ECEN 615
Methods of Electric Power Systems Analysis

Lecture 11: Sparse Vector Methods, Contingency Analysis, Sensitivity Methods

Prof. Tom Overbye

Dept. of Electrical and Computer Engineering

Texas A&M University

overbye@tamu.edu

Announcements

- Read Chapter 7 from the book (the term reliability is now used instead of security)
- Homework 3 should be done before the exam put does not need to be turned in
- First exam is Tuesday October 8 in class; closed book, closed notes. One 8.5 by 11 inch note sheet and calculators allowed
  - The exam from 2018 has been posted

Sparse Vector Methods

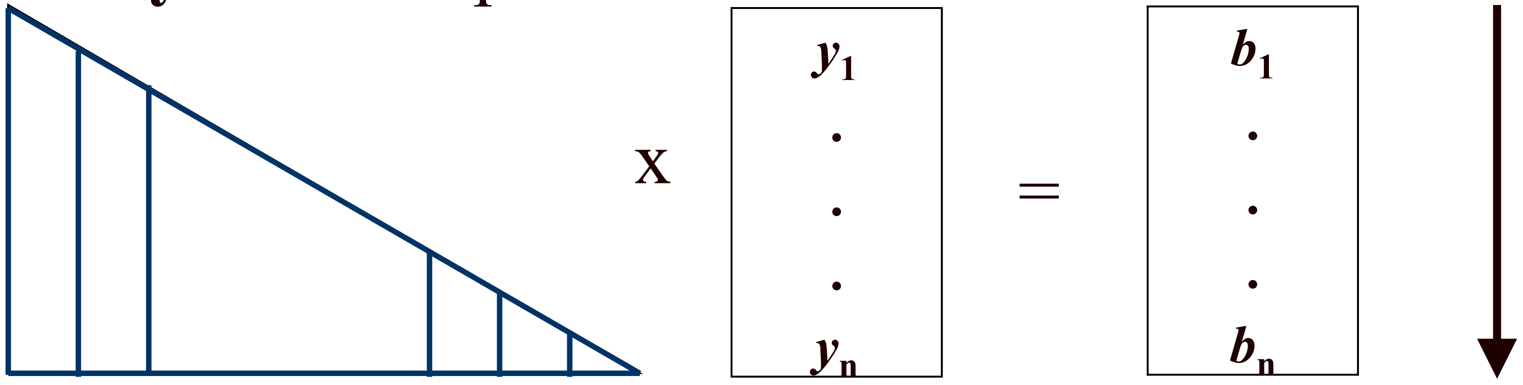
- Sparse vector methods are useful for cases in solving **Ax**=**b** in which
  - **A** is sparse
  - **b** is sparse
  - only certain elements of **x** are needed
- In these right circumstances sparse vector methods can result in extremely fast solutions!
- A common example is to find selected elements of the inverse of **A**, such as diagonal elements.

Sparse Vector Methods

- Often times multiple solutions with varying **b** values are required
  - **A** only needs to be factored once, with its factored form used many times
- Key reference is

W.F. Tinney, V. Brandwajn, and S.M. Chan, "Sparse Vector Methods", *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-104, no. 2, February 1985, pp. 295-300

Sparse Vector Methods Introduced

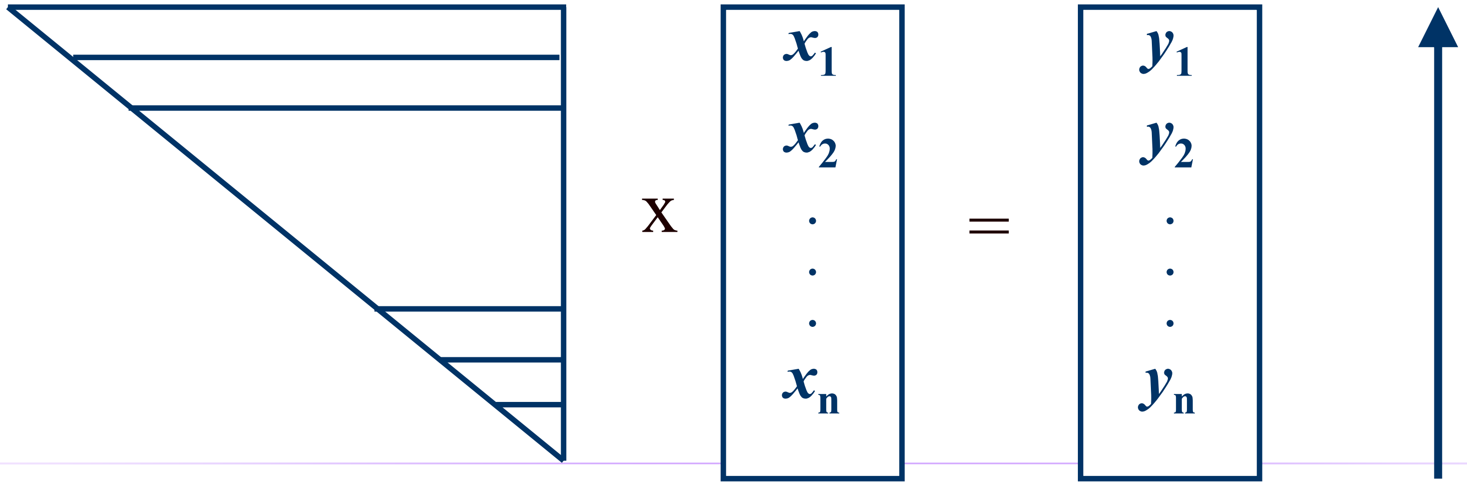
- Assume we are solving **Ax = b** with **A** factored so we solve **LUx = b** by first doing the forward substitution to solve **Ly = b** and then the backward substitution to solve **Ux = y**
- A key insight: In the solution of **Ly = b** if **b** is sparse then only certain columns of **L** are required, and **y** is often sparse
- 

Fast Forward Substitution

- If **b** is sparse, then the fast forward (FF) substitution takes advantage of the fact that we only need certain columns of **L**
- We define {FF} as the set of columns of L needed for the solution of **Ly** = **b**; this is equal to the nonzero elements of **y**
- In general the solution of **Ux = y** will NOT result in **x** being a sparse vector
- However, oftentimes only certain elements of **x** are desired
  - E.g., the sensitivity of the flows on certain lines to a change in generation at a single bus; or a diagonal of $\mathbf{A}^{-1}$

Fast Backward Substitution

- In the case in which only certain elements of **x** are desired, then we only need to use certain rows in **U** below the desired elements of **x**; define these columns as {FB}
- This is known as a fast backward substitution (FB), which is used to replace the standard backward substitution



Factorization Paths

- We observe that
  - {FF} depends on the sparsity structures of **L** and **b**
  - {FB} depends on the sparsity structures of **U** and **x**
- The idea of the factorization path provides a systematic way to construct these sets
- A factorization path is an ordered set of nodes associated with the structure of the matrix
- For FF the factorization path provides an ordered list of the columns of **L**
- For FB the factorization path provides an ordered list of the rows of **U**

Factorization Path

- The factorization path (f.p.) is traversed in the forward direction for FF and in the reverse direction for FB
  - Factorization paths should be built using doubly linked lists
- A singleton vector is a vector with just one nonzero element. If this value is equal to one then it is a unit vector as well.

Factorization Path, cont.

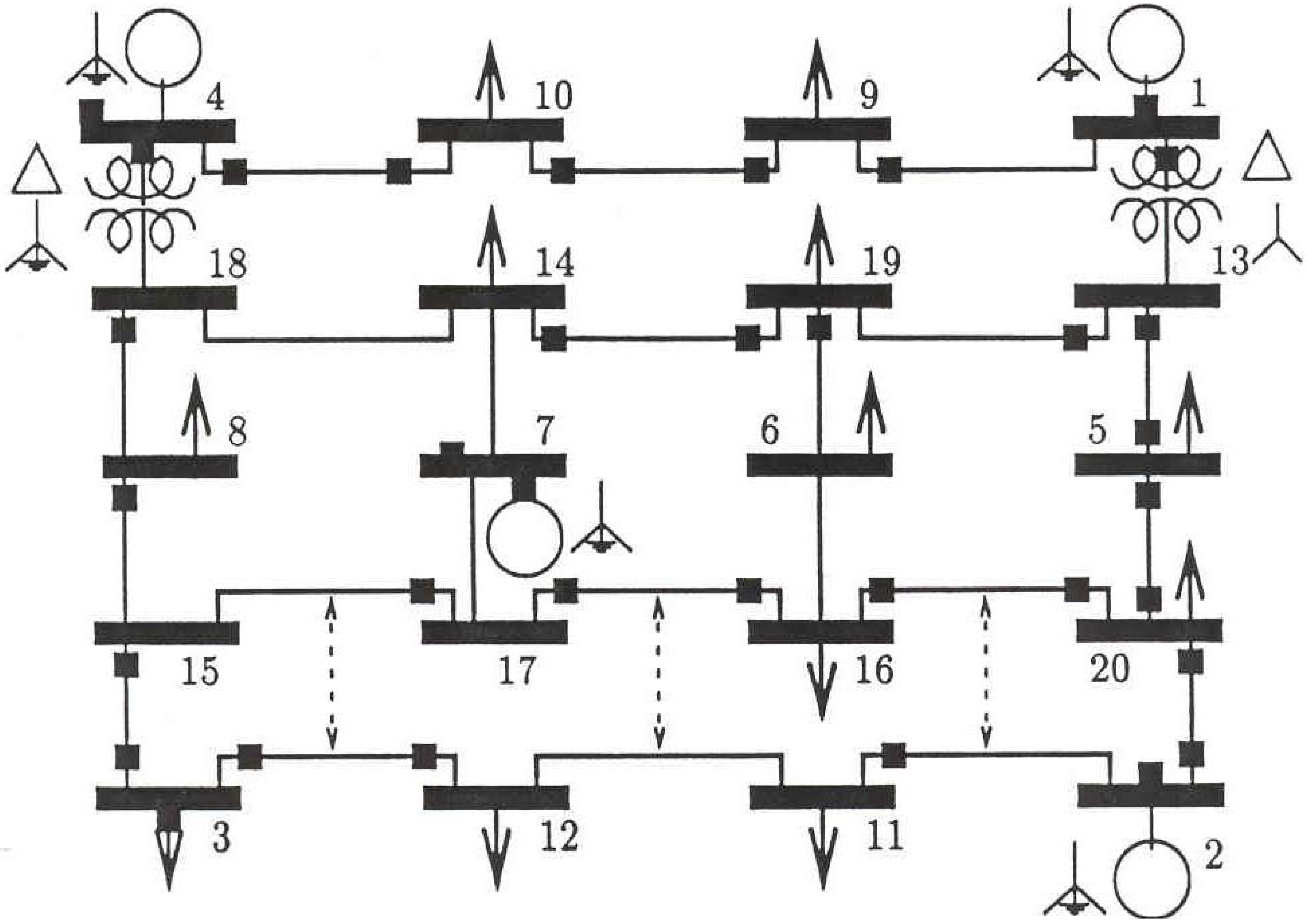
- With a sparse matrix structure ordered based upon the permutation vector order the path for a singleton with a now zero at position arow is build using the following code:

```
p1:= rowDiag[arow];
 While p1 <> nil Do Begin
  AddToPath(p1.col);   // Setup a doubly linked list!
  p1 := rowDiag[p1.col].next;
 End;
```

Path Table and Path Graph

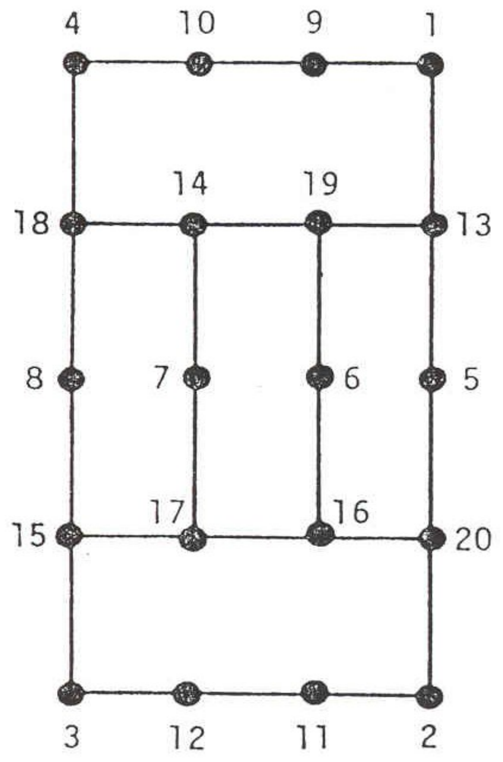
- The factorization path table is a vector that tells the next element in the factorization path for each row in the matrix
- The factorization path graph shows a pictorial view of the path table

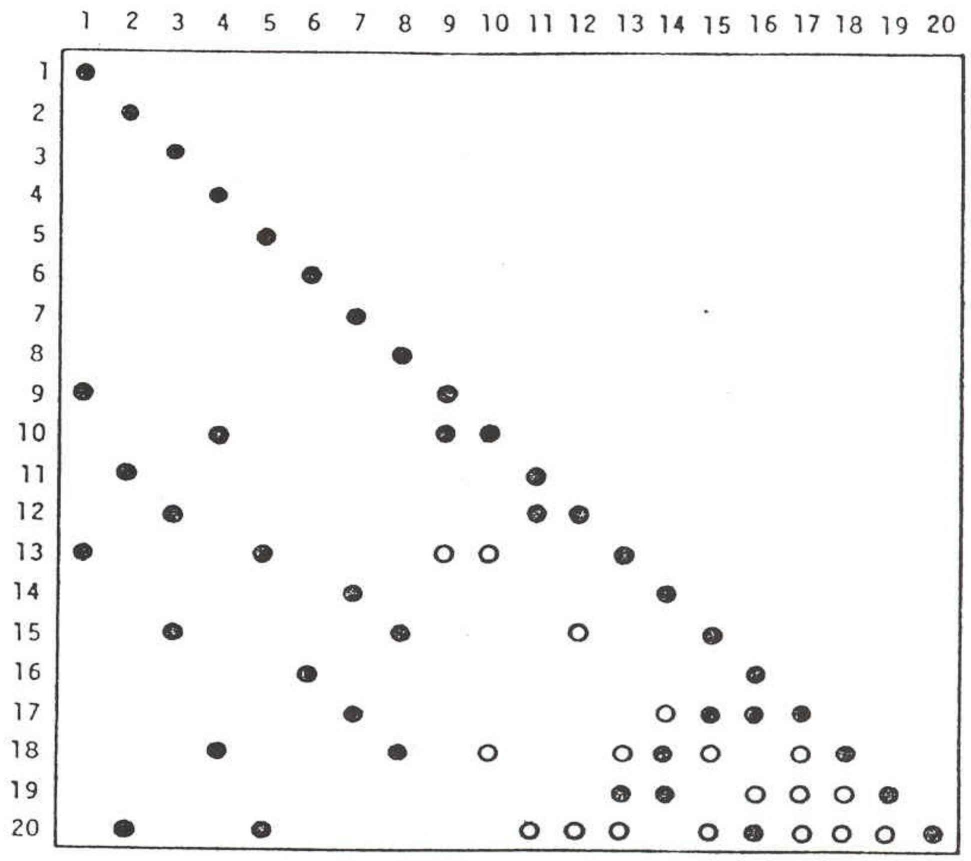
20 Bus Example



20 Bus Example

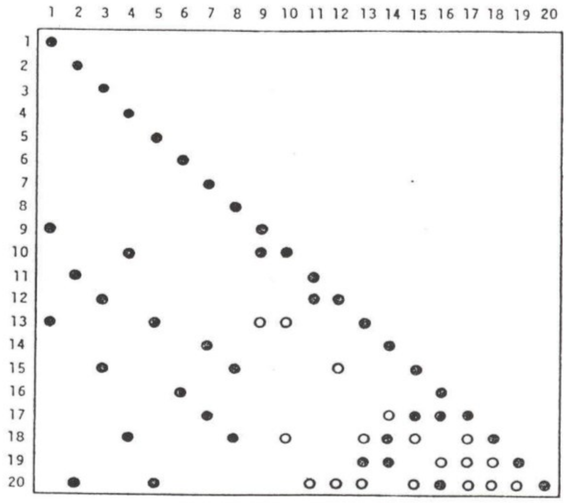
Only showing **L**

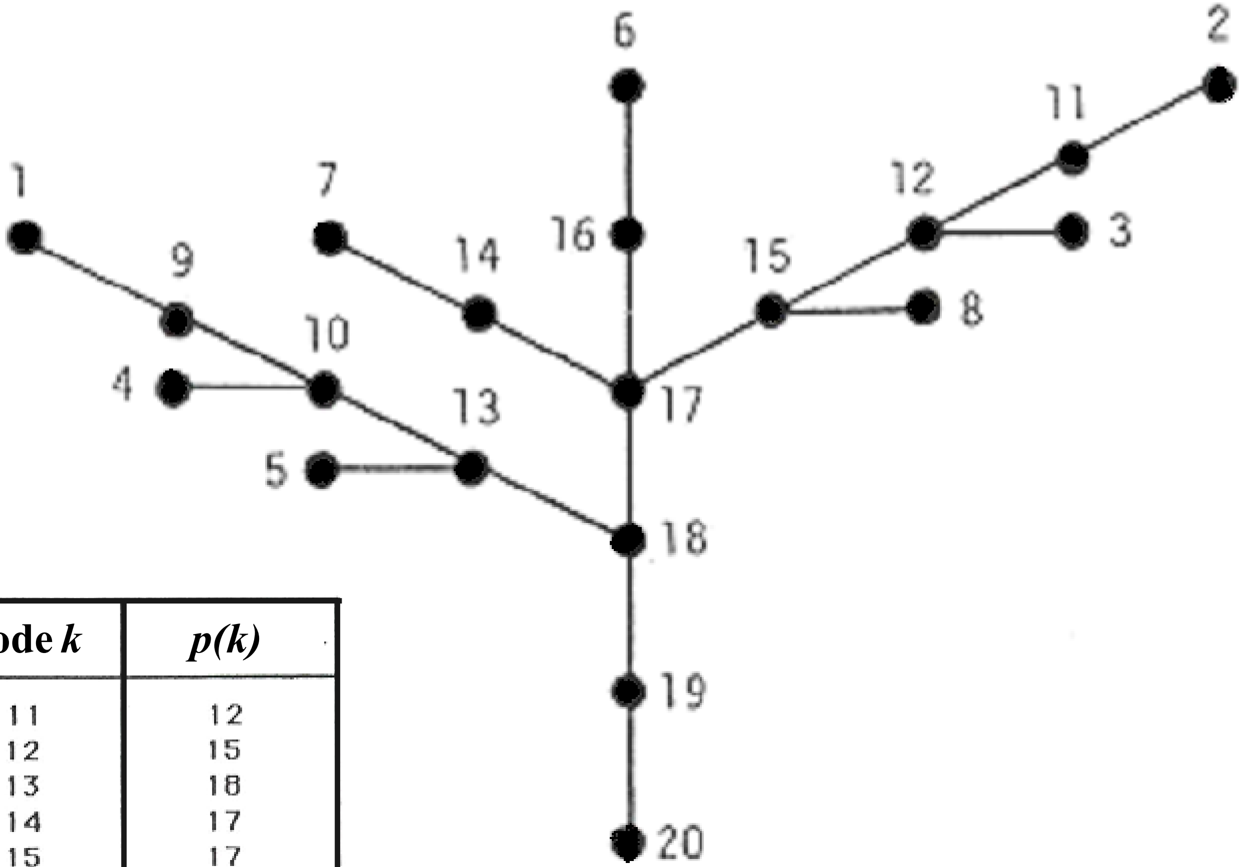




● Original Element (A and L)
○ Fill-In Element (L only)

20 Bus Example





node *k*	*p(k)*	node *k*	*p(k)*
1	9	11	12
2	11	12	15
3	12	13	18
4	10	14	17
5	13	15	17
6	16	16	17
7	14	17	18
8	15	18	19
9	10	19	20
10	13	20	0

20 Bus Example

- Suppose we wish to evaluate a sparse vector with the nonzero elements for components 2, 6, 7, and 12
- From the path table or path graph, we obtain the following

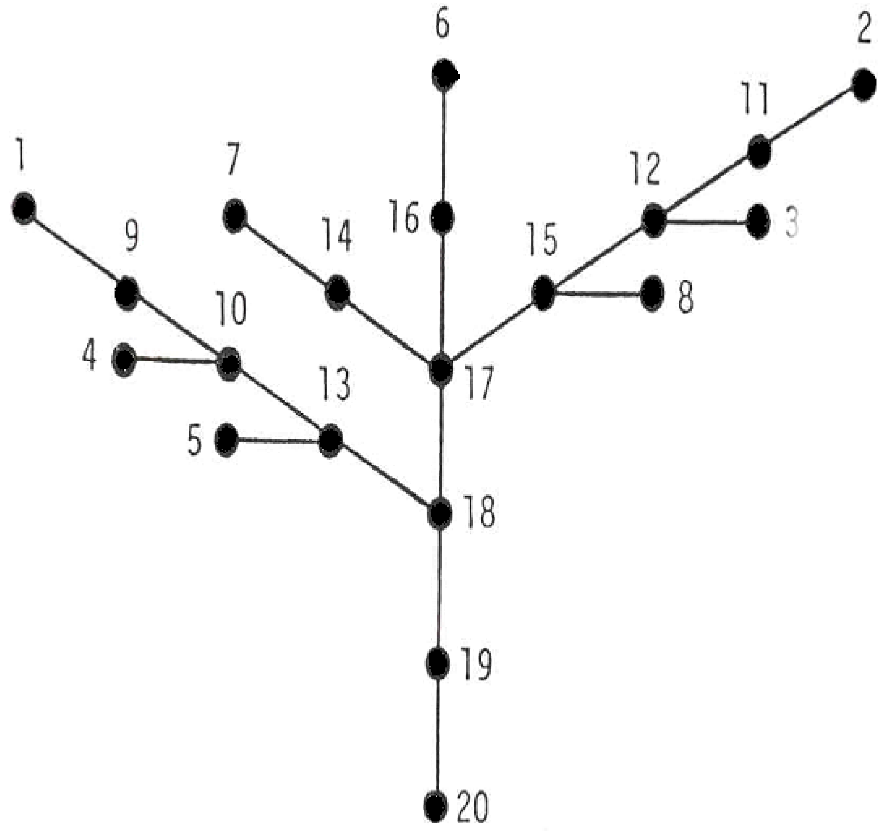
$$\begin{aligned} 2 &\rightarrow \textit{f.p.}\ \{2, 11, 12, 15, 17, 18, 19, 20\} \\ 6 &\rightarrow \textit{f.p.}\ \{6, 16, 17, 18, 19, 20\} \\ 7 &\rightarrow \textit{f.p.}\ \{7, 14, 17, 18, 19, 20\} \\ 12 &\rightarrow \textit{f.p. already contained in that for node } 2 \end{aligned}$$

- This gives the following path elements

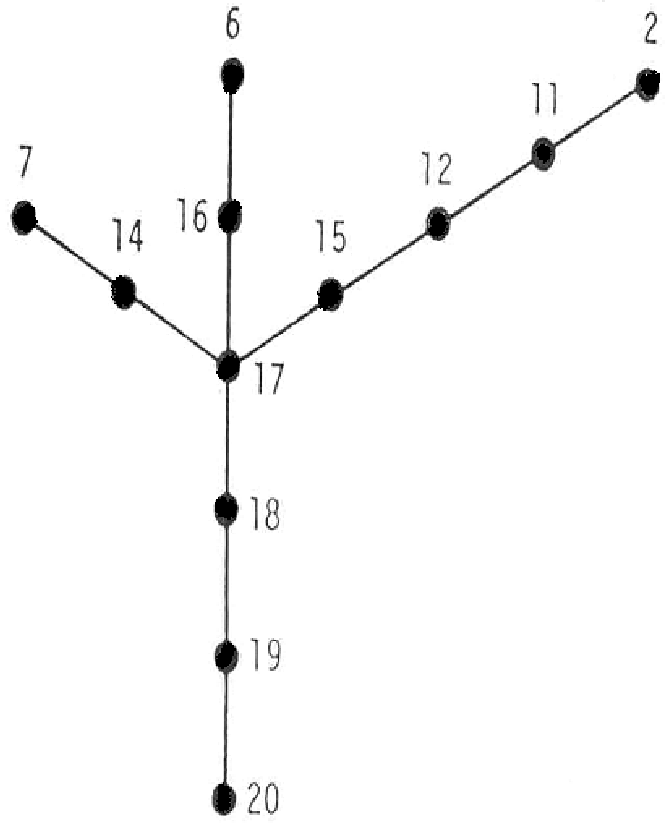
$$\{7, 14, 6, 16, 2, 11, 12, 15, 17, 18, 19, 20\}$$

20 Bus Example

Full path



Desired subset

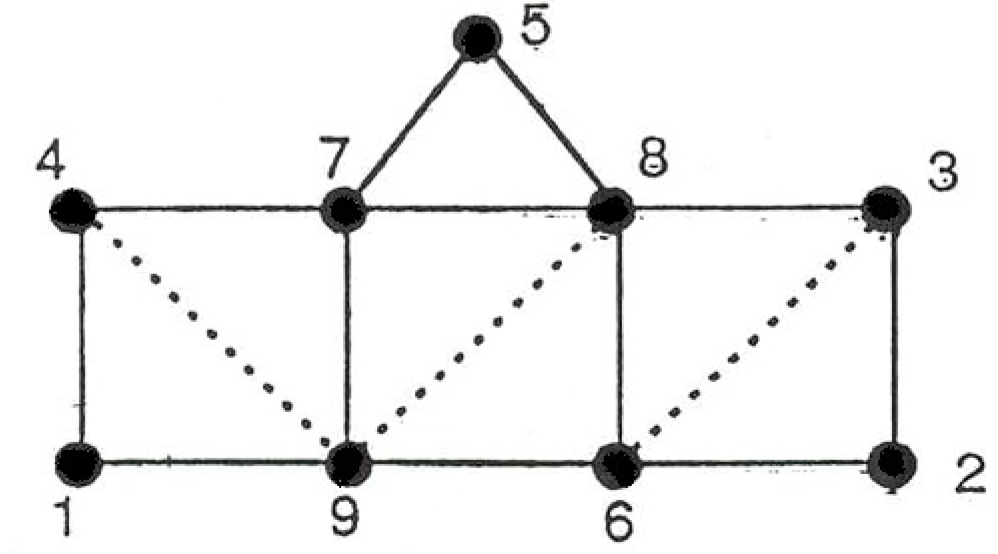


Remarks

- Since various permutations may be used to order a particular path subgroup, a precedence rule is introduced to make the ordering valid
- This involves some sorting operation; for the *FF*, the order value of a node cannot be computed until the order values of all lower numbered nodes are defined
- The order of processing branches above a junction node is arbitrary; for each branch, however, the precedence rule in force applies
- We can paraphrase this statement as: perform first everything above the junction point using the precedence ordering in each branch

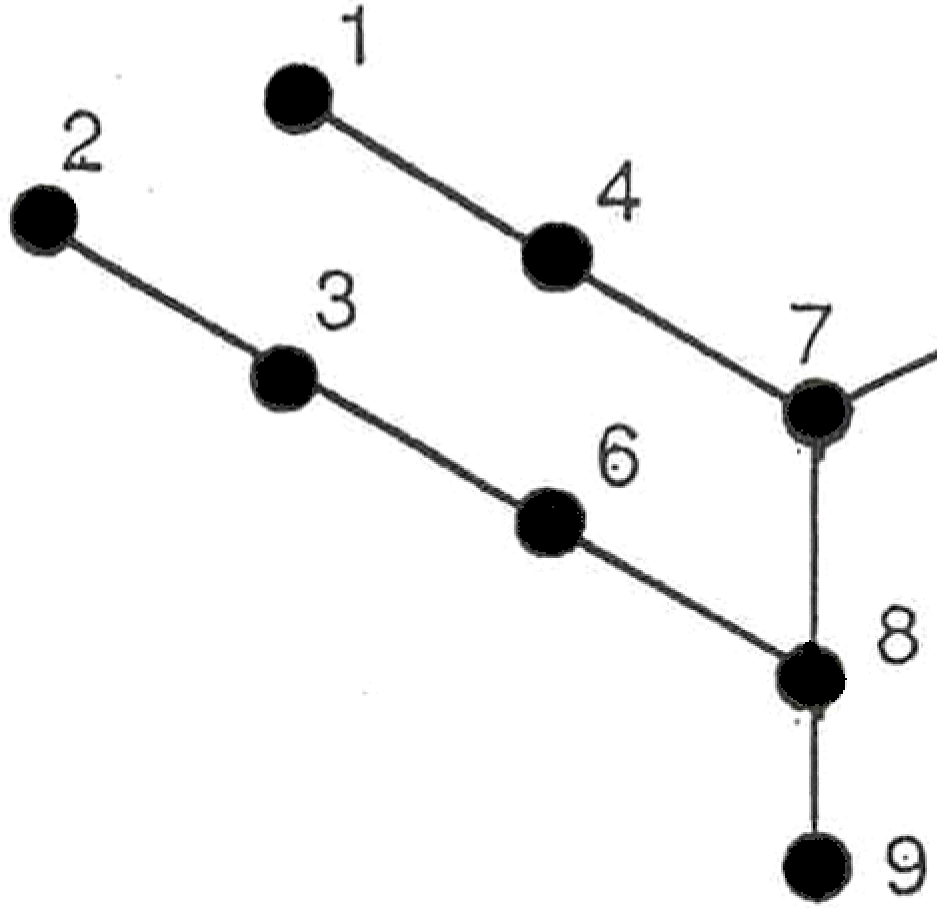
Nine Bus Example

- We next consider the example of the 9-bus network shown below



- For the given ordering, the sparsity structure leads to the following path graph and the table

Nine Bus Example

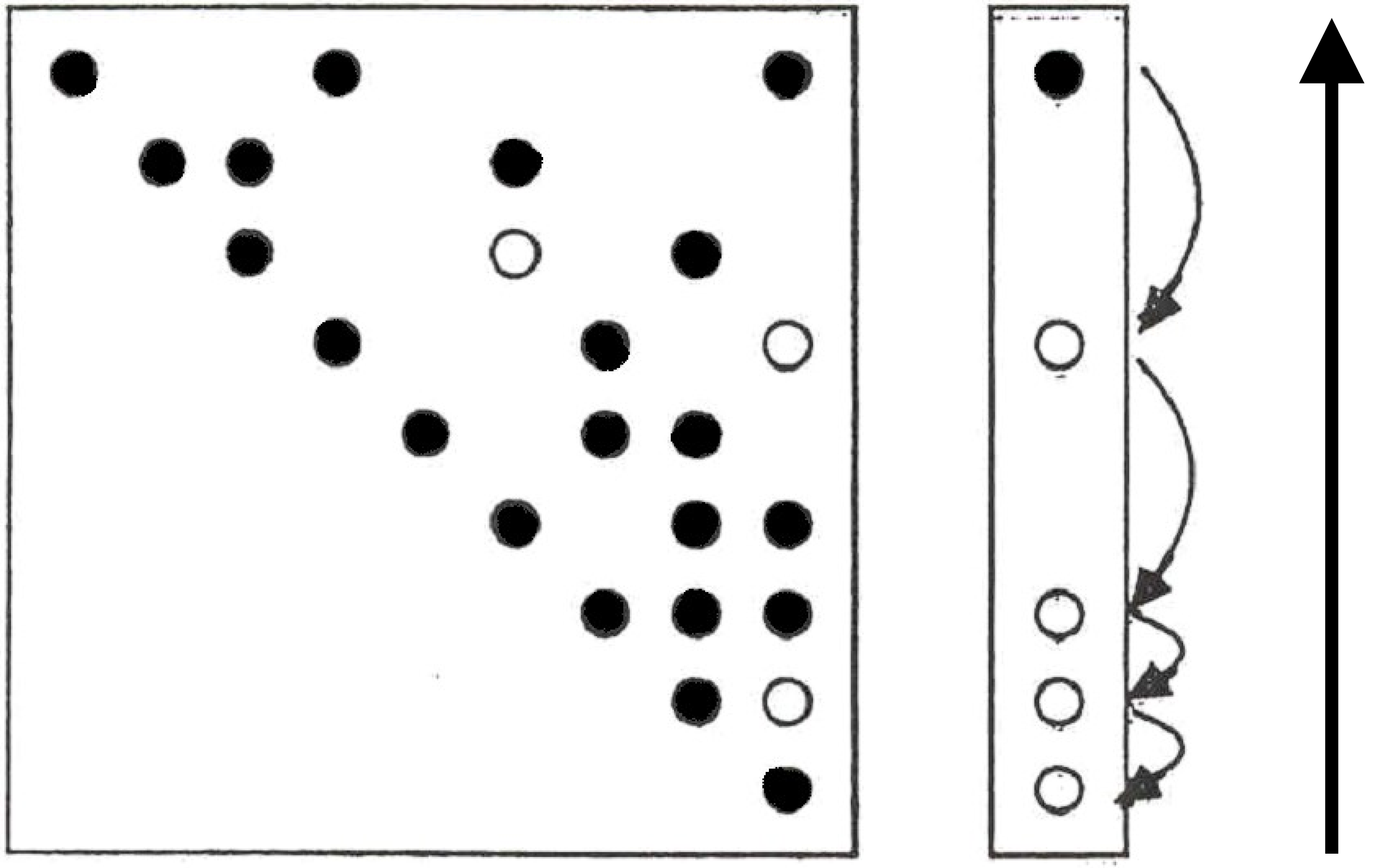


k	*p(k)*
1	4
2	3
3	6
4	7
5	7
6	8
7	8
8	9
9	0

Nine Bus Example

- Suppose next we are interested in the value determination of only component, node 1
  - That is, calculating a diagonal of the inverse of the original matrix
- *FF* involves going down the path from 1-4-7-8-9, and the *FB* requires coming back up, 9-8-7-4-1
- This example makes evident the savings in operations we may realize from the effective use of a sparse vector scheme

Nine Bus Example



Example Application

- In ongoing geomagnetic disturbance modeling work we need to determine the sensitivity of the results to the assumed substation grounding resistance
  - Since the induced voltages are quasi-dc, the network is modeled by setting up the conductance matrix $\mathbf{G} = \mathbf{R}^{-1}$
  - Initial work focused on calculating the driving point impedance values, which required knowing diagonal elements of **R**, which were easily calculated with sparse vector methods
  - But R_{ii} depends on the assumed grounding values are nearby substations, so we need to determine this impact as well; so we'd like small blocks of the inverse of **R**, which will require using the unions of the factorization paths to get some R_{ij}

Ordering for Shorter Paths

- The paper 1990 IEEE Transactions on Power Systems paper “Partitioned Sparse A^{-1} Methods” (by Alvarado, Yu and Betancourt) they introduce ordering methods for decreasing the length of the factorization paths
- Factorization paths also indicate the degree to which parallel processing could be used in solving $\mathbf{Ax} = \mathbf{b}$ by **LU** factorization
  - Operations in the various paths could be performed in parallel

Acknowledgement
We thank Mr. Brian Johnson for helping produce the sparse matrix topology maps.

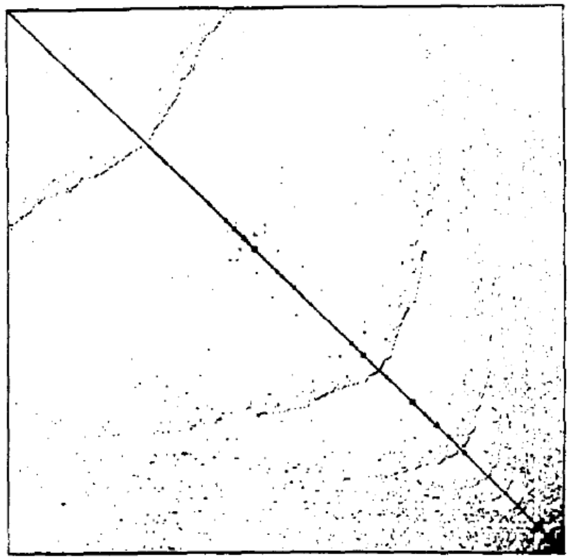

(a) Ordering according to Tinney Scheme 2

Image from Alvarado 1990 paper

Computation with Complex and Blocked Matrices

- In the previous analysis we have implicitly assumed that the values involved were real numbers (stored as singles or doubles in memory)
- Nothing in the previous analysis prevents using other data structures for analysis
  - Complex numbers would be needed if factoring the bus admittance matrix ($\mathbf{Y}_{\text{bus}}$); this is directly supported in some programming languages and can be easily added to others; all values are complex numbers
  - Two by two block matrices are common for power flow Jacobian factorization; for this we use 2 by 2 blocks in the matrices and 2 by 1 blocks in the vectors

2 by 2 Block Matrix Computation

- By treating our data structures as two by two blocks, we reduce the computation required to add fills substantially
  - Half the number of rows, and four times fewer elements
- Overall computation is reduced somewhat since we have four times fewer elements, but we do have more computation per element

2 by 2 Block Matrix Example

- In the backward substitution we had

```
For i := n downto 1 Do Begin
  k = rowPerm[i];
  p1 := rowDiag[k].next;
  While p1 <> nil Do Begin
    bvector[k] = bvector[k] – p1.value*bvector[p1.col];
    p1 := p1.next;
  End;
  bvector[k] := bvector[k]/rowDiag[k].value;
End;
```

2 by 2 Block Matrix Example

- We replace the scalar bvector entries by objects with fields .r and .i (for the real and imaginary parts) and we replace the p1.value field with four fields .ul, .ur, .ll and .lr corresponding to the upper left, upper right, lower left and lower right values.
- The first multiply goes from

```
bvector[k] = bvector[k] – p1.value*bvector[p1.col]
```

to

$$\begin{bmatrix} \text{bvector[k].r} \\ \text{bvector[k].i} \end{bmatrix} = \begin{bmatrix} \text{bvector[k].r} \\ \text{bvector[k].i} \end{bmatrix} - \begin{bmatrix} \text{p1.ul} & \text{p1.ur} \\ \text{p1.ll} & \text{p1.lr} \end{bmatrix} \times \begin{bmatrix} \text{bvector[p1.col].r} \\ \text{bvector[p1.col].i} \end{bmatrix}$$

2 by 2 Block Matrix Example

- The second numeric calculation changes from
  bvector[k] := bvector[k]/rowDiag[k].value
- To

$$\begin{bmatrix} \text{bvector[k].r} \\ \text{bvector[k].i} \end{bmatrix} = \begin{bmatrix} \text{bvector[k].r} \\ \text{bvector[k].i} \end{bmatrix} - \begin{bmatrix} \text{rowDiag[k].ul} & \text{rowDiag[k].ur} \\ \text{rowDiag[k].ll} & \text{rowDiag[k].lr} \end{bmatrix}^{-1} \times \begin{bmatrix} \text{bvector[p1.col].r} \\ \text{bvector[p1.col].i} \end{bmatrix}$$

- Which can be coded by directly doing the inverse as

$$\begin{bmatrix} \text{bvector[k].r} \\ \text{bvector[k].i} \end{bmatrix} = \begin{bmatrix} \text{bvector[k].r} \\ \text{bvector[k].i} \end{bmatrix} - \frac{1}{\text{det}} \begin{bmatrix} \text{rowDiag[k].lr} & \text{-rowDiag[k].ur} \\ \text{-rowDiag[k].ll} & \text{rowDiag[k].ul} \end{bmatrix} \times \begin{bmatrix} \text{bvector[p1.col].r} \\ \text{bvector[p1.col].i} \end{bmatrix}$$

with

$$\text{det} = \text{rowDiag[k].ul} \times \text{rowDiag[k].lr} - \text{rowDiag[k].ll} \times \text{rowDiag[k].ur}$$

Sparse Matrix and Vector Method Summary

- Previous slides have presented sparse matrix and sparse vector methods commonly used in power system and some circuit analysis applications
- These methods are widely used, and have the ability to substantially speed up power system computations
- They will be applied as necessary throughout the remainder of the course
- We'll now move on to sensitivity analysis with a quick introduction of contingency analysis

Contingency Analysis

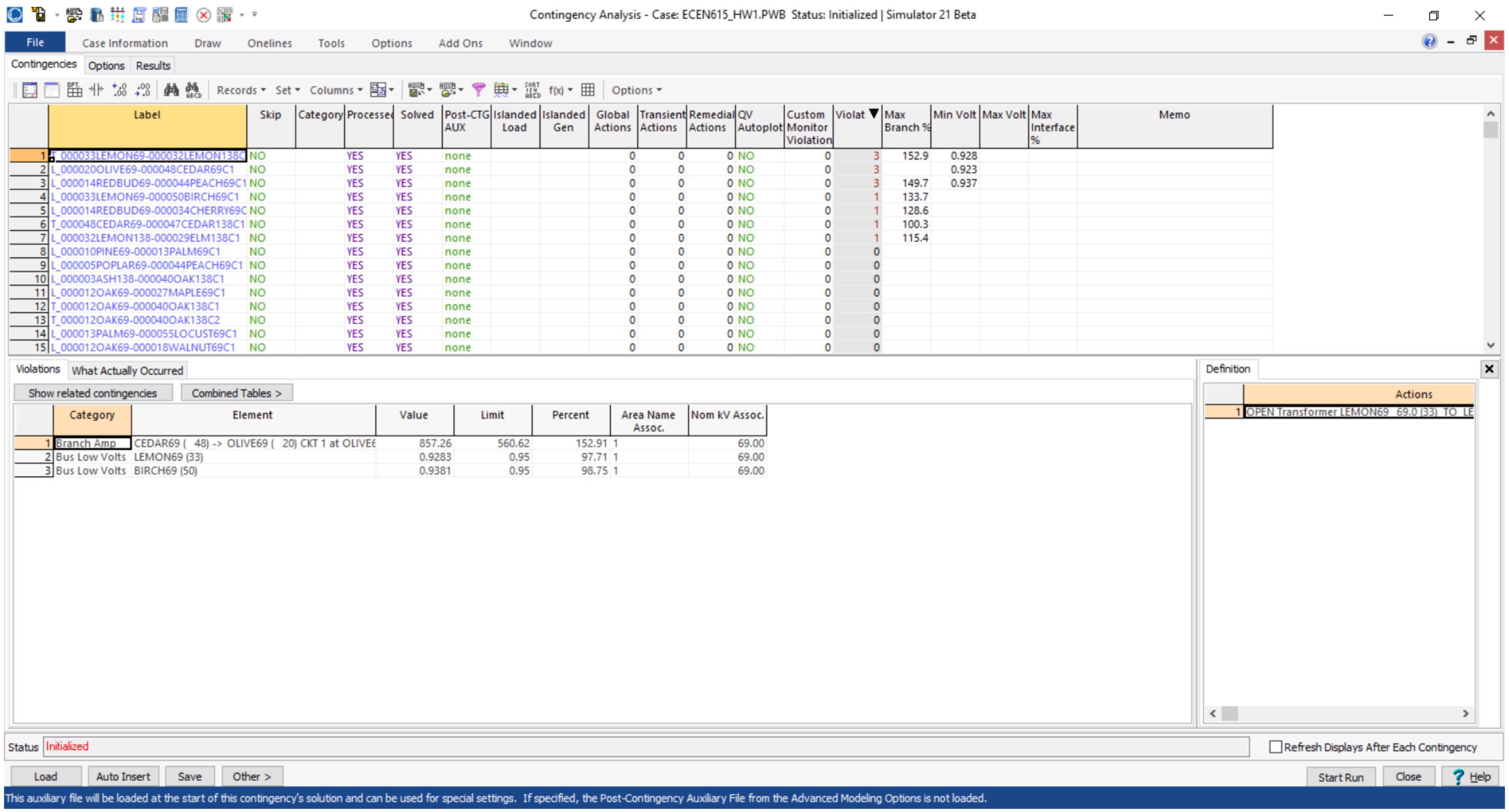
- Contingency analysis is the process of checking the impact of statistically likely contingencies
  - Example contingencies include the loss of a generator, the loss of a transmission line or the loss of all transmission lines in a common corridor
  - Statistically likely contingencies can be quite involved, and might include automatic or operator actions, such as switching load
- Reliable power system operation requires that the system be able to operate with no unacceptable violations even when these contingencies occur
  - N-1 reliable operation considers the loss of any single element

Contingency Analysis

- Of course this process can be automated with the usual approach of first defining a contingency set, and then sequentially applying the contingencies and checking for violations
  - This process can naturally be done in parallel
  - Contingency sets can get quite large, especially if one considers N-2 (outages of two elements) or N-1-1 (initial outage, followed by adjustment, then second outage
- The assumption is usually most contingencies will not cause problems, so screening methods can be used to quickly eliminate many contingencies
  - We'll cover these later

Contingency Analysis in PowerWorld

- Automated using the Contingency Analysis tool



Contingency Analysis - Case: ECEN615_HW1.PWB Status: Initialized | Simulator 21 Beta

	Label	Skip	Category	Processed	Solved	Post-CTG AUX	Islanded Load	Islanded Gen	Global Actions	Transient Actions	Remedial Actions	QV Autoplot	Custom Monitor Violation	Violat	Max Branch %	Min Volt	Max Volt	Max Interface %	Memo
1	T_000033LEMON69-000032LEMON138C	NO		YES	YES	none			0	0	0	NO	0	3	152.9	0.928			
2	L_000020OLIVE69-000048CEDAR69C1	NO		YES	YES	none			0	0	0	NO	0	3		0.923			
3	L_000014REDBUD69-000044PEACH69C1	NO		YES	YES	none			0	0	0	NO	0	3	149.7	0.937			
4	L_000033LEMON69-000050BIRCH69C1	NO		YES	YES	none			0	0	0	NO	0	1	133.7				
5	L_000014REDBUD69-000034CHERRY69C	NO		YES	YES	none			0	0	0	NO	0	1	128.6				
6	T_000048CEDAR69-000047CEDAR138C1	NO		YES	YES	none			0	0	0	NO	0	1	100.3				
7	L_000032LEMON138-000029ELM138C1	NO		YES	YES	none			0	0	0	NO	0	1	115.4				
8	L_000010PINE69-000013PALM69C1	NO		YES	YES	none			0	0	0	NO	0	0					
9	L_000005POPLAR69-000044PEACH69C1	NO		YES	YES	none			0	0	0	NO	0	0					
10	L_000003ASH138-000040OAK138C1	NO		YES	YES	none			0	0	0	NO	0	0					
11	L_000012OAK69-000027MAPLE69C1	NO		YES	YES	none			0	0	0	NO	0	0					
12	T_000012OAK69-000040OAK138C1	NO		YES	YES	none			0	0	0	NO	0	0					
13	T_000012OAK69-000040OAK138C2	NO		YES	YES	none			0	0	0	NO	0	0					
14	L_000013PALM69-000055LOCUST69C1	NO		YES	YES	none			0	0	0	NO	0	0					
15	L_000012OAK69-000018WALNUT69C1	NO		YES	YES	none			0	0	0	NO	0	0					

Violations | What Actually Occurred

	Category	Element	Value	Limit	Percent	Area Name Assoc.	Nom kV Assoc.
1	Branch Amp	CEDAR69 (48) -> OLIVE69 (20) CKT 1 at OLIVE6	857.26	560.62	152.91	1	69.00
2	Bus Low Volts	LEMON69 (33)	0.9283	0.95	97.71	1	69.00
3	Bus Low Volts	BIRCH69 (50)	0.9381	0.95	98.75	1	69.00

Definition

	Actions
1	OPEN Transformer LEMON69 69.0 (33) TO LE

Status: Initialized

Refresh Displays After Each Contingency

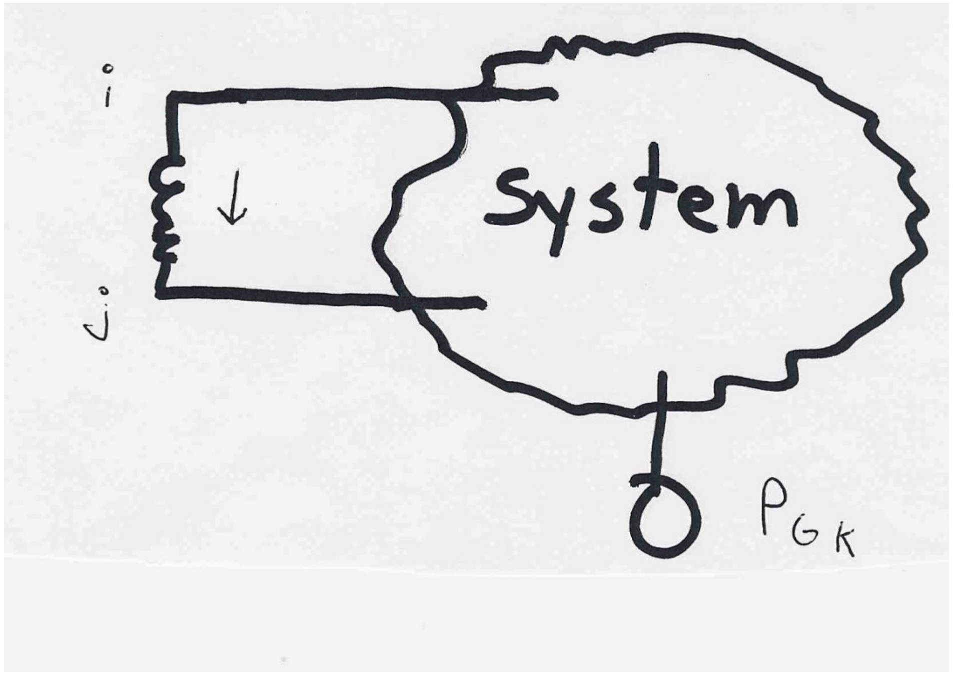
This auxiliary file will be loaded at the start of this contingency's solution and can be used for special settings. If specified, the Post-Contingency Auxiliary File from the Advanced Modeling Options is not loaded.

Power System Control and Sensitivities

- A major issue with power system operation is the limited capacity of the transmission system
  - lines/transformers have limits (usually thermal)
  - no direct way of controlling flow down a transmission line (e.g., there are no valves to close to limit flow)
  - open transmission system access associated with industry restructuring is stressing the system in new ways
- We need to indirectly control transmission line flow by changing the generator outputs
- Similar control issues with voltage

Indirect Transmission Line Control

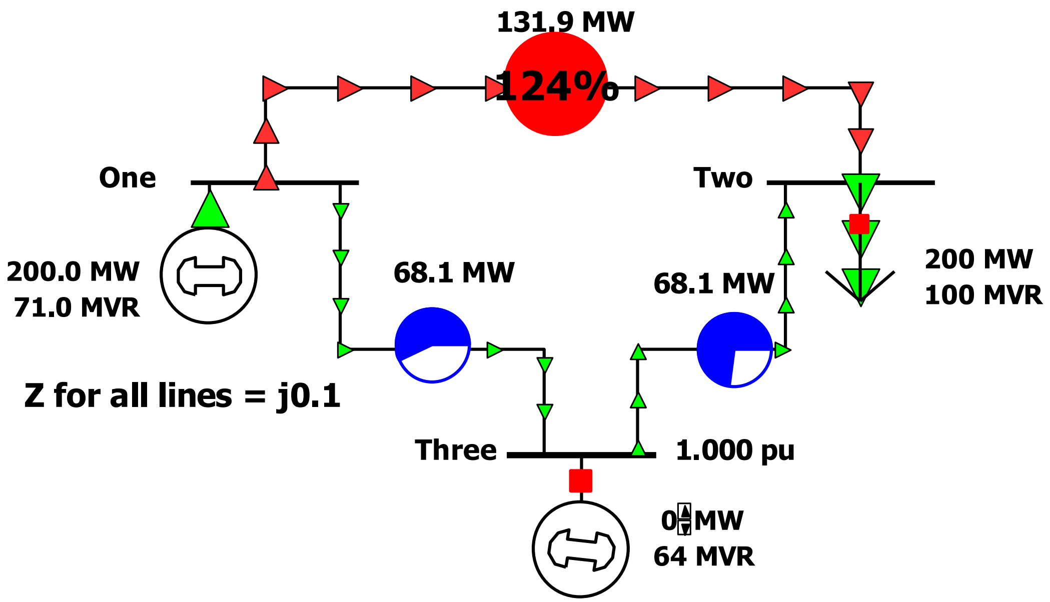
- What we would like to determine is how a change in generation at bus k affects the power flow on a line from bus i to bus j.



The assumption is that the change in generation is absorbed by the slack bus

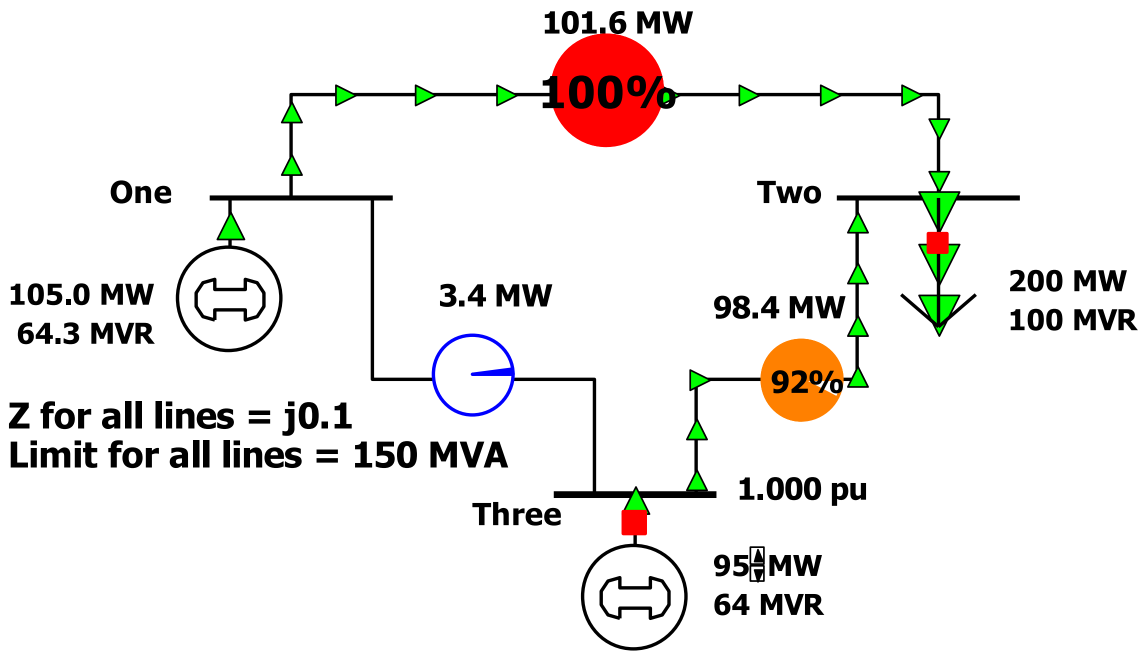
Power Flow Simulation - Before

- One way to determine the impact of a generator change is to compare a before/after power flow.
- For example below is a three bus case with an overload



Power Flow Simulation - After

- Increasing the generation at bus 3 by 95 MW (and hence decreasing it at bus 1 by a corresponding amount), results in a 30.3 MW drop in the MW flow on the line from bus 1 to 2, and a 64.7 MW drop on the flow from 1 to 3.



Expressed as a percent, 30.3/95 =32% and 64.7/95=68%

Analytic Calculation of Sensitivities

- Calculating control sensitivities by repeat power flow solutions is tedious and would require many power flow solutions. An alternative approach is to analytically calculate these values

The power flow from bus i to bus j is

$$P_{ij} \approx \frac{|V_i||V_j|}{X_{ij}} \sin(\theta_i - \theta_j) \approx \frac{\theta_i - \theta_j}{X_{ij}}$$

So $\Delta P_{ij} \approx \dfrac{\Delta\theta_i - \Delta\theta_j}{X_{ij}}$ We just need to get $\dfrac{\Delta\theta_{ij}}{\Delta P_{Gk}}$

Analytic Sensitivities

From the fast decoupled power flow we know

$$\Delta\boldsymbol{\theta} = \mathbf{B}^{-1}\Delta\mathbf{P}(\mathbf{x})$$

So to get the change in $\Delta\boldsymbol{\theta}$ due to a change of generation at bus k, just set $\Delta\mathbf{P}(\mathbf{x})$ equal to all zeros except a minus one at position k.

$$\Delta\mathbf{P} = \begin{bmatrix} 0 \\ \vdots \\ -1 \\ 0 \\ \vdots \end{bmatrix} \begin{matrix} \\ \\ \leftarrow \text{ Bus k} \\ \\ \\ \end{matrix}$$

Three Bus Sensitivity Example

For a three bus, three line case with $Z_{\text{line}} = j0.1$

$$\mathbf{Y}_{\text{bus}} = j\begin{bmatrix} -20 & 10 & 10 \\ 10 & -20 & 10 \\ 10 & 10 & -20 \end{bmatrix} \rightarrow \mathbf{B} = \begin{bmatrix} -20 & 10 \\ 10 & -20 \end{bmatrix}$$

Hence for a change of generation at bus 3

$$\begin{bmatrix} \Delta\theta_2 \\ \Delta\theta_3 \end{bmatrix} = \begin{bmatrix} -20 & 10 \\ 10 & -20 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 0.0333 \\ 0.0667 \end{bmatrix}$$

Then $\Delta P_{3\text{ to }1} = \dfrac{0.0667 - 0}{0.1} = 0.667 \text{ pu}$

$\Delta P_{3\text{ to }2} = 0.333 \text{ pu} \qquad \Delta P_{2\text{ to }1} = 0.333 \text{ pu}$

More General Sensitivity Analysis: Notation

- We consider a system with *n* buses and L lines given by the set given by the set $L \;@\{\ell \;\; \ell \;\; \cdots \;\; \ell$
  - Some authors designate the slack as bus zero; an alternative approach, that is easier to implement in cases with multiple islands and hence slacks, is to allow any bus to be the slack, and just set its associated equations to trivial equations just stating that the slack bus voltage is constant
- We may denote the kth transmission line or transformer in the system, $\boxed{?}_k$, as

$$\ell_{..} \; @ \quad {}_{..} \quad {}_{k}),$$

from node

to node

Notation, cont.

- We’ll denote the real power flowing on ℓ_k from bus i to bus j as f_k
- The vector of real power flows on the L lines is:

$$\mathbf{f} \; @ \, [f_{\ell} \quad f_{\ell} \quad \cdots$$

which we simplify to $\mathbf{f} = [f_1, f_2, \cdots$

- The bus real and reactive power injection vectors are

$$\mathbf{p} \; @ \, [p^1, p^2, \cdots \quad ^T$$

$$\mathbf{q} \; @ \, [q^1, q^2, \cdots \quad ^T$$

Notation, cont.

- The series admittance of line ⍰ is $g_{⍰} + jb_{⍰}$ and we define $$\tilde{L} \; @ \; \ldots g\{b_1, b_2, \cdots \}$$
- We define the L×N incidence matrix

$$\mathbf{A} \; @ \begin{bmatrix} \mathbf{a}_1^T \\ \mathbf{a}_2^T \\ \vdots \\ \mathbf{a}_L^T \end{bmatrix}$$

The component j of $\mathbf{a}_i$ is nonzero whenever line $⍰_i$ is coincident with node j. Hence **A** is quite sparse, with two nonzeros per row