Campus de São Carlos

"Observabilidade em redes de energia elétrica: métodos de identificação de medidas críticas"

Autor:

João Bosco Augusto London Junior

Orientador:

Prof. Newton Geraldo Bretas

UNIVERSIDADE DE SÃO PAULO

**ESCOLA DE ENGENHARIA
DE SÃO CARLOS**

✓

OBSERVABILIDADE EM REDES DE ENERGIA ELÉTRICA : MÉTODOS DE IDENTIFICAÇÃO DE MEDIDAS CRÍTICAS

DEDALUS - Acervo - EESC

31100035712

JOÃO BOSCO AUGUSTO LONDON JUNIOR

Dissertação apresentada à Escola de Engenharia de São Carlos, da Universidade de São Paulo, como parte dos requisitos para obtenção do Título de mestre em Engenharia Elétrica.

ORIENTADOR: PROF. NEWTON G. BRETAS

São Carlos
1997

Class.	TESE-EESC
Cutt.	3305
Tombo	T071/97

Área: Engenharia Elétrica

1x 0747598

Ficha catalográfica preparada pela Seção de Tratamento
da Informação do Serviço de Biblioteca - EESC-USP

L847o London Junior, João Bosco Augusto
Observabilidade em redes de energia elétrica :
métodos de identificação de medidas críticas /
João Bosco Augusto London Junior. -- São Carlos,
1997.

Dissertação (Mestrado). -- Escola de Engenharia
de São Carlos-Universidade de São Paulo, 1997.
Área: Engenharia Elétrica
Orientador: Prof. Dr. Newton G. Bretas

1. Observabilidade. 2. Medidas críticas.
I. Título

FOLHA DE APROVAÇÃO

Dissertação defendida e aprovada em 20-3-1997
pela Comissão Julgadora:

Prof. Titular **NEWTON GERALDO BRETAS (Orientador)**
(Escola de Engenharia de São Carlos - Universidade de São Paulo)

Prof. Doutor **MARINHO GOMES DE ANDRADE FILHO**
(Instituto de Ciências Matemáticas de São Carlos - Universidade de São Paulo)

Prof. Doutor **CARLOS ALBERTO FAVARIN MURARI**
(Universidade de Campinas - UNICAMP)

Prof. Doutor. **GERALDO ROBERTO M DA COSTA**
Coordenador da Área de Engenharia Elétrica

Prof. Assoc. **JOSÉ CARLOS A. CINTRA**
Presidente da Comissão de Pós-Graduação

Este trabalho é dedicado com muito carinho aos meus pais, João Bosco e Daise, que com amor, sabedoria e vigor têm me inspirado a ser o melhor que posso ser.

AGRADECIMENTOS

Ao professor Newton Geraldo Bretas pela excelente orientação fornecida durante a elaboração deste trabalho.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq, pela concessão da bolsa de estudo.

A todos os colegas, professores e funcionários do departamento de Engenharia Elétrica da EESC /USP, pela amizade.

A meus pais, que tornaram possível a realização deste trabalho, sempre acreditando em mim.

A minha namorada Meire Carrijo, pelo apoio, carinho e compreensão.

Ao amigo Mori, pela grande ajuda.

Sumário

Lista de Figuras.....	I
Lista de Tabelas.....	II
Resumo.....	III
Abstract.....	V
1 - Introdução.....	1
2 - Observabilidade de redes de energia elétrica.....	5
2.1 - Histórico.....	5
2.2 - Metodologia combinatorial.....	8
2.2.1 - Algoritmo.....	10
2.3 - Análise de observabilidade baseada na redução simbólica da matriz jacobiana.....	11
2.3.1 - Descrição do método.....	11
2.3.2 - Algoritmo.....	13
2.4 - Metodologia baseada na fatoração triangular da matriz ganho.....	15
2.4.1 - Algoritmo.....	16
2.5 - Metodologia baseada na fatoração triangular da matriz ganho e conceitos contidos em caminhos de grafo.....	17
2.5.1 - Algumas propriedades dos vetores esparsos e de caminhos de fatoração.....	17
2.5.2 - Observabilidade de redes com caminhos de fatoração.....	22
2.5.3 - Medidas irrelevantes e ramos não observáveis.....	22

2.5.4 - Adição de medidas usando caminhos de fatoração.....	24
2.5.5 - Algoritmo.....	25
2.5.6 - Esquema para Identificação de subredes Observáveis.....	25
3 - Identificação de medidas críticas.....	27
3.1 - Método combinatorial para identificação de medidas críticas...	27
3.1.1 - Algoritmo.....	28
3.2 - Identificação de medidas críticas, via matrizes unimodulares M e A.....	31
4 - Métodos Propostos.....	37
4.1 - Identificação de medidas críticas através de caminhos de grafo.....	38
4.1.1 - Descrição do método.....	38
4.1.2 - Algoritmo.....	40
4.1.3 - Exemplo.....	42
4.2 - Identificação de medidas críticas através da matriz jacobiana.....	46
4.2.1 - Metodologia.....	46
4.2.2 - Descrição do método.....	49
4.2.3 - Algoritmo.....	50
4.2.4 - Exemplo.....	51
5 - Testes realizados e análise dos resultados.....	54
5.1 - Testes realizados.....	55
5.1.1 - Testes com o sistema de 6 barras da IEEE.....	55
5.1.2 - Testes com o sistema de 14 barras da IEEE.....	58
5.1.3 - Testes com o sistema de 121 barras da ELETROSUL.....	61
5.2 - Análise dos resultados.....	63

6 - Conclusão.....	64
Anexo A.....	67
Anexo B.....	70
Anexo C.....	98
Referências Bibliográficas.....	111
Apêndice.....	115

Lista de Figuras

Figura 2.1 - Matriz A.....	18
Figura 2.2 - Matriz G tornada triangular.....	19
Figura 2.3 - Matriz G com um pivô zero.....	20
Figura 2.4 - Sistema de equações representando as três subredes	23
Figura 3.1 - Representação das novas conceituações (TST, MST, FRST).....	36
Figura 4.1 - Topologia da rede de 6 barras da IEEE, associada a um sistema de 6 medidas.....	42
Figura 4.2 - Exemplo de uma rede de 4 barras.....	47
Figura 5.1 - Topologia da rede de 6 barras da IEEE.....	53
Figura 5.2 - Topologia da rede de 14 barras da IEEE.....	58
Figura 7.1 - Arquivo de entrada de dados.....	68
Figura 7.2 - Arquivo de saída de dados.....	69
Figura 7.3 - Fluxograma correspondente à metodologia que permite identificar as medidas críticas, através dos caminhos de grafo.....	71
Figura 7.4 - Fluxograma correspondente à metodologia que permite identificar as medidas críticas, através da estrutura da matriz jacobiana.....	99

Lista de tabelas

Tabela 4.1 - Conjuntos de 2 variáveis de cada medida.....	48
Tabela 5.1 - Resultados do teste com a rede de 6 barras (situação 1).....	56
Tabela 5.2 - Resultados do teste com a rede de 6 barras (situação 2).....	56
Tabela 5.3 - Resultados do teste com a rede de 6 barras (situação 3).....	56
Tabela 5.4 - Resultados do teste com a rede de 6 barras (situação 4).....	57
Tabela 5.5 - Resultados do teste com a rede de 6 barras (situação 5).....	57
Tabela 5.6 - Resultados do teste com a rede de 6 barras (situação 6).....	58
Tabela 5.7 - Resultados do teste com a rede de 14 barras (situação 1).....	59
Tabela 5.8 - Resultados do teste com a rede de 14 barras (situação 2).....	59
Tabela 5.9 - Resultados do teste com a rede de 14 barras (situação 3).....	60
Tabela 5.10 - Resultados do teste com a rede de 14 barras (situação 4)....	60
Tabela 5.11 - Conjunto de medidas para o teste com a rede de 121 barras.....	61
Tabela 5.12 - Resultados do teste com a rede de 121 barras.....	62

Resumo

Uma das mais importantes responsabilidades dos Centros de Controle e Operação das Companhias de Energia Elétrica é manter a confiabilidade de serviço em um sistema de potência. E, para se ter uma operação confiável dos sistemas de potência, é de fundamental importância identificar as medidas críticas, para depois fortalecer os sistemas de medidas, através de pseudo-medidas, tornando-as medidas redundantes, ou seja, medidas que não afetam diretamente a observabilidade dos sistemas.

O nosso objetivo é criar métodos mais eficientes para a identificação das medidas críticas. Fazemos referência, de maneira sucinta, a alguns métodos existentes para análise de observabilidade, bem assim a dois métodos antigos para a identificação de medidas críticas. O primeiro método é de natureza combinatorial; o segundo apoia-se nas matrizes unimodulares M (matriz incidência de medidas para ramos) e A (matriz incidência de ramos para nós), necessitando, entretanto, de um algoritmo combinatorial para a sua aplicabilidade e tornando-o, conseqüentemente, por assim dizermos, um método lento. Estamos apresentando dois métodos para a identificação de medidas críticas: o primeiro é baseado na teoria desenvolvida por BRETAS (1996a), para análise de observabilidade, com a utilização dos caminhos de grafo; o outro método fundamenta-se no

princípio desenvolvido por SLUTSKER & SCUDDER (1987), permitindo a identificação através da análise da estrutura da matriz jacobiana.

Foram ambos adaptados a programas desenvolvidos em turbo c; os resultados dos testes, com sistemas de 6, 14 e 121 barras, mostram a eficiência dos mesmos.

Abstract

One of the most important functions of the control and operation centers is to keep reliability of service in a electrical power system. In order to obtain a reliable operation of the power system, it is important to identify the critical measurements, and then to improve the measurement system using pseudo measurements. The goal of this work is to determine more efficient methods to critical measurement identification. A brief review of the some methods for observability analysis as well as two methodologies to identify critical measurements are presented. The first method has a combinatorial nature; the second one is supported by unimodular matrices M (incidence matrix of measurements for branches) and A (incidence matrix of branch for nodes). The second method needs a combinatorial algorithm to be feasible, so that it becomes a slow method.

Two new methods to critical measurement identification are presented in this work: the first one is based on the theory developed by BRETAS (1996a), to analyse observability using graph paths; the second method is supported by the SLUTSKER & SCUDDER (1987) theory, where identification is reached throught the analysis of the jacobian matrix.

Both methods were implemented in turbo c; test results for the systems with 6, 14 and 121 buses present the efficient of the created methods.

1- Introdução

Nos Centros de Controle e Operação das Companhias de Energia Elétrica, a estimação de estado constitui-se hoje numa ferramenta fundamental para se ter uma operação adequada e segura dos mesmos. Estimação de estado é o processo para determinar o estado de um sistema elétrico, ou seja, determinar as magnitudes de tensão e os ângulos de fase nas barras, através de um conjunto de medidas, constituído usualmente de: fluxo de potência ativa e reativa nas linhas, injeção de potência ativa e reativa e algumas magnitudes de tensão nos barramentos. Uma questão fundamental, que se tem em vista, é saber se é possível, através dessas medidas, determinar as magnitudes de tensão e os ângulos em todas as barras do sistema de potência. Em caso afirmativo, tal sistema é dito observável. Logo, o estimador de estado possibilitar-nos-á conhecer, com um certo grau de confiança, os estados do sistema de potência. Através dessas informações, torna-se possível a realização do controle e operação adequada dos sistemas elétricos de potência. Por conseguinte, a estimação de estado só pode ser realizada se o sistema de potência for observável. Contudo, o conjunto de medidas que torna um sistema observável pode conter algumas que são vitais para a observabilidade do mesmo, porquanto

a perda dessas medidas faria o sistema tornar-se não observável. São as chamadas Medidas Críticas. O objetivo deste trabalho é desenvolver métodos para a identificação de medidas críticas, de maneira rápida e simples, o que permitirá o fortalecimento do sistema de medidas, através de pseudo-medidas, tornando aquelas medidas redundantes, isto é, medidas que não afetam diretamente a observabilidade do sistema.

Como a identificação de medidas críticas está diretamente relacionada ao princípio de observabilidade de sistemas elétricos, julgamos oportuno apresentar, neste trabalho, um pequeno histórico de pesquisas desenvolvidas nessa área. Pareceu-nos de bom alvitre apresentar, também, de maneira sucinta, alguns métodos de análise de observabilidade, o que facilitará o desenvolvimento de métodos e algoritmos para a identificação de medidas críticas.

Em seguida, são apresentados dois métodos para a identificação de medidas críticas; o primeiro é um método puramente combinatorial, não exigindo nenhum tipo de cálculo, dependendo apenas da topologia da rede e do tipo e localização das medidas. O segundo método destina-se à identificação das medidas críticas, através das matrizes unimodulares M (matriz incidência de medidas para ramos) e A (matriz incidência de ramos para nós).

Neste trabalho, foram desenvolvidos dois métodos para a identificação de medidas críticas. O primeiro utiliza como princípio a teoria desenvolvida por BRETAS (1996a), para a análise de observabilidade utilizando caminhos de grafo. Estes caminhos de grafo são obtidos da

matriz ganho tornada triangular. O outro método proposto neste trabalho corresponde a um tempo menor de execução, não necessitando de nenhum cálculo (nem mesmo do cálculo da matriz ganho), permitindo-nos assim identificar as medidas críticas analisando diretamente a estrutura da matriz jacobiana de medidas, com base na teoria desenvolvida por SLUTSKER & SCUDDER (1987), para a análise de observabilidade.

Os dois métodos foram implantados através de programas desenvolvidos em turbo c (Anexos A, B e C) e testados nos sistemas de 6 e 14 barras da IEEE e no sistema real de 121 barras da ELETROSUL.

No capítulo II deste trabalho, é apresentado um pequeno histórico das pesquisas desenvolvidas na área de observabilidade de redes, e, detalhadamente, as seguintes metodologias para sua análise:

- Metodologia combinatorial;
- Metodologia baseada na redução simbólica da matriz ganho;
- Metodologia baseada na fatoração triangular da matriz ganho;
- Metodologia baseada na fatoração triangular da matriz ganho e nos conceitos contidos em caminhos de grafo;

No capítulo III, dois métodos para a identificação de medidas críticas são apresentados: O primeiro é um método combinatorial; o segundo trata da identificação das medidas críticas, através das matrizes unimodulares M e A .

No capítulo IV foram desenvolvidas as duas metodologias para a identificação de medidas críticas, propostas neste trabalho. A primeira possibilita identificar as medidas críticas através dos caminhos de grafo,

obtidos da matriz ganho tornada triangular; a segunda o faz analisando a estrutura da matriz jacobiana.

Os testes realizados, juntamente com a análise dos resultados, são apresentados no capítulo V, estando a conclusão deste trabalho no capítulo VI.

2 - Observabilidade de redes de energia elétrica

A estimação de estado é hoje, como já enfatizado, uma ferramenta fundamental para se ter uma operação adequada e segura dos sistemas de potência. Entretanto, foi em 1970 que SCHWEPPE & WILDES (1970) propuseram, originalmente, o uso das técnicas de estimação de estado para sistemas de potência. À questão da observabilidade de rede, propriamente dita, foi dada mais atenção a partir de 1973.

Com o objetivo de mostrar como foi o desenvolvimento das pesquisas na área em pauta, é apresentado, a seguir, um pequeno histórico salientando alguns trabalhos elaborados na área.

2.1 - Histórico

FETZER & ANDERSON (1975) desenvolveram uma metodologia para análise de observabilidade, tendo em vista o seu conceito contido na teoria de controle linear. A grande dificuldade desse método, para a sua aplicabilidade prática, estava no requerimento computacional, o que inviabilizava o seu estudo on-line. Visando à obtenção de métodos que

pudessem ser usados em estudos on-line, as pesquisas foram em busca de procedimentos mais lógicos.

Seguindo este caminho, CLEMENTS & WOLLENBERG (1975) desenvolveram um algoritmo para a análise de observabilidade, usando a topologia da rede e as leis de Kirchhoff, introduzindo ainda o conceito de ilhas observáveis. Tal algoritmo era um procedimento heurístico que considerava, primeiramente, as medidas de fluxo, depois as medidas de injeção. ALLMONG et alii (1980) observaram que o algoritmo desenvolvido por CLEMENTS & WOLLENBERG (1975) poderia propiciar resultados conservativos, em classificando, possivelmente, como não observável um sistema observável, razão pela qual apresentaram uma versão modificada. Posteriormente, HORTON & MASIELLO (1977) estenderam o método de Clements & Wollenberg para o estimador de estado desacoplado. HANDSCHIN & BONGERS (1972), com base na condição necessária para observabilidade, apresentaram um método que permite testar a conectividade da matriz jacobiana. KRUMPHOLZ et alii (1980), através do conceito de redes topologicamente observáveis, desenvolveram um algoritmo para a análise de observabilidade, utilizando conceitos da teoria de grafos. Os conceitos de observabilidade algébrica, numérica e topológica foram introduzidos nesse trabalho, mas os autores não apresentaram testes com sistemas reais.

QUINTANA et alii (1982) apresentaram um método também baseado no conceito de rede topologicamente observável, usando um algoritmo "matroid intersection", com testes em sistemas reais.

Buscando menos complexidade computacional, em 1985 MONTICELLI & WU (1985a) desenvolveram um método novo, possibilitando testar a observabilidade da rede e identificar ilhas observáveis, quando o sistema não é observável como um todo, fazendo-o através da fatoração triangular da matriz ganho. Embora esse método fosse um simples implemento, com a utilização apenas de subrotinas já disponíveis nos programas de estimador de estado, o mesmo é muito sensível a valores numéricos.

SLUTSKER & SCUDDER (1987) idealizaram um método baseado na simbólica redução da matriz jacobiana. O algoritmo é extremamente simples e corresponde a um tempo diminuto de execução dos trabalhos.

NUCERA & GILLES (1991) desenvolveram um algoritmo tendo em vista o aperfeiçoamento combinatorial, o qual se nos apresenta como um algoritmo topológico muito rápido, por assim dizermos. No mesmo ano, MORI & TSUZUKI (1991) apresentaram outro método baseado na mínima árvore representativa e na teoria de grafos.

Em 1995, BRETAS (1996a) julgou por bem desenvolver um método novo, com a finalidade de testar a observabilidade da rede e identificar ilhas observáveis, quando o sistema não é observável como um todo, assim o fazendo através da triangulação da matriz ganho e dos conceitos contidos nos caminhos de grafo. Tal método necessita apenas de subrotinas já disponíveis nos programas de estimadores de estado e não requer solução alguma de sistemas de equações algébricas, sendo assim de fácil adaptação.

Dentre as metodologias citadas, importa serem ora apresentadas, de maneira sucinta, apenas quatro:

- 1- Metodologia combinatorial;
- 2- Metodologia baseada na redução simbólica da matriz jacobiana;
- 3- Metodologia baseada na fatoração triangular da matriz ganho;
- 4- Metodologia baseada na fatoração triangular da matriz ganho e nos conceitos contidos em caminhos de grafo;

Estas metodologias são apresentadas em razão de que:

- Da metodologia 1; metodologia combinatorial; originou-se um método para identificação de medidas críticas; daí a sua importância.
- As metodologias 2 e 4 serviram de base para o desenvolvimento dos métodos propostos neste trabalho.
- A metodologia 3, por fundamentar-se na fatoração triangular da matriz ganho, em que também se alicerça a metodologia 4, que é, como se disse, uma das bases em que está apoiado o presente trabalho.

2.2 - Metodologia combinatorial

A questão da observabilidade de redes está relacionada ao posto da matriz Jacobiana H . Entretanto, o cálculo do posto consome muito tempo, dependendo da matriz em consideração, e, além disso, após o seu cálculo, se o mesmo não for completo, nenhuma informação sobre observabilidade

seria obtida. Estas informações relacionam-se com a topologia da rede, motivando assim um algoritmo com bases topológicas.

KRUMPHOLZ et alii (1980) apresentaram um algoritmo para a análise de observabilidade, que é puramente combinatorial, usando apenas a topologia da rede. Um sistema de potência é topologicamente observável, com respeito a um conjunto de medidas, se, e somente se, existir associada a tal sistema uma árvore representativa de posto completo.

Árvore representativa é uma árvore abrangendo todas as barras da rede; uma árvore é de posto completo, se for possível atribuir-se uma medida distinta para cada ramo desta árvore. Assim, a questão de observabilidade requer a existência de uma árvore representativa de posto completo. Portanto, um procedimento que construa tal árvore, quando ela existir, é necessário.

Na tentativa de construir uma árvore representativa de posto completo, inicia-se com a coleção F de medidas de fluxo nos ramos. A porção da rede constituída pelos ramos que contém essas medidas de fluxos, juntamente com os nós conectados por elas, recebe o nome de árvore. O conjunto de todas as árvores desconectadas recebe o nome de floresta. Para demonstrar a existência de redes observáveis, é necessário aumentar F para uma árvore representativa de posto completo, adicionando-se-lhe novos ramos, e, para cada ramo adicionado, deve designar-se uma medida ainda não relacionada. O aumento de F é feito então na tentativa de construir-se uma árvore, contendo todas as barras da

rede sem medida de injeção, indicando assim que a rede é observável, e, quando não for possível construir tal árvore, a rede é não observável.

2.2.1 - Algoritmo

O algoritmo é composto de duas partes, sendo que na primeira se processam as medidas de fluxo em ramos e na segunda parte se processam as medidas de injeções, que são incidentes a ramos com medida de fluxo e àqueles sem medida de fluxo concomitantemente (são as chamadas injeções limitadas).

Na primeira parte se constrói uma floresta F de ramos com medidas de fluxo. Aqueles ramos que formam loop, com ramos da floresta F , são identificados e eliminados, porque não contribuem para o aumento da floresta. As medidas de injeção, incidentes somente a ramos com medidas de fluxo, são também eliminadas. Assim é formada a floresta F de ramos e todas as injeções limitadas são identificadas.

A segunda parte do algoritmo é uma seqüência de passos, cada um dos quais envolvendo o processamento de uma injeção limitada. Logo, esta segunda parte é iterativa e prossegue até construir-se uma árvore representativa de posto completo, ou até serem processadas todas as medidas de injeção limitada.

O algoritmo em questão, como visto, adiciona um ramo de cada vez para aumentar F ; conseqüentemente, o número de operações de máquina, necessário para a sua execução, está relacionado ao número de barras e

de ramos da rede. Logo, para redes de dimensões elevadas, esse tempo é muito alto, tornando assim inviável o algoritmo.

2.3 - Análise de observabilidade baseada na redução simbólica da matriz jacobiana.

A análise de observabilidade é uma busca de porções observáveis de um sistema de potência, podendo muitas vezes referir-se a todo o sistema (sistemas observáveis). Às porções observáveis do sistema de potência, dá-se o nome de ilhas observáveis. SLUTSKER & SCUDDER (1987) apresentaram um método para a análise de observabilidade, baseado na redução simbólica da matriz jacobiana transposta, sem a exigência de nenhum cálculo e sem natureza combinatorial. A seguir, uma descrição sucinta deste método é apresentada:

2.3.1 - Descrição do método

A técnica identifica grupos observáveis, através da redução simbólica da matriz jacobiana transposta H^t , relacionada às medidas. Assim, cada coluna da matriz H^t corresponde a uma equação de medida e as linhas correspondem às variáveis relacionadas por estas equações. A redução da matriz H^t é realizada, passo a passo, através da eliminação de grupos observáveis. As etapas desta redução são caracterizadas por:

- 1- Identificação de equações redundantes e eliminação das mesmas.

2- Identificação das ilhas observáveis e agrupamento destas com as ilhas observáveis já identificadas.

O método consiste de 3 passos principais. No primeiro passo, a matriz jacobiana simbólica é construída (simbólica, porque para esta técnica o que interessa é a posição dos elementos não nulos em H^t e não o valor real de cada elemento). No segundo passo, as equações de medidas de fluxo são eliminadas, eliminando-se as respectivas colunas de H^t ; eliminando-se ainda as linhas correspondentes às variáveis participantes das equações eliminadas, com exceção da variável de referência (variável de referência é aquela que mais vezes aparece no conjunto das medidas). Após o processamento do passo dois, a nova matriz H^t apresenta um conjunto de vetores linearmente independentes correspondentes as equações de medidas de injeções, sendo que a dimensão da matriz, como um todo, diminuirá bastante. Normalmente, o número de barras restantes fica em torno de 15 a 20 % do número original de barras. No terceiro passo, a redução de H^t continua, agora através da eliminação das medidas de injeção. Primeiramente são eliminadas as medidas de injeção que relacionam apenas duas variáveis e o processo de eliminação correspondente é exatamente igual ao processo anterior. Quando todas as injeções que relacionam somente duas variáveis já tiverem sido processadas, começa-se então a busca de um subsistema de equações solúveis (um subsistema solúvel é definido como um conjunto de K equações linearmente independentes, que relacionam $K+1$ variáveis), pois a toda ilha observável associa-se pelo menos um subsistema solúvel. A

eliminação de uma ilha observável é realizada seqüencialmente, pesquisando e eliminando os seus subsistemas solúveis. A eliminação de um subsistema solúvel é feita eliminando-se as linhas e colunas de H^t , correspondentes às variáveis e às equações de medidas de tal subsistema, respectivamente. No final do processamento, a matriz H^t constitui-se somente das barras cujas variáveis não se relacionam, uma vez que todos os subsistemas solúveis já foram identificados e eliminados. Logo, nenhuma das barras remanescentes pode ser agrupada às ilhas já identificadas. Assim, não é mais possível reduzir H^t . A dimensão dos grupos observáveis é caracterizada pela valência da barra de referência. Inicialmente, a valência de todas as barras de H^t é igual a 1. Em cada passo de redução, a valência da barra de referência de um grupo observável é somada à valência da barra que está sendo eliminada. As barras que são eliminadas ficam com valência zero. Terminado o processamento, as barras com valência diferente de zero identificam as ilhas observáveis. A seguir, é apresentado um algoritmo para o desenvolvimento do processo.

2.3.2 - Algoritmo

passo 1 - Construa a matriz H^t simbólica, para a rede a ser analisada.

passo 2 - Reduza H^t utilizando as equações de fluxo, identificando e eliminando as equações redundantes.

passo 3 - Reduza H^t utilizando equações de medidas de injeção com duas variáveis.

passo 4 - Escolha a barra com maior valência encontrada (primeiro critério), ou a que estiver presente em mais equações (segundo critério), para ser a barra de referência. Se todas as barras já foram processadas, siga para o passo 11.

passo 5 - Identifique as primeiras vizinhas da barra de referência (as primeiras vizinhas são as barras que possuem, no mínimo, uma equação em comum com a barra de referência).

passo 6 - Escolha a barra inicial para começar o processo de busca de subsistemas solúveis.

passo 7 - Realize a busca de um subsistema solúvel.

passo 8 - Se um subsistema solúvel não for encontrado, siga para o próximo passo. Caso contrário reduza H^t , eliminando este sistema. Incremente a valência da barra de referência. Agrupe aos grupos observáveis, já pesquisados, o novo grupo observável.

passo 9 - Se todas as primeiras vizinhas já tiverem sido analisadas, siga para o próximo passo; caso contrário volte ao passo 6.

passo 10 - Se todas as barras remanescentes já tiverem sido tratadas como barra de referência, siga para o próximo passo. Caso contrário, marque a barra de referência como processada e volte ao passo 4.

passo 11 - Marque as ilhas identificadas como observáveis, se tiverem pelo menos uma medida de tensão. Fim da análise.

SLUTSKER & SCUDDER (1987) concluem, através dos testes realizados, que esta técnica é extremamente rápida.

2.4 - Metodologia baseada na fatoração triangular da matriz ganho

MONTICELLI & WU (1985a) apresentaram uma teoria com base na Fatoração triangular da matriz ganho G , utilizando o modelo linear de estimador de estado. O algoritmo é caracterizado por usar um conceito extremamente simples, utilizando subrotinas já disponíveis nos estimadores de estado, bem como por ser de fácil implantação. O mesmo visa a testar a observabilidade da rede e encontrar ilhas observáveis, quando a rede não é observável como um todo.

Através da teoria em pauta, a análise da observabilidade é feita durante a fatoração triangular da matriz ganho G , que é dada pela seguinte equação:

$$G = H^t W H \quad (2.1)$$

onde:

H - é a matriz jacobiana, que é formada tomando as impedâncias de linha com valor unitário;

W - é a matriz de ponderação (definida positiva), atribuindo aos pesos valores unitários.

O sistema é observável, quando apenas o último elemento da diagonal principal da matriz G , tornada triangular, for igual a zero; quando isto acontece, é necessária apenas uma fatoração da matriz G . Mas, quando se encontrar mais de um pivô igual a zero, o sistema associado a essa matriz é não observável e requer a solução das equações do estimador de estado linear, toda vez que o sistema é novamente fatorado. Logo, o método é iterativo, quando a rede é não observável, requerendo

tantas soluções de estimador de estado linear quantas forem as novas fatorações.

Os passos do algoritmo proposto por MONTICELLI & WU (1985b) são mostrados a seguir.

2.4.1 - Algoritmo

passo 1-Inicie o conjunto de medidas de interesse, como constando de todas as medidas disponíveis.

passo 2-Atualize a rede de energia de interesse, removendo todos os ramos que não tenham medida de fluxo, nem mesmo medida de injeção nos seus nós terminais.

passo 3-Forme a matriz ganho G .

passo 4-Efetue a fatoração triangular de G , introduzindo pseudo-medidas, sempre que um pivô zero for encontrado e pare somente quando este pivô zero surgir necessariamente no fim. Caso contrário prossiga para o passo 5.

passo 5- Resolva as equações do estimador de $G \underline{\theta} = H^t W Z$, para $\underline{\theta}$, considerando todos os valores das medidas iguais a zero, exceto para as pseudo-medidas, que assumem os valores $\theta = 0, 1, 2$, e assim por diante.

passo 6- Estime o valor do fluxo dos ramos $P_{km} = x_{km}^{-1} (\underline{\theta}_k - \underline{\theta}_m)$, sendo θ_k e θ_m as componentes k -ésima e m -ésima de θ , para todos os ramos $k-m$, dentro da rede de potência obtida através do passo 2. Pare, se não houver fluxo de potência ($P_{km}=0$). Caso contrário, prossiga para o passo 7.

passo 7- Atualize a rede de potência de interesse, removendo todos os ramos $k-m$, em que $P_{k-m} \neq 0$; são os ramos não observáveis.

passo 8- Atualize o conjunto de medidas de interesse, removendo as medidas de injeção de potência, provenientes das barras adjacentes, a pelo menos um dos ramos removidos no passo 7.

passo 9-Retorne ao passo 2.

2.5-Metodologia baseada na fatoração triangular da matriz ganho e conceitos contidos em caminhos de grafo

BRETAS (1996a) desenvolveu uma teoria sobre observabilidade, fundamentando-a nos princípios da fatoração triangular da matriz ganho G , bem como nos conceitos contidos em caminhos de grafo. Essa teoria adota subrotinas já disponíveis nos centros de operação e não requer solução alguma de sistemas de equações algébricas. Para apresentá-la importa enunciar algumas propriedades dos vetores esparsos e de caminhos de fatoração.

2.5.1-Algumas propriedades dos vetores esparsos e de caminhos de fatoração

Seja o sistema linear dado pela equação:

$$A\underline{x} = \underline{b} \quad (2.2)$$

Sendo:

- A , uma matriz não singular;

- $\underline{b}$, o vetor independente;

$-\underline{x}$, o vetor de estado a calcular;

O que normalmente se deseja é a atualização do vetor de estado, quando ocorre alguma mudança no vetor independente, com o mínimo de operações possíveis. TINNEY et alii (1985) apresentaram uma solução deste problema, através dos vetores esparsos.

Teorema I

Suponha uma rede de energia, a que está associado um conjunto de medidas, tais que os n seguintes subconjuntos de equações possam ser escritos:

$$A_1 \underline{x}_1 = \underline{b}_1, \quad A_2 \underline{x}_2 = \underline{b}_2, \quad \dots, \quad A_n \underline{x}_n = \underline{b}_n$$

Com o vetor de estado: $\underline{x}_t = [\underline{x}_1, \underline{x}_2, \dots, \underline{x}_n]$; o vetor independente $\underline{b}_t = [\underline{b}_1, \underline{b}_2, \dots, \underline{b}_n]$ e a matriz A dada por (figura 2.1):

Figura 2.1 $\Rightarrow$ Matriz A

onde, cada submatriz A_i é uma matriz não singular.

Então a rede terá n caminhos de grafo, cada um associado a seu subconjunto de equações e definido de forma única. Cada caminho de grafo será desconectado um do outro, visto que os subgrupos de variáveis relacionados aos mesmos são subgrupos desacoplados.

Teorema II

O caminho de fatoração, associado ao vetor independente $\underline{b}$ de $G\underline{x}=\underline{b}$, resulta em um caminho de grafo conectado, quando somente o último elemento da diagonal principal da matriz G triangular for igual a zero.

A triangulação de G reduz esta matriz, quando nenhum ângulo de fase da rede é definido como referência à forma mostrada na figura a seguir (2.2):

Figura 2.2 $\Rightarrow$ Matriz G tornada triangular

A área sombreada corresponde a possíveis elementos não zero. A submatriz, sem a última linha e a última coluna, é não singular, existindo assim apenas um caminho de grafo associado a ela. É necessário que, pelo menos, um dos elementos da última coluna de G faça conexão com o grafo da submatriz anterior; caso contrário a variável correspondente àquela coluna seria uma variável isolada do sistema; como consequência, a coluna associada à mesma variável, mas em H , não será uma combinação linear das colunas anteriores e o caminho de grafo associado àquela matriz tem que ser um único caminho de grafo conectado.

Teorema III

No processo de triangulação da matriz ganho, se um pivô zero for encontrado, é sinal de que os nós remanescentes estarão contidos em outro caminho de grafo, não apresentando conexão alguma com os grafos anteriores.

Em MONTICELLI & WU (1985a), mostra-se que, na triangulação da matriz G , quando um pivô zero for encontrado, as correspondentes coluna e linha serão compostas de zeros, como mostrado a seguir (figura 2.3):

Figura 2.3 $\Rightarrow$ Matriz G com um pivô zero

Neste caso, apenas um caminho de grafo é associado à submatriz já tornada triangular (Teorema II), e como os elementos restantes desta linha são iguais a zero, não conectam este nó a nenhum nó subsequente; desta maneira o caminho de grafo será desconectado dos subseqüentes.

Deste teorema se depreende que a fatoração da matriz G irá resultar em tantos caminhos de grafos, quantos forem os pivôs zeros encontrados.

Teorema IV

Se, para o sistema de equações do teorema I, com a matriz A substituída pela matriz ganho G , uma equação de medida for adicionada, relacionando nós do mesmo caminho de grafo, a fatoração da matriz ganho

irá mudar, mas não alterará o número de caminhos de grafos, os quais continuarão desconectados um dos outros, como originalmente. Isto porque as medidas não provocarão novos caminhos entre aqueles existentes, já que não relacionam nenhuma de suas variáveis (isto se refere tanto à medida de injeção como à medida de fluxo).

Teorema V

Se, para o sistema de equações do teorema I, com a matriz A substituída pela matriz ganho G , uma nova medida for adicionada, relacionando diferentes caminhos de grafos, duas situações podem ocorrer:

a) A nova medida relaciona apenas duas variáveis (medida de fluxo de potência ou equivalente); então os dois caminhos de grafo, que possuem duas de suas variáveis relacionadas através dessa medida, serão unidos em um único caminho de grafo.

b) A nova medida relaciona mais de duas variáveis (medida de injeção), e:

b.1) As variáveis pertencem somente a dois grafos distintos. Neste caso, os dois caminhos de grafos originais se transformam em um único caminho de grafo.

b.2) As variáveis mencionadas pertencem a mais de dois caminhos distintos. Então, o número isolado de caminhos de grafo, que possuam variáveis relacionadas através dessa medida, será diminuído de um.

2.5.2-Observabilidade de redes com caminhos de fatoração

Consideremos a matriz ganho G , dada pela equação 2.1. Se, na fatoração triangular de G , existir somente um caminho de grafo, o sistema é observável, mas, se existir mais de um caminho de grafo, podem ocorrer duas situações:

1- Não havendo medidas de injeção relacionando nós de caminhos de grafo diferentes (mais de dois); então o sistema não é observável como um todo e cada subrede, associada àqueles caminhos de grafo, constitui-se em ilha observável da rede.

2- Havendo medidas de injeção relacionando nós de caminhos de grafo diferentes; neste caso o sistema não é observável como um todo, não sendo possível dizer-se que as redes, associadas aos caminhos de grafo isolados, sejam observáveis. Para torná-las ilhas observáveis, importa identificar e eliminar as medidas de injeção que conectam esses subgrafos; depois refatorar a nova matriz ganho, e, se não existirem mais aquelas medidas, os caminhos de grafo relacionados com a nova matriz refatorada são ilhas observáveis da rede.

2.5.3 - Medidas irrelevantes e ramos não observáveis

Suponha um estado $\underline{\theta}$ não observável, obtido de $H \underline{\theta} = \underline{0}$, arranjando $\underline{\theta}$ de modo que os componentes associados com subconjuntos observáveis isolados sejam agrupados entre si.

Sejam três grupos de valores, $\underline{\theta}_\alpha$, $\underline{\theta}_\beta$ e $\underline{\theta}_\gamma$, isto é :

$\underline{\theta} = (\underline{\theta}_\alpha, \underline{\theta}_\beta, \underline{\theta}_\gamma)$, sendo a subrede $\underline{\theta}_\alpha$ formada pelos seus próprios nós e pelos ramos que os conectam; analogamente para as subredes $\underline{\theta}_\beta$ e $\underline{\theta}_\gamma$.

Agrupando agora as medidas associadas com os três grupos, de uma maneira isolada, e colocando também as medidas (injeção) que conectam nós pertencentes a diferentes subredes, resulta o sistema de equações traduzido pela figura a seguir (figura 2.4):

$$\begin{bmatrix} H & & \\ & H_\beta & \\ & & H_\gamma \\ H_{\alpha\beta\gamma} & & \end{bmatrix} \begin{bmatrix} \underline{\theta}_\alpha \\ \underline{\theta}_\beta \\ \underline{\theta}_\gamma \end{bmatrix} = \underline{Q}$$

Figura 2.4 $\Rightarrow$ Sistema de equações representando as três subredes

Os subgrupos $\underline{\theta}_\alpha$, $\underline{\theta}_\beta$ e $\underline{\theta}_\gamma$ são encontrados sem precisar resolver $G \underline{\theta} = \underline{Q}$, pois os nós dos subgrupos serão constituídos pelos nós pertencentes aos três caminhos de grafos distintos, encontrados na forma triangular da matriz G .

O conjunto de medidas, correspondente a $H_{\alpha\beta\gamma}$, é constituído por medidas (injeções), que relacionam variáveis de diferentes caminhos de grafos (Teorema V). Os ramos não observáveis serão aqueles que não participam de nenhum caminho de grafo existente. Desta forma, aquelas medidas são irrelevantes, no sentido de encontrar possíveis subredes observáveis. No entanto, o principal objetivo em vista é transformar a rede em uma rede observável como um todo.

2.5.4 - Adição de medidas usando caminhos de fatoração

Quando se analisa observabilidade de rede, podemos encontrar apenas um caminho de grafo associado a uma determinada rede. Esta situação é a desejada, pois indica que a rede em análise é observável como um todo. Mas é possível acontecer que se obtenha uma rede com vários caminhos de grafo associados a ela. Quando isto acontecer e não existirem medidas de injeção, relacionando nós de subredes associadas a caminhos de grafos diferentes, tais subredes são observáveis isoladamente, e, quando aquelas medidas existirem, as subredes são apenas candidatas a subredes observáveis.

Entretanto, o que se pretende, neste caso, é determinar um conjunto mínimo de medidas, a ser adicionado ao conjunto de medidas já existentes, para tornar a rede observável como um todo. A solução para isso é a adição de medidas de fluxo ou injeção, que relacionem nós de todos os caminhos de grafo, ligando-os entre si e transformando assim esses caminhos de grafo, interligados, em um único caminho de grafo, tornando a rede observável como um todo. Estas medidas adicionais objetivaram, apenas, forçar as subredes a terem o mesmo ângulo de fase como referência, não contaminando o estimador de estado em si. É bom destacar que, através desta teoria, torna-se a rede observável como um todo, sem a necessidade de qualquer refatoração de matriz, não sendo necessário, também, conhecer as possíveis subredes observáveis.

A seguir, é apresentado o algoritmo desenvolvido por BRETAS et alii (1996b), baseado na teoria desenvolvida pelo próprio autor.

2.5.5 - Algoritmo

As subredes observáveis do sistema serão resultantes da interseção das subredes correspondentes aos modelos $P\theta$ e QV .

Os passos do algoritmo são os seguintes:

passo 1- Com o grupo de medidas disponíveis montar a matriz H .

passo 2- Montar a matriz ganho G_0^{DC} .

passo 3- Realizar a fatoração triangular da matriz ganho G_0^{DC} .

passo 4- Encontrar os caminhos de fatoração, associados à matriz triangular G . Em caso de obter-se apenas um único caminho de grafo, a rede é observável; então pare. Caso contrário continue.

passo 5- Se, associado com a fatoração, existir mais de um caminho de grafo, a rede, como um todo, é não observável.

É apresentado agora um esquema, para encontrar possíveis subredes observáveis, quando a rede não é observável como um todo.

2.5.6 - Esquema para identificação de subredes observáveis

passo 1- Se for encontrado mais de um caminho de grafo, associados com a triangulação de G , temos duas situações:

a) Não existindo medidas (injeção), relacionando nós de diferentes caminhos de grafo, as subredes associadas a cada caminho de grafo já constituirão subredes observáveis; então pare. Caso contrário continue.

b) Existindo medidas (injeção), relacionando nós de diferentes caminhos de grafo, nada se pode dizer sobre a observabilidade das subredes associadas com esses caminhos de grafo.

passo 2- Identifique então tais medidas e descarte-as do conjunto original de medidas, já que são irrelevantes, considerando estimação de estado.

passo 3- Atualize a fatoração triangular.

passo 4- Retorne ao passo 1.

Este algoritmo pode tornar-se iterativo, caso ocorra a situação b do passo 1. A razão disto é que, quando as medidas irrelevantes são identificadas e descartadas, outras medidas irrelevantes podem aparecer.

3 - Identificação de medidas críticas

O objetivo deste trabalho é elaborar novos métodos para a identificação de medidas críticas, mais simples e que possam ser desenvolvidos em menos tempo.

Poucas pesquisas foram desenvolvidas na busca por métodos para a identificação de medidas críticas; para mostrarmos como procederam à identificação dessas medidas e a fim de servir de base para comparação com os métodos que ora vamos apresentar, serão divulgados, neste capítulo, dois métodos já existentes. O primeiro é um método combinatorial e o segundo trata da identificação das medidas críticas, através das matrizes unimodulares M e A .

3.1 - Método combinatorial para identificação de medidas críticas

KRUMPHOLZ et alii (1980) desenvolveram um algoritmo para análise de observabilidade de sistemas de potência (item 2.2 página 8) e então o adaptaram para identificação de medidas críticas, como mostrado por CLEMENTS et alii (1981). É bom lembrar que o algoritmo proposto é puramente combinatorial, não exigindo nenhum tipo de cálculo. Necessita,

entretanto, da topologia da rede e da localização das medidas. A seguir, são apresentados os principais passos do algoritmo para análise de observabilidade, já apresentados no item 2.2.1, página 10, tendo em vista mostrar depois como este algoritmo é usado para a identificação de medidas críticas.

3.1.1 - Algoritmo

O algoritmo desenvolvido por KRUMPHOLZ et alii (1980), para análise de observabilidade, tem bases topológicas, sendo bom lembrar, portanto, antes de apresentá-lo, que um sistema de potência é topologicamente observável, relacionado a um conjunto de medidas, se existir associado a esse sistema uma árvore representativa de posto completo (uma árvore incidente a todas as barras da rede, possuindo uma medida associada a cada um de seus ramos). Assim, a idéia do algoritmo é construir tal árvore. Se for possível esta construção, o sistema é observável. O algoritmo é composto de duas partes principais. Na primeira parte são processadas as medidas de fluxo. Através destas medidas, é construída uma floresta F, de ramos com medidas de fluxo, sendo também identificadas as medidas de injeção limitadas (injeção limitada é a injeção que é incidente a ramos com medidas de fluxo e àqueles sem medida de fluxo, concomitantemente). Na segunda parte do algoritmo, as injeções limitadas são processadas, uma de cada vez, na tentativa de estender-se a floresta F a todas as barras, sem medidas de injeção de rede, obtendo assim uma árvore representativa de posto completo.

Para entender a idéia deste algoritmo, é indicado fazer-se analogia com vetores. Seja então o algoritmo a seguir, próprio para selecionar uma base vetorial para um determinado conjunto de vetores. Iniciemos escolhendo um vetor qualquer da coleção disponível de vetores, denominando-o de v_1 . Suponhamos que, de alguma forma, um conjunto linearmente independente $\{v_1, \dots, v_i\}$ tenha sido encontrado. Seleccionemos outro vetor v_{i+1} da coleção. Se v_{i+1} é linearmente independente de $\{v_1, \dots, v_i\}$, juntemos v_{i+1} ao conjunto anterior e o novo conjunto torna-se $\{v_1, \dots, v_i, v_{i+1}\}$. Se v_{i+1} é dependente do subconjunto $\{v_1, \dots, v_i\}$, então v_{i+1} é classificado como redundante e é excluído da coleção, pois v_{i+1} não aumenta a dimensão do subespaço gerado pelos vetores $v_1, \dots, v_i$. Desta forma, em cada passo deste algoritmo, o conjunto $\{v_1, \dots, v_i\}$ consiste de vetores linearmente independentes. Quando um determinado número de vetores forem acrescentados ao conjunto $\{v_1, \dots, v_i\}$, o suficiente para que o espaço do subconjunto seja igual ao espaço vetorial do conjunto original de vetores, uma base foi encontrada.

Tentaremos então transferir a lógica anterior para o ambiente de grafos. A noção de redundância topológica e árvore de posto completo equivale à noção algébrica de dependência e independência linear de vetores. Em cada passo do algoritmo de observabilidade, é feita uma tentativa de aumentar a floresta F , adicionando-lhe novos ramos, cada qual associado a uma medida de injeção ainda não relacionada. E assim, as árvores que formam a floresta F de ramos vão se agrupando, até que se chegue a uma árvore contendo todos os nós, sem medida de injeção da

rede. Logo, o conjunto de medidas, associado aos ramos dessa árvore de posto completo, corresponde a um conjunto de vetores linearmente independentes. Se uma determinada medida não puder ser usada para aumentar a floresta F , a medida é topologicamente redundante, pois as informações trazidas por essa medida não podem ser usadas para aumentar o conjunto de informações, conseguido através das outras medidas.

Voltando novamente à analogia com vetores, na determinação da base para uma coleção de vetores, é possível, em geral, que diversas bases possam ser selecionadas. O algoritmo apresentado anteriormente, para determinação da base de um conjunto de vetores, classifica os vetores não críticos como redundantes e estes são identificados da seguinte forma:

Quando um vetor v é tirado da coleção original de um dado conjunto de vetores e o espaço vetorial desse conjunto continua com a mesma dimensão, isto indica que o vetor V é linearmente dependente dos demais vetores. Caso contrário, o vetor é crítico. Passando agora para o conjunto de medidas, em cada execução do algoritmo de observabilidade é encontrada, analogamente, uma medida crítica. Após a construção da floresta F , o processamento das injeções limitadas começa buscando estender a floresta F a todas as barras, sem medida de injeção da rede. Quando uma medida não puder aumentar o tamanho de F , o algoritmo gera uma subrede observável, para a qual a medida é redundante. Assim, as medidas utilizadas para tornar essa subrede observável e a medida redundante identificada constituem um conjunto dependente de medidas, no sentido de que qualquer medida possa ser removida deste conjunto, sem

afetar a observabilidade da subrede observável, ou a observabilidade de toda a rede. Portanto, as medidas críticas são aquelas que não pertencem a qualquer subrede observável, gerada pelo algoritmo.

A grande desvantagem deste método, assim como do método combinatorial para análise de observabilidade, é que ambos estão sujeitos a uma explosão combinatorial, tornando-os lentos.

3.2 - Identificação de medidas críticas, via matrizes unimodulares M e A

Tendo em vista a propriedade segundo a qual a matriz incidência de medidas para ramos M e a matriz incidência de ramos para nós A serem matrizes unimodulares, isto é, matrizes que têm a propriedade de possuir os determinantes de cada submatriz quadrada iguais a -1, 1 ou 0, CRAINIC et alii (1990) apresentam uma aproximação algébrica para avaliar a rigidez de um sistema de medidas. A rigidez de um sistema de medidas é definida como o número total de N subconjuntos de medidas, linearmente independentes, existentes no conjunto de M medidas, sendo $N \leq M$.

Onde: $N = n - 1$ (para o modelo Pδ)

$N = n$ (Para o modelo QV)

n é o número de nós da rede

Para fazer a identificação de medidas críticas, os autores apresentam algumas novas definições, que são dadas a seguir.

Se a matriz produto $W = M A^t$, onde W é a matriz de incidência de medidas para nós $m \times (n - 1)$, tem Gramiano ($G(W) = \det(W^t W)$) igual a

zero , a rede é não observável; caso contrário ($G(W) \neq 0$), a rede é observável (ver GANTMACHER (1977)). Se o Gramiano é invariante, então existem n matrizes distintas A e n matrizes distintas W . Como resultado obtêm-se as seguintes conceituações:

- a) Conceituação de árvore topologicamente representativa (TST);
- b) Conceituação de árvore representativa de medidas (MST);
- c) Conceituação de árvore representativa de posto completo (FRST);

O conceito de TST é idêntico à noção de árvore representativa, usado na teoria de grafo, isto é, uma árvore contendo todos os n nós e os $(n-1)$ ramos de uma determinada rede de potência. O número total de TSTs de uma rede será representado por r .

Uma FRST é uma árvore contendo todos os n nós da rede, com a propriedade de ter, para cada um dos seus $(n-1)$ ramos, uma medida atribuída. A atribuição de uma medida para um determinado ramo é feita da seguinte forma:

- 1) Medida de fluxo $\Rightarrow$ associada ao ramo que a contém;
- 2) Medida de injeção $\Rightarrow$ associada a um dos ramos incidentes ao nó da injeção;

O número total de FRSTs será representado por ρ .

Árvore representativa de medidas (MST) representa qualquer subconjunto de $(n-1)$ medidas, linearmente independentes (medidas de potência ativa e reativa). Para uma rede observável, o número total de MSTs, designado por μ , representa a conectividade do sistema de medida. Para melhor entendimento das novas conceituações, observe a figura 3.1 (página 36).

Na figura 3.1, temos uma rede composta por 3 nós e 3 ramos , à qual está associado um conjunto de 6 medidas, sendo 3 medidas de injeção e 3 medidas de fluxo. Através dessa rede obtemos 3 TSTs, representadas pelas redes (a), (b) e (c). Associando a essas 3 TSTs o conjunto de medidas dado, podemos obter 6 MSTs, que são representados por $\mu_1, \mu_2, \mu_3, \mu_4, \mu_5$ e μ_6 . Da associação de todos os TSTs com os MSTs, obtemos 13 FRSTs (veja figura 3.1, página 36).

Fazendo um paralelo entre o número de TSTs (r) e o número de MSTs (μ) , temos que : se r é a medida da conectividade da rede, quanto maior for o valor de r , maior será a conectividade topológica da rede; se μ representa a rigidez do sistema de medidas, quanto maior for o valor de μ , maior será a rigidez desse sistema. Assim, ambos, r e μ , podem ser usados para identificar, respectivamente, os ramos e as medidas críticas, isto é: sendo o grau do i -ésimo ramo o número de vezes que este ramo aparece no número r de TSTs, representado por $d(b_i)$, o i -ésimo ramo é crítico se :

$$r = d(b_i) \quad (3.1)$$

Sendo o grau do par de ramos (i, j)-ésimo o número de vezes que esse par de ramos aparece no número r de TSTs, representado por $d(b_i, b_j)$, o (i, j)ésimo par de ramos é crítico se :

$$r = d(b_i) + d(b_j) - d(b_i, b_j) \quad (3.2)$$

Se a rede tiver pelo menos um ramo crítico , denominar-se-á rede conectada por 1 ramo. Mas, se não tiver nenhum ramo crítico, mas possuir no mínimo um par de ramos críticos, será chamada de rede conectada por 2 ramos. Por exemplo, o sistema de 14 barras da IEEE, mostrado na figura

5.2, página 58, é uma rede conectada por 1 ramo; já a rede apresentada na figura 3.1, página 36, é uma rede conectada por 2 ramos, pois o número r de TSTs obtidos dessa rede é igual a 3, sendo que nenhum dos seus ramos aparece 3 vezes no número r de TSTs, mas apenas 2 vezes. Assim, o ramo (1-2) aparece na primeira TST (rede (a) figura 3.1) e na última (rede (c) figura 3.1). Logo, através da equação 3.1, conclui-se que essa rede não é conectada por um ramo, ou seja, não possui nenhum ramo crítico. Portanto, a retirada de qualquer um dos ramos de tal rede não vai tirar a sua conectividade, pois a mesma continuará conectando todos os seus nós, através dos ramos restantes. Contudo, analisando agora o par de ramos, por exemplo o par ((1-2) , (2-3)), verificar-se-á que o mesmo aparece apenas 1 vez no número total de TSTs; logo, o grau desse par de ramos é: $d((1-2) , (2-3)) = 1$. Levando à equação (3.2), temos que:

$$r = d(1-2) + d(2-3) - b((1-2) , (2-3))$$

sendo:

$$r = 3$$

$$d(1-2) = 2$$

$$d(2-3) = 2$$

$$d((1-2) , (2-3)) = 1$$

resolvendo a equação chegamos a:

$$3 = 2 + 2 - 1$$

$$3 = 3$$

Assim, a rede em estudo é conectada por 2 ramos.

Consequentemente, a retirada de qualquer conjunto de 2 ramos da rede vai

tirar-lhe a conectividade, porquanto a mesma não vai conseguir conectar todos os seus nós com os ramos restantes.

Similarmente, sendo o grau da i -ésima medida o número de vezes que a mesma aparece no número μ de MSTs, representado por $d(m_i)$, a medida é crítica se:

$$\mu = d(m_i) \quad (3.3)$$

Por exemplo, utilizando para teste, novamente, o sistema de 14 barras da IEEE, mostrado na figura 5.2, página 58, com o conjunto de medidas dado por CLEMENTS et alii. (1981), tal sistema de medida apresenta a rigidez de $\mu = 18$ e as seguintes medidas críticas: as injeções nas barras 9,11,12,13 e 14, bem assim os fluxos nos ramos 13 e 15.

Definindo grau do par de medidas (i, j) -ésimo, como sendo o número de vezes que esse par de medidas aparece no número μ de MSTs, representado por $d(m_i, m_j)$, o par de medidas em questão é crítico se:

$$\mu = d(m_i) + d(m_j) - d(m_i, m_j) \quad (3.4)$$

Para determinar o número r de TSTs e o número ρ de FRSTs, os autores apresentam fórmulas analíticas, o que não fazem, entretanto, para a determinação do número μ de MSTs. Para obter o valor numérico do μ , e, ao mesmo tempo, analisá-lo, é usado um algoritmo do tipo combinatorial, que consiste em verificar todas as matrizes não singulares W_i , de ordem $(n-1)$, que possam ser derivadas da matriz W $m \times (n-1)$. Este processo pode tornar-se muito lento na maioria dos casos.

Figura 3.1 $\Rightarrow$ Representação das novas conceituações (TST, MST, FRST)

4 - Métodos propostos

Neste capítulo serão propostos dois métodos para identificação de medidas críticas, a seguir descritos. Assim o fizemos na tentativa de obter métodos mais simples e de aplicação em tempo reduzido. O primeiro método proposto fundamenta-se no desenvolvido por BRETAS (1996a), para análise de observabilidade. Isto porque o método de Bretas analisa a observabilidade da rede, com lógica e simplicidade, sem exigir solução alguma de sistemas de equações algébricas, sendo portanto de fácil implantação. Destarte, o primeiro método proposto trata da identificação das medidas críticas, através dos caminhos de grafo.

O segundo método foi desenvolvido seguindo um outro caminho. Tendo em vista a grande velocidade de execução, correspondente ao método de análise de observabilidade, através da redução simbólica da matriz jacobiana desenvolvido por SLUTSKER & SCUDDER (1987), afigurou-se-nos razoável instituir um método que identificasse as medidas críticas analisando a estrutura dessa matriz. É um método muito prático, em razão de sua maior velocidade de execução, oriunda de não exigir cálculo algum e não ser de natureza combinatorial.

4.1 - Identificação de medidas críticas através de caminhos de grafo

Utilizando o princípio proposto por BRETAS (1996a), para análise de observabilidade de sistemas elétricos, estamos apresentando uma metodologia para identificação de medidas críticas (através dos caminhos de grafo, obtidos da fatoração triangular da matriz ganho G). Os caminhos de grafo são construídos através de uma busca dos elementos não zero, na parte superior da matriz ganho tornada triangular. O elemento não zero indica que existe uma conexão entre as barras representadas pela coluna e as representadas pela linha desse elemento.

4.1.1 - Descrição do método

A análise é diferenciada: uma destina-se à medida de injeção e outra à medida de fluxo; é feita medida por medida. Primeiramente é feita a análise de todas as medidas de injeção, para depois ser feita a análise das medidas de fluxo, isto porque, para realizar a análise das medidas de fluxo, é necessário saber quais são as medidas de injeção críticas. Procede-se à análise, através das seguintes providências:

1) Medida de injeção

Suponha uma medida de injeção de potência na barra I ;

Cumpra verificar, no grafo, todas as barras ligadas à barra I , buscando, na linha correspondente à barra I , as barras relativas às colunas em que aparecem elementos diferentes de zero, buscando ainda as medidas associadas às barras (não é necessário que essas barras sejam ligadas à barra I topologicamente). Se for possível chegar a todas essas

barras, partindo da barra I , através de um caminho topológico, a cada um de cujos ramos tenha sido fixada uma medida distinta, sem utilizar a medida de injeção em I , a medida é redundante; pois não é necessária para manter a conectividade dessa rede; caso contrário, a medida é crítica.

Toda vez que for determinada uma medida crítica, esta medida é automaticamente transformada em um fluxo equivalente, no ramo em que ela é crítica.

2) Medida de fluxo

Suponha uma medida de fluxo de potência no ramo J :

- A seguir identifique as barras adjacentes a esse ramo. Vamos chamá-las de J_{inicial} e J_{final} . Para descobrir se essa medida de fluxo é crítica ou não, são necessários dois testes, que são os seguintes:

- teste 1: Se existir em, pelo menos, uma dessas barras (J_{inicial} ou J_{final}) alguma medida de injeção redundante, a medida de fluxo no ramo J é redundante; caso contrário efetue o teste seguinte:

- teste 2: Se for possível ir da barra J_{inicial} à barra J_{final} , através de um caminho topológico, a cada um de cujos ramos tenha sido fixada uma medida distinta, analogamente ao que foi dito anteriormente (para análise de medida de injeção), sem utilizar a medida de fluxo no ramo J , esta medida é redundante; caso contrário, é crítica.

obs.: Para aumentar a velocidade de execução neste método, sempre que a análise de uma medida de injeção estiver concluída, faz-se uma redução

simbólica das partes observáveis do sistema, isto é, eliminam-se da rede as barras que não possuam medida de injeção a analisar e só estejam ligadas a barras com a mesma característica (sem medida de injeção a analisar), reduzindo assim o tamanho do grafo e consequentemente a duração da busca.

Depois de analisar todas as medidas de injeção, inicia-se a análise das medidas de fluxo, a qual então se realiza considerando a rede inteira, sem a eliminação feita durante a análise das medidas de injeção.

Para ficar mais clara essa metodologia, é dado um exemplo da sua aplicação, na rede de 6 barras da IEEE, com o conjunto de medidas mostrado na figura 4.1 (página 42).

4.1.2 - Algoritmo

Os passos do algoritmo são os seguintes:

Passo 1 - Com o grupo disponível de medidas, montar a matriz jacobiana H

Passo 2 - Montar a matriz ganho G

Passo 3 - Realizar a fatoração triangular da matriz ganho G

Passo 4 - Encontrar os caminhos de grafo associados à matriz ganho tornada triangular

Passo 5 - Selecionar a primeira medida de injeção

Passo 6 - Buscar, no grafo, todas as barras ligadas àquela onde a injeção é medida

Passo 7 - Se for possível chegar a todas as barras definidas no item 6, sem usar a medida de injeção em questão, a medida é redundante; caso contrário, é crítica

Passo 8 - Em existindo alguma barra sem medida de injeção a analisar e ligada apenas a barras com a mesma característica (sem medida de injeção a analisar), elimine-se essa barra para as próximas análises de medida de injeção

Passo 9 - Havendo mais alguma medida de injeção a analisar, retorne ao passo 6; se não houver, passe à consideração do passo seguinte

Passo 10 - Selecione a primeira medida de fluxo

Passo 11 - Identifique as barras adjacentes ao ramo, onde o fluxo é medido

Passo 12 - Se existir, em alguma dessas barras definidas no item 11, uma medida de injeção redundante, o fluxo é redundante; caso contrário, passar ao próximo passo

Passo 13 - Sendo possível unir as duas barras adjacentes, através de um outro caminho topológico, a cada um de cujos ramos tenha sido fixada uma medida distinta, sem utilizar a medida de fluxo aludida, o fluxo é redundante; caso contrário, é crítica

Passo 14 - Se existir mais medida de fluxo a analisar, volte ao passo 11; em não existindo, impõe-se a paralisação.

4.1.3 - Exemplo

Figura 4.1 ⇒ Topologia da rede de 6 barras da IEEE, associada a um conjunto de 6 medidas

O primeiro passo consiste em montar a matriz ganho, para depois torná-la triangular e então obter os caminhos de grafo correspondentes.

Seja a matriz jacobiana correspondente ao sistema da figura (4.1); atribuindo às reatâncias de linha o valor 1 e considerando a matriz peso de medidas como sendo uma matriz unitária; daí resulta uma matriz ganho que, tornada triangular, apresenta-se da seguinte forma:

$$G\Delta = \begin{bmatrix} 1 & -0.53 & -0.47 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -3 & 1 & 1 \\ 0 & 0 & 0 & 1 & -0.56 & -0.44 \\ 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Como existe apenas um caminho de grafo, associado à fatoração triangular dessa matriz ganho, segundo BRETAS (1996a), o sistema é observável. E da matriz ganho tornada triangular resultam os seguintes ramos de grafo:

$$1 \Rightarrow 2 \Rightarrow 3$$

$$2 \Rightarrow 1 \Rightarrow 3$$

$$3 \Rightarrow 1 \Rightarrow 2 \Rightarrow 4 \Rightarrow 5 \Rightarrow 6$$

$$4 \Rightarrow 3 \Rightarrow 5 \Rightarrow 6$$

$$5 \Rightarrow 3 \Rightarrow 4 \Rightarrow 6$$

$$6 \Rightarrow 3 \Rightarrow 4 \Rightarrow 5$$

Início da análise.

Análise de medidas de injeção:

- M1 (injeção na barra 1);

No grafo, a barra 1 está ligada às barras 2 e 3, sendo que da barra 1 é possível ir à barra 2, através do ramo 1-2, e este ramo pode ser associado à medida M3; logo, para o mesmo, a medida M1 não é crítica, mas, da barra 1 para a barra 3, não existe um caminho sem a utilização da medida M1, pois os caminhos topológico que existem são: o ramo 1-3, ou através dos ramos 1-2 e 2-3; para que se possa associar a cada ramo destes caminhos uma medida, a medida M3 é necessária. Assim, a medida de injeção na barra 1 é crítica neste ramo (1-3), então, tal medida é simbolicamente transformada num fluxo no ramo 1-3, para as próximas análises.

Antes de fazer a análise da próxima medida de injeção, faz-se uma análise das barras 1 e 2, e, como estas barras não possuem medidas de

injeção a serem analisadas e também não são adjacentes a nenhuma barra que contenha medida de injeção a analisar, tais barras são eliminadas, pois formam partes observáveis da rede, e o grafo fica:

$$3 \Rightarrow 4 \Rightarrow 5 \Rightarrow 6$$

$$4 \Rightarrow 3 \Rightarrow 5 \Rightarrow 6$$

$$5 \Rightarrow 3 \Rightarrow 4 \Rightarrow 6$$

$$6 \Rightarrow 3 \Rightarrow 4 \Rightarrow 5$$

- Medida M4 (injeção na barra 4):

No grafo, a barra 4 está ligada às barras 3, 5 e 6. Da barra 4 à barra 3, só existe um caminho topológico, que é o ramo 3-4, e a única medida possível de ser associada ao mesmo é a medida M4. Portanto, essa medida é crítica e é transformada simbolicamente num fluxo, no ramo 3-4, para as próximas análises.

Agora a barra 3 é eliminada e o grafo fica :

$$4 \Rightarrow 5 \Rightarrow 6$$

$$5 \Rightarrow 4 \Rightarrow 6$$

$$6 \Rightarrow 4 \Rightarrow 5$$

- Medida M5 (injeção na barra 5):

No grafo, a barra 5 está ligada às barras 4 e 6 :

- Para ir da barra 5 à barra 4, existe apenas o ramo 4-5, e, como podemos associar a esse ramo a medida de fluxo M2, para tal ramo a medida M5 não é crítica. Continuando, para ir da barra 5 à barra 6, temos o

seguinte caminho topológico: os ramos 5-4 e 4-6. De 5 para 4, temos o fluxo M2, e, para o ramo 4-6, podemos associar a medida M6 (injeção na barra 6). Portanto, a medida M5 também não é crítica para esse ramo; logo é uma medida redundante.

Agora a barra 5 é eliminada, e o grafo fica:

$$4 \Rightarrow 6$$

$$6 \Rightarrow 4$$

- Medida M6 (injeção na barra 6):

No grafo, a barra 6 está ligada apenas à barra 4. O único caminho para irmos da barra 6 à barra 4 é o ramo 4-6; como já não existe a medida de injeção em 4, pois a mesma já foi identificada como medida crítica e designada para o ramo 3-4, a única medida que pode ser associada ao ramo 4-6 é a medida M6, portanto, esta medida M6 é uma medida crítica, e, para outras análises, ela é convertida em um fluxo associado ao ramo 4-6.

Como todas as medidas de injeção já foram analisadas, inicia-se então a análise das medidas de fluxo, para a qual consideramos o caminho de grafo usado no começo da análise das medidas de injeção.

- M3 (fluxo no ramo 1-2):

- teste 1: não existe medida de injeção redundante em nenhuma das barras adjacentes a esse ramo, que são as barras 1 e 2.

- teste 2: um outro caminho topológico alternativo, para unir essas duas barras 1 e 2, seria através dos ramos 1-3 e 3-2, mas, em razão de não haver medidas disponíveis a serem associadas a esses dois ramos, a medida M3 é crítica.

- M2 (fluxo no ramo 4-5):

- teste 1: a medida de injeção na barra 5 é redundante, podendo assim ser usada para irmos da barra 4 à barra 5; conseqüentemente, a medida M2 é redundante.

E assim é feita a análise das medidas, através dos caminhos de grafo.

No anexo B são apresentados o fluxograma e o programa, correspondentes a esse método.

4.2 - Identificação de medidas críticas através da matriz jacobiana

Buscando um método mais rápido para a identificação de medidas críticas, que não necessitasse de nenhum cálculo e que não fosse de natureza combinatorial, os estudos foram direcionados para a matriz jacobiana, por ser esta matriz que relaciona o conjunto de medidas com a topologia de rede.

Apenas uma metodologia analisa a observabilidade, através de uma redução simbólica da matriz jacobiana desenvolvida por SLUTSKER & SCUDDER (1987); normalmente tal matriz é utilizada, apenas, como ponto de partida para estudos.

4.2.1 - Metodologia

Este método (2º método proposto) tem como objetivo identificar as medidas críticas, analisando a estrutura da matriz jacobiana, importando

investigar quais elementos desta matriz são diferentes de zero e não os valores reais desses elementos, porquanto, através da localização dos mesmos, obtém-se a relação entre as medidas e os estados do sistema. Veja a figura a seguir (figura 4.2):

Figura 4.2 $\Rightarrow$ Exemplo de uma rede de 4 barras

Para o conjunto de medidas, associado à rede mostrada na figura 4.2, obtêm-se a seguinte matriz jacobiana:

$$H = \begin{bmatrix} X & X & & \\ X & X & X & X \\ & X & & X \\ & & & \end{bmatrix}$$

sendo que X representa os elementos não nulos da matriz.

As linhas correspondem às equações de medida, enquanto que as colunas correspondem às variáveis relacionadas pelas medidas. Cada coluna corresponde a uma determinada barra da rede.

Quando uma medida relaciona mais de duas variáveis (medida de injeção), essas variáveis são agrupadas em conjuntos de duas variáveis, tendo como referência a variável correspondente à barra de injeção; assim, da matriz jacobiana H obtemos os conjuntos de duas variáveis, mostrados na tabela a seguir:

Tabela 4.1 $\Rightarrow$ Conjuntos de 2 variáveis de cada medida

MEDIDAS	CONJUNTOS DE 2 VARIÁVEIS
M1 (FLUXO NO RAMO 1-2)	(1-2)
M2 (INJEÇÃO NA BARRA 2)	(1-2), (2-3) ,(2-4)
M4 (INJEÇÃO NA BARRA 4)	(2-4)

Através da matriz Jacobiana, podemos ter os conjuntos de duas variáveis inerentes a cada medida, podendo então ser identificado o conjunto fundamental para a observabilidade do sistema, isto é, o conjunto que só pode obter-se, através de uma dada medida, identificando assim as medidas críticas do sistema. Dentro do conjunto de vetores linhas, do qual a matriz jacobiana é formada, as equações das medidas redundantes são os vetores linearmente dependentes, pois podem ser obtidos através de uma combinação linear dos demais vetores. Logo, as medidas críticas são associadas aos vetores linearmente independentes do sistema.

O método analisa medida por medida, verificando se uma delas possui um conjunto de duas variáveis, que não pode ser obtido através das outras medidas. Para fazer a análise, é bom observar que as medidas de

fluxo só relacionam um conjunto de duas variáveis de cada vez; já as medidas de injeção podem relacionar um ou mais conjuntos. Assim, é necessário que o algoritmo sempre comece a análise pelas medidas de injeção, pois, se a medida for crítica, é necessário determinar qual o conjunto crítico de duas variáveis dessa medida e eliminar os demais conjuntos, caso a medida possua mais de um, alterando assim a estrutura da matriz jacobiana; a medida de fluxo, sendo crítica ou não, possui sempre, apenas, um conjunto de duas variáveis, portanto independente do tipo de medida (crítica ou redundante); a estrutura da matriz H não sofre alteração, pois nenhum grupo será apagado.

4.2.2 - Descrição do método

A verificação se a medida é crítica ou não é feita através de dois testes, que são usados tanto para medidas de injeção, como para medidas de fluxo. São os seguintes:

Teste 1-Todas as linhas da matriz H são percorridas, para verificar se o conjunto de duas variáveis em análise aparece mais de uma vez;

Teste 2-O algoritmo tenta outros caminhos, através da topologia da rede e das informações obtidas da matriz jacobiana, caminhos esses que unam as duas barras identificadas pelas duas variáveis, pertencentes aos conjuntos que a medida possui.

Se os dois testes acima não resultarem positivos , o conjunto de duas variáveis em análise é um conjunto exclusivo dessa medida. Como consequência, a medida em questão é uma medida crítica. Caso essa

medida possua outros conjuntos de duas variáveis, esses outros conjuntos são eliminados da matriz H e a análise continua. Se, pelo menos um dos dois testes for positivo, o conjunto em questão é redundante. O teste então continua para outro conjunto contido na medida, até que todos os conjuntos de duas variáveis sejam processados.

Obs.: Toda vez que uma medida de injeção for classificada como crítica, é necessário ser feita uma análise, com base nos conjuntos de duas variáveis dessa medida, os quais serão eliminados. Isto porque, caso algum desses conjuntos relacione alguma variável, correspondente à barra com medida de injeção já analisada e classificada como redundante, importa que a análise dessa medida redundante seja feita novamente. Tal conjunto oriundo da aludida medida redundante será eliminado, podendo a mesma mostrar-se então como uma medida crítica. Quando a injeção é classificada como redundante, os conjuntos de duas variáveis dados por ela não sofrem alterações.

4.2.3 - Algoritmo

A segunda metodologia proposta foi implantada e o seu algoritmo formal pode ser descrito da seguinte maneira:

- 1-Forme a matriz H correspondente;
- 2-Selecioner a primeira medida de injeção;

3-Realize os testes 1 e 2 anteriores:

a) Se todos os testes resultarem negativos, a medida é crítica e é feita então uma análise, fundamentada nas duas variáveis dos conjuntos que serão eliminados, caso existam. Se existir alguma medida de injeção, em alguma barra correspondente às variáveis eliminadas, cumpre voltar para o passo 2, com a injeção em questão;

b)-Se algum dos testes resultar positivo, a medida em análise é redundante ;

4-Se existir medida ainda não analisada, volte ao passo 2 e quando se processarem todas as medidas de injeção, o algoritmo processará as medidas de fluxo.

4.2.4 - Exemplo

Para melhor entendimento deste método, é mostrado um exemplo do seu funcionamento, no sistema de 6 barras mostrado na figura 4.1:

Através do conjunto de medidas mostrado na figura 4.1, a rede é observável.

- Construa simbolicamente a matriz jacobiana H.

$$H = \begin{bmatrix} X & X & & & & \\ & & & X & X & \\ X & X & X & & & \\ & & & X & X & X & X \\ & & & & X & X & \\ & & & & & X & X \end{bmatrix}$$

- M1 (injeção na barra 1): engloba os seguintes conjuntos de duas variáveis:

(1-2) e (1-3). O conjunto (1-2) aparece também na primeira linha da matriz , entretanto, o conjunto (1-3) só aparece uma vez. Tentando outro caminho que leve da barra 1 à barra 3, por exemplo o ramo 3-2, pois já se sabe que a barra 2 está conectada com a barra 1 mais de uma vez, verifica-se que o conjunto (3-2) só aparece uma vez (na própria medida M1). Tal conjunto é exclusivo dessa medida; assim, a medida de injeção M1 é CRÍTICA no ramo 1-3. Será então eliminado o conjunto de variáveis (1-2) dessa medida e como esse conjunto não relaciona nenhuma barra com medida de injeção, o processo continua e a nova matriz H fica:

$$H = \begin{bmatrix} X & X & & & & \\ & & & X & X & \\ X & & X & & & \\ & & X & X & X & X \\ & & & X & X & \\ & & & X & & X \end{bmatrix}$$

A próxima medida a ser analisada é a medida M4 (injeção na barra 4). Esta possui os seguintes conjuntos de duas variáveis: (3-4) , (4-5) e (4-6). O conjunto (3-4) só aparece uma vez e não há nenhum outro caminho que leve da barra 3 à barra 4. Logo, a medida M4 é CRÍTICA no ramo 3-4, e os outros conjuntos são eliminados.

A nova matriz H resultará em:

$$H = \begin{bmatrix} X & X & & & & \\ & & & X & X & \\ X & & X & & & \\ & & X & X & & \\ & & & X & X & \\ & & & X & & X \end{bmatrix}$$

- Medida M5 (injeção na barra 5): encerra apenas um conjunto que é: (4-5); como este conjunto aparece mais de uma vez na matriz H, o mesmo é REDUNDANTE e a medida o é também, sendo que a matriz H não sofre alteração.
- Medida M6 (injeção na barra 6): possui o conjunto (4-6), que aparece apenas uma vez na matriz e por não ter caminho alternativo de 4 para 6, a medida é CRÍTICA, mas, como tal medida não tem nenhum conjunto de duas variáveis a mais a ser eliminado, a matriz H não sofre mudança.
- Tendo em vista, agora, que todas as medidas de injeção já foram processadas, inicia-se então a análise das medidas de fluxos.
- Medida M3 (fluxo 1-2): encerra o conjunto (1-2), que aparece só uma vez e por não haver caminho alternativo de 1 para 2, a medida é CRÍTICA.
- Medida M2 (fluxo 4-5): encerra o conjunto (4-5), que aparece mais de uma vez. Logo, a medida é REDUNDANTE.

E assim é feita a análise das medidas, através da matriz jacobiana.

No anexo C são apresentados o fluxograma e o programa, correspondentes a esse método.

5 - Testes realizados e análise dos resultados

Estes testes têm como objetivo verificar a eficiência dos métodos propostos neste trabalho, para a identificação de medidas críticas, tendo sido realizados, tão somente, em redes observáveis como um todo. Contudo, os mencionados métodos podem ser adaptados para a identificação das medidas críticas, em redes que possuam ilhas observáveis, ou seja, identificação das medidas que comprometam a observabilidade dessas ilhas. Para analisar os resultados dos dois métodos, chamaremos de método 1 o método que identifica as medidas críticas, através de caminhos de grafo, e de método 2 o método que identifica as medidas críticas, através da matriz jacobiana.

A avaliação e comprovação dos resultados, obtidos através dos testes, realizam-se utilizando um programa que é a implementação do algoritmo desenvolvido por BRETAS et alii (1996b). Através desse programa, as medidas são testadas, uma por uma, da seguinte maneira: Verifica-se, primeiramente, se a rede é observável como um todo; em seguida, retira-se uma medida do conjunto de medidas associado a essa rede observável e testa a observabilidade da rede novamente; se a resposta

do programa for uma rede observável como um todo, a medida é redundante; caso contrário a medida é crítica.

5.1 - Testes realizados

Para realizar os testes, utilizamos a SPARCstation IPX.

5.1.1 - Teste com o sistema de 6 barras da IEEE

A topologia da rede de 6 barras da IEEE, usada para os testes, é a representada na figura (figura 5.1). Tal sistema foi utilizado para teste, por já ter sido usado em vários trabalhos desenvolvidos na área (veja MONTICELLI, A. et alii (1985)).

Figura 5.1 $\Rightarrow$ Topologia da rede de 6 barras da IEEE

SITUAÇÃO 1

Conjunto de 6 medidas, com 2 medidas de fluxo nos ramos 1 e 5; 4 medidas de injeção nas barras 1,4,5 e 6. Os resultados obtidos são os seguintes:

Tabela 5.1 $\Rightarrow$ Resultados do teste com rede de 6 barras (situação 1)

Métodos	Injeção crítica (barra)	Fluxo crítico (ramo)	Tempo de execução (seg.)
1	1,4,6	1	6
2	1,4,6	1	2

SITUAÇÃO 2

Conjunto de 6 medidas, com 2 medidas de fluxo nos ramos 1 e 5; 4 medidas de injeção nas barras 1,2,4 e 6.

Tabela 5.2 $\Rightarrow$ Resultados do teste com rede de 6 barras (situação 2)

Métodos	Injeção crítica (barra)	Fluxo crítico (ramo)	Tempo de execução (seg.)
1	4,6	5	7
2	4,6	5	2

SITUAÇÃO 3

Conjunto de 6 medidas, com 3 medidas de fluxo nos ramos 1,4 e 5; 3 medidas de injeção nas barras 1,4 e 6.

Tabela 5.3 $\Rightarrow$ Resultados do teste com rede de 6 barras (situação 3)

Métodos	Injeção crítica (barra)	Fluxo crítico (ramo)	Tempo de execução (seg.)
1	1	1	6
2	1	1	2

SITUAÇÃO 4

Conjunto de 6 medidas, com 3 medidas de fluxo nos ramos 1,3 e 5; 3 medidas de injeção nas barras 1,4, e 6.

Tabela 5.4 $\Rightarrow$ Resultados do teste com rede de 6 barras (situação 4)

Métodos	Injeção crítica (barra)	Fluxo crítico (ramo)	Tempo de execução (seg.)
1	4,6	5	6
2	4,6	5	2

SITUAÇÃO 5

Conjunto de 6 medidas, com 3 medidas de fluxo nos ramos 1,5 e 6; 3 medidas de injeção nas barras 1,4 e 6.

Tabela 5.5 $\Rightarrow$ Resultados do teste com rede de 6 barras (situação 5)

Métodos	Injeção crítica (barra)	Fluxo crítico (ramo)	Tempo de execução (seg.)
1	1,4	1,5	6
2	1,4	1,5	2

SITUAÇÃO 6

Conjunto de 7 medidas, com 2 medidas de fluxo nos ramos 1 e 5; 5 medidas de injeção nas barras 1,2,3,4 e 6.

Tabela 5.6 $\Rightarrow$ Resultados do teste com rede de 6 barras (situação 6)

Métodos	Injeção crítica (barra)	Fluxo crítico (ramo)	Tempo de execução (seg.)
1	nenhuma	nenhum	7
2	nenhuma	nenhum	2

5.1.2 - Teste com o sistema de 14 barras da IEEE

A topologia da rede de 14 barras da IEEE é mostrada na figura a seguir (figura 5.2), já tendo sido o sistema empregado no trabalho elaborado por CLEMENTS, K. A. et alii (1982).

Figura 5.2 $\Rightarrow$ Topologia da rede de 14 barras da IEEE

SITUAÇÃO 1

Conjunto de 15 medidas, com 6 medidas de fluxo nos ramos 1, 2, 8, 13, 14 e 15; 9 medidas de injeção nas barras 1, 2, 3, 4, 9, 11, 12, 13 e 14.

Tabela 5.7 ⇒ Resultados do teste com rede de 14 barras (situação 1)

Métodos	Injeção crítica (barra)	Fluxo crítico (ramo)	Tempo de execução (seg.)
1	9,11,12,13,14	13,15	45
2	9,11,12,13,14	13,15	6

SITUAÇÃO 2

Conjunto de 15 medidas, com 6 medidas de fluxo nos ramos 1,2,8,12,14 e 15; 9 medidas de injeção nas barras 1,2,3,4,9,10,12,13 e 14.

Tabela 5.8 ⇒ Resultados do teste com rede de 14 barras (situação 2)

Métodos	Injeção crítica (barra)	Fluxo crítico (ramo)	Tempo de execução (seg.)
1	9,10,12,13,14	12,15	50
2	9,10,12,13,14	12,15	5

SITUAÇÃO 3

Conjunto de 15 medidas, com 7 medidas de fluxo nos ramos 1,2,8,11,13,14 e 15; 8 medidas de injeção nas barras 2,3,6,7,9,11,13 e 14.

Tabela 5.9 ⇒ Resultados do teste com rede de 14 barras (situação 3)

Métodos	Injeção crítica (barra)	Fluxo crítico (ramo)	Tempo de execução (seg.)
1	nenhuma	1	47
2	nenhuma	1	5

SITUAÇÃO 4

Conjunto de 15 medidas, com 7 medidas de fluxo nos ramos 1,2,8,12,14,15 e 18; 8 medidas de injeção nas barras 2,4,7,9,10,12,13 e 14.

Tabela 5.10 ⇒ Resultados do teste com rede de 14 barras (situação 4)

Métodos	Injeção crítica (barra)	Fluxo crítico (ramo)	Tempo de execução (seg.)
1	nenhuma	1	52
2	nenhuma	1	4

5.1.3 - Teste com o sistema de 121 barras da ELETROSUL

A topologia da rede de 121 barras da ELETROSUL é mostrada no Apêndice. Este sistema foi utilizado para teste, porque é um sistema real e já foi utilizado em outros trabalhos da área (veja QUINTANA et alii (1982)).

A rede é testada com um conjunto de 132 medidas, sendo 71 medidas de injeções ; 61 medidas de fluxos, que estão dispostas segundo a tabela 5.11; os resultados são apresentados na tabela 5.12.

Tabela 5.11 ⇒ Conjunto de medidas para o teste com a rede de 121 barras

Medidas de Injeção (barra)	1, 3, 5, 6, 8, 10, 12, 13, 14, 17, 22, 25, 28, 29, 31, 33, 35, 39, 40, 41, 42, 45, 48, 49, 50, 53, 55, 56, 57, 59, 60, 61, 62, 65, 66, 67, 68, 69, 70, 71, 73, 78, 79, 80, 82, 84, 86, 87, 88, 89, 90, 91, 92, 96, 97, 99, 100, 101, 102, 104, 106, 109, 110, 111, 112, 113, 114, 116, 117, 119, 120
Medidas de Fluxo (ramo)	(5-82), (2-4), (83-84), (85-86), (83-85), (13-14), (14-15), (80-118), (12-80), (11-12), (2-12), (5-11), (116-119), (105-119), (75-76), (15-16), (52-54), (51-54), (76-77), (77-110), (21-51), (107-108), (105-107), (108-110), (21-39), (21-38), (116-117), (82-116), (7-9), (35-64), (34-35), (35-58), (35-51), (72-74), (74-77), (66-75), (32-34), (30-31), (63-64), (39-40), (23-24), (42-43), (43-44), (45-46), (46-47), (19-20), (20-21), (18-19), (17-18), (21-22), (95-103), (103-110), (95-98), (93-95), (98-99), (27-93), (93-94), (26-27), (25-26), (81-94), (36-37)

Tabela 5.12 ⇒ Resultados do teste com rede de 121 barras

Métodos	Injeção crítica (barra)	Fluxo crítico (ramo)	Tempo de execução (seg.)
1	1,3,5,6,8,10,22,28,29 31,33,49,50,53,55,56 57,59,60,61,62,67,68 69,70,71,73,78,79,82 91,92,96,97,104,106 109,112,114,116,120	(83-84), (83-85), (75-76) (52-54), (51-54), (76-77) (7- 9), (72-74), (74-77) (32-34), (30-31), (63-64) (23-24), (43-44), (46-47) (21-22), (95-98), (98-99) (27-93), (26-27)	1140
2	1,3,5,6,8,10,22,28,29 31,33,49,50,53,55,56 57,59,60,61,62,67,68 69,70,71,73,78,79,82 91,92,96,97,104,106 109,112,114,116,120	(83-84), (83-85), (75-76) (52-54), (51-54), (76-77) (7- 9), (72-74), (74-77) (32-34), (30-31), (63-64) (23-24), (43-44), (46-47) (21-22), (95-98), (98-99) (27-93), (26-27)	19

5.2 - Análise dos resultados

Através dos resultados obtidos, verifica-se que os dois métodos dão sempre a resposta esperada. A diferença entre ambos está na velocidade de execução dos trabalhos, sendo que esta velocidade, no método da matriz Jacobiana, é maior que a decorrente do método com os caminhos de grafo e esta diferença aumenta, em se aumentando as dimensões da rede.

O método que identifica as medidas críticas, através dos caminhos de grafo requer o cálculo da matriz ganho, sua triangulação e construção dos caminhos de grafo, dependendo, em segundo lugar, do tamanho e da topologia da rede, sendo que a duração do trabalho de busca, para encontrar um caminho que ligue duas barras, pode tornar-se muito elevado. Essas são as razões por que o método que utiliza os caminhos de grafo exige maior tempo de execução, em relação ao que analisa a estrutura da matriz jacobiana. Importa salientar que na confecção dos programas correspondentes aos métodos propostos neste trabalho, não se utilizou nenhuma técnica de otimização, para aumentar a velocidade de execução desses métodos. Tais programas foram desenvolvidos com o objetivo principal de provar a veracidade dos métodos e para apreciação da velocidade de execução dos mesmos e da sua não complexidade de implantação. Destarte, resta ainda a possibilidade de aumentar as velocidades de execução dos métodos em pauta, com a utilização das técnicas de otimização.

6 - Conclusão

Para a realização do controle e operação dos sistemas elétricos de potência, é necessário conhecer os estados do sistema, o que é possível através dos estimadores de estado. Mas, para ser realizada a estimação de estado, é importante que o sistema seja observável. Entretanto, um sistema observável pode tornar-se não observável, quando um simples problema no sistema de medição causar a perda de uma medida. São as chamadas medidas críticas. Portanto, para se ter uma operação confiável dos sistemas elétricos de potência, é de vital importância que essas medidas sejam identificadas e depois transformadas em medidas redundantes, através do fortalecimento do conjunto de medidas, adicionando-lhe pseudo-medidas. Foram criados, então, neste trabalho, dois métodos para a identificação de medidas críticas. O primeiro utiliza como princípio a teoria desenvolvida por BRETAS (1996a), para análise de observabilidade, utilizando os caminhos de grafo para identificar as medidas críticas; o segundo identifica as medidas críticas analisando a estrutura da matriz jacobiana, baseando-se na teoria desenvolvida por SLUTSKER & SCUDDER (1987), para análise de observabilidade. Cumpre salientar que

estes métodos foram propostos, na tentativa de tornar o melhor possível o trabalho de identificação de medidas críticas.

Para verificar se os métodos propostos trazem realmente vantagens, em relação aos em vigor, foram recordados, neste trabalho, dois métodos existentes. O primeiro é um método combinatorial, que é de difícil implantação e que, por sua natureza combinatorial, está sujeito a uma explosão combinatorial, tornando-se muito lenta a sua aplicação. Já o segundo visa a identificar as medidas críticas através das matrizes unimodulares M e A . Este método é mais eficiente que o primeiro, mas necessita ainda de um algoritmo combinatorial para a sua viabilidade, pois os autores não apresentam uma fórmula analítica para obter o número total de árvores representativas de medidas, o qual se obtém através de uma busca que se realiza por um algoritmo combinatorial. Consequentemente se torna maior o tempo em que se obtêm os resultados, mormente para redes de dimensões elevadas. Assim, em razão de esses métodos serem métodos lentos, por assim o denominarmos, e de difícil implantação, é que foram propostos, neste trabalho, dois métodos novos.

Dos métodos propostos, o método que utiliza os caminhos de grafo necessita também de uma busca, cuja duração se torna, com a ajuda desses caminhos de grafo, bem menor que a realizada pelo algoritmo combinatorial, proposto por CLEMENTS et alii (1981), que, segundo estes mesmos autores, está sujeito a uma explosão combinatorial, mas é ainda, por assim dizermos, um método lento. Entretanto, utilizando-se como princípio o método de análise de observabilidade, através da redução

simbólica da matriz jacobiana, para a identificação das medidas críticas, chegamos a um método de execução mais rápida, que é o segundo método desenvolvido neste trabalho. Não é um método de natureza combinatorial e não requer nenhum cálculo, nem mesmo o relativo à matriz ganho, sendo, além disso, de fácil implantação.

Como as pesquisas que se desenvolvem na área em pauta visam sempre a metodologias mais simples, que sejam de fácil implantação e levem em conta o tempo de execução dos trabalhos, afigura-se-nos que o nosso objetivo foi alcançado, com a apresentação dos novos caminhos para a identificação de medidas críticas, delineados nos dois métodos propostos, que nos parece satisfazerem amplamente aos requisitos aludidos.

Anexo A

Serão apresentados, nos Anexos B e C, os fluxogramas e os programas correspondentes aos dois métodos propostos, mas, antes, é necessário mostrar como os dados são fornecidos aos programas e como os programas fornecem os resultados.

A topologia e o conjunto de medidas, referentes à rede que será testada, são fornecidos aos programas através de um arquivo de dados, o qual se apresenta da seguinte forma:

1 - Dados de barra:

XXXXbF

Sendo:

XXXX $\Rightarrow$ um número de 1 a 9998, que indica o número da barra;

b $\Rightarrow$ um espaço em branco;

F $\Rightarrow$ flag de caracter, que indica a presença de medida de injeção quando assumir o valor 1;

2 - Dados de linha:

XXXXXbFbXXXXX

Sendo:

XXXXX $\Rightarrow$ um número inteiro de 1 a 9998, que indica as barras ligadas por esse ramo;

b $\Rightarrow$ um espaço em branco;

F $\Rightarrow$ flag de caracter, que indica a presença de medidas de fluxo quando assumir o valor 1;

É mostrado, na figura 7.1, o arquivo de entrada de dados, relacionado ao sistema de 6 barras da IEEE representado na figura 5.1, da página 55, com o conjunto de medidas usado para o teste da situação 1 (veja página 55):

1	1	
2		
3		
4	1	
5	1	
6	1	
9999		
4		3
4		6
4	1	5
2		3
1		3
1	1	2
9999		

Figura 7.1 $\Rightarrow$ Arquivo de entrada de dados

Obs.:

número 9999, que aparece duas vezes no arquivo, indica o final dos dados de barra e o final dos dados de linha, respectivamente.

Através desse arquivo de entrada de dados, o programa recebe os dados necessários para o processamento e fornece o resultado mediante um arquivo de saída de dados, o qual classifica as medidas como críticas ou redundantes. O arquivo de saída de dados, relacionado ao arquivo de entrada, mostrado na figura 7.1, é o seguinte (figura 7.2).

A INJECAO NO NO 1 E CRITICA
A INJECAO NO NO 4 E CRITICA
A INJECAO NO NO 5 E REDUNDANTE
A INJEÇÃO NO NO 6 E CRITICA
O FLUXO NO RAMO 1 E CRITICO
O FLUXO NO RAMO 5 E REDUNDANTE
NUMERO TOTAL DE ITERAÇÕES = 1

Figura 7.2 ⇒ Arquivo de saída de dados

Anexo B

FLUXOGRAMA (FIGURA 7.3) E PROGRAMA, REFERENTES AO PRIMEIRO MÉTODO PROPOSTO, QUE PERMITE IDENTIFICAR AS MEDIDAS CRÍTICAS ATRAVÉS DOS CAMINHOS DE GRAFO.

Figura 7.3 ⇒ Fluxograma correspondente à metodologia que permite identificar as medidas críticas, através dos caminhos de grafo

Programa

```
#define NB 130
#define NL 300
#define NL2 600
#define NMED 200
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <malloc.h>
#include <sys/types.h>
#include <sys/times.h>
```

/* ESTRUTURA DE DADOS PARA OS PONTEIROS */

```
typedef struct jb{
int no;
int mi;
int mf;
int ltop;
struct jb *next;
}vetop;
```

/* ESTRUTURA PRA TOPOLOGIA REAL */

```
typedef struct jba{
int bad;
struct jba *next;
}realtop;
```

/* DECLARACAO DAS VARIAVEIS GLOBAIS */

```
char tipo[NB], /* INDICA SE HA MEDICAO DE INJECAO DE POTENCIA NA
                BARRA*/
l_red[NB],
tip[NL], /* INDICA SE HA MEDICAO DE FLUXO DE POTENCIA NA
          LINHA */
conec; /* INDICA SE HA' CONEXAO ENTRE AS AREAS */
int nb, /* NUMERO DE BARRAS */
nmed, /* NUMERO DE MEDIDAS */
noex[NB], /* NUMERO EXTERNO DA BARRA */
elim[NB],
vjun[NB],
no_ja[NB],
nl, /* NUMERO DE LINHAS */
ni[NL], /* NUM INTERNO DA BARRA INICIAL DA LINHA */
nf[NL], /* NUM INTERNO DA BARRA FINAL DA LINHA */
it; /* NUMERO DE ITERACOES */

vetop *vtop[NB];

realtop *rtop[NB];

float Bkk[NL], /* ELEMENTOS DA DIAGONAL PRINCIPAL DA
```

```

                                MATRIZ */
G[NB][NB];
H[NMED][NB];

FILE *banco;                /* APONTA PARA O ARQUIVO DE ENTRADA E O DE SAIDA */

/* VARIAVEIS RELATIVAS A TOPOLOGIA DA REDE */

int npil[NB],                /* INDICE INICIAL DOS VETORES NOLIG, LILIG, NEXT*/
nolig[NL2],                 /* NUMERO DA BARRA ADJACENTE */
lilig[NL2],                 /* NUMERO DA LINHA */
next[NL2];                  /* PROXIMO INDICE, SE ZERO INDICA FIM */

int ilha[NB], prox[NB];

/* INICIALIZACAO DOS VETORES */

void inicia()
{
    register int i;

    nb = nl = 0;

    for ( i=0 ; i<NB ; i++ ) {
        noex[i] = npil[i] = 0;
        vtop[i] = NULL;
        rtop[i] = NULL;
    }

    for ( i=0 ; i<NL ; i++ ) {
        ni[i] = nf[i] = 0;
    }

    for ( i=0 ; i<NL2 ; i++ ) {
        nolig[i] = lilig[i] = next[i] = 0.0;
    }
}

/* ROTINA DE LEITURA DOS DADOS DE ENTRADA */

void constr(str, b, e, snum) /* ESTA SUBROTINA COLOCA PARTE DA
                                STRING STR EM SNUM */

char str[80],                /* STRING QUE CONTEM A LINHA */
snum[];                     /* STRING COM A PARTE DESEJADA DA LINHA */
int b,                       /* INDICA A POSICAO INICIAL EM STR */
e;                           /* INDICA A POSICAO FINAL EM STR */
{
    int i, j,                /* VARIAVEIS UTILIZADAS NO LOOPING */
    branco;                  /* VERIFICA SE SNUM ESTA PREENCHIDA DE ESPACOS */

    if (b<strlen(str)){

```



```

    snum[0] = '\0';
    branco = 1;
    for(i=b,j=0;i<=e;i++,j++){
        snum[j] = str[i];
        if (str[i]!=' ') branco = 0;
    }
    snum[j]='\0';
    if (branco == 1) snum = "0";
}
}
void leitura()
{
    int i, /* CONTADOR DO LOOPING */
    flag1, flag2, /* VARIÁVEIS DE CONTROLE DE SAÍDA DO LOOPING */
    niex, nfex; /* NÚMERO EXTERNO DOS NO'S INICIAL E FINAL */
    char linha[80], /* STRING DE LEITURA DA LINHA DO BANCO DE DADOS */
    sno[5],
    sniex[5],
    snfex[5];

    /* LEITURA DOS DADOS DE BARRA */

    nb = nmed = 0;

    fgets(linha, 80, banco);

    /*constr(linha, 0, 3, sno);
    noex[1] = atoi(sno);
    noex[1] = atoi(linha);*/
    puts("Iniciando a leitura");
    while ((noex[++nb]=atoi(linha))!=9999) {

        tipo[nb] = linha[5];
        if (tipo[nb]=='1') nmed++;
        fgets(linha,80,banco);
    }

    nb--;

    /* LEITURA DOS DADOS DE LINHA */

    nl = 0;

    fgets(linha, 80, banco);

    constr(linha, 0, 3, sniex);
    niex = atoi(sniex);

    while (niex!=9999){

        nl++;

        tip[nl] = linha[5];
        switch (tip[nl]) {
            case '1':
            case '2': nmed++;
                    break;

```

```

        case '3': nmed = nmed + 2;
    }

    constr(linha, 8, 11, snfex);
    nfex = atoi(snfex);

    flag1 = flag2 = i = 1;
    while ((flag1) || (flag2)) {
        if ((niex == noex[i]) && (flag1)) {
            ni[nl] = i;
            Bkk[i] = Bkk[i] + 1;
            flag1 = 0;
        }

        if ((nfex == noex[i]) && (flag2)) {
            nf[nl] = i;
            Bkk[i] = Bkk[i] + 1;
            flag2 = 0;
        }
        i++;
    }

    fgets(linha, 80, banco);

    constr(linha, 0, 3, sniex);
    niex = atoi(sniex);
}

fclose(banco);
}

/* ESTA FUNCAO MONTA OS VETORES NPIL, NOLIG, LILIG, NEXT, OS
   QUAIS DESCREVEM A TOPOLOGIA DA REDE. */

void topologia_rede()
{
    int i, bi, bf, flag;
    realtop *tmp, *tmp1, *tmp2;

    for (i=1; i<=nl; i++){
        flag=2;
        while (flag){
            /* printf ("\n %d\ %d %d ", flag, ni[i], nf[i]); */

            if (flag==2){
                bi=ni[i];
                bf=nf[i];
            }
            if (flag==1){
                bi=nf[i];
                bf=ni[i];
            }
        }
    }
}

```

```

/* printf ("\n %d , %d",bi,bf);*/

tmp=(realtop *) malloc (sizeof (realtop));
tmp->bad=bf;
tmp->next=NULL;
if (!rtop[bi]) {
    rtop[bi]=tmp;
/* puts("elemento novo");*/
}
else{
/* puts("elemento antigo");*/
tmp1=rtop[bi];
if (tmp1->bad > tmp->bad){
tmp->next=rtop[bi];
rtop[bi]=tmp;
}
else{

while( (tmp1)&&(tmp1->bad < tmp->bad) ){
    printf (" elemento proximo %d \n",tmp1->bad);
    tmp2=tmp1;
    tmp1=tmp1->next;
}
if (tmp1){
tmp2->next=tmp;
tmp->next=tmp1;
}
else{
tmp2->next=tmp;
tmp->next=NULL;
}
}
}
}
flag--;
}
}
}

```

```

void mostra_top()
{
    int i;
    realtop *tmp;
    fprintf (banco,"\nTOPOLOGIA REAL\n");
    for (i=1;i<=nb;i++){
        tmp=rtop[i];
        fprintf (banco,"%d ->",i);
        while(tmp){
            fprintf(banco," %d, ",tmp->bad);
            tmp=tmp->next;
        }
        fprintf (banco,"\n");
    }
}

```

```

void top_rede()
{
    int aux,
        i,

```

```
in,
ina,
k,
l,
n4,
n5,
lff = 0;
char flag,
kstop,
prim;

lff = 1;

for (i=1;i<=nb;i++) {
    npil[i] = 0;
}
for (i=1;i<=2*nl;i++) {
    next[i] = 0;
    nolog[i] = 0;
    lilig[i] = 0;
}
for (i=1;i<=nl;i++) {
    prim = '1';
    k = ni[i];
    l = nf[i];
    kstop = '0';
    do {
        if (kstop == '0' && prim != '1') {
            aux = k;
            k = l;
            l = aux;
            kstop = '1';
        }
        prim = '0';
        ina = 0;
        in = npil[k];
        if (in == 0) {
            npil[k] = lff;
        }
        else {
            n5 = next[l];
            n4 = next[nolog[in]];
            flag = '0';
            while (n5 > n4 && flag != '1') {
                ina = in;
                in = next[in];
                if (in == 0) {
                    next[lff] = in;
                    next[ina] = lff;
                    flag = '1';
                }
            }
            else {
                n4 = next[nolog[in]];
            }
        }
        if (flag != '1') {
            if (ina == 0) {
```



```

        next[lff] = in;
        npil[k] = lff;
    }
    else {
        next[lff] = in;
        next[ina] = lff;
    }
}
}
nolig[lff] = l;
lilig[lff] = i;
lff++;
}
while (kstop == '0');
}
}

/***** ROTINA QUE MONTA AS MATRIZES H E G *****/

void monta_H_Ht()
{
    int i, j, k, ind, lin;
    float temp1 ,temp2;
    puts ("Iniciando calculo de h_ht");
    for (i=0;i<NMED;i++)
        for(j=0;j<NB;j++)
            H[i][j] = 0.0;

    lin = 0;
    for (i=1;i<=nl;i++) {
        if (tip[i]=='1') {
            lin++;
            H[lin][ni[i]] = 1.0;
            H[lin][nf[i]] = -1.0;
        }
        if (tip[i]=='2') {
            lin++;
            H[lin][ni[i]] = -1.0;
            H[lin][nf[i]] = 1.0;
        }
        if (tip[i]=='3') {
            lin++;
            H[lin][ni[i]] = 1.0;
            H[lin][nf[i]] = -1.0;
            lin++;
            H[lin][ni[i]] = -1.0;
            H[lin][nf[i]] = 1.0;
        }
    }
    for (i=1;i<=nb;i++) {
        if (tipo[i]=='1') {
            lin++;
            ind = npil[i];
            while (ind!=0) {
                H[lin][nolig[ind]] = -2.0;
                ind = next[ind];
            }
            H[lin][i] = 2*Bkk[i];
        }
    }
}

```

```

for (i=1;i<=nb;i++)
    for (j=1;j<=nb;j++) {
        G[i][j] = 0.0;
        for (k=1;k<=nmed;k++) {
            temp1=H[k][i];
            temp2=H[k][j];
            G[i][j] += temp1*temp2;
        }
    }
}

```

/* METODO DE GAUSS */

```

void gauss()
{
    int i, j, k;
    double m, pivo;
    vetop *tmp, *tmp1;
    for (k=1; k<=nb; k++) {
        if (fabs(G[k][k])>1.0e-5) {
            m = 1.0/G[k][k];
            for (j=k; j<=nb; j++) G[k][j] *= m;
            for (i=k+1; i<=nb; i++) {
                pivo = G[i][k];
                for (j=k; j<=nb; j++) {
                    G[i][j] -= (pivo*G[k][j]);
                    if (fabs(G[i][j])<1.0e-5) G[i][j] = 0.0;
                }
            }
        }
        else G[k][k] = 0.0;
    }
}

```

/***** IMPRIMIR A MATRIZ G TORNADA TRIANGULAR *****/

```

for (i=1;i<=nb;i++){
    for (j=1;j<=nb;j++){
        fprintf(banco," %3.2f ",G[i][j]);
    }
    fprintf (banco,"\n");
}
for (i=1;i<=nb;i++){
    for (j=i+1;j<=nb;j++){
        if ((G[i][j]) && (i!=j)){
            tmp=(vetop *)malloc (sizeof (vetop));
            tmp->no=j;
            tmp->mf=0;
            tmp->next=NULL;
            if (tipo[j]=='1')
                tmp->mi=1;
            else
                tmp->mi=0;
            if (vtop[i]){
                tmp1=vtop[i];
                while (tmp1->next)
                    tmp1=tmp1->next;
            }
        }
    }
}

```

```

        tmp1->next=tmp;
    }else
        vtop[i]=tmp;
    tmp=(vetop *)malloc (sizeof (vetop));
    tmp->no=i;
    tmp->mf=0;
    tmp->next=NULL;
    if (tipo[i]=='1')
        tmp->mi=1;
    else
        tmp->mi=0;
    if (vtop[j]){
        tmp1=vtop[j];
        while (tmp1->next)
            tmp1=tmp1->next;
        tmp1->next=tmp;
    }else
        vtop[j]=tmp;
    }
}
for (i=1;i<=nl;i++){
    if (tip[i]=='1'){
        tmp=vtop[ni[i]];
        while ((tmp) && (tmp->no!=nf[i]))
            tmp=tmp->next;
        if (tmp)
            tmp->mf=1;
        tmp=vtop[nf[i]];
        while ((tmp) && (tmp->no!=ni[i]))
            tmp=tmp->next;
        if (tmp)
            tmp->mf=1;
    }
}
for (i=1;i<=nb;i++){
    tmp=vtop[i];
    while (tmp){
        fprintf (banco," ( %d , %d , %d , %d )",i,tmp->no,tmp->mi,tmp->mf);
        tmp=tmp->next;
    }
    fprintf (banco,"\n");
}
}

/* ROTINA PARA VERIFICAR A TOPOLOGIA DA REDE */

void verificar_top()
{
    int i;
    vetop *tmp;
    reatop *tmp1;

    for (i=1;i<=nb;i++){
        tmp=vtop[i];
        tmp1=rtop[i];
/*      puts ("entrei");          */
        while(tmp){

```

```

    while ((tmp1) && (tmp->no != tmp1->bad)){
/*      printf ("\nabriu,%d ,%d",tmp->no, tmp1->bad);*/
      tmp1=tmp->next;
    }
    if ((tmp1) && (tmp->no == tmp1->bad)){
      tmp->ltop=1;
    }
    if ((tmp1) && (tmp->no != tmp1->bad)){
      tmp->ltop=0;
    }
    tmp1=rtop[i];
    tmp=tmp->next;
  }
}
fprintf (banco,"\n COMPLETO\n");
for (i=1;i<=nb;i++){
  tmp=vtop[i];
  while (tmp){
    fprintf (banco," ( %d , %d ,%d , %d ,%d )",i,tmp->no,tmp->mi,tmp->mf,tmp->ltop);
    tmp=tmp->next;
  }
  fprintf (banco,"\n");
}
/* for (i=1;i<=nb;i++){
  vtop[i]=vtopf[i];
}*/
}

          /***** ROTINA PRA VERIFICAR INJECoes CRITICAS *****/

int proc_adj(no,adj)
int no;
int adj;
{
  int a=1;
  vetop *tmp,*tmp1;
  tmp=vtop[no];
  puts ("entrei sub proc_adj");
  while ((a) && (tmp)){
/*    if (tmp->no==adj)
      tmp=tmp->next;*/
    printf ("entrei wh no=%d,barra 1=%d,mi=%d,mf=%d\n",no,tmp->no,tmp->mi,tmp->mf);
    if ((tmp) && ((tmp->mi==1) || (tmp->mf==1)) && (tmp->ltop==1)){
      printf ("\nentrei if barra proc=%d,mi=%d,mf=%d,badj=%d\n",tmp->no,tmp->mi,tmp-
>mf,adj);
      tmp1=vtop[tmp->no];
      while (tmp1){
        if ((tmp1->no ==adj) && ((tmp1->mi==1) || (tmp1->mf==1)) && (tmp1->ltop==1))
break;
        printf ("\nNIVEL=%d,NO=%d",tmp->no,tmp1->no);
        tmp1=tmp1->next;
      }
      if (tmp1) a=0;
      else printf ("\nsai do loop\n");
    }
    printf ("\nANTES NEXT TMP\n");
    tmp1=vtop[4];
    while (tmp1){

```



```

printf ("\nnos_adj_4=%d\n",tmp1->no);
tmp1=tmp1->next;
}
tmp=tmp->next;
}
if (!a) puts("achei a");
return (!a);
}

int proc(no,adj,mi)
int no,mi;
int adj;
{
int a=1,x,y,z,b;
vetop *tmp,*tmp1;

tmp=vtop[no];
puts ("entrei sub proc");
while ((a) && (tmp)){
/*if (tmp->no==adj)
tmp=tmp->next;*/
printf ("entrei wh no=%d,barra 1=%d,mi=%d,mf=%d,injno=%d,ltop=%d\n",no,tmp-
>no,tmp->mi,tmp->mf,mi,tmp->ltop);
if (tmp->ltop==1){
if (mi==1){
puts ("x=1");
x=1;
}else x=0;
if(tmp->mi==1){
puts ("y=1");
y=1;
}else y=0;
if (tmp->mf==1){
puts ("z=1");
z=1;
}else z=0;
b=1;
if((x) || (y) || (z)){
b=1;
tmp1=vtop[tmp->no];
while((tmp1) && (b)){
printf("\nENTREINOVOno=%d
mi(9)=%d,mf(9)=%d,adj=%d,inj_no=%d,mia=%d,mfa=%d\n",tmp1->no,tmp1->mi,tmp1-
>mf,adj,mi,tmp->mi,tmp->mf);
if (((tmp1->ltop==1) && (tmp1->no==adj)){
if(((x==1) && (y==0) && (z==0)) || ((x==1) && (y==0) && (z==1)) || ((x==0) &&
(y==1) && (z==0)) || ((x==0) && (y==0) && (z==1))){
puts ("ENTREI NO 1 IF");
printf ("\nX=%d,Y=%d,Z=%d\n",mi,tmp->mi,tmp->mf);
if (tmp1->mf==1){
puts("ENTREI NO 1 IF, IF");
b=0;
}puts("SAIII DO 1IF,IF");
}
if(((x==1) && (y==1) && (z==1)) || ((x==1) && (y==1) && (z==0)) || ((x==0) &&
(y==1) && (z==1))){
puts ("ENTREI NO 2 IF");
printf ("\nX=%d,Y=%d,Z=%d\n",mi,tmp->mi,tmp->mf);
if ((tmp1->mf==1) || (y==1)){

```

```

        puts("ENTREI NO 2 IF, IF");
        b=0;
    }
    }
    }printf("\nSAIIII DO IF MAIOR , B=%d\n",b);
    tmp1=tmp1->next;
    }printf ("\nSAIIII DO WHILE , B=%d\n",b);
    }
    if (b==0){printf ("\nENTREI NO IF FINAL B=%d\n",b);
        a=0;
        printf ("\nACHEI A=%d ",a);
    }
    }
    tmp=tmp->next;
}
if (!a) puts("achei a");
return (!a);
}

```

/ SUBROTINA JUNÇÃO (reduz a dimensão da rede eliminando os grupos observáveis) */*

```

void junc(no,inj)
int no;
char inj;
{
    int cont,aloc1=0,aloc2=0,aloc3=0,aloc4=0,aloc5=0,i;
    vetop *tmp,*tmp1,*tmp2,*tmp3,*tmp4,*tmp5;
    printf ("\nENTREI NA SUB JUNCAO\n");

    printf ("\nNO=%d,inj=%d\n",no,inj);
    tmp=vetop[no];
    cont=1;
    while(tmp){
        if((tmp->ltop==1) && ((tmp->mi==1) || (tmp->mf==1) || (inj=='1'))){
            if (cont==1){
                aloc1=tmp->no;
            }
            if (cont==2){
                aloc2=tmp->no;
            }
            if (cont==3){
                aloc3=tmp->no;
            }
            if (cont==4){
                aloc4=tmp->no;
            }
            if (cont==5){
                aloc5=tmp->no;
            }
            cont++;
        }
        tmp=tmp->next;
    }
}

```

```
printf
("\nALOC1=%d,ALOC2=%d,ALOC3=%d,ALOC4=%d,ALOC5=%d,CONT=%d\n",aloc1,aloc2
,aloc3,aloc4,aloc5,cont);
if(cont==3){
    tmp1=vtop[aloc1];
    while(tmp1){
        if (tmp1->no==aloc2){
            tmp1->mf=1;
            tmp1->ltop=1;
        }
        tmp1=tmp1->next;
    }
    tmp2=vtop[aloc2];
    while(tmp2){
        if(tmp2->no==aloc1){
            tmp2->mf=1;
            tmp2->ltop=1;
        }
        tmp2=tmp2->next;
    }
}
if (cont==4){
    tmp1=vtop[aloc1];
    while(tmp1){
        if (tmp1->no==aloc2){
            tmp1->mf=1;
            tmp1->ltop=1;
        }
        if (tmp1->no==aloc3){
            tmp1->mf=1;
            tmp1->ltop=1;
        }
        if (tmp1->no==aloc4){
            tmp1->mf=1;
            tmp1->ltop=1;
        }
        tmp1=tmp1->next;
    }
    tmp2=vtop[aloc2];
    while(tmp2){
        if (tmp2->no==aloc1){
            tmp2->mf=1;
            tmp2->ltop=1;
        }
        if (tmp2->no==aloc3){
            tmp2->mf=1;
            tmp2->ltop=1;
        }
        if (tmp2->no==aloc4){
            tmp2->mf=1;
            tmp2->ltop=1;
        }
        tmp2=tmp2->next;
    }
    tmp3=vtop[aloc3];
    while(tmp3){
        if (tmp3->no==aloc1){
            tmp3->mf=1;
        }
    }
}
```

```
    tmp3->ltop=1;
}
if (tmp3->no==aloc2){
    tmp3->mf=1;
    tmp3->ltop=1;
}
if (tmp3->no==aloc4){
    tmp3->mf=1;
    tmp3->ltop=1;
}
tmp3=tmp3->next;
}
tmp4=vtop[aloc4];
while(tmp4){
    if(tmp4->no==aloc1){
        tmp4->mf=1;
        tmp4->ltop=1;
    }
    if (tmp4->no==aloc2){
        tmp4->mf=1;
        tmp4->ltop=1;
    }
    if (tmp4->no==aloc3){
        tmp4->mf=1;
        tmp4->ltop=1;
    }
    tmp4=tmp4->next;
}
}
if (cont==5){
    tmp1=vtop[aloc1];
    while(tmp1){
        if (tmp1->no==aloc2){
            tmp1->mf=1;
            tmp1->ltop=1;
        }
        if (tmp1->no==aloc3){
            tmp1->mf=1;
            tmp1->ltop=1;
        }
        if (tmp1->no==aloc4){
            tmp1->mf=1;
            tmp1->ltop=1;
        }
        if (tmp1->no==aloc5){
            tmp1->mf=1;
            tmp1->ltop=1;
        }
        tmp1=tmp1->next;
    }
    tmp2=vtop[aloc2];
    while(tmp2){
        if (tmp2->no==aloc1){
            tmp2->mf=1;
            tmp2->ltop=1;
        }
        if (tmp2->no==aloc3){
            tmp2->mf=1;
        }
    }
}
```



```
    tmp2->ltop=1;
}
if (tmp2->no==aloc4){
    tmp2->mf=1;
    tmp2->ltop=1;
}
if (tmp2->no==aloc5){
    tmp2->mf=1;
    tmp2->ltop=1;
}
tmp2=tmp2->next;
}
tmp3=vtop[aloc3];
while(tmp3){
    if (tmp3->no==aloc1){
        tmp3->mf=1;
        tmp3->ltop=1;
    }
    if (tmp3->no==aloc2){
        tmp3->mf=1;
        tmp3->ltop=1;
    }
    if (tmp3->no==aloc4){
        tmp3->mf=1;
        tmp3->ltop=1;
    }
    if (tmp3->no==aloc5){
        tmp3->mf=1;
        tmp3->ltop=1;
    }
    tmp3=tmp3->next;
}
tmp4=vtop[aloc4];
while(tmp4){
    if (tmp4->no==aloc1){
        tmp4->mf=1;
        tmp4->ltop=1;
    }
    if (tmp4->no==aloc2){
        tmp4->mf=1;
        tmp4->ltop=1;
    }
    if (tmp4->no==aloc3){
        tmp4->mf=1;
        tmp4->ltop=1;
    }
    if (tmp4->no==aloc5){
        tmp4->mf=1;
        tmp4->ltop=1;
    }
    tmp4=tmp4->next;
}
tmp5=vtop[aloc5];
while(tmp5){
    if (tmp5->no==aloc1){
        tmp5->mf=1;
        tmp5->ltop=1;
    }
}
```

```

    if (tmp5->no==aloc2){
        tmp5->mf=1;
        tmp5->ltop=1;
    }
    if (tmp5->no==aloc3){
        tmp5->mf=1;
        tmp5->ltop=1;
    }
    if (tmp5->no==aloc4){
        tmp5->mf=1;
        tmp5->ltop=1;
    }
    tmp5=tmp5->next;
}
}
for (i=1;i<=nb;i++){
    tmp=vtop[i];
    while(tmp){
        printf    ("\nNO=%d,mi=%d,mf=%d,ltop=%d,no_eli=%d",tmp->no,tmp->mi,tmp->mf,tmp-
>ltop,i);
        tmp=tmp->next;
    }
}
}

```

/***** ROTINA PARA TESTAR BARRAS PARA A ELIMINAÇÃO *****/

```

int t_el(bar,mi)
int bar;
char mi;
{
    int i,w=1;
    vetop *tmp6, *tmp5;
    printf ("\n ENTREI SUB T_ELI NO=%d",bar);
    /*tmp5=vtop[9];
    while (tmp5){
        printf    ("\n9->(noadj=%d,mi=%d,mf=%d,ltop=%d)\n",tmp5->no,tmp5->mi,tmp5-
>mf,tmp5->ltop);
        tmp5=tmp5->next;
    } */
    tmp6=vtop[bar];
    if ((mi=='1') && (no_ja[bar]==0)) w=0;
    if (w==1){
        while ((tmp6) && (w)){printf    ("\nno=%d,ltop=%d,mi=%d\n",tmp6->no,tmp6->ltop,tmp6-
>mi);
        if (tmp6->ltop==1){printf    ("\nno=%d,ltop=%d,mi=%d\n",tmp6->no,tmp6->ltop,tmp6-
>mi);
        if (tmp6->mi==1){
            if(((bar < tmp6->no) && (no_ja[tmp6->no]==0)) || (no_ja[tmp6->no]==0))
                w=0;
            printf ("\n(1)TEM MEDIDA DE INJECao NO ADJ=%d,W=%d\n",tmp6->no,w);
        }else{
            /* if (bar > tmp6->no) w=0;*/
            printf("\n(1)NAO TEM MEDIDA DE INJ NO ADJ=%d,w=%d\n",tmp6->no,w);
        }
        } puts ("\n\n SAIIIIIIIII DO IF");
        tmp6=tmp6->next;
    }
}

```

```

    }/*if*/
    return(w);
}
void p_flux(noinj,noadj)
    int noinj;
    int noadj;
{
    int i;
    vetop *tmp;

    for (i=1;i<=nb;i++){
        tmp=vtop[i];
        if (i==noinj){
            puts ("prim caso");
            l_red[i]=0;
            while (tmp->no!=noadj)
                tmp=tmp->next;
            tmp->mf=1;
        }else{
            while ((tmp) && (tmp->no<noinj))
                tmp=tmp->next;
            if ((tmp) && (tmp->no==noinj)){
                tmp->mi=0;
                if (i==noadj) tmp->mf=1;
                printf ("\nramo elim=%d",i);
            }
        }
    }
}

int procli_adj(no,adj,ti)
    int no;
    int adj;
    int ti;
{
    int a=1,b,x,y,z,x1,y1,x2,y2,y3,x1p3,xp3;
    int noi,nos;
    vetop *tmp,*tmp1,*tmp2,*tmp3;
    noi=no;
    nos=adj;
    tmp=vtop[no];
    puts ("entrei sub procli_adj");
    while ((a) && (tmp)){
        /* if (tmp->no==adj)
            tmp=tmp->next;*/
        printf ("entrei wh no=%d,barra 1=%d,mi=%d,mf=%d,injno=%d,ltop=%d\n",no,tmp->no,tmp->mi,tmp->mf,ti,tmp->ltop);
        printf ("\nx=%d,inj_no=%d\n",tmp->mi,ti);
        if ((tmp->ltop==1) && (tmp->no!=adj)){ /* tmp->no!=adj*/
            printf ("\nx=%d\n",tmp->mi);
            if (tmp->mi==1){
                puts ("x=1");
                x=1;
            }else {
                x=0;
                puts ("x=0");
            }
        }
        if(tmp->mf==1){

```

```

    puts ("y=1");
    y=1;
} else y=0;
if (ti==1){
    puts ("z=1");
    z=1;
} else z=0;
b=1;
if ((x) || (y) || (z)){
    /*b=1;*/
    tmp1=vtop[tmp->no];
    while((tmp1) && (b)){
        printf          ("\nENTREI          NOVO          no=%d
mi(9)=%d,mf(9)=%d,adj=%d,inj_no=%d,mia=%d,mfa=%d\n",tmp1->no,tmp1->mi,tmp1-
>mf,adj,ti,tmp->mi,tmp->mf);

        if((tmp1->ltop==1) && (tmp1->no==adj)){
            printf ("\n ANTES X=%d,Y=%d,Z=%d\n",tmp->mi,tmp->mf,ti);
            if(((z==1) && (x==1) && (y==1)) || ((z==1) && (x==1) && (y==0)) || ((z==0) &&
(x==1) && (y==1))){
                puts ("ENTREI NO 1 IF");
                printf ("\nX=%d,Y=%d,Z=%d\n",tmp->mi,tmp->mf,ti);
                if((tmp1->mf==1) || (x==1)){
                    puts("ENTREI NO 1 IF, IF");
                    b=0;
                } else{
                    puts("SAIIII DO 1IF,IF");
                    noi=tmp->no;
                }
            }
            if(((z==0) && (x==0) && (y==1)) || ((z==1) && (x==0) && (y==0)) || ((z==1) &&
(x==0) && (y==1)) || ((z==0) && (x==1) && (y==0))){
                puts ("ENTREI NO 2 IF");
                if ((tmp1->mf==1)){
                    puts("ENTREI NO 2 IF, IF");
                    b=0;
                } else{
                    printf("\nENTREI NO 2 IF MAIS NAO NO 2 IF, IF");

                    noi=tmp->no;
                }
            }
        }
        printf("\nSAIIII DO IF MAIOR , B=%d\n",b);
        if (((x==1) && (y==1) && (z==1)) || ((x==1) && (y==1) && (z==0)) || ((x==1) && (y==0)
&& (z==1))){
            xp3=1;
            puts ("xp3=1, pra entrar no 3 nivel");
        } else{
            puts ("xp3=0, pra entrar no 3 nivel");
            xp3=0;
        }
        printf ("\nB=%d,tmp1->no=%d ENTRE NO 3 NIVEL",b,tmp1->no);
        if ((b==1) && (tmp1->no!=adj) && (tmp1->no!=no)){
            if (tmp1->ltop==1){
                if ((tmp1->mi==1) && (xp3==1)){
                    puts ("x1=1");
                    x1=1;
                } else x1=0;
            }
        }
    }
}

```



```

        if(tmp1->mf==1){
            puts ("y1=1");
            y1=1;
        }else y1=0;
        /* if ((tmp->mi==1){
            puts ("z1=1");
            z1=1;
        }else z1=0;*/
        if ((x1) || (y1) || (xp3)){
            tmp2=vtop[tmp1->no];
            while((tmp2) && (b)){
                printf ("\nENTREI BAIXO no=%d,mi=%d,mf=%d,adj=%d,inj_no=%d\n",tmp2-
>no,tmp2->mi,tmp2->mf,adj,tmp1->mi);
                if (((tmp1->mi==1) && ((xp3==1) || (y1==1)))){
                    x1p3=1;
                    puts ("x1p3=1, dentro do 3 nivel");
                }else{
                    x1p3=0;
                    puts("x1p3=0, dentro do 3 nivel");
                }
                if((tmp2->ltop==1) && (tmp2->no==adj)){
                    printf ("\n ANTES X1=%d,Y1=%d,Z1=%d\n",tmp1->mi,tmp1->mf,tmp->mi);
                    if ((x1p3==1) || (tmp2->mf==1)){
                        puts ("entrei 1 if baixo bvai ser 0");
                        b=0;
                    }else{
                        noi=tmp1->no;
                    }
                }

                }printf("\nSAIIII DO IF MAIOR BAIXO , B=%d\n",b);

        if ((b==1) && (tmp2->noi!=adj) && (tmp2->noi!=tmp->no) && (tmp1->no!=no)){
            if (tmp2->ltop==1){

                if ((x1p3==1) || (tmp2->mf==1)){
                    puts ("x2=1");
                    x2=1;
                }else{
                    x2=0;
                    puts ("x2=0");
                }
            }

            /* if (tmp3->mf==1){
                y3=1;
                puts("y3=1");
            }else{
                puts ("y3=0");
                y3=0;
            } */
            puts ("antes do testes de x1p3 y2 x2");
            if ((x1p3) || (y2) || (x2)){
                tmp3=vtop[tmp2->no];
                while((tmp3) && (b)){
                    printf ("\nENTREI 4 NIVEL no=%d,mi=%d,mf=%d,adj=%d,inj_no=%d\n",tmp3-
>no,tmp3->mi,tmp3->mf,adj,tmp2->mi);
                    if((tmp3->ltop==1) && (tmp3->no==adj)){
                        printf ("\n ANTES 4 NIVEL X2=%d,Y2=%d,Z2=%d,tmp3->mf=%d\n",x2,tmp2-
>mf,tmp1->mi,tmp3->mf);

```

```

if ((x2==1) || (tmp3->mf==1)){
    puts (" ENCONTREI NO NIVEL 4 ");
    b=0;
} else{
    noi=tmp2->no;
}
} printf("\nSAIIII DO IF MAIOR 4 NIVEL , B=%d ,tmp3->mf=%d\n",b,tmp3->mf);
    tmp3=tmp3->next;
    /* fim wh tmp3*/
    /* fim do if x1,y1,z1 */
    /* fim if ltop */
} /* fim do if b*/
    tmp2=tmp2->next;
    /* fim wh tmp2*/
    /* fim do if x1,y1,z1 */
    /* fim if ltop */
} /* fim do if b*/
    tmp1=tmp1->next;
} printf ("\nSAIIII DO WHILE , B=%d\n",b);
}
if (b==0){printf ("\nENTREI NO IF FINAL B=%d\n",b);
    a=0;
    printf ("\nACHEI A=%d ",a);
    puts ("saii do if x'y'z");
}
tmp=tmp->next;
}
if (a==1){
    printf ("\n\n\n      CHAMAR P_FLUX NOS=%d,NOI=%d  \n\n\n",nos,noi);
    /*if (noi!=ladj)*/
    p_flux(nos,noi);
}
printf("\n\n\n\n\n a=%d\n\n\n\n\n",a);
if (!a) puts("achei a");
return (!a);
}
void elimi_b(be)
int be;
{
    int i;
    int b=0;
    vetop *tmp, *tmp1, *tmp2, *tmp5;
    puts ("entrei sub eliminar barra");
    tmp=vetop[be];
    for (i=1;i<=nb;i++){
        tmp5=vetop[i];
        printf ("\nentrei for,binj=%d,be=%d\n",i,be);
        tmp1=tmp5;
        printf ("\nconferindo: binj=%d,be=%d,no=%d\n",i,be,tmp1->no);
        if (tmp1->no!=be){
            b=1;
            while((tmp1) && (tmp1->no!=be)){
                tmp1=tmp1->next;
            }
        }
        if ((tmp1) && (tmp1->no==be) && (b==0)){
            printf ("\nAPAGANDO NO=%d\n",tmp1->no);

```

```

    tmp1->no=0;
    tmp1->mi=0;
    tmp1->mf=0;
    tmp1->ltop=0;
    free(tmp1);*/
}
if ((tmp1) && (tmp1->no==be) && (b==1)){
    printf ("SAI W,no=%d,binj=%d",tmp1->no,i);
    tmp2=tmp5;
    printf ("\n TESTE TMP2,no=%d",tmp2->no);
    while ((tmp2) && (tmp2->next!=tmp1))
        tmp2=tmp2->next;
    if (tmp2)
        tmp2->next=tmp1->next;
    free(tmp1);
    tmp1->no=0;
    tmp1->mi=0;
    tmp1->mf=0;
    tmp1->ltop=0;
}
/* */
}
}

for (i=1;i<=nb;i++){
    if (tipo[i]=='1'){
        puts ("entrei 1");
        no_ja[i]=1;
        tmp=vtop[i];
        while (tmp){
            puts ("entrei 2");
            printf ("\ninj=%d,no=%d,  mi=%d,  mf=%d,lig.topol=%d\n",i,tmp->no,tmp->mi,tmp-
>mf,tmp->ltop);
            if (tmp->no!=0){
                if (tmp->ltop==1){
                    printf ("entrei 3,ltop=%d\n",tmp->ltop);
                    if ((tmp->mi) || (tmp->mf)){
                        printf ("entrei 4,mi=%d, mf=%d\n",tmp->mi,tmp->mf);
                    }else{
                        if (!procli_adj(tmp->no,i,tmp->mi)){
                            fprintf (banco,"\n\nA INJECAO %d E CRITICA(1)\n\n",i);
                            /* p_flux(i,noi);*/
                            tmp4=vtop[i];
                            while (tmp4){
                                printf ("\nNOS do %d no=%d,mi=%d,mf=%d,ltop=%d\n",i,tmp4->no,tmp4-
>mi,tmp4->mf,tmp4->ltop);
                                tmp4=tmp4->next;
                            }
                        }
                        /* if (i==4){
                            tmp4=vtop[i];
                            while (tmp4){
                                printf ("\nNOS ADJ A 4=%d\n",tmp4->no);
                                tmp4=tmp4->next;
                            }
                        }*/
                        break;
                    }
                }else{

```

```

    printf ("\nentrei 4,ltop=%d,no=%d\n",tmp->ltop,tmp->no);
    /* for (l=1;l<=nb;l++){
        tmp5=vtop[l];
        while (tmp5){
            printf (" \n%d->(noadj=%d,mi=%d,mf=%d,ltop=%d)\n",l,tmp5->no,tmp5->mi,tmp5->mf,tmp5->ltop);
            tmp5=tmp5->next;
        }
    } */
    if (lprocli_adj(tmp->no,i,tmp->mi)){
        fprintf(banco," \n \n A INJECAO %d E CRITICA(2)\n \n",i);
        printf (" \n \n O NOI E %d",noi);
        /* p_flux(i,noi); */
        tmp4=vtop[i];
        while (tmp4){
            printf (" \n NOS do %d no=%d,mi=%d,mf=%d,ltop=%d\n",i,tmp4->no,tmp4->mi,tmp4->mf,tmp4->ltop);
            tmp4=tmp4->next;
        }
        break;
        /*tmp->next=NULL; */
    }
}
/* tmp4=vtop[5];
while (tmp4){
    printf (" \n ANTES DO NEXT: NOS ADJ A 5=%d\n",tmp4->no);
    tmp4=tmp4->next;
} */
tmp=tmp->next;
}
/* no_ja[i]=1; */
printf (" \n E HORA NOINJ=%d\n",i);

/* if (t_el(i,tipos[i]){
    printf(" \n E P/ ELIMINAR O NO %d\n",i);
    elimi_b(i);
    elim[i]=1;
}
else
    printf(" \n NAO E P/ ELIMINAR O NO (1) %d\n",i);

}else{
    printf (" \n TESTE PRA BARRAS SEM INJ");
    if (t_el(i,tipos[i]){
        printf (" \n E P/ ELIMINAR O NO SEM INJ %d\n",i);
        elimi_b(i);
        elim[i]=1;
    }
    else
        printf (" \n NAO E P/ ELIMINAR O NO SEM INJ %d\n",i); */
}
/* no_ja[i]=1; */
if (i==1){
    for (n=1;n<nb;n++){
        if (elim[n]==0){printf (" \n ELIM[%d]=0\n",n);
            if (t_el(n,tipos[n])){
                printf (" \n E PRA ELIMINAR O NO %d\n",n);

```



```

        elim[n]=1;
        elimi_b(n);
/*      elim[n]=1;*/
    }else{
        printf ("\nNAO E PRA ELIMINAR O NO (1) %d\n",n);
    }
    }else printf ("\n nao entrou pra eliminar qualquer um,no=%d\n",n);
    }/* do for */
    }/* do if maior */
puts (" SAI DO 1 ROTINA");
if ((il==nb) && (il==1)){puts ("ENTREI ROTINA NOVA 2");
    p=i;
    printf ("\n esta na analise de numero %d\n, elim[%d]=%d",p,p,elim[p]);
    if (elim[p]==0){ printf("\nELIM[%d]=0\n",p);
        if (t_el(p,tipo[p])){
            printf ("\nE PRA ELIMINAR O NO %d\n",p);
            elim[p]=1;
            elimi_b(p);
/*      elim[p]=1;*/
        }else{ /* do 3 if */
            printf ("\nNAO E PRA ELIMINAR O NO (2) %d\n",p);
        }
    }else printf ("\n nao entrou pra eliminar qualquer um,no=%d\n",p);
/* do 2 if */
    tmp7=vtop[i];
    while (tmp7){
        if ((tmp7->ltop==1) && (tmp7->no!=nb)){
            if (elim[tmp7->no]==0){printf ("\nELIM[%d]=0\n",tmp7->no);
                if (t_el(tmp7->no,tipo[tmp7->no])){
                    printf ("\nE PRA ELIMINAR O NO %d (teste)\n",tmp7->no);
                    elim[tmp7->no]=1;
                    elimi_b(tmp7->no);
/*      elim[tmp7->no]=1;*/
                    printf("\n elim[%d]=%d\n",tmp7->no,elim[tmp7->no]);
                }else{ /* do 3 if */
                    printf ("\nNAO E PRA ELIMINAR O NO (3) %d\n",tmp7->no);
                }
            }else printf ("\n nao entrou pra eliminar qualquer um,no=%d\n",tmp7->no);
/* do 2 if */
        } /* do 1 if */
        tmp7=tmp7->next;
    }/*do wh*/
} /* do if maior*/
puts ("vou chamar juncao");
for (n=1;n<=nb;n++){
    printf("\n ENTREI CHAM JUC elim[%d]=%d,vjunc[%d]=%d\n",n,elim[n],n,vjun[n]);
    if ((elim[n]==1) && (vjun[n]==0)){
        printf ("\n CHAMEI SUB JUNCAO (nova) \n");
        if (l_red[i]=='1') puts("lred=1");
        if (ll_red[i]=='1') puts ("ll_red!=1");
        junc(n,l_red[n]);
        vjun[n]=1;
    }
}
}/* for */
} /* inicio pro*/

/* FIM DA ROTINA PRA TESTE DE INJEcoes */

```

/*****ROTINA TESTE DE FLUXOS *****/

```

void t_fluxo()
{
    int i,j,w=1;
    vetop *tmp,*tmp1;
    puts("ENTREI SUB FLUXO");
    /* printf ("\n COMPLETO\n");
    for (i=1;i<=nb;i++){
        tmp=vetop[i];
        while (tmp){
            printf (" ( %d , %d , %d , %d )",i,tmp->no,tmp->mi,tmp->mf,tmp->ltop);
            tmp=tmp->next;
        }
        printf ("\n");
    }
    for (i=1;i<=nb;i++){
        if (l_red[i]=='1') printf ("\nNO %d DEU 1\n",i);
        if (ll_red[i]=='1') printf ("\nNO %d DEU 0\n",i);
    }
    */
    for (i=1;i<=nl;i++){
        if (tip[i]=='1'){
            j=ni[i];
            tmp=vetop[j];
            printf ("\nNO INICIAL= %d, NO FINAL=%d\n",ni[i],nf[i]);
            while ((tmp) && (w)){
                if ((tmp->mi==1) && (tmp->ltop==1) && (tmp->no==nf[i])){
                    if (l_red[tmp->no]=='1'){
                        printf("\nA INJECAO %d ESTA LIVRE (1)\n",tmp->no);
                        w=0;
                    }else{
                        printf ("\nA INJECAO %d NAO ESTA LIVRE (1)\n",tmp->no);
                    }
                }
            }
            if (w==1){
                printf("\nENTREI 2 W,INJECAO %d\n",ni[i]);
                if (tmp->no==nf[i]){
                    if (proc_adj(tmp->no,ni[i])){
                        printf ("\nENCONTREI CAMINHO ALTERNATIVO (1)\n");
                        w=0;
                    }
                }
            }
            tmp=tmp->next;
        }
    }
    if(w==1){
        j=nf[i];
        tmp=vetop[j];
        printf ("\nNO INICIAL=%d, NO FINAL=%d\n",nf[i],ni[i]);
        while((tmp) && (w)){
            puts ("ENTREI IF INV 1");
            if ((tmp->mi==1) && (tmp->ltop==1) && (tmp->no==ni[i])){
                if (l_red[tmp->no]=='1'){
                    printf ("\nA INJECAO %d ESTA LIVRE (2)\n",tmp->no);
                    w=0;
                }else{
                    printf ("\nA INJECAO %d NAO ESTA LIVRE\n",tmp->no);
                }
            }
        }
    }
}

```

```

    }
  }
  if(w==1){
    if (tmp->no==ni[i]){
      if (proc_adj(tmp->no,nf[i])){
        printf ("\nENCONTREI CAMINHO ALTERNATIVO (2)\n");
        w=0;
      }
    }
  }
  tmp=tmp->next;
}
}
if (w==1) fprintf (banco, "\n\nO FLUXO DE %d A %d E CRITICO\n\n", ni[i], nf[i]);
if (w==0) printf ("\n\nO FLUXO DE %d A %d E REDUNDANTE\n\n", ni[i], nf[i]);
w=1;
}
}
}
void main(argc, argv)
int argc;
char *argv[];
{
  int i;
  char nomearq[30];          /* RECEBE O NOME DO ARQUIVO DE ENTRADA,
                             CASO ESTE NAO SEJA DEFINIDO NA CHAMADA DO
                             PROGRAMA */
  printf("DIGITE O NOME DO ARQUIVO DE ENTRADA: ");
  scanf("%s", nomearq);
  if ((banco=fopen(nomearq, "r"))==NULL){
    fprintf(banco, "Arquivo %s nao pode ser aberto.\n", nomearq);
    exit(1);
  }
}
else if ((banco=fopen(argv[1], "r"))==NULL){
  fprintf(banco, "Arquivo %s nao pode ser aberto.\n", argv[1]);
  exit(1);
}
}
inicia();
leitura();
puts("Topologia...");
topologia_rede();
top_rede();
printf("DIGITE O NOME DO ARQUIVO DE saida : ");
scanf("%s", nomearq);
if ((banco=fopen(nomearq, "w"))==NULL){
  fprintf(banco, "Arquivo %s nao pode ser aberto.\n", nomearq);
}
monta_H_Ht();
puts (" monta H_Ht");
gauss();
mostra_top();
verificar_top();
t_inj();
for (i=1; i<=nb; i++){
  vtop[i]=0;
}
gauss();

```

```
mostra_top();  
verificar_top();  
t_fluxo();  
  
fprintf(banco, "\n\n\nNUMERO TOTAL DE ITERACOES = %3d", it);  
fclose (banco);  
}
```


Anexo C

FLUXOGRAMA (FIGURA 7.4) E PROGRAMA, REFERENTES AO SEGUNDO MÉTODO PROPOSTO, O QUAL POSSIBILITA IDENTIFICAR AS MEDIDAS CRÍTICAS, ATRAVÉS DA MATRIZ JACOBIANA.

Figura 7.4 ⇒ Fluxograma correspondente à metodologia que permite identificar as medidas críticas, através da estrutura da matriz Jacobiana

Programa

/* DEFINIÇÕES DAS BIBLIOTECAS QUE SERÃO USADAS*/

```
#define NB 130
#define NL 300
#define NL2 600
#define NMED 200
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <malloc.h>
#include <sys/types.h>
#include <sys/times.h>
```

/* DECLARACAO DAS VARIAVEIS GLOBAIS */

```
char tipo[NB], /* INDICA SE HA MEDICAO DE INJECAO DE POTENCIA
                NA BARRA */
tip[NL], /* INDICA SE HA MEDICAO DE FLUXO DE POTENCIA NA
          LINHA */
conec; /* INDICA SE HA CONEXAO ENTRE AS AREAS */
int
inj[NB],
cont[NB],
saida_inj[NB],
nb, /* NUMERO DE BARRAS */
nmed, /* NUMERO DE MEDIDAS */
noex[NB], /* NUMERO EXTERNO DA BARRA */
nl, /* NUMERO DE LINHAS */
ni[NL], /* NUM INTERNO DA BARRA INICIAL DA LINHA */
nf[NL], /* NUM INTERNO DA BARRA FINAL DA LINHA */
it; /* NUMERO DE ITERACOES */

float Bkk[NL], /* ELEMENTOS DA DIAGONAL PRINCIPAL DA MATRIZ */
G[NB][NB];
H[NMED][NB];
```

FILE *banco; /* APONTA PARA O ARQUIVO DE ENTRADA E O DE SAIDA */

/* VARIAVEIS RELATIVAS A TOPOLOGIA DA REDE */

```
int npil[NB], /* INDICE INICIAL DOS VETORES NOLIG, LILIG, NEXT*/
nolig[NL2], /* NUMERO DA BARRA ADJACENTE */
lilig[NL2], /* NUMERO DA LINHA */
next[NL2]; /* PROXIMO INDICE, SE ZERO INDICA FIM */
int ilha[NB], prox[NB];
```

/* INICIACAO DOS VETORES */

```
void inicia()
```

```
{
register int i;
```

```

nb = nl = 0;

for ( i=0 ; i<NB ; i++ ) {
    noex[i] = npil[i] = 0;
}

for ( i=0 ; i<NL ; i++ ) {
    ni[i] = nf[i] = 0;
}
for ( i=0 ; i<NL2 ; i++ ) {
    nolig[i] = lilig[i] = next[i] = 0.0;
}
}

/* ROTINA DE LEITURA DOS DADOS DE ENTRADA */

void constr(str, b, e, snum) /* ESTA SUBROTINA COLOCA PARTE DA
                           STRING STR EM SNUM */

char str[80], /* STRING QUE CONTEM A LINHA */
snum[]; /* STRING COM A PARTE DESEJADA DA LINHA */
int b, /* INDICA A POSICAO INICIAL EM STR */
e; /* INDICA A POSICAO FINAL EM STR */
{
    int i, j, /* VARIAVEIS UTILIZADAS NO LOOPING */
    branco; /* VERIFICA SE SNUM ESTA PREENCHIDA DE ESPACOS */

    if (b<strlen(str)){
        snum[0] = '\0';
        branco = 1;
        for(i=b,j=0;i<=e;i++,j++){
            snum[j] = str[i];
            if (str[i]!=' ') branco = 0;
        }
        snum[j]='\0';
        if (branco == 1) snum = "0";
    }
}

void leitura()

{
    int i, /* CONTADOR DO LOOPING */
    flag1, flag2, /* VARIAVEIS DE CONTROLE DE SAIDA DO LOOPING */
    niex, nfex; /* NUMERO EXTERNO DOS NO'S INICIAL E FINAL */
    char linha[80], /*STRING DE LEITURA DA LINHA DO BANCO DE DADOS*/
    sno[5],
    sniex[5],
    snfex[5];

/* LEITURA DOS DADOS DE BARRA */

nb = nmed = 0;
fgets(linha, 80, banco);

/*constr(linha, 0, 3, sno);
noex[1] = atoi(sno);
noex[1] = atoi(linha);*/

```



```
puts("Iniciando a leitura");
while ((noex[++nb]=atoi(linha))!=9999) {

    tipo[nb] = linha[5];
    if (tipo[nb]=='1') nmed++;
    fgets(linha,80,banco);
}

/* LEITURA DOS DADOS DE LINHA */

nl = 0;

fgets(linha, 80, banco);

constr(linha, 0, 3, sniex);
niex = atoi(sniex);

while (niex!=9999){

    nl++;

    tip[nl] = linha[5];
    switch (tip[nl]) {
        case '1':
        case '2': nmed++;
                    break;
        case '3': nmed = nmed + 2;
    }
    constr(linha, 8, 11, snfex);
    nfex = atoi(snfex);
    flag1 = flag2 = i = 1;

    while ((flag1) || (flag2)) {

        if ((niex == noex[i]) && (flag1)) {
            ni[nl] = i;
            Bkk[i] = Bkk[i] + 1;
            flag1 = 0;
        }

        if ((nfex == noex[i]) && (flag2)) {
            nf[nl] = i;
            Bkk[i] = Bkk[i] + 1;
            flag2 = 0;
        }
        i++;
    }
    fgets(linha,80,banco);
    constr(linha, 0, 3, sniex);
    niex = atoi(sniex);
}
fclose(banco);
}
```

/* ESTA FUNCAO MONTA OS VETORES NPIL, NOLIG, LILIG, NEXT, OS
QUAIS DESCREVEM A TOPOLOGIA DA REDE. */

```
void topologia_rede()
{
    int aux,
        i,
        in,
        ina,
        k,
        l,
        n4,
        n5,
        lff = 0;
    char flag,
        kstop,
        prim;
    lff = 1;
    for (i=1;i<=nb;i++) {
        npil[i] = 0;
    }
    for (i=1;i<=2*nl;i++) {
        next[i] = 0;
        nolog[i] = 0;
        lilig[i] = 0;
    }
    for (i=1;i<=nl;i++) {
        prim = '1';
        k = ni[i];
        l = nf[i];
        kstop = '0';
        do {
            if (kstop == '0' && prim != '1') {
                aux = k;
                k = l;
                l = aux;
                kstop = '1';
            }
            prim = '0';
            ina = 0;
            in = npil[k];
            if (in == 0) {
                npil[k] = lff;
            }
            else {
                n5 = next[l];
                n4 = next[nolog[in]];
                flag = '0';
                while (n5 > n4 && flag != '1') {
                    ina = in;
                    in = next[in];
                    if (in == 0) {
                        next[lff] = in;
                        next[ina] = lff;
                        flag = '1';
                    }
                }
            }
        }
        else {
            else {
```

```

        n4 = next[nolig[in]];
    }
}
if (flag != '1') {
    if (ina == 0) {
        next[lff] = in;
        npil[k] = lff;
    }
    else {
        next[lff] = in;
        next[ina] = lff;
    }
}
}
nolig[lff] = l;
lilig[lff] = i;
lff++;
}
while (kstop == '0');
}

}

void monta_H_Ht()

{
int i, j, k, ind, lin, c_inj, nzero[NB], n, a, nfl, ord[NB], menor, cv;
int x, proc[NB][NB], c=0, el=0, d, p, vol, h, b[NB], cont[NB][NB], v, m, saida_inj[NB], noi;
float temp1 ,temp2;
puts ("Iniciando calculo de h_ht");

for (i=0; i<NMED; i++)
    for (j=0; j<NB; j++)
        H[i][j] = 0.0;

lin = 0;

for (i=1; i<=nl; i++) {
    if (tip[i]=='1') {
        lin++;
        H[lin][ni[i]] = 1.0;
        H[lin][nf[i]] = -1.0;
    }
    if (tip[i]=='2') {
        lin++;
        H[lin][ni[i]] = -1.0;
        H[lin][nf[i]] = 1.0;
    }
    if (tip[i]=='3') {
        lin++;
        H[lin][ni[i]] = 1.0;
        H[lin][nf[i]] = -1.0;
        lin++;
        H[lin][ni[i]] = -1.0;
        H[lin][nf[i]] = 1.0;
    }
}
}
c_inj=lin+1;

```

```

for (i=1;i<=nb;i++) {
    if (tipo[i]=='1') {
        lin++;
        ind = npil[i];
        while (ind!=0) {
            H[lin][nolig[ind]] = -2.0;
            ind = next[ind];
        }
        H[lin][i] = 2*Bkk[i];
    }
}
puts (" TESTE 1 ");
nb--;
printf("c_inj=%d\n",c_inj);
/*for (i=1;i<=nmed;i++){
    if (i==c_inj) puts ("comeca inj");
    for (j=1;j<=nb;j++){
        if (H[i][j]) printf ("x"); else printf("0");
    }
    printf("\n");
}*/

```

***** ANALISE DE MEDIDAS CRÍTICAS *****

/* INICIALIZANDO AS VARIÁVEIS */

```

for (i=1;i<=nb;i++){
    for (j=1;j<=nb;j++){
        cont[i][j]=0;
    }/*for j */
}/* for i */
for (i=1;i<=nb;i++){
    saida_inj[i]=0;
    b[i]=0;
}
a=1;

d=0;

for (k=1;k<=nb;k++){
    if (tipo[k]=='1'){
        inj[k]=c_inj;
        c_inj++;
    }
}
for (i=1;i<=nb;i++){

    if (tipo[i]=='1'){
        noi=i;
        /* printf("\n NO INJECAO=%d,noi=%d,inj=%d\n",i,noi,inj[i]);

```



```

printf ("\ncont[i=%d][proc[11]]=%d\n",9,cont[9][11]);*/

for (j=1;j<=nb;j++){
    if ((H[inj[i]][j]) && (j!=i)){
        b[i]++;
        proc[i][b[i]]=j;
    }/* H[inj][j] */
}/*for j */
for(k=1;k<=b[i];k++){
    printf ("\nproc[i=%d][k=%d]=%d\n",i,k,proc[i][k]);
}/* for k printf */
for (j=1;j<=b[i];j++){
    for (k=1;k<=nmed;k++){
        if(H[k][proc[i][j]]){
            if (H[k][i]){
                if (i < (proc[i][j])){
                    /* printf ("\ni=%d,proc=%d\n",i,proc[i][j]);*/
                    /* if (cont[i][proc[i][j]]){*/
                        cont[i][proc[i][j]]=cont[i][proc[i][j]] + 1;
                    /* */
                }else{
                    if((!cont[proc[i][j]][i]) || (el!=0)){
                        cont[proc[i][j]][i]++;
                        el++;
                    }/*if */
                }/*else */
            }/* if H[k][i] */
        }/* if H[i][proc]*/
    }/* for k */
    el=0;
}/* for j*/
}/*tipo*/

}/*for i */
i=1;
for (m=2;m<=2;m++){
/* for (i=1;i<=nb;i++){*/
v=0;
while((i<=nb) && (!v)){
    if (tipo[i]=='1'){
        noi=i;
        printf("\n NO INJECAO=%d,noi=%d,inj=%d\n",i,noi,inj[i]);
        printf ("\ncont[i=%d][proc[13]]=%d\n",12,cont[12][13]);
        for (k=1;k<=b[i];k++){
            if(a==1){
                if(i < (proc[i][k])){
                    if(cont[i][proc[i][k]] <= 1){
a=0;
                        puts ("a=0");
                    }/* if <= */
                }else{ /* if proc */
                    if (cont[proc[i][k]][i] <=1){
                        /* busca[c]=proc[i][k];*/
                        a=0;
                    }/*if else */
                }/*else */
            }
            printf ("\na=%d,proc=%d\n",a,proc[i][k]);

```

```

if (a==0){
    printf ("\n ENTREI proc=%d,c=%d\n",proc[i][k],c);
    for (h=1;h<=b[i];h++){
        for (j=1;j<=nmed;j++){
            if (H[j][proc[i][h]]){
                if ((H[j][proc[i][k]]) && (proc[i][h]!=proc[i][k])){
                    c++;
                    printf ("\nproc=%d,proc2=%d,c=%d,j=%d\n",proc[i][k],proc[i][h],c,j);
                }/*if */
            }/*if */
        }/*for j */
    }/*for h */
    if(c > 1) d=1;
    c=0;
    printf ("\n d=%d,proc=%d\n",d,proc[i][k]);
    if(d==1){
        a=1;
    }else{
        nfl=proc[i][k];
    }
    /* a==0 */
    d=0;
    /*if a */
    /* for k */
    if(a==1){
        printf("\nA INJECAO %d E REDUNDANTE \n",i);
        saida_inj[i]=1;
    }else{ /*if a */
        printf ("\n A INJECAO %d E CRITICA, VAI PARA %d\n",i,nfl);
        saida_inj[i]=2;
        for (h=1;h<=b[i];h++){
            if (i < (proc[i][h])){
                if (proc[i][h]!=nfl){
                    cont[i][proc[i][h]]=cont[i][proc[i][h]] -1;
                }/*2 if*/
            }else{ /* if*/
                if (proc[i][h]!=nfl){
                    cont[proc[i][h]][i]=cont[proc[i][h]][i] -1;
                }/*if else*/
            }/*else */
        }/*for h*/
        tipo[i]='0';
        for (k=1;k<=nb;k++){
            /* printf ("\n ENTREI ELIM \n");*/
            if((H[inj[i]][k]) && ((k==i) || (k==nfl))){
                /* puts("x");*/
                H[inj[i]][k]=1;
            }else{ /*if */
                H[inj[i]][k]=0;
                /* puts("0");*/
            }/*else*/
        }/*for k */
    }/*else */
    for (k=1;k<=b[i];k++){
        if (i < (proc[i][k])){
            if ((cont[i][proc[i][k]]<=1) && (proc[i][k]!=nfl)){
                if((tipo[proc[i][k]]=='1') && (i>proc[i][k])){

```

```

        vol=proc[i][k];
        /* i=nb;*/
        v=1;
        /* 3 if */
        /* 2 if */
    }else{ /* if */
        printf ("\ncont[proc=%d][i=%d]=%d\n",proc[i][k],i,cont[proc[i][k]][i]);
        if ((cont[proc[i][k]][i]<=1) && (proc[i][k]!=nfl)){puts("ENTREI");
            if (tipo[proc[i][k]]=='1'){
                vol=proc[i][k];
                printf ("\nEXISTE INJ NO NO %d\n",vol);
                /*i=nb;*/
                v=1;
                /* 2 if */
            }/* if */
        }/*else */
    }/*for k*/
    nfl=0;
    a=1;
    /* b=0;*/
    c=0;
    /* if tipo */
    i++;
    /* for i */
    printf ("\n SAI DO I VOL=%d\n",vol);
    if (v==1){

        i=vol;
        m--;
        vol=0;
    }/*if */
    }/*for m */
    for (k=1;k<=nb;k++){
        if (saida_inj[k]==1){
            fprintf (banco,"\n\n A INJECAO NO NO %d E REDUNDANTE\n\n",k);
        }
        if (saida_inj[k]==2){
            fprintf (banco,"\n\n A INJECAO NO NO %d E CRITICA\n\n",k);
        }
    }

    /* do for k*/

    for (k=1;k<=nl;k++){
        if (saida_flux[k]==1){
            fprintf (banco,"\n\n O FLUXO NO RAMO %d E REDUNDANTE\n\n",k);
        }
        if (saida_flux[k]==2){
            fprintf (banco,"\n\n O FLUXO NO RAMO %d E CRITICA\n\n",k);
        }
    }

    /* do for k*/

    /* puts (" BAIXO ");
    for (i=1;i<=nmed;i++){
        for (j=1;j<=nb;j++){
            if (H[i][j]) printf (" x "); else printf(" 0 ");

```

```

    }
    printf("\n");
}*/
}

/* METODO DE GAUSS */

void verif_conec_fluxo()

{
int adj, i, ind, ilh, n1, n2;

for(i=1;i<=nl;i++) {
    if (tip[i]!=' ') {
        if (ilha[ni[i]]!=ilha[nf[i]]) {
            n1 = 0; n2 = 0;
            ind = npil[ni[i]];
            ilh = ilha[nf[i]];
            while (ind!=0) {
                adj = nolog[ind];
                if (ilh==ilha[adj]) {
                    if (tipo[adj]=='1') n1++;
                    if (tip[lilig[ind]]!=' ') n1++;
                }

                ind = next[ind];
            }
            ind = npil[nf[i]];
            ilh = ilha[ni[i]];
            while (ind!=0) {
                adj = nolog[ind];
                if (ilh==ilha[adj]) {
                    if (tipo[adj]=='1') n2++;
                    if (tip[lilig[ind]]!=' ') n2++;
                }
                ind = next[ind];
            }
        }
        if (n1>n2) ilha[ni[i]] = ilha[nf[i]];
        else ilha[nf[i]] = ilha[ni[i]];
    }
}

}

void verif_conexao()

{
int i, ind, ilh;

conec = 'f';
fprintf(banco, "\n\n\nNUMERO DAS BARRAS ONDE FORAM ELIMINADAS MEDIDAS:\n");
for (i=1;i<=nb;i++) {
    if (tipo[i]=='1') {
        ilh = ilha[i];
        ind = npil[i];
        while (ind) {
            if (ilh!=ilha[nolog[ind]]) {
                fprintf(banco, "%3d\n", i);
                tipo[i] = '0';
            }
        }
    }
}
}

```



```

        nmed--;
        ind = 0;
        conec = 'v';
    } else
        ind = next[ind];
    }
}

void main(argc, argv)

int argc;
char *argv[];

{
    char nomearq[30]; /* RECEBE O NOME DO ARQUIVO DE ENTRADA,
                       CASO ESTE NAO SEJA DEFINIDO NA CHAMADA DO
                       PROGRAMA */

    if (argc!=2){
        printf("DIGITE O NOME DO ARQUIVO DE ENTRADA: ");
        scanf("%s",nomearq);
        if ((banco=fopen(nomearq,"r"))==NULL){
            fprintf(banco,"Arquivo %s nao pode ser aberto.\n",nomearq);
            exit(1);
        }
    }
    else if ((banco=fopen(argv[1],"r"))==NULL){
        fprintf(banco,"Arquivo %s nao pode ser aberto.\n",argv[1]);
        exit(1);
    }
    inicia();
    leitura();
    puts("Topologia...");
    topologia_rede();

    printf("DIGITE O NOME DO ARQUIVO DE saida : ");
    scanf("%s",nomearq);
    if ((banco=fopen(nomearq,"w"))==NULL){
        fprintf(banco,"Arquivo %s nao pode ser aberto.\n",nomearq);
    }

    conec = 'v';
    it = 0;

    it++;
    monta_H_Ht();
    verif_conec_fluxo();

    verif_conexao();*/
    /*} */
    fprintf(banco,"\n\n\n\nNUMERO TOTAL DE ITERACOES = %3d",it);
    fclose (banco);
}

```

Referências Bibliográficas

1. ALLMONG, J. J. ; IRISARRI, G. D. & SASSON, A. M. ,"- An Examination of Solvability for State Estimation Algorithms". IEEE PES Winter Meeting. Paper N° A 80 008 -3, 1980.
2. BRETAS, N. G. "- Network Observability: A theory based on triangular factorization and path graph concepts - Part I". International conference on Automatic Control, cidade do Porto, Portugal, Setembro 1996.
- 3 BRETAS, N. G. ; Déo, E. C. & Mori, R. N. "- Network Observability: Algorithms based on triangular factorization and path graph concepts - Parte II". International conference on Automatic Control, cidade do Porto, Portugal, Setembro 1996.
4. BRETAS, N. G., "- Network Observability: Theory and algorithms based on triangular factorization and path graph concepts". IEE Proceedings, Generation, Transmission and Distribution, VOL. 143, N° 1, pp. 123- 128, Janeiro 1996.
5. CLEMENTS, K. A. & WOLLENBERG, B. F. "- An algorithm for Observability Determination in Power System State Estimation". IEEE PES Summer Meeting, Paper A 75 447-3, 1975.

6. CLEMENTS, K. A. ; KRUMPHOLZ, G. R. & DAVIS, P. W. "- Power System State Estimation Residual Analysis: an algorithm using network topology". IEEE Transactions on Power Apparatus and Systems, VOL. 100, N° 4, pp.1779- 1787, Abril 1981.
7. CLEMENTS, K. A. ; KRUMPHOLZ, G. R. & DAVIS, P. W. "- Power System State Estimation with Measurement Deficiency: an algorithm that Determines the Maximal Observable Subnetwork". IEEE Transactions on Power Apparatus and Systems, VOL. 101, N° 9, pp. 3044- 3052, Setembro 1982.
8. CLEMENTS, K. A. ; KRUMPHOLZ, G. R. & DAVIS, P. W. "- Power System State Estimation with Measurement Deficiency: an Observability /Measurement Placement Algorithm". IEEE Trans. Power Apparatus and Systems, VOL. PAS - 102, pp. 2012 - 2020, Julho 1983.
9. CRAINIC, E. D. ; HORISBERGER, H. P. ; DO, X. D. & MUKHEDKAR, D., "-Power Network Observability: The Assesement of the Measurement System Strength". IEEE Transactions on Power Systems, VOL. 5, N° 4, pp. 1267- 1285, Novembro 1990.
10. FETZER, E. E. & ANDERSON, P. M. "- Observability in the State Estimation of Power Systems". IEEE Transactions on Power Apparatus and Systems, VOL. PAS - 94, pp. 1981 - 1988, Novembro - Dezembro 1975.
11. GANTMACHER, F. R. "*The Theory of Matrices*", VOL.1, Chelsea, 1977.

-
12. HANDSCHIN, E. & BONGERS, C., "- Theoretical and Practical Considerations in the Design of State Estimators for Electric Power Systems". In *Computerized Operation of Power Systems*, S. C. Savulescu, Editor, Elsevier, pp. 104 - 136, 1972.
13. HORTON, J. S. & MASIELLO, R. D., "- On-line Decoupled Observability Processing", *PICA Proc.*, pp. 420 - 426, Maio 1977.
14. KRUMPHOLZ, G. R.; CLEMENTS, K. A. & DAVIS, P. W. "- Power System Observability: a practical algorithm using network topology". IEEE Transactions on Power Apparatus and Systems, VOL. PAS 99, N° 4, pp. 1534- 1542, Julho - Agosto 1980.
15. MONTICELLI, A. & WU, F. F. "- Network Observability: Theory". IEEE Transactions on Power Apparatus and Systems, VOL. 104, N° 5, pp. 1042- 1048, Maio 1985.
16. MONTICELLI, A. & WU, F. F. "- Network Observability: Identification of Observability Islands and Measurement Placement". IEEE Transactions on Power Apparatus and Systems, VOL. 104, N° 5, pp. 1035- 1041, Maio 1985.
17. MORI, H. & TSUZUKI, S. "- A Fast Method for Topological Observability Analysis Using a Minimum Spanning Tree Technique". IEEE Transactions on Power Systems, VOL. 6, N° 2, pp. 491 - 499, Maio 1991.
18. MOROZOWSKI, M. F. , "- *Matrizes Esparsas em Redes de Potência: Técnicas de Operação*", Rio de Janeiro - Livros Técnicos e Científicos Editora S. A., 1981.

-
19. NUCERA, R. R. & GILLES, M. L. "- Observability Analysis: A New Topological Algorithm". IEEE Transactions on Power Systems, VOL. 6, N° 2, pp. 466 - 475, Maio 1991.
20. QUINTANA, V. H.; SIMOES-COSTA, A. & MANDEL, A. "- Power System Topological Observability Using a Direct Graph-Theoretic Approach". IEEE Transactions on Power Apparatus and Systems, VOL. PAS 101, N° 3, pp. 617- 626, Março 1982.
21. SCHWEPPE, F. C. & WILDES, J. , "- Power System Static State Estimation, Part I : Exact Model". IEEE Transactions on Power Apparatus and Systems, VOL. PAS - 89, pp. 120 - 125, Janeiro 1970.
22. SLUTSKER, I. W. & SCUDDER, J. M. "- Network Observability Analysis Through Measurement Jacobian Matrix Reduction". IEEE Transactions on Power Systems, VOL. PWRS - 2, N° 2, pp. 331- 337, Maio 1987.
23. TINNEY, W. F. ; BRANDWAJN, V. & CHAN, S. M. "- Sparse Vector Methods". IEEE Transactions on Power Apparatus and Systems, VOL. 104, N° 2, pp. 295- 300, Fevereiro 1985.