading. Overwrites the
    # existing file if the file exists. If the file does not exist, creates
    # a new file for reading and writing.
    conFile = open(conFileName2 + '.con', 'w+')

    # .con file header
    conFile.write("COM PSS(R)E 32\n")
    conFile.write("COM\n")
    conFile.write("COM LOSS OF SINGLE POLE OF A DC LINE IN SUBSYSTEM '%s' \n\n"
                  % subSysLabel)

    # Contingencies for removing each specified Pole of a DC Line
    for DCBrnname2, frombusnum2, tobusnum2 in zip(allDCBrnNames2, FrombusNums2, TobusNums2):
        conFile.write("CONTINGENCY '%d_%d_%s'\n"
                      %(frombusnum2, tobusnum2, DCBrnname2))
        conFile.write('BLOCK TWOTERMDC %s\n' %DCBrnname2)
        conFile.write('END \n\n')

    conFile.write('END')
    conFile.close()

    # Run Multi-level AC Contingency Solution
    psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
    psspy.dfax([1,1], subSysLabel2, subSysLabel2, conFileName2 + '.con',
               conFileName2)

    psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                       [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                       [subSysLabel + '_DSP',subSysLabel + '_HV',
                        subSysLabel + '_HV',subSysLabel + '_HV',
                        subSysLabel + '_HV',
                        subSysLabel + '_BULK',subSysLabel + '_BULK'],
```

```
                    conFileName2 + '.dfx',
                    conFileName2 + '_SWSHUNTTRIP_' + 'SwS-BUS%d'
                    %busnum,
                    "","",subSysLabel + '.trp')

    # Generate final excel file with results
    pssexcel.accc(conFileName2 + '_SWSHUNTTRIP_' + 'SwS-BUS%d' %busnum + ".acc",
                  ['e','b','v'],colabel='',stype='contingency',
                  busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
                  xlsfile = conFileName2 + '_SWSHUNTTRIP_' + 'SwS-BUS%d' %busnum + '.xls',
                  sheet='',overwritesheet=True,show=False)

    # Generate final text file with results
    psspy.report_output(islct=2, filarg="./" + conFileName2 + '_SWSHUNTTRIP_' + 'SwS-BUS%d'
%busnum + '.txt',
                        options=[0])
    psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                                   [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                                   conFileName2 + '_SWSHUNTTRIP_' + 'SwS-BUS%d'
                                   %busnum)
    psspy.report_output(islct=2, options=[0])


print('\nP6_I3 Category Complete!')
print('NOTE: It is highly recommended to create a new folder where')
print('      to move all the files created by this program.\n')
```

A.8 Category P6_I4

```
"""
THIS PROGRAM PERFORMS P6 CONTINGENCIES PER TPL-001-4
INITIAL CONDITION 4 WITH EVENTS 1 THROUGH 4
"""

"""
First Part is to calculate Initial Condition I4.
That is, loss of a Single pole of a DC Line followed by
System adjustments
"""

import os,sys

PSSE_LOCATION = r"C:\Program Files\PTI\PSSE32\PSSBIN"
sys.path.append(PSSE_LOCATION)
os.environ['PATH'] = os.environ['PATH'] + ';' +  PSSE_LOCATION

import psspy
import redirect
redirect.psse2py()

from sys import argv
if len(sys.argv) > 1:
    script, CASE1, areas, subSysLabel = argv
else:
    CASE1 = raw_input("Base case name, without *.sav extension: ")
    areas = raw_input("Area number where contingencies will take place: ")
    subSysLabel = raw_input("Area name where contingencies will take place: ")

areas = [int(areas)]


#---------------------------------
# PSS/E Saved case

CASE = CASE1 + '.sav'
```

```
psspy.psseinit(50000)
psspy.prompt_output(4,"",[0,0])
psspy.alert_output(4,"",[0,0])

# Defining a subsystem with buses in study area
sid = 0
usekv = 1
numarea = len(areas)
subsysID = sid

psspy.case(CASE)
basekv=[1,999]
psspy.bsys(sid, usekv, basekv, numarea, areas)

conFileName1 = "P6_P1_E5"

# Returns a list with DC Line names for only in-service
# two-terminal dc lines
ierr, (allDCBrnNames,) = psspy.a2trmdcchar(subsysID, 3, 1, 'DCNAME')
print "DCBrnID =", allDCBrnNames

# Returns a list with From or Rectifier Bus Numbers for only in-service
# two-terminal dc lines
ierr, (FrombusNums,) = psspy.a2trmdcint(subsysID, 3, 1, 'FROMNUMBER')
print "list of FROM bus numbers DCBrn ", FrombusNums

# Returns a list with To or Inverter Bus Numbers for only in-service
# two-terminal dc lines
ierr, (TobusNums,) = psspy.a2trmdcint(subsysID, 3, 1, 'TONUMBER')
print "list of TO bus numbers DCBrn ", TobusNums

# Loop to lose each Single pole of a DC Line, perform system adjustments,
# and study the respective events
for DCBrnname, frombusnum, tobusnum in zip(allDCBrnNames, FrombusNums, TobusNums):
    """
    For Loop Part to create a temporary contingency file for each Single
    pole of a DC Line lost in ORIGINAL case file
    """
    psspy.case(CASE)
    psspy.prompt_output(4,"",[0,0])
    psspy.alert_output(4,"",[0,0])

    # Opens a file object for both writing and reading. Overwrites the
    # existing file if the file exists. If the file does not exist, creates
    # a new file for reading and writing.
    conFile = open(conFileName1 + '.con', 'w+')

    # .con file header
    conFile.write("COM PSS(R)E 32\n")
    conFile.write("COM\n")
    conFile.write("COM LOSS OF SINGLE POLE OF A DC LINE IN SUBSYSTEM '%s' \n\n"
                  % subSysLabel)

    # Contingency for removing each specified Single pole of a DC Line, one at a time.
    conFile.write("CONTINGENCY '%d_%d_%s'\n"
                  %(frombusnum, tobusnum, DCBrnname))
    conFile.write('BLOCK TWOTERMDC %s\n' %DCBrnname)
    conFile.write('END \n\n')
    conFile.write('END')
    conFile.close()

    psspy.dfax([1,1], subSysLabel , subSysLabel , conFileName1 + '.con',
               conFileName1)

    # System adjustment: Units Redispatch while Imposing Contingency
    psspy.gendsp([2,0],[subSysLabel + '_DSP','%d_%d_%s'
                        %(frombusnum, tobusnum, DCBrnname)],
                 conFileName1 + '.dfx',"","")

    # The next piece of code is to avoid large islands in the solution
    ierr,buses = psspy.tree(1,0)
```

```
while buses > 0:
    ierr,buses = psspy.tree(2,1)

# After redispatching, power flow Solution is solved a few times.
# Further System adjustments: enable stepping tap, phase shift,
# dc tap, and switched shunt adjustments.
psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
psspy.fdns([1,2,1,1,1,0,99,0])
psspy.fdns([1,2,1,1,1,0,99,0])
psspy.fdns([1,2,1,1,1,0,99,0])
psspy.fdns([1,2,1,1,1,0,99,0])
psspy.fdns([1,2,1,1,1,0,99,0])

# The new solved case is saved. This will be the Initial Condition
psspy.save(CASE1 + '_SPDCLINTRIP_' + 'L%d_%d_%s'
          %(frombusnum, tobusnum, " ".join(DCBrnname.split())) + '.sav')

# New working case to be used with the reamining contingecy events,
# E1 through E3.
CASE2 = CASE1 + '_SPDCLINTRIP_' + 'L%d_%d_%s' %(frombusnum, tobusnum, DCBrnname) + '.sav'

psspy.case(CASE2)
psspy.prompt_output(4,"",[0,0])
psspy.alert_output(4,"",[0,0])

ierr, bus, mismatch = psspy.maxmsm()
print bus

mva = psspy.sysmsm()
if mva > 0.7:
    continue

# Defining a subsystem with buses in study area 1
sid2 = sid
usekv2 = usekv
areas2 = areas
numarea2 = len(areas2)
subsysID2 = sid2
subSysLabel2 = subSysLabel


"""
For Loop Part to perform all P6_I4_E1 contingencies.
"""
psspy.case(CASE2)
basekv2=[100,999]
psspy.bsys(sid2, usekv2, basekv2, numarea2, areas2)

# Temporary CON file name with contingencies
conFileName2 = "P6_I4_E1"

# Returns a list with Circuit Identifiers for only in-service
# non-transformer branches
ierr, (allBrnID2,) = psspy.abrnchar(subsysID2, 1, 3, 1, 1, 'ID')
print "BrnID =", allBrnID2

# Returns a list with From Bus Numbers for only in-service
# non-transformer branches
ierr, (FrombusNums2,) = psspy.abrnint(subsysID2, 1, 3, 1, 1, 'FROMNUMBER')
print "list of FROM bus numbers Brn ", FrombusNums2

# Returns a list with To Bus Numbers for only in-service
# non-transformer branches
ierr, (TobusNums2,) = psspy.abrnint(subsysID2, 1, 3, 1, 1, 'TONUMBER')
print "list of TO bus numbers Brn ", TobusNums2

# Opens a file object for both writing and reading. Overwrites the
# existing file if the file exists. If the file does not exist, creates
# a new file for reading and writing.
conFile = open(conFileName2 + '.con', 'w+')
```

```
    # .con file header
    conFile.write("COM PSS(R)E 32\n")
    conFile.write("COM\n")
    conFile.write("COM LOSS OF SINGLE TRANSMISSION LINE IN SUBSYSTEM '%s' \n\n"
                  % subSysLabel2)

    # Contingencies for removing each specified Transmission Circuit
    for Brnid2, frombusnum2, tobusnum2 in zip(allBrnID2, FrombusNums2, TobusNums2):
        conFile.write("CONTINGENCY '%d_%d_%d'\n"
                      %(frombusnum2, tobusnum2, int(Brnid2)))
        conFile.write('DISCONNECT BRANCH FROM BUS %d TO BUS %d CIRCUIT %d\n'
                      %(frombusnum2, tobusnum2, int(Brnid2)))
        conFile.write('END \n\n')

    conFile.write('END')
    conFile.close()

    # Run Multi-level AC Contingency Solution
    psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
    psspy.dfax([1,1], subSysLabel2, subSysLabel2, conFileName2 + '.con',
               conFileName2)

    psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                       [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                       [subSysLabel + '_DSP',subSysLabel + '_HV',
                        subSysLabel + '_HV',subSysLabel + '_HV',
                        subSysLabel + '_HV',
                        subSysLabel + '_BULK',subSysLabel + '_BULK'],
                      conFileName2 + '.dfx',
                      conFileName2 + '_SPDCLINTRIP_' + 'L%d_%d_%s'
                      %(frombusnum, tobusnum, DCBrnname),
                      "","",subSysLabel + '.trp')


    # Generate final excel file with results
    import pssexcel

    ACCFile=sys.path[0]

    pssexcel.accc(conFileName2 + '_SPDCLINTRIP_' + 'L%d_%d_%s' %(frombusnum, tobusnum, "
".join(DCBrnname.split())) + ".acc",
                  ['e','b','v'],colabel='',stype='contingency',
                  busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
                  xlsfile = conFileName2 + '_SPDCLINTRIP_' + 'L%d_%d_%s' %(frombusnum, tobusnum,
" ".join(DCBrnname.split())) + '.xls',
                  sheet='',overwritesheet=True,show=False)

    # Generate final text file with results
    psspy.report_output(islct=2, filarg="./" + conFileName2 + '_SPDCLINTRIP_' + 'L%d_%d_%s'
%(frombusnum, tobusnum, " ".join(DCBrnname.split())) + '.txt',
                        options=[0])
    psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                                   [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                                   conFileName2 + '_SPDCLINTRIP_' + 'L%d_%d_%s'
                                   %(frombusnum, tobusnum, " ".join(DCBrnname.split())))
    psspy.report_output(islct=2, options=[0])


    """
    For Loop Part to perform all P6_I4_E2 contingencies.
    """
    psspy.case(CASE2)
    basekv2=[100,999]
    psspy.bsys(sid2, usekv2, basekv2, numarea2, areas2)

    # Temporary CON file name with contingencies
    conFileName2 = "P6_I4_E2"

    # Returns a list with Circuit Identifiers for only in-service
    # two-winding transformers branches
```

```
    ierr, (allTrnID2,) = psspy.atrnchar(subsysID2, 1, 3, 1, 1, 'ID')
    print "TrnID =", allTrnID2

    # Returns a list with From Bus Numbers for only in-service
    # two-winding transformers branches
    ierr, (FrombusNums2,) = psspy.atrnint(subsysID2, 1, 3, 1, 1, 'FROMNUMBER')
    print "list of FROM bus numbers Trn ", FrombusNums2

    # Returns a list with To Bus Numbers for only in-service
    # two-winding transformers branches
    ierr, (TobusNums2,) = psspy.atrnint(subsysID2, 1, 3, 1, 1, 'TONUMBER')
    print "list of TO bus numbers Trn ", TobusNums2

    # Opens a file object for both writing and reading. Overwrites the
    # existing file if the file exists. If the file does not exist, creates
    # a new file for reading and writing.
    conFile = open(conFileName2 + '.con', 'w+')

    # .con file header
    conFile.write("COM PSS(R)E 32\n")
    conFile.write("COM\n")
    conFile.write("COM LOSS OF SINGLE TRANSFORMER IN SUBSYSTEM '%s' \n\n"
                  % subSysLabel2)

    # Contingencies for removing each specified Transformer
    for Trnid2, frombusnum2, tobusnum2 in zip(allTrnID2, FrombusNums2, TobusNums2):
        conFile.write("CONTINGENCY '%d_%d_%d'\n"
                      %(frombusnum2, tobusnum2, int(Trnid2)))
        conFile.write('DISCONNECT BRANCH FROM BUS %d TO BUS %d CIRCUIT %d\n'
                      %(frombusnum2, tobusnum2, int(Trnid2)))
        conFile.write('END \n\n')

    conFile.write('END')
    conFile.close()

    # Run Multi-level AC Contingency Solution
    psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
    psspy.dfax([1,1], subSysLabel2, subSysLabel2, conFileName2 + '.con',
               conFileName2)

    psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                        [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                        [subSysLabel + '_DSP',subSysLabel + '_HV',
                         subSysLabel + '_HV',subSysLabel + '_HV',
                         subSysLabel + '_HV',
                         subSysLabel + '_BULK',subSysLabel + '_BULK'],
                       conFileName2 + '.dfx',
                       conFileName2 + '_SPDCLINTRIP_' + 'L%d_%d_%s'
                       %(frombusnum, tobusnum, DCBrnname),
                       "","",subSysLabel + '.trp')


    # Generate final excel file with results

    pssexcel.accc(conFileName2 + '_SPDCLINTRIP_' + 'L%d_%d_%s' %(frombusnum, tobusnum, "
".join(DCBrnname.split())) + ".acc",
                  ['e','b','v'],colabel='',stype='contingency',
                  busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
                  xlsfile = conFileName2 + '_SPDCLINTRIP_' + 'L%d_%d_%s' %(frombusnum, tobusnum,
" ".join(DCBrnname.split())) + '.xls',
                  sheet='',overwritesheet=True,show=False)

    # Generate final text file with results
    psspy.report_output(islct=2, filarg="./" + conFileName2 + '_SPDCLINTRIP_' + 'L%d_%d_%s'
%(frombusnum, tobusnum, " ".join(DCBrnname.split())) + '.txt',
                        options=[0])
    psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                                   [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                                   conFileName2 + '_SPDCLINTRIP_' + 'L%d_%d_%s'
                                   %(frombusnum, tobusnum, " ".join(DCBrnname.split())))
    psspy.report_output(islct=2, options=[0])
```

```
"""
For Loop Part to perform all P6_I4_E3 contingencies.
"""
psspy.case(CASE2)
basekv2=[100,999]
psspy.bsys(sid2, usekv2, basekv2, numarea2, areas2)

# Temporary CON file name with contingencies
conFileName2 = "P6_I4_E3"

### Part for FIXED Shunts Starts ###
# Returns a list with Shunt Identifiers for only in-service
# fixed shunts at in-service buses
ierr, (allFxShuntID2,) = psspy.afxshuntchar(subsysID2, 1, 'ID')
print "FxShuntID =", allFxShuntID2

# Returns a list with Bus Numbers for only in-service
# fixed shunts at in-service buses
ierr, (busNums2,) = psspy.afxshuntint(subsysID2, 1, 'NUMBER')
print "list of bus numbers Fixed Shunts ", busNums2

### Part for SWITCHED Shunts Starts ###
# Returns a list with Bus Numbers for only in-service
# switched shunts at in-service buses
ierr, (busNumsSw2,) = psspy.aswshint(sid2, 1, 'NUMBER')
print "list of bus numbers Switched Shunts ", busNumsSw2

# Opens a file object for both writing and reading. Overwrites the
# existing file if the file exists. If the file does not exist, creates
# a new file for reading and writing.
conFile = open(conFileName2 + '.con', 'w+')

# .con file header
conFile.write("COM PSS(R)E 32\n")
conFile.write("COM\n")
conFile.write("COM LOSS OF SINGLE SHUNT DEVICE IN SUBSYSTEM '%s' \n\n"
              % subSysLabel2)

# Contingencies for removing each specified fixed shunt
for FxShuntid2, busnum2 in zip(allFxShuntID2, busNums2):
    conFile.write("CONTINGENCY 'F%dB%d'\n"
                  %( int(FxShuntid2), busnum2))
    conFile.write('REMOVE SHUNT %s FROM BUS %d\n' %(FxShuntid2, busnum2))
    conFile.write('END \n\n')

# Contingencies for removing each specified switched shunt
for busnum2 in busNumsSw2:
    conFile.write("CONTINGENCY 'SwB%d'\n" %busnum2)
    conFile.write('REMOVE SWSHUNT FROM BUS %d\n' %busnum2)
    conFile.write('END\n\n')

conFile.write('END')
conFile.close()

# Run Multi-level AC Contingency Solution
psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
psspy.dfax([1,1], subSysLabel2, subSysLabel2, conFileName2 + '.con',
           conFileName2)

psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                   [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                   [subSysLabel + '_DSP',subSysLabel + '_HV',
                    subSysLabel + '_HV',subSysLabel + '_HV',
                    subSysLabel + '_HV',
                    subSysLabel + '_BULK',subSysLabel + '_BULK'],
                  conFileName2 + '.dfx',
                  conFileName2 + '_SPDCLINTRIP_' + 'L%d_%d_%s'
                  %(frombusnum, tobusnum, DCBrnname),
                  "","",subSysLabel + '.trp')
```

```
    # Generate final excel file with results

    pssexcel.accc(conFileName2 + '_SPDCLINTRIP_' + 'L%d_%d_%s' %(frombusnum, tobusnum, "
".join(DCBrnname.split())) + ".acc",
                  ['e','b','v'],colabel='',stype='contingency',
                  busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
                  xlsfile = conFileName2 + '_SPDCLINTRIP_' + 'L%d_%d_%s' %(frombusnum, tobusnum,
" ".join(DCBrnname.split())) + '.xls',
                  sheet='',overwritesheet=True,show=False)

    # Generate final text file with results
    psspy.report_output(islct=2, filarg="./" + conFileName2 + '_SPDCLINTRIP_' + 'L%d_%d_%s'
%(frombusnum, tobusnum, " ".join(DCBrnname.split())) + '.txt',
                        options=[0])
    psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                                   [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                                   conFileName2 + '_SPDCLINTRIP_' + 'L%d_%d_%s'
                                   %(frombusnum, tobusnum, " ".join(DCBrnname.split())))
    psspy.report_output(islct=2, options=[0])


    """
    For Loop Part to perform all P6_I4_E4 contingencies.
    """
    psspy.case(CASE2)
    basekv2=[1,999]
    psspy.bsys(sid2, usekv2, basekv2, numarea2, areas2)

    # CON file name with contingencies
    conFileName2 = "P6_I4_E4"

    # Returns a list with DC Line names for only in-service
    # two-terminal dc lines
    ierr, (allDCBrnNames2,) = psspy.a2trmdcchar(subsysID2, 3, 1, 'DCNAME')
    print "DCBrnID =", allDCBrnNames2

    # Returns a list with From or Rectifier Bus Numbers for only in-service
    # two-terminal dc lines
    ierr, (FrombusNums2,) = psspy.a2trmdcint(subsysID2, 3, 1, 'FROMNUMBER')
    print "list of FROM bus numbers DCBrn ", FrombusNums2

    # Returns a list with To or Inverter Bus Numbers for only in-service
    # two-terminal dc lines
    ierr, (TobusNums2,) = psspy.a2trmdcint(subsysID2, 3, 1, 'TONUMBER')
    print "list of TO bus numbers DCBrn ", TobusNums2

    # Opens a file object for both writing and reading. Overwrites the
    # existing file if the file exists. If the file does not exist, creates
    # a new file for reading and writing.
    conFile = open(conFileName2 + '.con', 'w+')

    # .con file header
    conFile.write("COM PSS(R)E 32\n")
    conFile.write("COM\n")
    conFile.write("COM LOSS OF SINGLE POLE OF A DC LINE IN SUBSYSTEM '%s' \n\n"
                  % subSysLabel)

    # Contingencies for removing each specified Pole of a DC Line
    for DCBrnname2, frombusnum2, tobusnum2 in zip(allDCBrnNames2, FrombusNums2, TobusNums2):
        conFile.write("CONTINGENCY '%d_%d_%s'\n"
                      %(frombusnum2, tobusnum2, DCBrnname2))
        conFile.write('BLOCK TWOTERMDC %s\n' %DCBrnname2)
        conFile.write('END \n\n')

    conFile.write('END')
    conFile.close()

    # Run Multi-level AC Contingency Solution
    psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
    psspy.dfax([1,1], subSysLabel2, subSysLabel2, conFileName2 + '.con',
```

```
                conFileName2)

    psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                        [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                        [subSysLabel + '_DSP',subSysLabel + '_HV',
                         subSysLabel + '_HV',subSysLabel + '_HV',
                         subSysLabel + '_HV',
                         subSysLabel + '_BULK',subSysLabel + '_BULK'],
                       conFileName2 + '.dfx',
                       conFileName2 + '_SPDCLINTRIP_' + 'L%d_%d_%s'
                       %(frombusnum, tobusnum, DCBrnname),
                       "","",subSysLabel + '.trp')

    # Generate final excel file with results
    pssexcel.accc(conFileName2 + '_SPDCLINTRIP_' + 'L%d_%d_%s' %(frombusnum, tobusnum, "
".join(DCBrnname.split())) + ".acc",
                  ['e','b','v'],colabel='',stype='contingency',
                  busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
                  xlsfile = conFileName2 + '_SPDCLINTRIP_' + 'L%d_%d_%s' %(frombusnum, tobusnum,
" ".join(DCBrnname.split())) + '.xls',
                  sheet='',overwritesheet=True,show=False)

    # Generate final text file with results
    psspy.report_output(islct=2, filarg="./" + conFileName2 + '_SPDCLINTRIP_' + 'L%d_%d_%s'
%(frombusnum, tobusnum, " ".join(DCBrnname.split())) + '.txt',
                        options=[0])
    psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                                   [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                                   conFileName2 + '_SPDCLINTRIP_' + 'L%d_%d_%s'
                                   %(frombusnum, tobusnum, " ".join(DCBrnname.split())))
    psspy.report_output(islct=2, options=[0])


print('\nP6_I4 Category Complete!')
print('NOTE: It is highly recommended to create a new folder where')
print('      to move all the files created by this program.\n')
```

A.9 Category P7

```
"""
THIS PROGRAM PERFORMS P7 CONTINGENCIES PER TPL-001-4
EVENTS 1 AND 2
"""

import os,sys

PSSE_LOCATION = r"C:\Program Files\PTI\PSSE32\PSSBIN"
sys.path.append(PSSE_LOCATION)
os.environ['PATH'] = os.environ['PATH'] + ';' +  PSSE_LOCATION

import psspy
import redirect
redirect.psse2py()

from sys import argv
if len(sys.argv) > 1:
    script, CASE1, areas, subSysLabel = argv
else:
    CASE1 = raw_input("Base case name, without *.sav extension: ")
    areas = raw_input("Area number where contingencies will take place: ")
    subSysLabel = raw_input("Area name where contingencies will take place: ")

areas = [int(areas)]


#-------------------------------
```

```
# PSS/E Saved case

CASE = CASE1 + '.sav'

psspy.psseinit(50000)
psspy.prompt_output(4,"",[0,0])
psspy.alert_output(4,"",[0,0])

# Defining a subsystem with buses in study area
sid = 0
usekv = 1
numarea = len(areas)
subsysID = sid


"""
Program to perform all P7_E1 contingencies.
"""
psspy.case(CASE)
basekv=[100,999]
psspy.bsys(sid, usekv, basekv, numarea, areas)

# CON file name with contingencies
conFileName1 = "P7_E1"
conFileName = conFileName1 + '.con'

# Returns a list with Circuit Identifiers for only in-service
# non-transformer branches
ierr, (allBrnID,) = psspy.abrnchar(subsysID, 1, 3, 1, 1, 'ID')
print "BrnID =", allBrnID

# Returns a list with From Bus Numbers for only in-service
# non-transformer branches
ierr, (FrombusNums,) = psspy.abrnint(subsysID, 1, 3, 1, 1, 'FROMNUMBER')
print "list of FROM bus numbers Brn ", FrombusNums

# Returns a list with To Bus Numbers for only in-service
# non-transformer branches
ierr, (TobusNums,) = psspy.abrnint(subsysID, 1, 3, 1, 1, 'TONUMBER')
print "list of TO bus numbers Brn ", TobusNums

# Opens a file object for both writing and reading. Overwrites the
# existing file if the file exists. If the file does not exist, creates
# a new file for reading and writing.
conFile = open(conFileName, 'w+')

# .con file header
conFile.write("COM PSS(R)E 32\n")
conFile.write("COM\n")
conFile.write("COM LOSS OF MULTIPLE ADJACENT TRANSMISSION LINES ON COMMON STRUCTURE IN SUBSYSTEM
'%s' \n\n"
              % subSysLabel)

# Contingencies for removing each specified Transmission Circuit
for Brnid, frombusnum, tobusnum in zip(allBrnID, FrombusNums, TobusNums):
    if  int(Brnid) != 1:
        if int(Brnid) % 2 == 0:
            count = range(int(Brnid) - 1, int(Brnid) + 1)
            conFile.write("CONTINGENCY 'L%d_%d_%d%d'\n" %(frombusnum, tobusnum, int(Brnid)-1,
int(Brnid)))
            for counter in count:
                conFile.write('DISCONNECT BRANCH FROM BUS %d TO BUS %d CIRCUIT %d\n'
                              %(frombusnum, tobusnum, counter))
            conFile.write('END \n\n')

conFile.write('END')
conFile.close()

# Run Multi-level AC Contingency Solution
psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
psspy.dfax([1,1],subSysLabel,subSysLabel,conFileName,
```

```
           conFileName1)

psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                       [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                       [subSysLabel + '_DSP',subSysLabel + '_HV',
                        subSysLabel + '_HV',subSysLabel + '_HV',
                        subSysLabel + '_HV',
                        subSysLabel + '_BULK',subSysLabel + '_BULK'],
                      conFileName1 + '.dfx',conFileName1,"","",subSysLabel + '.trp')

# Generate final excel file with results
import pssexcel

ACCFile=sys.path[0]

pssexcel.accc(conFileName1 + '.acc',['e','b','v'],colabel='',stype='contingency',
              busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
              xlsfile='ACCCResults_' + conFileName1 + '.xls',sheet='',
              overwritesheet=True,show=False)

# Generate final text file with results
psspy.report_output(islct=2, filarg="./" + 'ACCCResults_' + conFileName1 + '.txt',
                    options=[0])
psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                               [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                               conFileName1)
psspy.report_output(islct=2, options=[0])


"""
Program to perform all P7_E2 contingencies.
"""
psspy.case(CASE)
basekv=[1,999]
psspy.bsys(sid, usekv, basekv, numarea, areas)

# CON file name with contingencies
conFileName1 = "P7_E2"
conFileName = conFileName1 + '.con'

# Returns a list with DC Line names for only in-service
# two-terminal dc lines
ierr, (allDCBrnNames,) = psspy.a2trmdcchar(subsysID, 3, 1, 'DCNAME')
print "DCBrnID =", allDCBrnNames

# Returns a list with From or Rectifier Bus Numbers for only in-service
# two-terminal dc lines
ierr, (FrombusNums,) = psspy.a2trmdcint(subsysID, 3, 1, 'FROMNUMBER')
print "list of FROM bus numbers DCBrn ", FrombusNums

# Returns a list with To or Inverter Bus Numbers for only in-service
# two-terminal dc lines
ierr, (TobusNums,) = psspy.a2trmdcint(subsysID, 3, 1, 'TONUMBER')
print "list of TO bus numbers DCBrn ", TobusNums

# Opens a file object for both writing and reading. Overwrites the
# existing file if the file exists. If the file does not exist, creates
# a new file for reading and writing.
conFile = open(conFileName, 'w+')

# .con file header
conFile.write("COM PSS(R)E 32\n")
conFile.write("COM\n")
conFile.write("COM LOSS OF A BIPOLAR DC LINE IN SUBSYSTEM '%s' \n\n"
              % subSysLabel)

# Contingencies for removing each specified Pole of a DC Line
for DCBrnname, frombusnum, tobusnum in zip(allDCBrnNames, FrombusNums, TobusNums):
     if ('_' in DCBrnname) and (int(DCBrnname.rsplit('_',1)[1]) == 2):
        line_number = int(DCBrnname.rsplit('_',1)[0])
        count = [1,2]
```

```
        conFile.write("CONTINGENCY 'D%d_%d'\n" %(frombusnum, tobusnum))
        for counter in count:
            conFile.write('BLOCK TWOTERMDC %d_%d\n' %(line_number, counter))
        conFile.write('END \n\n')

conFile.write('END')
conFile.close()

# Run Multi-level AC Contingency Solution
psspy.solution_parameters_3(intgar2=200,realar5=0.1,realar15=0.05)
psspy.dfax([1,1],subSysLabel,subSysLabel,conFileName,
           conFileName1)

psspy.accc_trip_cor_2( 0.7,[0,2,1,1,0,0,0,2],[1,1,5],[1,1,0,0,0,0,1,1],
                       [ 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
                       [subSysLabel + '_DSP',subSysLabel + '_HV',
                        subSysLabel + '_HV',subSysLabel + '_HV',
                        subSysLabel + '_HV',
                        subSysLabel + '_BULK',subSysLabel + '_BULK'],
                      conFileName1 + '.dfx',conFileName1,"","",subSysLabel + '.trp')


# Generate final excel file with results
pssexcel.accc(conFileName1 + '.acc',['e','b','v'],colabel='',stype='contingency',
              busmsm=0.5,sysmsm=7.0,rating='a',namesplit=True,
              xlsfile='ACCCResults_' + conFileName1 + '.xls',sheet='',
              overwritesheet=True,show=False)

# Generate final text file with results
psspy.report_output(islct=2, filarg="./" + 'ACCCResults_' + conFileName1 + '.txt',
                    options=[0])
psspy.accc_single_run_report_2([3,1,1,0,1,1,1,0,1],[0,0,0,0,6000],
                               [ 0.5, 5.0, 90.0,0.0, 5.0,0.0, 99999.],
                               conFileName1)
psspy.report_output(islct=2, options=[0])


print('\nP7 Category Complete!')
print('NOTE: It is highly recommended to create a new folder where')
print('      to move all the files created by this program.\n')
```

A.10 Home Script

```
# Central app
# Marcos Ayala

import os

Base_Case = raw_input("Base case name, without *.sav extension: ")
Area_Number = raw_input("Area number where contingencies will take place: ")
Area_Name = raw_input("Area name where contingencies will take place: ")
Category = raw_input("Choose between P1, P2, P3, P4, P6_I1, P6_I2, P6_I3, P6_I4, or P7: ")

if Category == 'P1':
    os.system("P1_Contingency_File.py %s %s %s" %(Base_Case, Area_Number, Area_Name))
elif Category == 'P2':
    os.system("P2_Contingency_File.py %s %s %s" %(Base_Case, Area_Number, Area_Name))
elif Category == 'P3':
    os.system("P3_Contingency_File.py %s %s %s" %(Base_Case, Area_Number, Area_Name))
elif Category == 'P4':
    os.system("P4_Contingency_File.py %s %s %s" %(Base_Case, Area_Number, Area_Name))
elif Category == 'P6_I1':
    os.system("P6_I1_Contingency_File.py %s %s %s" %(Base_Case, Area_Number, Area_Name))
elif Category == 'P6_I2':
    os.system("P6_I2_Contingency_File.py %s %s %s" %(Base_Case, Area_Number, Area_Name))
elif Category == 'P6_I3':
```

```
    os.system("P6_I3_Contingency_File.py %s %s %s" %(Base_Case, Area_Number, Area_Name))
elif Category == 'P6_I4':
    os.system("P6_I4_Contingency_File.py %s %s %s" %(Base_Case, Area_Number, Area_Name))
elif Category == 'P7':
    os.system("P7_Contingency_File.py %s %s %s" %(Base_Case, Area_Number, Area_Name))
else:
    print 'You did not enter the information requested appropriately./n'
```