© 2009 Charles Davis

MULTIPLE-ELEMENT CONTINGENCY SCREENING

BY

CHARLES DAVIS

B.S., Louisiana Tech University, 2002
M.S., University of Illinois at Urbana-Champaign, 2005

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2009

Urbana, Illinois

Doctoral Committee:

Professor Thomas J. Overbye, Chair
Professor Peter W. Sauer
Professor David Nicol
Professor Alejandro Dominguez-Garcia
Dr. Santiago Grijalva

ABSTRACT

The main focus of this work is to efficiently determine the double-outage contingencies that threaten the operation of the power system. This work is necessary because enormous numbers of double-outage contingencies exist for even relatively small systems, and new standards require system operators to begin considering more than single-outage contingencies. Without an efficient method for predicting the severe contingencies, the entire set of double-outage contingencies has to be solved, and this means solving many millions of contingencies. Several algorithms are presented to detect double-outage contingencies that result in violations. The algorithms use varying amounts of information. However, at most they use linear sensitivities, line limit information, and line flow information. The output of the screening algorithms is compared to the full set of results for the double-outage contingency analysis, solved using the dc power flow. The results show that, even using a very limited amount of information about the system, it is possible to predict a very high percentage of the double-outage contingencies that result in violations.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Over the past decade, the electric industry in the United States has changed drastically. Deregulation and the introduction of markets have changed the way the power system is owned and operated. Open access to the transmission system has given independent power producers access to the bulk transmission system. Previously, utilities owned both transmission and generation, which meant that they had the ability to plan for generation and transmission together [1]. They also were also able to control access to their transmission networks [2].

Deregulation has changed both the planning and operation of the power system [3]. Utilities have lost direct control of their transmission networks. Since the adoption of the Energy Policy Act of 1992, utilities have been forced to allow independent power producers access to their transmission networks. Order 888 from the Federal Energy Regulatory Commission (FERC) opened up transmission networks to third-party power transfers [4]. Combined with a low rate of investment in the transmission system, these changes in network usage have resulted in a network that is increasingly stressed. To address this issue, the North American Electric Reliability Corporation (NERC) has introduced standards requiring that operators assess system performance under multiple outages [5]. Processing multiple outages, however, is quite computationally intensive, and there are still technical challenges to overcome when processing the huge number of potential events. As multiple outages begin to be

considered, the size of the contingency list grows rapidly. In particular,

$$list\ size = \binom{L}{k} = \frac{L!}{k!(L-k)!} \tag{1.1}$$

where L is the number of branches in the system and k is the number of outaged lines.

For the double-outage case ($k = 2$), the binomial coefficient can be expanded to

$$\binom{L}{2} = \frac{L(L-1)}{2} = \frac{L^2 - L}{2} \tag{1.2}$$

For the triple-outage case,

$$\binom{L}{3} = \frac{L(L-1)(L-2)}{6} = \frac{L^3 - L^2 + 2L}{6} \tag{1.3}$$

Continuing along these lines, it is easy to show that for k simultaneous outages, $O(L^k)$ power flow solutions are required. For power systems, the number of lines tends to scale linearly with the number of buses (i.e., $L \approx 1.5 \cdot n$), where n is the number of buses in the system. Using this relation, we can convert $O(L^k)$ to a function of n, the number of buses in the system:

$$O(L^k) = O((1.5n)^k) = O((1.5)^k n^k) = O(n^k) \tag{1.4}$$

According to [6], solving the power flow with Newton's method requires $O(n^{1.4})$ computations, and solving the power flow using linear methods requires $O(n^{1.2})$ computations. Combining the results from [6] with (1.4) gives the total computational effort for solving k simultaneous outages. The computational effort to solve multiple-outage contingency analysis using Newton's method is

$$CE_N = O(n^{1.4}) \cdot O(n^k) = O(n^{k+1.4}) \tag{1.5}$$

and the computational effort using linear methods is

$$CE_L = O(n^{1.2}) \cdot O(n^k) = O(n^{k+1.2}) \tag{1.6}$$

For the double-outage case, the computational complexity is $O(n^{3.4})$ when Newton's method is used and $O(n^{3.2})$ when linear methods are used. Thus, solving every double-outage contingency is quite computationally intensive, to the point of being intractable for even relatively small systems, even when linear methods are used. It is the goal of this dissertation to reduce the necessary computations by taking advantage of the fact that most lines in the system are weakly coupled. An algorithm is presented that removes the need to consider contingencies that do not impact each other.

The algorithm works in two parts. First, a reduced system model is constructed based on single-outage line outage distribution factor (LODF) values. Then, the reduced system model is used to create a contingency list that contains every double-outage contingency that appears in the reduced model structure. This contingency list is much smaller than the list containing every double-outage. Initial results show that the reduced list is only a few percent of the size of the initial double-outage contingency list. This means that the algorithms can effectively screen out most of the double-outage contingencies, leaving only the ones that impact each other above a given threshold.

Several algorithms to reduce the computational burden of processing multiple contingencies are presented in Chapter 6. These algorithms use several different methods to generate a list of serious double outages. The algorithms use varying amounts of information about the system. The simplest ones use only the system topology information to determine which lines in the system are tightly coupled. The results of the screening algorithms are compared against the (dc)

contingency analysis results for a 5395-bus section of the North American Eastern Interconnect. Details for the large case can be found in Appendix A. Background on contingency-ranking methods is presented in Chapter 4. The results from the large-case contingency analysis are presented in Chapter 5.

Contingency screening is the main focus of this work. However, there are several other results presented in this dissertation. Chapter 3 discusses a metric for measuring the coupling of a multiple-outage contingency. The metric is useful for examining and detecting the conditions when linear sensitivities fail. The metric approaches infinity in the event that a system island forms, and the reasons for this are examined. In the event that the outaged lines are completely decoupled, the metric is 1.0. This indicates the special case where superposition will work to approximate flow changes using linear sensitives. The statistical distribution of the condition numbers are examined for the IEEE 300-bus test case and the large case described in Appendix A. The distributions show that the larger the system the less the outages interact with each other. This result is expected because, in larger systems, the lines can be expected to impact each other less. For example, a line in Florida could be expected to have little impact on a line in Wisconsin. The special case when this observation breaks down is also explored. The metric is also very closely related to the screening algorithms based on the impact-tracking structure.

The algorithms that are used to generate lists of serious contingencies also create information that can be used for weak-element identification. A weak-element in a power system is a line or a transformer that is prone to limit violations. Because the algorithms work by identifying which lines impact each other, the information can also be used for weak-element identification. Lines that are heavily impacted by a large number of lines are likely to be important to the operation of the system. Chapter 8 discusses methods for weak element

identification and characterization using the tracking structures generated by the screening algorithms.

The mathematics for a bound on the change in flow for the linear sensitives are developed in Chapter 9. This chapter develops the flow bound using matrix and vector norms. Numerical results are presented for the IEEE 14-bus case.

An advanced technique for flow arrow visualization is presented in Chapter 10. Chapter 10 presents a new implementation for efficiently generating a large number of flow arrows, which are used to indicate the direction and magnitude of line flows on a one-line diagram. The implementation uses the geometry shader, a new feature on graphical processors, to generate flow arrows in parallel without burdening the central processor of the computer. Timing results are presented showing that, on modern hardware, the geometry shader method is faster than traditional methods. Finally, conclusions are presented in Chapter 11.

CHAPTER 2

LINEAR ANALYSIS

2.1 Linear Analysis

Linear analysis has a long history in power systems analysis. It has been used for at least 45 years to approximate the power system state quickly without the computational expense of solving the full ac power flow [7]. The speed and relatively accurate results of linear methods make them popular for on-line use. The NERC operations manual defines flowgates using linear sensitivities for monitoring the system state [8]. Their efficiency makes them useful for large-scale problems such as available transfer capability calculations [9], [10].

The fundamental basis for power system sensitivities consists of the power flow equations, a nonlinear set of equations that expresses the conservation of complex power at every bus [11]. The power flow equations simply state that the sum of the complex power at a bus must be equal to zero. They can be expressed as a real and imaginary equation. The real equation,

$$f_i^p(\theta, \mathbf{V}) = -P_i^g + P_i^l + \sum_{k \in C} |V_i||V_k| \left(g_{ik}\cos(\theta_{ik}) + b_{ik}\sin(\theta_{ik})\right) = 0 \qquad (2.1)$$

represents the conservation of real power flow at a bus, and

$$f_i^q(\theta, \mathbf{V}) = -Q_i^g + Q_i^l + \sum_{k \in C} |V_i||V_k| \left(g_{ik}\sin(\theta_{ik}) - b_{ik}\cos(\theta_{ik})\right) = 0 \qquad (2.2)$$

represents the conservation of reactive power at a bus, where V_i is the voltage at bus i and θ_i is the angle at bus i. Together, the voltages and angles describe the state of the system. In other words, the system can be completely described if all the bus voltage magnitudes and angles are known. The admittance values are parameters that are determined by transmission line geometry or transformer properties. In the power flow equations above, g_{ij} is the real part of the complex admittance on the line between bus i and bus j, and b_{ij} is the imaginary part of the complex admittance on the line between bus i and bus j. The generation values are treated as controllable inputs because the generator's output is a controllable quantity, and the load values are usually treated as a given, reflecting the fact that the load in a power system is an aggregate value from many uncontrollable customers.

Together the angles and voltages are concatenated to form a state vector

$$\mathbf{x} = [\mathbf{V}\ \boldsymbol{\theta}]^T \tag{2.3}$$

The boldface $\mathbf{V}$ and $\boldsymbol{\theta}$ are used to denote vectors of bus voltages and bus angles. Every bus where the voltage is a free (i.e., not controlled to a specific value by a generator or other voltage control device) variable has a voltage entry in the $\mathbf{V}$ vector, and every bus where the angle is a free variable has an entry in the $\boldsymbol{\theta}$ vector. The only bus where the angle is fixed is the slack bus. Once the state variables have been solved for, all other system quantities (e.g., line flows) can be found. The line admittance are parameters in the power flow.

The controllable quantities may also be collected into a vector, which is usually called $\mathbf{u}$. The controllable quantities are typically generation, so a

typical control vector may look like

$$\mathbf{u} = [P_i^g \ \cdots P_j^g]^T \tag{2.4}$$

where P_i^g and P_j^g represent the real power generation at bus i and bus j. Reactive generation is not included in the vector of controls because the reactive generation varies as needed to control the voltage. It is not controlled directly.

Using the $\mathbf{x}$ and $\mathbf{u}$ notation, the power flow equations may be written as

$$\mathbf{f}(\mathbf{x}, \mathbf{u}) = \mathbf{0} \tag{2.5}$$

The boldface denotes that each of the elements is a vector. The vector $\mathbf{f}$ is a vector of the real and reactive power flow equations given in (2.1) and (2.2):

$$\mathbf{f}(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} f_i^p \\ f_i^q \end{bmatrix} = \mathbf{0} \tag{2.6}$$

An example bus is shown in Figure 2.1. This figure illustrates the conservation of power at bus i. The power flow equations state that the complex power at bus i must sum to zero. This requires that the power injected by the generator $P_i^g + jQ_i^g$, withdrawn by the load $P_i^l + jQ_i^l$, and supplied by the two lines connected to bus i must equal zero.

One feature of the power flow that makes it hard to deal with is that it is a nonlinear problem. This means that iterative techniques must be used to arrive at a solution. Solving the nonlinear power flow is usually done using Newton's method [12], [13], because it has good convergence characteristics and works well with sparse matrices. However, Gauss-Seidel and other algorithms have been applied in the past [14] because of their simplicity and memory efficiency.

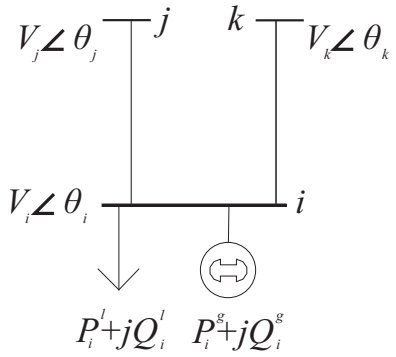

Figure 2.1: Power conservation at bus i

To avoid the difficulties of nonlinear systems, the power flow can be reduced into a linear system, which has many desirable mathematical properties. In particular, for a connected power system, there is always a solution and the solution can be calculated by solving a linear system. The dc assumptions provide the justifications for reducing the power flow equations into a linear system [13], [15]. The dc assumptions are

- There are no resistive losses.
- Bus voltages are close to 1.0 pu.
- Angle differences across lines are small.

These assumptions usually hold for ordinary power systems under ordinary conditions. Of course, there are no guarantees.

The first assumption, that there are no resistive losses in the system, means that the g_{ij} term in the power flow becomes zero, which causes (2.1) to reduce to

$$f_i^p(\mathbf{x}) = -P_i^g + P_i^l + \sum_{k \in C} |V_i||V_k| \left(b_{ik} \sin(\theta_{ik})\right) \tag{2.7}$$

The assumption that the bus voltages are close to 1.0 pu removes the voltages

from (2.1), which further reduces the power flow equations to

$$f_i^p(\mathbf{x}) = -P_i^g + P_i^l + \sum_{k \in C} (b_{ik} \sin(\theta_{ik})) \tag{2.8}$$

Finally, the assumption that the angle differences across lines are small allows the application of the small-angle approximation; i.e. $\sin(\theta_{ik}) \approx \theta_{ik}$. Application of the small-angle approximation removes the sine term from the power flow equation to give

$$f_i^p(\mathbf{x}) = -P_i^g + P_i^l + \sum_{k \in C} (b_{ik} \cdot \theta_{ik}) \tag{2.9}$$

The equation above is written for a single bus. However, the conservation of power at every bus can be written as a matrix equation. When the conservation of power is written as a matrix equation, the dc power flow equations are produced:

$$\mathbf{B}\boldsymbol{\theta} = \mathbf{P} \tag{2.10}$$

The linear system of equations can be used as a basis for the derivation of sensitivities. Many useful sensitivities have been developed taking this approach. A formal approach for analyzing distribution factors is presented in [16]. This paper derives distribution factors, starting with the system Z-bus matrix, and compares the results to previous treatments [17].

One useful type of sensitivity tells how line flows respond to a change in generation dispatch. Power transfer distribution factors (PTDFs) are the linear sensitivities of line flows (f) to a point-to-point transfer of power ($T_{(i,j)}$) [18], [19]:

$$p_\alpha, (i,j) = \frac{\Delta f_\alpha}{T_{(i,j)}} = \frac{1}{x_\alpha} \mathbf{a}_\alpha \mathbf{B}^{-1} \mathbf{a}_{(i,j)}^T \tag{2.11}$$

where α denotes the line whose flow is being altered by the transfer; i and j are the buses where the injection and withdraw take place. The symbol $\mathbf{a}_\alpha$ denotes

a row vector containing a 1 and -1 at the from and to positions of line α, and $\mathbf{a}_{(i,j)}$ is a row vector containing a 1 and -1 at positions i and j, respectively. In practice, PTDFs are used to approximate the change in flow on a transmission line caused by a change in injection and withdraw. It should be noted that for an injection across a radial line, the total amount of the transfer must flow over the radial line. This means that

$$p_{\alpha,\alpha} = \frac{\Delta f_\alpha}{T_\alpha} = 1.0 \tag{2.12}$$

when line α is a radial line. This observation is extended in [20] to say that the sum of PTDFs over a topological cut is 1.0. In other words, the sum of PTDFs over any set of lines connecting two islands in the system is 1.0. This fact can be used to detect radial lines quickly.

At first glance, it would appear that calculating four values of $\mathbf{B}^{-1}$ requires the factorization of $\mathbf{B}$ and two forward-backward solutions — one to obtain each column. However, the structure of the $\mathbf{a}$ vectors allows the calculation to be even faster. The fact that the $\mathbf{a}$ vectors' only nonzero entires are a 1 and a -1 means that we are essentially subtracting two columns from each other in the inverse, which can be done with one forward-backward solution that uses the entire $\mathbf{a}$ vector (with both a 1 and a -1) instead of a single element at a time. Thus, calculation of a PTDF value requires the factorization of the $\mathbf{B}$ matrix as well as a single forward-backward solution. The process can be further improved by the use of fast-forward and fast-backward techniques [21] to calculate the four values needed from the inverse of $\mathbf{B}$.

In fact, this process can be easily adapted to give much more than a single PTDF value. Again, this can be realized with the help of Figure 2.2, which shows that the forward-backward solution process results in a vector – not just

two single values. The only thing that is needed to convert this vector of values into a vector of PTDF values are the line impedances. So, if we define

$$\mathbf{X} = \begin{bmatrix} x_1 & \cdots & 0 \\ 0 & \ddots & 0 \\ 0 & \cdots & x_L \end{bmatrix} = \text{diag}(x_1, \ldots, x_L) \tag{2.13}$$

to be an $L \times L$ diagonal matrix of line impedance values and

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_L \end{bmatrix} \tag{2.14}$$

to be an $N \times L$ incidence matrix (where L is the number of lines in the system and N is the number of buses), then the scalar expression (2.11) can be extended to

$$\mathbf{P} = \mathbf{XAB}^{-1}\mathbf{a}_{(\mathbf{i},\mathbf{j})} \tag{2.15}$$

which gives a vector of PTDF values on every single line for a transfer from bus i to bus j.

Another important set of sensitivities can be built from PTDFs. Line outage distribution factors (LODFs) are linear sensitivities of line flows to the preoutage flow on an outaged line (β) [18]:

$$d_{\alpha,\beta} = \frac{\Delta f_{\alpha,\beta}}{f_\beta} = \frac{p_{\alpha,\beta}}{1 - p_{\beta,\beta}} \tag{2.16}$$

LODFs can be used to calculate the change in flow on line α after the outage of line β:

$$\Delta f_{\alpha,\beta} = d_{\alpha,\beta} f_\beta \tag{2.17}$$

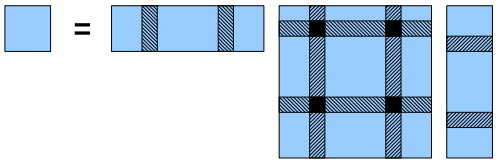

Figure 2.2: Matrix structure of (2.11)

It can be observed that when line β is radial,

$$p_{\beta,\beta} = 1.0 \tag{2.18}$$

which results in

$$d_{\alpha,\beta} = \infty \tag{2.19}$$

Thus, for radial lines, the self-PTDF value is 1.0 and the LODF value calculated using (2.16) is ∞. However, to have physical meaning, the LODF for any line onto itself must be -1.0 ($-d_{\beta,\beta} = 1$). This is because an open line must have zero flow.

2.2 LODF Derivation

The matrix $\mathbf{M}$ accounts for the fact that outages affect each other. If this is not the case, then the change in flows can be calculated using superposition. In the single-outage case, the equation for the change in flow (2.31) involves only scalar quantities. The following section begins with a LODF derivation for the

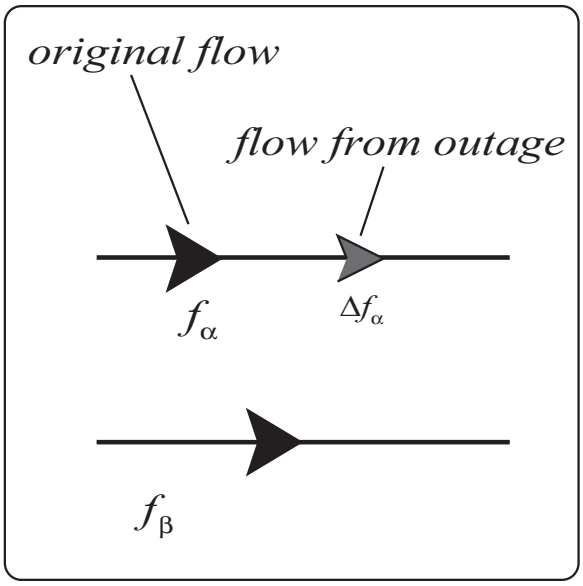

Figure 2.3: Impact on line α for the outage of line β

single-outage case. Then the derivation is extended for a more general multiple-outage case.

The derivation of single-outage LODFs begins with the definition of an LODF (2.16). By definition, an LODF is the change in flow on line α as a percentage of the preoutage flow on line β. This is illustrated in Figure 2.3. A useful expression for the LODF in terms of PTDFs can be derived by assuming that the transfer that simulates zero flow on line β, T_β, is known. This transfer is the transfer that causes line β to appear open to the rest of the system. This transfer is illustrated in Figure 2.4. The transfer across line β also affects the flow on line β, which may be calculated using

$$f_\beta^{new} = f_\beta + p_{\beta,\beta} T_\beta \tag{2.20}$$

where f_β^{new} is the flow on line β after it has been altered by the transfer.

The next step is to use the conservation of power at one terminal bus of line β. The conservation of power is preformed at the circle in Figure 2.4. By

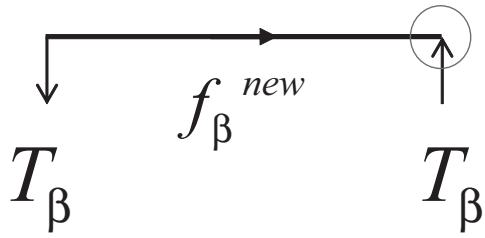

Figure 2.4: Transfer simulating outage of line β

summing the power coming into and going out of this surface, we can write

$$T_\beta = f_\beta + p_{\beta,\beta} \cdot T_\beta \tag{2.21}$$

Now, the change in flow on line α can be written in terms of PTDFs:

$$\Delta f_{\alpha,\beta} = p_{\alpha,\beta} \cdot T_\beta \tag{2.22}$$

After this is done, (2.21) can be solved for f_β to obtain

$$f_\beta = T_\beta - p_{\beta,\beta} \cdot T_\beta \tag{2.23}$$

Finally the expressions for f_β and Δf_α can be substituted into the definition of LODF (2.16) to give

$$d_{\alpha,\beta} = \frac{\Delta f_\alpha}{f_\beta} = \frac{p_{\alpha,\beta}}{1 - p_{\beta,\beta}} \tag{2.24}$$

which is the the final expression for a LODF in terms of PTDFs.

To extend the single-outage derivation to the multiple-outage case, we consider the impact of line β and line δ on line α. Transfers to simulate outages will result in altered flows ($\widetilde{f}_\beta$,$\widetilde{f}_\delta$). The altered flows are unknown, and they impact each other. However, using the tilde notation, a system of equations can be written and solved for the altered flows. In this case, we write the equation for $\widetilde{f}_\beta$ assuming that the altered flow $\widetilde{f}_\delta$ is known, and vice versa. This gives a

system of equations

$$\widetilde{f}_\beta = f_\beta + d_{\beta,\delta}\widetilde{f}_\delta \tag{2.25}$$

$$\widetilde{f}_\delta = f_\delta + d_{\delta,\beta}\widetilde{f}_\beta \tag{2.26}$$

This system of equations can be rewritten in matrix form to give

$$\begin{bmatrix} f_\beta \\ f_\delta \end{bmatrix} = \begin{bmatrix} 1 & -d_{\beta,\delta} \\ -d_{\delta,\beta} & 1 \end{bmatrix} \begin{bmatrix} \widetilde{f}_\beta \\ \widetilde{f}_\delta \end{bmatrix} \tag{2.27}$$

Now we can solve the above equation for the altered flows to get

$$\begin{bmatrix} \widetilde{f}_\beta \\ \widetilde{f}_\delta \end{bmatrix} = \begin{bmatrix} 1 & -d_{\beta,\delta} \\ -d_{\delta,\beta} & 1 \end{bmatrix}^{-1} \begin{bmatrix} f_\beta \\ f_\delta \end{bmatrix} \tag{2.28}$$

The final step is to apply these results to contingency analysis. To simulate the change in flows due to the outage of line β and line δ, multiply the adjusted flows by the sensitivities onto the line of interest. For example, to calculate the change in flow on line α, the adjusted flows are multiplied by the LODFs of the outaged lines onto line α:

$$\Delta f_\alpha = \begin{bmatrix} d_{\alpha,\beta} & d_{\alpha,\delta} \end{bmatrix} \begin{bmatrix} \widetilde{f}_\beta \\ \widetilde{f}_\delta \end{bmatrix} \tag{2.29}$$

If the expression in (2.28) for the altered flows is substituted into (2.29), then the expression for the change in flows can be written as

$$\Delta f_\alpha = \begin{bmatrix} d_{\alpha,\beta} & d_{\alpha,\delta} \end{bmatrix} \begin{bmatrix} 1 & -d_{\beta,\delta} \\ -d_{\delta,\beta} & 1 \end{bmatrix}^{-1} \begin{bmatrix} f_\beta \\ f_\delta \end{bmatrix} \tag{2.30}$$

Linear sensitivities have been extended to approximate changes due to multiple-line outages [18], [22]. The formulation in [23] results in a simple matrix equation for postoutage flow,

$$\Delta f_\alpha = \mathbf{L}_\alpha \mathbf{M}^{-1} \mathbf{F} \tag{2.31}$$

where $\mathbf{M} \in \mathbb{R}^{k \times k}$ is a matrix of LODF values relating the outaged lines to each other, $\mathbf{L}_\alpha \in \mathbb{R}^{1 \times k}$ is a row vector relating the outaged lines to line α, and $\mathbf{F} \in \mathbb{R}^{k \times 1}$ is a vector of preoutage flows.

The matrix $\mathbf{M}$ compensates for multiple-line outages. For the double-outage of line β and line δ, $\mathbf{M}$ can be expressed as

$$\mathbf{M} = \begin{bmatrix} 1 & -d_{\beta,\delta} \\ -d_{\delta,\beta} & 1 \end{bmatrix} \tag{2.32}$$

The vector $\mathbf{L}_\alpha$ is

$$\mathbf{L}_\alpha = [d_{\alpha,\beta}\ d_{\alpha,\delta}\ ] \tag{2.33}$$

and $\mathbf{F}$ contains the preoutage flows on lines β and δ,

$$\mathbf{F}^T = [f_\beta\ f_\delta\ ] \tag{2.34}$$

If these values are substituted into (2.31), then Δf_α can be written as

$$\Delta f_\alpha = [d_{\alpha,\beta}\ d_{\alpha,\delta}\ ] \begin{bmatrix} 1 & -d_{\beta,\delta} \\ -d_{\delta,\beta} & 1 \end{bmatrix}^{-1} \begin{bmatrix} f_\beta \\ f_\delta \end{bmatrix} \tag{2.35}$$

For a single-line outage that results in islanding, the LODF value goes to ∞. Similar behavior is seen for multiple-line outages. When the outage of multiple

lines results in islanding, the matrix $\mathbf{M}$ becomes singular. This behavior is explored in detail in Chapter 3.

The presentation above expresses the change in flow for single (2.17) or double outages (2.35). However, there is no limit to the number of outaged lines that this methodology is able to handle. The expression (2.31) can be scaled to arbitrary dimensions by simply expanding the number of outaged lines to be considered.

CHAPTER 3

CONDITION NUMBERS

Linear sensitivities are popular methods for approximating the change in power system quantities after a change in the system. This chapter examines the formulas that are used to calculate line outage distribution factors (LODFs) for both single and double outages. Studying these formulas shows cases where they break down. In particular, the formulas fail when system islanding occurs. This happens for both single- and multiple-line outage sensitivities. The reasons islanding causes the formulas to fail are studied in depth, and comments are made about the use of linear sensitivities in these cases. A metric of interaction — a measure of how close the system is brought to islanding by a particular outage — is developed based on the analysis. The metric is proposed as a useful method of categorizing line outages, and it is used in later chapters as a tool to help rank contingencies. Statistics are presented for the IEEE 300-bus case and a large system based on a North American utility. In both cases the statistics show that the distribution of the condition numbers indicates that most double outages are nearly decoupled.

3.1 Introduction

Linear methods have been used for at least 45 years to approximate the change in power system state quickly without the computational expense of solving the full ac power flow [7]. As discussed in detail in Section 2.1, LODFs are the linear

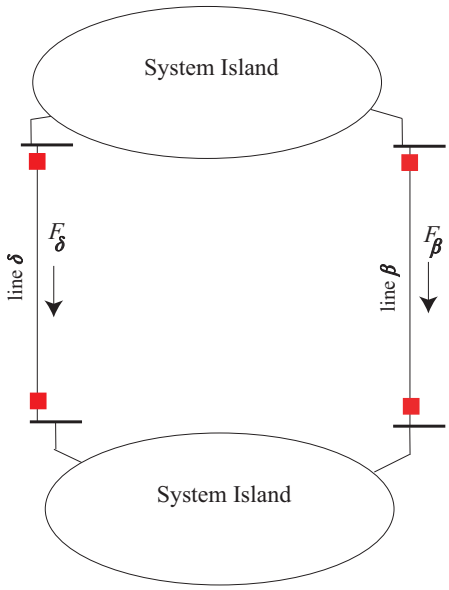

Figure 3.1: System islands connected by tie lines

sensitivities of line flows to a line outage [18]. Per the definition of LODF (2.16), they provide a metric of the impact a line outage has on the other lines in the system. Using LODFs as a metric, it is possible to determine efficiently which lines impact each other in the system.

Typically, LODF values tend to decay as the distance from an outaged line increases. However, this tendency does not hold in all cases. In particular, the tendency of LODFs to decay breaks down in the event of islanding. In the event of islanding, it is possible for the outage of one line to have a large impact on another distant line. Consider the case illustrated in Figure 3.1. Figure 3.1 shows two system islands connected by two tie lines. In the event that one line outages, all the power must flow on the other line because of the conservation of power. The conservation of power must hold for the entire system, which includes both system islands. It does not matter how far (geographically or electrically) the two lines are from each other. Because the net load in the two systems is constant, the outage of one line between the islands will result in the entire transfer shifting to the other line. Figure 3.1 illustrates two lines

connecting two system islands, so the outage of one line will result in power being forced onto the second line to maintain the balance of power. This situation can be extended to more than two outages by treating the outage of every line until the second to last line normally (i.e., with LODFs and OTDFs). The last two lines then form a situation like the one shown in Figure 3.1.

The operation of the power grid has changed dramatically over the past decade. Deregulation has changed the way the system is owned and operated [1]. Considering the impact of multiple outages has become more relevant after the introduction of new NERC standards setting performance requirements in the event of multiple outages [5]. These standards are intended to ensure the reliability of the system in the new deregulated environment.

Considering multiple outage contingencies is common for certain special cases. System operators maintain a list of contingencies, and it is not uncommon for a contingency definition to involve more than one piece of equipment. They know from experience the outages that will threaten their systems. The metric is proposed as a quick way to search exhaustively for coupled contingencies.

The analysis in this chapter is based on the linear analysis material presented in Chapter 2. Section 3.2 examines the special cases where the formulas for calculating linear sensitivities fail and the definitions must be applied to achieve reasonable results. Section 3.3 introduces a measure of outage coupling. Section 3.4 presents statistical data from the IEEE 300-bus test case [24]. Statistical data from a large system are presented in Section 3.5, and conclusions are presented in Section 3.6.

3.2 Analysis of Linear Sensitivities

The application of linear sensitivities for contingency analysis has certain caveats. For example, the singularity in

$$d_{\alpha,\beta} = \frac{p_{\alpha,\beta}}{1 - p_{\beta,\beta}} \tag{3.1}$$

occurs when the self-PTDF ($p_{\beta,\beta}$) value is 1. Traditionally, this is handled by the application of the LODF definition (2.16) to arrive at sensible results. In this section the situations are examined in which these caveats appear for both the single- and double-outage cases, and this analysis is used to make statements about the use of these sensitivities in these situations.

3.2.1 Single-outage sensitivities

In the scalar case (i.e., single-line outage), the equation for the change in flow on a line for an outage of another line is given in

$$\Delta f_\alpha = d_{\alpha,\beta} f_\beta \tag{3.2}$$

which can be written in terms of PTDFs as

$$\Delta f_\alpha = \frac{p_{\alpha,\beta}}{1 - p_{\beta,\beta}} f_\beta \tag{3.3}$$

By examining this equation, it may be observed that $\Delta f_\alpha \to \infty$ as $p_{\beta,\beta} \to 1.0$. This may be interpreted in terms of system topology for the case when the self-PTDF value is 1.0. In this case, the fact that the self-PTDF value is 1.0 indicates that a transfer from one end of the line to the other must flow entirely across that line. There is no alternate path for power to flow on. In other words,

the line is radial.

In practice, when this situation is encountered, the typical approach is to resort to the definition of an LODF (2.16) and use the fact that the line is radial to say that the self-LODF must be -1 and that the other LODFs are 0. Using -1 is equivalent to saying that the outage of this line will zero its flow. Setting the other LODF values to zero says that this line will not impact any other lines when it outages. By doing this, we have created a situation in which equation (3.2) may still be used to be used to calculate the change in flows.

However, there is a problem with this approach. Namely, we are not considering the fact that the system load or generation is changing when the radial line is opened. Except in the special case that the flow on the radial line is zero, the outage of the radial line will result in a generation imbalance within the power system island. Generation will no longer equal load. Instead of applying the definition, the mathematics seem to suggest that the LODF value is undefined in the event of islanding.

Simply setting the self-LODF to -1 and the others to 0 will not account for the imbalance. In order to predict the effects of the outage on the line flows correctly, the generation response must be modeled. The important characteristic of this case is not that a line has been outaged. Instead, the important characteristic is that some load or generation has been lost. This means that modeling flow changes cannot be done realistically without modeling the generation's response to an outage.

In the normal use of distribution factors (i.e., without islanding), the conservation of power is maintained. The net power within an island does not change. In reality there will be a small change caused by the change in losses as power flows are redistributed. However, this change is typically small, which is one reason the dc sensitivities tend to work well.

In the case that a radial line is outaged, a load or generator is disconnected from the system. In this event, there is a net change in the island's load or generation, and conservation of power is no longer satisfied. This suggests that modeling the system redispatch must be done to simulate the outage of a radial line.

3.2.2 Double-outage sensitivities

In the double-outage case we consider the matrix equation for the change in flow on line α written for the outage of line β and line δ,

$$\Delta f_\alpha = [d_{\alpha,\beta}\ d_{\alpha,\delta}\ ] \begin{bmatrix} 1 & -d_{\beta,\delta} \\ -d_{\delta,\beta} & 1 \end{bmatrix}^{-1} \begin{bmatrix} f_\beta \\ f_\delta \end{bmatrix} \tag{3.4}$$

If we expand this expression using (3.1) to put it in terms of PTDFs,

$$\Delta f_\alpha = \begin{bmatrix} \frac{p_{\alpha,\beta}}{1-p_{\beta,\beta}} & \frac{p_{\alpha,\delta}}{1-p_{\delta,\delta}} \end{bmatrix} \begin{bmatrix} 1 & \frac{-p_{\beta,\delta}}{1-p_{\delta,\delta}} \\ \frac{p_{\delta,\beta}}{1-p_{\beta,\beta}} & 1 \end{bmatrix}^{-1} \begin{bmatrix} f_\beta \\ f_\delta \end{bmatrix} \tag{3.5}$$

we can repeat the analysis preformed for the scalar case. Examining Equation (3.5) it is clear that strange behavior will arise in the event that $p_{\beta,\beta} = 1.0$ or $p_{\delta,\delta} = 1.0$. In these cases, the matrix $\mathbf{M}$ becomes extremely ill conditioned. In fact, as the self-PTDF approaches 1.0, the matrix approaches singularity. Thus, we can conclude that this formula will not apply in the event one of the outaged lines is radial. It is not too surprising to discover that double-outage sensitivities do not work when single-outage sensitivities fail.

A more interesting result may be had by examining the case when two lines connect two islands in the system outage, as illustrated in Figure 3.1. In this case, the conservation of power requires that the net flow between the islands

remain the same. This requires that the line that remains in service pick up the amount of power flow necessary to maintain conservation of power when the other is outaged. It may be recalled from the LODF derivation in Section 2.2 that the conservation of power is used in the derivation of the LODF from PTDF values.

In terms of LODFs, the requirement is that the LODFs of two lines in Figure 3.1 onto each other will be 1.0. If the matrix $\mathbf{M}$ is evaluated with these values, we get

$$\mathbf{M} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{3.6}$$

which is clearly singular. Right-multiplying by any vector of the form [1 1] will result in a zero. In this case, Equation (3.4) cannot be evaluated, because $\mathbf{M}^{-1}$ does not exist.

The outage of these two lines is analogous to the outage of a radial line for the single-outage case. When both lines outage, there will be an imbalance in load and generation in the island (except in the special case that the net flow across the lines is zero).

We can put the analysis in terms of PTDFs,

$$\mathbf{M} = \begin{bmatrix} 1 & \frac{-p_{\beta,\delta}}{1-p_{\delta,\delta}} \\ \frac{p_{\delta,\beta}}{1-p_{\beta,\beta}} & 1 \end{bmatrix} \tag{3.7}$$

to examine why the matrix $\mathbf{M}$ becomes singular when an island forms in the system. After putting $\mathbf{M}$ in terms of PTDFs, we can make use of the fact that the sum of PTDFs is 1.0 for a cutset [20], [22]. For our analysis, the cutset is the two lines connecting the two islands in the system.

Because the conservation of power must hold, a transfer from one island to

the other will flow entirely on one line if the other is outaged. Since we are simulating outages using transfers, this is important to keep in mind. It means that the transfer simulating the outage of line β can flow only on line δ and vice versa, as illustrated in Figure 3.1. This gives

$$p_{\delta,\beta} + p_{\beta,\beta} = 1 \tag{3.8}$$

and

$$p_{\beta,\delta} + p_{\delta,\delta} = 1 \tag{3.9}$$

which is simply a statement that a transfer from one island into the other will flow on the lines connecting the two. This fact was originally observed in [20].

Now, we can examine the determinant of $\mathbf{M}$ to explore the origins of the singularity.

$$\det(\mathbf{M}) = \frac{1}{1 - d_{\beta,\delta} d_{\delta,\beta}} \tag{3.10}$$

In terms of PTDFs, the determinant of $\mathbf{M}$ is

$$\det(\mathbf{M}) = \frac{1}{1 - \frac{p_{\beta,\delta}}{1-p_{\delta,\delta}} \frac{p_{\delta,\beta}}{1-p_{\beta,\beta}}} \tag{3.11}$$

We can find the origin of the singularity by looking at the denominator of the determinant and using the expressions (3.8) and (3.9) above. The denominator of the determinant is

$$1 - \left(\frac{p_{\delta,\beta}}{1 - p_{\beta,\beta}}\right)\left(\frac{p_{\beta,\delta}}{1 - p_{\delta,\delta}}\right) \tag{3.12}$$

Singularity will arise whenever this quantity is zero. If we solve the expressions in (3.8) and (3.9) for $p_{\delta,\beta}$ and $p_{\beta,\delta}$ and substitute into (3.12), we arrive at

$$1 - \left(\frac{1 - p_{\beta,\beta}}{1 - p_{\beta,\beta}}\right)\left(\frac{1 - p_{\delta,\delta}}{1 - p_{\delta,\delta}}\right) \tag{3.13}$$

which will give us zero in the denominator of the determinant. This will clearly cause **M** to be singular.

From this analysis, we have shown that in the event of islanding, the determinant of **M** will be zero, indicating that the matrix is singular. Also, we have examined the reason for this. The determinant becomes zero because of the conservation of power. Enforcing this constraint requires that a transfer from one island to another flow through the lines connecting them. In the special case that the lines connecting the two islands are outaged, cancellation occurs in the determinant. This result has been shown for the double-outage case. However, it is true for any number of outages. Any time the sum of PTDF values is 1.0, the matrix **M** will be singular. This is because the sum of PTDF values equaling 1.0 indicates islanding.

3.3 Metric of Outage Coupling

The analysis in Section 3.2 indicates that the matrix **M** is singular in the event that a system island forms. This section extends this observation to develop a measure of coupling between outages. The coupling between outages refers to the impact the outaged lines have upon each other. This information is reflected in the off-diagonal values of **M** (the diagonal values are always 1.0).

This section proposes the condition number of **M** as a metric of outage interaction. The condition number is suggested because the condition number can be thought of as a distance to singularity [25], [26], and, as the previous section discussed, islanding and singularity are linked. The condition of the matrix **M** can be thought of in power system terms as the distance to islanding. In other words, the larger the condition number of **M**, the closer the contingency brings the system to islanding.

The condition number of $\mathbf{M}$ is defined as

$$\kappa(\mathbf{M}) = \frac{||\mathbf{M}||}{||\mathbf{M}^{-1}||} \tag{3.14}$$

If the condition number is evaluated using the 2-norm, then (3.14) becomes

$$\kappa_2 = \frac{\sigma_{max}(\mathbf{M})}{\sigma_{min}(\mathbf{M})} \tag{3.15}$$

where σ_max and σ_min are the maximum and minimum singular values. For the case that $\mathbf{M}$ is a 2-by-2 matrix, there is a simple closed-form expression for the condition number evaluated using the 2-norm,

$$\kappa_2 = \frac{1 + \sqrt{d_{\delta,\beta} d_{\beta,\delta}}}{1 - \sqrt{d_{\delta,\beta} d_{\beta,\delta}}} \tag{3.16}$$

Evaluating this expression is quite efficient for a single outage. It requires only one multiplication, one division, and one square root calculation.

As mentioned above, $\mathbf{M}$ becomes singular when a contingency causes the formation of an island in the system. This results in the maximum possible condition number for the matrix $\mathbf{M}$, $\kappa(\mathbf{M}) = \infty$. The lowest possible condition number for the matrix $\mathbf{M}$ corresponds to the case when the outages are completely decoupled, i.e., when the off-diagonal terms of $\mathbf{M}$ are zeros. In this case, $\mathbf{M}$ is the identity matrix, and the condition number is 1.0.

It can be noted that, in the event that $\mathbf{M}$ is the identity matrix, (3.4) reduces to

$$\Delta f_\alpha = \begin{bmatrix} d_{\alpha,\beta} & d_{\alpha,\delta} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} f_\beta \\ f_\delta \end{bmatrix} \tag{3.17}$$

which is the same as

$$\Delta f_\alpha = \begin{bmatrix} d_{\alpha,\beta} & d_{\alpha,\delta} \end{bmatrix} \begin{bmatrix} f_\beta \\ f_\delta \end{bmatrix} \tag{3.18}$$

This is the special condition under which superposition works to calculate flow changes. When the outages are completely decoupled, $\mathbf{M}$ and $\mathbf{M}^{-1}$ are the identity matrix, so the change in flow can be calculated by simply adding the LODF–flow products:

$$\Delta f_\alpha = d_{\alpha,\beta} f_\beta + d_{\alpha,\delta} f_\delta \tag{3.19}$$

The off-diagonal terms in $\mathbf{M}$ will be identically zero only in the event of the outage of two radial lines. However, they will be close to zero for outages that have small LODF values onto each other.

One useful observation that follows from this analysis can be made by noting that the denominator of Equation (3.16) can provide a quick way to check for islanding resulting from a double outage. In the event that the product of the off-diagonal LODF values is 1.0,

$$d_{\delta,\beta} \cdot d_{\beta,\delta} = 1 \tag{3.20}$$

the denominator of (3.16) is zero, which results in a condition number of infinity, which is associated with islanding. Thus, to check for islanding, we can simply check to see whether the product of the off-diagonal LODF values is 1.0.

The condition number gives a metric indicating the degree of coupling of the individual outages involved in a double-outage contingency. The larger the condition number, the more the outages interact. A condition number close to 1.0 indicates that the outages have only a small impact on each other. As the condition number of the matrix $\mathbf{M}$ approaches infinity, the system comes closer to islanding. The following sections examine the statistical distributions of the

condition numbers for two different systems.

3.4 Condition Number Statistics

Using the IEEE 300-bus test case [24], condition numbers were calculated using the 2-norm for every double-outage contingency that does not result in island formation. For the 411 lines in the IEEE 300-bus test case, there are 84,255 double-outage contingencies.

Table 3.1: Condition number statistics

Max. κ_2	29.63
Min. κ_2	1.00
Avg. κ_2	1.060
Std. dev. κ_2	0.4313

Some statistics for the IEEE 300-bus double-outage condition numbers are given in Table 3.1. The statistics show that the minimum condition number is 1.0, and the average value is very close to 1.0, indicating that most matrices are nearly perfectly conditioned; i.e., the off-diagonal values are very near 0. This is significant because it means that, even for a relatively small system, most outages are essentially decoupled. In larger systems, the amount of interaction could be expected to be even smaller.

The probability density function (PDF) and cumulative distribution function (CDF) were also calculated for the data set, denoted by $F(x)$ and $P(x)$, respectively. The PDF shows where in the distribution the values tend to cluster, and the CDF shows what percentage of values fall below a given value. Plots of the CDF and PDF are shown for the IEEE 300-bus case in Figure 3.2. The plots show that the condition numbers are clustered near 1.0, indicating that most outages have weak interactions. A detail of the PDF around 1.0 is

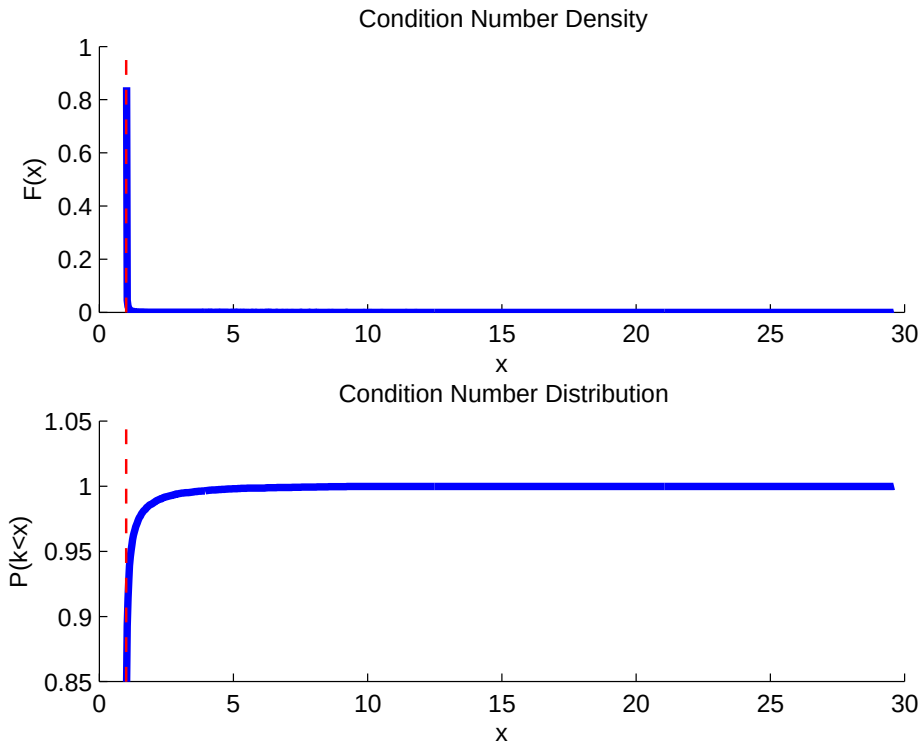

Figure 3.2: Condition number distributions

shown in Figure 3.3. The detail illustrates how quickly the condition number distribution decays. In fact, 92.3% of the data points are between 1.0 and 1.1.

The bottom graph in Figure 3.2 shows the CDF (the integral of the PDF). The CDF answers the question, What is the probability that a condition number is greater than a given value x? Several points from the CDF are listed in Table 3.2. The data in this table can be used to study how condition numbers are distributed versus $\kappa(\mathbf{M})$. For a selection of $\kappa(\mathbf{M}) = 2.258$, 99% of contingencies will have a condition number that is smaller.

The maximum condition number for any outage that does not result in islanding occurs for the outage of the lines 3-150 and 7-131. The topology around these two lines is shown in Figure 3.4. The two outaged lines are tie lines between System 1 and System 2.

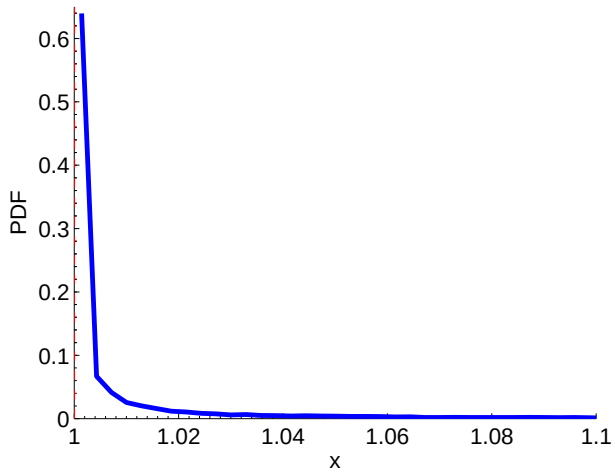

Figure 3.3: PDF near $\kappa = 1.0$

Table 3.2: Condition number distribution

$P(x)$	x
0.64	1.001
0.80	1.016
0.90	1.064
0.95	1.193
0.99	2.258

The M matrix created by the outage of 3-150 and 7-131 is

$$\mathbf{M} = \begin{bmatrix} 1 & -0.942 \\ -0.927 & 1 \end{bmatrix} \tag{3.21}$$

which has the maximum condition number of any outage, $\kappa(\mathbf{M}) = 29.63$. An examination of the topology (Figure 3.4) around the outages reveals the reason. The two lines 3-150 and 7-131 connect System 1 and System 2, two areas within the IEEE 300-bus test case. There are connections between System 1 and System 2 through the other areas, but 3-150 and 7-131 are the only two direct connections. The two outaged lines serve as the primary connections between

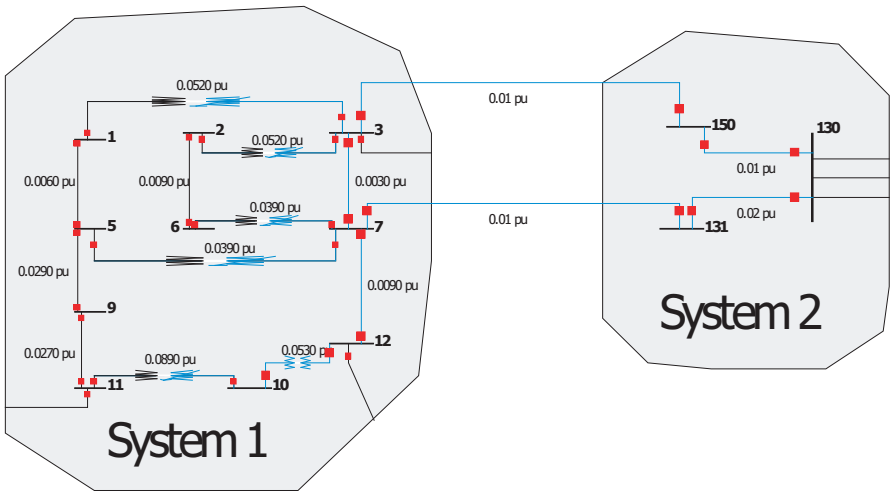

Figure 3.4: Topology near 3-150 and 7-131

the two areas, so power from one of the outaged lines will redistribute onto the other and vice versa. The two lines are operating in a parallel capacity.

The empirical results show that the condition numbers are distributed very close to 1.0. This means that the vast majority of outages are essentially decoupled. In general, the larger the system, the more the distribution can be expected to be even more heavily weighted around 1.0. This is because, in larger systems, the impact of lines upon each other is smaller.

3.5 Large-System Distribution

The condition numbers were calculated for a much larger 5395-bus 7616-line case from a large North American utility. The distributions from the larger case display the same behavior as the distributions from the IEEE 300-bus test case. As can be seen in Figure 3.5, the distribution of condition numbers is weighted very heavily toward 1.0. This is reflecting the fact that most outages are nearly decoupled. The off-diagonal LODF values are nearly 0.0 for the majority of

double outages.

Figure 3.6 shows a close up of the density plot near $\kappa = 1.0$. It can be seen that the density decays very quickly away from $\kappa = 1.0$. In fact, the density for the large case decays in a very similar way to the density for the IEEE 300-bus test case.

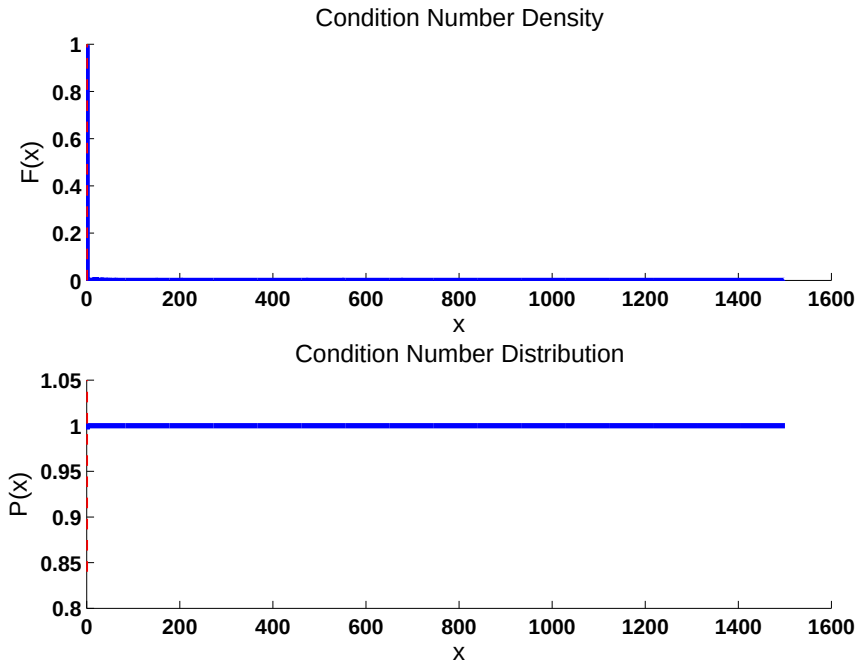

Figure 3.5: Distribution of condition numbers

3.5.1 Large-system statistics

As with the distribution plots, the statistics for the large system (Table 3.3) are very close to the values for the IEEE 300-bus test case. For both cases, the average is very near 1.0. Surprisingly, the standard deviation is larger for the large case. This is unexpected because the distribution for the large case is more clustered around 1.0, which can be seen by examining the distribution tables for each case. The larger condition number is most likely explained by the larger number of high-value condition numbers in the large case. Indeed, the most

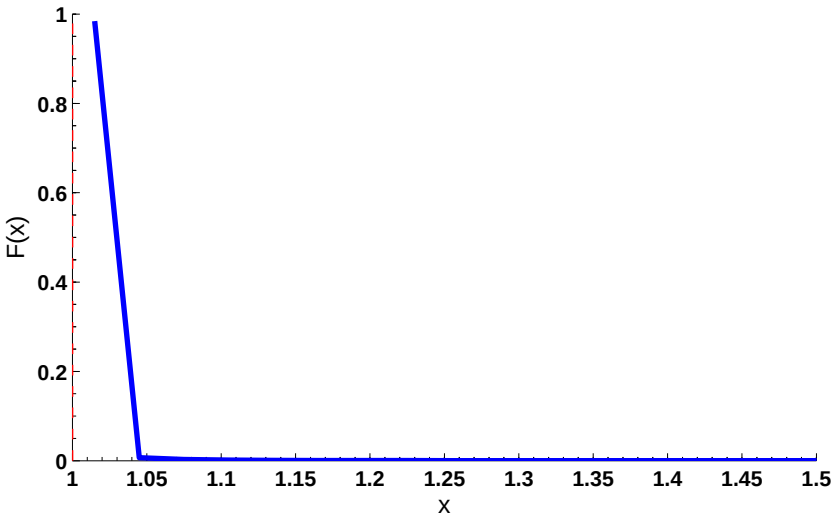

Figure 3.6: Distribution of condition numbers near 1.0

glaring difference between the IEEE 300-bus test system statistics and the large-case statistics is the largest condition number value. The largest condition number for the large case is 50 times larger. The double-outage contingency that results in this large condition number is examined below.

Table 3.3: Condition number statistics

Max. κ_2	1500.3
Min. κ_2	1.00
Avg. κ_2	1.0046
Std. dev. κ_2	0.4976

Table 3.4 contains values for the cumulative distribution function for the large-case condition numbers. Upon examining these values, it is immediately clear that the condition numbers are very heavily clustered around 1.0. The first row in the table indicates that 92.05% of the condition numbers have a value less than 1.0015. This is a much stronger clustering than was seen in the IEEE 300-bus case. For the IEEE 300-bus test case, only 64% of the condition numbers have a value less than 1.001. The difference in the distributions can be

explained by the system size. The IEEE 300-bus system is small compared to the 5395-bus large case. In the smaller case, a higher percentage of the outages will interact with each other because everything is closer together. In the larger case, everything is farther apart electrically and geographically.

Table 3.4: Condition number distribution

$P(x \leq \kappa)$	x
0.9205	1.0015
0.9735	1.0075
0.9943	1.0975
0.9995	1.9971
0.99998	10.0035

3.5.2 Large-system largest κ

The largest condition number for the large system is 1500.33. This condition number occurs for the outage of the line connection bus 18027 to bus 29539 and the line connecting bus 29537 to bus 29539. The $\mathbf{M}$ matrix formed by this double outage is

$$\mathbf{M} = \begin{bmatrix} 1 & -0.99882 \\ -0.998515 & 1 \end{bmatrix} \tag{3.22}$$

and the condition number of $\mathbf{M}$ is 1500.33, which is quite large. An examination of the system topology around the outaged lines gives insight into the large LODF values and resulting condition number. The topology at bus 29539 is shown in Figure 3.7. Clearly, these two lines are connected to the same bus (bus 29539), and the outage of one of these lines results in a large distribution of flow onto the other line.

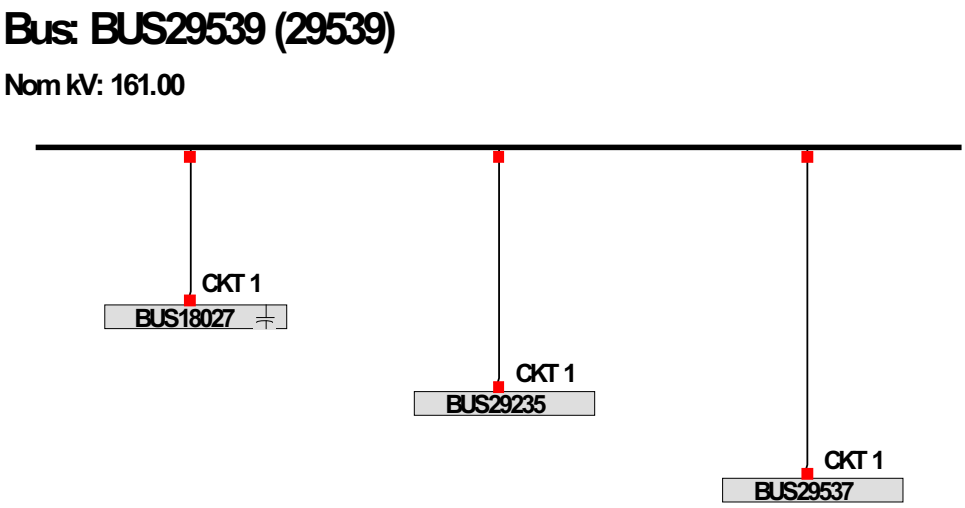

Figure 3.7: Topology at bus 29539

3.6 Conclusion

This chapter examines the cases when the formulas for linear sensitivities break down and fail to give reasonable results. The cause of the breakdown can be traced to islanding of the system. In particular, enforcing the conservation of power results in singularity in $\mathbf{M}$ when the outages result in islanding. The results were presented for the double-outage case. However, they are extensible to higher dimensions. Whenever the sum of PTDFs across a group of lines is 1.0, $\mathbf{M}$ will be singular. The observation that the sum of PTDFs over a cutset is 1.0 was first made in [20]. This chapter connects that observation to the singularity of $\mathbf{M}$ and proposes the condition number as a metric of interaction.

The condition number was proposed as a metric of interaction after an examation of the behavior of $\mathbf{M}$. The metric measures the distance of the matrix $\mathbf{M}$ to singularity. In terms of system topology, the metric can be thought of as a measure of the distance to islanding or a measure of the coupling of the outages. The larger the condition number, the more coupled the outages. An examination of the largest condition number for the IEEE 300-bus case showed

that the outaged lines impacted each other heavily and provided the only direct connection between two sections of the system.

To examine the behavior of the condition number, statistics were examined for the IEEE 300-bus case and a large case representing a portion of the eastern interconnect. For each case, the metric was calculated for every double-outage contingency. In both cases, the distribution of the condition numbers was found to be very heavily weighted around 1.0, the lowest possible condition number. This is especially true for the large case. For the large case, 92.05% of the double-outage contingencies have a condition number less than 1.0015, indicating that most of the double-outage contingencies behave almost as independent outages, which could be expected for a large case. In a large case, it is unlikely that two randomly chosen outages will have a large impact on each other because, in all likelihood, they will be seperated by a large distance both geographically and electrically. Condition numbers provide insight into the operation of screening algorithms. The condition number is used as a tool to analyze the operation of a screening algorithm in Section 6.2.3.

CHAPTER 4

REVIEW OF CONTINGENCY SELECTION

4.1 Contingency Selection

Power system operators are continuously monitoring their systems. Periodically contingency analysis is run to assess the security of the system. The on-line contingency analysis uses a list of predefined events and a state estimator model. The list of events is determined ahead of time and contains the likely events that could cause overloads or equipment damage. Since these studies are performed on-line, the size of the list is very important. The larger the size of the contingency list, the longer it will take to perform the contingency analysis. On the other hand, the list must be thorough enough to capture every significant event. The challenge with contingency selection is to accurately select the dangerous events while not including innocuous events. To accomplish this, contingency selection algorithms rank contingencies in order of severity, where severity is a function of the magnitude of the postcontingent violation. If the contingencies are correctly ranked, then evaluating the serious contingencies is simply a matter of starting with the most serious contingency and stopping when no further violations are found.

This section presents a literature review of the previous work in contingency selection. It should be noted that contingency selection and contingency screening are not the same. Contingency screening — as the term is used in this dissertation — is a complementary process to contingency selection.

Contingency selection selects bad contingencies, and contingency screening removes contingencies that need not be considered as a multiple-outage event. By removing contingencies from consideration, contingency screening eliminates a large fraction of contingencies from ever needing to be considered. Research into contingency selection began in the late 1970s and was a subject of interest into the early 1990s. By the late 1990s, fewer research papers on contingency selection were being published. This decrease seems to be because the improvements in computers made it possible to handle single outages efficiently, so there was no longer any real need for contingency selection. Recently, however, NERC has introduced standards requiring the consideration of multiple outages. Even with the most powerful modern computers, dealing with every multiple-outage contingency is intractable.

Contingency selection describes the method of determining which contingencies are important enough to add to the on-line contingency list. The initial work was done in [27]. The approach was to define a performance index. For line flows, the performance index is defined as

$$\mathrm{PI}_{MW} = \sum_{l=1}^{L} \frac{W_l}{2n} \left(f_\alpha / f_\alpha^{Max} \right)^{2n} \tag{4.1}$$

where f_α is the real power flow on-line l, f_α^{Max} is the real power flow limit, n is an exponent that penalizes the overload, and W_l is a non-negative weighting coefficient. Although most work focused on real power indices, indices are also defined for voltage and reactive power limits. These indices are used to rank contingencies in order of severity. The more severe the limit violation, the larger the performance index will be, indicating a serious contingency. The weighting coefficients can be used to weight various type of equipment or known problem lines. For example, a 69 kV line may be given a lower weighting coefficient than

a 500 kV line because the 500 kV line is judged to be more important to the operation of the system.

Of course, to evaluate these performance indices (PIs), the value of the flows must be known. To avoid this calculation, which involves actually solving the contingency analysis, the authors of [27] make the observation that, to rank contingencies, only the relative change in PI with respect to outages needs to be known. Based on this insight, the authors develop a method based on the sensitivities of the performance index (4.1). The authors do mention distribution factors as an efficient way to screen contingencies. However, they deem it unsuitable because it does not allow for the detection of voltage violations. They note that some misorderings occur in their numerical tests, but they believe this is not important as long as the ordering trend is generally correct. It may be noted at this point that the numerical examples were calculated for small systems (the largest contained 29 buses).

In [28] the methods of [27] are applied to the much larger American Electric Power (AEP) system. The results indicated that the methods in [27] had a tendency to misclassify contingencies. Important contingencies were missed, and unimportant ones were added to the list. In an attempt to improve the performance of the PI sensitivity methods, the sensitivities were modified to use second-order terms. However, the results were still unreliable. Better results were obtained using the dc power flow to calculate the performance indices for line flows. Solving the dc power flow requires one forward and backward solution of the factored $\mathbf{B}$ matrix. Of course, the dc power flow is unable to handle voltage violations, and no mention was made of generator outages.

The problems with the initial work are addressed in [29]. This method also uses PI sensitivities. However, the performance index sensitivities are presented in the form of a Taylor series expansion, and an accurate approximation is

presented. The new approximation gives better results than considering only the first-order terms, at the expense of computational time. A methodology for evaluating the effectiveness of a contingency selection algorithm is also presented in [29]. In particular a capture rate, r_N, is defined as "the fraction of worst N contingencies appearing in the first N entries in the ranking." A capture rate of 1.0 means that all of the worst contingencies are listed first in the ranking. There is also an analysis of the problems that can occur with using the performance index sensitivities. First, the authors identified the problem of masking. This phenomenon occurs when several lines are loaded near their limits. The performance index sensitivities appear the same as for an overloaded line (interestingly, this means that the more heavily loaded the system, the worse the method will perform). The authors also identified the error introduced by using only first-order terms to predict the performance index as a source of misclassification. The adapted method in [29] performs much better than the initial work [27]. In fact the adapted method works as well as the dc load flow, and results were presented for much larger systems than were presented in [27]. The experimental system in [29] has 239 buses and 380 branches.

Shortly after the publication of [29], the authors of [28] published a work in which they further developed the idea of using the dc load flow for contingency selection. The dc load flow requires one forward/backward solution of the factored $\mathbf{B}$ matrix. The authors compared this to higher-order sensitivity methods and conclude that the dc power flow has the same accuracy with equivalent computational complexity [30]. There are also some comments on the masking problem in [30]. In particular, the authors propose using two different, complementary performance indices to deal with the masking problem.

Work on contingency screening was continued in [31]. In [31] the previous work is evaluated, and new algorithms are proposed to calculate the

performance index sensitivities for single and multiple outages. This is the first mention of ranking contingencies with more than one element outage in the literature. The PI sensitivities are developed for single-branch outages, secondary outages, and multiple-line outages. Secondary outages are outages caused by the occurrence of some other outage [32].

The new methods of evaluating sensitivities, presented in [31], were evaluated on the IEEE 30-bus case. The results indicate that the use of distribution factors yields better results. The capture rate is 1.0 for the LODF-based performance indices. However, the capture is only 0.8 when the sensitivity of the performance index is used for single-branch outages. For secondary outages, the results are the same. Because of the small size of the system, it is difficult to draw any authoritative conclusions about double-branch outage sensitivities. However, these outages did tend to have high performance index sensitivities, which causes them to be highly ranked.

The one good thing about the sensitivity methods presented in [31] is their speed. They are significantly faster than even calculating the LODF values. The analysis of the various screening algorithms shows that calculating the performance indices using LODFs has a computational order of $NC \cdot t$, where NC is the number of contingencies and t is the time for one forward/backward solution time. The fastest PI sensitivity method has a computational order of $3 \cdot t$. The speed is the justification the authors use to recommend the use of the PI sensitivity methods.

An analysis of the performance index methods with improvements in calculation can be found in [33]. There a broad overview of the PI sensitivity methods is presented as well as a discussion of the computational characteristics of each algorithm. A derivation for the change in flows using linear sensitivities is presented. These derivations are then applied to calculate the performance

indices for line and generator outages. Because the sensitivities are based on the system $\mathbf{B}$ matrix, voltage violations cannot be handled. The methods are quite efficient, but they assume that the inverse of $\mathbf{B}$ is available. This means that most of the computational effort is spent at the beginning of the process calculating the inverse of $\mathbf{B}$. The explicit use of $\mathbf{B}^{-1}$ also means that the method can be applied only to relatively small systems. For larger systems, the memory requirements will be prohibitive.

The performance indices based on the formulation in (4.1) are dependent upon the value of the weights. None of the work mentioned so far discusses the selection of the weights except to say that they can be used to weight critical lines more. A more systematic approach for selecting the weighting values is presented in [34]. The work in [34] tests contingency selection algorithms and develops a theory for selecting weights and thresholds. The work in [35] extends this work, fixing some of the issues with the parameter selection method.

Voltage violations were first mentioned in the first paper on contingency selection [27], but the performance indices for real power flows became the focus of attention for quite some time. However, there was still work being done to rank contingencies using voltage- and reactive power–based metrics as well. A good review of early contingency selection algorithms, including voltage- and reactive power–based indices, is presented in [36]. More recent work using voltage nose curve sensitivities is presented in [37]. Some early work to consider the impact of reactive power on contingency selection was done in [38]. In that work the first iteration of the ac power flow is used to calculate a PI for voltage and reactive power. In [39] an algorithm is presented for voltage contingency selection. The algorithm is based on a systemwide performance index that penalizes deviations from the desired system voltage profile. Work was done to extend contingency screening to include voltage violations [40]. The method

starts by identifying buses where a significant change in voltage is likely to occur and creating a voltage subnetwork, screening out buses where a voltage change is not likely. Then an efficient process is used to determine the voltage at the voltage-sensitive buses. A contingency selection algorithm based on reactive power changes at buses after a contingency is presented in [41].

Several more exotic nonlinear techniques have been applied to the problem of contingency selection over the years. For example, in [42] decision theory is applied to contingency selection. In [43] and [44] neural networks are applied to the contingency selection problem. A more recent contingency-ranking algorithm based on neural networks is presented in [45]. An interesting method of contingency selection based on substation topology is proposed in [46]. This method begins to take into account the actual likelihoods of multiple outages.

4.2 Contingency-Ranking Methods

Contingency ranking provides a method to measure the severity of a contingency. In general, the more severe the contingency, the higher the ranking. This chapter discusses several contingency-ranking methods. Traditional contingency-ranking methods, as developed by Wollenberg in the late 1970s [27], are presented first. These methods use pentalty functions to calculate a performance index (PI), which is typically composed of a weighted sum of limit violations. The PI methods were not particularly successful at contingency slection, but they do provide an attractive method to aggregate the results from contingency analysis in order to indicate the severity of a contingency. This chapter also presents another method based on organizing the contingency analysis results into a matrix, which can then be used to calculate line-based or contingency-based metrics. The line-based metrics are described in [47] and [48] as aggregate

contingency overload (ACO) values. These methods are applied to contingency analysis results in order to determine which contingencies are most severe.

4.2.1 Penalty function methods

Traditional penalty function methods are rather simple. As discussed in Section 4.1, they were initially chosen because of their appealing form and because it is relatively easy to compute PI sensitivities. The PI sensitivities formed the basis for early contingency-ranking algorithms [27]. The idea was to calculate the sensitivities for the performance indices, which can be very efficently computed, in order to rank contingencies without approximating or calculating postcontingent flow values. While these methods were fast, they tended to be highly inaccurate [28]. Other methods of computing the performance index sensitivies were tried [30] [29], but eventally the methods were shown to have the same computational requirements as the dc power flow [33].

When used for ranking contingencies, the results are assumed to be available. When used for ranking, the traditional PI methods sum a performance index over the violations. For every violating line, sum the performance index. The performance index has the form

$$\mathrm{PI}_{MW} = \sum_{l \in \mathcal{V}} \frac{W_l}{2n} \left(f_\alpha / f_\alpha^{Max} \right)^{2n} \tag{4.2}$$

as originally given in (4.1). The difference is that the sum is now only over $\mathcal{V}$, the set of violations. For simplicity, the weighting function was chosen to be 0.5 and n was picked to be 1.0, corresponding to the choices in the literature.

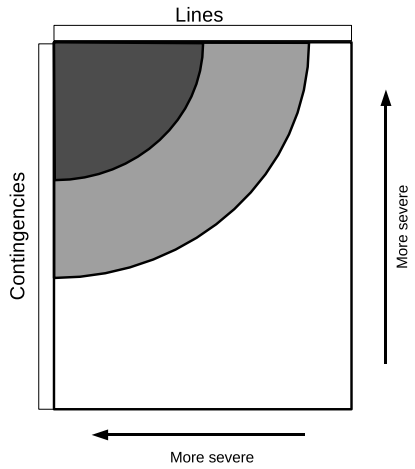

Figure 4.1: Sorted matrix ranking

Substituting these values gives the expression for the PI as

$$\mathrm{PI}_{MW} = \sum_{l=1}^{L} \left(f_\alpha / f_\alpha^{Max} \right)^2 \tag{4.3}$$

4.2.2 Sorted-matrix ranking

Sorted-matrix ranking a a new method that is useful for organizing the results from extremely large data sets. The idea behind sorted-matrix ranking is to organize the contingency results in a matrix as shown in Figure 4.1. One dimension of the matrix has a number of entries equal to the number of lines in the system. The other dimension has a number of entries equal to the number of contingencies. When a violation occurs, an entry is made in the matrix. For large systems, the matrix is too large to visualize. For example, a system with 7616 lines and 200 000 violations would require a monitor with a resolution of 7616 by 200 000 in order to represent each violation as a single pixel. These resolutions are well beyond the capabilities of modern monitors.

However, there is still a way to get meaningful information out of the matrix.

Creating an aggregate quantity by summing the violations along the rows and the columns gives a per-line and a per-contingency measure of severity [48]. By examining the distributions for these metrics, we can see the "shadows" on the axes. These "shadows" can be expressed as aggregate values on a per-line and a per-contingency basis. The values entered into the matrix are the percent over load values. These values are the amount a line is over limit as a percentage. For line α, the percent over limit can be written as

$$POL = \frac{f_\alpha}{f_\alpha^{max}} \tag{4.4}$$

The POL values can be summed over contingencies (columns in Figure 4.1) or over lines (rows in Figure 4.1) to give a per-line or a per-contingency value. The line aggregate overload can be written as

$$LAG = \sum_{k \in Contingencies} POL_k \tag{4.5}$$

The contingency aggregate overload can be written as

$$CAG = \sum_{k \in Lines} POL_k \tag{4.6}$$

The first obvious advantage in organizing the data this way is that line-based data can be generated. The PI methods offer no information about how often a line overloads. This information is lost when the sum is performed. The fact that the sorted matrix maintains the line information means that the sorted matrix may be used for weak-element identification as well as severe contingency identification.

CHAPTER 5

LARGE-CASE CONTINGENCY ANALYSIS RESULTS

This chapter presents the contingency analysis results for a large 5395-bus 7616-line section of the North American Eastern interconnect. Details about this case can be found in Appendix A. By presenting the ranking methods for the large case, a baseline for comparison is provided for the contingency screening results presented in Chapter 6. In other words, this chapter uses the full contingency analysis results to determine what contingencies threaten the system. The unprocessed contingency analysis results are discussed, and the contingency-ranking methods discussed in Chapter 4 are used to process the results further. The most severe contingencies are examined to see what types of situations result in a severe contingency. The purpose of applying the techniques in Chapter 6 to the contingency analysis results for the large case is to determine which of the contingencies are the most serious. Knowing the "worst of the worst" is useful when evaluating the effectiveness of the screening algorithms.

The severity of a contingency depends upon many factors. For example, the system topology and the system loading level will both play a large role in determining whether or not a contingency threatens the overall security of the system. Since the purpose of this chapter is to determine which contingencies are the most significant, it is necessary first to conceptually define the criteria for a significant contingency. By describing the conditions for a severe contingency, it becomes possible to process the contingency analysis results into a list of contingencies that threaten the system. This list of severe contingencies

serves as a way to check the results of the contingency-screening algorithms.

In loose language, a significant contingency is a contingency that threatens the safe operation of the system. A significant contingency threatens the bulk transmission system. For our purposes, we are less concerned with the distribution and subtransmission system. To take this into account, filtered contingency lists are created along with contingency lists based on the full contingency analysis results. To filter out the minor lines, only lines with a limit of 100 MW or more are used. This filtering alone removes 47.4% of the lines in the system from consideration. However, it should be noted that only 70% of the lines in the system have a specified limit over zero.

Significant contingencies should also occur within the region that the model can accurately simulate. In other words, lines at the edge of the system, where the external system has been equivalenced, are neglected. There are 915 lines in the system that have been flagged as having been generated when the external system was equivalenced. They are the connections between the system of interest and the external system.

The magnitude of an overload also an important factor in determining the severity of an overload. Clearly, the larger the magnitude of an overload on a line, the more severe the contingency is. For most lines in the system, a nominal limit is specified along with an emergency limit. The emergency limit is a relaxed limit at which the line can be operated during emergency conditions. However, the emergency limit can be used only for a short period of time (on the order of 30 minutes). System operators must correct the overload quickly to avoid damaging the line or transformer. To summarize, severe contingencies

- Result in violations on important lines
- Do not occur on lines created by the equivalencing process

- Violate the emergency rating

The methods in Chapter 4 are designed to identify the most severe contingencies. For example, the PI-based methods quadratically penalize the percent overload. This is intended to create a large performance index for a severely violating contingency.

The chapter begins by presenting the single-outage contingency analysis results for the large case. The single-outage results are important to consider because they will show up in the double-outage contingency analysis. In other words, a single line whose outage results in a violation will also create violations when combined with most of the other lines in the system. This means that the contingencies with violations for a single outage will also show up in the double-outage results. The reasons for the most severe single-outage contingencies are also examined.

After the single-outage analysis, the results of the double-outage contingency analysis are discussed for the large case. As mentioned above, the case has 7616 lines, which means that there are 7616 single-outage branch contingencies and 28 997 920 double-outage events involving lines and transformers. All contingencies for the single-outage and double-outage cases were solved using the dc power flow. The branch limit violations were recorded, and the results are ranked. The contingencies that result in islanding are not considered as part of the result set. The results will be ranked using the methods described in Chapter 4. These results will provide a baseline for evaluating the contingency selection algorithms presented in Chapter 6.

5.1 Large-Case Single-Outage Contingency Analysis

Before considering the double-outage violations, it is a good idea to see what single-outage contingencies result in violations, because these contingencies will inevitably also cause violations in the double-outage analysis. There are no base case violations in the system. However, there are 81 single-outage contingencies that result in violations. There are 116 total violations. The violations have limit violations between 271.5% and 100.04%. Of the 7616 single-outage contingencies, 739 result in islanding.

The worst violation occurs for the outage of the line 759 to 758 circuit 99. In this case two lines act in parallel to connect a generator to the system. When the line 759 to 758 circuit 99 is outaged, all of the generator's output is forced to flow through the parallel path, which has a much lower limit. In real life, this outage is not likely to cause problems, because the circuit 99 identifier indicates that the branch was created by the equivalencing process. This means that this outage is at the edge of the system under study. Because they were created by the equivalencing process, circuit 99 lines are not considered in the contingency analysis results.

The second highest postcontingent violation (207.8%) occurs for the outage of the 345/138 kV tranformer connecting bus 2702 to 2704. In this scenario, the outage of the transformer causes the overload of two lines serving load at bus 27157. The outage results in two overloads that occur on relatively small lines. A 22 MW capacity is overloaded to 45.72 MW, and an 80 MW capacity line is overloaded to 82.79 MW.

A similar scenario is seen in many cases that result in large postcontingent violations. In fact, most violations occur on lines with low limits. Ninety-two of the limit violations occur on lines with a limit below 100 MW. In most

situations, the loss of such a small line would not threaten the overall security of the overall bulk transmission system. However, if not dealt with properly, these contingencies have the potential to result in the loss of load or a cascading outage, so it is still important to flag these violations.

Along with the numerous violations on small lines, there are several cases in which large lines are overloaded in the event of an outage. There are nine contingencies that result in overloads on lines with limits of 400 MW. These violations occur on four different lines. However, they are all relatively small overloads. The limit is not violated by much. Table 5.1 has a listing of the major violations, where a major violation is defined as an overload on a line with a limit of at least 100 MW. All of the violations on the 400 MW lines are less than 0.2%, so for all practical purposes, these lines are being operated right at their limit. These violations are probably not a serious threat to the operation of the system because these lines could be operated at emergency ratings for quite some time. Also, the violations are so minor that a small redispatch of the generation could probably fix the problem.

Table 5.1: Major violation summary

From Bus	To Bus	Limit Used	Circuit	Violations	Max % Overload
57062	40122	400	1	2	100.1
56794	40122	400	1	2	100.2
57062	40121	400	1	2	100.1
56794	40121	400	1	2	100.1
31436	31437	386	1	1	122.9
53139	53170	335	1	2	113.2
31023	33362	249	1	1	112.9
33351	33352	249	1	1	112.8
53154	53170	247	1	1	100.0
27618	33352	223	1	1	102.5
27093	27380	126	1	4	146.9
27045	27380	126	1	2	124.9
25783	40007	112	1	1	108.0

The more serious violations occur on the lines with limits between 112 MW and 386 MW. In these cases, the violations are significantly over the line's rating. In the worst case, the outage of line 27093 to 27045 circuit 1 results in an overload of 146.9% on the line from 27096 to 27380. In this scenario, the outage of line 27093 to 27045 circuit 1 results in an overload on a line connected to the same bus. Both these lines are supplying power from generators at this bus to the rest of the system. When the line outages, there is no longer enough transmission capability to get the power from the generators out to the rest of the system. The overload can be fixed by reducing the output at the nearby generators. In practice, there is most likely a special protection scheme that forces the generators to lower their outputs in the event that one of the four contingencies that overload this line occurs. Both the system operators and the operators of the generator should be very aware that an outage at the interface between the generator and the system will result in the generator being forced to operate at reduced capacity.

The next most serious violation is 124.9% on the line from 27045 to 27380 circuit 1 for the outage of the line from 27093 to 27380 circuit 1. This contingency is very similar to the one that results in the 146.9% overload, and the solution — backing down generation at Bus 27093 — should be the same. The outage of the line from 27323 to 27370 circuit 1 also resuts in an overload on the line from 27045 to 27380 circuit 1.

The 122.9% violation is a similar situation. One line connecting a generator to the rest of the system outages, and the resulting flow redistribution results in an overload. The other violations for the outage of a single line all follow the same pattern, and they all stem from the redistribution of flows from the generator at bus 53139.

In summary, the single-outage contingency analysis shows problems that affect

small lines and generators. Most of the violations occur on small lines that carry only a relatively small amount of power. The outages that affect larger lines involve the redistribution of flows away from generators. Typically, the outage of one of multiple lines connecting a generator to the system results in an overload in another line connecting the generator to the system. These situations should be easily fixable by a relatively minor redispatch. Simply backing down the generators that no longer have adequate transmission capacity to supply their power to the grid should relieve the overloads. However, since any information about these schemes is absent from the case, these contingencies must still be considered important.

5.2 Double-Outage Contingency Analysis

The ultimate goal is to be able to tell which contingencies threaten the system in order to provide a way to gauge the accuracy of the screening results. The screening process can be said to be successful if the algorithm correctly picks out the severe contingencies without flagging a large number of nonthreatening events as potentially dangerous. For completeness, the double-outage contingency analysis is divided into two sections. The first section discusses the full set of data. All of the violations are considered, no matter how small the line is on which they occur. The second section considers a filtered set of data, from which the smaller elements (violations on lines with a limit of 100 MW or less) have been removed.

The double-outage contingency analysis yields a huge amount of data. One of the biggest challenges in dealing with nearly 29 million contingencies is processing the data into a useful form.

For the double-outage contingency set, there are 549 119 contingencies that

result in violations. There are a total of 791 092 violations. The contingencies that result in islanding are removed from the results as was done with the single-outage results. When the results are filtered by removing the violations on lines with limits less than 100 MW, there are a total of 112 049 contingencies that result in violations. This is 20.4% of the contingencies that resulted in violations before the data were filtered, indicating that most of the violations occur on lines with limits less than 100 MW.

Another important set of double-outage contingencies to consider are the contingencies that result in violations only when both lines are outaged. If only one or the other of the lines is outaged, there are no resulting violations. A list of these violations can be constructed using the single-outage results along with the double-outage results. By searching the single-outage contingency analysis results while processing the double-outage contingency analysis results, it is possible to see whether one of the lines involved in the double-outage contingency also results in a violation when it is outaged by itself. When the large-case results are examined, there are 1851 double-outage contingencies that have violations only when both elements are outaged.

To process these data and determine which contingencies threaten the system, contingency-ranking methods are applied. First, the PI methods [27] are applied to the data to determine the severe contingencies. The contingencies are also ranked using the method in [48], which generates an index by building a matrix of the lines and contingencies and summing the rows and columns.

The traditional PI methods sum a performance index over the violations. For every violating line, sum the performance index. The performance index has the form

$$\mathrm{PI}_{MW} = \sum_{l=1}^{L} \frac{W_l}{2n} \left(f_\alpha / f_\alpha^{Max} \right)^{2n} \tag{5.1}$$

as originally given in (4.1). The weighting function was chosen to be 0.5 and n was picked to be 1.0, corresponding to the choices in the literature. Substituting these values gives the expression for the PI as

$$\mathrm{PI}_{MW} = \sum_{l=1}^{L} \left(f_\alpha / f_\alpha^{Max} \right)^2 \tag{5.2}$$

Performance indices range between 0.5 and 3.24. The double-outage contingency analysis was solved using PowerWorld Simulator [49]. Every contingency that resulted in a violation was recorded along with the line the violation(s) occurred on and the magnitude(s) of the violation(s). For every single contingency that resulted in violations, the performance index (PI) was evaluated. The analysis results in 613 236 contingencies with violations. The minimum performance index is 1.0, and the maximum performance index is 86. Figure 5.1 has a plot of the distribution of the performance indices. The distribution shows that most of the performance indices are less than 10. The average PI is 2.29, and the standard deviation is 1.68. Table 5.2 is a table of the largest PI values.

Table 5.2 shows that the maximum performance indices decay rapidly. This trend can be shown in more detail in the density plot of the PI values, which is shown in Figure 5.2. The density plot was created from 613 236 PI values, which were calculated from the contingency analysis results. This figure shows how the PI values are clustered, i.e., what the most common PI values are. The largest value is 86.09, which is the largest horizontal value the density plot takes. The smallest horizontal value is 1.0, which corresponds to the smallest PI value. The density plot (Figure 5.2) clearly shows that the PI values are clustered near 1.0, with most of the values falling between 1 and 7. The PI values in Table 5.2 extend over most of the horizontal range [88.09, 16.9] in Figure 5.2, indicating

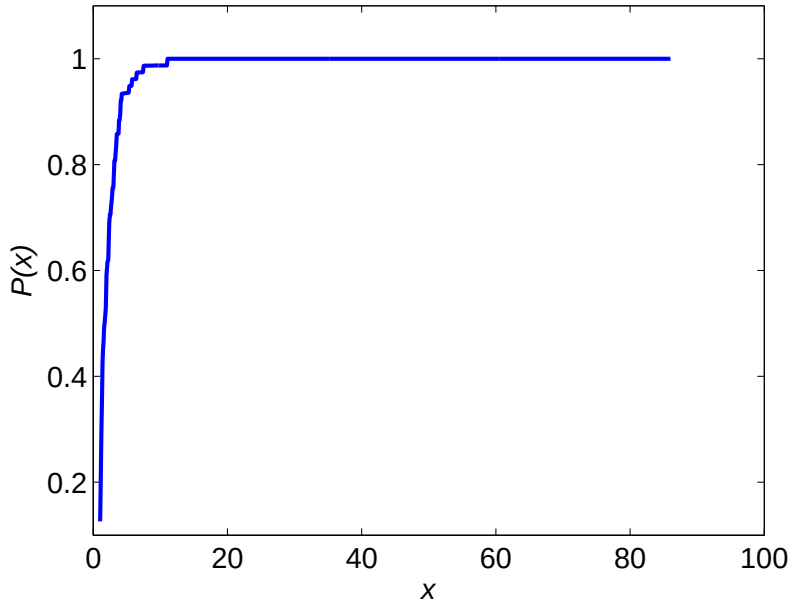

Figure 5.1: PI distribution

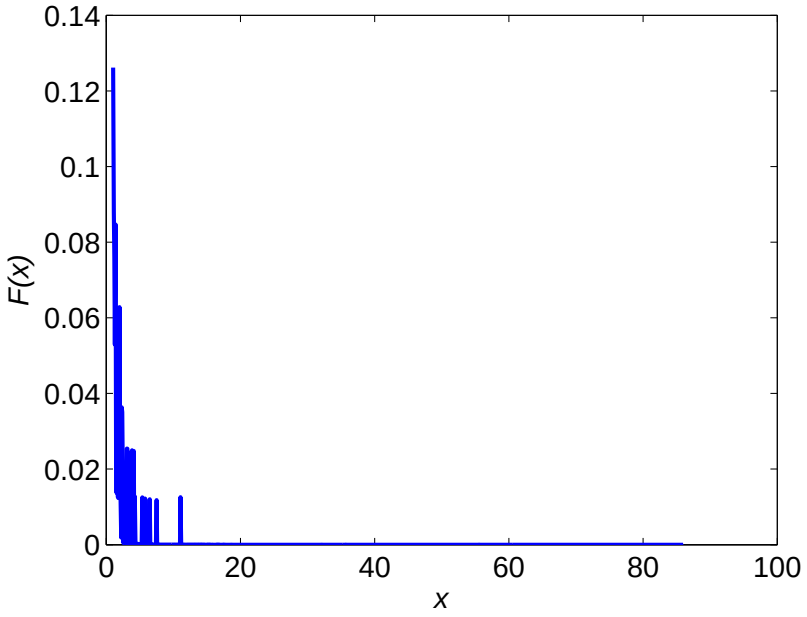

Figure 5.2: PI density

that these are by far the most extreme values. The low value of the density plot – the nearness of the plot to the horizontal axis – shows that there are very few values in this range.

Table 5.2: Largest 10 PI values

PI rank	PI value	Line 1	Line 2
1	86.09	25388 to 25181	26557 to 25181
2	55.62	27618 to 33352	31023 to 33362
3	35.69	27313 to 27002	27323 to 27002
4	32.1	34182 to 34206	34182 to 34206
5	21.65	757 to 758	758 to 759
6	21.26	757 to 759	760 to 40114
7	18.63	758 to 759	27323 to 27370
8	17.61	758 to 759	25388 to 25181
9	17.49	25388 to 25181	27323 to 27370
10	16.9	465 to 26943	758 to 759

5.2.1 Analysis of large PI values

The next several paragraphs study some of the contingencies that result in the largest PI values in depth. In particular, the contingencies that have the highest three PI values are explored. The violations are listed, and the reasons for the large PI values are considered. The topology around the contingent lines is shown to give insight about why the PI values are so high.

The maximum performance index for a double-outage contingency occurs for the outage of the line connecting bus 25388 to bus 25181 and the line from bus 26557 to bus 25181. These two lines are both connected to the same bus (25181), which has one other line connected to it as well as 120.9 MW of load. The topology near this bus is shown in Figure 5.3. The contingent lines are are drawn using dashed lines, and the lines with violations are drawn with pie charts showing their precontingent loading. Clearly, in this situation the outage of the two contingent lines will force the load to be served by the third line, which has

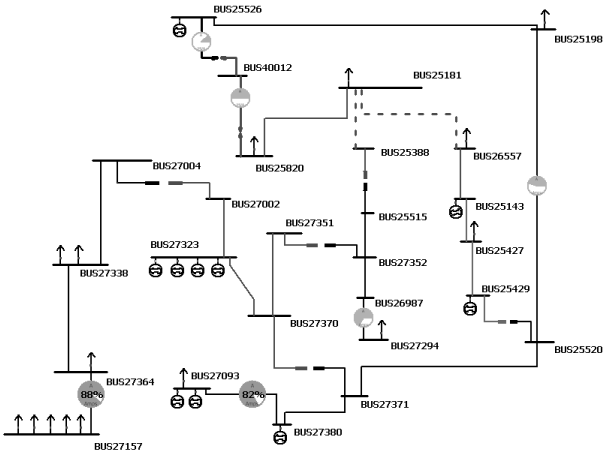

Figure 5.3: Topology at bus 25181

a limit of 40 MW, and this will result in a large overload. In fact, the ac solution will diverge if an attempt is made to solve this contingency. The topology shown in Figure 5.3 gives insight as to why the contingency results in so many severe violations. The oneline diagram in Figure 5.3 shows the topology around the contingent and violating lines. Note that there are other lines connecting these buses to the rest of the system. The first observation that can be made is that the violating lines and the contingent lines are very near to each other. In fact the two contingent lines (drawn with dashed lines) are both connected to the same bus. The violating lines are also nearby.

The two contingent lines are serving a load pocket composed of bus 25820 and bus 25181. When these lines are outaged, the power to serve the load must be supplied along the path from bus 25526 to bus 40012 to bus 25820, and this path does not have sufficient capacity to supply the load pocket. The other violations are more difficult to explain. However, the single outage of bus 25181 to bus 26557 results in violations on the line from bus 27364 to bus 27157 and the line between bus 26987 and bus 27294. The overloads are caused because the outage forces more power to flow along these lines in order to continue serving the load. The overload of the line between bus 25198 and bus 25520 is

unique because it is the only overload that is heavily dependent upon the outage of both lines. All of the other overloads occur (or nearly occur) when only one of the contingent lines is open. This overload appears to be because the two lines are joining two coupled areas of the system, and the outage of both lines forces the power flows between the coupled areas onto the line between bus 25520 and bus 25198.

The violations that result from the outage of the line from bus 25388 to bus 25181 and the line from bus 26557 to bus 25181 are listed in Table 5.3. There are six violations that are included in the largest PI value. The PI was calculated using (5.2), so each of the violations in Table 5.3 contributed to the final PI value of 86.09. However, the largest overload value, 663%, results in a contribution PI value of 43.96. If this violation were considered alone, it would result in the third higest PI in the system. The second most severe violation (598%) for this outage contributes 35.82 to the PI value. The rest of the violations contribute 6.32 collectively. The two largest violations dominate the PI calculations.

Table 5.3: Violations contributing to PI of 86.09

Violating Line	Flow(MW)	Limit(MW)	Overload(%)	Nominal voltage(kV)
25526 to 40012	165.75	25	663	138
40012 to 25820	155.6	26	598.44	69
27294 to 26987	48.29	32	150.91	138
27364 to 27157	26.79	22	121.77	138
25520 to 25198	226.05	186	121.53	138
27093 to 27380	130.82	126	103.83	138

The second largest PI value (55.62) result from the outage of the line connecting bus 27618 to bus 33352 and the line connecting bus 31023 to bus 33362. The five violations that contribute to this PI value are listed in Table 5.4. The largest violation for this contingency results in a line loaded at 565.15% of its limit, which results in a contribution of 31.94 to the PI value. The second

largest violation is 441.17%. This violation contributes 19.46 to the PI value. The other violations contribute 4.22 to the PI value.

Table 5.4: Violations contributing to PI of 55.62

Violating Line	Flow(MW)	Limit(MW)	Overload(%)	Nominal voltage(kV)
33356 to 264	237.36	42	565.15	69
33361 to 33356	247.06	56	441.17	161
27555 to 27619	68.42	50	136.84	161
33351 to 33352	280.74	249	112.75	161
33352 to 33353	258.4	249	103.77	161

The third largest PI value is significantly smaller than the largest two values. The PI value is 35.69. This PI value is 50.4 less than the largest PI and 19.93 less than the second largest value. The third largest PI value results from violations from the outage of the line connecting bus 27313 to bus 27002 and the line connecting bus 27323 to bus 27002. The violations created by this contingency are listed in Table 5.5. The 570.62% overload contributes 32.56 to the PI, and the 176.99% overload contributes 3.13.

Table 5.5: Violations contributing to PI of 35.69

Violating Line	Flow(MW)	Limit(MW)	Overload(%)	Nominal voltage(kV)
27157 to 27364	125.54	22	570.62	138
570 to 27157	141.59	80	176.99	138

Several conclusions can be drawn from the examination of the large PI values. First, it may be observed that the largest PI values result from extremely large overloads. This makes sense because (5.2) penalizes overloads quadratically. These large overloads or violations that result in large PI values tend to dominate the PI calculation. The large-overload contributions are larger than 99% of the PI values. It does not appear that there are large numbers of small violations that are combining to form a large PI value. The largest number of violations caused by any single contingency is 7, and the corresponding PI value

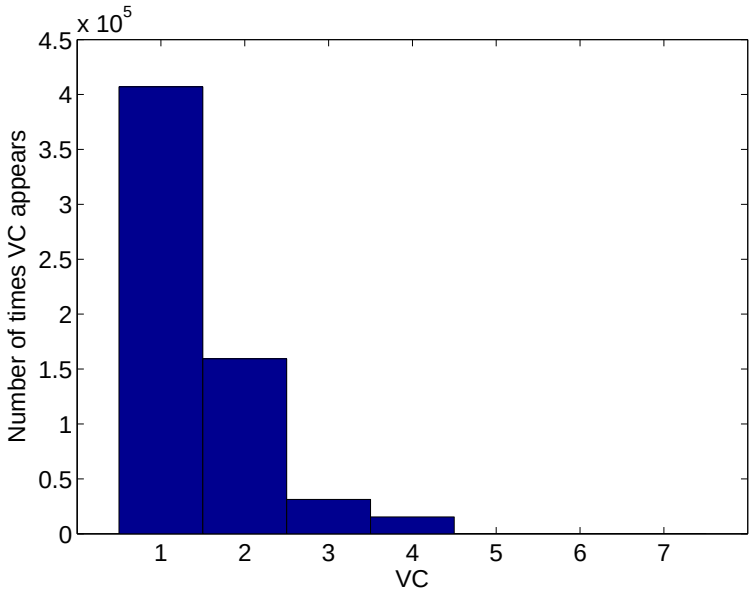

Figure 5.4: Violations per contingency histogram

is only 9.86, which is above average but far from the largest value. A histogram of the number of violations per PI is shown in Figure 5.4. This histogram shows the number of violations per contingency for each of the contingencies that cause violations. The masking problem mentioned in [29] does not appear to be a problem for this system. There is no case in which a large number of small outages add to a large PI.

In summarizing the information given by the largest PI values, it is useful to ask, "What is the PI telling us about the contingency?" To answer the question, it may first be observed that the PI method penalizes overloads quadratically and the number of overloads linearly. Large overloads are heavily penalized, whereas the additional violations only add incrementally to the final PI value. This means that the masking problem would require a large number of violations to appear as large as a single large limit violation. The data show that this does not happen. It may also be useful to observe that if the PI is calculated summing over all lines, the value would be swamped. Even if the lines

are only loaded at a fraction of their limits, summing 7616 small values swamps out the contributions from the violating elements.

5.2.2 PI contingency list

The ultimate goal of using the ranking method is to tell which contingencies are the worst. Since the only lines that have a PI value are lines with violations, in some sense they are all severe contingencies. The PI method allows for discriminating between the severity of the contingencies.

By examining the PI data, it can be found that 90% of the contingencies with violations have PI values less than 4.02. This value is chosen as the threshold between severe and not severe. There are 62 785 contingencies with PI values greater than 4.02. In terms of the distribution plots, this selection criterion picks out the contingencies that fall to the right of the value 4.02 in the distribution plots.

5.2.3 Filtered PI contingency list

A contingency list may also be produced using the filtered PI data (the data from which contingencies with violations on lines less than 100 MW have been removed). When a list is constructed using the same 90% criterion, the threshold PI value is 4.1056. When the contingeinces that have PI values greater than 4.1056 are selected, the resulting contingency list has 49 504 elements.

5.3 Sorted-Matrix Ranking

The idea behind sorted-matrix ranking is to organize the contingency results in a matrix as shown in Figure 4.1. One dimension of the matrix is equal to the

number of lines in the system. The other dimension is equal to the number of contingencies. Entries are made into the matrix when a violation occurs. When a violation occurs, an entry is made into the matrix. For large systems, the matrix is too large to visualize. For example, a system with 7616 lines and 700 000 violations would require a monitor with a resolution of 7616 by 700 000 in order to represent each violation as a single pixel. These resolutions are well beyond the capabilities of modern monitors. Therefore, instead of looking at pictures, the aggregate values are used. In particular, the distributions for the CAG and the LAG are used to analyze the contingency analysis results on a per-line and a per-contingency basis.

5.3.1 Per-line analysis

The following paragraphs examine the double-outage contingency results for the large case on a per line basis. In other words, we will examine the LAG values calculated for every contingency that results in at least one violation. The data show that a large number of the violations appear on just a few lines, which is interesting because it indicates that these lines are weak points in the system. The data set is examined using several different filters. The filters cause lines with a nominal limit below a specified threshold to be ignored. For example, if the threshold is set to 100 MW, data on lines with a limit below 100 MW are filtered out and not considered in the analysis.

Figure 5.5 contains the number of violations per line for various levels of filtering (this is essentially an unnormalized distribution plot). There are four lines in Figure 5.5 corresponding to no filtering and to removing data for lines with limits less than 100 MW, 200 MW, and 300 MW. Figure 5.5 shows that an extraordinary number of violations occur on some lines. There are 280 727

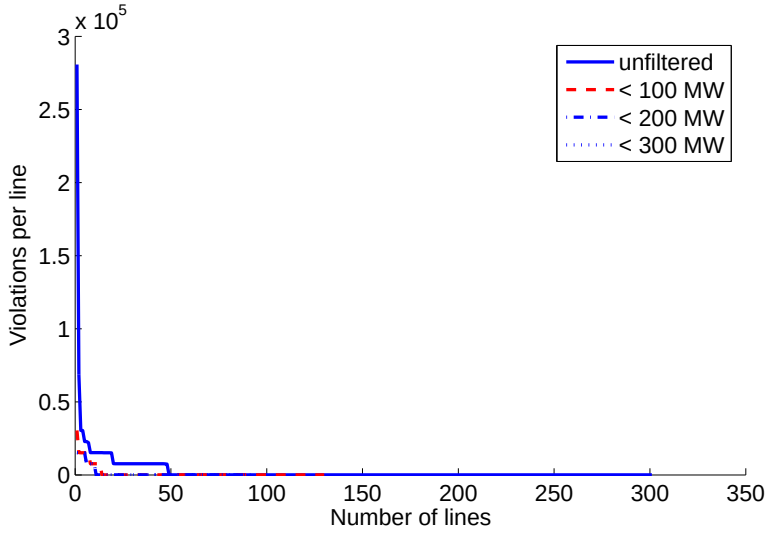

Figure 5.5: Number of violations per line

violations that occur on line 4468, which connects bus 27289 to bus 27153. The figure of 280 727 violations represents 31.82% of the total number of unfiltered violations. The average percentage over limit is 21.1%. Line 4468 has a limit of 240 MW, and a base case loading of −71.5 MW.

Figure 5.6 shows a close-up view of Figure 5.5. The close-up view shows more detail around the origin. This makes it easier to see the effect that the filtering has. Unsurprisingly, the filtering lowers the total number of violations, which explains why removing more data from consideration lowers the values of the plots. As the filtering limit is increased, more data is removed because only lines with larger limits are still included.

Figure 5.7 contains plots of the cumulative percent overload, i.e., the percentage over limit, summed for a line over all violations. This corresponds to summing the columns in Figure 4.1. The data in Figure 5.7 are closely related to the data in Figure 5.5, so it is not surprising that the plots are similar. Figure 5.7 shows that there are about 175 lines with very large cumulative percent overload values. The curves in Figure 5.8, a close-up version of

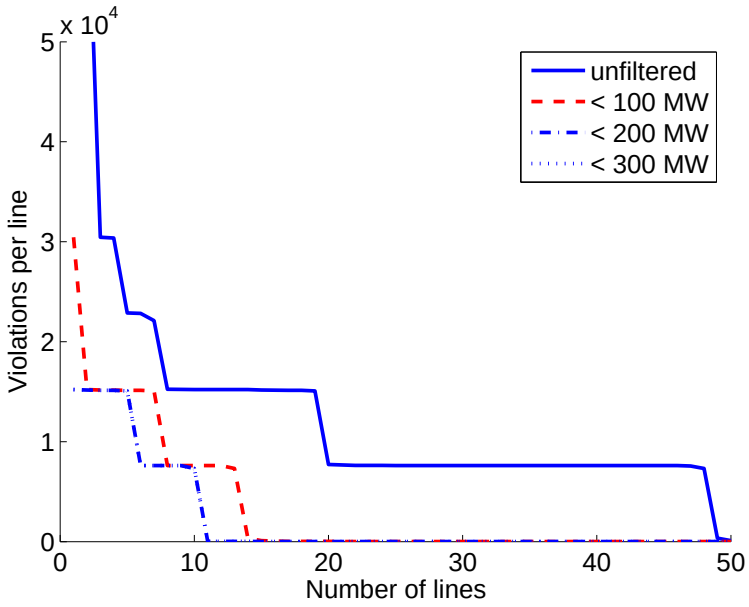

Figure 5.6: Number of violations per line, close-up

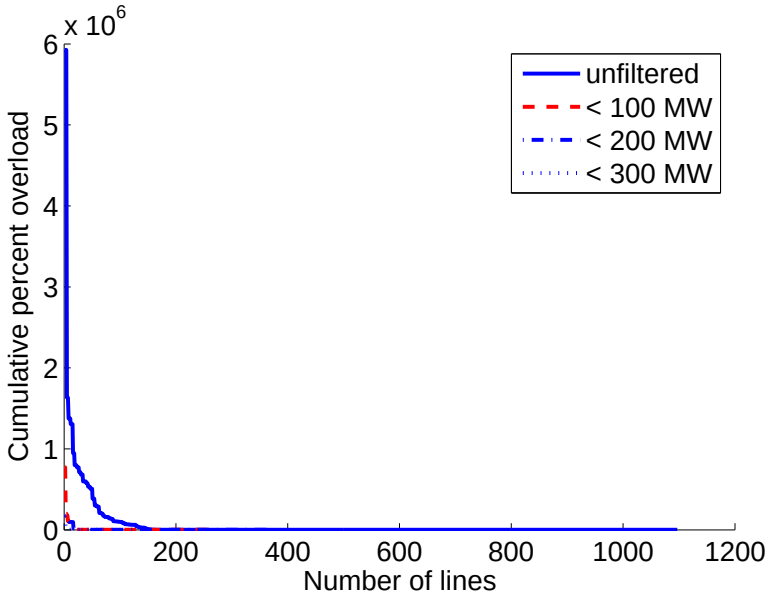

Figure 5.7: Cumulative percent overload per line

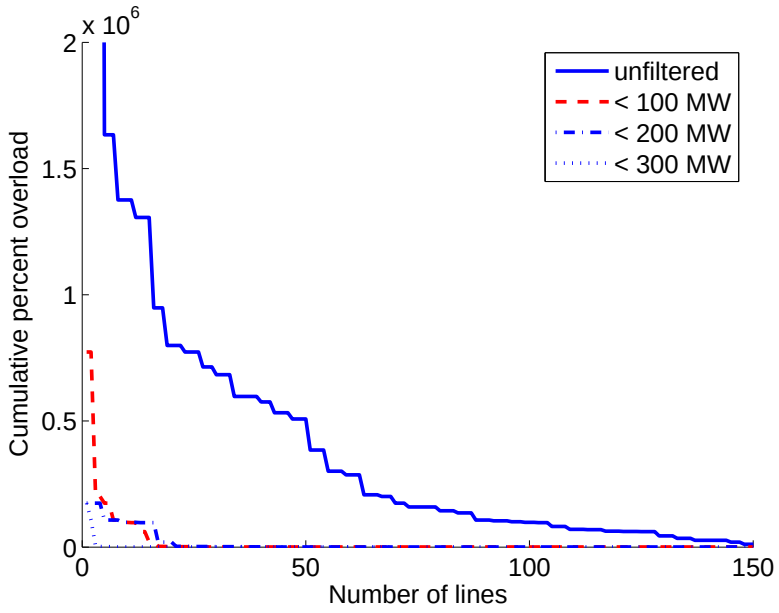

Figure 5.8: Cumulative percent overload per line, close-up

Figure 5.7, show a large difference between the filtered and the unfiltered data. The clustering of the curves for the 100 MW and the 200 MW filters is also much easier to discern. The 100 MW threshold curve shows a large spike near the vertical axis. The filtering seems to cut the top off the spikes near the vertical axis. If the unfiltered and the 100 MW filter curve are examined, it can be seen that the 100 MW curve is much lower. This means that the lines with a cumulative line overload value tend to have lower limits. The trend continues for the curve corresponding to the 200 MW cutoff value, which has an even lower peak.

In examining the violations in a per-line basis, it is also interesting to see how the filtering affects the total number of violations. Table 5.6 contains the number of violations for varying the cutoff. From examining this table, it is clear that most of the violations occur on lines with low limits. Removing lines with limits below 100 MW removes 714 658 violations from the data set. It can also be observed that there are three plateaus in the data. For limits between

100 MW and 200 MW the number of violations is between 113 804 and 167 608, and for limits between 250 MW and 400 MW the values are in the tens of thousands (60 691 to 83 602). Finally, for lines with large limits (450 MW to 500 MW) the number of violations stays constant at 48.

Table 5.6: Violations contributing to PI of 55.62

Violations	Lower limit (MW)
882 266	none
167 608	100
114 328	150
113 804	200
83 602	250
83 598	300
68 336	350
60 691	400
48	450
48	500

The per-line aggregates are very useful for determining what lines are weak in the system. In the large case studied here, a single line has over 30% of the violations, and they are significant violations. The average overload is over 20%, meaning that on average this line is overloaded over 120% of its limit. Clearly, this is a weak element in the system. As the figures of the per-line aggregate overload and the number of violations on a line are considered, it is easy to see that the nearer the vertical axis, the weaker the line.

5.3.2 Per-contingency analysis

The previous section discussed the LAG values, which are calculated for a line by summing over the contingencies. This section examines the CAG values. These values are calculated for a contingency. They indicate the aggregate overload for all lines in the system that a contingency causes. The trends in these data match the trends in the line-based aggregate values. However, the

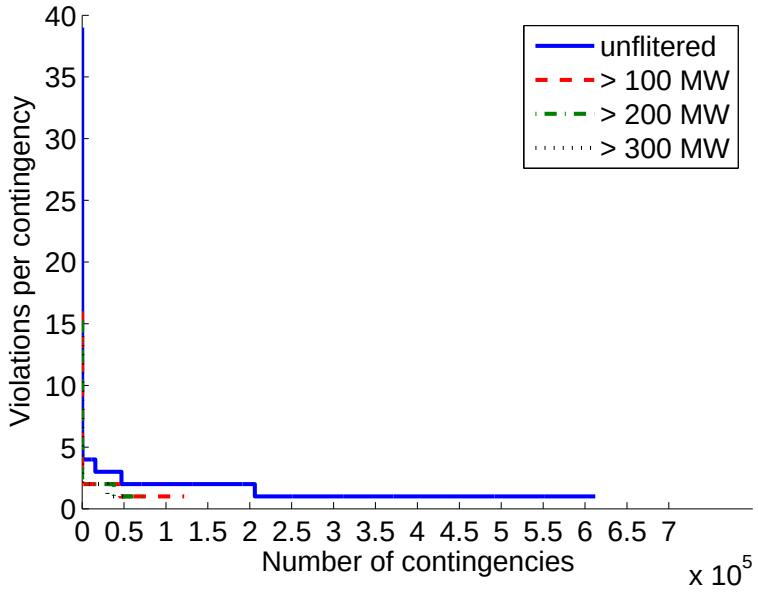

Figure 5.9: Number of violations per contingency

number of violations caused by a single contingency is much lower. The maximum number of violations caused by any single contingency is 39. These violations are caused for the outage of the line from bus 464 to bus 483 and the line from bus 99900 to bus 99997.

Figure 5.9 contains a sorted plot of the number of violations per contingency. As mentioned above, a single contingency results in 39 violations, which corresponds to the far left-hand side of the figure. This figure shows that very few contingencies result in a large number of violations. As the plot of the unfiltered data shows, the number of violations per contingency quickly drops to 4. The curves for the filtered data have a maximum value of 2, indicating that the contingencies that result in large numbers of violations impact low-limit lines. A close-up view of Figure 5.9 can be found in Figure 5.10. An examination of the vertical axis of the close-up view shows that there are a few contingencies that have 5 violations on lines with limits of 100 MW or more. The main difference between the various filtering thresholds is the number of

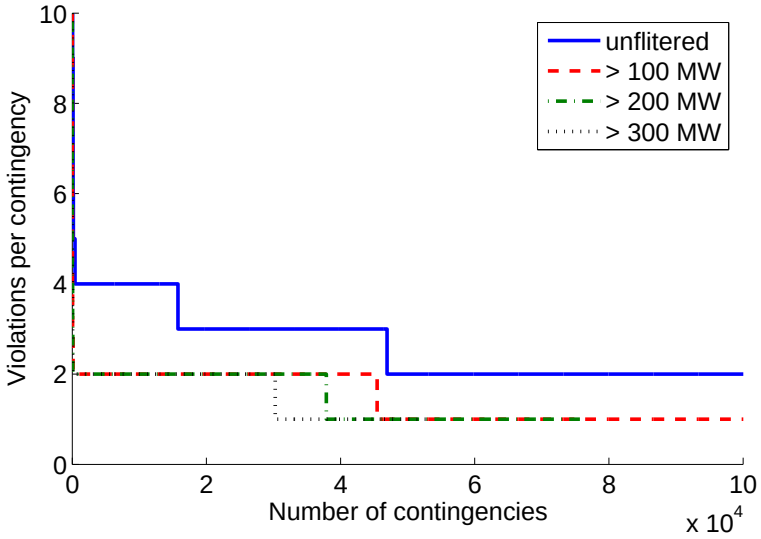

Figure 5.10: Number of violations per contingency, close-up

contingencies. The plots show that the higher the filtering threshold, the smaller the number of violations caused by a single contingency.

To generate a list of severe contingencies from the sorted-matrix results, a list was created by picking out the contingencies with the highest CAG values. Specifically, the contingencies that had CAG values in the top 50 percentile were chosen to form the list of severe contingencies. This list contains 986 contingencies. Of course, this list is not a complete list of contingencies that threaten the system — every contingency that results in a violation does that — but the list does capture the most severe contingencies selected by the sorted-matrix ranking method.

5.4 Conclusions

This chapter examined the single-outage and double-outage contingency results for the large case. The point of this chapter is examine the results and determine which double-outage contingencies threaten the system. The single-outage

results are included because they will also show up in the double-outage results. Several methods were used to develop lists of severe double-outage contingencies. The first approach is to create a list that contains every double-outage contingency that results in violations. This is the most conservative method, and it does not differentiate between the severities of contingencies. Several other techniques are applied to the full list of results in order to determine which contingencies are the "worst of the worst." The methods described in Chapter 4 were used to pick out the worst contingencies from the full set of contingencies that cause violations. These lists provide the basis for judging the effectiveness of the screening algorithms developed in Chapter 6.

CHAPTER 6

CONTINGENCY SELECTION ALGORITHMS

6.1 Introduction

Considering multiple-outage contingencies is becoming increasingly important because of the way the transmission grid is being used today. In the deregulated environment, the transmission system is being utilized in ways the designers never contemplated. Combined with continual load growth, the change in system operations has resulted in a transmission system that is increasingly stressed. One way this is being dealt with is through the introduction of new standards requiring system operators to meet performance requirements in the event of multiple-element outages [5].

This chapter explores the use of several algorithms to reduce the computational burden of processing the large number of multiple-outage contingencies. These algorithms use differing amounts of information about the system to generate lists of potentially severe contingencies. At one extreme, only topology information is used. At the other extreme, it is assumed that line limit and flow data are also available. The various algorithms are presented in turn, and the properties of each algorithm are discussed.

The algorithms function in two parts. The first part generates a structure that keeps track of the impact of the lines upon each other. The impact-tracking structure (ITS) is created using only linear sensitivities to measure the impact of lines upon each other. The flow-tracking structure (FTS) is created by

quantifying the impact on a line using the flow change that results from an outage. The limit-tracking structure (LTS) is created by measuring the impact of an outage using the percentage change in line limit resulting from an outage. The overload-tracking structure (OTS) records entries based on a padded distance to limit violation. Each of these tracking structure is described in detail in the following sections.

The second stage of the algorithm generates a list of unique contingencies from the tracking structure. Two basic techniques are used to generate the list of contingencies. The first method works by generating the combinations of the row elements in the tracking structure and removing the nonunique elements. The reasoning behind this method is that the resulting list of contingencies will contain all the double-outage combinations that impact a given line. The second method, used only with the OTS, works by generating the combinations of the tracking structure entries with every other line in the system. The logic behind this method is that, when a single outage brings the system very near to a postcontingent violation, that outage will also create a violation when it is paired with an outage of most of the other branches in the system.

The algorithms are applied to the large case, and their success is judged against the full contingency analysis results discussed in Chapter 5 as well as other sets of serious contingencies that were flagged using several contingency-ranking methods. The results of the screening algorithms are compared to two sets of results for each ranking method. One set is based on a filtered set of contingency results, and the other is based on the complete double-outage contingency analysis results.

The results show that the OTS is very good at detecting double-outage contingencies that result in violations from what are essentially the single-outage contingency analysis results. The other algorithms perform much worse,

although, the ITS is good at detecting double-outage contingencies in which a violation occurs only when both lines are outaged. Finally, conclusions about the strengths and weaknesses of the various algorithms are discussed.

The screening algorithms are broken into two separate parts. The first part generates a tracking structure, and the second part generates a list of contingencies. The ITS, FTS, and LTS all use the same contingency generation algorithm. The OTS uses a different method to generate a list of contingencies from the tracking structure. The goal of all these algorithms is to processes double-outage contingencies while avoiding the $O(n^{3.2})$ computational order.

This chapter begins with an introduction of the ITS screening algorithm, which is based on single-outage LODF values. The connection between the ITS screening algorithm and the condition number of $\mathbf{M}$ is discussed. After the ITS screening algorithm is demonstrated for a small case, the algorithm is modified through the introduction of new information. The inclusion of differing amounts of information results in the FTS, LTS, and OTS screening algorithms. The results of the screening algorithms are discussed in Chapter 7.

6.2 Screening Algorithms

The ITS screening algorithm is broken into two separate parts, described in the following subsections. The goal of these algorithms is to processes double-outage contingencies while avoiding the $O(n^{3.2})$ computational order mentioned in the introduction. To reduce the computational time, a structure is built using single-outage LODF values, which only requires the computation of L^2 LODF values, and has a computational order of $O(n^{2.2})$. The structure tracks the lines that impact each other while ignoring the lines with little influence. This takes advantage of the fact that most lines in a large system have little impact on each

other. By neglecting the lines with little impact on each other, a very sparse structure that tracks the impact of lines onto each other is generated.

6.2.1 Impact-tracking structure construction

The impact-tracking structure (ITS) is a structure designed to track the impact of lines onto each other. The structure is composed of a list of lists. Every branch in the system has a list that contains the lines whose outage has the ability to change the flow on that branch. The LODF is used as the measure of impact. This structure is illustrated in Figure 6.1.

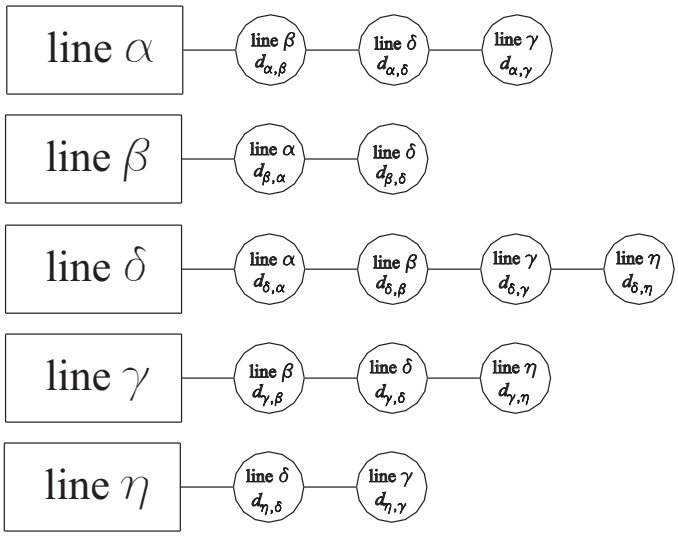

Figure 6.1: Impact-tracking structure

The algorithm for constructing the ITS is given in Algorithm 1. The algorithm is a modification of single-outage contingency analysis using LODFs. The algorithm calculates the LODFs for the outage of every line in the system and records the values above a given threshold. Single-outage contingency analysis would also use the flow and limit information to calculate a postcontingent flow. The result is a structure in which every line in the system is associated with a list of lines that impact it.

The threshold d^* is a an input parameter that specifies a lower bound for recording an impact. Values below d^* are not added to the ITS. Varying the value of d^* will change the number of elements that are added to the ITS. The larger the value of d^*, the fewer values are added to the ITS. The size of the ITS will impact the amount of time it takes to generate a contingency list. The impact on the run time is discussed further below.

Input: List of lines, LODF threshold d^*
Output: ITS
foreach *Line* α **do**
  foreach *Line* β **do**
    if $\alpha \neq \beta$ **then**
      Calculate $d_{\alpha,\beta}$ **if** $|d_{\alpha,\beta}| \geq d^*$ **then**
        Add entry at Row α for line β;
      end
    end
  end
end

Algorithm 1: ITS construction algorithm

It may be noted that an ITS with a larger value of d^* can be generated from an ITS with a lower cutoff value by simply removing the elements with an LODF below the new cutoff. This fact can be used to eliminate the need to repeatedly calculate the L^2 LODF values required to construct the ITS. The trade-off is a larger ITS. In the most extreme case, d^* may be set to zero. In this case, every one of the L^2 LODF values will be recorded. This will result in an extremely large ITS and the algorithm to generate the contingency list will become extremely slow (although the result can be anticipated: the algorithm will simply return every double-outage contingency). Engineering judgment, knowledge of the system, and the study requirements must be used to choose an appropriate value for d^*. For the studies presented in this paper, the NERC flowgate cutoff of 5% [50] was used as a lower bound for d^*. For a well-chosen value of d^*, the ITS is a very sparse structure, which reflects the fact that the

outage of a line typically affects only a few lines in a large system.

A sample of the tracking structure is shown in Figure 6.1. This figure illustrates how each line is associated with a list of lines, which have been added based on their LODF values. Only lines whose impact is larger than the threshold are added to the list. For example, the first row shows that line α is impacted only by three other lines (line β, line δ, and line γ). This means that only these three lines impact line α above the cutoff threshold. Although the outage of other lines in the system would impact the flow on line α, the amount of the impact — measured using LODF values — is small.

6.2.2 ITS contingency selection algorithm

Once the ITS has been constructed, contingency selection is a two-step process. First, every possible double-outage contingency is generated for each row of the ITS. Second, the nonunique outages are removed from the list. When the algorithm terminates, a list of flagged contingencies has been generated in which every contingency generated by the process involves lines that both impact a third line. The list is dubbed FL for flagged list. The algorithm for generating FL from the ITS is given in Algorithm 2. This algorithm is also used for generating contingencies from the flow-tracking structure and the limit-tracking structures.

Input: Impact-Tracking Structure ITS
Output: List of Flagged Contingencies FL
foreach *Row r in the ITS* **do**
    Generate all combinations of the elements in r;
    Add combinations to FL;
end
Remove nonunique elements from FL.

Algorithm 2: List generation algorithm

For the example ITS shown in Fig. 6.1, generating the combinations for the

first row would generate

$$\binom{3}{2} = 3 \tag{6.1}$$

double-outage contingencies. For the entire structure, generating the combinations of each row results in 14 double-outage contingencies. However, these 14 contingencies are not unique, because the same contingency shows up more than once. For example, the contingency involving line β and line γ is generated for the first and third rows. This means we must remove the nonunique elements from the flagged list.

Removing the nonunique elements from the list FL is the final step in the screening algorithm. There are several approaches to remove the non unique elements. However, care must be taken to choose an efficient method. A naive approach can easily result in a factorial time algorithm to remove the nonunique elements, and this must be avoided.

One efficient approach is to sort the list, then pass through once recording unique elements. Sorting is a heavily studied problem in computer science, and many efficient sorting algorithms exist [51]. The final pass-through, in which the nonunique elements are removed, is clearly linear time.

Another even more efficient approach is to use data structures that allow only unique elements to be inserted. These structures are maintained in a sorted state, so the sorting and comparing mentioned above is automatically performed upon insertion to the structure. When an item is added, a binary search is used to determine whether the item already exists in the structure. If the element is not already in the structure, it is added. Otherwise, it is discarded. The searching can be done efficiently because the structure is maintained in a sorted state. This kind of structure is known as a sorted associative container [52]. A standard implementation of this type of container is distributed as part of the

standard template library (STL) [53].

To examine the computational effort of generating the list of flagged contingencies, we need to know the average row length of the ITS, because it will determine the number of outages generated by the combinatorics, which in turn determines the number of insertion operations needed. If we define average row length to be R, then generating every double-outage combination has a computational effort $O(R^2)$. Insertion into a sorted associative container is $O(\log(R))$ in the worst case [52], and there will be R^2 insertions. This gives a computational complexity of $O(\log(R^2)$. Using the properties of logarithms,

$$O(\log(R^2) = O(2\log(R)) \tag{6.2}$$

In practice, the computational effort to run Algorithm 2 depends on the choice of d^*, which is ultimately what determines R. There is no simple relationship between d^* and R, although it can be observed that as d^* decreases, R increases.

6.2.3 Condition numbers and the ITS

The condition number was developed as a metric of the interaction of outaged lines in Chapter 3. This section discusses the relationship between the ITS and the condition number. The ITS is closely related to the condition of $\mathbf{M}$. In fact, the ITS is an algorithm that is guaranteed to generate every contingency with a condition number above a given threshold. The threshold is given by

$$\kappa^* = \kappa\left(\begin{bmatrix} 1 & -d^* \\ -d^* & 1 \end{bmatrix}\right) \tag{6.3}$$

Thus, picking the value of d^* picks κ^*. With this in mind, it can be seen that the ITS screening algorithms are really a contingency number–based screening

algorithm. The ITS screening algorithms are more efficient because they do not require the calculation of every condition number, but the results are the same as those of filtering by the condition number. The condition number also gives some insight to the behavior of the ITS screening algorithm. As discussed in Chapter 7, the ITS screening algorithm is good at detecting double outages that result in violations only when both elements are outaged. These types of violations tend to occur in situations where the outaged lines have an impact on each other, and the condition number is a way of measuring the interaction of the outaged lines.

6.2.4 IEEE 14-bus example system

The IEEE 14-bus test case [24] was chosen to illustrate the screening algorithms because it is small enough to be manageable and large enough that all the lines do not impact each other. The IEEE 14-bus test case has 20 lines, which are given unique names in Fig. 6.2. These names are used to identify lines in the impact-tracking structure, shown in Figure 6.3. They are also used in Table 6.1 to identify lines in the LODF matrix. A one-line diagram for the IEEE 14-bus system is shown in Figure 6.2.

The algorithms presented in Section 6.2 were applied to the IEEE 14-bus test case system. The impact-tracking structure that is generated is shown in Figure 6.3. For this particular structure the LODF cutoff, d^*, is 10%.

The ITS construction algorithm can be illustrated using LODF values for the IEEE 14-bus system, given in Table 6.1. The first row, corresponding to line 1, contains the LODF values for the outage of lines 2, 3, 4, 5, 6, and 7. For example, row 1, column 3 contains $d_{line\ 3,\ line\ 1}$. The algorithm works by proceeding down the row and adding an entry to the list corresponding to the

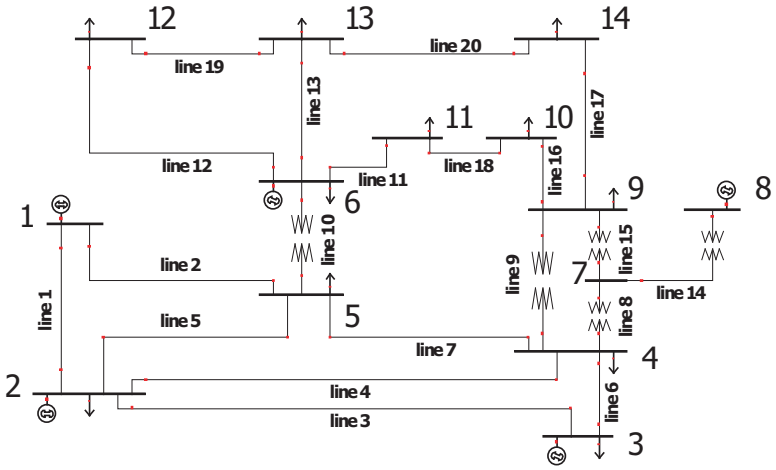

Figure 6.2: IEEE 14-bus one-line diagram

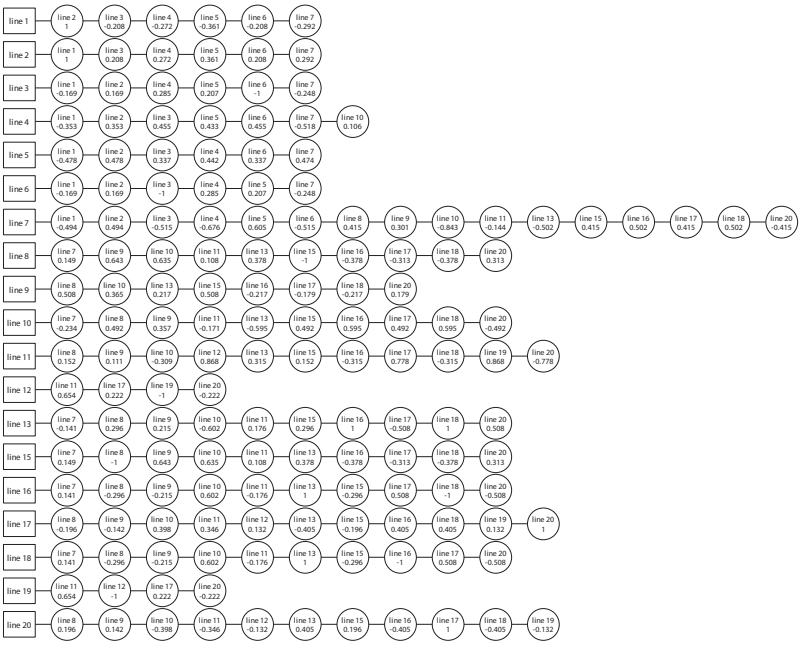

Figure 6.3: IEEE 14-bus ITS

Table 6.1: IEEE 14-bus LODF matrix

line	1	2	3	4	5	6	7	8	9	10	11	12	13	15	16	17	18	19	20
1	**-100**	**100**	**-16.9**	**-35.3**	**-47.8**	**-16.9**	**-49.4**	-1.8	-1.0	2.8	1.7	0.3	0.9	-1.8	-1.7	-1.1	-1.7	0.3	1.1
2	**100**	**-100**	**16.9**	**35.3**	**47.8**	**16.9**	**49.4**	1.8	1.0	-2.8	-1.7	-0.3	-0.9	1.8	1.7	1.1	1.7	-0.3	-1.1
3	**-20.8**	**20.8**	**-100**	**45.5**	**33.7**	**-100**	**-51.5**	-1.9	-1.1	2.9	1.8	0.3	0.9	-1.9	-1.8	-1.2	-1.8	0.3	1.2
4	**-27.2**	**27.2**	**28.6**	**-100**	**44.2**	**28.6**	**-67.6**	-2.4	-1.4	3.8	2.3	0.3	1.2	-2.4	-2.3	-1.5	-2.3	0.3	1.5
5	**-36.1**	**36.1**	**20.7**	**43.3**	**-100**	**20.7**	**60.5**	2.2	1.3	-3.4	-2.1	-0.3	-1.1	2.2	2.1	1.4	2.1	-0.3	-1.4
6	**-20.8**	**20.8**	**-100**	**45.5**	**33.7**	**-100**	**-51.5**	-1.9	-1.1	2.9	1.8	0.3	0.9	-1.9	-1.8	-1.2	-1.8	0.3	1.2
7	**-29.2**	**29.2**	**-24.8**	**-51.8**	**47.4**	**-24.8**	**-100**	**14.9**	8.6	**-23.4**	**-14.1**	-2.1	-7.3	**14.9**	**14.1**	9.3	**14.1**	-2.1	-9.3
8	-2.9	2.9	-2.5	-5.2	4.8	-2.5	**41.5**	**-100**	**50.8**	**49.2**	**29.6**	4.4	**15.2**	**-100**	**-29.6**	**-19.6**	**-29.6**	4.4	**19.6**
9	-2.1	2.1	-1.8	-3.8	3.5	-1.8	**30.1**	**64.3**	**-100**	**35.7**	**21.5**	3.2	**11.1**	**64.3**	**-21.5**	**-14.2**	**-21.5**	3.2	**14.2**
10	6.0	-6.0	5.1	**10.6**	-9.7	5.1	**-84.4**	**63.5**	**36.5**	**-100**	**-60.2**	-8.8	**-30.9**	**63.5**	**60.2**	**39.8**	**60.2**	-8.8	**-39.8**
11	3.6	-3.6	3.0	6.3	-5.8	3.0	**-50.2**	**37.8**	**21.7**	**-59.5**	**-100**	9.0	**31.5**	**37.8**	**100**	**-40.5**	**100**	9.0	**40.5**
12	0.4	-0.4	0.3	0.7	-0.6	0.3	-5.5	4.1	2.4	-6.5	6.7	**-100**	**86.8**	4.1	-6.7	**13.2**	-6.7	**-100**	**-13.2**
13	1.0	-1.0	0.9	1.8	-1.7	0.9	**-14.4**	**10.8**	6.2	**-17.1**	**17.6**	**65.4**	**-100**	**10.8**	**-17.6**	**34.6**	**-17.6**	**65.4**	**-34.6**
15	-2.9	2.9	-2.5	-5.2	4.8	-2.5	**41.5**	**-100**	**50.8**	**49.2**	**29.6**	4.4	**15.2**	**-100**	**-29.6**	**-19.6**	**-29.6**	4.4	**19.6**
16	-3.6	3.6	-3.0	-6.3	5.8	-3.0	**50.2**	**-37.8**	**-21.7**	**59.5**	**100**	-9.0	**-31.5**	**-37.8**	**-100**	**40.5**	**-100**	-9.0	**-40.5**
17	-2.9	2.9	-2.5	-5.2	4.8	-2.5	**41.5**	**-31.3**	**-18.0**	**49.2**	**-50.8**	**22.2**	**77.8**	**-31.3**	**50.8**	**-100**	**50.8**	**22.2**	**100**
18	-3.6	3.6	-3.0	-6.3	5.8	-3.0	**50.2**	**-37.8**	**-21.7**	**59.5**	**100**	-9.0	**-31.5**	**-37.8**	**-100**	**40.5**	**-100**	-9.0	**-40.5**
19	0.4	-0.4	0.3	0.7	-0.6	0.3	-5.5	4.1	2.4	-6.5	6.7	**-100**	**86.8**	4.1	-6.7	**13.2**	-6.7	**-100**	**-13.2**
20	2.9	-2.9	2.5	5.2	-4.8	2.5	**-41.5**	**31.3**	**18.0**	**-49.2**	**50.8**	**-22.2**	**-77.8**	**31.3**	**-50.8**	**100**	**-50.8**	**-22.2**	**-100**

column every time the LODF value exceeds the threshold, $d^* = 10\%$. Thus, for the first row of Table 6.1, there are entries made in the lists of line 2, line 3, line 4, line 5, line 6, and line 7. When this process is repeated for every line, the final product is the ITS.

The ITS for the IEEE 14-bus case with $d^* = 10\%$ is shown in Figure 6.3. The average length of a row is 8.5. The shortest row is 3 (for line 12 and line 19), and the longest row is for line 7. The fact that the row corresponding to line 7 is is so long indicates that most outages on the system impact line 7. In fact, after examining the LODF values, it can be seen that most outages affect line 7 strongly. The length of row 7 indicates that it is a central line in the system.

6.3 Incorporation of Line Flow and Limit Information

The screening algorithm discussed in Section 6.2 is purely based on the topology of the system, meaning that the only pieces of information needed to preform the screening algorithm are the line status and impedance values. The incorporation of line limit and flow information can improve the screening algorithm because it allows for differentiation between small, lightly loaded lines

and large, heavily loaded lines. This section adapts the screening algorithm to use line limit and flow information. Examples based on the IEEE 14-bus test case are presented to illustrate the operation and output of the tracking structure construction algorithms. Results of the screening algorithms from each algorithm are presented in Chapter 7 for the large system.

6.3.1 Incorporation of line flow information

The ITS construction algorithm (Algorithm 1) given in Section 6.2 works by calculating the LODF for the outage of every line and building a list for each line that records the lines that impact it above a given threshold. These impacts are measured with LODFs, which by definition are the changes in flow on a line as a percentage of the preoutage flow of the outaged line (2.16). Thus, multiplying the LODF by the preoutage flow on the outaged line gives the change in flow on the line of interest (2.31).

When these values are recorded in the ITS, they are recorded as LODFs. If line flow information is available, then replacing the ITS entries with flow changes becomes possible. Thus, Algorithm 1 is altered to become the flow-tracking structure (FTS) construction algorithm, given in Algorithm 3. It should be noted that the flow changes are calculated before entry into the FTS. This means that it is not possible to simply multiply the values in the ITS by the line flow values to generate the FTS. Doing so would require storing the full $L \times L$ LODF matrix.

The FTS construction algorithm builds a list of lines whose outage will affect the flow on the line of interest above a certain MW amount. This is theorized to provide a better gauge of the impact of the actual effect the outage of a line has, because it measures an actual change in flow. For example, a lightly loaded line

Input: List of lines
Input: Flow change threshold f^*
Input: Line Flows
Output: FTS
foreach *Line* α **do**
  foreach *Line* β **do**
    if $\alpha \neq \beta$ **then**
      Calculate $d_{\alpha,\beta}$; Calculate Δf_α; **if** $\Delta f_\alpha \geq f^*$ **then**
        Add entry at Row α for line β;
      end
    end
  end
end

Algorithm 3: FTS construction algorithm

may show up prominently in the ITS because it has high LODF values onto other lines. In the FTS, the outage of a lightly loaded line will not show up because the LODF-flow product will be low. However, it should be noted that the values in the FTS are single-outage flow changes: the flow change is calculated using the single-outage (scalar) LODF and the base case flow values. As with the ITS, there is no knowledge of the effect of double outages in the FTS.

The algorithm for generating a list of contingencies from the FTS is the same as the one used by the ITS screening algorithms. In other words, the list of contingencies is generated by generating the combinations of the elements in the row lists and removing the nonunique elements, as given in Algorithm 2. The use of this algorithm is based on the same intuition that lies behind the ITS.

The FTS for the IEEE 14-bus test case [24] is shown in Figure 6.4. This FTS was generated using a 1 MW cutoff, meaning that a change of less than 1 MW (predicted using the LODF and the base case flow) was not included in the structure. Compared to the ITS for the IEEE 14-bus test case, the FTS for a 1 MW change is noticeably broader. The average row length is 10 for the FTS in

Figure 6.4. However, the maximum row length is the same in both cases. For both the ITS and the FTS, the row corresponding to line 6 has a length of 16. Thus, the fact that line 6 is impacted by many outages is detected by both structures.

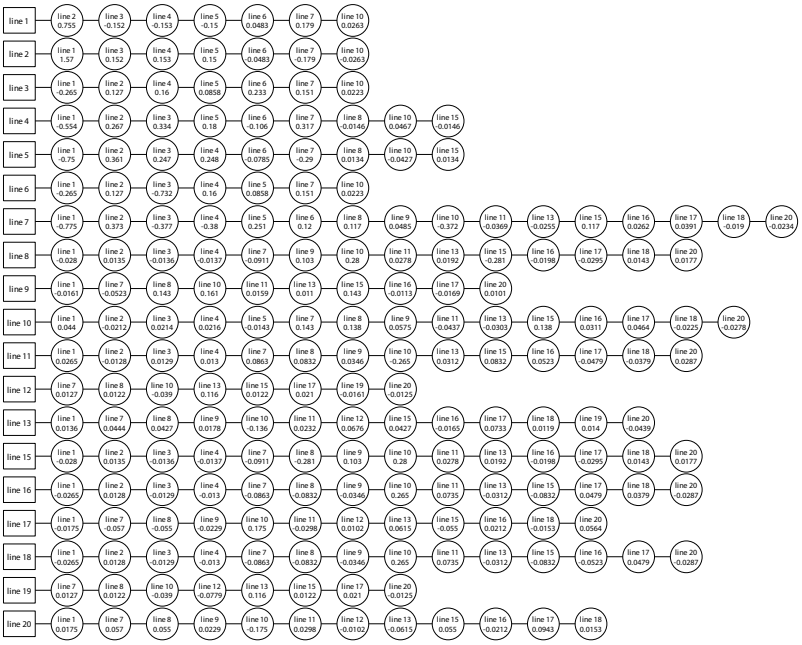

Figure 6.4: FTS for the IEEE 14-bus test case (1 MW cutoff)

6.3.2 Incorporation of line limit information

The next evolution in refining the screening algorithm is to add line limit information. This goes one step beyond the flow-tracking structure, which considers only flow information. The addition of line limit information allows the relative impact of a flow change to be gauged. When the limit information is added, the impact can be considered in terms of a percentage change in line limit. The change in flow as a percentage of the affected line's limit can be

written as

$$g_{\alpha,\beta} = \frac{\Delta f_{\alpha,\beta}}{L^{\alpha}} \tag{6.4}$$

which can be evaluated using LODFs as

$$g_{\alpha,\beta} = \frac{d_{\alpha,\beta} f_{\beta}}{L^{\alpha}} \tag{6.5}$$

where line β is the outaged line, and line α is the impacted line.

This changes the screening algorithm to the one shown in Algorithm 4. This formulation adds an entry into the tracking structure when the change (as a fraction of the line flow limit) is greater than the specified value l^*. The end result of Algorithm 4 is the limit-tracking structure (LTS). The LTS for the IEEE 14-bus test case is shown in Figure 6.5.

Input: List of lines
Input: Change threshold l^*
Input: Line Flows
Output: LTS
foreach *Line* α **do**
  foreach *Line* β **do**
    if $\alpha \neq \beta$ **then**
      Calculate $d_{\alpha,\beta}$; Calculate Δg_{α}; **if** $\Delta g_{\alpha} \geq l^*$ **then**
        Add entry at Row α for line β;
      end
    end
  end
end

Algorithm 4: LTS construction algorithm

The algorithm for generating the list of contingencies from the LTS is given in Algorithm 2. This is the same algorithm that is used for the ITS and the FTS, and the intuition remains the same. Algorithm 2 works by generating a list of double-outage contingencies in which each element of the double-outage pair impacts a third line. By generating the combinations of the row entries, the list

of contingencies is made up of lines that both have an impact on a third.

The LTS for the IEEE 14-bus test case was calculated using flow limits of 100 MW on every line. This limit was chosen arbitrarily because the test case has no limits. Realistic power system cases include limit information. However, for illustrative purposes, picking a limit will suffice.

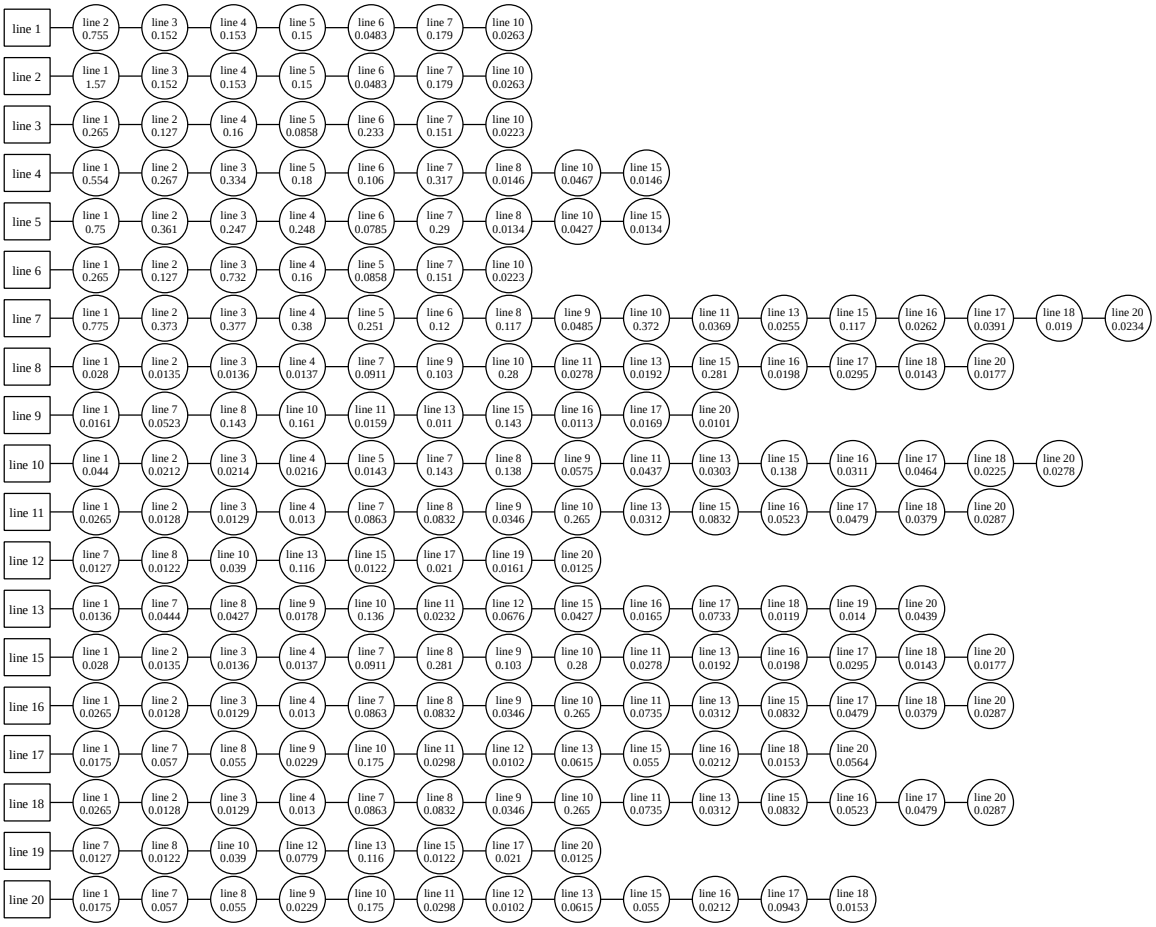

Figure 6.5: LTS for the IEEE 14-bus test case (0.01)

The LTS was created for the IEEE 14-bus case with a change in flow as a percentage of line limit of 1%. In other words, if the outage of a line resulted in a change in flow that was at least 1% of the line's limit, the outage was added to the LTS. The LTS for the IEEE 14-bus test case is shown in Figure 6.5. The rows of the LTS in Figure 6.5 are made up of circles that contain a line identifier

and the change in flow as a fraction between 0 and 1 of the impacted line's limit, where 0 indicates no impact and 1 indicates a change equal to the line's limit.

6.3.3 Combination of limit and flow information

From examining the LTS, it is fairly simple to see how the structure can be extended to include line flow information, and indeed the incorporation of flow information along with limit and sensitivity data is the final enhancement to Algorithm 1. Adding line flow information allows for finer discrimination because it allows for the detection of the instances when an outage actually reduces the flow on a line. Using current line flow information also allows us to know how close a line is to overload in the base case, which in most cases also indicates how likely the line is to overload in the event of an outage.

The structure that is built is dubbed the overload-tracking structure (OTS). The OTS construction algorithm is given in Algorithm 5. The algorithm accepts a list of lines in the system, the current line flows, and an overload threshold value, o^*, as inputs. The overload threshold is a margin away from a single-outage postcontingent overload. In other words, specifying an overload threshold value of 5% means that outages that result in a flow of 95% of rated limit will be included in the OTS. An o^* value of 10% indicates that outages that result in a flow of 90% of rated limit will be included in the OTS. The final output of Algorithm 5 is the OTS.

The OTS for the IEEE 14-bus test case is given in Figure 6.6. The OTS was generated using an overload threshold value (o^*) of 10%. The rows of the OTS contain a line identifier and the postcontingent flow as a percentage of the line's limit. Again, since there are no line limits in the IEEE 14-bus test case, a limit of 100 MW was used for every line. Each circle contains a line identifier and the

Input: List of lines
Input: Flow change threshold o^*
Input: Line Flows
Output: OTS
foreach *Line* α **do**
  foreach *Line* β **do**
    if $\alpha \neq \beta$ **then**
      Calculate $d_{\alpha,\beta}$; Calculate Δf_α; **if** $\Delta f_\alpha \geq o^*$ **then**
        Add entry at Row α for line β;
      end
    end
  end
end

Algorithm 5: OTS construction algorithm

predicted change in flow for the outage of the line identified in the circle. In other words, when the line identified in the circle is outaged, the postcontingent flow — calcluated using linear sensitivites — on the line that the row belongs to is the other value in the circle. The overload threshold value, o^*, means that values less than 100% can be included in the OTS, which explains the enries in the 90s on the second row.

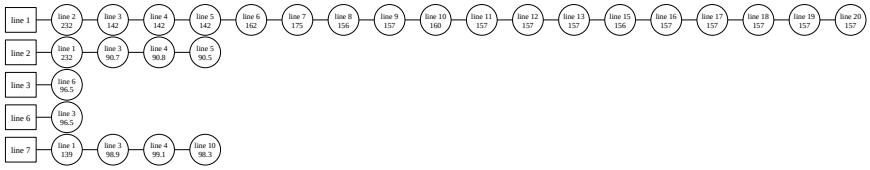

Figure 6.6: OTS for the IEEE 14-bus test case

The most striking feature of the OTS is its size. It is much smaller than any of the other tracking structures. The length of the rows is also different from those in the the other tracking structures. The longest row for the OTS is the first row, corresponding to line 1. The longest row for the other tracking structures is the row corresponding to line 7. The average row length is only 1.4 (note that the zero-length rows are not shown). However, there is still a lot of information in the OTS. In fact, if the o^* value is 0, the OTS shows the

single-outage contingency analysis results. For larger values of o^*, the OTS contains the lines that are close to a postcontingent violation. The length of the first row and the fact that none of the postcontingent flow values is less than 100 mean that every contingency (except for the radial line 14) results in an overload on line 1. This is because the base case flow on line 1 is 156.9 MW. It is violating in the base case.

The other four OTS rows are a little more interesting. They do not violate the 100 MW limit in the base case. The base case flows for the lines with rows in the OTS are given in Table 6.2. As mentioned before, the base case flow on line 1 violates the 100 MW limit. However, the base case flows on line 2, line 3, line 6, and line 7 are all well below the limit. That makes these lines interesting. The fact that they have rows in the OTS means that they have outages that bring them to within the overload threshold value, o^*, of violating their limits. A careful look at the postcontingent flows in the OTS shows that line 2 and line 7 have contingencies that result in violations. For line 2, the outage of line 1 results in an overload of 232%. For line 7, the outage of line 1 results in an overload of 139%.

Table 6.2: Base case flows on lines with OTS rows

Line	From bus	To bus	Circuit	Base case flow (MW)
line 1	1	2	1	159.9
line 2	1	5	1	75.5
line 3	2	3	1	73.2
line 6	4	4	1	-23.3
line 7	5	5	1	-61.2

The overload-tracking structure uses a different algorithm for generating the list of contingencies from the tracking structure. The motivation behind using a different tracking structure is the desire to capture more contingencies. The

algorithm is given in Algorithm 6. This algorithm generates a list of contingencies by combining the entries in the tracking structure with the other lines in the system. This is designed to capture the single-outage contingencies that result in violations when they are paired with other lines in the system.

Input: Overload Tracking Structure OTS
Output: List of Flagged Contingencies FL
foreach *Row r in the ITS* **do**
    foreach *Element e in row r* **do**
        Generate combination of e with every other line in the system
    end
end
Remove nonunique elements from FL.

Algorithm 6: List generation algorithm for the overload-tracking structure

The insight behind combining the entries in the OTS with every other line in the system can be seen by realizing that the OTS essentially contains the single-outage contingency analysis results. For a threshold (o^*) of 0, the OTS contains the lines whose outage results in a violation in the single-outage contingency analysis. Increasing the value of o^* pads the results, but the intuition is the same. Combining the lines that resulted in a violation for the single-outage contingency analysis with every other line in the system is assuming that the single outage violation will persist in the double-outage violations.

CHAPTER 7

LARGE-SYSTEM SCREENING RESULTS

The screening algorithms in Section 6.2 were each applied to a 5395-bus, 7616-line system based on a large North American utility. Considering every double-line outage in the system results in a contingency list with 28 997 920 events. The goal of these algorithms is to generate lists of contingencies that capture the severe contingencies without including a large number of innocuous events.

The results from the screening algorithms are compared against several lists of severe contingencies. The most exhaustive list includes every contingency that results in a violation. This list contains 546 557 contingencies that result in violations. Thus, the perfect contingency-screening algorithm would generate the 546 557 contingencies in this list with no extras. Since this list includes many contingencies that result in violations only on relatively small lines, another list of contingencies was generated by removing the contingencies that result in violations only on lines with limits less than 100 MW. This list is approximately one-fifth the size of the unfiltered contingency list. It contains 112 049 elements. Another list of interest is the list of double-outage contingencies that result in violations only when both of the elements are outaged. This list is referred to as the double-violation list, and it contains 1851 contingencies.

The results of the screening algorithm, as well as information about the tracking structures, are presented for the different screening algorithms. The results show that the ITS algorithms are the best for detecting the contingencies

in the double-violation list and that the OTS algorithms are very good at detecting severe contingencies in general.

7.1 ITS Results

This section discusses the contingency-screening results generated by the impact-tracking structure algorithms described in Section 6.2. The results of the screening algorithm as well as information about the ITS are presented for five different values of d^*. The minimum value was selected to be 5%, based on the NERC threshold for transaction curtailment [50]. The threshold value is varied from the minimum 5% to 25% in increments of 5%. This is done to examine the behavior of the screening algorithm as its key parameter is varied.

The experimental results show that the screening algorithms do, in fact, screen out a very high percentage of the double-outage events. Also, it is clear that the larger the value of d^*, the fewer contingencies are flagged. This is to be expected because the ITS for a larger threshold value is a subset of an ITS with a smaller d^*. In other words, increasing d^* can only remove entries from the ITS. Unfortunately, the algorithm fails to capture a large number of severe contingencies.

Table 7.1 contains the screening results for the ITS screening algorithms, comparing them against the full list of severe contingencies. The table contains the number of severe contingencies captured, the number of extra (nonsevere) contingencies generated by the algorithm, the number of missed severe contingencies, and the percentage of contingencies captured. The percentage is calculated by dividing the number captured by the total number of severe contingencies. For example, the 6.6% captured for the d^* value of 0.05 is

calculated as

$$\%captured = 100 \cdot \frac{36\ 153}{546\ 557} = 6.6\% \tag{7.1}$$

Examining Table 7.1, it is clear that the ITS based algorithms fail to capture a large number of severe contingencies. For the most conservative value of d^*, only 6.6% of the severe contingencies are captured.

Table 7.1: ITS screening results

Full violation list				
d^*	Captured	Extra	Missed	Captured (%)
0.05	36 153	1 374 307	511 966	6.6
0.10	13 877	514 485	534 242	2.5
0.15	7 038	274 220	540 181	1.3
0.20	4 790	168 124	543 329	0.87
0.25	2 909	113 858	545 210	0.053

Table 7.2 contains the results of comparing the ITS algorithms' results against the filtered list of severe contingencies. These results are actually worse than the results in Table 7.1 in the sense that a lower percentage of the severe contingencies is captured.

Table 7.2: ITS screening results

Filtered violation list				
d^*	Captured	Extra	Missed	Captured (%)
0.05	5 777	1 404 683	106 272	5.2
0.10	2 539	525 823	109 810	2.3
0.15	1 388	279 870	110 661	1.2
0.20	837	172 077	111 212	0.75
0.25	583	116 184	111 466	0.52

The results of the ITS screening algorithms is compared to the double-violation list in Table 7.3. These results show more promise. The algorithms capture between 30.6% and 66.5% of the contingencies in the double-violation list. While these values may not sound particularly high, capturing the contingencies in the double-violation list is difficult, and the ITS

algorithms have the best performance in this area. The reason that the ITS performs well at generating contingencies in the double-violation list is the way that the list of contingencies is generated from the ITS. The list is generated by taking the combinations of the elements in the ITS rows. This generates a list of double-outage contingencies in which each line of a contingency impacts a line above the d^* threshold.

Table 7.3: ITS screening results

Double violation list				
d^*	Captured	Extra	Missed	Captured (%)
0.05	1 230	1 409 230	621	66.5
0.10	1 016	527 346	835	54.9
0.15	785	280 473	1 066	42.4
0.20	653	172 261	1 198	35.3
0.25	567	116 200	1 284	30.6

7.2 FTS Results

The FTS algorithms work by building a tracking structure, where the threshold for entry is based on the change in flow resulting from an outage. The LTS was generated using four different threshold values, from 5 MW to 30 MW.

The results of the FTS algorithms are compared against three different sets of contingency analysis results. The results are compared against the full contingency analysis set in Table 7.4. Table 7.5 contain the FTS screening results compared to a filtered set of contingency analysis results, and Table 7.6 compares the screening results against the double outages that result in violations only when both lines are outaged.

The performance of the FTS algorithms is pretty poor for the full results set and the filtered results set. The percentage of captured contingencies is rather small. For the full contingency analysis results set, only 4.55% of the

contingencies that result in violations are caught by the algorithm. Even fewer (4.25%) of the contingenceis that result in violations are detected.

Table 7.4: FTS screening results

Full violation list				
f^*	captured	extra	missed	captured (%)
0.05	24 963	488 878	52 156	4.55
0.10	10 601	187 724	537 518	1.93
0.20	3 622	61 408	544 497	0.66
0.30	1 983	29 771	546 136	0.36

Table 7.5: FTS screening results

Filtered violation set				
f^*	captured	extra	missed	captured (%)
0.05	4 758	509 083	107 291	4.25
0.10	2 076	196 249	109 973	1.85
0.20	892	64 138	111 157	0.80
0.30	553	31 201	111 496	0.49

The FTS screening algorithm results are much better when the results are compared to the double-violation list. Around half of the double-outage contingencies that result in violations only when both lines are outaged are caught by this algorithm.

Table 7.6: FTS screening results

Double violation list				
l^*	Captured	Extra	Missed	Captured (%)
0.05	930	512 911	921	50.24
0.10	615	197 710	1 236	33.23
0.20	333	64 697	1 518	17.99
0.30	252	31 502	1 599	13.6

7.3 LTS Results

The LTS algorithms work by constructing a tracking structure based on the amount of change as a percentage of the lines limit. The LTS was generated

using four different threshold values from 0.05 to 0.20 in steps of 0.05. A 0.05 threshold value indicates that the outage of a line results in a change of 5% with respect to the impacted line's limit. In other words, when an outage occurs, the impacted line sees a change in flow of at least 5% of its limit.

The results of the LTS screening algorithms can be found in Table 7.7, Table 7.8, and Table 7.9. In general, these algorithms perform in a similar manner to the ITS algorithms. The algorithms have very poor performance when compared to the lists of severe contingencies.

Table 7.7: LTS screening results

Full violation list				
l^*	Captured	Extra	Missed	Captured (%)
0.05	10 115	69 668	538 004	1.85
0.10	3 883	17 822	544 236	0.71
0.15	1 832	6 765	546 287	0.33
0.20	1 033	3 123	547 086	0.19

Examining Table 7.8, it can be seen that the results of the LTS algorithms perform even worse when compared to the filtered list of contingencies.

Table 7.8: LTS screening results

Filtered violation set				
l^*	Captured	Extra	Missed	Captured (%)
0.05	1 726	78 057	110 323	1.5
0.10	642	21 063	111 407	0.57
0.15	328	8 269	111 721	0.29
0.20	209	3 947	111 840	0.19

Table 7.9: LTS screening results

Double violation list				
l^*	Captured	Extra	Missed	Captured (%)
0.05	1 003	78 780	848	54.2
0.10	515	21 190	1 336	27.8
0.15	341	8 256	1 510	18.2
0.20	208	3 948	1 643	11.2

7.4 OTS Results

The standout among the screening algorithms is the OTS algorithms. The OTS algorithms are most different from the previous algorithms in the way that the list of contingencies is generated. Instead of generating a list from the combinations of the tracking structure rows, the OTS generates a list of contingencies by combining the OTS entries with the other lines in the system. The insight behind this can be seen by considering a percent overload, o^*, value of 0. For a percent overload value of 0, entries are made in the OTS whenever a single-outage contingency results in a violation. The OTS contingency generation algorithm works because the single-outage violations also result in violations when they are outaged along with (most) other lines in the system.

The results in Table 7.10 show that the OTS algorithms do a very good job of predicting the double-outage contingencies that will result in violations. It does this by using a padded version of the single-outage contingency analysis results and predicting that a single outage that creates a violation will also create a double-outage violation. Using this method, 99.72% of the double-outage contingencies that result in violations from the single-outage contingency analysis alone are captured, and with very few extra contingencies. Solving 598 504 contingencies will result in capturing over 99% of the double-outages with violations. This is only 2.06% of the total number of double-outage contingencies. As the padding is increased, the number of captured contingencies increases. However, beyond a certain point the number of extra contingencies explodes. As can be seen by examining the last two rows in Table 7.10, changing the value of o^* from 0.050 to 0.075 resulted in the number of extra contingencies increasing by a factor of ten. There is only one missed contingency; however capturing this many contingencies comes at the expense of creating a

contingency list that contains nearly all of the double-outage contingencies.

Table 7.10: OTS screening results

Full violation list				
o^*	Captured	Extra	Missed	Captured (%)
0.000	546 557	51 947	1 562	99.72
0.025	547 025	126 794	1 094	99.8
0.050	548 118	276 683	653	99.88
0.075	548 770	27 739 966	1	99.9998

Table 7.11 compares the contingency screening results against the filtered violation list generated by the full contingency analysis. The results are very good. The percentage captured is slightly lower than the full set of violations. However, the number of missed contingencies is lower. This result indicates that the OTS screening algorithms are capable of detecting the double-outage contingencies that result in violations on lines with limits over 100 MW. This means that the results are not being skewed by a large number of violations on small lines.

Table 7.11: OTS screening results

Filtered violation list				
o^*	Captured	Extra	Missed	Captured (%)
0.000	111 459	487 045	590	99.473
0.025	111 800	562 019	249	99.778
0.050	111 867	712 282	182	99.838
0.075	112 048	28 176 036	1	99.9991

Detecting double contingencies that only result in violations when both lines are out is an area of relative weakness for the OTS screening algorithms. As Table 7.12 shows, the performance of the algorithms is not as good as it is when the results are compared against the full violation set and the filtered violation list. However, the performance is not particularly bad when compared with the results of the other screening algorithms.

Table 7.12: OTS screening results

Double violation list				
o^*	Captured	Extra	Missed	Captured (%)
0.000	0	598 504	1 851	0
0.025	468	673 351	1 383	25.28
0.050	909	823 240	942	49.11
0.075	1 555	28 286 529	296	84.0

7.5 Ranking Results

This section compares the results of the contingency-screening algorithms with the lists of serious contingencies developed in Chapter 5. The goal is to gauge the performance of the screening algorithms with respect to detecting severe contingencies.

7.5.1 PI ranking results

The contingency screening results for the screening algorithms are compared to the list of severe contingencies identified by the PI contingency-ranking method. Two lists were generated using the PI method. One list is based on applying the PI contingency-ranking methods to the full contingency analysis results. This list, referred to as the PI full list, has 62 785 entries. The second list was generated by using the PI ranking methods on the filtered set of violations — violations on lines with limits less than 100 MW were removed.

The results of the ITS screening algorithms, compared to the lists of severe contingencies generated by the PI ranking methods, are given in Table 7.13 and Table 7.14.

Table 7.15 contains the results of FTS screening algorithms compared to the severe contingency set generated from the full set of contingency analysis results ranked using the PI methods. These results show that the FTS screening algorithms capture only a few percent of the severe contingencies. At best, the

Table 7.13: ITS screening results

PI full list				
d^*	Captured	Extra	Missed	Captured (%)
0.05	4 358	58 427	58 427	6.941
0.10	1 926	60 859	60 859	3.068
0.15	1 088	61 697	61 697	1.732
0.20	810	61 975	61 975	1.29
0.25	485	62 300	62 300	0.773

Table 7.14: ITS screening results

PI filtered list				
d^*	Captured	Extra	Missed	Captured (%)
0.05	3 872	1 406 588	45 632	7.8215
0.10	1 794	526 568	44 710	3.624
0.15	1 021	280 237	48 483	2.0625
0.20	759	172 155	48 745	1.5332
0.25	449	116 318	49 055	0.907

algorithm managed to capture 5.605% of the severe contingencies.

Table 7.15: FTS screening results

PI full list				
f^*	captured	extra	missed	captured (%)
0.05	3 519	510 322	59 266	5.605
0.10	1 535	196 790	61 250	2.45
0.20	524	64 506	62 261	0.8346
0.30	314	31 440	62 471	0.500

The results in Table 7.16 show the performance of the FTS screening algorithms when compared against the filtered list set of contingency results. By comparing the values in Table 7.16 to the results in Table 7.15, it can be seen that the FTS screening algorithms perform slightly better on the filtered result set. This is a slightly more encouraging result. However, the overall capture rate is still dismal. In the best case, the FTS screening methods managed to capture 6.788% of the severe contingencies.

The results for the LTS screening algorithms are given in Table 7.17 and Table 7.18. These results are even worse than the FTS screening results. The

Table 7.16: FTS screening results

PI filtered list				
f^*	captured	extra	missed	captured (%)
0.05	3 356	510 458	46 148	6.788
0.10	1 498	196 827	48 006	3.026
0.20	515	64 515	48 989	1.040
0.30	305	34 119	49 199	0.616

LTS screening results show the same behavior as the FTS results. The algorithms fail to capture many severe contingencies, but the method performs slightly better for the list of filtered contingencies.

Table 7.17: LTS screening results

PI full list				
l^*	Captured	Extra	Missed	Captured (%)
0.05	1 702	78 081	61 083	2.71
0.10	820	20 885	61 965	1.31
0.15	435	8 162	62 350	0.693
0.20	290	3 866	62 495	0.462

Table 7.18: LTS screening results

PI filtered list				
l^*	Captured	Extra	Missed	Captured (%)
0.05	1 605	78 178	47 899	3.242
0.10	790	20 915	48 714	1.596
0.15	414	8 183	49 090	0.836
0.20	275	3 381	49 229	0.556

The OTS screening results show a large departure from the other screening algorithms. The OTS screening algorithms work very well. The results of the OTS screening algorithms are compared in Table 7.19 to the unfiltered list of severe contingencies identified by the PI-ranking methods. The OTS screening results are compared to the filtered list of contingencies in Table 7.20. The results given in both tables are very strong. However, the performance of the OTS screening algorithms is slightly better for the filtered list of contingencies.

Both tables show capture results over 99%. Increasing the threshold o^* does increase the capture rate slightly at the expense of increasing the number of extra contingencies. However, the performance is always good. In the worst case, only 0.193%, a total number of 121, of the contingencies are missed.

Table 7.19: OTS screening results

PI full list				
o^*	Captured	Extra	Missed	Captured (%)
0.000	62 664	535 840	121	99.807
0.025	62 688	611 151	117	99.814
0.050	62 669	761 480	116	99.815
0.075	62 727	28 225 357	58	99.908

Table 7.20: OTS screening results

PI filtered list				
o^*	Captured	Extra	Missed	Captured (%)
0.000	49 395	549 109	109	99.7798
0.025	49 399	624 420	105	99.7879
0.050	49 400	774 749	104	99.7899
0.075	45 454	28 238 630	50	99.899

7.5.2 Sorted-matrix ranking results

The sorted-matrix ranking techniques discussed in Chapter 5 are applied to the full contingency analysis results for the large case in order to generate lists of severe contingencies. The sorted-matrix techniques are applied to two sets of contingency analysis results. One set of data contains every violation, while the other set is filtered to remove the violations that occur on lines with a limit of less than 100 MW.

The lists of severe contingencies were made by examining the distributions of the contingency aggregate values. Distribution plots for these values are given in Chapter 5. The goal of applying the sorted-matrix methods is to develop a list

of the most severe contingencies. In order to make a list of the most serious contingencies, the contingencies that fall in the upper 50% of the distribution are selected. The unfiltered list has a total number of 290 875 double-outage contingencies when the lower 50 percentiles are removed, and the filtered list has a total number of 56 212 double-outage contingencies. The screening results are compared to the two lists for each of the screening algorithms.

The results of the ITS screening algorithms are compared to the sorted-matrix ranking results in Table 7.21 and Table 7.22. The results show that the ITS screening algorithms do a poor job of detecting the contingencies flagged by the sorted-matrix ranking method. For the unfiltered list the best capture rate is only 6.54%, and the number of extra contingencies is quite large. The list of contingencies generated by the ITS screening algorithm is much larger than the list of contingencies flagged by the sorted-matrix ranking method and only a few percent of the severe contingencies are detected. The screening results are compared aginst the filtered sorted matrix list in Table 7.22. These results are slightly better than the results in Table 7.21, but still only a few percent of the serious contingencies are detected.

Table 7.21: ITS screening results

Sorted matrix list				
d^*	Captured	Extra	Missed	Captured (%)
0.05	19 015	1 391 445	271 860	6.54
0.10	7 552	520 810	283 323	2.60
0.15	3 994	277 264	286 881	1.37
0.20	2 799	170 115	288 076	0.962
0.25	1 611	115 156	289 264	0.554

Table 7.23 compares the results of the FTS screening algorithms to the sorted-matrix list of severe contingencies. The FTS results are compared against the sorted-matrix filtered list of severe contingencies in Table 7.24. The capture rates are low for both sets of contingencies. According to these results, the FTS

Table 7.22: ITS screening results

Sorted matrix filtered list				
d^*	Captured	Extra	Missed	Captured (%)
0.05	4 247	1 406 213	51 965	7.555
0.10	1 907	526 455	54 305	3.393
0.15	1 093	280 165	55 119	1.944
0.20	631	172 283	55 581	1.123
0.25	458	116 309	55 754	0.8148

is not an effective way of detecting severe contingencies. These results agree with the results for the FTS algorithms for the other lists of severe contingencies.

Table 7.23: FTS screening results

Sorted matrix list				
f^*	Captured	Extra	Missed	Captured (%)
0.05	13 967	499 874	276 908	4.80
0.10	5 891	192 434	284 984	2.03
0.20	1 899	63 131	288 976	0.653
0.30	1 011	30 743	289 864	0.348

Table 7.24: FTS screening results

Sorted matrix filtered list				
f^*	Captured	Extra	Missed	Captured (%)
0.05	3 669	510 172	52 543	6.53
0.10	1 641	196 684	54 571	2.93
0.20	703	64 327	55 509	1.25
0.30	448	31 309	55 767	0.792

The list of contingencies generated by the LTS screening algorithms are compared to the list of serious contingencies generated by the sorted matrix methods in Table 7.25 and Table 7.26. These results are similar to the results of the LTS algorithms when they are compared to the other lists of severe contingencies. The LTS does a poor job of detecting severe contingencies. The capture rates are very low. In fact, the LTS has the lowest capture rates of any of the screening algorithms.

Table 7.25: LTS screening results

Sorted matrix list				
l^*	Captured	Extra	Missed	Captured (%)
0.05	5 906	73 877	284 969	2.30
0.10	2 463	19 242	288 412	0.847
0.15	1 266	7 331	289 609	0.435
0.20	768	3 388	290 107	0.264

Table 7.26: LTS screening results

Sorted matrix filtered list				
l^*	Captured	Extra	Missed	Captured (%)
0.05	1 377	78 406	54 835	2.45
0.10	485	21 220	55 727	0.863
0.15	238	8 359	55 974	0.423
0.20	156	4 000	56 056	0.278

The OTS screening algorithms are very good at detecting the severe contingencies detected by the sorted-matrix ranking method. The results for the unfiltered sorted-matrix list are given in Table 7.27, and the screening algorithm results is compared to the results of the sorted-matrix ranking applied to a filtered data set in Table 7.28. The OTS is again the standout algorithm. It has very high capture rates, meaning that it is very good at detecting the contingencies that the sorted-matrix ranking method flagged as severe. The other algorithms tend to perform poorly.

Table 7.27: OTS screening results

Sorted matrix list				
o^*	Captured	Extra	Missed	Captured (%)
0.000	290 666	307 838	209	99.928
0.025	290 678	383 141	197	99.932
0.050	290 692	533 457	183	99.937
0.075	290 874	27 997 210	1	99.9997

Table 7.28: OTS screening results

Sorted matrix filtered list				
o^*	Captured	Extra	Missed	Captured (%)
0.000	56 065	542 439	147	99.738
0.025	56 083	617 736	129	99.771
0.050	56 106	768 043	106	99.811
0.075	56 211	27 739 966	1	99.9998

7.6 Tracking Structure Statistics

Along with the results from the various screening algorithms, information about the tracking structures was collected. The information includes the size (i.e., the number of nonzero entries), the number of rows with at least one entry, the number of rows with no entries, the maximum length of any row, and the average row length. The average row length is calculated by dividing the number of entries by the total number of rows. The total number of rows is equal to the number of lines in the system, since there is a row for each line in the system. Also, the number of nonzero rows and the number of zero-length rows should add to equal the number of lines in the system.

The size data for the ITS is given in Table 7.29. The size information shows that the larger the value of d^*, the fewer entries are made in the ITS. This makes sense because raising the value of d^* means that a line must have a greater LODF value to be entered into the tracking structure. The size decreases very rapidly as the threshold, d^*, increases. The average row and maximum row length also decrease rapidly.

The size data for the FTS and the LTS are shown in Table 7.30 and Table 7.31, respectively. The sizes of these two data structures is small compared to the size values for the ITS. These two structures are considered together because they are so similar. In both cases, the size of the structure decreases with an increase in the threshold value. Also, the entries in the

Table 7.29: ITS size data

d^*	ITS size	Avg. row length	Max. row length	Nonzero rows
5%	284 507	37	339	6 326
10%	150 853	19	180	6 191
15%	100 105	13	96	6 099
20%	72 260	9	67	6 018
25%	55 560	7	49	5 929

tracking structure are fairly well distributed. This is indicated by the large number of nonzero rows.

Table 7.30: FTS size data

f^*	FTS size	Avg. row length	Max. row length	Nonzero rows
0.05	148 266	19	244	6 097
0.10	74 781	9	150	5 495
0.20	33 616	4	89	4 291
0.30	20 283	2	54	3 585

Table 7.31: LTS size data

f^*	LTS size	Avg. row length	Max. row length	Nonzero rows
0.05	41 995	5	122	4 571
0.10	16 575	2	55	4 058
0.15	8 631	1	34	3 179
0.20	4 903	0	25	2 248

Information for the OTS is shown in Table 7.32. The size information shows that it has some unique properties compared to the other tracking structures. First, the size of the OTS increases as the amount of padding, o^*, increases. This is because the more padding is added, the more entries are made into the tracking structure. Also, the size of the OTS is very small. There are only 13 056 entries in the largest case. This is much smaller than the other tracking

structures. Examining the number of nonzero rows, it can be seen that the entries in the OTS appear at only a few rows.

Table 7.32: OTS size data

o^*	OTS size	Avg. row length	Max. row length	Nonzero rows
0.000	113	0.015	35	48
0.025	130	0.017	38	56
0.050	159	0.021	49	61
0.075	13 056	1.710	6424	76

The size information about the various screening algorithms gives insight into the inner workings of the various algorithms. For example, the run time of the ITS, FTS, and LTS list generation algorithms is dependent upon the lengths of the rows in the various tracking structures, because the list of contingencies is generated by taking the combinations of the entries in a row. The run time of the list generation portion of the OTS screening algorithm is dependent on the OTS size because the contingency list is generated by taking the entries in the OTS in combination with every line in the system.

7.7 Conclusions

This chapter compares the results of several screening algorithms against the results of the full contingency analysis for the large system. The incremental algorithms — the ones that slightly extend the amount of information available — show rather poor performance. At best they are able to capture only a few percent of the double-outage contingencies that result in violations. However, there are two standout algorithms that perform quite well. The ITS screening algorithms do the best job of detecting double-outage contingencies that result in violations only when both lines are outaged. This seems to be because the

ITS screening algorithms do a good job of detecting contingencies that both impact a third line. The OTS does a very good job in general. The list generated by the OTS screening algorithms is only 2% of the size of the complete list of double-outage contingencies, yet it is able to detect over 99% of the double-outage contingencies that result in violations.

CHAPTER 8

WEAK-ELEMENT IDENTIFICATION

The screening algorithms are based on generating tracking structures that track the impact of a line outage on the other lines in the system using several different metrics. Knowing how lines impact each other in a power system can be useful for more than generating contingency lists. For example, knowing that the outage of every line in the system heavily impacts some line gives some information about that line. Thus, the information in the tracking structures can be used for other applications. One natural application is weak-element identification.

8.1 Introduction

In a power system a weak element is a line or transformer that is likely to become overloaded. Weak elements are identified using contingency analysis. The more often the element has a violation in the contingency analysis, the weaker it is. The fact that a line frequently violates in the contingency analysis suggests that the system would benefit from its upgrade. Identifying weak elements is important because these elements will result in constraints in operation. If there are multiple elements whose outage results in the overload of a particular line, then in order to operate the system safely, those elements must be operated well below their limits in order to make sure that their outage does not result in an overload. In other words, the presence of weak elements forces

the underutilization of other parts of the transmission system. This chapter examines the use of the tracking structures for identifying weak elements in the system. The tracking structures were introduced in Chapter 6. A tracking structure is a list of lists, where every line in the system is assigned a list that contains the lines whose outage impact it. Each list entry has two pieces of data: (1) the line whose outage results in an impact, and (2) the amount of the impact, I. This chapter develops metrics of a lines weakness based on the tracking structure for the impact-tracking structure and the overload-tracking structure.

8.2 Weak-Element Characterization

The idea behind building a tracking structure is to keep track of the lines that impact each other. This type of information is also useful for weak-element identification. Several factors characterize the weakness of a particular element. For example, does a single outage result in a 400% overload, or do 100 outages result in a 101% overload? These types of characterizations can be made by examining different aspects of the tracking structures. The list below contains the parameters that characterize the weakness of an element based on the tracking structure.

- Frequency of appearance in the tracking structure, F
- Length of row in the tracking structure, L
- Maximum row entry, M
- Average row entry, A
- Average value per entry, E

The tracking structure contains enough information to say which elements are likely to cause problems and which elements are likely to have problems. The elements that are likely to cause problems will appear in the lists for many lines. Thus, the frequency at which a line appears in the tracking structure indicates how likely the outage of that line is to cause violations. In a similar manner, the magnitudes of the entries in the tracking structures give an indication of how much an outage is likely to impact other lines. The length of a row in the tracking structure gives an indication of how many lines in the system impact a given line. The more entries in a line's row, the more outages impact that line. The fact that many outages impact a single line indicates that that line is important to the system. The maximum entry in a row indicates the greatest impact that any outage has on that line. For the ITS, the impact is measured with LODF values. The OTS measures impact with change in flow as a percentage of the line's limit. The average tracking structure row entry gives an indication of how much each outage impacts a line.

The metrics can be broadly divided into two groups: row-based metrics and entry-based metrics. The row-based metrics calculate values on a per row basis, and the entry-based metrics calculate values based on the entries into the tracking structures.

Each of these metrics can be given a precise mathematical definition. The frequency of appearance for a line α, F_α, is defined as the number of times a line shows up in a tracking structure. The frequency is calculated by counting the number of times line α appears in the tracking structure. The length of a row in the tracking structure for line α, L_α, is simply the number of elements in the row for line α. The maximum tracking structure entry for line α's row is the

maximum of the absolute values of the impact values in the structure:

$$M_\alpha = \max_{row\ \alpha} |I| \tag{8.1}$$

The average is similarly the average impact, I, for a row

$$A_\alpha = \frac{\sum_{row\ \alpha} |I|}{L_\alpha} \tag{8.2}$$

The average value per entry, E, is an element-based metric that is used to characterize the outage of a line on the rest of the system. The average value per entry is calculated by finding the average magnitude of the entries in the tracking structure for a particular line. For example, if line α has six entries in a tracking structure, then E is calculated by finding the average magnitude of the six entries.

8.3 Example System

The IEEE 14-bus test case [24] is used as the sample system for illustrating the various methods of characterizing weak elements. The case has 14 buses and 20 lines. Detailed information about the 14 bus case can be found in Chapter 6, which contains a one-line diagram as well as the matrix of LODF values for the 14-bus case. The ITS for the IEEE 14-bus test case is shown in Figure 8.1. This tracking structure was generated with a d^* of 0.10.

At first glance, it may be observed that the ITS for the IEEE 14-bus test system is broad and tall. With the exception of the radial line 14, every line in the system has several entries in its row. Thus, this ITS indicates a large degree of interaction between the lines in the system.

The row-based metrics for the ITS are given in Table 8.1. The longest row in

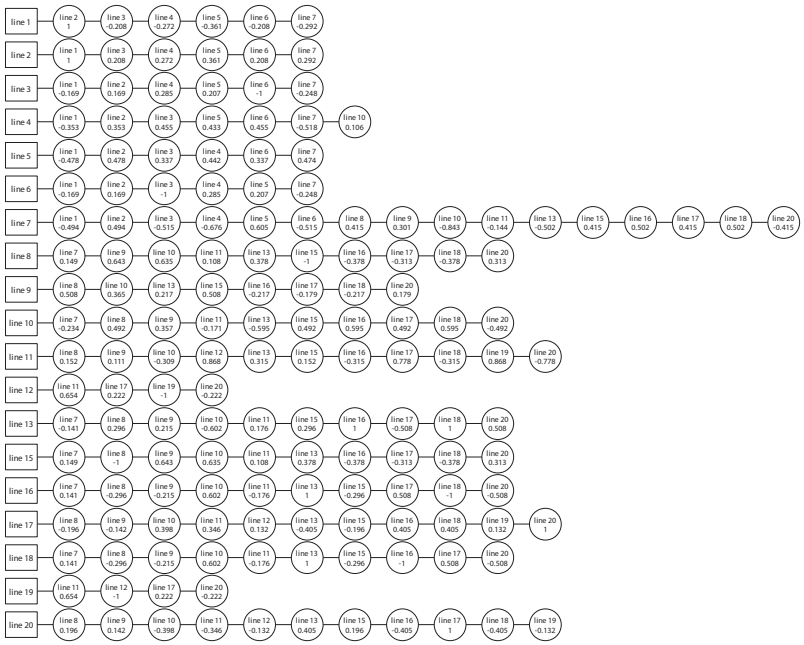

Figure 8.1: IEEE 14-bus ITS

the ITS corresponds to line 7. This is an indication that line 7 is an important line to the system. On nearly every other line in the system, an outage has an impact on line 7. The average value per entry for line 7 is 0.48, meaning that there is a large impact for most outages. The maximum impact from any single outage is 0.843 for line 7. The maximum average impacts occur for line 19 and line 12. Each has an average, A, of 0.5245. The maximum impact is 1.0, and this occurs for many lines. The value of 1.0 indicates that the outage results in the outaged line's flow transferring entirely to the line.

The entry-based metrics for the ITS are shown in Table 8.2. These metrics count the number of times an entry is made into the ITS. For example, line 20 has 12 entries into the ITS. This means that the outage of line 20 impacts 12 separate lines. Based on this metric, line 20 and line 7 have the most impact on the rest of the system. The largest value for the average entry is 0.533. This value occurs for the entries on line 19 and line 4. This indicates that on average,

Table 8.1: Weak-element row metrics for the ITS

Line	L	A	Max
20	11	0.341	1.00
19	4	0.524	1.00
18	10	0.474	1.00
17	11	0.341	1.00
16	10	0.474	1.00
15	10	0.429	1.00
13	10	0.474	1.00
12	4	0.524	1.00
11	11	0.451	0.868
10	10	0.451	0.595
9	8	0.298	0.508
8	10	0.429	1.00
7	16	0.484	0.843
6	6	0.346	1.00
5	6	0.424	0.478
4	7	0.381	0.518
3	6	0.346	1.00
2	6	0.390	1.00
1	6	0.390	1.00

the outage of these lines has the most impact on the rest of the system.

The OTS for the IEEE 14-bus test system is shown in Figure 8.2. The tracking structure was generated using an o^* value of 10%, meaning that outages that result in a postcontingent flow that is within 10% of a line's limit are added to the list.

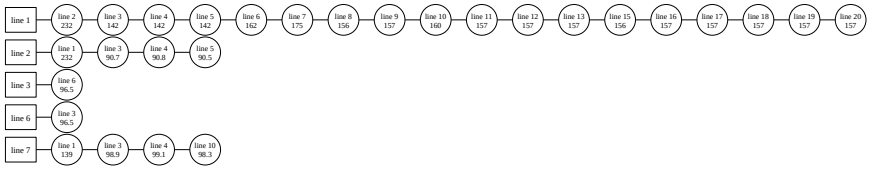

Figure 8.2: OTS for the IEEE 14-bus test case

The row-based metrics for the OTS are shown in Table 8.3. The first striking feature about the row metrics is the fact that there only a few of them. This

Table 8.2: Weak-element entry metrics for the ITS

Line	Count	E
20	12	0.455
19	4	0.533
18	10	0.520
17	12	0.455
16	10	0.520
15	10	0.385
13	10	0.520
12	4	0.533
11	11	0.278
10	11	0.499
9	10	0.298
8	10	0.385
7	12	0.252
6	6	0.454
5	6	0.362
4	6	0.372
3	6	0.454
2	6	0.444
1	6	0.444

reflects the fact that most rows of the OTS have no entries. It is also apparent that the few rows that do have entries are relatively short. The exception is the row corresponding to line 1, which has 18 entries.

Table 8.3: Weak-element row metrics for the OTS

line	L	A	max
7	4	108.825	139
6	1	96.5	96.5
3	1	96.5	96.5
2	4	126	232
1	18	160	232

The element-based metrics for the OTS are given in Table 8.4. These values give information about how often a line appears in the OTS and the value

associated with that entry. Line 3 has the most appearances in the OTS, meaning that an outage on this line is more likely than one one any other line to result in a near overload. The average entry value, E, for line 3 is 107.0, meaning that on average the outage of line 3 results in an overload of 107%.

Table 8.4: Weak-element entry metrics for the OTS

Line	Count	E
20	1	157.0
19	1	157.0
18	1	157.0
17	1	157.0
16	1	157.0
15	1	156.0
13	1	157.0
12	1	157.0
11	1	157.0
10	2	129.2
9	1	157.0
8	1	156.0
7	1	175.0
6	2	129.3
5	2	116.3
4	3	110.6
3	4	107.0
2	1	232.0
1	2	185.5

8.4 Large System

The metrics for weak-element identification were calculated for the large case described in Appendix A. The metrics were calculated for tracking structures built with two of the most successful parameter values. The ITS was constructed using a threshold, d^*, of 0.05. The OTS was constructed using a threshold, o^*, of 0.05. These two threshold values were the most successful at

generating contingency lists that captured contingencies with a list of limited size. Statistics for these structures can be found in Chapter 5. The screening results for the large case can also be found in Chapter 5. There are too many lines in the large system to present the metrics for each line. However, the metrics can be examined to determine the weakest elements in the system. This section discusses the weakest elements in the large system based on the metrics discussed in this chapter.

The row-based metrics for the ITS show that the longest row has 339 entries. These entries are on the row corresponding to line 5428. This indicates that a large number of lines impact line 5428. The average value per entry for line 5428 is 0.131, indicating that outages of many other lines have a substantial impact on line 5428. The maximum impact of any line on line 5428 is 0.618.

The element-based metrics for the ITS show that line 4370 makes 156 appearances in the ITS, the most for any line or transformer. The average magnitude for an entry in the ITS for line 4370 is 0.14, meaning that the outage of this line has an average LODF value of 0.14 onto the other lines in the system. This line could be considered a critical element because it affects so many other branches in the system. The maximum average entry is 1.0, indicating that there are parallel lines in the system that impact only each other.

The maximum row length in the OTS is 49 elements. The line corresponding to this row is line 4477. The fact that this line has 49 entries in its row indicates that there are 49 separate contingencies that result in overloads or near overloads on this line. The average value for an entry on this row is 114.8, which is well over the value of 100 that indicates an overload. The largest value in this row is 208, meaning that there is one outage that results in a postcontingent flow that is over twice the limit of line 4477. The weakness of line 4477 can be put into perspective by considering the next longest row in the OTS. The next

longest row in the OTS has 10 entries, so the row corresponding to line 4477 is almost 5 times longer. The second longest row belongs to line 4234.

There are 109 lines that have entries in the OTS for the large system. The most any element appears in the OTS for the large system is 5 times. Line 4667 has five entries in the OTS, meaning that the outage of this line results in more overloads than any other line in the system. The average value for an entry for line 4667 is 127.1, and the maximum value is 171. Line 974 has the largest average entry value, 232.5.

8.5 Conclusions

This chapter presents several methods of identifying critical elements in a power system. The methods break down to row-based metrics, which characterize how the system affects a line, and element-based metrics, which describe how a line affects the rest of the system. The metrics were presented for the two most successful screening algorithms: the ITS and the OTS. The ITS and OTS tracking structures are based on different information and they give different characterizations of weakness. The ITS is based on LODF values, so the ITS metrics characterize lines based on the impacts they have on each other. The OTS incorporates a padded version of single-outage contingency analysis results. The OTS seems to be a better indicator of an element's weakness. The results from the OTS are much more definitive. Line 4477 is the weakest element in the system. The ITS results are not as clear-cut.

CHAPTER 9

LINEAR FLOW BOUND

In Chapter 2 an expression for the change in flow was found for the outage of one or more lines. This chapter examines the expression for change in line flow (2.31) and uses matrix and vector norm properties to find a bound on the change in flow. The bound may be expressed in terms of the maximum singular value of the inverse of **M**, the norm of the vector of preoutage flows, and the norm of the vector of LODF values onto the line of interest.

First, the bound is derived for an arbitrary number of outages. Then an analytical expression for the bound is found for the double-outage case. The analytical flow bound is compared to the expression for the change in line flow (2.31) to examine the possible sources of error. Empirical results comparing the flow bound with (2.31) are presented for the IEEE 14-bus and the IEEE 118-bus test cases. Statistics are presented for each test case, and maximum error cases examined.

The test case results show that the flow bound works. There are no cases in which the change calculated using (2.31) is greater than the bound on the change. The test case results also show that the error is roughly uniformly distributed among the lines.

The flow bound is an interesting analytical result. In simplified form, it can be used as a quick prescreen for the change in flow. However, the calculation may be slower than simply evaluating (2.31) to find the change in flow, which limits the use of the flow bound for practical purposes.

9.1 Flow Bound Derivation

This section derives an upper bound on the change in flow. The bound is based on analysis of the linear methods developed in Chapter 2. The analysis begins by taking the 2-norm of (2.31):

$$||\Delta f_\alpha||_2 = ||\mathbf{L}_\alpha \mathbf{M}^{-1}\mathbf{F}||_2 \tag{9.1}$$

Then, using a consistent norm (i.e., a norm that obeys the submultiplicative property $||AB|| \leq ||A|| \cdot ||B||$, which includes the p-norms [25]), we can convert the equality into an inequality

$$||\Delta f_\alpha||_2 \leq ||\mathbf{L}_\alpha||_2 ||\mathbf{M}^{-1}\mathbf{F}||_2 \tag{9.2}$$

If $\mathbf{F}$ is rewritten as $\mathbf{F}\frac{||\mathbf{F}||_2}{||\mathbf{F}||_2}$, then the submultiplicative property may be applied to the right-hand side of the previous inequality, and the factors may be grouped by parentheses. This causes the rightmost factor in the right-hand side of (9.2) to become

$$\left\| \left(\mathbf{M}^{-1} \frac{\mathbf{F}}{||\mathbf{F}||_2} \right) (||\mathbf{F}||_2) \right\|_2 \leq \left\| \mathbf{M}^{-1} \frac{\mathbf{F}}{||\mathbf{F}||_2} \right\|_2 \|\mathbf{F}\|_2 \tag{9.3}$$

which we may substitute back into the original expression to give

$$||\Delta f_\alpha||_2 \leq ||\mathbf{L}_\alpha||_2 \left\| \mathbf{M}^{-1} \frac{\mathbf{F}}{||\mathbf{F}||_2} \right\|_2 \|\mathbf{F}\|_2 \tag{9.4}$$

This expression may be simplified because $\frac{\mathbf{F}}{||\mathbf{F}||_2}$ is a unit vector. If a consistent norm is applied to the right-hand side of the previous expression, then the $\frac{\mathbf{F}}{||\mathbf{F}||_2}$ may be discarded to give

$$||\Delta f_\alpha||_2 \leq ||\mathbf{L}_\alpha||_2 \left\|\mathbf{M}^{-1}\right\|_2 \|\mathbf{F}\|_2 \tag{9.5}$$

At this point it may be recognized that the 2-norm of the inverse of $\mathbf{M}$, $||\mathbf{M}||_2$, is the square root of the largest eigenvalue of $\left(\mathbf{M}^{-1}\right)^T\left(\mathbf{M}^{-1}\right)$ [25], which is of course the largest singular value of $\mathbf{M}^{-1}$. Using $\sigma_{max}\left(\mathbf{M}^{-1}\right)$ to represent the largest singular value of $\mathbf{M}^{-1}$, we can write the flow bound as

$$||\Delta f_\alpha||_2 \leq ||\mathbf{L}_\alpha||||\mathbf{F}||\sigma_{max}(\mathbf{M}^{-1}) \tag{9.6}$$

9.1.1 Double-outage flow bound

The expression in (9.6) gives a general-form upper bound on the flow change when using linear sensitivities in terms of the pre-outage flows, the LODFs onto the lines of interest, and the singular values of the matrix $\mathbf{M^{-1}}$.

The above expression works for any number of outages. From this point forward, the focus will be on the double-outage case. For the double-outage case, $\mathbf{M}$ is a 2×2 matrix that can be written as

$$\mathbf{M} = \begin{bmatrix} 1 & -d_{\beta,\delta} \\ -d_{\delta,\beta} & 1 \end{bmatrix} \tag{9.7}$$

where the outaged lines are line β and line δ. If we are interested in the impact on line α, then $\mathbf{L}_\alpha$ can be written as a 1×2 vector that contains the LODFs of the outaged lines onto the line α:

$$\mathbf{L}_\alpha = \begin{bmatrix} d_{\alpha,\beta} & d_{\alpha,\delta} \end{bmatrix} \tag{9.8}$$

$\mathbf{F}$ is a 2×1 vector that contains the preoutage flows on the outaged lines:

$$\mathbf{F} = \begin{bmatrix} f_\beta \\ f_\delta \end{bmatrix} \tag{9.9}$$

Substituting these values into the expression for the flow bound (9.6) and evaluating the 2-norm gives

$$||\Delta f||_2 \leq \left(\sqrt{d_{\alpha,\beta}^2 + d_{\alpha,\delta}^2}\right)\left(\sqrt{f_\beta^2 + f_\delta^2}\right)\sigma_{max}\left(\mathbf{M}^{-1}\right) \tag{9.10}$$

Since (9.10) depends upon the singular values of $\mathbf{M}^{-1}$, we will now focus on finding the singular values of the inverse of $\mathbf{M}$. We will begin by finding the expression for $\mathbf{M}^{-1}$. For the double-outage case, $\mathbf{M}$ is a 2×2 matrix whose inverse can be written as

$$\mathbf{M}^{-1} = \begin{bmatrix} \frac{1}{1-d_{\beta,\delta}d_{\delta,\beta}} & \frac{d_{\beta,\delta}}{1-d_{\beta,\delta}d_{\delta,\beta}} \\ \frac{d_{\delta,\beta}}{1-d_{\beta,\delta}d_{\delta,\beta}} & \frac{1}{1-d_{\beta,\delta}d_{\delta,\beta}} \end{bmatrix} \tag{9.11}$$

Now that we have an expression for $\mathbf{M}^{-1}$, we can find the singular values by calculating the square roots of the eigenvalues of

$$\mathbf{T} = \left(\mathbf{M}^{-1}\right)^T\left(\mathbf{M}^{-1}\right) \tag{9.12}$$

Substituting in for $\mathbf{M}^{-1}$ gives

$$\mathbf{T} = \begin{bmatrix} \frac{1}{1-d_{\beta,\delta}d_{\delta,\beta}} & \frac{d_{\beta,\delta}}{1-d_{\beta,\delta}d_{\delta,\beta}} \\ \frac{d_{\delta,\beta}}{1-d_{\beta,\delta}d_{\delta,\beta}} & \frac{1}{1-d_{\beta,\delta}d_{\delta,\beta}} \end{bmatrix}^T \begin{bmatrix} \frac{1}{1-d_{\beta,\delta}d_{\delta,\beta}} & \frac{d_{\beta,\delta}}{1-d_{\beta,\delta}d_{\delta,\beta}} \\ \frac{d_{\delta,\beta}}{1-d_{\beta,\delta}d_{\delta,\beta}} & \frac{1}{1-d_{\beta,\delta}d_{\delta,\beta}} \end{bmatrix} \tag{9.13}$$

which can be reduced to

$$\mathbf{T} = \frac{1}{\left(-1 + d_{\beta,\delta} \cdot d_{\delta,\beta}\right)^2}\begin{bmatrix} 1 + (d_{\delta,\beta})^2 & d_{\beta,\delta} \cdot d_{\delta,\beta} \\ d_{\beta,\delta} \cdot d_{\delta,\beta} & 1 + (d_{\beta,\delta})^2 \end{bmatrix} \tag{9.14}$$

Now that we have found an expression for $\mathbf{T}$, we need to find the eigenvalues of

$\mathbf{T}$, which can be found by solving

$$\det(\lambda \mathbf{I} - \mathbf{T}) = 0 \tag{9.15}$$

which can be expanded to

$$\left(-\lambda + \sqrt{\frac{1 + d_{\beta,\delta}^2}{\left(-1 + d_{\beta,\delta} \cdot d_{\delta,\beta}\right)^2}}\right)\left(-\lambda + \sqrt{\frac{1 + d_{\delta,\beta}^2}{\left(-1 + d_{\beta,\delta} \cdot d_{\delta,\beta}\right)^2}}\right) - \frac{d_{\beta,\delta} + d_{\delta,\beta}}{\left(-1 + d_{\beta,\delta} \cdot d_{\delta,\beta}\right)} = 0 \tag{9.16}$$

which can be expanded to

$$\frac{-1 + d_{\beta,\delta} \cdot d_{\delta,\beta} + \lambda^2(-1 + d_{\beta,\delta} \cdot d_{\delta,\beta} -)\lambda(-2 + d_{\beta,\delta}^2 + d_{\delta,\beta}^2)}{-1 + d_{\beta,\delta} \cdot d_{\delta,\beta}} \tag{9.17}$$

which, when solved for λ, gives eigenvalues of

$$\lambda_1 = \frac{2 + d_{\beta,\delta}^2 + d_{\delta,\beta}^2 - (d_{\beta,\delta} + d_{\delta,\beta})\sqrt{4 + (d_{\beta,\delta} - d_{\delta,\beta})^2}\,(d_{\beta,\delta} + d_{\delta,\beta})}{2\left(-1 + d_{\delta,\beta} \cdot d_{\beta,\delta}\right)^2} \tag{9.18}$$

and

$$\lambda_2 = \frac{2 + d_{\beta,\delta}^2 + d_{\delta,\beta}^2 + (d_{\beta,\delta} + d_{\delta,\beta})\sqrt{4 + (d_{\beta,\delta} - d_{\delta,\beta})^2}\,(d_{\beta,\delta} + d_{\delta,\beta})}{2\left(-1 + d_{\delta,\beta} \cdot d_{\beta,\delta}\right)^2} \tag{9.19}$$

The singular values can be found by taking the square roots of the eigenvalues. Thus, $\sigma_1 = \sqrt{\lambda_1}$ and $\sigma_2 = \sqrt{\lambda_2}$. This finally allows the flow bound to be expressed as

$$||\Delta f||_2 \leq \left(\sqrt{d_{\alpha,\beta}^2 + d_{\alpha,\delta}^2}\right)\left(\sqrt{f_\beta^2 + f_\delta^2}\right)\max\left(\sigma_1, \sigma_2\right) \tag{9.20}$$

The right-hand side of 9.20 is an upper bound on the change in flow. For convinence, it is refered to as B.

Since the signs on the individual LODF values $d_{\delta,\beta}$ and $d_{\beta,\delta}$ may be positive or

negative, it is not possible to know ahead of time which of the singular values will be greater. However, the difference between (9.18) and (9.19) can be taken to determine which of the values is greater. Subtracting (9.18) from (9.19) gives

$$\lambda_1 - \lambda_2 = -\frac{\sqrt{4 + (d_{\beta,\delta} - d_{\delta,\beta})^2}(d_{\beta,\delta} + d_{\delta,\beta})}{(-1 + d_{\beta,\delta}d_{\delta,\beta})^2} \tag{9.21}$$

which will be positive in the event that λ_1 is greater than λ_2 and negative in the event that λ_2 is greater than λ_1.

Now, we may use the fact that LODF values must be between 1 and -1 to determine the cases when λ_1 is greater than λ_2 and vice versa. First, note that $\sqrt{4 + (d_{\beta,\delta} - d_{\delta,\beta})^2}$ should always be real, because $(d_{\beta,\delta} - d_{\delta,\beta})^2$ is always less than 4. The denominator of (9.21) should always be greater than or equal to zero. In the extreme case that both LODF values are 1.0, the denominator will be zero. However, in that case, the matrix $\mathbf{M}$ is singular and its inverse does not exist. This degenerate case, which is associated with islanding, is discussed in detail in Chapter 3. So, for nonislanding cases, the denominator will always be positive. Thus, the determining factor in the sign is

$$-(d_{\beta,\delta} + d_{\delta,\beta}) \tag{9.22}$$

When (9.22) is positive, λ_1 is the largest eigenvalue and $\sqrt{\lambda_1}$ is the largest singular value. When (9.22) is negative λ_2 is the largest eigenvalue, and $\sqrt{\lambda_2}$ is the largest singular value. Table 9.1 has a breakdown of the sign of (9.22) with respect to the possible signs of the LODF values, which can be used to predetermine the maximum singular value instead of calculating both values and comparing them. This can save considerable computational effort.

Table 9.1: Table of signs

	$d_{\delta,\beta} < 0$	$d_{\delta,\beta} = 0$	$d_{\delta,\beta} > 0$
$d_{\beta,\delta} < 0$	+	+	+ if $d_{\beta,\delta} > d_{\delta,\beta}$
$d_{\beta,\delta} = 0$	+		–
$d_{\beta,\delta} > 0$	+ if $d_{\delta,\beta} > d_{\beta,\delta}$	–	–

9.2 IEEE 14-Bus Test Case Results

The flow bound described in Section 9.1.1 was calculated for every double outage for the IEEE 14-bus test case [24]. For the IEEE 14-bus test case there are 190 double outages, which are generated from the double-outage combinations of the 20 lines in the system. The flow bound is compared to the predicted change in flow using linear methods (2.31).

The error between the flow bound $||\Delta f||_2$ and the predicted change in dc flow f_{dc} is defined as

$$e = ||\Delta f||_2 - |\Delta f_{dc}| \tag{9.23}$$

The absolute value of the flow change calculated using (2.31) is taken for comparison with the flow bound. Although the change in flow may have any sign, the flow bound must be positive. So, to compare the values, the absolute value of the flow bound is used in (9.23). It may be noted that e should always be positive because the flow bound should always be greater than the actual change. Also, the closer the error is to 0, the tighter the flow bound.

For the experimental data, the minimum value of error is 0.0, and the maximum value is 163 MW. The maximum error occurs on the line from bus 2 to bus 3 for the outages of the line from bus 1 to bus 2 and the line from bus 3 to bus 4. These lines are shown highlighted in Figure 9.1. The outaged lines are highlighted in red. The line on which the maximum error occurs is highlighted in blue.

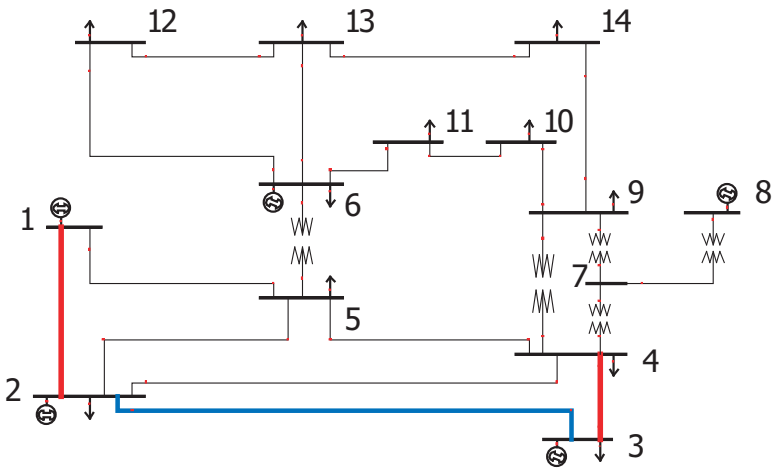

Figure 9.1: Maximum error

For the maximum error case, the **L** vector is

$$\mathbf{L} = [-0.169 \quad -1.0] \tag{9.24}$$

the **f** vector is

$$\mathbf{f} = [147.9 \quad -24.15]^T (\text{MW}) \tag{9.25}$$

and **M** is

$$\mathbf{M} = \begin{bmatrix} 1 & 0.208 \\ 0.169 & 1 \end{bmatrix} \tag{9.26}$$

Using these values the flow bound (9.6) can be evaluated as

$$||\Delta f||_2 = 187.15 \text{ MW} \tag{9.27}$$

which is much larger than the actual change in flow calculated using dc methods,

$$||\Delta f_{dc}||_2 = 24.15 MW \tag{9.28}$$

The flow bound on the other lines for the outage of the lines between bus 1

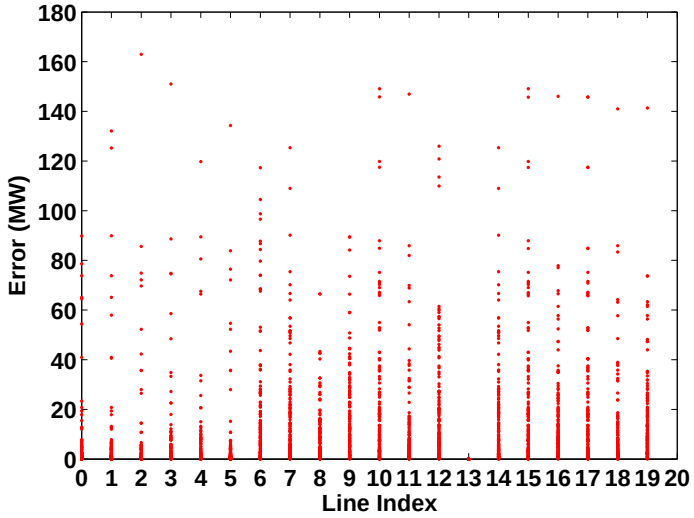

Figure 9.2: Error vs. line index

and bus 2 and the line between bus 3 and bus 4 is much tighter, which indicates that the error in the flow bound is caused by a factor other than the matrix $\mathbf{M}$.

To explore the behavior of the flow bound, several relationships were explored. Figure 9.2 shows a plot of error versus line index. This plot shows that the error is roughly uniformly distributed among the lines. This indicates that the error is coming from all the lines, meaning that there is no particular line contributing a large fraction of the error. Aside from the conclusion that the error is roughly uniformly distributed, the most interesting feature of Figure 9.2 is the fact that line index 13 has very small errors. Line index 13 corresponds to the line between bus 7 and bus 8, which is a radial line that connects a generator bus to the system. The fact that the line is radial and the generator output is 0.0 MW, meaning the line has no flow, are the reasons the error is so small.

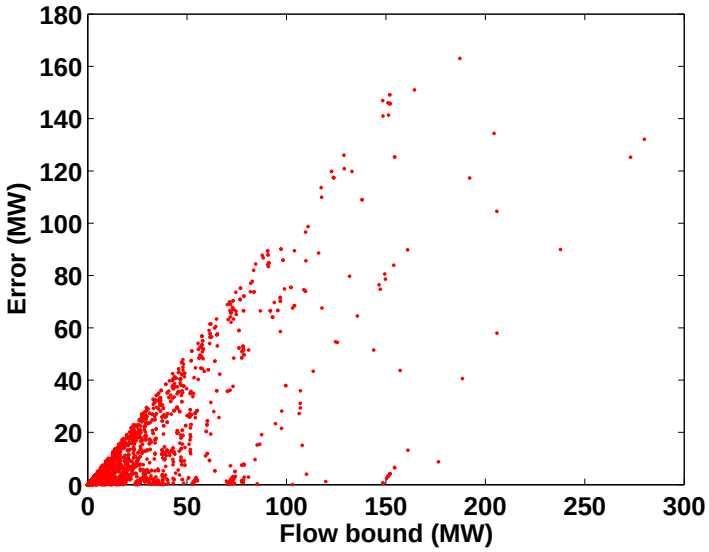

Figure 9.3: Error vs. flow bound

Figure 9.3 shows a more interesting graph. Figure 9.3 shows a scatter plot of the error versus the flow bound. This plot has the interesting feature that the data lie entirely below the $y = x$ line. This is a graphical indication that the error is always less than the flow bound. This should be expected, because having points above the $y = x$ line would indicate that the error, e, is greater than the flow bound, which would mean that $||f_{dc}||$ would have to be negative. Another interesting feature in Figure 9.3 are the parallel lines that seem to appear.

Figure 9.4 shows plots of the error density and the error distribution. These plots show that error is fairly well distributed over a wide range. This is indicated in the density plot by the fact that the value near 0 is small — only around 0.06. The distribution shows the integral of the density plot, so the fact that the error is widely distributed is indicated by the slowness with which the curve approaches 1.0.

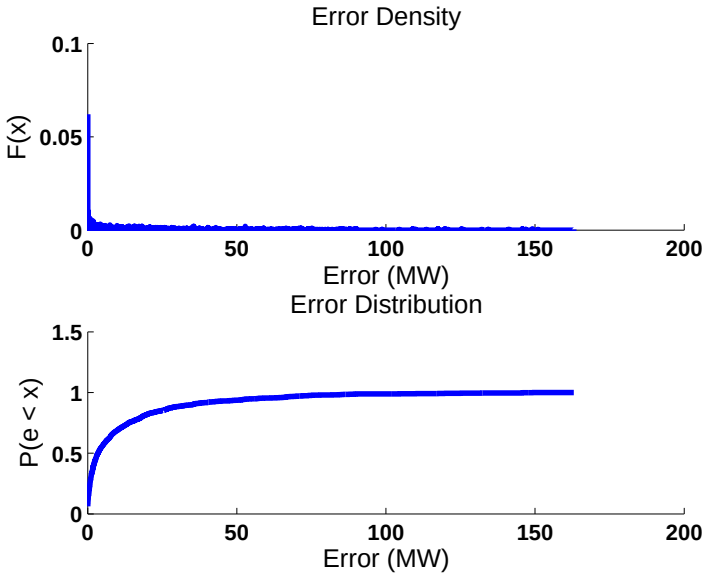

Figure 9.4: Error distribution

CHAPTER 10

GEOMETRY SHADER FLOW VISUALIZATION

10.1 Introduction

This chapter introduces an efficient means of visualizing line flow information, which can be useful for visualization of real-time information and speeding up existing one-line diagrams. Rendering speed is one of the bottle necks for wide-area visualization, and large numbers of flow arrows can negatively impact the rendering speed of a one-line diagram. This chapter focuses on speeding up one of the most computationally intensive parts of a modern power system visualization: generating flow arrows. Generating large numbers of flow arrows is one of the most time-consuming processes involved in rendering a one-line diagram because of the sheer number of them that can be visible. Using traditional techniques, the position of each flow arrow must be calculated every time the screen is rendered, and large diagrams can have millions of flow arrows. This means millions of positions must be calculated and their proper vertices sent out to be rendered. While flow arrows are the specific case discussed in this chapter, the techniques are applicable to other power system one-line objects (e.g., pie charts). The techniques can also be applied to other types of visualizations entirely. For example, the visualization techniques can also be adapted to visualize contingency information, such as the information in the ITS.

The methods discussed in this chapter utilize a new feature in modern graphics hardware known as a geometry shader to generate line flow symbols

quickly. Introduced in the GeForce 8 series graphics cards from Nvidia, geometry shaders are a new feature that allow new vertices to be created after vertices have been transformed [54]. Before the introduction of the geometry shader, the vertex shader could move and color a vertex, but it could not create new vertices. The ability to create new vertices is ideal for the generation of flow arrows because the burden of positioning each flow arrow can be moved onto the graphical processing unit (GPU). Drawing flow arrows becomes a matter of enabling a shader and redrawing the transmission lines.

The advantage of using a GPU to generate the flow arrows is that it has the ability to generate many flow arrows in parallel, which is much quicker than determining the flow arrow size, speed, and direction on the central processing unit (CPU) before they are drawn. A detailed comparison of the two algorithms is discussed later in this chapter.

The ultimate objective of this research is to increase the operational awareness of power system operators, which was cited as one cause of the August 2003 blackout [55]. The increase in rendering speed allowed by these methods allows for more frequently updated graphics. This does more than simply improve user experience. It allows new sensor data, such as the measurements being generated by phasor measurement units (PMUs), to be incorporated into power system visualizations in real-time.

The speed of new sensors such as PMUs means that the rendering speed of power system visualization is becoming more important. PMUs report data roughly 60 times faster than the traditional supervisory control and data acquisition (SCADA) systems. SCADA systems retrieve new data roughly every two seconds, while PMUs are capable of reporting new measurements 30 times per second. The work in [56] combines traditional SCADA data with PMU data to create real-time displays, and the work in [57] uses GPU techniques to

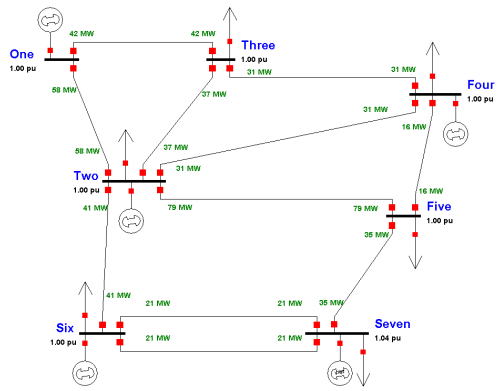

Figure 10.1: One-line diagram using text fields

dramatically speed up the generation of power system contours, which are a very powerful technique to display power system information [58]. The work presented in this chapter is another step toward having one-line displays capable of advanced visualization techniques rendered at interactive rates.

10.2 Flow Arrow Background

In traditional power system visualizations, line flows are listed as text fields alongside a line. The sign of field gives the direction of the flow, which depends on the definition of the line, and the magnitude of the field tells how much power is flowing on the line. The flow as a percentage of the line's limit may also be included. Figure 10.1 shows a one-line diagram that uses text fields to indicate line flows. This type of display has all the necessary information; however, it is difficult to get a picture of the overall behavior of a system by looking at a one-line digram using only text fields to describe the flow. This is because it takes a large amount of attention and concentration from the operator to read and interpret the line flow fields and form a picture of what is happening in the system. Flow arrows were introduced as a way to quickly

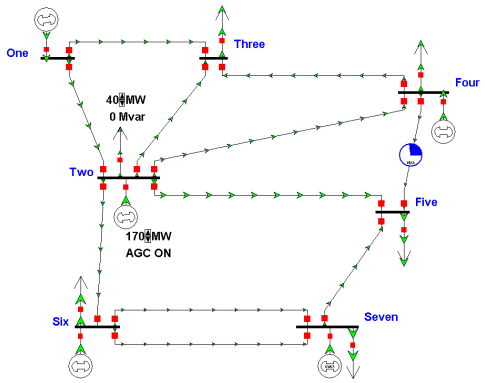

Figure 10.2: One-line diagram using flow arrows

convey information about how power is flowing in the system [58]. Flow arrows give an impression of the direction of power flow without the need to examine the signs of text-based fields [59]. Flow arrows are now used in either static or animated form on most one-line diagrams. Figure 10.2 shows a one-line diagram that uses flow arrows to indicate the direction of line flows. By comparing Figure 10.2 and Figure 10.1, it becomes clear that it is much easier to tell line flow directions and relative magnitudes on the diagram using flow arrows. The work in [60] presents a human factors evaluation of the use of flow arrows.

Typically, the size and speed of flow arrows are proportional to the amount of power flowing on the transmission line. Also, flow arrows are usually colored to indicate the line's nearness to its limit. For example, a flow arrow may be colored red to indicate that the flow on that line has exceeded its limit. Orange is usually used to indicate that the line is approaching its limit, and green represents nominal operation. So far, we have described several parameters that affect the flow arrow visualization that are all a function of the flow through the line, which may be calculated using power flow software or provided from measurements in the field.

The relevant flow arrow parameters are defined as follows:

- Flow arrow size, S
- Flow arrow direction, D
- Flow arrow density, ρ
- Flow arrow color, C

10.3 Traditional Flow Arrow Generation

The traditional process for generating flow arrows is to set the parameters based on the line's current flow value and limit. Then, the flow arrow positions are determined along the transmission line, and the arrows are drawn. The traditional algorithm is written below in Algorithm 7. A couple of practical observations can be made about the traditional flow arrow generation algorithm. First, the algorithm will scale with the number of branches in the system. Second, most of the work will be done calculating positions and drawing flow arrows.

Input: Line flow information
Input: Line limit information
Input: Line geometry
Output: Graphical flow arrows
foreach *Transmission line* **do**
    Determine flow arrow parameters from flow and limit information
    foreach *Line Section k* **do**
        Calculate number of flow arrows to draw, N_k
        for *i = 1 to* N_k **do**
            Calculate position
            Draw flow arrow
        end
    end
end

Algorithm 7: Traditional flow arrow generation algorithm

A line section is a part of a transmission line defined by two vertices — the

normal geometric definition of a line segment. A transmission line typically has several sections. The determination of the flow arrow parameters can be done in several ways. However, the computation to determine these parameters should be relatively minimal. Most of the work in the algorithm is done in calculating the positions and drawing the flow arrows. Calculating the positions can be relatively computationally intensive because the rotation of the arrows to match the line can involve sine and cosine calculations. The speed with which flow arrows can be drawn depends upon the memory bandwidth, which will determine how quickly the flow arrow vertices can be sent into the video card. For modern systems, this should not be a bottleneck. However, it should be noted that since OpenGL uses immediate-mode rendering — objects are drawn as soon as the command is issued — the time waiting to send the flow arrow vertices into the video card will be spent idling. Thus, for most implementations using modern hardware, the slowest part of drawing the flow arrows is calculating their positions.

A typical transmission line configuration is shown in Figure 10.3. This transmission line has three sections, and each section has a different length. Section 1 has length l_1, section 2 has length l_2, and section 3 has length l_3. In general a line may have any number of sections, but 3 is a good number for illustrative purposes. Algorithm 7 can be used to place flow arrows along the length of this line.

The first step is to determine the flow arrow parameters from the flow and limit information for the line. This will determine the flow arrow size, S, the flow arrow direction, D, the flow arrow density, ρ, the flow arrow color, C, and the flow arrow speed, v. For the execution of the rest of the algorithm, the most important parameter is ρ, because it determines the number of flow arrows in a line section.

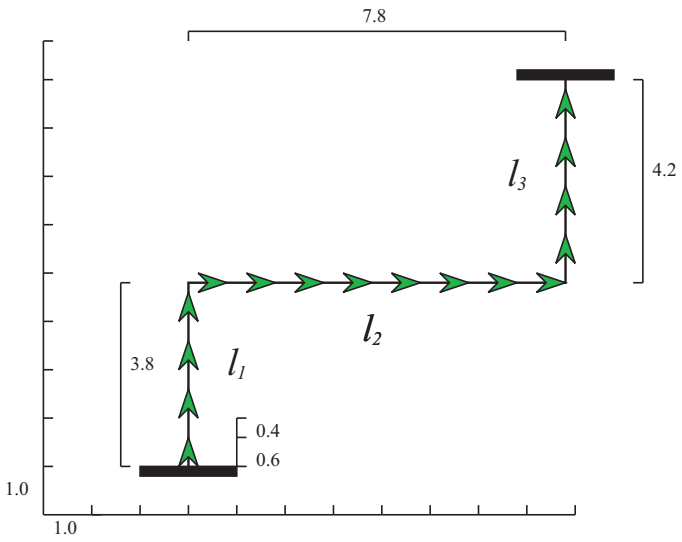

Figure 10.3: Sections of a transmission line

For a given line section k, the number of flow arrows in that section can be calculated using the length of the line section, l_k, the length of the flow arrow head, a, and the length of the arrow shaft from the point of one arrow head to the tail of the next, b. Here it should be noted that ρ is the length of one complete flow arrow

$$\rho = a + b \tag{10.1}$$

Mathematically the number of flow arrowheads in a section can be written as

$$N_k = \lfloor \rho \cdot l_k \rfloor + \lceil \frac{\rho \cdot l_k - \lfloor \rho \cdot l_k \rfloor}{a} \rceil \tag{10.2}$$

This equation says that the number of flow arrowheads in section k is the integer part of the density–length product, plus one more if there is enough room at the end for another arrowhead. The second term uses

$$\rho \cdot l_k - \lfloor \rho \cdot l_k \rfloor \tag{10.3}$$

to calculate the mantissa (i.e., the decimal part) of the density–length product.

By dividing the mantissa by the length of a flow arrowhead, a, it can be determined whether there is enough room on the section to fit another flow arrowhead. Taking the ceiling will give 1 if there is enough room. Otherwise, the ceiling function will return 0.

For the illustration in Figure 10.3, the length of the first line section l_1 is 3.8, and $\rho = 1.0$. This density corresponds to the length of the flow arrowhead itself as well as the length of the spacing in between the flow arrowheads. As shown on the figure, the length of the flow arrowhead is 0.6, and there is a distance of 0.4 between the end of one arrowhead and the beginning of another. By evaluating (10.2), the number of flow arrowsheads that fit in this section can be found. Evaluating the first term of (10.2) gives

$$\lfloor 1.0 \cdot 3.8 \rfloor = 3.0 \tag{10.4}$$

and evaluating the second term gives

$$\lceil \frac{\rho \cdot l_k - \lfloor \rho \cdot l_k \rfloor}{a} \rceil = \lceil \frac{3.8 - 3.0}{0.6} \rceil = 1.0 \tag{10.5}$$

Finally, combining the terms gives the number of flow arrowheads that fit in the first section:

$$N_1 = 3.0 + 1.0 = 4.0 \tag{10.6}$$

So, as shown in Figure 10.3, it is possible to fit four flow arrowheads on the first section. The same process can be repeated to find the number of flow arrowheads on the other two sections. However, there is a small complication. In order to keep the distance between the flow arrows constant, the flow arrowheads on the

following sections must be drawn with an offset. The offset can be calculated as

$$o_{k+1} = l_k - L(N_k) \tag{10.7}$$

where o_{k+1} represents the offset into the next line section, and $L(N_k)$ is a function that calculates the distance required to draw N_k flow arrowheads from the tail of the first arrowhead to the tip of the last arrowhead. The distance function $L(N_k)$ can be written as

$$L(N_k) = N_k \cdot a + (N_k - 1) \cdot b \tag{10.8}$$

For the example shown in Figure 10.3, the distance function can be evaluated as

$$L(4) = 4 \cdot 0.6 + (4 - 1) \cdot 0.4 = 3.6 \tag{10.9}$$

This gives a total offset into the next section of

$$o_2 = 3.8 - 3.6 = 0.2 \tag{10.10}$$

This means that when the flow arrows are drawn on the next section, the first flow arrowhead will begin at 0.2 instead of 0, where 0 is understood to mean the beginning of the line section.

Once the number of flow arrows per section is known, the inner part of the flow arrow generation algorithm can be executed. The execution time of the inner loop will be proportional to the number of flow arrows that need to be generated, which is in turn directly proportional to the line length. It is this part (the inner loop) of the algorithm that can be implemented using the geometry shader.

10.4 GPU Background

The traditional rendering pipeline described in [61] is based on fixed operations. For example, the lighting calculations were done by custom hardware that was designed to do lighting calculations very quickly. In this architecture, a fixed operation was done by a chip and passed down the graphics pipeline to the next chip. This type of fixed architecture has the advantage of being fast, but it is not very flexible. The graphics pipeline has slowly become more flexible over time. It is now possible to use a small program to replace the custom hardware in the pipeline [62]. For example, a small lighting program can be written to replace the "lighting chip." These small programs are referred to as shaders because they are traditionally used to calculate the shading that would result from lighting and other effects. This capability adds a lot of flexibility. With the addition of programmable shaders, developers are no longer restricted to the hardware-provided functionality. They are free to customize selected steps in the pipeline. For example, shaders allow the addition of new lighting models. The first two programmable shaders to be added were the vertex and fragment shaders. A vertex shader acts on a vertex. It is able to alter the position and the color of the vertex. A fragment shader determines the color of the pixels — more generally known as fragments — on the screen. The newest type of shader, called the geometry shader, has the ability to actually create vertices, which makes it well suited to the generation of flow arrows. A line can be sent to the geometry shader, which can then emit vertices representing flow arrowheads.

A good background on the development of the graphics pipeline can be found in [62]. GPUs were developed to relieve the CPU of the need to manipulate pixels directly, and they have been growing in power and versatility since their introduction in the 1990s [62]. Geometry shaders are currently supported only

by the OpenGL shading language (GLSL) [63] and DirectX 10 [64].

GPUs have been greatly increasing in power. In terms of raw floating-point operations per second (flops), modern GPUs have more computing capability than CPUs. However, this power comes with a big caveat. The algorithm must be parallelizable in order to run efficiently on the GPU. GPUs have been designed to perform the calculations needed to render a graphical image, and these calculations are very parallelizable [64].

Along with the high-level limitations of GPU computation, there are limits on the capability of the geometry shader. For example, there is a limit on the number of vertices that the geometry shader can emit [54]. This means that a long line must be divided into pieces if the flow arrows are going to be drawn down its entire length. The limit for emitted vertices on the development hardware (GeForce 8800) is 128. Depending upon the type of flow arrow (a simple triangle or a barbed arrowhead), the number of flow arrows that can be emitted is 42 or 21. Two triangles (a total of six vertices) are needed generate a barbed arrowhead because of the geometry shader's limited output types.

10.5 Geometry Shader Flow Arrow Generation

The geometry shader acts on a per-primitive basis. Primitives are selected types of geometries (e.g., lines, triangles, quadrangles) that OpenGL uses to draw objects. For the purposes of generating flow arrows, the geometry shader is set to output triangles and accept lines. At present, geometry shaders accept only two vertices from a line at a time [54]. This means that the geometry shader is restricted to acting on a section of a line at a time. However, this is not a significant problem, because the geometry shader is able to replace the inner loop of the traditional flow arrow generation algorithm. This allows the

algorithm to execute in parallel. For each section of the line, the geometry shader makes all the flow arrows for that line in parallel. This allows for a more efficient process of generating flow arrows.

Input: Line flow information
Input: Line limit information
Input: Line geometry
Output: Graphical flow arrows
Enable Geometry Shader
foreach *Transmission line* **do**
    Determine flow arrow parameters from flow and limit information
    foreach *Line Section k* **do**
        Calculate number of flow arrows to draw, N_k
        Draw line section
    end
end
Disable Geometry Shader

Algorithm 8: Geometry shader flow arrow generation algorithm

The difference between the traditional algorithm (Algorithm 7) and Algorithm 8 is that the inner part of the loop has been replaced. Instead of calculating positions and sending those positions to the GPU for rendering, Algorithm 9 sends two vertices defining a line section, and the geometry shader generates the flow arrows on the GPU.

The algorithm that executes on the GPU is given in Algorithm 9. The geometry shader program is a relatively simple program, which is a common feature among all shader programs. A vertex program executes for every vertex, and a fragment executes for every fragment. Since there are usually large numbers of vertices and fragments, keeping the programs simple helps to keep rendering speeds fast.

Input: Time t
Input: Offset o
Input: Number of flow arrows N
Input: Line vertex v_1
Input: Line vertex v_2
Input: Color texture CT
Input: Parameter texture PT
Output: Triangles for flow arrows
Calculate direction of line $d = v_2 - v_1$
Calculate length of line $mag = ||d||$
Normalize direction $\widehat{n} = \frac{d}{||d||}$
Define 4 vertices for a barbed flow arrow
$k = 0$
while $k < N$ **do**
    Look up vertex color in CT
    Set position Emit back vertex
    Look up vertex color in CT
    Emit front vertex
    Look up vertex color in CT
    Emit center vertex
    Look up vertex color in CT
    Emit back vertex
    Look up vertex color in CT
    Emit front vertex
    Look up vertex color in CT
    Emit center vertex
end

Algorithm 9: Geometry shader

10.6 Implementation

There are several practical details that must be considered when implementing a geometry shader. The first question is whether the method makes sense for the application. In this case, the analysis above shows that the geometry shader is appropriate for generation of flow arrows. In other words, the geometry shader offers advantages over traditional methods for flow arrow generation. The limitations of the geometry shader must also be considered. The limitations on the input and output primitives should be considered. For the flow arrow application, the geometry shader can accept only a line segment (i.e., two

vertices). It is not able to accept a line strip or list of vertices. This limits the scope of the work that can be done in the geometry shader. Any calculation that requires knowledge above the line segment scope can not be done in the geometry shader. This is why the number of flow arrows per line section is calculated on the CPU instead of inside the geometry shader. There is not enough information inside the geometry shader to do this calculation. The addition of new input parameters would remove this obstacle. The output type is a triangle strip, which fits very well with the flow arrow generation application because flow arrows are essentially stylized triangles.

Another practical consideration is getting the program parameters into the geometry shader. The driver passes the vertex information to the geometry shader when the shader program is enabled, but the flow arrow parameters still need to be set. This means that information needs to be sent into the geometry shader on the GPU. Information can be passed into the geometry shader in a number of ways. There are several built-in variables to which the geometry shader has access in GLSL that can be used to pass information into the geometry shader. For example, texture coordinates can be accessed in the geometry shader after being set in the main program. It is also possible to pass information directly into the shader program using uniform and varying variables. The difference between them is that a uniform variable has a constant value throughout the execution of the geometry shader, while a varying variable is allowed to change [62].

The geometry shader also has access to several texture units, which can be used to store precomputed values that can be used to enhance the digram. For example, the textures can be used to store a color map, which can be used to change the color of the flow arrows as the relative flow. Textures may also be generated at run time to pass information into the geometry shader. Though

texture generation may be a timeconsuming process, the use of textures is best suited to lookup tables such as the previously mentioned color map. If the flow on a transmission line is a certain percentage of that line's limit, then the texture can be used to tell what color the flow arrow should be. The same idea can be used for flow arrow sizes. By knowing what the flow on a line is as a percentage of its limit, the geometry shader can determine the size and color of the flow arrow based on textures. The two textures that are used to store this information are called the color texture (CT) and the parameter texture (PT). The color texture contains a color map, and the parameter texture stores the other parameter values such as size.

Since the geometry shader acts on a section of the line at a time, the implementation presented here cannot handle flows that are not uniform throughout that section. Thus, this method — as it is presented here — cannot show a line consuming Vars, where the reactive line flow is entering on both ends of the line. It should be possible, however, to extend the implementation to fit the flow arrows in an hourglass-shaped envelope. This is left as future work.

The flow arrows shown in Figure 10.4 were generated in the geometry shader using Algorithm 8 and Algorithm 9. Generating these flow arrows was done by passing in the number of flow arrows per section and the offset. Texture coordinates were used to pass this information into the geometry shader. Percentage line flow is also passed into the geometry shader using texture coordinates. The time, which is used to calculate offsets for animating the flow arrows, is passed in using a uniform variable.

Figure 10.5 shows a European system with flow arrows generated on the GPU. The size and color of the flow arrows are determined by texture lookups. This is an example of using the texture coordinates to set the flow arrow parameters. When each transmission line is drawn, a texture coordinate is set, which tells

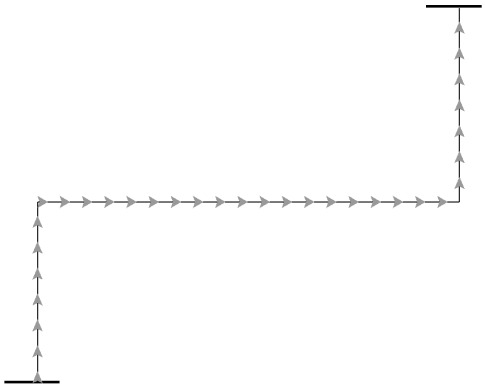

Figure 10.4: Geometry shader – generated flow arrows

the geometry shader where to look inside a texture to find the color and the size of the flow arrows. The textures used when generating this image are shown in Figure 10.6. These textures were used to set the color and the size of the flow arrows. The size texture is very dark because all of the values it contains are relatively small, and values close to 0 appear black. The color and size values could be passed to the geometry shader using uniform variables. However, the use of textures minimizes the amount of information that needs to be passed between the CPU and the GPU, and minimizing this communication improves performance.

10.7 Timing Results

This section compares the geometry shader–based flow arrow generation algorithms with the traditional CPU-based flow arrow generation techniques. The results are presented for two different test cases: the IEEE 118-bus test case and a 1254-bus case representing the European power system. These cases are described in further detail below. The performance of the GPU flow arrow

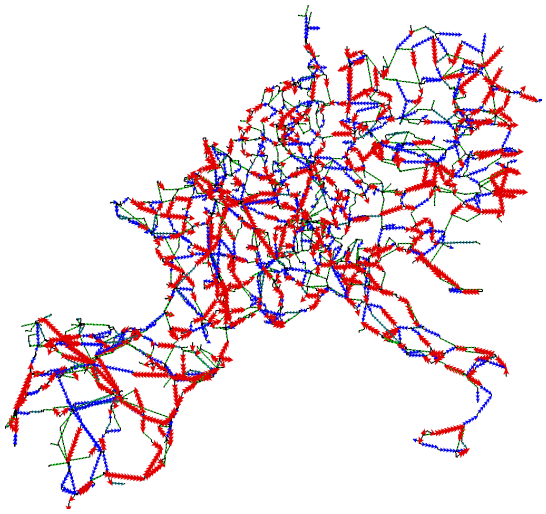

Figure 10.5: Geometry shader – generated flow arrows

Figure 10.6: Color and size textures

generation techniques is discussed for each of the test cases. Finally, conclusions about the algorithms' performance are discussed.

The tests were preformed using two different hardware configurations. The core system consists of an Intel Core 2 Quad CPU running at 2.6 GHz with 6 GB of random access memory. One configuration uses a GeForce 8800 GTS graphics card. The card has 320 megabytes of memory and has a core GPU clock rate of 1.35 GHz. This graphics card is among the first generation of graphics cards to

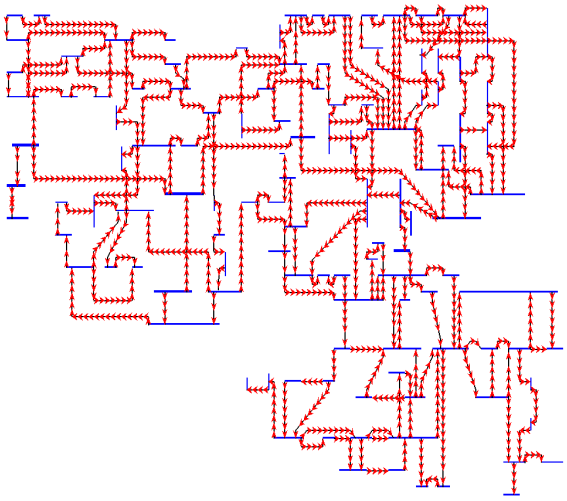

Figure 10.7: IEEE 118-bus case with CPU-generated flow arrows

support geometry shader functionality. It has 96 stream processors, meaning that it has 96 programmable processors to dedicate to parallel shader operation [65]. The second configuration uses a more powerful GeForce GTX 295 with 895 MB of memory and a total of 480 processor cores. The number of processor cores is important because these are the "small processors" that will work in parallel to generate the vertices that constitue flow arrows. Clearly, the GeForce GTX 295 has a strong advantage in the number of processor cores.

The IEEE 118-bus test case has 186 transmission lines. The IEEE 118-bus case makes a good test system because it is moderately sized. It is not so large as to overwhelm the capabilities of either flow arrow generation algorithm. Figure 10.7 shows the one-line diagram with flow arrows generated by the CPU. Figure 10.8 shows the same one-line diagram with flow arrows generated by the geometry shader. The first observation about these figures should be that they are visually identical.

The timing results for the two algorithms are given in Table 10.1. The frame rate in frames per second (fps) is calculated using an average over 1000 frames. Timings were taken for both hardware configurations described in the

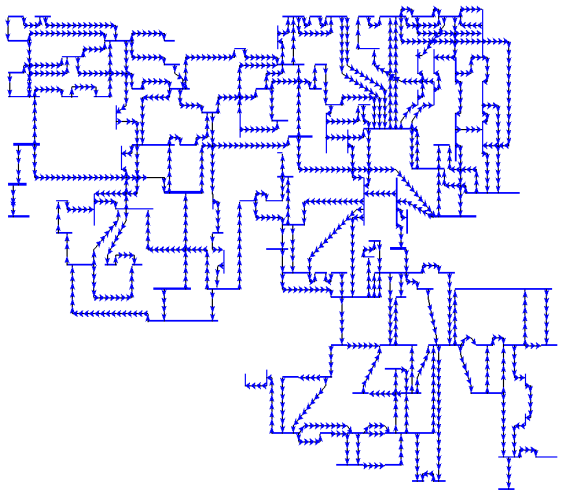

Figure 10.8: IEEE 118-bus case with GPU-generated flow arrows

introduction. Examining these results, it can be seen that for the older GeForce 8800 the two algorithms have very very similar performance. The CPU-based algorithms is faster by about 4.5%, but the results are not strikingly different. Both algorithms preform very quickly. The reason the performances of the CPU-based algorithms and the GPU-based algorithms seem to be so similar for the GeForce 8800 is the limited number of processor cores on the GPU that can be dedicated to flow arrow generation. This hypothesis is supported by the timing results from the GeForce GTX 295, which has a vastly larger number of GPU processor cores. As can be seen in Table 10.1, the GPU-based flow arrow generation algorithm is significantly faster than the CPU-based algorithm. The GPU-based algorithm runs about 2.4 times faster than the CPU flow arrow generation algorithm on the GeForce GTX 295.

Table 10.1: IEEE 118 bus-case timing results

Oneline only (fps)	CPU (fps)	GPU (fps)
GeForce 8800 GTS		
2673.3	761.5	398.4
GeForce GTS 295		
3616.0	755.6	1270.5

The European test case has 1254 buses and 1944 transmission lines, so it is about 100 times larger than the IEEE 118 bus-test case in terms of the number of flow arrows. The fact that the European case is relatively large makes interactive rendering rates a challenge. However, the size makes it a good representation of a wide-area visualization.

The results of the CPU- and GPU-based flow arrow generation algorithms are shown in Figure 10.9 and Figure 10.10, respectively. Again, the results are visually identical. The timing results are presented in Table 10.2. The results show that the CPU-based flow arrow generation algorithms are actually about twice as fast as the GPU-based flow arrow generation on the older GeForce 8800 hardware. However, on the modern GeForce GTX 295, with significantly more processing cores, the GPU-based generation is significantly faster. As with the 118-bus test case results, generating flow arrows on the GPU with the GeForce GTX 295 is about 2.4 times faster than generating the flow arrows on the GeForce 8800. The fact that the modern hardware is much faster than the CPU-based algorithm indicates that the geometry shader is well suited to flow arrow generation, because the geometry shader is able to take advantage of the rapidly increasing power in the GPU.

Table 10.2: European case timing results

Oneline only (fps)	CPU (fps)	GPU (fps)
GeForce 8800 GTS		
681.2	156.5	72.0
GeForce GTS 295		
651.9	155.9	318.6

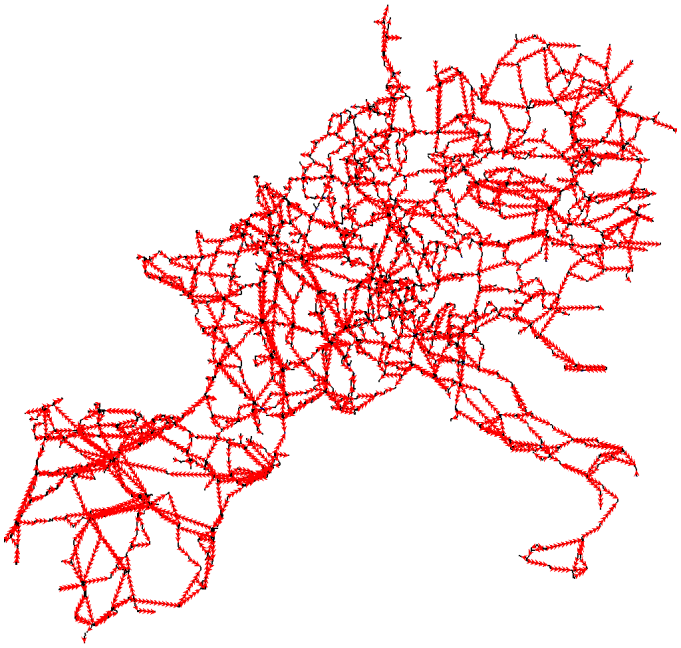

Figure 10.9: European case with CPU-generated flow arrows

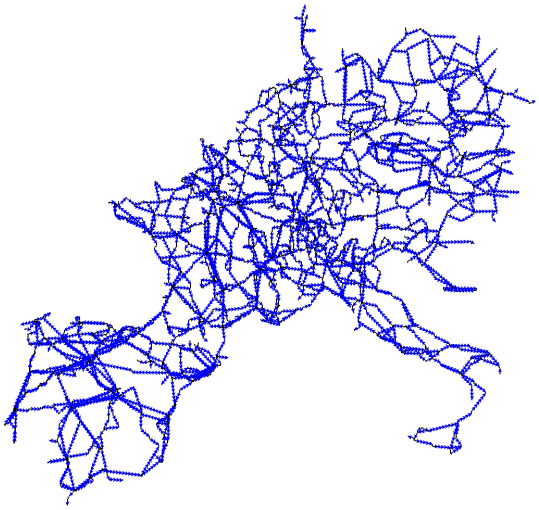

Figure 10.10: European case with GPU-generated flow arrows

10.8 Conclusion

This chapter discussed the use of the geometry shader to visualize flow arrows. The application of geometry shaders to flow arrows works well because it is easy and efficient to generate flow arrows in parallel (on the graphics card). Arguably, generating flow arrows in this method is even easier than the traditional techniques. At least the application code is simpler, and the geometry shader code is the same for every line.

The timing results showed that the geometry shader flow arrow generation method outperforms the CPU-based algorithm on modern hardware by a significant margin. For the 1944-line European case, generating flow arrows using the geometry shader is twice as fast as doing so on the CPU. Although the results are much worse for older hardware, the fact that GPUs have been increasing in power exponentially means that future graphics processors will be even more powerful, and the geometry shader flow arrow generation algorithm is very well suited to future applications.

The geometry shader may also be applied to a large number of power system visualization applications. For example, it should be relatively straightforward to generate buses in the geometry shader. Generating buses could be done by storing the bus position, size, and orientation in memory on the graphics card and using the geometry shader to access these values and generate the buses very efficiently.

CHAPTER 11

CONCLUSION

The main focus of the work in this dissertation was the development of contingency-screening algorithms capable of processing double-outages. However, contributions are also made in the areas of analysis and visualization. The screening algorithms presented in this thesis are very capable. The screening algorithms were evaluated against the full results of the double-outage contingency analysis. Also, various methods are used to generate lists of severe contingencies against which the results of the screening algorithms are checked. The results in Chapter 6 show that they are capable of capturing 99.4% of the double-outage contingencies without including a large number of unimportant events. The various algorithms have different strengths. The overload tracking structure (OTS) algorithms are very sucessful at predicing double outages that will result in violations. However, they are not as strong at detecting double outages that will result in violations only when both lines are outaged. This turns out to be a difficult class of contingencies to capture. The impact-tracking structure (ITS) algorithms are the most successful at capturing this type of contingency. This appears to be because the ITS is designed to generate double-outage contingencies in which both lines impact a third.

The examination of the matrix $\mathbf{M}$ revealed the reasons that the matrix becomes singular in the event of islanding. The insights from this analysis inspired the use of the condition number of the matrix $\mathbf{M}$ as a measure of the coupling of the lines involved in a double outage. Distributions of this metric

show that — as expected — most outages in a large power system involve lines that are nearly decoupled. Analysis was also done to study an upper bound on the flow predicted by the linear methods. The flow bound provides an interesting analytical tool to study the equation for the linear change in flow, and if some approximations are made, it can be made faster to evaluate than the actual equation for linear change.

Work was done to exploit the parallel capabilities of modern graphics processors to generate flow arrows efficiently. This work allows for the generation of flow arrows at interactive rates for large one-lines, which has the potential to increase situational awareness. The timing results show that the GPU-based methods are significantly faster on modern hardware.

APPENDIX A

LARGE SYSTEM DETAILS

The large system represents a portion of the North American Eastern Interconnect with 5395 buses and 7616 lines. The model contains 735 generators and 3443 loads, and there are 214 switched shunts in the system. The load in the system is 126 GW and 103 GVAR. The case is a winter case from a summer peaking system. There are a total of 28 997 920 double-outage contingencies involving lines and transformers.

REFERENCES

[1] M. Shahidehpour, W. Tinney, and Y. Fu, "Impact of security on power systems operation," *Proceedings of the IEEE*, vol. 93, no. 11, pp. 2013–2025, 2005.

[2] R. Christe, B. Wollenberg, and I. Wangensteen, "Transmission management in the deregulated environment," *Proceedings of the IEEE*, vol. 88, pp. 170–195, 2000.

[3] D. Shirmohammadi, B. Wollenberg, A. Vojdani, P. Sandrin, M. Pereira, F. Rahimi, T. Schneider, and B. Stott, "Transmission dispatch and congestion management in the emerging energy market structures," *IEEE Transactions on Power Systems*, vol. 13, no. 4, pp. 1466–1474, Nov. 1998.

[4] Federal Energy Regulatory Comission (FERC), "Order no. 888," 2006. [Online]. Available: http://www.ferc.gov/legal/maj-ord-reg/land-docs/order888.asp

[5] North American Electric Reliability Corporation (NERC), "System performance following loss of two or more bulk electric system elements (category c)," 2007. [Online]. Available: ftp://www.nerc.com/pub/sys/all_updl/standards/rs/TPL-003-0.pdf

[6] F. L. Alvarado, "Computational complexity in power systems," *IEEE Transactions on Power Apparatus and Systems*, vol. 95, no. 4, pp. 1316–1323, Nov. 1976.

[7] A. H. El-Abiad and G. W. Stagg, "Automatic evaluation of power system performance – effects of line and transformer outages," *AIEE Transactions on Power Apparatus and Systems*, vol. 81, pp. 712–716, Feb. 1963.

[8] North American Electric Reliability Corporation (NERC), "Nerc operating manual," 2007. [Online]. Available: http://www.nerc.com/~oc/operatingmanual.html

[9] G. Ejebe, J. Waight, M. Santos-Nieto, and W. Tinney, "Fast calculation of linear available transfer capability," *IEEE Transactions on Power Systems*, vol. 15, no. 3, pp. 1112–1116, Aug. 2000.

[10] S. Grijalva, P. W. Sauer, and J. D. Weber, "Enhancement of linear atc calculations by the incorporation of reactive power flows," *IEEE Transactions on Power Systems*, vol. 18, no. 2, pp. 619–624, May 2003.

[11] J. J. Grainger and W. D. Stevenson Jr., *Power System Analysis*. New York, New York: McGraw Hill, 1994.

[12] W. Tinney and C. Hart, "Power flow solution by newton's method," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-89, no. 11, pp. 1449–1460, Nov. 1967.

[13] B. Stott and O. Alsac, "Fast decoupled load flow," *IEEE Transactions on Power Apparatus and Systems*, vol. 93, no. 3, pp. 859–869, May 1974.

[14] J. Ward and H. Hale, "Digital computer solution of power-flow problems," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-75, no. 3, pp. 398–404, Jan. 1956.

[15] K. Purchala, L. Meeus, D. Van Dommelen, and R. Belmans, "Usefulness of dc power flow for active power flow analysis," in *IEEE PES General Meeting*, vol. 1, Jun. 2005, pp. 454–459.

[16] P. Sauer, "On the formulation of power distribution factors for linear load flow methods," *IEEE Transactions on Power Apparatus and Systems*, vol. 100, no. 2, pp. 760–770, Feb. 1981.

[17] C. MacArthur, "Transmission limitations computed by superposition," *AIEE Transactions on Power Apparatus and Systems*, vol. 80, pp. 827–831, Dec. 1961.

[18] A. J. Wood and B. F. Wollenberg, *Power Generation Operation and Control*. New York, NY: John Wiley and Sons, 1996.

[19] North American Electric Reliability Corporation (NERC), *Transmission Transfer Capability: A Reference Document for Calculating and Reporting the Electric Power Transfer Capability of Interconnected Electric Systems*. NERC, 1995.

[20] S. Grijalva, "Complex flow-based non-linear atc screening," Ph.D. dissertation, University of Illinois at Urbana-Champaign, Urbana, IL, Jul. 2002.

[21] W. Tinney, V. Brandwajn, and S. Chan, "Sparse vector methods," *IEEE Transactions on Power Apparatus and Systems*, vol. 104, no. 2, pp. 295–301, Feb. 1985.

[22] T. Guler and G. Gross, "Detection of island formation and identification of causal factors under multiple line outages," *IEEE Transactions on Power Systems*, vol. 22, no. 2, pp. 505–513, May 2007.

[23] C. M. Davis, T. J. Overbye, and J. D. Weber, "Index of cascadeability using linear contingency evaluation," in *Proc. North American Power Symposium*, Moscow, ID, Aug. 2004, pp. 329–336.

[24] U. of Washington, "Power system test case archive," 2007. [Online]. Available: http://www.ee.washington.edu/research/pstca/

[25] G. H. Golub and C. F. VanLoan, *Matrix Computations*. Johns Hopkins University Press, 1996.

[26] J. Rice, "A theory of condition," *SIAM Journal of Numerical Analysis*, vol. 3, pp. 287–310, 1966.

[27] G. C. Ejebe and B. F. Wollenberg, "Automatic contingency selection," *IEEE Transactions on Power Apparatus and Systems*, vol. 98, no. 1, pp. 97–109, Jan. 1979.

[28] G. Irisarri, D. Levner, and A. Sasson, "Automatic contingency selection for on-line security analysis - real-time tests," *IEEE Transactions on Power Apparatus and Systems*, vol. 98, pp. 1552–1559, Sep. 1979.

[29] T. Mikolinnas and B. Wollenberg, "An advanced contingency selection algorithm," *IEEE Transactions on Power Apparatus Systems*, vol. 100, no. 2, pp. 608–617, Feb. 1981.

[30] G. Irisarri and A. Sasson, "An automatic contingency selection method for on-line security analysis," *IEEE Transactions on Power Apparatus and Systems*, vol. 100, pp. 1838–1844, Apr. 1981.

[31] S. Vemuri and R. Usher, "On-line automatic contingency selection algorithms," *IEEE Transactions on Power Apparatus and Systems*, vol. 102, no. 2, pp. 346–354, Feb. 1983.

[32] "IEEE Standard Terms for Reporting and Analyzing Outage Occurrences and Outage States of Electrical Transmission Facilities," IEEE, Piscataway, NJ, 1987.

[33] B. Stott, O. Alsac, and F. Alvarado, "Analytical and computational improvements in performance-index ranking algorithms for networks," *International Journal of Electrical Power and Energy Systems*, vol. 7, no. 3, pp. 154–160, Jul. 1985.

[34] T. Halpin, R. Fischl, and R. Fink, "Analysis of automatic contingency selection algorithms," *IEEE Transactions on Power Apparatus and Systems*, vol. 103, no. 5, pp. 938–945, May 1984.

[35] L. Fink, K. Carlsen, R. Fischl, and H. Puttgen, "Power system security assessment," in *23rd IEEE Conference on Decision and Control*, vol. 23, Dec. 1984, pp. 478–480.

[36] A. Ekwue, "A review of automatic contingency selection algorithms for online security analysis," in *Third International Conference on Power System Monitoring and Control*, Jun. 1991, pp. 152–155.

[37] S. Greene, I. Dobson, and F. L. Alvarado, "Contingency ranking for voltage collapse via sensitivities from a single nose curve," *IEEE Transactions on Power Systems*, vol. 14, no. 1, pp. 232–240, Feb. 1999.

[38] F. Albuyeh, A. Bose, and B. Heath, "Reactive power considerations in automatic contingency selection," *IEEE Transactions on Power Apparatus and Systems*, vol. 101, no. 1, pp. 107–112, Jan. 1982.

[39] G. Irisarri, S. Hodges, and A. Sasson, "AEP automatic contingency selector: Branch outage impacts on load bus voltage profile," *IEEE Transactions on Power Systems*, vol. 1, no. 6, pp. 37–44, May 1986.

[40] G. C. Ejebe, H. P. V. Meetern, and B. F. Wollenberg, "Fast contingency screening and evaluation for voltage security analysis," *IEEE Transactions on Power Systems*, vol. 3, no. 4, pp. 1582–1590, Nov. 1988.

[41] J. Zhu and G. Xu, "Approach to automatic contingency selection by reactive type performance index," *Proceedings of the IEEE*, vol. 138, pp. 65–68, 1991.

[42] R. Fischl, T. Halpin, and A. Guvenis, "The application of decision theory to contingency selection," *IEEE Transactions on Circuits and Systems*, vol. 29, no. 11, pp. 712–723, Nov. 1982.

[43] R. Fischl, M. Kam, J. Chow, and S. Ricciardi, "Screening power system contingencies using a back-propagation trained multiperceptron," in *IEEE International Symposium on Circuits and Systems*, vol. 1, May 1989, pp. 486–489.

[44] J. Chow, R. Fischl, M. Kam, H. Yan, and S. Ricciardi, "An improved hopfield model for power system contingency classification," in *IEEE International Symposium on Circuits and Systems*, vol. 4, May 1990, pp. 2925–2928.

[45] K. Lo, L. Peng, J. Macqueen, A. Ekwue, and D. Cheng, "Fast real power contingency ranking using a counterpropagation network," *IEEE Transactions on Power Systems*, vol. 13, no. 4, pp. 1259–1264, Nov. 1998.

[46] Q. Chen and J. McCalley, "Identifying high risk n-k contingencies for online security assessment," *IEEE Transactions on Power Systems*, vol. 20, no. 2, pp. 823–834, May 2005.

[47] S. Grijalva and A. Visnesky Jr., "Spatial representation of the effect of new generation on network security," in *IEEE PES General Meeting*, Oct. 2004, pp. 144–149.

[48] S. Grijalva and A. Visnesky Jr., "The effect of generation on network security: spatial representation, metrics, and policy," *IEEE Transactions on Power Systems*, vol. 21, no. 3, pp. 1388–1395, Aug. 2006.

[49] PowerWorld Corporation, "Powerworld," 2009. [Online]. Available: http://www.powerworld.com

[50] NERC, "Nerc iro-006-03 reliability coordination transmission loading relief," 2006. [Online]. Available: ftp://www.nerc.com/pub/sys/all_updl/standards/IRO-006-3.pdf

[51] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. Boston, MA: McGraw Hill, 2001.

[52] D. R. Musser, G. J. Derge, and A. Saini, *STL Tutorial and Reference Guide*, 2nd ed. Addison Wesley, 2001.

[53] SGI, "Standard template library programmer's guide," 2007. [Online]. Available: http://www.sgi.com/tech/stl/index.html

[54] P. Brown and B. Lichtenbelt, "Ext_geometry_shader4," 2007. [Online]. Available: http://www.opengl.org/registry/specs/EXT/geometry_shader4.txt

[55] U.-C. P. S. O. T. Force, "Final report on the august 14th blackout in the united states and canada," 2004. [Online]. Available: https://reports.energy.gov/

[56] R. Klump, R. Wilson, and K. Martin, "Visualizing real-time security threats using hybrid SCADA/PMU measurement displays," in *Proc. 38th Annual Hawaii International Conference on System Sciences (HICSS 2005)*, Jan. 2005.

[57] J. Tate and T. Overbye, "Contouring for power systems using graphical processing units," in *Proc. 41st Annual Hawaii International Conference on System Sciences (HICSS 2008)*, Jan. 2008, p. 168.

[58] J. Weber and T. Overbye, "Voltage contours for power system visualization," *IEEE Transactions on Power Systems*, vol. 15, no. 1, pp. 404–409, Feb. 2000.

[59] M. Laufenberg, "Visualization approaches integrating real-time market data," *Power Systems Conference and Exposition, IEEE PES*, pp. 1550–1555, vol.3., Oct. 2004.

[60] D. Wiegmann, G. Essenberg, T. Overbye, and Y. Sun, "Human factor aspects of power system flow animation," *IEEE Transactions on Power Systems*, vol. 20, no. 3, pp. 1233–1240, Aug. 2005.

[61] D. Shreiner, M. Woo, J. Neider, and T. Davis, *OpenGL Programming Guide*, 5th ed. Reading, MA: Addison Wesley, 2005.

[62] R. Fernando and M. Kilgard, *The Cg Tutorial*. Boston, MA: Addison Wesley, 2003.

[63] R. J. R. et. al., *OpenGL Shading Language*, 2nd ed. Boston, MA: Addison Wesley, 2006.

[64] H. Nguyen, Ed., *GPU gems 3*. Nvidia, 2008.

[65] Nvidia Corporation, "Geforce 8800," 2009. [Online]. Available: http://www.nvidia.com/page/geforce_8800.html

AUTHOR'S BIOGRAPHY

Charles Davis (Matt) was born in Hot Springs, AR, on May 6, 1980, to Tom and Suzan. He has one younger brother, Andrew, and one younger sister, Beth. Matt attended Louisiana Tech University in Ruston, LA, where he received a B.S. in electrical engineering, summa cum laude, in May 2002. He then received an M.S. in electrical engineering from the University of Illinois at Urbana-Champaign in August 2005, applying linear factors for power system security. Matt's research interests are in power system security, computation, and visualization. He has published four conference papers.

Matt has worked part-time at PowerWorld Corporation since starting graduate school, where he as worked on several projects involving power system modeling, security and visualization. During the 2005-06 academic year Matt received the Grainger Outstanding Power Engineering Student Award.