Serviço de Pós-Graduação EESC/USP
EXEMPLAR REVISADO
Data de entrada no Serviço: 29/10/01
Ass.: Guilherme

TÉCNICAS DE ESPARSIDADE EM SISTEMAS ESTÁTICOS DE ENERGIA ELÉTRICA

Sandra Fiorelli de Almeida Penteado Simeão

Dissertação apresentada à Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos parciais para obtenção do título de Mestre em Engenharia Elétrica

DEDALUS - Acervo - EESC
31100036816

ORIENTADOR: Prof. Dr. Geraldo R. M. da Costa

São Carlos
2001

EESC-USP
Biblioteca
CAMPUS DE S. CARLOS

Classe. TESE-EESC
Cutt. 5396
Tombo T0262/01

dt 12/2 991

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

S589t

Simeão, Sandra Fiorelli de Almeida Penteado
Técnicas de esparsidade em sistemas estáticos de energia elétrica / Sandra Fiorelli de Almeida Penteado Simeão. -- São Carlos, 2001.

Dissertação (Mestrado) -- Escola de Engenharia de São Carlos-Universidade de São Paulo, 2001.
Área: Engenharia Elétrica.
Orientador: Prof. Dr. Geraldo R. M. da Costa.

1. Esparsidade. 2. Bi-fatoração. 3. Eliminação esparsa e simétrica de Gauss. 4. MA28. 5. Vetores esparsos. 6. Refatoração parcial. I. Título.

FOLHA DE JULGAMENTO

Candidata: Licenciada **SANDRA FIORELLI DE ALMEIDA PENTEADO SIMEÃO**

Dissertação defendida e julgada em 27-09-2001 perante a Comissão Julgadora:

APROVADA

Prof. Dr. **GERALDO ROBERTO MARTINS DA COSTA (Orientador)**
(Escola de Engenharia de São Carlos/USP)

APROVADA

Prof. Dr. **PAULO ROGÉRIO POLITANO**
(Universidade Estadual de Campinas/UNICAMP)

APROVADA

Prof. Dr. **DILSON AMÂNCIO ALVES**
(UNESP/Campus de Ilha Solteira)

p/ Prof. Doutor **MURILO ARAUJO ROMERO**
Coordenador do Programa de Pós-Graduação
em Engenharia Elétrica

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

Deus - princípio e fim - obrigada por me permitir
alcançar mais este objetivo em minha vida!
Maria - Mãe das mães - obrigada por sempre
me oferecer o seu colo!

"Até um tolo sabe contar as sementes
que há em uma maçã;
só Deus sabe contar as maçãs todas
que há em uma semente"

Robert H. Shuller

Antonio, meu Pai, de quem herdei a determinação e a organização.
Neide, minha Mãe, de quem herdei a fé e a perseverança.
O fruto do meu trabalho é também seu fruto, pois me deram a vida e a coragem para vivê-la.

Marido e Amigo Vitor, primeiro e incansável incentivador, em quem encontrei o amor, a humildade e a verdade.
Este trabalho também é um produto seu, pois sem seu apoio crítico, cumplicidade, paciência e compreensão, ele não se concretizaria.

Filho João Vitor, bênção divina, sua inocência e carinho me mostram, a cada dia, o amor puro e sincero.
Que você possa se orgulhar de sua mãe no futuro.

A vocês, motivos de minha existência, dedico este trabalho:
a maçã que Deus enxergou em minhas sementes ...

AGRADECIMENTOS

Ao Professor Dr. Geraldo Roberto Martins da Costa, pela competência e extremo profissionalismo com que conduziu a sua orientação, e ainda, pela paciência, disponibilidade e amizade durante o desenvolvimento deste trabalho.

À Sueli e Francisco; Sr. Waldomiro e D. Iolanda; Cristiane, Alex e Ana Beatriz; Pedro, Inara, Tiago e Samuel, pelo incentivo e torcida.

Aos colegas, professores e funcionários, do Departamento de Ciências Exatas e Naturais e do Centro de Pesquisa e Pós-Graduação, da Universidade do Sagrado Coração, pelo apoio e idéias.

Às Irmãs da USC, particularmente à Profa. Dra. Ir. Alice Garcia de Moraes, por confiar em meu potencial, e à Profa. Ms. Ir. Geni da Silva, pela compreensão.

Aos alunos (passados, presentes e futuros) do curso de Matemática da USC, por serem o motivo e por permitirem compartilhar os novos conhecimentos adquiridos.

À Profa. Ms. Léa Sílvia Braga de Castro Sá, pelas correções e sugestões.

Ao Vitor (meu marido), também pelo suporte computacional e logístico.

À Edméa, por tantos nós desfeitos.

À Kátia, pelos primeiros contatos em São Carlos.

Aos amigos do LOSEP: Alessandra, Biagi, Cristiano, Luiz Claudio, Leandro e Thales, pelo acolhimento, troca de idéias e convivência. Especialmente à Vanusa e ao Edmárcio, que, incansavelmente, me ajudaram em tantos momentos críticos.

Às Professoras Andréa e Roberta do Departamento de Computação da UNESP-Bauru, pelas informações computacionais técnicas.

Aos companheiros de viagem: Fátima, Marcos, Giovana e Mauro, por tantas horas de trocas de experiências.

A todos os professores e funcionários do Departamento de Engenharia Elétrica da EESC-USP-São Carlos e das Bibliotecas, pela atenção e colaboração.

À CAPES, Coordenadoria de Aperfeiçoamento de Pessoal de Nível Superior, pelo auxílio financeiro concedido durante a realização deste trabalho.

A "TODOS" os que me acompanharam e comigo conviveram nesta jornada, agradeço-lhes com profundo respeito, a torcida, as orações e, principalmente, a amizade.

SUMÁRIO

LISTA DE FIGURAS xi

LISTA DE TABELAS xiii

LISTA DE ABREVIATURAS xiv

LISTA DE SÍMBOLOS xvi

RESUMO xviii

ABSTRACT xix

1 - INTRODUÇÃO 1

1.1 - Motivação 2

1.2 - Áreas de aplicação de Técnicas de Esparsidade 4

1.3 - Objetivos 7

2 - FUNDAMENTAÇÃO TEÓRICA 10

2.1 - Introdução ao Fluxo de Carga Ótimo 10

2.2 - Aspectos Matemáticos Gerais 12

2.2.1 - Sistemas Triangulares 14

2.2.2 - Eliminação de *Gauss* 16

2.2.3 - Fatoração LU 19

2.3 - Partição de Matrizes 25

2.4 - Matrizes Esparsas 27

3 - HISTÓRICO 29

4 - ORDENAÇÃO 58

4.1 - Aspectos Gerais da Ordenação 58

4.2 - Esquemas de Ordenação 63

5 - TÉCNICAS DE ESPARSIDADE 95

5.1 - Técnicas para exploração da esparsidade da Matriz de Admitância de Rede - TINNEY & SATO (1963) 96

5.2 - Fatoração Triangular Otimamente Ordenada - TINNEY & WALKER (1967) 98

5.3 - Método da Bi-Fatoração - ZOLLENKOPF (1971) 101

5.4 - Método da Eliminação Esparsa e Simétrica de Gauss - EISENSTAT, SCHULTZ & SHERMAN (1981) 105

5.5 - Métodos de Vetores Esparsos - TINNEY, BRANDWAJN & CHAN (1985) 113

5.6 - Método da Substituição Esparsa *Forward* e *Back* - ALVARADO, MONG & ENNS (1985) 119

5.7 - Método da Refatoração Parcial da Matriz - CHAN & BRANDWAJN (1986) 120

5.8 - Melhoria na Técnica de Matriz/Vetor Esparso - Aplicações nos Cálculos do Fluxo de Potência On-Line - RISTANOVIC', BJERLOGRLIC'& BABIC' (1989) 124

5.9 - Fatores Inversos da Matriz Esparsa - ENNS, TINNEY & ALVARADO (1990) 126

5.10 - Métodos Particionados da Inversa da Matriz Esparsa - ALVARADO, YU & BETANCOURT (1990) 130

5.11 - Técnicas de Aproximação do Vetor Esparso - BACHER, EJEBE & TINNEY (1991) 134

6 - IMPLEMENTAÇÃO E TESTES 139

6.1 - Armazenamento da Matriz 140

6.2 - Testes e Comparações 145

7 - CONCLUSÕES 150

REFERÊNCIAS BIBLIOGRÁFICAS 153

LISTA DE FIGURAS

FIGURA 1.1 - Levantamento bibliográfico sobre Técnicas de Esparsidade 3

FIGURA 4.1 - Rede com 5 nós 60

FIGURA 4.2 - Exemplo de redução de uma rede - eliminação de uma barra 63

FIGURA 4.3 - Formas da Matriz de Banda 64

FIGURA 4.4 - Submatriz reduzida antes do k-ésimo passo da Fatoração LU 67

FIGURA 4.5 - Estrutura de esparsidade da matriz U (14 nós - ordenação T-2) 70

FIGURA 4.6 - *Path Graph* 70

FIGURA 4.7 - Estrutura de esparsidade de U^{-1} (ordenação T-2) 71

FIGURA 4.8 - Sistema de 14 nós ordenado pelo algoritmo A1 74

FIGURA 4.9 - Sistema de 14 nós ordenado pelo algoritmo A2 74

FIGURA 4.10 - Sistema de 14 nós ordenado pelo algoritmo A3 74

FIGURA 4.11 - Sistema de 10 nós 76

FIGURA 4.12 - Estrutura de U segundo MD-ML 76

FIGURA 4.13 - *Filled graph* antes dos nós 1, 8 e 9 serem eliminados 76

FIGURA 4.14 - Crescimento do *path graph* considerando três diferentes hipóteses 77

FIGURA 4.15 - *Path graph* antes dos nós 1 e 8 terem sido eliminados 77

FIGURA 4.16 - Estrutura de U seguindo o algoritmo MD-MNP 79

FIGURA 4.17 - Modelo de Esparsidade de L e L^{-1} 83

FIGURA 4.18 - Rede exemplo de 20 nós 88

FIGURA 4.19 - Topologia da matriz *A* 88

FIGURA 4.20 - Matriz Aumentada depois do nono passo da eliminação 89

FIGURA 5.1 - Esquema de comparação entre os métodos da Decomposição Triangular e Bi-Fatoração 102

FIGURA 5.2 - Fatores Triangulares ***L*** e ***R*** da matriz 103

FIGURA 5.3 - Esquema de armazenamento sem compressão da matriz *A* 108

FIGURA 5.4 - Esquema de armazenamento sem compressão da matriz ***U*** 110

FIGURA 5.5 - Esquema de armazenamento comprimido da matriz ***U*** 110

FIGURA 5.6 - Visão esquemática do pacote *YSMP I* 113

FIGURA 5.7 - Rede exemplo de 20 nós 115

FIGURA 5.8 - Estrutura de Esparsidade das Matrizes *A* e ***L*** 115

FIGURA 5.9 - *Path Graph* 116

FIGURA 5.10 - Sub-grafo 117

FIGURA 5.11 - Matriz que apresenta as modificações na linha/coluna 2 121

FIGURA 5.12 - *Path* de fatoração ilustrando o conceito de *lenght* 132

FIGURA 6.1 - Rede exemplo de seis nós 140

FIGURA 6.2 - Arquivo de dados de uma rede elétrica de 6 nós 141

FIGURA 6.3 - Esquema de armazenamento do método da Bi-fatoração 142

FIGURA 6.4 - Esquema de armazenamento segundo pacote MA28 143

FIGURA 6.5 - Esquema de armazenamento sem compressão da matriz *A* 144

FIGURA 6.6 - Esquema de armazenamento sem compressão da matriz ***U*** 145

FIGURA 6.7 - Comparação entre os tempos de processamento (em segundos) para as três técnicas de Esparsidade 147

FIGURA 6.8 - Desempenho das técnicas relativo ao sistema de 14 barras 148

FIGURA 6.9 - Desempenho das técnicas relativo ao sistema de 30 barras 148

FIGURA 6.10 - Desempenho das técnicas relativo ao sistema de 57 barras 149

FIGURA 6.11 - Desempenho das técnicas relativo ao sistema de 118 barras 149

LISTA DE TABELAS

TABELA 2.1 - Dimensões da matriz $\nabla^2 La$ e número de elementos para o sistema IEEE30 12

TABELA 3.1 - Comparações entre cronometragens de tempo para algumas matrizes 41

TABELA 3.2 - Comparações entre cronometragens de tempo para algumas matrizes utilizando-se o MA18 e MA28 42

TABELA 4.1 - Comparação de estratégias de ordenação sobre matrizes esparsas assimétricas 68

TABELA 4.2 - Números possível e atual de *fill-in* no décimo passo 90

TABELA 4.3 - Número de *fill-ins* em *C/W* e número de multiplicações necessárias para obtê-los 91

TABELA 4.4 - Tempo de computação das soluções via método dos vetores esparsos 94

TABELA 5.1 - *Path Table* 116

TABELA 5.2 - Comparação entre os Esquemas 2, W e W PARTICIONADO e observações sobre as Partições 130

TABELA 5.3 - Número de Partições e *fill-ins* para os casos *PA1, PA2* e *PA3* 133

TABELA 5.4 - Partições e número de *fill-ins* nas matrizes *W* depois da ordenação 134

TABELA 6.1 - Tempos de processamento (em segundos) para os quatro sistemas em relação às três técnicas 147

LISTA DE ABREVIATURAS

FC: Fluxo de carga;

FCO: Fluxo de carga ótimo;

SEP: Sistema Elétrico de Potência;

IEEE: Institute of Electrical and Electronics Engineers;

NB: Número de barras;

NR: Número de ramos;

GE: Grau de esparsidade;

HSL: Harwell Subroutine Library;

YSMP: Yale Sparse Matrix Package;

PMR: Partial Matrix Refatorization;

PV: Barra de tensão controlada;

T-2: Esquema 2 de ordenação de TINNEY;

MD: Minimum degree;

ML: Minimum length;

MD-ML: Minimum degree - minimum length;

MNP: Minimum number of predecessors;

MD-MNP - Minimum degree - minimum number of predecessors;

ML-MD: Minimum length - minimum degree;

MIF - Minimum inverse fill-in;

MF-MIF: Minimum fill-in - minimum inverse fill-in;

MFI-1: Minimum fill-in 1;

MFI-2: Minimum fill-in 2;

MFI1-ML: Minimum fill-in 1 - minimum length;

MFI2-ML: Minimum fill-in 2 - minimum length;

ASP: Approximate Sparse Vector

LISTA DE SÍMBOLOS

La - Função Lagrangiana;

f(x) - Função objetivo;

g(x) - Vetor das equações do problema de fluxo de carga;

h(x) - Vetor das inequações funcionais do problema;

x - Vetor das variáveis de estado do problema;

$\nabla^2 La$ - Matriz Lagrangiana;

$\Delta\varphi$ - Vetor de correções;

∇La - Gradiente de La;

A - Matriz dos coeficientes;

b - Vetor independente (constantes);

A^{-1} - Matriz Inversa de A;

a_{ij} - Elemento da matriz A situado na linha i e coluna j;

L - Matriz Triangular Inferior;

l_{ij} - Elemento da matriz L situado na linha i e coluna j;

U - Matriz Triangular Superior;

u_{ij} - Elemento da matriz U situado na linha i e coluna j;

M - Matriz de Transformação de Gauss;

L^{-1} - Matriz Inversa de L;

L^t - Matriz Transposta de L;

U^1 - Matriz Inversa de U;

U^t - Matriz Transposta de U;

D - Matriz Diagonal;

W - Matriz de Permutações;

G_f - Filled Graph;

I - Vetor das Injeções de Corrente;

Y - Matriz Admitância Nodal;

E - Vetor das Tensões Nodais;

T - Matriz Elementar;

I - Matriz Identidade

RESUMO

SIMEÃO, S.F.A.P. (2001). *Técnicas de Esparsidade em Sistemas Estáticos de Energia Elétrica.* São Carlos, 2001, 200p. Dissertação (Mestrado) - Escola de Engenharia de São Carlos, Universidade de São Paulo.

Neste trabalho foi realizado um grande levantamento de técnicas de esparsidade relacionadas a sistemas estáticos de energia elétrica. Tais técnicas visam, do ponto de vista computacional, ao aumento da eficiência na solução da rede elétrica objetivando, além da resolução em si, a redução dos requisitos de memória, armazenamento e tempo de processamento. Para tanto, uma extensa revisão bibliográfica foi compilada, apresentando um posicionamento histórico e uma ampla visão do desenvolvimento teórico. Os testes comparativos realizados para sistemas de 14, 30, 57 e 118 barras, sobre a implementação de três das técnicas mais empregadas, apontou a Bi-fatoração como tendo o melhor desempenho médio. Para sistemas pequenos, a Eliminação Esparsa e Simétrica de Gauss apresentou melhores resultados. Este trabalho fornecerá subsídios conceituais e metodológicos a técnicos e pesquisadores da área.

Palavras-chave: Esparsidade, Bi-fatoração; Eliminação Esparsa e Simétrica de Gauss; MA28; Vetores Esparsos; Refatoração Parcial.

ABSTRACT

SIMEÃO, S.F.A.P. (2001). *Técnicas de Esparsidade aplicadas a Sistemas Estáticos de Energia Elétrica.* São Carlos, 2001, 200p. Dissertação (Mestrado) - Escola de Engenharia de São Carlos, Universidade de São Paulo.

In this work a great survey of sparsity techniques related to static systems of electric power was accomplished. Such techniques seek, for of the computational point of view, the increase of the efficiency in the solution of the electric net aiming, besides the resolution of itself, the reduction of memory requirements, the storage and time processing. For that, an extensive bibliographic review was compiled providing a historic positioning and a broad view of theoretic development. The comparative tests accomplished for systems of 14, 30, 57 and 118 buses, on the implementation of three of the most employed techniques, it pointed out an Bi-factorisation as best medium performance. For small systems, the Sparse Symmetric Gaussian Elimination showed the best results. This work will supply conceptual and methodological subsidies to technicians and researchers of the area.

Keywords: Sparsity; Bi-factorisation; Sparse Symmetric Gaussian Elimination; MA28; Sparse Vector; Partial Refactorization.

1 - INTRODUÇÃO

A resolução de uma rede elétrica envolve o manuseio de sistemas algébricos lineares de grandes dimensões com matrizes de coeficientes altamente esparsas. A eficiência na resolução destes depende, em grande parte, do modelo e da técnica de esparsidade utilizados.

Um clássico problema desta área é o fluxo de carga, que, quando resolvido pelo Método de *Newton-Raphson*, gera uma matriz Jacobiana, cujo grau de esparsidade pode atingir 99,9%, dependendo somente da configuração do sistema. Esta é uma matriz quadrada, real, simétrica em estrutura, mas assimétrica em valores, e definida positiva.

Outra matriz, que aparece neste tipo de estudo, é a relacionada com o fluxo de carga ótimo. Trata-se da Matriz Lagrangiana, cujos coeficientes são as derivadas parciais de

segunda ordem. Esta é quadrada, estritamente simétrica (posições e valores) e com zeros na diagonal (COSTA et al., 2000).

Dessa forma, os sistemas lineares que possuem este tipo de matriz coeficiente são denominados *sistemas esparsos* e sua resolução é objeto de estudo de técnicos e pesquisadores da área de sistemas de potência, através da utilização de métodos e procedimentos desenvolvidos sob o título geral de *"técnicas de esparsidade"*. Estas técnicas visam ao aumento da eficiência na solução de uma rede elétrica através de diversas modalidades de resolução dos sistemas lineares, que não a inversão, pois, se para um sistema comum a determinação da inversa da matriz dos coeficientes é inviável, mais crítico torna-se, então, para a matriz esparsa, uma vez que, em geral, sua inversa é cheia.

Do ponto de vista computacional, as técnicas visam, além da resolução dos sistemas em si, a redução dos requisitos de memória, armazenamento e tempo de processamento, necessários à solução dos problemas de rede.

1.1 - Motivação

Como verificado, a utilização de técnicas de esparsidade constitui-se o âmago da busca de soluções eficazes para os problemas em sistemas estáticos de energia. É, então, justamente neste ponto, que o presente trabalho insere-se. A procura por estas soluções mais eficientes, dos pontos de vista computacional e estrutural, e, ainda, condizentes com a realidade dos problemas de rede elétrica, foi a motivação que o conduziu.

Para tanto, uma extensa revisão bibliográfica foi compilada, a fim de fornecer, além do posicionamento histórico, uma ampla revisão do desenvolvimento teórico. É necessário ressaltar que tal revisão não é exaustiva, uma vez que se procurou enfocar somente os trabalhos realizados na área de sistemas estáticos de energia utilizando-se de computadores seriais e técnicas de programação para tal. Verificou-se, assim, que os primórdios do emprego das técnicas de esparsidade remontam à década de 50, sendo seguido de um grande avanço em pesquisas nas décadas de 70 e 90 com o advento de modernos computadores e tecnologia da informação.

A figura 1.1 representa a evolução da literatura sobre técnicas de esparsidade citadas nesta pesquisa.

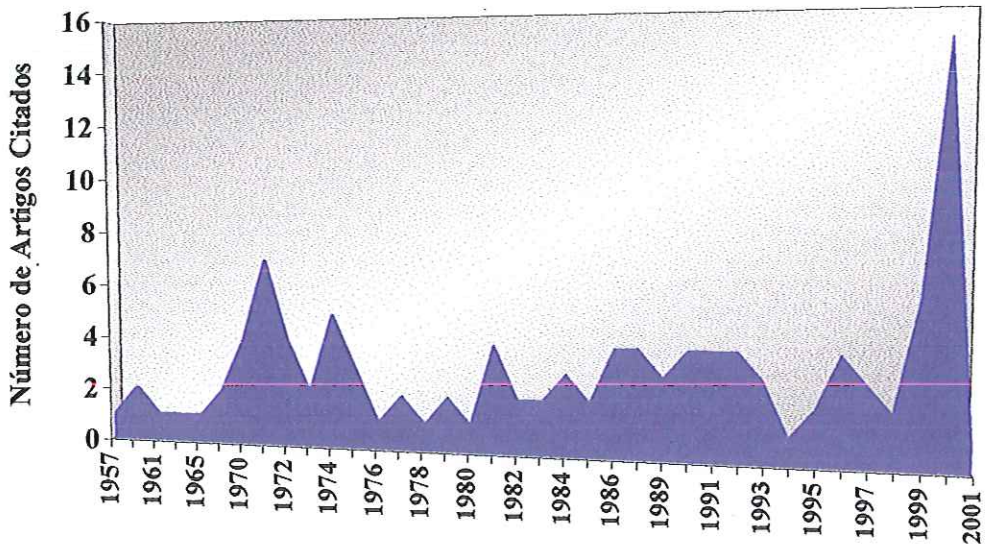


FIGURA 1.1 - Levantamento bibliográfico sobre Técnicas de Esparsidade

1.2 - Áreas de aplicação de Técnicas de Esparsidade

É muito difícil listar todas as áreas nas quais a tecnologia de matrizes esparsas é empregada, dado seu amplo espectro de utilização. Justamente, devido ao tamanho da lista, esta foi dividida em categorias principais e cada uma subdividida em áreas mais específicas. Algumas destas áreas de aplicações têm tido uma influência muito profunda tanto na pesquisa de matrizes esparsas quanto na própria área de aplicação em si.

- Análise Numérica:
  - Cálculo de valores próprios dos sistemas (DUFF, 1977; CHEN & DEMMEL, 2000)
  - Equações integrais (HASHIMOTO, 1970; CANNING, 1992)
  - Equações lineares e não lineares (DUFF, 1977)
  - Programação Linear (PARK, KIM, SEOL et al, 2000)
  - Equações Diferenciais Ordinárias (LINIGER & WILLOUGHBY, 1970; CARVER & MACEWEN, 1981)
  - Equações Diferenciais Parciais (DUFF, 1977)
  - Métodos de Projeção (MAAS, PEREVERZEV, RAMLAU et al., 2001)

- Matemática:
  - Análise Combinatória (DUFF, 1977)
  - Computação (MEIJSTER & WUBS, 2000; JONES & PLASSMANN, 2000)
  - Programação Dinâminca (MARTELLI & MONTANARI, 1972)
  - Teoria dos Grafos (HARARY, 1971; KAVEH, 1991)
  - Modelagem (BETTERIDGE, 1974)

- Pesquisa Operacional (GILL, MURRAY, SAUNDERS et al., 1984; ARORA & LI, 1993)
- Estatística (GENTLEMAN, 1974; ARAUJO, MEDINA & APONTE, 1999)

- Engenharia:
  - Métodos de Compensação sobre redes (VAN NESS & GRIFFIN, 1961; TINNEY & HART, 1967; OGBUOBIRI, 1970; TINNEY, 1971; MEYER & ALBERTSON, 1971; LEE, 1993; FREITAS & COSTA, 1999; HONG & WANG, 2000)
  - Análise no domínio da freqüência (ERISMAN & SPIES, 1972)
  - Análise de Linhas de Transmissão (RADHAKRISHNAN & CHEW, 1999)
  - Processamento de Sinais (STOICA, MOSES, SODERSTROM et al., 1991; BRITANAK & RAO, 1999)
  - Redes Neurais (COOK & GILBERT, 1988)
  - Engenharia Química (ZHONG, SHAO, ZHANG et al., 2000; NEMETH, COULAUD, MONARD et al., 2000)
  - Engenharia Bioquímica (SOMMERFELD, SONDHI, SPURLOCK et al., 1981)
  - Engenharia Biomédica (WOO, HUA, WEBSTER et al., 1994; SIGITANI, IIGUNI & MAEDA, 2000)
  - Engenharia Nuclear (LEIMBACH & ZELLER, 1982)

- Ciências Aplicadas:
  - Sistemas Hidráulicos (WANG, KIM & YOON, 2000; JHA, HERATH & MUSIAKE, 2000)

- Análise Estrutural (JOHNSEN & ROY, 1974; WANG, YU, YONG et al., 2000)
- Problemas de Vibração (JENNINGS & ORR, 1971)
- Genética (TIER & SMITH, 1989; VARONA & MISZTAL, 1999)
- Poluição Atmosférica (JACOBSON, LU, TURCO et al., 1996)
- Informática (MUHLBACHER, 1980; McLAY, SWIFT & CAREY, 1996)
- Simulação de Reservatórios (GHAHAN & JENNINGS, 1979)

- Ciências:
  - Cristalografia (HENNESSY, BUCHANAN, et al., 2000)
  - Física (PETIT , BEIDEN, TEMMERMAN et al., 2000; WEDEKIND & McKAY, 2000)
  - Desenvolvimento Automobilístico (FARRCHER & GROTE, 1971)

- Humanidades:
  - Economia (PACE, 1997)
  - Ciências Sociais (DRUD, 1978)
  - Sociologia (ROSS & HARARY, 1959)
  - Desenvolvimento Urbano (ROGERS, 1971)
  - Educação (MISLEVY, BEATON, KAPLAN et al., 1992)

A ampla variedade de áreas de estudo citadas, cada qual com suas peculiaridades, mostra-nos, claramente, a existência de uma vasta gama de formulações possíveis para o mesmo problema básico.

Evidentemente, somente um tratamento geral, de natureza analítico-matemática, traria resultados aplicáveis irrestritamente a todas as áreas. No entanto, uma abordagem sob

este ângulo daria origem, certamente, a métodos de solução bem menos eficientes do que os provenientes de um tratamento mais específico. Isto se justifica se considerarmos que um método geral teria que levar em conta situações comuns a uma determinada área e de remota possibilidade de ocorrência em outras. Por outro lado, uma formulação excessivamente particularizada traria, como inevitável conseqüência, restrições demasiadamente severas à aplicabilidade dos métodos e técnicas dela decorrentes.

1.3 - Objetivos

Mediante o exposto, este trabalho apresenta o estado da arte sobre técnicas de esparsidade relacionadas a sistemas estáticos de energia elétrica, especialmente aplicadas ao problema do fluxo de carga e fluxo de carga ótimo; mostra a evolução destas técnicas, desde a eliminação de *Gauss* simples até, mais recentemente, dos métodos de vetores esparsos, compensação e refatoração parcial; métodos de ordenação gerais e específicos para algumas das técnicas; considera a implementação de duas destas técnicas (Bi-fatoração e Eliminação Esparsa e Simétrica de Gauss) comparadas entre si e com um pacote da *Harwell Subroutine Library* - MA28; tudo isto alicerçado sobre uma fundamentação matemática básica.

Enfim, a reunião de todos estes itens aponta para o objetivo maior deste trabalho, que é subsidiar técnicos e pesquisadores da área de sistemas estáticos de energia elétrica, particularmente aqueles pertencentes ao grupo de pesquisas do LOSEP (Laboratório de Otimização em Sistemas Elétricos de Potência) do Departamento de Engenharia Elétrica da Escola de Engenharia de São Carlos, com conceitos, métodos e procedimentos desenvolvidos sob o título geral de "Técnicas de Esparsidade".

Objetiva-se, também, que este trabalho seja um norteador para estes especialistas, facilitando-lhes a decisão para o uso de alguma das técnicas abordadas.

O presente trabalho encontra-se dividido em 7 capítulos:

Capítulo 1 - *Introdução* - apresenta a motivação, áreas de estudo do assunto abordado, objetivos e sua organização.

Capítulo 2 - *Fundamentação Teórica* - é feita uma revisão de álgebra matricial enfocando os itens de maior interesse para a compreensão do estudo, bem como itens relativos à área de sistemas estáticos de energia elétrica.

Capítulo 3 - *Histórico* - apresenta o estado da arte relativo às técnicas de esparsidade através de um extenso levantamento bibliográfico, a fim de fornecer o suporte teórico necessário.

Capítulo 4 - *Ordenação* - discorre sobre as particularidades da ordenação da matriz, apresentando técnicas específicas para este fim.

Capítulo 5 - *Técnicas de Esparsidade* - apresenta 11 Técnicas de Esparsidade aplicadas a sistemas estáticos de energia elétrica.

Capítulo 6 - *Implementação e Testes* - discorre sobre a aquisição e armazenamento dos dados para execução dos testes, que são feitos através de uma rotina denominada Topologia da Rede, assim como as fases que compõem o processo de implementação dos algoritmos da Bi-fatoração e Eliminação Simétrica de *Gauss* e utilização

do pacote MA28. Os testes foram feitos para alguns problemas, apresentando-se seus resultados numéricos comparativos.

Capítulo 7 - *Conclusões* - retomando as considerações iniciais, são apresentadas as conclusões pertinentes.

2 - FUNDAMENTAÇÃO TEÓRICA

2.1 - Introdução ao Fluxo de Carga Ótimo

O fluxo de carga ótimo é utilizado como ferramenta para determinar o melhor ponto de operação de um sistema elétrico de potência (SEP), contribuindo para um melhor desempenho deste.

Trata-se de um problema de programação não linear, restrito, não convexo e de grande porte, com a seguinte formulação:

$$\begin{aligned} &\text{Minimizar } f(x) \\ &\text{sujeito a: } g(x) = 0 \\ &\qquad\qquad h(x) \leq 0 \\ &\qquad\qquad x_{mín} < x < x_{máx} \end{aligned} \tag{2.1}$$

sendo:

f(x) - função objetivo que representa o desempenho do sistema;

g(x) - vetor que corresponde às equações do problema do fluxo de carga;

h(x) - vetor que corresponde às inequações funcionais do problema;

x - vetor das variáveis de estado do problema.

São diversos os métodos existentes para sua solução, alguns caracterizados por transformar o problema original (2.1) em um problema irrestrito através da utilização de uma função auxiliar. Para os problemas de sistemas estáticos de energia elétrica, esta função é a Lagrangiana - La.

Aplicando as condições de otimalidade ao novo problema, obtém-se sistemas de equações não lineares que são tratados pelo método de *Newton-Raphson*, a fim de linearizar os sistemas.

Os sistemas lineares têm, então, a seguinte forma:

$$\nabla^2 La.\Delta\varphi = -\nabla La \tag{2.2}$$

sendo:

$\nabla^2 La$ - matriz Lagrangiana;

$\Delta\varphi$ - vetor de correções;

∇La - gradiente de La

A matriz Lagrangiana é quadrada, esparsa, estritamente simétrica (posições e valores) e com zeros na diagonal. Sua dimensão é determinada de acordo com o método de FCO que trata o problema. A tabela 2.1 (COSTA et al., 2000) apresenta, a título de exemplo, as dimensões e o número de elementos não zero para o sistema IEEE30, conforme o método empregado.

TABELA 2.1 - Dimensões da matriz $\nabla^2 La$ e número de elementos para o sistema IEEE30

Método	Dimensões de $\nabla^2 La$	Número de elementos não zero em $\nabla^2 La$	Grau de Esparsidade
ASP	**121**	**1.416**	**90,33%**
PDLB	**272**	**1.909**	**97,42%**
PD	**115**	**1.655**	**87,49%**

ASP - Conjunto Ativo e Penalidade

PDLB - Primal-Dual Barreira Logarítmica

PD - Primal-Dual

2.2 - Aspectos Matemáticos Gerais

Como verificado, os sistemas algébricos lineares, associados a problemas de análise da rede elétrica, são compostos de matrizes de coeficientes altamente esparsas e de grandes dimensões (2.2).

Tais sistemas lineares, não homogêneos, são representados, de uma forma geral, por:

$$Ax = b \tag{2.3}$$

sendo:

A = matriz dos coeficientes

x = vetor dependente (variáveis)

b = vetor independente (constantes)

Então $A \in \Re^{nxn}$ é uma matriz, não singular, com elementos:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \tag{2.4}$$

e x e b são vetores (n x 1) com componentes $(x_1, x_2, \dots, x_n)$ e $(b_1, b_2, \dots, b_n)$, respectivamente.

Resolver tais sistemas algébricos lineares, portanto, consiste no problema central da computação científica, e significa determinar uma *n-lista* de números reais ou complexos, caso ela exista, que satisfaça as n equações do sistema, simultaneamente.

Uma maneira de resolver a eq. (2.3) seria pela simples obtenção da inversa da matriz A, A^{-1}, ou seja: $x = A^{-1} . b$. No entanto, calcular explicitamente a matriz A^{-1} e, em seguida, efetuar o produto $A^{-1} . b$, é desaconselhável, pois o número de operações envolvidas é grande, o que torna este processo, do ponto de vista computacional, não

competitivo, uma vez que é extremamente necessário levar em consideração: tempo de processamento, memória necessária e esquema de armazenamento intermediário.

A afirmação feita por STEWART (1973, p.106) ressalta a preocupação com a inviabilidade do cálculo da inversa:

> " ... *we shall be at pains to stress that in most applications the computation of* A^{-1} *is both unnecessary and inordinately time consuming.* " .

Dessa forma, o uso de métodos numéricos para a solução da eq. (2.3) é indicado, mais especificamente aqueles que consistem em transformar o sistema linear apresentado, através de operações aritméticas elementares, até que seja obtido um sistema equivalente ao inicial, mas de solução imediata ou, pelo menos, mais simples. Tais métodos são denominados Métodos Diretos, dos quais o principal representante é o associado a *Gauss*.

Portanto, o *Método da Eliminação de Gauss* consiste em transformar, através de sucessivas eliminações de variáveis, o sistema dado em um sistema triangular, com resolução mais simplificada.

2.2.1 - Sistemas Triangulares

Um sistema triangular é aquele que possui a matriz dos coeficientes triangular superior ou inferior.

Uma matriz $A = (a_{ij})$ é triangular inferior se $a_{ij} = 0$ para $i < j$, tendo, então, a seguinte forma:

$$A = \begin{bmatrix} a_{11} & 0 & 0 & \dots & 0 \\ a_{21} & a_{22} & 0 & \dots & 0 \\ a_{31} & a_{32} & a_{33} & \dots & 0 \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix} \tag{2.5}$$

Similarmente, uma matriz triangular superior é aquela na qual $a_{ij} = 0$ para $i > j$.

A resolução de um sistema triangular obedece a dois algoritmos distintos, de acordo com sua característica.

Considerando um sistema triangular inferior $Lx = b$, o algoritmo pertinente é conhecido como "*FORWARD SUBSTITUTION*". O procedimento geral para obtenção dos valores de x_i é dado por:

$$x_i = \left(b_i - \sum_{j=1}^{i-1} l_{ij} x_j \right) \Big/ l_{ii} \tag{2.6}$$

sendo, l_{ij} o elemento da linha i e coluna j pertencente à matriz triangular inferior.

O algoritmo análogo para sistemas triangulares superiores, $Ux = b$, é chamado "*BACKWARD SUBSTITUTION*" . Sua fórmula para x_i é prescrita por:

$$x_i = \left(b_i - \sum_{j=i+1}^{n1} u_{ij} x_j \right) \Big/ u_{ii} \tag{2.7}$$

sendo, u_{ij} o elemento da linha i e coluna j pertencente à matriz triangular superior.

Deve-se salientar que os dois procedimentos gerais descritos nas eq. (2.6) e (2.7) , respectivamente, seguem uma ordem de acesso às matrizes, por linha, conhecida como *ROW ORIENTED VERSION* , isto é, a i-ésima linha é acessada no i-ésimo passo, sendo também conhecida como "forma do produto interno" .

De maneira similar, os procedimentos podem ser realizados considerando-se um acesso por colunas - *COLUMN ORIENTED VERSION* - bastando , para tanto, apenas a reversão da ordem de *loop*.

2.2.2 - Eliminação de *Gauss*

Apesar de o estudo das matrizes ter atraído maior atenção de pesquisadores depois do advento dos computadores (década de 50) , seus primórdios são do século XIX.

Conforme já abordado anteriormente, o Método de Eliminação de *Gauss* consiste em transformar, convenientemente, o sistema linear original, para obter um sistema linear equivalente com matriz dos coeficientes triangular superior.

A seqüência dos passos realizada até se chegar no sistema triangular superior, denominada "fase da eliminação" (*forward elimination*), é descrita a seguir para o seguinte sistema:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \cdot \quad \cdot \quad \cdot \quad \cdot \\ \cdot \quad \cdot \quad \cdot \quad \cdot \\ \cdot \quad \cdot \quad \cdot \quad \cdot \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n \end{cases} \tag{2.8}$$

FASE I:

Passo 1:

- Normalizar a linha 1 , ou seja, dividi-la por $a_{11} \neq 0$.
- Então, tem-se: $a_{1j} = a_{1j} / a_{11}$

  $b_1 = b_1 / a_{11}$ para $1 \leq j \leq n$

Passo 2:

- Eliminar x_1 da 2ª , 3ª , n-ésima equações. Isto pode se obtido através de combinações lineares apropriadas das equações, ou seja:

$$a_{ij} = a_{ij} - (a_{i1} / a_{11}) * a_{1j}$$

$$b_i = b_i - (a_{i1} / a_{11}) * b_i \qquad \begin{cases} \text{para } 1 \leq j \leq n \\ \text{e } 2 \leq i \leq n \end{cases}$$

Depois dessas operações, o sistema (2.8) torna-se:

$$\begin{cases} a'_{11}x_1 + a'_{12}x_2 + \ldots + a'_{1n}x_n = b'_1 \\ 0 \quad + a'_{22}x_2 + \ldots + a'_{2n}x_n = b'_2 \\ \cdot \quad \cdot \quad \cdot \quad \cdot \\ \cdot \quad \cdot \quad \cdot \quad \cdot \\ \cdot \quad \cdot \quad \cdot \quad \cdot \\ 0 \quad + a'_{n2}x_2 + \ldots + a'_{nn}x_n = b'_n \end{cases} \tag{2.9}$$

Vale observar que os coeficientes da 2ª , 3ª , ... , n-ésima equações são diferentes dos coeficientes originais.

Passo 3:

- Normalizar a linha 2
- Eliminar x_2 da 3ª , 4ª, ... , n-ésima equações, sem, entretanto, alterar os coeficientes da 1ª equação e os "zeros" da 1ª coluna, ou seja:

$$a'_{ij} = a'_{ij} - (a'_{i2} / a'_{22}) * a'_{2j}$$

$$b'_i = b'_i - (a'_{i2} / a'_{22}) * b'_i \qquad \begin{cases} \text{para } 2 \leq j \leq n \\ \quad \text{e } 3 \leq i \leq n \end{cases}$$

Assim, usando este tipo de operação, as variáveis x_1 , x_2 , ... , x_n, vão sendo eliminadas das colunas correspondentes, até que, finalmente, o sistema transforma-se em triangular superior, com a seguinte aparência:

$$\begin{cases} a''_{11}x_1 + a''_{12}x_2 + \ldots + a''_{1n}x_n = b''_1 \\ a''_{22}x_2 + \ldots + a''_{2n}x_n = b''_2 \\ \vdots \\ a''_{n-1,n-1}x_{n-1} + a''_{n-1,n}x_n = b''_{n-1} \\ a''_{nn}x_n = b''_n \end{cases} \tag{2.10}$$

FASE II:

Nesta fase, conhecida como "fase de substituição" , é que ocorre a resolução propriamente dita, isto é, a determinação dos valores das variáveis x_n , x_{n-1} , x_{n-2} , ... , x_1. Para tanto, utiliza-se o procedimento de "*BACKWARD SUBSTITUTION*" descrito na eq. (2.7).

É importante verificar que os elementos a_{ii} , citados nos passos correspondentes à normalização, constituem os "pivôs" de cada etapa do processo.

Cabe aqui, também, uma observação quanto às estratégias de pivoteamento - parcial ou completo - que são aplicadas para casos particulares de elemento pivô nulo ou próximo de zero. As descrições das duas estratégias encontram-se em GOLUB (1989, p.110 e 117), para os casos parcial e completo, respectivamente.

2.2.3 - Fatoração LU

Baseado no Método de Eliminação de Gauss, destaca-se este outro método direto de resolução de sistemas lineares, que consiste em decompor a Matriz *A* dos

coeficientes, em um produto de dois ou mais fatores: as matrizes L e U, e, em seguida, resolver uma seqüência de sistemas lineares, que conduzirá à solução do sistema original.

Tal método é particularmente utilizado para problemas que envolvem grandes variações do vetor b , como é o caso dos sistemas lineares associados a muitos problemas de engenharia elétrica. Neste tipo de problemas, a matriz A não sofre alterações, e aí está a grande vantagem dos métodos de fatoração; pois, uma vez calculados os fatores, basta aplicá-los para a resolução de inúmeros sistemas.

Como mencionado, então, a matriz A pode ser decomposta como:

$$A = L * U \tag{2.11}$$

onde L e U são os fatores, sendo:

L = matriz triangular inferior com diagonal unitária

U = matriz triangular superior

Assim, a solução do sistema original (2.3) é encontrada através de dois passos:

Passo 1:

- Reescrever $Ax = b$ através de seus fatores (L e U):

$$LUx = b \tag{2.12}$$

- Resolver, então, utilizando Eliminação de *Gauss*, o sistema intermediário:

$$Ly = b \tag{2.13}$$

através de "*FORWARD SUBSTITUTION*" - procedimento (2.6)

Passo 2:

- Utilizando-se do vetor y, calcular finalmente:

$$Ux = y \tag{2.14}$$

através de "*BACKWARD SUBSTITUTION*" - procedimento (2.7)

Portanto: $Ax = LUx = Ly = b$.

Na verdade, é importantíssimo destacar como são constituídos e obtidos os fatores L e U.

Considerando-se a matriz A dos coeficientes do sistema original descrito em (2.8), as fases para a construção dos fatores L e U são abordadas a seguir:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix} \tag{2.15}$$

FASE I:

Passo 1:

- Destacando o pivô $a_{11} \neq 0$, calcular os "multiplicadores" da coluna 1: m_{21}, m_{31}, ... , m_{n1}, componentes para a construção das "Matrizes de Transformação de *Gauss*", da seguinte forma:

$$m_{21} = a_{21} / a_{11} \; ; \; m_{31} = a_{31} / a_{11} \; ; \; \ldots \; ; \; m_{n1} = a_{n1} / a_{11}$$

- Utilizando-se dos multiplicadores, introduzir "zeros" na 1ª coluna, obtendo-se a matriz A', como segue:

$$A' = \begin{bmatrix} a'_{11} & a'_{12} & a'_{13} & \ldots & a'_{1n} \\ 0 & a'_{22} & a'_{23} & \ldots & a'_{2n} \\ 0 & a'_{32} & a'_{33} & \ldots & a'_{3n} \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ 0 & a'_{n2} & a'_{n3} & \ldots & a'_{nn} \end{bmatrix} \qquad (2.16)$$

- Sendo M_1 a matriz de transformação de *Gauss*, composta dos multiplicadores da 1ª coluna, A' é obtida através da multiplicação das matrizes M_1 e A. Assim:

$$\begin{bmatrix} 1 & 0 & \ldots & 0 \\ -m_{21} & 1 & \ldots & 0 \\ \ldots & \ldots & \ldots & 0 \\ -m_{n1} & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{bmatrix} = \begin{bmatrix} a'_{11} & a'_{21} & \ldots & a'_{1n} \\ 0 & a'_{22} & \ldots & a'_{2n} \\ 0 & \ldots & \ldots & \ldots \\ 0 & a'_{n2} & \ldots & a'_{nn} \end{bmatrix}$$

Logo: $M_1 * A = A'$ (2.17)

é a mesma matriz obtida no final da etapa 1 do processo de *Gauss*.

Passo 2:

- Continuando o processo para a coluna 2, os multiplicadores necessários são:

  $m_{32} = a_{32} / a_{22}$; $m_{42} = a_{42} / a_{22}$; ... ; $m_{n2} = a_{n2} / a_{22}$

- A'' é, então, obtida fazendo-se:

  $M_2 * A' = A''$ (2.18)

- Das eq. (2.17) e (2.18), vem:

  $M_2 * M_1 * A = A''$ (2.19)

  que é a mesma matriz obtida no final da etapa 2 do método de eliminação de *Gauss*.

Passo 3 até n-1:

- Prosseguindo dessa forma, chegaremos então à matriz $A^{(n-1)'}$, triangular superior, assim descrita:

  $M_{n-1} * M_{n-2} * \ldots * M_2 * M_1 * A = A^{(n-1)'}$

- Então:

$$A = (M_{n-1} * M_{n-2} * \ldots * M_2 * M_1)^{-1} * A^{(n-1)'} = (M_1)^{-1} * (M_2)^{-1} * \ldots * (M_{n-2})^{-1} * (M_{n-1})^{-1} * A^{(n-1)'}$$

$$A = \begin{bmatrix} 1 & 0 & 0 & \dots & \dots & 0 \\ m_{21} & 1 & 0 & \dots & \dots & 0 \\ m_{31} & m_{32} & 1 & 0 & \dots & 0 \\ m_{41} & m_{42} & m_{43} & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & 0 \\ m_{n1} & m_{n2} & \dots & \dots & m_{n,n-1} & 1 \end{bmatrix} * \begin{bmatrix} a'_{11} & a'_{12} & a'_{13} & \dots & \dots & a'_{1n} \\ 0 & a'_{22} & a'_{23} & \dots & \dots & a'_{2n} \\ 0 & 0 & a'_{33} & a'_{34} & \dots & a'_{3n} \\ 0 & 0 & 0 & a'_{44} & \dots & a'_{4n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & a'_{nn} \end{bmatrix} = LU$$

Ou seja:

$$L = (M_1)^{-1} * (M_2)^{-1} * \dots * (M_{n-2})^{-1} * (M_{n-1})^{-1}$$

$$U = A^{(n-1)'}$$

- Isto é, a matriz A foi fatorada em duas matrizes triangulares L e U, sendo que os elementos do fator L, " l_{ij} " para i > j , são os multiplicadores m_{ij} obtidos no processo de eliminação de *Gauss*. Os elementos do fator U são aqueles obtidos no final da fase de triangularização do método.

FASE II:

É nesta fase que o sistema é resolvido através da aplicação dos procedimentos (2.13) e (2.14).

Comparando os dois métodos, é importante observar que as combinações lineares, efetuadas durante o processo de eliminação de *Gauss,* podem ser interpretadas como operações com as Matrizes de Transformação de *Gauss*, ou seja, em um estágio

qualquer do processo de eliminação, para eliminar-se o elemento a_{ij} da matriz dos coeficientes, basta premultiplicá-la pela Matriz de Transformação de *Gauss* composta do multiplicador referente àquele elemento.

2.3 - Partição de Matrizes

Um método muito útil de manipulação de uma matriz, chamado *partição*, consiste em identificar várias partes de uma matriz como submatrizes que serão tratadas como simples elementos quando da aplicação das regras usuais de multiplicação e adição.

Em (2.20), a matriz de ordem 3 é particionada em quatro submatrizes pelas linhas tracejadas horizontal e vertical.

$$A=\begin{bmatrix} a_{11} & a_{12} & | & a_{13} \\ a_{21} & a_{22} & | & a_{23} \\ - & - & - & - \\ a_{31} & a_{32} & | & a_{33} \end{bmatrix} \tag{2.20}$$

Tal matriz A pode, então, ser escrita como:

$$A=\begin{bmatrix} D & E \\ F & G \end{bmatrix} \tag{2.21}$$

sendo as submatrizes:

$$D=\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \qquad E=\begin{bmatrix} a_{13} \\ a_{23} \end{bmatrix} \qquad F=\begin{bmatrix} a_{31} & a_{32} \end{bmatrix} \qquad G=a_{33} \tag{2.22}$$

Para indicar os passos para a multiplicação, em termos de submatrizes, assume-se que ***A*** deva ser pós-multiplicada por uma outra matriz ***B*** para formar o produto ***C***, tal que:

$$B=\begin{bmatrix} b_{11} \\ b_{21} \\ -- \\ b_{31} \end{bmatrix} \tag{2.23}$$

Sua respectiva partição e submatrizes são indicadas em (2.24):

$$B=\begin{bmatrix} H \\ J \end{bmatrix} \qquad H=\begin{bmatrix} b_{11} \\ b_{21} \end{bmatrix} \qquad J=b_{31} \tag{2.24}$$

Então, o produto é:

$$C=AB=\begin{bmatrix} D & E \\ F & G \end{bmatrix}\cdot\begin{bmatrix} H \\ J \end{bmatrix} \tag{2.25}$$

As submatrizes são consideradas como simples elementos para obter:

$$C=\begin{bmatrix} DH + EJ \\ FH + GJ \end{bmatrix} \tag{2.26}$$

O produto é finalmente determinado, executando-se as multiplicações e adições indicadas para as submatrizes.

As matrizes a serem multiplicadas devem ser compatíveis originariamente. Cada linha de partição vertical entre as colunas r e $r+1$ do primeiro fator requer uma linha de partição horizontal entre as linhas r e $r+1$ do segundo fator para que se efetue a multiplicação das submatrizes de modo conveniente. Linhas de partição horizontal podem

ser traçadas entre quaisquer linhas da matriz do primeiro fator e linhas verticais de partição entre quaisquer colunas do segundo fator ou ainda omitidas em uma delas ou em ambas.

A partição de matrizes foi amplamente utilizada por SUN et al. (1984) e por ALVARADO et al. (1990).

2.4 - Matrizes Esparsas

Como mencionado anteriormente, muitas das matrizes de coeficientes associadas aos sistemas lineares que traduzem, matematicamente, o comportamento de redes de energia elétrica, além de possuírem grandes dimensões, são altamente esparsas, isto é, apresentam pequena percentagem de elementos não nulos. A matriz Lagrangiana $\nabla^2 La$ é um exemplo.

A matriz Lagrangiana, particularmente, tem um alto "grau de esparsidade" , dado em função do número de barras e número de ramos, chegando a percentuais como 99,9%. Verifica-se, que em sistemas reais, o "grau de esparsidade" desta, cresce conforme aumentam as dimensões da rede elétrica.

O grau de esparsidade de uma matriz é definido como a porcentagem de elementos nulos dessa matriz. Para uma matriz de um sistema de NB barras e NR ramos, com referência no nó-terra, o grau de esparsidade, *GE*, é dado através da fórmula:

$$GE = \frac{NB^2 - (NB + 2NR)}{NB^2} \cdot 100\% \tag{2.27}$$

Assim, os sistemas lineares que possuem este tipo de matriz coeficiente são denominados *sistemas esparsos* e sua resolução faz-se através da utilização de metodologias específicas de solução denominadas *"técnicas de esparsidade"*.

Além da resolução em si, estas técnicas visam, do ponto de vista computacional, à redução de memória utilizada, armazenamento e tempo de processamento necessários à solução.

Como exemplo, consideremos, novamente, a matriz Lagrangiana e, ainda, um comprimento básico de palavra de computador, de 8 "bits". Para um sistema de 500 barras, teríamos então:

- para a alocação da matriz, sem considerar sua esparsidade, seriam necessários:

  $(500 \times 500 \times 8) / 1.024 \cong 2.000$ Kbytes;

- levando-se em conta um armazenamento esparso e um fator de incidência de 1,8 ligações por barra, tem-se cerca de 900 ligações, e, conseqüentemente:

  $(900 + 900 + 500) \times 8 / 1.024 \cong 18$ Kbytes.

Em suma, as técnicas de esparsidade, aplicadas a sistemas estáticos de energia, são metodologias específicas de solução de grandes sistemas esparsos, que têm, como principal objetivo, a maximização da eficiência numérica e computacional na obtenção destas soluções. Para tanto, exploram exaustivamente as características estruturais básicas das matrizes dos coeficientes envolvidas nos problemas de fluxo de carga e fluxo de carga ótimo.

3 - HISTÓRICO

A análise histórica sobre técnicas de solução de sistemas lineares esparsos retrata seus primórdios na década de 50. Porém, se for considerado que a grande maioria destas técnicas utiliza, como princípio básico, a Eliminação de Gauss, esta análise regride alguns anos no tempo.

A metodologia da Eliminação de *Gauss* é assim denominada em homenagem ao matemático alemão *Carl Friedrich Gauss* (1777-1855), que a introduziu. Ele é universalmente considerado como o maior matemático do século XIX e, ao lado de *Arquimedes* e *Isaac Newton*, como um dos maiores de todos os tempos (EVES, 1997).

Gauss recebeu o título de "Príncipe da Matemática" graças ao seu vasto legado nas áreas de Álgebra, Teoria dos Números, Geometria Diferencial, Magnetismo, entre outras, e

intenso incentivo ao progresso da matemática. É ele o responsável pela primeira demonstração plenamente satisfatória do Teorema Fundamental da Álgebra.

Concentrando-nos agora em técnicas mais recentes, encontramos em 1957 a primeira citação, relativa à estratégia de ordenação/pivoteamento de MARKOWITZ, indicada para matrizes assimétricas de grandes problemas gerais.

Na área de sistemas de potência, por sua vez, a primeira foi feita por SHIPLEY & COLEMAN (1959). Tal citação referia-se a um método direto, bastante eficiente, de inversão da matriz, que se mostrou superior a outros utilizados na mesma época. A principal característica do método era a exploração da estrutura esparsa e simétrica das matrizes, através do armazenamento e do processamento somente dos elementos diferentes de zero do triângulo superior das mesmas, o que contribuía para o ganho de velocidade e necessidade de memória, melhorando, de uma maneira geral, o tempo de processamento final.

Em 1961, VAN NESS & GRIFFIN anunciaram uma nova técnica conhecida como "Método de Eliminação para Solução do Fluxo de Potência" , baseada na adaptação do método de eliminação às equações dos sistemas de potência. Este método, implantado logo após sua publicação, além de outras características favoráveis, era reconhecidamente mais rápido e resolvia muitos problemas que, até então, apresentavam dificuldades de resolução.

Até a publicação desta técnica, os métodos para solução do fluxo de potência eram todos iterativos e baseavam-se em critérios de deslocamentos sucessivos, dos quais o principal era o de *Gauss-Seidel*. Este método, apesar da simplicidade de programação, número de cálculos relativamente pequeno e reduzidos requisitos de memória, era sujeito a sérios problemas de convergência, especialmente com a presença de alguns elementos de

reatância negativa e quando do aumento do número de barras do sistema, já que o tempo necessário para execução de cada passo do processo era uma função do número de barras.

A vantagem do método da eliminação sobre o iterativo, baseado em seu desempenho computacional mais veloz, foi expresso, através de um exemplo da época, para um problema de 24 barras: enquanto o tempo aproximado de resolução do sistema, pelo método iterativo, era de 51 minutos, o da eliminação era em torno de 6,8 minutos.

Apesar do sucesso alcançado pela técnica, as limitações começaram a surgir, já que os requisitos de memória e tempo de computação cresciam rapidamente com o aumento do tamanho dos sistemas. O limitante, em termos de memória (32 K para as máquinas da época), estava em torno de sistemas de 200 barras.

Foi, então, que as deficiências técnicas conduziram dois pesquisadores - TINNEY & SATO (1963) - a publicarem o artigo "Técnicas para exploração da Esparsidade da Matriz de Admitância de Rede" , que descrevia as idéias básicas de técnicas que ainda continuam presentes nos métodos atuais.

Tais idéias básicas resumem-se nos dois itens seguintes:

- determinação de uma seqüência de operações que resulta na diminuição do número destas, levando, conseqüentemente, à redução dos requisitos de memória e tempo de processamento;
- preservação dos operadores e da seqüência de operações para uso em soluções repetidas.

Estas técnicas dependem, de alguma maneira, da redução à forma triangular ou triangularização da matriz dos coeficientes. Assim, o número de operações com os elementos diferentes de zero e o número de termos adicionais introduzidos durante o processo (elementos *fill-in*), variam com a seqüência na qual as linhas são processadas. Esta seqüência, na verdade, é determinante na aplicação das técnicas.

Através da aplicação das técnicas propostas, significantes avanços em métodos de solução de muitos problemas de rede foram obtidos. Pode-se considerar, portanto, que estas "técnicas" representaram um "marco" no estudo das *técnicas de esparsidade.*

GAMM, KRUMM & SHER (1965) introduziram a ordenação da matriz através do esquema da Matriz de Banda.

Finalmente em 1967 , TINNEY & WALKER estenderam as técnicas descritas em 1963, dando-lhes forma definitiva e o nome de "*Método da Fatoração Triangular Otimamente Ordenada*" .

O método é composto de três partes:

- preservação dos operadores e da seqüência de operações para uso em soluções repetidas;
- determinação de uma seqüência ótima de operações de ordenação, que tende a conservar a esparsidade do sistema original;
- soluções diretas, que apresentam a solução propriamente dita.

O aperfeiçoamento do método está relacionado à segunda parte e, apesar da primeira parte já ter sido explorada anteriormente, suas possibilidades computacionais foram

aqui expandidas, com o emprego de técnicas de programação mais sofisticadas; no entanto, o armazenamento dos operadores em tabelas distintas continua prevalecendo.

Os autores, preocupados não somente com a ordenação, mas sim com o processo de *ordenação ótima*, introduzem os denominados "Critérios de Ordenação", subdivididos, de acordo com sua complexidade, em Esquema A ou 1, Esquema B ou 2 e Esquema C ou 3.

Enfim, para a solução de problemas de rede, estabelecendo uma comparação entre a determinação da inversa e a aplicação da forma fatorada, as vantagens da fatoração são indiscutíveis, principalmente quando a matriz dos coeficientes é esparsa. Algumas dessas vantagens são:

- a tabela de fatores pode ser obtida em menor tempo que a inversa; pois, ao contrário desta, a tabela de fatores de uma matriz esparsa pode ser quase tão esparsa quanto a matriz original;
- a exigência de armazenamento é pequena, permitindo que sistemas maiores sejam resolvidos;
- as soluções diretas podem ser obtidas rapidamente;
- a precisão dos resultados é consideravelmente melhor;
- as modificações necessárias para a organização da matriz podem ser realizadas muito mais rápido.

Portanto, as características da matriz de coeficientes - esparsa, simétrica e diagonalmente dominante - associadas ao desenvolvimento de técnicas especiais que visam armazenar e manipular somente elementos não triviais para manter a esparsidade da matriz dada, constituem os fatores básicos do sucesso da Fatoração Triangular Otimamente Ordenada.

Dentre os diversos estudos na área de sistemas de potência, que envolvem a manipulação de matrizes esparsas, TINNEY & HART (1967), em seu artigo intitulado "*Power Flow Solution by Newton's Method*" , exploraram os benefícios do método da Fatoração Triangular Otimamente Ordenada, proposto anteriormente.

Até então, para a resolução de problemas de fluxo de potência, o método utilizado era o de *Newton-Raphson* simplesmente, isto é, sem o tratamento prévio das matrizes esparsas associadas ao problema. Face às dificuldades encontradas diante do aumento de tamanho dos sistemas, percebeu-se que o entrave principal não estava no Método de *Newton-Raphson* em si, mas sim no método de resolução do sistema de equações lineares associado. O reconhecimento deste fato conduziu, então, a diversas pesquisas que resultaram na utilização da *Fatoração Triangular Otimamente Ordenada* para o tratamento anterior das matrizes de coeficientes, cuja eficiência nos resultados consagrou o Método de *Newton-Raphson* como o método mais poderoso para a solução do fluxo de potência e ainda possibilitou obter significativas reduções de memória, tempo computacional e precisão numérica.

Dando continuidade às pesquisas, no início da década de 70, OGBUOBIRI, TINNEY & WALKER desenvolveram um novo método de renumeração da rede através de seu rompimento em sub-redes minimamente interconectadas - *clusters* (grupos). Pelo estabelecimento de correspondências entre modelos de matrizes e grafos diretos, uma partição seqüencial binária é usada para decompor os nós do grafo em grupos.

Na realidade, a tentativa deste novo método era introduzir um novo algoritmo de ordenação ótima; pois, até então, os existentes eram relacionados aos esquemas de Matriz de Banda e de Ordenação Pseudoótima (Esquemas 1, 2 e 3). Entre esses quatro esquemas conhecidos, o Esquema 2 era o que se mostrava mais eficiente em termos de tempo de

processamento e otimalidade; porém, não identificava os grupos diretamente, razão pela qual o método de renumeração por grupos empregou o Esquema 3, notadamente eficiente na identificação de grupos bem definidos.

Infelizmente, baseado em testes realizados, ficou claro que o método não teve a repercussão esperada, pois a ordenação ótima não se mostrou prática para sistemas superiores a 200 nós. Mesmo assim, este método foi o embrião da associação de técnicas de esparsidade com Teoria dos Grafos, amplamente explorada posteriormente.

Nos anos de 1971 e 1972, através de BERRY e HSIEH & GHAUSI, respectivamente, algumas metodologias de ordenação foram sugeridas. A segunda proposta era uma estratégia para escolha do pivô baseada na predição probabilística dos *fill-ins*. Não tiveram muito impacto geral, pois eram voltadas para problemas mais específicos.

Foi, então, a busca da eficiência computacional aliada à obtenção de resultados mais consistentes, que levou ZOLLENKOPF (1971) a sugerir o método conhecido como "Bi-Fatoração" . Tal método é baseado na combinação das principais características de dois outros métodos: a *Decomposição Triangular* e o *Produto das Inversas*.

Este método possui algumas restrições: deve ser usado somente para matrizes esparsas definidas positivas que não tenham zero na diagonal principal e com estrutura de esparsidade estritamente simétrica, mas não necessariamente numérica.

Visando reduzir o tempo de processamento e necessidade de armazenamento, uma seqüência ótima de ordenação e pivoteamento, bem como um bom esquema de armazenamento e técnicas especiais de programação, são essenciais. O processo de

programação, portanto, é tão importante quanto o método em si. Para efeito didático, o processo todo foi dividido em três partes:

- Simulação do processo de redução e ordenação
- Redução
- Solução direta

O número total de operações de adição e multiplicação na determinação da solução direta para o método da bi-fatoração, é o mesmo que no método da decomposição triangular; entretanto, uma importante vantagem do método da bi-fatoração está na programação, porque a estrutura simétrica da matriz é completamente explorada. Por esta razão, o método da bi-fatoração requer somente metade das informações de indexação que o método da decomposição triangular.

Assim, pode-se dizer que as principais vantagens do método da Bi-Fatoração são:

- Apenas uma ordenação é necessária, ou seja, quando está sendo resolvido um sistema com várias iterações, a ordenação só é feita na primeira iteração, as outras são iguais a primeira, levando, assim, a um ganho em tempo de processamento;
- Utilização de pouca memória do computador, pelo fato de apenas os elementos diferentes de zero serem armazenados nas etapas do processo;
- Exploração, ao máximo, da estrutura simétrica de esparsidade da matriz;
- Necessidade de reduzidos requisitos de processamento para o procedimento de seleção do pivô;

- Aplicação de esquemas de armazenamento e programação sem qualquer renumeração ou rearranjo dos índices dos termos da matriz de acordo com a seqüência de pivoteamento definida.

Em 1973, TINNEY, POWELL & PETERSON introduzem uma maneira de obter-se um sistema equivalente para a redução da rede, com plena exploração da esparsidade, necessitando, para tanto, reter certos nós não essenciais que seriam normalmente eliminados. Na redução convencional, todos os nós não essenciais são freqüentemente eliminados, resultando em um sistema equivalente tão densamente interconectado que os métodos de matrizes esparsas não podem ser efetivamente utilizados. Assim, o trabalho proposto pelos autores identifica os principais fatores que afetam a esparsidade de redes equivalentes e descreve um algoritmo prático para determinação dos nós não essenciais que seriam retidos.

DUFF & REID (1974) realizaram um estudo comparativo somente com as estratégias de ordenação/pivoteamento conhecidas até então. Foram relacionadas, principalmente, as de MARKOWITZ e TINNEY 2.

Ainda em 1974, NAKHLA, SINGHAL & VLACH e HSIEH apresentaram alguns novos critérios de ordenação.

Até então todos os métodos sugeridos forneciam a solução do sistema esparso em sua totalidade. Porém, dependendo da aplicação, somente certos elementos explícitos de A^{-1} são necessários, motivação que levou ERISMAN & TINNEY (1975) a pesquisarem um algoritmo recursivo para o cálculo destes a partir dos fatores L e U, baseado nos relacionamentos de TAKAHASHI et al. (1973) na formação da matriz impedância e em suas aplicações ao estudo de curto-circuito e sensibilidade.

O algoritmo sugerido produz um novo relacionamento de dependência entre os elementos da inversa, que permite um cálculo eficiente de subconjuntos destes elementos. É possível obter uma coluna por vez de A^{-1} fazendo o vetor b igual a coluna apropriada da matriz identidade, destacando algum elemento específico desejado. Toma por base uma simples decomposição da matriz:

$$A^{-1} = D^{-1} L^{-1} + (I - U) A^{-1} \tag{3.1}$$

$$A^{-1} = U^{-1} D^{-1} + A^{-1} (I - L) \tag{3.2}$$

sendo $A = LDU$ padrão.

A extensão do algoritmo ao caso esparso geral não foi realizada, pois constatou-se que era específico para o cálculo de poucos elementos; todavia constituiu-se em prenúncio de pesquisas futuras.

A Empresa inglesa *AEA TECHNOLOGY Engineering Software* é a responsável pela Biblioteca de Subrotinas *Harwell* (HSL) e pela coleção de matrizes esparsas *Harwell-Boeing,* utilizadas para testes de software's. A HSL é uma coleção de pacotes em Linguagem Fortran, existente desde 1963, voltada para grande escala de cálculos científicos, escrita e desenvolvida pelo Grupo de Análise Numérica do Laboratório *Rutherford Appleton* e por outros especialistas e colaboradores. São inúmeros os pacotes disponíveis e encontram-se agrupados por área, sendo identificados pelas duas primeiras letras de seu nome. Os pacotes MA são os destinados para problemas de Álgebra Linear voltados para a solução de sistemas lineares. Dentre estes, vários são específicos para o manuseio de sistemas esparsos, destacando-se alguns que são utilizados na área de sistemas de energia elétrica. Destacam-se

então: o MA18 - CURTIS & REID (1971); MA28 - DUFF (1977); MA27 (1982), MA48 (1993) e MA47 - DUFF & REID (1995); MA57 e MA67 (2000).

Cada uma destas rotinas resolve problemas peculiares, como apresentado a seguir:

- MA18: sistemas esparsos assimétricos (o primeiro relacionado a sistemas esparsos);
- MA28: sistemas esparsos assimétricos: fatoração e solução (aperfeiçoamento do MA18);
- MA27: sistemas esparsos simétricos e não necessariamente definidos;
- MA48: sistemas esparsos assimétricos: método direto convencional;
- MA47: sistemas esparsos simétricos e indefinidos com zeros na diagonal da matriz A (aperfeiçoamento do MA27);
- MA57: sistemas esparsos simétricos e não definidos: método multifrontal;
- MA67: sistemas esparsos simétricos e indefinidos com zeros na diagonal, freqüentemente provenientes de problemas de otimização (aperfeiçoamento do MA47).

Como mencionado anteriormente, o MA18 (CURTIS & REID, 1971) foi o primeiro pacote para sistemas esparsos. Então, motivados em melhorar o desempenho e a flexibilidade do mesmo, DUFF & REID (1979) realizaram uma série de alterações em seu código, principalmente relativas à manipulação das estruturas de dados.

Quando de sua apresentação, o MA18 foi usado como uma rotina isolada, ou agregado a pacotes de programas em muitos centros de pesquisa pelo mundo. Apesar de seu vasto emprego, algumas críticas foram realizadas pelos usuários, tais como:

- a rotina não era facilmente portável, pois foi desenvolvida beneficiando-se das facilidades peculiares à máquina IBM 360/370;
- a interface de dados não era muito conveniente ao usuário, particularmente quando produzia uma matriz com modelo de não zero idêntico a alguma tratada anteriormente;
- a fase de decomposição inicial era lenta quando a forma fatorada possuía mais do que uma pequena proporção de não zeros por linha;
- a fase de solução era lenta.

Inicialmente são dois os passos que compõem o MA18, com suas respectivas funções:

- *ANALYSE*: escolha dos pivôs e identificação dos *fill-ins*, utilizando para o segundo passo a estratégia de MARKOWITZ (1957);
- *FACTOR*: manuseio dos pivôs e determinação real dos novos elementos, ou seja, a fatoração em si.

Para muitos conjuntos de equações com diferentes vetores $\boldsymbol{b}$ e matrizes A idênticas, o código provê uma outra subrotina independente denominada *OPERATE*, que se utiliza dos resultados, ainda disponíveis, do *ANALYSE* e *FACTOR*, para apresentar a nova solução.

Uma outra possibilidade é permutar a matriz A por blocos triangulares da forma:

$$PAQ = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ 0 & A_{22} & \dots & A_{2n} \\ 0 & 0 & . & . \\ . & . & . & . \\ . & . & . & . \\ 0 & 0 & \dots & A_{nn} \end{bmatrix} \tag{3.3}$$

sendo P e Q, matrizes de Permutação. Assim, um outro estágio, encarregado por encontrar os blocos triangulares, denominado *BLOCK*, é incrementado.

Em termos de tempo de processamento, a maior porcentagem é relativa aos estágios *ANALYSE* e *FACTOR*, como verificado na tabela 3.1, onde os tempos em milissegundos foram verificados para uma máquina IBM 370/168.

TABELA 3.1 - Comparações entre cronometragens de tempo para algumas matrizes

Ordem	**117**	**199**	**292**	**822**
Número de Elementos	**2.449**	**701**	**2.208**	**4.841**
BLOCK	**70**	**30**	**100**	**250**
ANALYSE	**1.470**	**210**	**960**	**480**
FACTOR	**280**	**50**	**210**	**180**
OPERATE	**20**	**10**	**23**	**40**

As alterações estruturais propostas pelos autores também foram incorporadas ao código MA28 - DUFF, recentemente operacionalizado (1977). Tal como o anterior, este pacote também objetiva a solução de sistemas assimétricos e é composto pelos mesmos

quatro estágios. Apesar das semelhanças, este novo pacote tem algumas vantagens, tais como:

- interface muito mais conveniente ao usuário;
- conferência de dados mais apropriada;
- manuseio de problemas mais gerais, incluindo matrizes A singulares e vetores b esparsos, bem como problemas com maior quantidade de elementos não zero em A;
- utilização da linguagem "*FORTRAN standard*" tornou-o mais portável.

A estrutura básica de armazenamento utilizada no MA28 é composta por listas encadeadas com ponteiros que avançam para frente e para trás.

As mesmas quatro matrizes da tabela 3.1, submetidas aos estágios do MA28, em comparação com o MA18, apresentaram cronometragens de tempo menores (milisegundos para uma máquina IBM 370/168), como pode ser observado na tabela 3.2.

TABELA 3.2 - Comparações entre cronometragens de tempo para algumas matrizes utilizando-se o MA18 e MA28

MATRIZ		*ANALYSE*			*FACTOR*			*OPERATE*		
N	NZ	MA18	MA28	MA28 + *BLOCK*	MA18	MA28	MA28 + *BLOCK*	MA18	MA28	MA28 + *BLOCK*
147	2.449	5.410	1.540	1.620	700	300	320	37	20	20
199	701	410	240	250	70	50	60	13	10	10
292	2.208	2.920	830	920	470	220	210	37	20	20
822	4.841	3.090	1.760	940	260	260	210	27	43	40

As diferenças de tempo entre as tabelas 3.1 e 3.2 são porque, na 3.2, os tempos foram tomados em execuções separadas, nas quais as cronometragens internas, relativas à tabela 3.1, não foram tomadas.

Alguns anos mais tarde - 1981 - EISENSTAT, SCHULTZ & SHERMAN, pesquisadores da área de Ciências da Computação, aproveitando o desenvolvimento verificado até então nas *técnicas de esparsidade* , propuseram uma nova metodologia para a procura de soluções para grandes sistemas, que possuem quatro características marcantes: a matriz dos coeficientes é quadrada, esparsa, simétrica e definida positiva. Portanto, tal metodologia pertence ao rol das técnicas estudadas até aqui, e é referenciada como *Eliminação Esparsa e Simétrica de Gauss.*

A técnica é descrita em termos de algoritmos bastante eficientes, que podem ser implementados mediante pouca exigência computacional, e objetivam a redução dos requisitos de memória e tempo de processamento. Para tanto, utiliza, como ferramenta básica, a Eliminação Simétrica de *Gauss.* Para o sistema linear $\boldsymbol{Ax} = \boldsymbol{b}$ (2.3), temos que $\boldsymbol{A}$ pode ser sempre fatorada na forma:

$$\boldsymbol{A} = \boldsymbol{U}^t . \boldsymbol{D} . \boldsymbol{U} \tag{3.4}$$

sendo: $\boldsymbol{U}$ a matriz unitária triangular superior

$\boldsymbol{D}$ a matriz diagonal.

A Eliminação de *Gauss*, portanto, é equivalente à primeira fatoração de $\boldsymbol{A}$ desta maneira e, então, a obtenção de $\boldsymbol{x}$ é dada através da resolução sucessiva dos sistemas:

$$U^t y = b \quad , \quad Dz = y \quad , \quad Ux = z \tag{3.5}$$

Para melhorar o desempenho computacional, deve-se salientar que é desejável fatorar A e obter x, sem armazenar e operar os elementos nulos de A e U. Para tanto, utilizam-se estruturas de armazenamento bastante compactas, compostas de listas encadeadas e vetores, e, ainda, os algoritmos usam uma maneira eficiente de minimizar os *loops* de repetição e os testes (para verificar elementos diferentes de zero) através do manuseio de conjuntos, que são baseados em informações das matrizes A e U, e que constituem o grande diferencial desta técnica.

A implementação da técnica consta de um pacote voltado para a solução de sistemas lineares esparsos, denominado *YSMP - YALE SPARSE MATRIX PACKAGE*. No artigo de 1982, os autores EISENSTAT, GURSKY, SCHULTZ & SHERMAN dão todos os parâmetros para a utilização do pacote sob os mais diversos pontos de vista. Neste caso, é apresentado o pacote *YSMP I - O código simétrico.*

Em 1985, TINNEY, BRANDWAJN & CHAN introduziram os Métodos de Vetores Esparsos para os problemas relativos a sistemas de potência. Estes métodos enfatizam a eficiência dos algoritmos de solução de sistemas esparsos e também a obtenção de somente um subconjunto do vetor desconhecido x, pela exploração da esparsidade do vetor independente b. Tais características já haviam sido exploradas por HACHTEL, BRAYTON & GUSTAVSON (1971) e por DUFF & REID (1975), com ênfase para outros tipos de problemas.

A motivação primeira para sua utilização foi a aquisição de soluções mais rápidas para problemas que contam com vetores esparsos, tais como, fluxo de potência ótimo, despacho econômico, análise de erros e de contingência.

A idéia fundamental é que muitos problemas de sistemas de potência requerem soluções repetidas para grandes conjuntos de equações lineares esparsas, onde o vetor $\boldsymbol{b}$ é também esparso. Entretanto, o vetor solução x não é esparso em geral. O termo "vetor esparso", portanto, refere-se ou ao vetor esparso $\boldsymbol{b}$ ou a um subconjunto do vetor $\boldsymbol{x}$ contendo somente os elementos de interesse. O exato significado é sempre claro para o contexto.

O processo todo constitui-se, na verdade, de uma maneira criativa e muito rápida de localização dos elementos dentro da matriz, sem necessidade de pesquisa, fazendo uso de uma tabela chamada *Path Table*, que é definida como uma lista ordenada dos elementos não zero para as colunas de L (ou linhas de U) e apresenta, para cada nó da coluna k da matriz L (ou linha k da matriz U), o próximo elemento não zero.

Os métodos de vetores esparsos constituíram um ícone no estudo das técnicas de esparsidade; pois, grande parte das pesquisas posteriores basearam-se em seus conceitos. Estes são completamente gerais e podem ser aplicados em associação com outras técnicas ou a qualquer programa que busque a solução de sistemas lineares utilizando vetores esparsos.

Ainda no mesmo ano, 1985, ALVARADO, MONG & ENNS seguem com a idéia de Vetores Esparsos, sugerindo um algoritmo para a solução da maioria dos problemas de curto circuito, onde faltas simultâneas em paralelo combinadas com saídas, linhas não balanceadas, cargas e faltas em série podem ser resolvidas com a mesma tranqüilidade que uma simples falta trifásica. Tal algoritmo foi considerado computacionalmente eficiente, pois utilizava-se de compensação e soluções esparsas *forward* e *back*. A implementação do algoritmo é um diferencial; pois, utilizando-se das possibilidades computacionais da época, permitiu ao usuário interagir com a máquina, através de simulações em seu monitor, fazendo uso de macros que o programa oferecia.

Em fevereiro de 1986, CHAN & BRANDWAJN introduziram dois novos métodos de atualização dos fatores ***LDU*** da matriz através de refatoração parcial. Ambos os métodos, identificados como *PR1* e *PR2*, têm grande impacto sobre aplicações nas quais a fatoração da matriz representa significativa porcentagem do esforço computacional. Muitos problemas em sistemas de potência caem nesta categoria, destacando-se: o fluxo de potência por Newton; desacoplado rápido; análise de segurança, estabilidade transitória, fluxo de potência ótimo e estimação de estado. Seu uso apropriado, aliado a uma programação habilidosa, podem melhorar significativamente as aplicações citadas. No caso da solução do fluxo de potência por Newton, a velocidade chega a ser duplicada.

Estes dois métodos estão intimamente relacionados aos conceitos de vetores esparsos (*path's*), recentemente abordados. O desenvolvimento de ambos foi motivado pela observação de que as alterações em uma linha/coluna da matriz afetam somente um pequeno subconjunto de linhas/colunas subseqüentes. Estão baseados em outros cinco métodos de refatoração, que produziam resultados inferiores: Refatoração Completa, Métodos de Compensação, Refatoração Parcial Simples, Refatoração Parcial Simples com Ordenação Especial e Método de Benett.

As melhorias dos métodos de refatoração parcial 1 e 2 originam-se de dois atributos:

1. A matriz original é ordenada sem considerar as modificações futuras. Nenhum nó escapa da ordenação esparsa direcionada.
2. Os métodos de vetores esparsos são usados para identificar os fatores da matriz afetados pelas mudanças da mesma. Somente estes fatores são atualizados.

As diferenças básicas entre *PR1* e *PR2* são:

- O primeiro usa o *path* para, seletivamente, refatorar um subconjunto de linhas/colunas da matriz;
- O segundo usa o *path* para, seletivamente, atualizar os fatores da matriz.

Nos testes realizados para grandes matrizes simétricas do tipo admitância, verificou-se que *PR2* é mais rápido que *PR1*, se as modificações da matriz são de rank menor ou igual a 8. Caso contrário, *PR1* é mais eficiente.

No ano de 1988, três novos métodos de ordenação foram introduzidos. Dois deles de GÓMEZ & FRANQUELO, visam à melhoria dos métodos de vetores esparsos e estabelecem comparações com o algoritmo de mínimo grau (TINNEY 2).

BETANCOURT, por sua vez, oferece um novo esquema de ordenação para trabalhar em conjunto com a refatoração parcial. Trata-se de um algoritmo heurístico baseado em simples manipulações de grafos.

Seguindo com as pesquisas sobre refatoração parcial, RISTANOVIC', BJERLOGRLIC' & BABIC' (1989) propõem várias melhorias analíticas e computacionais às aplicações com esparsidade, apresentando um novo método de refatoração parcial da matriz (*PMR*) e algoritmo de ordenação. A metodologia proposta é aplicada ao fluxo de potência desacoplado rápido que inclui o tratamento das violações *tap's* - ULTC e geração de potência reativa nas barras PV.

De acordo com os autores, o método proposto é mais rápido que os existentes, principalmente para casos de um número razoavelmente pequeno de mudanças simultâneas. Usando conceitos de vetores esparsos, não requer pesquisa durante a refatoração nem a re-leitura de colunas e linhas do *path* de fatoração.

O algoritmo de ordenação que incrementa a eficiência do método é tal que, a cada passo, o nó pivô é aquele que minimiza os *fill-ins* e o número de operações de ponto flutuante sobre os *path's* de fatoração.

Em associação ao método há a criação de um programa muito rápido para o fluxo de carga *on-line*. Quando técnicas de compensação média para simulação de remoção/adição de linhas são incorporadas, o programa proposto é muito conveniente para análise de contingência AC completo.

No trabalho desenvolvido por ENNS, TINNEY & ALVARADO (1990), as propriedades estruturais e computacionais das matrizes esparsas são analisadas, especificamente as relacionadas com as inversas dos seus fatores produzidos durante a fatoração ou decomposição, enfocando também a partição das matrizes para reduzir a formação de elementos não zero nos fatores inversos. Estes fatores são também esparsos e emprestam-se a operações paralelas na fase de soluções diretas ou repetidas dos problemas. Todas as multiplicações necessárias para as soluções repetidas podem ser executadas em paralelo, usando os fatores inversos, enquanto a fatoração é um processo essencialmente serial.

Neste trabalho também foi introduzido um novo esquema de ordenação conhecido como ESQUEMA W. Este surgiu dos esforços pela busca de melhor desempenho dos métodos de vetores esparsos pela redução do comprimento médio do *path* de fatoração durante as soluções repetidas.

Assim sendo, verifica-se que, para grandes problemas de redes de potência e também elementos finitos, o novo esquema de ordenação proposto é bastante benéfico para muitas soluções repetidas com vetores esparsos.

Também no ano de 1990, ALVARADO, YU & BETANCOURT propuseram a solução da eq. 2.3 pela construção de componentes fatoradas das inversas de L e U (fatores triangulares de A), baseada na idéia das matrizes W particionadas.

Este trabalho descreve um novo procedimento com matrizes esparsas que agrupa quatro itens: ordenação, partição, construção das matrizes W e uso das matrizes W ao invés de soluções repetidas.

A metodologia proposta está inteiramente relacionada aos métodos de vetores esparsos. Segundo os autores, é possível ter "zero" *fill-ins* através dos algoritmos de inversa particionada com surpreendente baixo custo computacional. Um acesso à partição W sem produzir elementos adicionais está baseada no *path graph* associado ao sistema. Se todos os nós do mesmo nível são numerados primeiro que todos os nós do próximo nível, particionando a matriz de acordo com os níveis resulta em nenhum *fill-in* em W, desde que os nós dentro de uma partição não sejam predecessores de outros. Dependendo da necessidade, quando nenhum *fill-in* é produzido, o número de partições varia de 6 a 15; se 20% de *fill-in* são permitidos, então o número de partições pode ser menor que 3. Portanto, a partição é a chave da eficiência do método.

Retomando a conceituação dos métodos dos vetores esparsos em toda a sua plenitude, BACHER, EJEBE & TINNEY (1991) apresentaram quatro novas técnicas denominadas "Aproximação do Vetor Esparso", especialmente aplicáveis para grandes problemas de redes de potência para os quais a rapidez das soluções é particularmente importante e também é vantajoso preservar um certo montante de precisão para aumento da velocidade.

Os métodos de vetores esparsos aumentam a velocidade dos algoritmos usando o princípio de "pulos" de operações. Por exemplo, ele pula as operações nulas na substituição *forward* dos vetores esparsos; pula as operações desnecessárias na substituição *back* para certas seleções e pula operações nulas na modificação dos fatores da matriz para refletir as mudanças da matriz. Assim, pular operações cujos resultados são suficientemente pequenos, de acordo com alguns critérios pré-definidos (e que não podem ser negligenciados), constitui a base das técnicas de aproximação.

Das quatro técnicas apresentadas, três são aproximações das técnicas de vetores esparsos já estabelecidas: *SKIP FORWARD* (*Fast Forward*), *SKIP BACK* (*Fast Back*) e *SKIP FACTOR UPDATE* (*Factor Update*). A quarta técnica, *SKIP BACK BY COLUMNS*, é inteiramente nova e não tem correlação com os vetores esparsos, sendo uma aproximação da substituição *back* cheia por colunas.

Pode-se dizer, assim, que as técnicas de aproximação do vetor esparso suplementam, mas não substituem, as técnicas de vetores esparsos. O uso de uma ou outra é particularmente específico aos problemas. Em geral, para grandes aplicações, as técnicas de aproximação apresentam resultados mais vantajosos.

VEMPATI, SLUTSKER & TINNEY (1992) associam os métodos de vetores esparsos à fatoração ortogonal, a mais estável estratégia numérica, para estimação de estado de sistemas de potência, originando os Métodos de Vetores Esparsos Ortogonais. Estes agilizam os algoritmos de estimação ortogonal de diversas maneiras. Trata-se de uma técnica muito particular para problemas peculiares de estimação de estado e localização adaptativa, não sendo indicado para outras situações.

ZANG & TINNEY (1995) retomam os estudos dos métodos de refatoração parcial apresentando uma nova versão de um programa específico para esta técnica, com alterações em sua formulação, tornando-o mais veloz no processamento das soluções de redes, pois elimina todas as restrições sobre algum número ou tipo de mudança nos parâmetros ou topologia da rede. Não envolve aproximações, transgressões exatas de modelos, mas preserva a esparsidade da matriz.

Tal como as técnicas de esparsidade, a refatoração parcial é, primariamente, uma técnica de programação e, para tanto, os autores sugerem uma maneira de implementar esta nova versão, dando detalhes sobre armazenamento, concentrando-se na modelagem e uso de vetores indexados básicos e adicionais e, ainda, aspectos sobre a programação em si, onde as técnicas de listas encadeadas padrão são utilizadas para guiar todas as operações de fatoração.

As vantagens da nova versão dependem da sua aplicação a problemas específicos, rendendo, para tais, grande economia de tempo de processamento.

Introduzindo mais um pacote de códigos para solução de sistemas lineares esparsos, DUFF & REID (1996a) discutem a metodologia do MA47 (1995), um conjunto de rotinas específico para o caso indefinido, cuja matriz A é simétrica e tem um significativo número de zeros na diagonal.

Dentre as rotinas desenvolvidas pelos autores, a MA27 (1982) - indicada para solução de sistemas simétricos - pode ser considerada a predecessora da atual. Aquela utiliza uma técnica de solução multifrontal e necessita de adaptações para o manuseio de matrizes indefinidas. Na fase preliminar de análise, que escolhe a seqüência de pivoteamento a partir do modelo de esparsidade, assume-se que a matriz é definida; então, todas as entradas da

diagonal são diferentes de zero. Para o caso indefinido, esta seqüência de pivoteamento é modificada na fase de fatoração para manter a estabilidade pelo retardamento no uso do pivô, se é muito pequeno, ou substituição de dois pivôs por um bloco 2 x 2. Os problemas, cuja matriz possui zeros na diagonal, submetidos ao MA27, encontram dificuldades com relação aos *fill-ins* que, durante a fase de fatoração, aparecem em quantidade significativamente maior que o previsto na fase de análise. Assim, devido a casos especiais como este, é que o código MA47 foi adotado. Também usa um método multifrontal e, graças às suas especificidades, oferece muitas oportunidades para simplificação no manuseio dos problemas com grande melhoria na eficiência e velocidade de execução, particularmente em máquinas vetoriais e paralelas.

O pacote é composto por quatro subrotinas:

- *INITIALIZE*: atribui valores-padrão para os vetores, que juntos controlam a execução do pacote;
- *ANALYSE*: acessa o modelo de esparsidade de A e escolhe um bloco de pivôs. Também calcula outros dados necessários à fatoração;
- *FACTORIZE*: acessa a matriz A junto com o conjunto dos blocos de pivôs recomendados. Executa a fatoração incluindo permutações adicionais, necessárias para a estabilidade numérica;
- *SOLVE*: utiliza os resultados da subrotina anterior para apresentar a solução do sistema.

A intenção original dos autores era substituir o código MA27 pelo MA47, entretanto este último foi penalizado por sua complexidade adicional de não ser capaz de aproveitar-se das vantagens estruturais da matriz. Em testes realizados para um conjunto de dez matrizes com zeros na diagonal da coleção de matrizes esparsas *Harwell-Boeing* -

DUFF, GRIMES & LEWIS (1992), houve oscilação entre os resultados dos dois códigos: ora o MA27 tinha um desempenho melhor, ora o MA47.

Em suma, estes resultados indicaram que a eficiência do MA47 é muito dependente dos detalhes de montagem da estrutura da árvore associada à matriz. De qualquer forma, para problemas bem comportados, sua eficiência é indiscutível.

Visando à obtenção de melhores resultados com o método dos vetores esparsos, LEE & CHEN (1993) propõem um novo algoritmo de ordenação que minimiza localmente o aparecimento de novos elementos na inversa da matriz triangular inferior durante o processo de fatoração. Nos testes realizados para comprovar a eficiência do método, os autores concluíram que o mesmo também é de grande utilidade no desenvolvimento dos métodos de refatoração parcial.

DUFF & REID (1996b) descrevem os princípios de funcionamento do pacote MA48 (1993), também destinado à solução de sistemas assimétricos, tal qual o MA18 e MA28. Segundo os autores, cada novo pacote visa substituir os anteriores, objetivo esse nem sempre alcançado, por problemas de ordem técnica. Assim, existem usuários distribuídos pelo mundo, utilizando as mais variadas versões dos códigos MA.

As inovações introduzidas no código do MA48 estão relacionadas à reestruturação das fases simbólica e numérica com o incremento da velocidade e necessidade de armazenamento, sem sacrificar a estabilidade numérica. Tais características são atribuídas aos itens abaixo relacionados:

- o algoritmo é rápido e robusto;
- a interface é amigável;

- o código muda o processamento para matriz cheia quando a matriz reduzida é suficientemente densa, usando Subprogramas de Álgebra Linear Básica (BLAS), níveis 1, 2 e 3 (DONGARRA et al., 1990);
- a seqüência de pivoteamento é normalmente escolhida automaticamente, de qualquer parte da matriz, mas ainda pode ser limitada à diagonal ou ainda ser específica;
- caso a alocação de espaço de armazenamento seja insuficiente, o algoritmo evolui até o final para estimar a porção necessária;
- o código determina os blocos triangulares e faz uso deles;
- matrizes singulares ou retangulares são permitidas;
- refinamentos iterativos da solução estão disponíveis para melhorar a precisão ou estimação do erro.

O coração do MA48 encontra-se em uma subrotina separada chamada MA50, que é acionada para efetuar o processamento numérico dos blocos de A.

Dois novos métodos de ordenação - MFI1 e MFI2 - que podem ser usados para reduzir o número de elementos nos fatores inversos da matriz esparsa, foram propostos por WANG & GOOI (1997). Estes são baseados na diagonalização de A através do uso de uma matriz de transformação C. Ao invés da convencional decomposição LDU, seguida de uma série de multiplicações pela matriz W, a matriz C gera, em somente um simples estágio, uma nova seqüência de nós para a rede e todos os elementos desta.

Continuando com suas pesquisas sobre algoritmos de ordenação, GOOI & WANG (1998) propõem mais dois novos: MFI1-ML e MFI2-ML, também baseados na matriz de transformação C. O desempenho destes, e dos dois propostos anteriormente, foi comparado com outros sete algoritmos de ordenação para cinco sistemas. Em termos de tempo de

computação através da solução via método dos vetores esparsos, os resultados obtidos dos algoritmos propostos para todos os sistemas foram sempre os menores.

No trabalho desenvolvido por ALVES, ASADA & MONTICELLI (1999), uma análise crítica foi realizada sobre a comparação do desempenho de métodos diretos e iterativos de solução de sistemas lineares, para o cálculo do fluxo de potência e análise de contingência.

Inicialmente, algumas diferenças básicas entre os dois métodos são apontadas pelos autores:

- por métodos diretos entende-se preservação da esparsidade na eliminação de *Gauss* baseada em aproximações; por métodos iterativos entende-se o pré-condicionamento dos métodos de Gradiente Conjugado, tal como os baseados na fatoração incompleta da matriz A (como *Cholesky* incompleto e pré-condicionadores K_0);
- os métodos iterativos são bem maleáveis quando utilizados com paralelismo e vetorização; os métodos diretos são conhecidos por serem mais rígidos para se encaixarem nas operações em paralelo;
- os esquemas de ordenação para preservação da esparsidade são uma parte crucial dos métodos diretos (TINNEY *2*) e estão inteiramente relacionados à fatoração completa dos coeficientes da matriz A; para os métodos iterativos, os esquemas de ordenação não são normalmente aplicáveis, pois estes são baseados na fatoração incompleta;
- enquanto o objetivo dos métodos diretos é, geralmente, minimizar o armazenamento e o número de operações aritméticas, o foco dos métodos

iterativos, onde os *fill-ins* não são considerados, está relacionado com os parâmetros da rede e estrutura da matriz.

Para que fosse possível, e mais significativa, a comparação entre os métodos direto e iterativo, foi necessário submeter os testes às mesmas condições gerais, sendo, a ordenação, o ponto crítico. Por este motivo, um novo esquema de ordenação, adequado ao método iterativo, foi proposto pelos autores. Segundo eles, este novo esquema traz para os métodos iterativos os mesmos benefícios que o esquema de TINNEY *2* traz para os métodos diretos.

Embora a comparação fosse importante, a causa principal do novo esquema de ordenação foi funcional, isto é, a preocupação com a convergência do método do Gradiente Conjugado Pré-condicionado, pois apesar dos *fill-ins* não serem considerados, os parâmetros da rede podem levar a divergências. Assim, o algoritmo *"Minimum Spanning Tree"* foi aplicado. A idéia fundamental do mesmo é obter uma árvore extendida contendo as ligações mais fortes da rede e então numerar os nós da árvore de maneira que as extremidades da árvore são processadas primeiro; uma vez que as extremidades são processadas, as mesmas são descartadas e novas extremidades são numeradas até não restar extremidade alguma.

Os testes foram realizados levando em consideração o método iterativo do Gradiente Conjugado Pré-Condicionado aliado ao novo esquema de ordenação proposto e o método direto aplicado foi baseado em procedimentos da eliminação gaussiana em associação com o esquema de ordenação TINNEY 2. Os sistemas submetidos aos métodos foram IEEE-30, IEEE-57, IEEE-118, IEEE-300, BR-810, BR-1663 e a conclusão geral é que os métodos diretos ainda são mais rápidos, apesar de os métodos iterativos serem mais fáceis de implementar. Um fato interessante observado foi que, mudando de máquina, o método iterativo dava melhores resultados, pois a arquitetura desta outra máquina (*RISC*)

tinha algumas características que favoreciam a implementação dos produtos internos de vetores, necessários aos métodos de Gradiente Conjugado Pré-Condicionado.

Dessa maneira, graças às comparações, um novo esquema de ordenação, baseado na fatoração incompleta, foi sugerido e, ainda, um novo esquema de pré-condicionamento, baseado na fatoração completa, foi utilizado especificamente para problemas de análise de contingência.

Em 2000, dentre vários novos pacotes, a Biblioteca de Subrotinas *Harwell* publicou mais dois relativos a resolução de sistemas esparsos, que se encaixam nos padrões de problemas de sistemas de energia elétrica. São eles:

- MA57: resolve sistemas de equações cuja matriz é esparsa, simétrica e não definida. É considerada uma atualização da rotina MA27 (1982). A principal vantagem é o uso da BLAS nível 3 (DONGARRA et al., 1990). Além disso, somada à funcionalidade do MA27, este novo código inclui: a capacidade de reiniciar a fatoração após uma parada por deficiências de memória; a incorporação da estratégia de modificar a matriz para definida positiva durante a fatoração; salvar múltiplos vetores $\boldsymbol{b}$; retornar estimativas de erro e executar refinamentos iterativamente.
- MA67: resolve sistemas de equações cuja matriz é esparsa, simétrica, não definida e com zeros na diagonal, freqüentemente originadas por problemas de otimização. Este usa as idéias do MA47 (1995), que tinha a desvantagem de explorar o modelo de esparsidade antecipadamente, não sendo muito eficiente para a fatoração atualizada com números reais. O MA67 evita isto pela avaliação numérica dos elementos, quando da escolha da seqüência de pivoteamento.

4 - ORDENAÇÃO

Não se pode falar em "Técnicas de Esparsidade" sem citar a metodologia da ordenação. Esta não é somente uma parte integrante das técnicas, por vezes chega a ser seu âmago.

Por este motivo, este capítulo é destinado à exploração dos esquemas de ordenação empregados nas várias técnicas abordadas.

4.1 - Aspectos Gerais da Ordenação

Na busca da solução de um sistema esparso através da Eliminação de *Gauss* (ou redução à forma triangular), o objetivo normal é escolher uma seqüência de

pivoteamento na qual o número de operações aritméticas e/ou o número total de elementos gerados adicionalmente durante o processo - *fill-ins* - seja mínimo.

O termo ordenação é bastante abrangente, uma vez que representa a associação dos efeitos da seqüência de operações no processo da eliminação Gaussiana ao processo de redução física da rede elétrica. Pode-se dizer que a ordenação da matriz esparsa é simplesmente a renumeração dos nós em um grafo para modificar as características da matriz correspondente.

Enfim, independente do termo empregado, o processo da ordenação tem três finalidades marcantes:

- o controle da precisão numérica visando à manutenção da estabilidade do sistema;
- a preservação da esparsidade da matriz;
- a otimização das estruturas de dados.

Em geral, estes objetivos não são compatíveis. No entanto, a natureza física das redes elétricas e a precisão numérica dos computadores, tende a eliminar a preocupação excessiva com eficiência numérica e aumentar a necessidade de exploração da esparsidade e capacidade de armazenamento.

Dessa maneira, a seqüência na qual as linhas são processadas é determinante na aplicação das técnicas de esparsidade.

Tomando como exemplo a rede da figura 4.1, observa-se que a mesma consiste de 4 barras independentes numeradas de 1 a 4, sendo 0 a barra de referência.

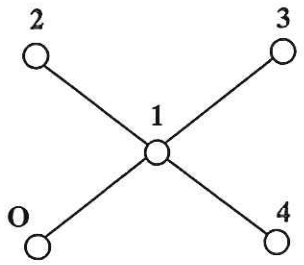


FIGURA 4.1 - Rede com 5 nós

A matriz YBARRA, associada à rede exemplo, é expressa em (4.1).

$$Y = \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} X & X & X & X \\ X & X & & \\ X & & X & \\ X & & & X \end{bmatrix} \end{array} \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \qquad (4.1)$$

A eq. (4.2) apresenta a matriz Y após a redução da primeira linha.

$$Y' = \begin{bmatrix} 1 & X & X & X \\ & X & \otimes & \otimes \\ & \otimes & X & \otimes \\ & \otimes & \otimes & X \end{bmatrix} \qquad (4.2)$$

Observa-se na matriz Y' que, durante o processo de eliminação, foram introduzidos elementos onde, inicialmente, não existiam. Tais elementos estão representados na matriz (4.2) pelo símbolo $\otimes$.

Seguindo com o processo de redução à forma triangular - eliminação - obtém-se, ao final, a matriz A - triangular superior - como segue:

$$A = \begin{bmatrix} 1 & X & X & X \\ & 1 & \otimes & \otimes \\ & & 1 & \otimes \\ & & & 1 \end{bmatrix} \tag{4.3}$$

Verifica-se que as operações de redução foram executadas na ordem em que a matriz foi, inicialmente, disposta. Fazendo, então, o processamento da quarta linha em primeiro lugar, que é equivalente a fazer a inversão das linhas e colunas 1 e 4, obtém-se a matriz representada em (4.4):

$$Y = \begin{array}{c} \begin{array}{cccc} 4 & 2 & 3 & 1 \end{array} \\ \begin{bmatrix} X & & & X \\ & X & & X \\ & & X & X \\ X & X & X & X \end{bmatrix} \end{array} \begin{array}{c} \\ 4 \\ 2 \\ 3 \\ 1 \end{array} \tag{4.4}$$

As próximas matrizes (4.5) e (4.6) apresentam, respectivamente, os resultados após o processamento da primeira linha (quarta linha, na realidade) - matriz Y' - e, finalmente, a matriz triangular superior A :

$$Y' = \begin{bmatrix} 1 & & & X \\ & X & & X \\ & & X & X \\ & X & X & X \end{bmatrix} \tag{4.5}$$

Em comparação com a matriz Y' do item (4.2), não houve acréscimo de novos elementos após a eliminação efetuada na primeira linha.

$$A = \begin{bmatrix} 1 & & & X \\ & 1 & & X \\ & & 1 & X \\ & & & 1 \end{bmatrix} \tag{4.6}$$

Estes dois exemplos demonstram como a seqüência de operações ou rearranjo das linhas e colunas influenciam no número de operações e aparecimento de novos elementos. Em número de operações, para o primeiro exemplo foram necessários: 4 divisões, 6 multiplicações e 10 multiplicações-adições e, em termos de memória, ao menos 6 palavras de computador. O segundo exemplo, no entanto, necessitou de 4 divisões, 3 multiplicações e 3 multiplicações-adições e, ainda, 3 palavras de computador.

É importantíssimo salientar que as duplas de matrizes representadas em (4.1) e (4.4) e em (4.3) e (4.6) são, respectivamente, equivalentes entre si, e ainda, que os efeitos da variação da seqüência de linhas e colunas no processo de redução à forma triangular, são interpretados como a redução física da rede elétrica, no qual as barras são eliminadas segundo a ordem dada pela nova seqüência.

Assim, as reduções da rede exemplo da figura 4.1, efetuadas de duas maneiras diferentes, conforme mostraram os exemplos, podem ser representadas como na figura 4.2.

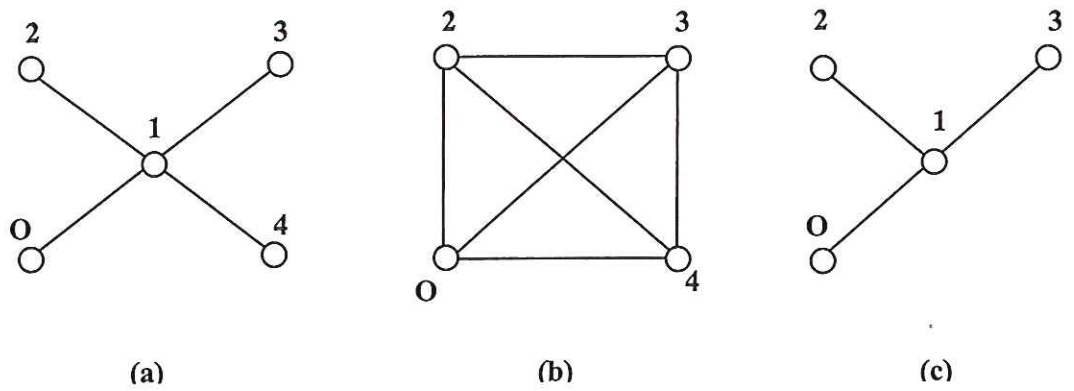


FIGURA 4.2 - Exemplo de redução de uma rede - eliminação de uma barra
(a) Rede Original (b) Rede Reduzida - 1° exemplo (c) Rede Reduzida - 2° exemplo

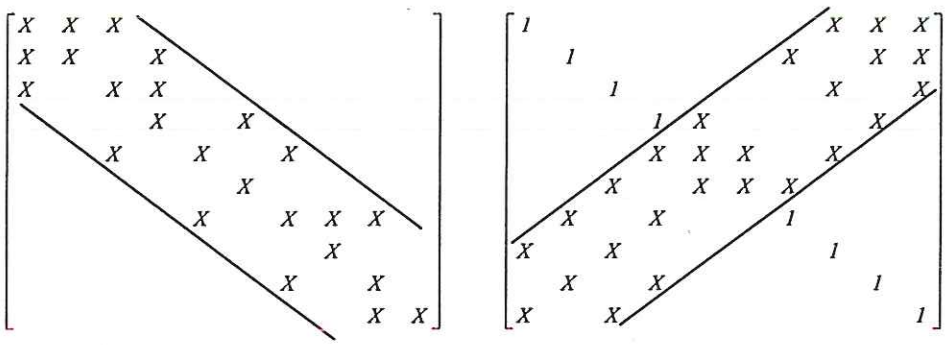
4.2 - Esquemas de Ordenação

Historicamente, os primeiros estudos relatados a respeito de ordenação da matriz esparsa em sistemas de potência estão relacionados a MARKOWITZ (1957); TINNEY & SATO (1963); GAMM, KRUMM & SHER (1965) e TINNEY & WALKER (1967). Apesar de todos terem sido apresentados como algoritmos "ótimos" de ordenação, nenhum atingiu o objetivo no sentido global e aqueles que mais se aproximaram foram denominados Pseudo-ótimos.

A mais antiga das técnicas conhecidas - MARKOWITZ (1957) - proposta para matrizes assimétricas, escolhe como pivô, a cada estágio, um elemento diagonal não zero que tenha o menor número de não zeros nesta linha. Este número de não zeros é o número na matriz reduzida antes do começo do estágio e será, em geral, diferente do número inicial da linha.

Para implementação do critério de MARKOWITZ, é necessário o conhecimento do modelo de esparsidade da submatriz reduzida a cada estágio da eliminação. Isto requer acesso às linhas e colunas, pois as posições das entradas não zero da coluna do pivô são necessárias para determinar quais linhas mudaram.

Estudos posteriores realizados por TINNEY & SATO (1963) e GAMM, KRUMM & SHER (1965) mostraram a ordenação através do Esquema de Matriz de Banda. O objetivo deste esquema era restringir os elementos da matriz a uma faixa estreita ao redor da diagonal principal ou, alternativamente, ao redor da diagonal secundária. Seu intuito era isolar os elementos não zero da matriz em regiões relativamente pequenas para poderem ser armazenadas de maneira mais compacta na memória do computador, ao invés da matriz inteira. A figura 4.3 ilustra o modelo geral de matrizes de banda.



(a) **(b)**

FIGURA 4.3 - Formas da Matriz de Banda
(a) Banda na Diagonal Principal (b) Banda na Diagonal Secundária

O maior prejuízo deste esquema é que a matriz de banda pode ainda ser esparsa, porém tende a tornar-se solidamente cheia durante o processo de eliminação. Esta óbvia

desvantagem pode ser reduzida pela proposta de banda ótima, entretanto não é computacionalmente atrativa.

Levando em consideração os estudos anteriores, TINNEY & WALKER (1967) propuseram três esquemas mais eficientes que o Esquema de Banda e, portanto, muito mais utilizados. Tais esquemas são aplicáveis para matrizes simétricas e, com certeza, encontram seus pivôs na diagonal, produzindo, assim, uma fatoração estável.

1 - ESQUEMA A ou 1 : conhecido como ordenação estática. Nele, a ordenação é feita segundo o menor número de elementos diferentes de zero nas linhas (ou colunas). Se duas ou mais linhas (colunas) apresentam a mesma quantidade de elementos, a escolha entre elas é arbitrária. A classificação é feita somente uma vez, sobre a matriz original, seguindo-se a eliminação em ordem crescente do número de elementos em cada linha ou coluna.

2 - ESQUEMA B ou 2: chamado de ordenação dinâmica. Aqui a ordenação é feita para a matriz original, segundo o menor número de elementos diferentes de zero nas linhas (ou colunas). Cabe aqui, também, a escolha arbitrária, como referenciado no esquema A. A cada fase do processo de eliminação, nova classificação é realizada e assim, sucessivamente, até o final do processo.

3 - ESQUEMA C ou 3: também conhecido como ordenação dinâmica. A cada passo do processo, é escolhida para ser eliminada a linha (ou coluna) que, se eliminada, introduza um número mínimo de elementos na matriz resultante. Se mais de uma linha (coluna) se qualificar, escolhe-se arbitrariamente. Este

critério envolve uma simulação experimental de todas as possíveis alternativas do processo de eliminação a cada passo.

Dos três esquemas, pode-se dizer que o ESQUEMA 1 é o mais simples e também o que fornece resultados mais modestos, apesar de ser bem melhor que uma renumeração aleatória (observa-se uma melhoria de 4:1 sobre nenhuma ordenação). O ESQUEMA 2 apresenta melhorias consideráveis em relação ao 1, sem representar um acréscimo significativo do custo computacional. Já o ESQUEMA 3, que é o melhor dos três, tem um custo computacional que, em geral, não justifica as eventuais melhorias de ordenação.

Estes três esquemas tornaram-se célebres e amplamente aceitos e empregados. Ficaram conhecidos como TINNEY 1, 2 e 3, sendo que o 2 (o mais popular) também é referenciado como algoritmo de ordenação de mínimo grau; pois, associando um grafo com a matriz esparsa simétrica, esta estratégia corresponde à escolha do nó, para a próxima eliminação, que tenha a menor quantidade de eixos conectados a ele. Pode-se dizer que a ordenação de mínimo grau é uma ordenação de MARKOWITZ aplicada ao problema simétrico, dadas suas semelhanças (DUFF, REISMAN & REID - 1986, p.130).

Outras metodologias de ordenação sugeridas na mesma época - OGBUOBIRI, TINNEY & WALKER (1970); BERRY (1971) e HSIEH & GHAUSI (1972) - mostraram-se eficientes somente para alguns problemas específicos. A primeira mostrou-se prática para sistemas superiores a 200 nós; a segunda relata que algumas estratégias locais de ordenação, ao invés de minimizarem o aparecimento de novos elementos, às vezes o incentiva; a terceira apresenta alguns conceitos probabilísticos para a pesquisa do pivô, visando à diminuição do tempo extra de computação gasto para prever o aparecimento de *fill-ins* quando das mudanças na matriz.

DUFF & REID (1974) subdividem as técnicas de ordenação/pivoteamento em duas categorias:

- Estratégias de ordenação *a priori*, em que as colunas (ou linhas) são primeiro ordenadas e, então, a cada estágio da eliminação, o pivô é escolhido de dentro da primeira coluna da submatriz reduzida (figura 4.4). Neste caso a ordem é escolhida diretamente do modelo de esparsidade original. Encaixa-se neste caso o esquema de TINNEY 1.

- Estratégias de ordenação local, quando o pivô é selecionado dentre todos os não zeros na submatriz reduzida - figura 4.4 - usando a estrutura de atualização neste estágio da eliminação. Envolve armazenamento e atualização do modelo de esparsidade da matriz remanescente. A técnica de MARKOWITZ é a mais popular das estratégias locais para casos assimétricos seguida de TINNEY 2 para problemas simétricos.

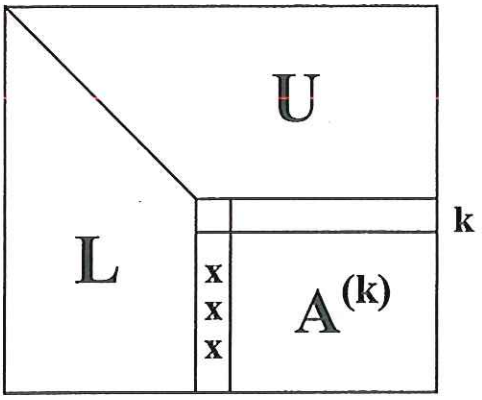


FIGURA 4.4 - Submatriz reduzida antes do k-ésimo passo da Fatoração LU

A tabela 4.1 apresenta resultados comparativos dos testes realizados pelos autores para algumas matrizes assimétricas. Foram utilizados os dois tipos de estratégias, sendo a local representada por MARKOWITZ.

TABELA 4.1 - Comparação de estratégias de ordenação sobre matrizes esparsas assimétricas

Ordem da matriz	Número de elementos	MARKOWITZ		ESTRATÉGIA *A PRIORI*	
		Fill-in	Multiplicação/ Divisão	Fill-in	Multiplicação/ Divisão
100	394	584	3.526	1.035	8.105
100	297	199	830	363	1.532
54	291	90	737	184	1.243
57	281	34	505	127	871
199	701	686	3.189	2.066	16.069

Portanto, percebe-se claramente as vantagens da ordenação local.

No mesmo ano - 1974 - NAKHLA, SINGHAL & VLACH sugeriram um algoritmo alternativo de ordenação, que se baseia no seguinte critério: a porção não ordenada da matriz atualizada é pesquisada para os elementos que criarão, se escolhidos como pivô, a mínima quantidade de novos elementos. Se mais do que um elemento se habilita, o escolhido é aquele que proporcionará o menor número de operações. Apesar de o método apresentar próximo de 15,2% de *fill-ins* a menos, em testes com matrizes 30 x 30, comparado aos algoritmos de BERRY e HSIEH & GHAUSI, o mesmo não se estabeleceu.

Também em 1974, HSIEH apresentou dois algoritmos de pivoteamento/ordenação: um determinístico e outro probabilístico. Ficou demonstrado que a aproximação

probabilistica oferecia vantagens explícitas em termos de tempo da ordenação e total de operações. Seu desenvolvimento foi baseado na disposição aleatória dos elementos em grandes matrizes. Para matrizes pequenas, entretanto, o algoritmo determinístico mostrou-se até dez vezes mais rápido que o outro.

Após a introdução dos métodos de vetores esparsos por TINNEY, BRANDWAJN & CHAN (1985), alguns métodos de ordenação específicos vieram completá-los. Um destes foi sugerido por GÓMEZ & FRANQUELO (1988a) e intitula-se **Ordenação por Nós**. O método é dividido em três algoritmos que objetivam a melhoria da esparsidade de U^{-1} sem sacrificar a esparsidade de U.

Considerando a fatoração simétrica da matriz, tem-se $U^t U$, então uma nova matriz $F=U^t + U$ é chamada de *filled graph* (G_F) e será usada oportunamente.

Como toda a fundamentação conceitual do método é baseada nos vetores esparsos, a Teoria dos Grafos é aqui plenamente explorada. Assim, se um elemento a_{ii} é escolhido como pivô e é eliminado, o grafo reduzido associado com a matriz reduzida é obtido do grafo original pela exclusão do vértice i e arcos incidentes e adição de arcos entre alguns pares de vértices que são vizinhos de i, mas não vizinhos entre si.

Dessa maneira, um exemplo de sistema de 14 nós (figura 4.5) foi submetido, inicialmente, à ordenação T-2 (o nó 1 é desconsiderado), e depois aos três algoritmos relativos ao método. Os resultados foram comparados entre si e com T-2.

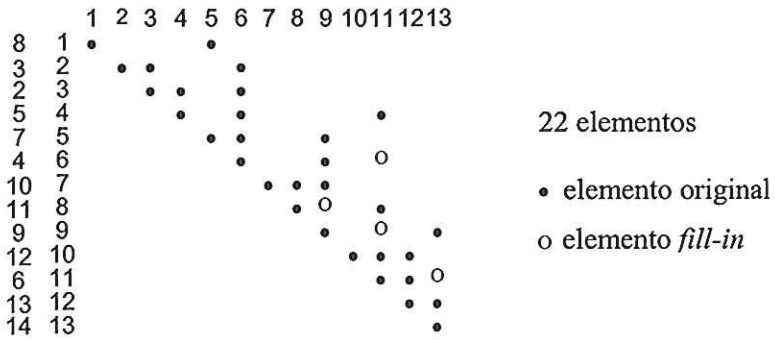


FIGURA 4.5 - Estrutura de esparsidade da matriz *U* (14 nós - ordenação T-2)

O *path graph* para esta rede, que compactamente descreve todos os *path's*, é mostrado na figura 4.6. Este grafo é uma árvore com senso de direção de baixo para cima para *FF* e o contrário para *FB*.

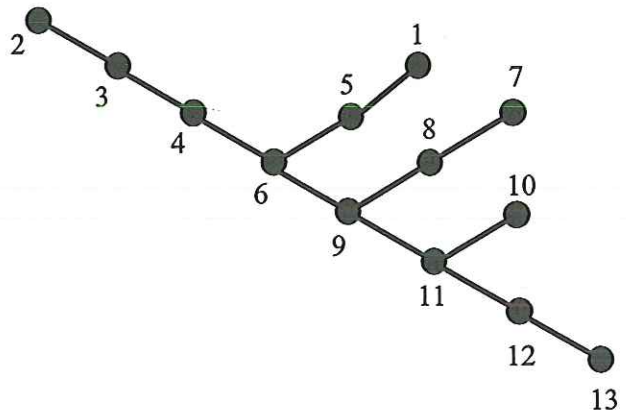
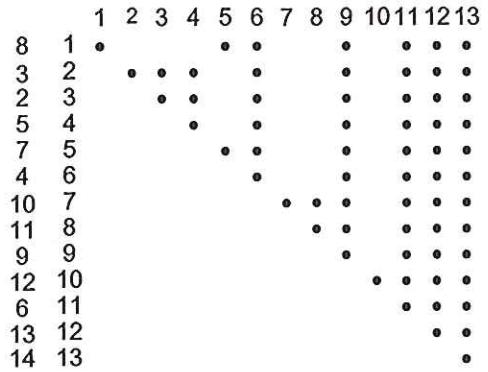


FIGURA 4.6 - *Path Graph*

Há um relacionamento entre este *path graph* e a estrutura de esparsidade de U^{-1}. Os nós pertencentes ao k-ésimo nó do *path* coincidem com as colunas não zero da k-ésima linha de U^{-1} (figura 4.7).



51 elementos

FIGURA 4.7 - Estrutura de esparsidade de U^{-1} (ordenação T-2)

Cada permutação da ordenação por nós, na qual todo nó é numerado depois de seus predecessores, não altera a forma da árvore e o número de elementos em U e U^{-1}, somente resulta na relocação destes elementos. Conseqüentemente, o esforço necessário para executar os processos de *forward* e *backward* permanecem inalterados. Entretanto, se uma variante do método que usa o vetor completo (que considera a direção de entrada em b) é usada, o esforço computacional será diferente. O algoritmo T-2 normalmente agrupa os elementos na parte inferior direita de U, porém uma distribuição mais regular destes elementos pode ser adquirida, se os nós são renumerados por exaustão de cada ramo da árvore antes de agregar os outros, levam em consideração a regra de precedência acima mencionada. Para o vetor esparso, o número médio de operações necessárias para executar esta hábil variação, é muito pequeno.

Como já observado, o objetivo do método dos vetores esparsos é fazer com que U e U^{-1} permaneçam tão esparsas quanto possível, assim o número de elementos em U^{-1} está diretamente relacionado à média entre os *path length*. Já o número de elementos em U, relaciona-se à média de multiplicações/adições dos processos de substituiçao *forward* ou *back* sobre cada linha.

Assim, os três algoritmos são descritos a seguir, precedidos do algoritmo T-2. Para tanto, destaca-se o significado dos vetores utilizados nos mesmos:

- $f(i)$ = posição do nó i na ordenação final. Inicialmente é zero. $f(i) = 0$ significa que o nó i ainda não foi considerado;
- $D(i)$ = grau do nó i no grafo reduzido, cujo valor inicial é o grau de cada nó no grafo original;
- $B(i)$ = número de nós j adjacentes a i no grafo, tal que $f(j) \neq 0$ e $f(j) < f(i)$. Inicialmente é zero;
- $M(i)$ = máximo $f(j)$, tal que j é adjacente a i no grafo e $f(j) < f(i)$. Inicialmente deve ser zero.

ALGORITMO DA ORDENAÇÃO DE MÍNIMO GRAU (T-2):

1. Faça K = 1
2. Escolha um nó i com f(i)=0 tal que D(i) é mínimo. As ligações são quebradas arbitrariamente, embora para a conveniência da programação, o primeiro ou o último nó na ordem natural, é o escolhido. Faça f(i) = k.
3. A cada j adjacente a i, tal que f(j)=0, faça D(j)=D(j)-1.
4. Para cada par de nós m e n adjacentes a i, mas não adjacentes entre si, tal que, f(m)=f(n)=0, crie um novo eixo unindo m e n e incremente D(m) e D(n) em uma unidade.
5. Se k=N, pare.
   Caso contrário, faça k=k+1 e vá para o passo 2.

Os três algoritmos propostos diferem do T-2 somente nos passos 2 e 3. Portanto, os passos 1, 4 e 5 não serão reescritos.

ALGORITMO 1 (A1):

1. Escolha um nó i com f(i)=0 tal que D(i) é mínimo. Se mais do que um nó atende este critério, selecione um com mínimo B(i). Faça f(i) = k.
2. Para cada j adjacente a i, tal que, f(j)=0, faça D(j)=D(j)-1 e B(j)=B(j)+1.

ALGORITMO 2 (A2):

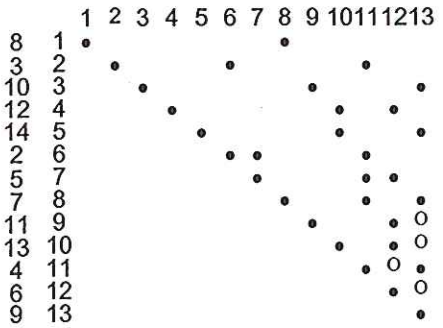
2. Escolha um nó i com f(i)=0 tal que D(i) é mínimo. Se mais do que um nó atende este critério, selecione um com mínimo M(i). Faça f(i) = k.
3. Para cada j adjacente a i, tal que, f(j)=0, faça D(j)=D(j)-1 e M(j)=k.

ALGORITMO 3 (A3):

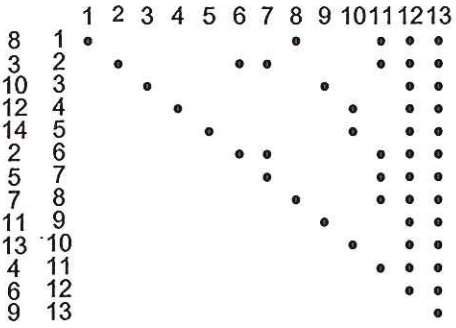
2. Idem a A1.
3. A cada j adjacente a i, tal que, f(j)=0, faça B(j)=B(j)+1.

A complexidade destes algoritmos é praticamente a mesma do T-2. Qualquer rotina do T-2 pode ser adaptada para implementar estes algoritmos.

O sistema teste de 14 nós da figura 4.5 foi submetido à ordenação pelos algoritmos A1, A2 e A3. Os resultados são descritos nas figuras 4.8, 4.9 e 4.10, respectivamente. Em comparação ao algoritmo T-2 (figura 4.7), nota-se que a esparsidade de U^{-1} é consideravelmente melhorada em todos os três casos, especialmente em A3, apesar de U apresentar-se um pouco menos esparsa neste caso.

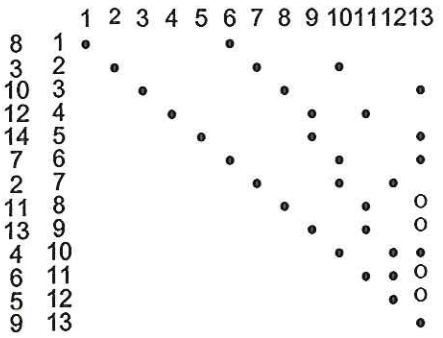


$U = 22$ elementos

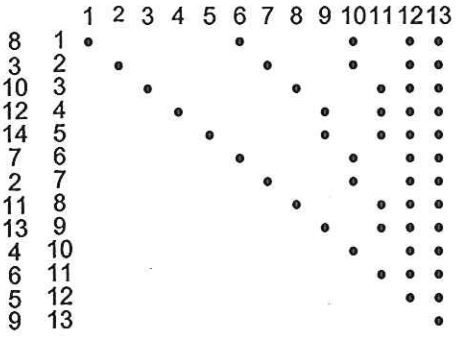


$U^{-1} = 35$ elementos

FIGURA 4.8 - Sistema de 14 nós ordenado pelo algoritmo A1

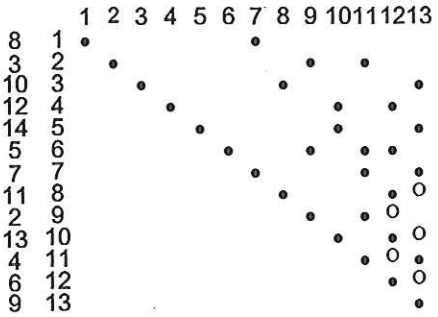


$U = 22$ elementos

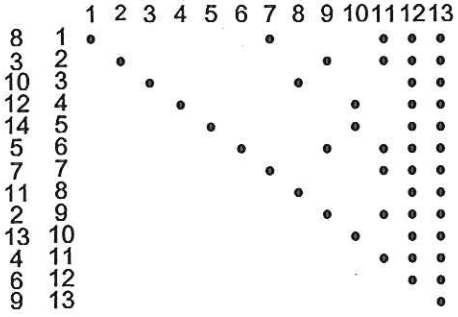


$U^{-1} = 37$ elementos

FIGURA 4.9 - Sistema de 14 nós ordenado pelo algoritmo A2



$U = 23$ elementos



$U^{-1} = 34$ elementos

FIGURA 4.10 - Sistema de 14 nós ordenado pelo algoritmo A3

Comparando os três resultados obtidos, observa-se que a principal debilidade do algoritmo T-2 é o grande número de ligações que aparecem em seu passo 2. A maneira destas ligações serem quebradas não tem uma influência forte sobre a estrutura de esparsidade de U (DUFF, ERISMAN & REID - 1986, Cap. VII), mas não é o caso quando U^{-1} ou, equivalentemente, a média do *path lenght* é considerada. Isto é, o procedimento da estratégia T-2 depende muitíssimo da ordem natural dos nós, o que não é uma característica muito favorável para a eficiência.

Reconhecendo este fato, os três algoritmos atuam de maneira diferente atingindo resultados melhores, sendo que A1 é claramente superior aos outros (em média 20% menos operações aritméticas), incluindo T-2, o qual é superado por A2 e, sob certas circunstâncias, por A3.

Dando prosseguimento ao estudo dos algoritmos de ordenação, os mesmos autores, em artigo publicado em novembro do mesmo ano, introduzem uma nova estratégia visando à melhoria do algoritmo TINNEY-2, denominada Grau mínimo - Comprimento mínimo - MD-ML (*Minimum degree - Minimum length*). Esta usa o conceito de *length* do nó no *path graph*, que é atualizado facilmente. Então, *length* de um nó i é o comprimento máximo de todos os nós que o precedem no *path graph*. Constatou-se, em comparações feitas com o método anterior A1, que a estratégia MD-ML é muito melhor.

Em um sistema exemplo de 10 nós, figura 4.11, consideremos que os nós 1, 8 e 9 permanecem para serem eliminados e os outros já o foram na ordem: 5, 10, 4, 7, 2, 6 e 3, como mostra a matriz U representada na figura 4.12.

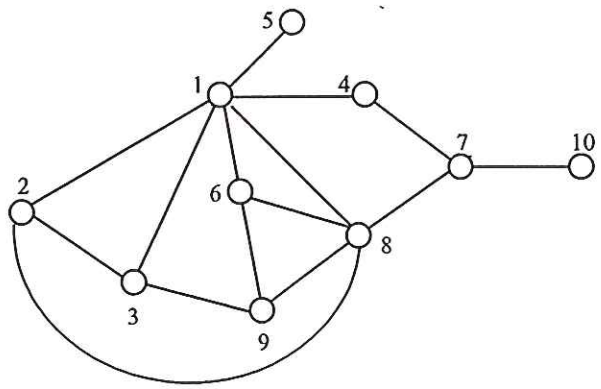


FIGURA 4.11 - Sistema de 10 nós

		1	2	3	4	5	6	7	8	9	10
5	1	X							X		
10	2		X		X						
4	3			X	X				X		
7	4				X				O	X	
2	5					X		X	X	X	
6	6						X		X	X	X
3	7							X	X	O	X
1	8								X	X	O
8	9									X	X
9	10										X

X - elementos fora da diagonal
O - elementos *fill-in*

FIGURA 4.12 - Estrutura de *U* segundo MD-ML

Assim, o *filled graph*, que representa a matriz da figura 4.12, é mostrado na figura 4.13.

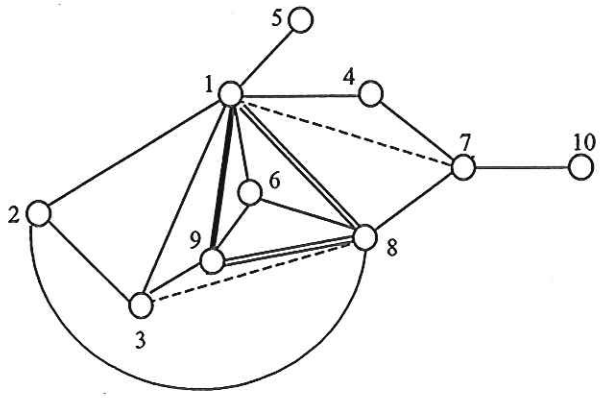


FIGURA 4.13 - *Filled graph* antes dos nós 1, 8 e 9 serem eliminados

As figuras 4.14(a), 4.14(b) e 4.14(c) mostram como o *path graph* cresceria se os nós 1, 8 e 9 fossem eliminados neste ponto, respectivamente.

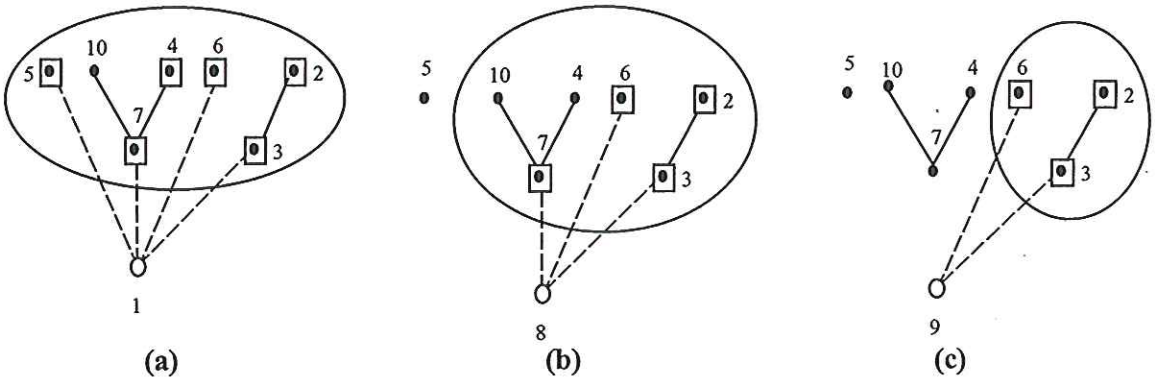


FIGURA 4.14 - Crescimento do *path graph* considerando três diferentes hipóteses

O método propõe, como critério de quebra de ligações, o uso de um conjunto de nós que precedem o nó sob consideração no *path graph*. Isto é equivalente a minimizar localmente o número de elementos não zero em U^{-1}.

Seguindo esta estratégia, o nó 9 é o primeiro a ser eliminado - figura 4.14(c), resultando no *path graph* mostrado na figura 4.15, onde os conjuntos de nós precedentes estão contornados. Observa-se que o próximo nó escolhido é o 8 e por último o 1.

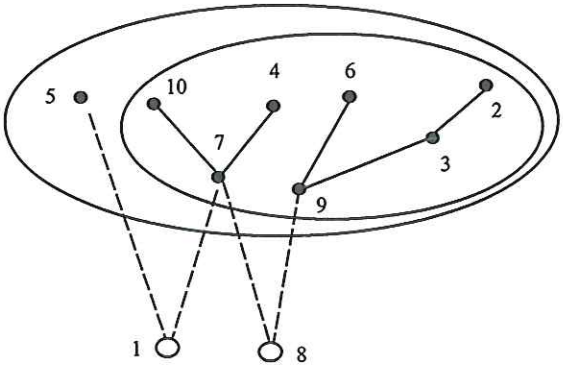


FIGURA 4.15 - *Path graph* antes dos nós 1 e 8 terem sido eliminados

Uma análise cuidadosa das figuras 4.14 e 4.15 aponta para a necessidade de um método de atualização dos conjuntos precedentes para cada nó eliminado. Este método foi denominado, pelos autores, como MD-MNP - mínimo grau-mínimo número de predecessores.

O algoritmo do MD-MNP é apresentado a seguir, sendo acompanhado das descrições dos vetores utilizados.

- *P(i)* - número de nós no *path graph* que precedem o nó *i*. Inicialmente vale 1.
- *F(i)* - variável booleana. É verdadeira se o nó *i* é o último nó adicionado a algum componente conectado no *path graph*, isto é, se o nó *i* é um nó de fronteira. Inicialmente vale F. Para os nós de fronteira, é necessário apenas conhecer o *P(k)*.

ALGORITMO MD-MNP:

1. Faça k = 1.
2. Escolha um nó *i* com *f(i)* = 0, tal que *D(i)* é mínimo. Quebre as ligações que aparecerem neste processo selecionando o nó com mínimo *P(i)*. Faça *f(i)* = k, *F(i)* = 'V'.
3. Se k = n, fim.
4. Para cada j adjacente a *i* no *filled graph*, faça:
   - Se *f*(j)=0 então faça *P*(j)=*P*(j)+*P*(i), *D*(j)=*D*(j)-1
   - Se *F*(j)='V', então faça *F*(j)='F' e para cada *m* adjacente a *j*, tal que, *f(m)*=0, faça *P*(m)=*P*(m)-*P*(j)
5. Para cada par de nós *m* e *n* , adjacentes a *i*, mas não a si próprios, tal que *f*(m)=*f*(n)=0, crie um novo eixo unindo *m* e *n*, e incremente *D*(m), *D*(n) em uma unidade.
6. Faça k=k+1 e vá para o passo 2.

A figura 4.16 mostra os resultados da aplicação do algoritmo ao sistema de 10 nós, que produziu uma ordenação diferente.

		1	2	3	4	5	6	7	8	9	10
10	1	X			X						
5	2		X								X
4	3			X	X						X
7	4				X					X	O
9	5					X		X	X	X	
2	6						X		X	X	X
6	7							X	O	X	X
3	8								X	O	X
8	9									X	X
1	10										X

X - elementos fora da diagonal
O - elementos *fill-in*

FIGURA 4.16 - Estrutura de U seguindo o algoritmo MD-MNP

Os testes comparativos realizados para quatro sistemas (118, 265, 448 e 661 nós) apresentaram sempre os melhores resultados relativos, entre outros, a número de não zeros em U e U^{-1}, quantidade de operações de adição/multiplicação no processo de fatoração completo e tempo de execução da ordenação, para o algoritmo MD-MNP.

As únicas desvantagens referem-se a maior dificuldade para calcular ou atualizar os parâmetros dos conjuntos de precedência e ao fato de tratar-se de uma estratégia de minimização local. Usar o critério MNP como principal, retorna um número excessivo de *fill-ins*, produzindo *path's* mais longos. Por este motivo, ele somente é usado como critério de quebra de ligações.

BETANCOURT (1988) reforça a pesquisa dos métodos de refatoração parcial, apresentando um método heurístico de ordenação baseado na simples manipulação de grafos desordenados. Seu objetivo é minimizar o comprimento dos *path's* de fatoração, enquanto preserva a esparsidade da matriz. Este método, aplicado à inversão do fator triangular L, reduz o número de elementos não zero significativamente.

Este estudo é limitado a matrizes simétricas e definidas positiva.

Uma importante observação é que durante o processo de eliminação, os elementos da diagonal remanescentes na matriz reduzida determinam o comprimento do *path* no *k-ésimo* passo da fatoração. O conceito de comprimento do vértice é similar ao grau, usado para preservação da esparsidade. Também assumem que a_{pp} é escolhido como pivô e que $a_{ip}.a_{pi} \neq 0$ na matriz reduzida. Então, o comprimento l_i, correspondente a diagonal a_{ii} no *k-ésimo* passo, é dado por:

$$l_i = max\{ l_p + 1, l_i \} \tag{4.1}$$

sendo l_p o comprimento do pivô nesta fase da eliminação. Estas observações sugerem que os comprimentos dos *path's* podem ser determinados usando os grafos desordenados da matriz. Primeiro os vértices no grafo desordenado começam com o comprimento valendo zero. Quando um vértice é eliminado, os comprimentos dos vértices adjacentes são atualizados usando a eq. (4.1). O processo continua até que todos os vértices tenham sido eliminados. O comprimento do vértice neste ponto da eliminação é igual ao comprimento deste *path* no *path graph* de fatoração.

Com esta técnica, um novo algoritmo de ordenação é proposto, utilizando-se dos comprimentos dos grafos desordenados, ao invés do número de eixos conectados aos vértices (mínimo grau). Recebe a denominação de ML (mínimo *length*).

ALGORITMO ML:

1. Ordene os vértices tal que a cada passo do processo o vértice a ser eliminado seja o que tem o menor comprimento.
2. Quando dois ou mais vértices satisfazem este critério, a escolha é arbitrária.

Minimizar somente os comprimentos não produz uma grande eficiência, também é necessário minimizar os *fill-ins*. Para tanto, duas estratégias gerais aparecem associadas ao algoritmo ML: a estratégia MD-ML (mínimo grau - mínimo *length*) e a ML-MD (mínimo *length* - mínimo grau). Seguem seus algoritmos.

ALGORITMO MD-ML:

1. Escolha o próximo vértice usando mínimo grau.
2. Quebre as ligações usando mínimo comprimento.
3. Se mais do que um vértice satisfaz este critério, quebre uma ligação arbitrariamente.

ALGORITMO ML-MD:

1. Escolha o próximo vértice usando mínimo comprimento.
2. Quebre as ligações usando mínimo grau.
3. Se mais do que um vértice satisfaz este critério, quebre uma ligação arbitrariamente.

As ordenações dadas pelos algoritmos MD e MD-ML foram testadas em um programa de refatoração parcial. Em termos somente da ordenação, o número de *fill-ins* é praticamente o mesmo, assim como o tempo de CPU também é muito próximo. Porém, considerando o tempo total de operações para a refatoração parcial, a nova técnica de ordenação mostrou uma redução de mais de 25%.

Em 1993, LEE & CHEN oferecem mais uma estratégia considerada eficiente para os métodos de vetores esparsos. Aliado ao algoritmo proposto, que minimiza localmente o número de elementos não zero na inversa da matriz triangular inferior, são também mostradas duas versões refinadas que podem, geralmente, dar um comprimento menor do *path* de fatoração simples e/ou composto.

O algoritmo recebeu a denominação de MIF - mínimo *fill-in* na inversa e pode ser usado para implementar na prática o ESQUEMA W (ENNS, TINNEY & ALVARADO - 1990). A implementação do ESQUEMA W requer a construção de uma lista estruturada de fatores inversos. Entretanto, o espaço de armazenamento necessário é proibitivamente grande para grandes sistemas e o tempo de ordenação muito longo. Então, a idéia do MIF é derivar uma fórmula para contar o número de elementos não zero nos fatores inversos, sem construí-los. Na implementação, somente é necessário um vetor inteiro do tamanho da ordem da matriz, sendo, então, o tempo de ordenação muito menor que o ESQUEMA W. A formulação proposta também pode ser aplicada aos métodos das inversas particionadas dos fatores de A (ALVARADO, YU & BETANCOURT - 1990).

Como visto anteriormente, a matriz de coeficientes A pode ser representada através de um grafo desordenado. Dois vértices deste grafo são adjacentes, se eles são conectados por um eixo. O conjunto adjacente de um vértice v é o conjunto de todos os vértices que são adjacentes a v e é denotado por $Adj(v)$. Denotando d_v como o grau de um vértice v e $|S|$ como o número de elementos no conjunto S, isto é, sua cardinalidade, então $d_v = |Adj(v)|$.

Para prosseguimento na compreensão do método, é necessário conceituar "*fill-ins* na inversa" (IF), básico para a técnica descrita. Para entender sua essência, consideremos o sistema de 10 nós da figura 4.11.

Sua estrutura de adjacência pode ser listada como segue:

$Adj(1) = \{2, 3, 4, 5, 6, 8\}$	$Adj(2) = \{1, 3, 8\}$	$Adj(3) = \{1, 2, 9\}$	$Adj(4) = \{1, 7\}$
$Adj(5) = \{1\}$	$Adj(6) = \{1, 8, 9\}$	$Adj(7) = \{4, 8, 10\}$	$Adj(8) = \{1, 2, 6, 7, 9\}$
$Adj(9) = \{3, 6, 8\}$	$Adj(10) = \{7\}$		

Se a ordem de eliminação é selecionada arbitrariamente pela seqüência 10, 5, 4, 7, 9, 2, 6, 3, 8, 1, então 3 *fill-ins* serão criados em L e 4 IF's na sua inversa L^{-1}, conforme mostrado na figura 4.17.

		10	5	4	7	9	2	6	3	8	1
10	1	D									
5	2		D								
4	3			D							
7	4	X		X	D						
9	5					D					
2	6						D				
6	7					X		D			
3	8					X	X	F	D		
8	9	I		I	X	X	X	X	F	D	
1	10	I	X	X	F	I	X	X	X	X	D

D - elementos da diagonal
X - elementos fora da diagonal
F - elementos *fill-in*
I - elementos *fill-in* da inversa

FIGURA 4.17 - Modelo de Esparsidade de L e L^{-1}

Como abordado anteriormente, a estratégia de ordenação do MIF é selecionar o vértice para ser eliminado, que produza a menor quantidade de IF's em L^{-1} a cada passo da fatoração, sem construir a tabela de fatores de L^{-1}. O conceito do MIF é similar à estratégia de ordenação TINNEY-3, na qual o pivô é selecionado dentre os que criam o menor número de *fill-ins* a cada passo da fatoração. Como o MIF não requer a construção da tabela de fatores, substancial quantidade de memória e tempo de CPU são economizados.

Para implementar o MIF, é necessário resolver dois problemas:

- Identificar as posições dos IF's em L^{-1};
- Derivar uma fórmula para calcular o número de IF's em cada vértice.

Para tanto são definidos dois conjuntos, usando os conceitos de predecessores e fronteiras, para auxiliar na identificação das posições, e um vetor para o cálculo do número de IF's:

- *Pre(i)* - conjunto das posições não zero na coluna correspondente à linha do vértice *i* em L^{-1};
- *Fro(i)* - conjunto dos nós de fronteira que são adjacentes e precedem o vértice *i* no *path*;
- *I(i)* - número de IF's produzidos em L^{-1} pela eliminação do vértice *i*.

Tem-se, então, que se o vértice *m* é selecionado para ser eliminado e $Adj(m) = \{m_1, m_2, ..., m_\mu\}$ e $Fro(m) - (Fro(m) \cap Fro(m_i)) = \{\varphi^i_1, \varphi^i_2, ..., \varphi^i_{\mu i}\}$, $i = 1, 2, ..., \mu$, então as posições dos IF's nas colunas serão localizadas por:

$$\bigcup_{j=1}^{\mu_i} Pre(\varphi^i_j) \quad \text{para } i = 1,2,...,\mu \tag{4.2}$$

e o número de IF's produzidos em L^{-1} será dado por:

$$I_v = \sum_{i=1}^{n} (d_v - d_{\varphi_i} + 1) P_{\varphi_i} \tag{4.3}$$

sendo: $d_v = |Adj(v)|$ - grau do vértice v

$d\varphi_i = |Ind(\varphi_i)|$ - $i = 1, 2, \ldots, n$

$P\varphi_i = |Pre(\varphi_i)|$ - número de predecessores do vértice φ_i

Portanto, conhecidas as formas de manusear o método, segue o algoritmo completo denominado mínimo *fill-in* - mínimo *fill-in* na inversa (MF-MIF).

ALGORITMO MF-MIF:

0. Para i = 1, 2, ... , n, inicialize
   - número de *fill-ins*: $f_i=[d_i(d_i-1)-\sum|Adj(i)\cap Adj(b_j)|]/2$, $b_j\in Adj(i)$, sendo f_i o número de *fill-ins* em L
   - $P_i = 1$
   - $I_i = 0$
1. Selecione um vértice *m* para eliminação, tal que *fill-ins* f_m em U seja mínimo e quebre as ligações usando mínimo I_m
2. Para todo vértice adjacente $m_i\in Adj(m)$,
   - exclua o vértice *m* de $Adj(m_i)$
   - inclua a fronteira *m* em $Fro(m_i)$
   - atualize o grau $dm_i= dm_i-1$
   - se $f_m=0$, vá para o passo 4
3. Para quaisquer dois vértices desconectados m_i, $m_j\in Adj(m)$,
   - encontre $Adj(m_i)\cap Adj(m_j)$ e calcule a cardinalidade *v*
   - para toda intersecção do vértice μ_k, $\mu_k\in Adj(m_i)\cap Adj(m_j)$
     - atualize *fill-in*: $f\mu_k=f\mu_k-1$, $k= 1, 2,\ldots, v$
   - atualize as informações de *fill-in* nos vértices m_i e m_j:
     - $f_{mi}=f_{mi}+(d_{mi}-v)$
     - $f_{mj}=f_{mj}+(d_{mj}-v)$
   - adicione novos eixos (m_i, m_j):
     - insira vértice m_j em $Adj(m_i)$
     - insira vértice m_i em $Adj(m_j)$
     - atualize o grau $dm_i= dm_i+1$, $dm_j= dm_j+1$
4. Para $m_i\in Adj(m)$, atualize *fill-in* $f_{mi}=f_{mi}-(d_{mi}-d_{m+1})$
5. Para toda fronteira $\varphi_k\in Fro(m)$, $P_m=P_m+P\varphi_k$, $k=1,..,|Fro(m)|$
   - atualize a fronteira $|Fro(m_i)|= Fro(m_i)-(Fro(m_i)\cap Fro(m))$, para $m_i\in Adj(m)$
   - Faça $I_{mi}=0$
   - Para toda fronteira $\varphi_k\in Fro(m_i)$ $k=1,..,|Fro(m_i)|$:
     - $I_{mi}=I_{mi}+(d_{mi}-d\varphi_k+1)P\varphi_k$
6. Vá para o passo 1 até a ordenação terminar.

Como mencionado no início, duas versões refinadas para diminuição do *path* de fatoração para grupos simples ou compostos de *singleton's*, também são propostas. Estas referem-se aos fatores mínimos (d_i+I_i) e (f_i+I_i) que, além de não degradarem a esparsidade, incrementam os esforços computacionais para as substituições *forward* e *backward*.

Os resultados dos testes mostraram que o tempo adicional na implementação do critério MIF é muito pequeno. Entretanto, esta pequena fração de tempo pode ser compensada pela melhoria introduzida aos métodos de vetores esparsos e refatoração parcial.

WANG & GOOI (1997) propuseram dois novos métodos de ordenação - MFI1 e MFI2, baseados na diagonalização de A através do uso de uma matriz de transformação C, que podem ser usados para reduzir o número de elementos nos fatores inversos da matriz esparsa. Ao invés da convencional decomposição LDU, seguida de uma série de multiplicações pela matriz W, a matriz C gera, em somente um simples estágio, uma nova seqüência de nós para a rede e todos os elementos desta.

Durante o processo de eliminação, as matrizes de permutação são usadas para minimizar os elementos *fill-in* em C. A nova seqüência de nós é então obtida das matrizes de permutação. Se esta nova seqüência de nós é aplicada a A no método da matriz W, a matriz W terá a menor quantidade de *fill-ins* se comparado com outros métodos de ordenação.

Qualquer matriz A pode ser diagonalizada via eliminação de *Gauss*, através da transformação:

$$C_R A\, C_C = D \tag{4.4}$$

sendo: C_R e C_C as matrizes de transformação das linhas e colunas, repectivamente, e D a matriz diagonal.

A diagonalização de A é uma série de operações de linhas e colunas sobre esta, sendo C_C e C_R os respectivos registros das operações. Estas operações podem ser

executadas tal que C_R transforma A em uma matriz triangular superior e C_C transforma esta última na matriz diagonal.

Se a solução do sistema linear padrão existe, e A é simétrica, tem-se que a eq. (4.4) é reduzida a (4.5) e, em seguida, a (4.6).

$$C^t A\, C = D \tag{4.5}$$

$$A^{-1} = C\, D^{-1}\, C^t \tag{4.6}$$

Para obter a matriz C, uma matriz unitária é acrescentada abaixo da matriz A. Se a ordem de A é n, a ordem da matriz aumentada será $2n \times n$. Pela aplicação da eliminação de *Gauss* à matriz aumentada, as linhas e colunas dos elementos fora da diagonal no triângulo superior vão sendo eliminadas. Depois que a eliminação está completa, o triângulo superior será a matriz D e o triângulo inferior a matriz C.

Assim, para minimizar o número de *fill-ins* em C, adotam-se dois critérios:

- o algoritmo de mínimo *fill-in* 1 (MFI-1), calcula para cada nó na matriz A_i, o produto do número de elementos não zero fora da diagonal na linha pelo número de elementos não zero fora da diagonal na coluna da matriz aumentada. Este produto é o máximo possível de *fill-ins* que podem ocorrer na matriz aumentada, se a eliminação da linha e coluna do nó for efetuada.
- o algoritmo de mínimo *fill-in* 2 (MFI-2), calcula o número de *fill-ins* atuais para cada nó da matriz A_i, se a eliminação for executada.

Em nenhum destes critérios, os elementos da linha/coluna do nó com o mínimo *fill-in* serão substituídos acima e à esquerda da i-ésima posição e todos os nós correspondentes afetados serão substituídos abaixo e à direita, antes da i-ésima eliminação ser executada. O deslocamento pode ser expresso através de uma matriz de permutação P_i.

Considerando a rede exemplo de 20 nós representada na figura 4.18, apresenta-se na figura 4.19, a topologia da matriz A associada.

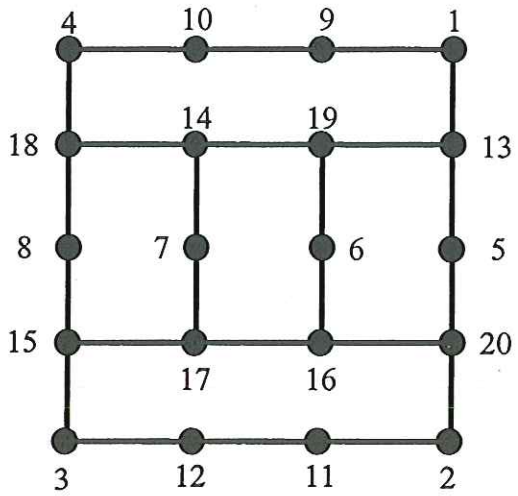


FIGURA 4.18 - Rede exemplo de 20 nós

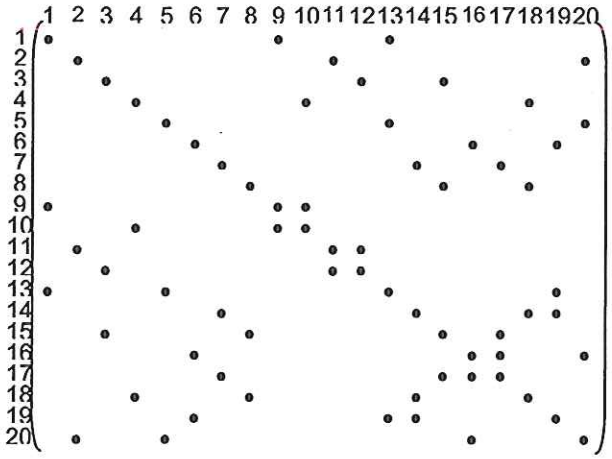


FIGURA 4.19 - Topologia da matriz A

Para ilustrar os algoritmos de mínimo *fill-in*, uma matriz 20 x 20 unitária é acrescentada abaixo da matriz A original. Os algoritmos MFI-1 e MFI-2 são aplicados para todos os 20 nós da matriz. O nó com o menor número possível/atual de *fill-in* será substituído, se necessário, pelo nó 1, antes da eliminação. Neste exemplo, nenhum deslocamento ocorre nos primeiros nove passos. Depois da nona eliminação, a matriz aumentada é como mostrada na figura 4.20, onde o "X" representa os elementos *fill-in* gerados por $C_1C_2...C_9$ via eliminação de *Gauss*. Os elementos não zero fora da diagonal em A_9 serão eliminados nos passos seguintes.

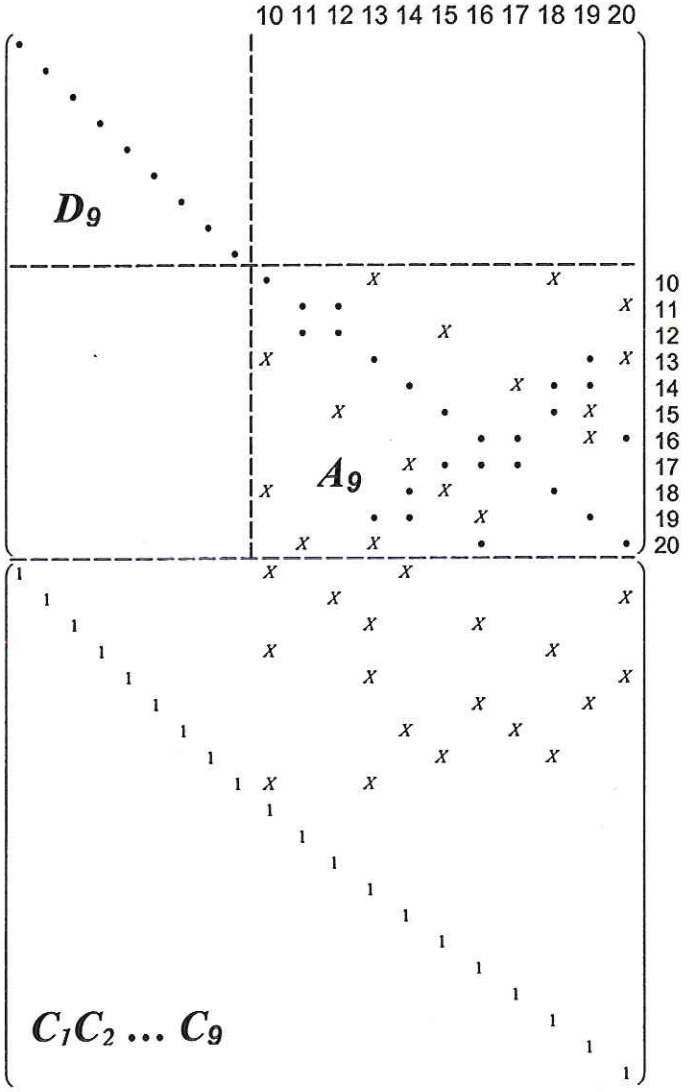


FIGURA 4.20 - Matriz Aumentada depois do nono passo da eliminação

A tabela 4.2 mostra os *fill-ins* calculados, usando MFI-1 ou MFI-2 no décimo passo.

TABELA 4.2 - Números possível e atual de *fill-in* no décimo passo

Número do Nó	Elementos não zero		*Fill-ins* (MFI-1/MFI-2)	
	Linha	Coluna	Possíveis	Atuais
10	2	4	8	6
11	2	3	6	5
12	2	3	6	5
13	3	6	18	15
14	3	4	12	11
15	3	5	15	13
16	3	4	12	11
17	3	4	12	11
18	3	5	15	14
19	3	4	12	11
20	3	5	15	13

Baseado nos resultados da tabela 4.2, é claro que os nós 11 e 12 produzem o mesmo número mínimo de *fill-ins*. Os elementos da linha e coluna do nó 11 em A_9 e as matrizes $C_1C_2...C_9$ serão substituídos com os elementos correspondentes do nó 10 antes do décimo passo da eliminação ocorrer. No final do décimo nono passo, a metade superior da matriz aumentada será a matriz diagonal D e a metade inferior a matriz C. A seqüência de nós baseada em MFI-1 ou MFI-2 será registrada nas matrizes de permutação $P_1P_2...P_{19}$. Esta seqüência deve ser aplicada à matriz A para o método da matriz W via decomposição LDU.

Para a implementação dos algoritmos MFI, uma eliminação simbólica é executada primeiro para determinar a seqüência de nós e a estrutura de C usando tecnologia de matrizes esparsas. A matriz de permutação é armazenada em um vetor de tamanho n. Este é inicializado com a seqüência de nós original, sendo esta atualizada a cada passo de acordo

com a permutação executada sobre A. O resultado final é um vetor que indica a nova seqüência de ordenação de A.

Foram realizados testes com quatro sistemas de potência reais: 118, 353, 707 e 1.084 nós, aplicando os algoritmos propostos em comparação com outros sete. Os resultados obtidos estão representados na tabela 4.3.

TABELA 4.3 - Número de *fill-ins* em *C/W* e número de multiplicações necessárias para obtê-los

	118 (179)		352 (423)		707 (969)		1.084 (1.441)	
TINNEY-2	1.055	2.812	4.442	8.152	17.705	45.233	28.164	64.197
TINNEY-3	1.097	2.887	5.520	9.962	20.479	52.980	33.966	78.889
A1	846	2.341	3.821	7.249	16.110	41.345	25.806	59.684
A2	833	2.342	4.105	7.599	17.588	45.336	24.409	56.180
A3	868	2.522	3.753	7.599	16.485	45.337	23.831	61.933
MD-ML	796	2.261	3.481	6.750	16.218	45.241	22.238	52.436
ML-MD	762	2.448	3.178	7.385	19.895	58.343	26.517	72.386
MFI-1	759	2.261	2.991	6.216	14.668	40.870	19.658	48.947
MFI-2	732	2.197	2.812	6.122	14.787	42.329	18.339	46.321

OBS: Os *fill-ins* são representados pelo número da esquerda e o da direita representa o número de multiplicações. O número entre parênteses indica a quantidade de elementos não zero fora da diagonal no triângulo superior de A

Assim, observa-se que os algoritmos apresentados produzem uma quantidade de *fill-ins* bem menor que os outros métodos testados, traduzindo-se em incremento favorável do tempo de processamento.

Dando prosseguimento às pesquisas sobre algoritmos de ordenação, GOOI & WANG (1998) propõem mais dois novos: MFI1-ML e MFI2-ML, também baseados na matriz de transformação C.

Estes dois novos métodos, na verdade, são aperfeiçoamentos dos dois anteriores - MFI-1 e MFI-2. Como verificado, o MFI-1 calcula o número possível de *fill-ins* para cada nó remanescente que não tenha sido diagonalizado, se este nó fosse o escolhido como próximo pivô. O MFI-2, por sua vez, calcula o número de *fill-ins* atual para cada nó remanescente, se este nó fosse o usado como próximo pivô. Em cada caso, o nó com o número mínimo possível/atual é então o escolhido para a próxima eliminação.

Se dois ou mais nós têm o mesmo número mínimo de *fill-ins*, o *path length* pode ser usado como um critério secundário para a seleção do nó, isto é, selecionar um nó de acordo com o mínimo comprimento dos nós que foram selecionados anteriormente baseado em MFI-1/MFI-2. Se há dois ou mais nós que satisfazem o critério secundário, um dos dois é qualificado arbitrariamente, através de uma variante do método MD-ML (*Minimum degree - Minimum length*), no qual o MD é substituído por MFI-1/MFI-2. Como resultado, dois novos algoritmos de ordenação foram introduzidos: MFI1-ML (mínimo *fill-in* 1 - mínimo *length*) e MFI2-ML (mínimo *fill-in* 2 - mínimo *length*).

Assim como todos os outros métodos de ordenação, a essência dos algoritmos propostos encontra-se no critério de seleção do pivô. Estes consideram os elementos *fill-in* na matriz C com respeito a A, melhor do que com relação a U. Esta é a forma mais direta de minimizar o número total de elementos nos fatores inversos de A. Então, baseado nesta aproximação, o cálculo de L e U não é necessário.

A implementação dos algoritmos, baseada no critério de minimizar o número de elementos não zero em C, segue os seguintes passos:

- Começar com o primeiro passo da diagonalização e para cada nó da matriz A, calcular o número de *fill-ins* na matriz operacional C usando um do métodos de ordenação propostos: MFI-1, MFI-2, MFI1-ML, MFI2-ML;
- Selecionar o nó que tem o número mínimo de *fill-in*. Se alguns nós têm o mesmo valor, selecionar qualquer um;
- Substituir o nó selecionado pela primeira posição através de permutação e eliminar os elementos fora da diagonal via processo de diagonalização. A eliminação associada com o primeiro nós selecionado como pivô é então completada;
- A seleção do nó, permutação e eliminação são repetidas para os nós remanescentes de A até o último nó. A cada passo, o nó selecionado será substituído na primeira posição da matriz A remanescente.

Os mesmos quatro sistemas do método anterior foram agora submetidos aos dois novos algoritmos e, comparando com mais outros cinco métodos de ordenação, foi construída a tabela 4.4, que apresenta o tempo de computação - em segundos - dos sete, utilizando-se do método dos vetores esparsos.

Dos quatro algoritmos propostos, MFI-1 é relativamente o mais fácil de implementar devido a sua simplicidade estrutural. O algoritmo MFI2-ML tem melhores resultados para grandes sistemas esparsos, mas sua implementação é muito complexa. Considerando, então, o desempenho e a complexidade dos algoritmos, MFI1-ML é a melhor opção para a implementação da solução dos métodos de vetores esparsos.

TABELA 4.4 - Tempo de computação das soluções via método dos vetores esparsos

	118	352	707	1.084
TINNEY-2	3.2	32.7	325.5	869.3
TINNEY-3	3.3	36.6	352.1	950.2
A1	3.0	30.7	302.7	828.7
A2	3.0	31.7	332.1	793.9
A3	3.1	31.7	326.8	871.0
MD-ML	2.9	29.5	330.1	764.4
ML-MD	3.0	29.2	412.9	958.4
MFI-1	2.9	27.9	294.5	753.7
MFI-2	2.9	27.6	301.2	722.2
MFI1-ML	2.9	27.9	294.6	730.3
MFI2-ML	2.9	27.6	301.2	724.5

Analisando a tabela 4.4, percebe-se claramente o melhor desempenho dos dois algoritmos propostos. Este melhor resultado deve-se ao uso da técnica da matriz C que produz um número mínimo de operações para substituição *forward* e *back*.

5 - TÉCNICAS DE ESPARSIDADE

Como visto anteriormente, uma técnica de esparsidade constitui-se em uma metodologia específica de solução de grandes sistemas esparsos, tendo como principal objetivo a maximização da eficiência numérica e computacional na obtenção destas soluções.

Assim, dentre as diversas técnicas abordadas no Histórico, algumas mereceram atenção especial e serão analisadas neste capítulo. Com a finalidade de acompanhar sua evolução, as mesmas são apresentadas em ordem cronológica.

5.1 - Técnicas para exploração da esparsidade da Matriz de Admitância de Rede - TINNEY & SATO (1963)

Tal técnica está totalmente embasada na redução à forma triangular ou triangularização da matriz dos coeficientes através da proposição de duas idéias básicas: a primeira - determinação de uma seqüência de operações - é realizada através da implementação de uma rotina, na qual é feita a contagem do número de elementos em cada linha e a posterior classificação por ordem crescente do número de elementos. Esta ordem, armazenada em uma tabela, representa a ordem na qual as linhas serão processadas, ou seja, as linhas com menor número de elementos serão processadas em primeiro lugar. Este esquema de ordenação é referenciado como ESQUEMA I .

No mesmo artigo, os autores fazem referência a um segundo método de ordenação - ESQUEMA II - consideravelmente menos eficiente e com vantagens para problemas tão específicos, que foi refutado com o tempo.

Determinada, então, a seqüência de operações, verifica-se a segunda idéia básica da técnica: determinação dos operadores e preservação dos mesmos.

Tomando a matriz *YBARRA* e os vetores I - injeções de corrente - e E - tensões nodais - como segue:

$$I = Y.E \tag{5.1}$$

observa-se que, freqüentemente, problemas de rede representados por este sistema linear requerem multi-soluções do mesmo, vindo daí a importância de se preservar os operadores.

Na verdade, estes operadores nada mais são do que os multiplicadores associados aos processos de eliminação de *Gauss* e fatoração LU. Cada multiplicador - atingido com os mesmos índices da linha e coluna referentes ao elemento a ser eliminado - é o único elemento diferente de uma matriz identidade, o que origina a chamada *Matriz Elementar*.

Dessa forma, as combinações lineares efetuadas com as linhas da matriz dos coeficientes Y com os elementos do vetor independente I, podem ser interpretadas como operações com as matrizes elementares do tipo:

$$T_{ij} = \begin{pmatrix} 1 & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & c_{ij} & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & 1 \end{pmatrix} \qquad T_{ii} = \begin{pmatrix} 1 & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & c_{ii} & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & 1 \end{pmatrix} \tag{5.2}$$

Em um estágio intermediário qualquer do processo de eliminação, para se zerar o elemento (i, j) da matriz dos coeficientes, basta premultiplicá-la pela matriz elementar T_{ij} com $c_{ij} = -a_{ij} / a_{jj}$, em que a_{ij} e a_{jj} são os valores dos elementos (i, j) e (j, j) da matriz. Para tornar unitário o elemento (i, i) da matriz dos coeficientes, basta premultiplicá-la pela matriz elementar T_{ii} com $c = 1/a_{ii}$. Ou seja, quando uma matriz elementar (T_{ij} ou T_{ii}) premultiplicar uma matriz de coeficientes, ela realizará nessa matriz as mesmas combinações lineares que devem ser efetuadas em uma matriz identidade de mesma ordem para se obter a matriz elementar em questão.

O método de eliminação de *Gauss*, descrito anteriormente, pode ser executado por uma seqüência de operações com matrizes elementares, em que cada premultiplicação por uma matriz elementar corresponde à eliminação de um elemento da matriz dos coeficientes ou então à normalização de um elemento da diagonal principal.

Assim, segundo o esquema de programaçã sugerido pelos autores, as operações especificadas por uma *Matriz Elementar* podem ser definidas por um elemento - o multiplicador - e um par de coordenadas, que são armazenados em tabelas distintas: uma para multiplicadores da diagonal principal e outra para os demais. Completando o esquema, o vetor E é transformado na solução do problema, através de uma seqüência de multiplicações de matrizes elementares pelo vetor independente I, como mostrado a seguir:

$$E = T_{1n}T_{1,n-1}T_{1,n-2} \dots T_{11}T_{22} \dots T_{n-1,n-1}T_{nn} \dots T_{n-2,1}T_{n-1,1}T_{n1}\, I \qquad (5.3)$$

Observa-se, então, que qualquer modificação ocorrida somente no vetor I não implica na triangularização da matriz Y novamente, bastando, para obter a solução, apenas efetuar as operações descritas na eq. (5.3).

5.2 - Fatoração Triangular Otimamente Ordenada - TINNEY & WALKER (1967)

Como já abordado na técnica descrita anteriormente, no que diz respeito ao armazenamento dos operadores, verificou-se que estes tratam dos multiplicadores dos processos de *Gauss* e *fatoração LU* . Então, nesta fase do método, mais aperfeiçoada, os autores fazem referência à chamada "Tabela de Fatores", descrita nas eq. (5.4). Esta simboliza o resultado final da triangularização da matriz dos coeficientes e armazenamento dos operadores relativos à mesma.

$$\begin{matrix} d_{11} & u_{12} & u_{13} & \cdot & \cdot & u_{1n} \\ l_{21} & d_{22} & u_{23} & \cdot & \cdot & u_{2n} \\ l_{31} & l_{32} & d_{33} & \cdot & \cdot & u_{3n} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ l_{n1} & l_{n2} & l_{n3} & \cdot & \cdot & d_{nn} \end{matrix} \qquad (5.4)$$

sendo:

$d_{ii} = 1/a_{ii}^{(i-1)}$

$u_{ij} = a_{ij}^{(i)} \quad , \; i < j$

$l_{ij} = a_{ij}^{(j-1)} \; , \; i > j$

É necessário observar que os parênteses nas eq. (5.4) são omitidos para enfatizar que não se trata de uma matriz, mas sim, de um esquema de armazenamento.

Estes operadores, definidos em (5.4), são organizados, posteriormente, em algumas matrizes especiais utilizadas para a determinação da solução direta dos sistemas lineares. Estas matrizes, não singulares, diferem da matriz unidade somente nas linhas ou nas colunas indicadas.

D_i : Linha i $= (0, 0, \ldots 0, d_{ii}, 0, \ldots 0, 0)$

L_i : Coluna $i = (0, 0, \ldots 0, 1, -l_{i+1,i}, -l_{i+2}, \ldots -l_{n-1,i}, -l_{n,i})^t$

L_i^* : Linha i $= (-l_{i,1}, -l_{i,2}, \ldots -l_{i,i-1}, 1, 0, \ldots 0, 0)$

U_i : Linha i $= (0, 0, \ldots 0, 1, -u_{i,i+1}, -u_{i,i+2}, \ldots -u_{i,n-1}, -u_{i,n})$

U_i^* : Coluna $i = (-u_{1,i}, -u_{2,i}, \ldots -u_{i-1,i}, 1, 0, \ldots 0, 0)^t$

As operações anteriores e as retro-substituições sobre o vetor $\boldsymbol{b}$ que o transformam em $\boldsymbol{x}$ podem ser expressas como pre-multiplicações pelas matrizes D_i, L_i, ou L_i^*, e U_i ou U_i^*.

Deste modo, a solução de $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$ eq. (2.3) pode ser expressa como indicado abaixo:

$$U_1 U_2 \ldots U_{n-2} U_{n-1} D_n L_{n-1} D_{n-1} L_{n-2} \ldots L_2 D_2 L_1 D_1 b = A^{-1} b = x \tag{5.5}$$

$$U_1 U_2 \ldots U_{n-2} U_{n-1} D_n L_n^* D_{n-1} L^*_{n-1} \ldots L^*_3 D_2 L^*_2 D_1 b = A^{-1} b = x \tag{5.6}$$

$$U^*_2 U^*_3 \ldots U^*_{n-1} D_n L_{n-1} D_{n-1} L_{n-2} \ldots L_2 D_2 L_1 D_1 b = A^{-1} b = x \tag{5.7}$$

$$U^*_2 U^*_3 \ldots U^*_{n-1} U^*_n D_n L_n^* D_{n-1} L^*_{n-1} \ldots L^*_3 D_2 L^*_2 D_1 b = A^{-1} b = x \tag{5.8}$$

Cada uma das quatro equações acima, descreve uma seqüência de operações sobre o vetor $\boldsymbol{b}$ que é equivalente a premultiplicá-lo pela inversa $\boldsymbol{A}^{-1}$. Dependendo das técnicas de programação empregadas, uma das quatro equações será a mais conveniente. Observa-se que, se $\boldsymbol{A}$ é simétrica, somente os termos d_{ii} e u_{ij} são necessários.

Em relação à ordenação, avanço expressivo do método, três critérios são introduzidos: Esquema A ou 1, B ou 2 e C ou 3. Dos três critérios, pode-se dizer que o ESQUEMA 1 é o mais simples e também o que fornece resultados mais pobres, apesar de ser bem melhor que uma numeração aleatória. O ESQUEMA 2 apresenta melhorias consideráveis em relação ao 1, sem representar um acréscimo significativo do custo computacional. Já o ESQUEMA 3, que é o melhor dos três, tem um custo computacional que, em geral, não justifica as eventuais melhorias de ordenação.

Assim, após o sistema ter sido submetido à ordenação e decomposição triangular, chega a fase das soluções diretas. A ordenação só é aplicável a matrizes simétricas, enquanto

EESC-USP Biblioteca CAMPUS DE S. CARLOS

que a decomposição triangular e as soluções diretas são aplicáveis a qualquer sistema não singular, desde que se tomem os cuidados necessários em relação à preservação da precisão numérica. Embora cada uma dessas componentes possa ser aplicada isoladamente, o grande potencial do método só é verificado pela aplicação simultânea, juntamente com a utilização de técnicas de programação adequadas.

5.3 - Método da Bi-Fatoração - ZOLLENKOPF (1971)

Tal método é baseado na combinação das principais características dos dois outros métodos: a *Decomposição Triangular* e o *Produto das Inversas* , e relaciona-se à equação:

$$L_n L_{n-1} \dots L_2 L_1 A R_1 R_2 \dots R_{n-1} R_n = I \tag{5.9}$$

sendo:

L = matriz de fatores triangular inferior (*left-hand*)

R = matriz de fatores triangular superior (*right-hand*)

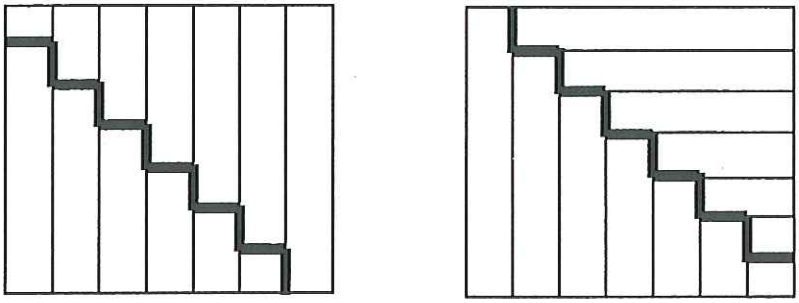
I = matriz identidade

A equação (5.9) pode ser modificada por simples transformações em:

$$A^{-1} = R_1 R_2 \dots R_{n-1} R_n L_n L_{n-1} \dots L_2 L_1 \tag{5.10}$$

Esta equação mostra que a inversa de A pode ser expressa por um múltiplo produto de $2n$ matrizes de fatores. É por este motivo que o termo "bi-fatoração" é utilizado.

Comparando os dois métodos - figura 5.1, observa-se que a principal diferença localiza-se na forma de acessar os elementos para o processo de redução. O método da Decomposição Triangular processa as linhas de duas matrizes triangulares, sendo que as linhas do triângulo superior não têm a mesma estrutura das linhas do triângulo inferior. No método da bi-fatoração, os fatores da matriz são formados pelas linhas do triângulo superior e colunas do triângulo inferior, as quais são idênticas para elementos não zero.



Decomposição Triangular **Bi-Fatoração**

FIGURA 5.1 - Esquema de comparação entre os métodos da Decomposição Triangular e Bi-Fatoração

As matrizes de fatores, apresentadas na eq. (5.9), são muito esparsas e diferem da matriz identidade apenas em alguns elementos, conforme verificado na figura 5.2.

- a triangular inferior $L^{(j)}$ difere da identidade somente na coluna j
- a triangular superior $R^{(j)}$, por sua vez, somente na linha j

$$L^{(j)} = \begin{pmatrix} 1 & & & 0 & & & \\ & \ldots & & \cdot & & & \\ & & 1 & 0 & & & \\ & & & l_{j,j}^{(j)} & & & \\ & & & l_{j+1,j}^{(j)} & 1 & & \\ & & & l_{j+2,j}^{(j)} & & 1 & \\ & & & \ldots & & & \ldots \\ & & & l_{n,j}^{(j)} & & & 1 \end{pmatrix} \qquad R^{(j)} = \begin{pmatrix} 1 & & & & & & & \\ & \ldots & & & & & & \\ & & \ldots & & & & & \\ 0 & \ldots & \ldots & 1 & r_{j,j+1}^{(j)} & r_{j,j+2}^{(j)} & \ldots & r_{j,n}^{(j)} \\ & & & & 1 & & & \\ & & & & & 1 & & \\ & & & & & & \ldots & \\ & & & & & & & 1 \end{pmatrix}$$

Coluna j Linha j

sendo:

$$\begin{cases} l_{jj}^{(j)} = 1 / a_{jj}^{(j-1)} \\ l_{ij}^{(j)} = - a_{ij}^{(j-1)} / a_{jj}^{(j-1)} \\ \text{para } i = (j + 1) \text{ até } n \end{cases}$$

sendo:

$$\begin{cases} r_{j,k}^{(j)} = - a_{jk}^{(j-1)} / a_{jj}^{(j-1)} \\ \text{para } k = (j + 1) \text{ até } n \end{cases}$$

FIGURA 5.2 - Fatores Triangulares *L* e *R* da matriz

O método é bastante rigoroso quanto aos seus pré-requisitos, pois o esquema de programação prosposto pelo autor, o grande diferencial do método, leva em consideração, além da esparsidade da matriz, que a mesma seja definida positiva, não tenha zeros na diagonal e seja simétrica. Por exemplo, no caso da simetria, se a matriz é simétrica de estrutura e valores, a j-ésima linha de ***R*** é idêntica a j-ésima coluna de ***L***, exceto no termo da diagonal. Entretanto, é desnecessário executar qualquer operação à direita da diagonal, levando, dessa forma, à execução de somente metade das operações de redução.

Ainda com relação à simetria, o esquema de armazenamento é outro fator de relevância, pois no caso simétrico os elementos são todos armazenados em uma única tabela (inclusive os da diagonal). No caso assimétrico, são necessárias três tabelas: uma para os

termos da diagonal, outra para os elementos das colunas e outra para os elementos das linhas.

Outro fator importantíssimo, do ponto de vista computacional, diz respeito à ordenação da matriz antes de iniciar a execução do programa, visando à minimização no aparecimento de novos elementos (*fill-in*). A estratégia aqui proposta é bastante semelhante ao Esquema 2 ou B introduzido por TINNEY & WALKER (1967). O princípio desta estratégia é selecionar a cada passo do processo de redução, para a coluna-pivô, aquela que contém o menor número de elementos.

O processo de programação, portanto, é tão importante quanto o método em si. Para efeito didático, o processo todo foi dividido em três partes:

- Simulação do processo de redução e ordenação
- Redução
- Solução direta

A etapa da simulação, por sua vez, é subdividida em duas etapas:

- *Pesquisa do Pivô:* na primeira etapa, entre todas as colunas, aquela com o menor número de elementos diferentes de zero é selecionada como coluna do pivô. Se mais do que uma coluna aceita este critério, a primeira delas é selecionada.

- *Modificação de índices e endereçamento:* através de comparação dos termos das colunas com a coluna-pivô, as informações sobre índices e endereçamentos são alteradas como segue: se a coluna processada contém o índice do pivô, o termo é cancelado; caso contrário, um novo termo aparece (*fill-in*).

A subrotina de redução opera sobre a imagem resultante do processo de simulação e ordenação. A redução dos coeficientes da matriz é guiada pela seqüência de pivoteamento contida em uma tabela. A cada estágio do processo de redução, somente alguns termos residuais da matriz reduzida, com índices correspondentes aos índices das linhas da coluna-pivô, têm que ser recalculados.

Finalizando, a terceira etapa do processo é que apresenta a solução propriamente dita. Através de sucessivas operações de multiplicação dos fatores da matriz pelos elementos do vetor b, que é originalmente armazenado, este é transformado no vetor solução.

5.4 - Método da Eliminação Esparsa e Simétrica de Gauss - EISENSTAT, SCHULTZ & SHERMAN (1981)

A técnica da *Eliminação Esparsa e Simétrica de Gauss* pode ser resumida em dois passos:

- fatoração da matriz A na forma:

$$A = U^t . D . U \tag{5.11}$$

- obtenção de x através da resolução dos sistemas:

$$U^t y = b \quad , \quad Dz = y \quad , \quad Ux = z \tag{5.12}$$

A mesma é baseada em algoritmos especiais que passam por três fases distintas, sendo estas compostas por rotinas claramente interdependentes intituladas *SYMFAC, NUMFAC* e *SOLVE*, descritas a seguir:

- a rotina *SYMFAC*, denominada fatoração simbólica, calcula a estrutura de zeros de $\boldsymbol{U}$;
- a *NUMFAC*, conhecida como fatoração numérica, usa a informação gerada pela *SYMFAC* para calcular os valores numéricos de $\boldsymbol{U}$ e $\boldsymbol{D}$;
- a rotina *SOLVE*, utiliza os resultados produzidas nos passos anteriores para determinar a solução através de *Forward e Back Substitution*.

Das três rotinas, a *NUMFAC* é a que requer, substancialmente, mais esforço computacional, uma vez que é nela que ocorre a redução da matriz.

A principal vantagem da separação dos passos é a flexibilidade. Por exemplo, se muitos sistemas têm matrizes $\boldsymbol{A}$ iguais e diferentes vetores $\boldsymbol{b}$, somente um passo *SYMFAC* e *NUMFAC* são necessários, e vários passos *SOLVE* para os diferentes vetores.

Associada à eficiência dos algoritmos, a técnica utiliza uma eficiente estrutura de dados para armazenamento das informações, consistindo em listas e vetores.

Visando à diminuição dos *loops* de repetição e testes (para verificar elementos diferentes de zero), os algoritmos baseiam-se no manuseio de conjuntos que são construídos com informações das matrizes $\boldsymbol{A}$ e $\boldsymbol{U}$, e que constituem o grande diferencial desta técnica.

- o conjunto $R^A{}_k$ das colunas $j > k$ para o qual $a_{kj} \neq 0$;

- o conjunto $R^U{}_k$ das colunas $j > k$ para o qual $u_{kj} \neq 0$;

- o conjunto C_k das linhas $i < k$ para o qual $u_{ik} \neq 0$;

- para cada $I < k$, o conjunto $R^U{}_{I,k}$ das colunas $j > k$ para o qual $u_{ij} \neq 0$.

Para efeito de implementação, os conjuntos $R^A{}_k$ e $R^U{}_k$ devem ser armazenados, pois são eles que descrevem a estrutura de zeros de A e U, respectivamente. Os outros dois conjuntos podem ser considerados como auxiliares e são determinados durante o progresso do programa.

Com respeito às estruturas de armazenamento, estas são orientadas por linha, pois as informações sobre a matriz são organizadas linha por linha. Elas assumem a forma vetorial ilustrada na figura 5.3, conhecida como esquema de armazenamento sem compressão (GUSTAVSON, 1972).

A título de ilustração, consideremos o exemplo de uma matriz A (5.13), e o seu esquema de armazenamento, ilustrado na figura 5.3.

$$A = \begin{bmatrix} X & X & X & 0 & 0 \\ X & X & 0 & X & 0 \\ X & 0 & X & X & X \\ 0 & X & X & X & 0 \\ 0 & 0 & X & 0 & X \end{bmatrix} \tag{5.13}$$

Observa-se que, devido à simetria da matriz A, somente os dados relativos ao triângulo superior e diagonal são armazenados.

A	a_{11}	a_{12}	a_{13}	a_{22}	a_{24}	a_{33}	a_{34}	a_{35}	a_{44}	a_{55}
k	1	2	3	4	5	6	7	8	9	10

JA	1	2	3	2	4	3	4	5	4	5

IA	1	4	6	9	10	11

FIGURA 5.3 - Esquema de armazenamento sem compressão da matriz *A*

O vetor *A* contém as entradas diferentes de zero do triângulo superior; o vetor *JA* contém os índices de colunas que correspondem às entradas diferentes de zero do vetor *A*; por fim, o vetor *IA* contém *n+1* apontadores que indicam as linhas de não-zeros em *A* e índices em *JA*, isto é, *A(IA(I))* é a primeira entrada armazenada na i-ésima linha e *A(IA(I+1) – 1)* é a última. Portanto, o tamanho da i-ésima linha é dado por *IA(I+1) – IA(I)*.

A matriz *U*, originada a partir da *A*, também será armazenada seguindo um esquema de armazenamento semelhante ao da matriz *A*, que será definido posteriormente, uma vez que a matriz *U* contém os elementos adicionais (*fill-ins*), que surgem durante o processo de eliminação. Cabe aqui uma observação relativa a este item: como visto, os conjuntos R^A_k e R^U_k são a base para o desenvolvimento de toda a técnica, e como mostra a definição do conjunto R^U_k, os elementos pertencentes a este conjunto são determinados através da operação *A* U {*fill-in*}, portanto é muito importante, para a eficiência da técnica, que o aparecimento de novos elementos e o número de operações sejam minimizados, e que o conjunto em questão seja o mais enxuto possível. É por este motivo que um passo de pré-processamento faz-se necessário: o passo da ordenação da matriz *A*.

Os autores não fazem referência a um critério de ordenação específico; portanto se a matriz A (5.13) é submetida ao Esquema de ordenação TINNEY 2, a matriz resultante e seus *fill-ins* são representadas em 5.14.

$$A = \begin{bmatrix} X & X & X & \otimes & \\ X & X & & X & \\ X & & X & X & X \\ \otimes & X & X & X & \\ & & X & & X \end{bmatrix} \tag{5.14}$$

Portanto, após este passo de ordenação, os vetores referentes à matriz U são construídos à semelhança dos relacionados à matriz A, conforme a figura 5.4 (a).

Como observado e pela própria definição da matriz U, os elementos da diagonal principal são todos iguais a 1 e ocorrem na primeira posição de cada linha, então para economia de espaço de armazenamento, estes não precisam ser guardados. Assim, a estrutura obtém a forma da figura 5.4 (b).

(a)

U	1	a_{12}	a_{13}	a_{14}	1	a_{24}	1	a_{34}	a_{35}	1	1
K	1	2	3	4	5	6	7	8	9	10	11

JU	1	2	3	4	2	4	3	4	5	4	5

IU	1	5	7	10	11	12

(b)

U	a_{12}	a_{13}	a_{14}	a_{24}	a_{34}	a_{35}
K	1	2	3	4	5	6

JU	2	3	4	4	4	5

IU	1	4	5	6	6	6

FIGURA 5.4 - Esquema de armazenamento sem compressão da matriz *U*

Para tornar o esquema de armazenamento mais compacto e eficiente, ainda, utiliza-se o esquema de armazenamento comprimido, como mostrado na figura 5.5. Na verdade, a compressão localiza-se sobre o vetor *JU* , uma vez que os vetores *U* e *IU* permanecem inalterados. Surge, então, um quarto vetor extra de ponteiros - *IJU* , necessário para localizar o começo dos índices de cada linha. Portanto, o vetor *U* contém as entradas diferentes de zero do triângulo superior, armazenadas linha por linha; as $n + 1$ entradas de *IU* delimitam as linhas de *U* , como antes; *JU* é o vetor comprimido de índices de colunas e as n entradas de *IJU* apontam para o primeiro índice de coluna em *JU* para cada linha.

U	a_{12}	a_{13}	a_{14}	a_{24}	a_{34}	a_{35}
K	1	2	3	4	5	6

JU	2	3	4	5

IU	1	4	5	6	6	6

IJU	1	2	3	4	4

FIGURA 5.5 - Esquema de armazenamento comprimido da matriz *U*

Observamos, então, que os elementos diferentes de zero da i-ésima linha são armazenados em $U(IU(i))$ até $U(IU(i+1)-1)$ e os índices de colunas correspondentes são armazenados em $JU(IJU(i))$ até $JU(IJU(i)+IU(i+1)-IU(i)-1)$.

Como já mencionado anteriormente, os algoritmos são divididos em três passos: *SYMFAC, NUMFAC* e *SOLVE*, sendo os mesmos descritos a seguir:

ALGORITMO 1: ***SYMFAC***

```
For K = 1 to N
      [R^U_K = 0];
For K = 1 to N-1
      [For j ∈ R^A_K
            [R^U_K = R^U_K ∪ {j}];
      For i ∈ C_K
            [For j ∈ R^U_I , k
                  [R^U_K = R^U_K ∪ {j}]]];
```

Neste passo, *SYMFAC,* ocorre a determinação do conjunto R^U_K a partir do conjunto R^A_K. As informações produzidas neste passo servem de entrada de dados para os passos subseqüentes. Os vetores ***A, JA, IA*** , portanto, são a entrada para a definição dos vetores ***U, JU, IU,*** ao final.

O algoritmo 2 descreve a rotina *NUMFAC*, ou seja, a eliminação propriamente dita. Observa-se que o algoritmo é uma versão orientada para linha e que as entradas relativas a diagonal são armazenadas em um vetor D de tamanho n, no qual $D(k) = d_{kk}$, e ainda, utiliza-se uma matriz auxiliar M que conterá, durante a execução do programa, entradas de U e de DU.

Os conjuntos C_k e $R^U_{I,k}$ são construídos durante o k-ésimo passo do algoritmo. Para calcular os conjuntos C_k, empregam-se conjuntos P_k que são construídos de uma forma, que durante o k-ésimo passo:

(i) $P_K = C_K$

(ii) para $j > k$, P_j contém os índices de linhas $i < k$, tal que

$$i \in C_j \text{ mas } i \notin C_m, k \leq m < j.$$

ALGORITMO 2: ***NUMFAC***

```
For K = 1 to N
      [P_K = 0];
For K = 1 to N
      [D(k) = a_kk;
      For j ∈ R^U_K
            [D(j) = 0];
      For j ∈ R^A_K
            [D(j) = a_kj];
      For i ∈ P_K
            [t=m_ik;
            m_ik = m_ik/D(i);
            D(k) = D(k)-t.m_ik;
            For j ∈ R^U_I,k
                  D(j)=D(j)-m_ik.m_ij];
            R^U_I,k = R^U_I,k - {k + 1};
            If R^U_I,k+1 ≠ 0 then
                  [j = min(R^U_I,k+1);
                  P_j = P_j ∪ {j}]];
      If R^U_K ≠ 0 then
            [j = min(R^U_k);
            P_j = P_j ∪ {k}]];
```

Finalmente, o algoritmo 3 descreve a rotina *SOLVE* que determina a solução do sistema, através de *Forward* ou *Back Substitution*.

ALGORITMO 3: ***SOLVE***

```
For K = 1 to N
      [yk = bk];
For K = 1 to N-1
      [For j ∈ R^U_K
            [yI = yI - ukj.yk];
For k = 1 to N
      [zk = yk/D(k)];
For k = N to 1
      [xk = zk;
      For j ∈ R^U_k
            [xk = xk - ukj.xj]];
```

O algoritmo *SOLVE* encaixa-se perfeitamente com a rotina *NUMFAC*, pois usa somente as entradas diferentes de zero das matrizes ***A, D*** e ***U***.

Em 1982, EISENSTAT, GURSKY, SCHULTZ & SHERMAN apresentaram detalhes do pacote *YSMPI - o código simétrico*, que traz a implementação da técnica descrita anteriormente.

O pacote é composto por diversas rotinas, sendo apresentado esquematicamente, na figura 5.6.

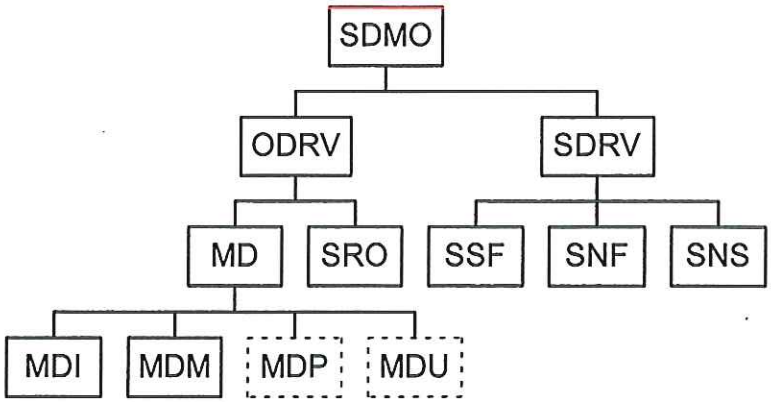


FIGURA 5.6 - Visão esquemática do pacote *YSMP I*

A rotina *SDMO* é a rotina principal, reponsável por chamar todas as outras. A rotina *ODRV* é a que faz a reordenação das variáveis e equações usando o algoritmo de mínimo grau e a *SDRV* encarrega-se da solução (subordinadas a ela encontram-se as rotinas que executam a fatoração simbólica, numérica e a solução *back*).

5.5 - Métodos de Vetores Esparsos - TINNEY, BRANDWAJN & CHAN (1985)

Os métodos de vetores esparsos significaram uma grande contribuição na pesquisa em técnicas de esparsidade, pois permitiram a solução para subconjuntos do vetor independente através da exploração da sua esparsidade. A partir de sua publicação, passaram a ser empregados diretamente sobre grandes problemas de rede e seus conceitos serviram como base para técnicas posteriores.

Através da introdução de novos conceitos e adaptação de outros, o método é completamente geral e resume-se em uma maneira criativa e muito eficiente de localização dos elementos na matriz através de uma tabela chamada *Path Table*.

A título de ilustração, é apresentado um exemplo de uma rede de 20 nós - figura 5.7, seguido de sua matriz, figura 5.8 que representa a estrutura de esparsidade das Matrizes A e L.

Os elementos representados por ● são os originais das duas matrizes e os representados por ○ são elementos *fill-in*.

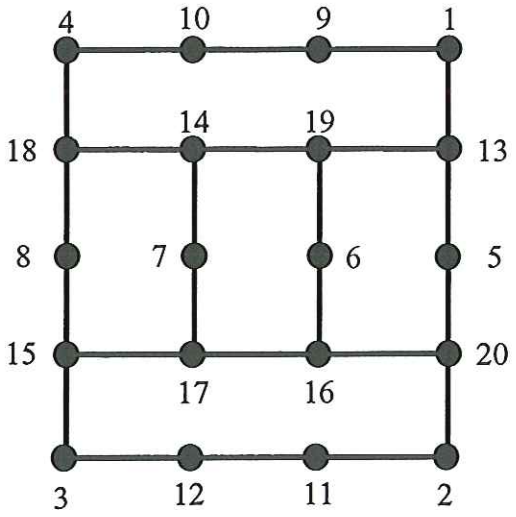


FIGURA 5.7 - Rede exemplo de 20 nós

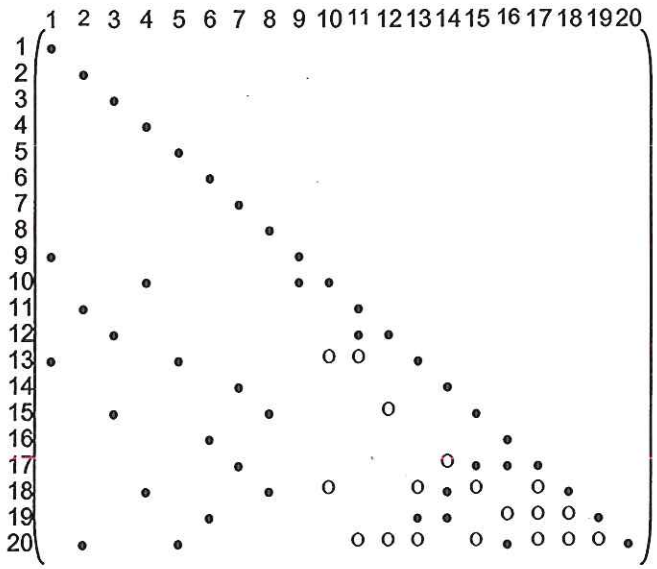


FIGURA 5.8 - Estrutura de Esparsidade das Matrizes *A* e *L*

A *Path Table* é definida como uma lista ordenada dos elementos não zero para as colunas de L (ou linhas de U) e apresenta, para cada nó da coluna k da matriz L (ou linha k

da matriz *U*), o próximo elemento não zero. Assim, a *Path Table* para a rede exemplo, a qual descreve compactamente todas as ligações, é mostrada na tabela 5.1.

TABELA 5.1 - *Path Table*

NÓ	NEXT	NÓ	NEXT
1	9	11	12
2	11	12	15
3	12	13	18
4	10	14	17
5	13	15	17
6	16	16	17
7	14	17	18
8	15	18	19
9	10	19	20
10	13	20	0

Pictograficamente, a *Path Table* pode ser melhor visualizada através de um grafo, como verificado na figura 5.9.

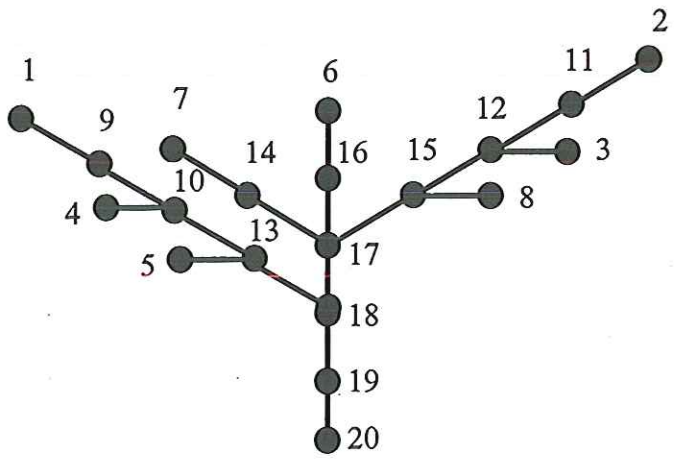


FIGURA 5.9 - *Path Graph*

Um grafo *G=(V,E)* constitui-se em uma outra forma de representação de uma matriz simétrica $A = (a_{ij})$. *V* é o conjunto dos vértices $\{v_1, v_2, \ldots, v_n\}$ e *E* é a coleção de

pares ordenados de elementos de V (eixos), tal que $(v_i, v_j) \in E$ se e somente se $a_{ij} \neq 0$ e $i \neq j$. Os vértices também são chamados nós, pontos, barras, etc e outros nomes para eixos são: ramos, arcos ou linhas. Se $(v_i, v_j) \in E$, então v_i e v_j são denominados adjacentes. O grau de um vértice é o número de vértices adjacentes a ele.

Dessa forma, uma grande contribuição do processo está na maneira de armazenar e localizar os elementos, prendendo-se ao fato de, somente uma pequena percentagem da *Path Table,* ser necessária, pois, na realidade, ela nunca é construída na totalidade. A *Path Table* só é montada para os nós relativos aos elementos diferentes de zero no vetor independente considerado, sendo então chamada de *Singleton*. Um *Singleton* é um vetor com somente um elemento não zero no local *k*. Pode-se dizer, então, que a *Path Table* é formada através da união de diversos *Singleton's*.

Se, por exemplo, for considerado um vetor esparso com elementos nas posições 2, 6, 7 e 12, os *Singleton's* relativos a cada elemento seriam: {2, 11, 12, 15, 17, 18, 19, 20} , {6, 16} e {7, 14}, sendo que o *Singleton* referente ao elemento 12 já está embutido no primeiro vetor. O sub-grafo associado está representado na figura 5.10.

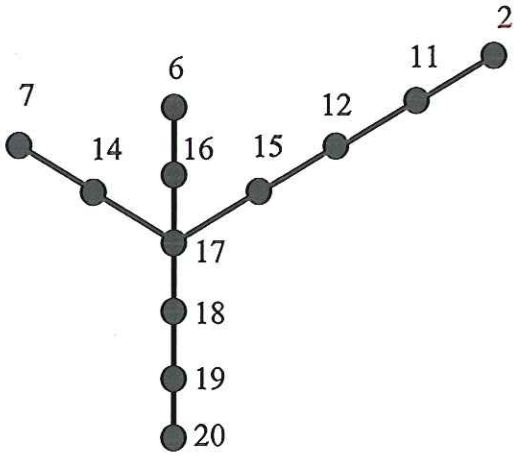


FIGURA 5.10 - Sub-grafo

A solução do sistema linear é dada, portanto, por sua fatoração padrão $A = LDU$ e então:

$$y = D^{-1}.\ L^{-1}.\ b \tag{5.15}$$

$$x = U^{-1}.\ y \tag{5.16}$$

A eq. 5.15 define uma seqüência de operações de *forward substitution* sobre LD, enquanto a eq. 5.16 uma seqüência de operações *back substitution* sobre U. Para o método dos vetores esparsos é essencial que a *forward solution* seja executada por colunas e que a *back solution* por linhas.

Se o vetor b é esparso, somente um subconjunto de colunas de L é necessário para a *forward solution* da eq. 5.15 sendo chamado *fast forward* - *FF*. Por outro lado, se somente certos elementos do vetor x são desejados, somente um subconjunto de linhas de U é necessário para a *back solution* da eq. 5.16, ficando determinado como *fast back* - *FB*. As colunas de *FF* correspondem uma por uma com os elementos não zero no vetor y da eq. 5.15, enquanto as linhas de *FB* correspondem uma por uma com o subconjunto desejado de x na eq. 5.16. A tarefa essencial do método é identificar, eficientemente, os subconjuntos L e U para *FF* e *FB*, respectivamente.

O subconjunto de colunas para *FF* é uma função das estruturas de esparsidade de L e b e o subconjunto de linhas para *FB* é também uma função da estrutura de esparsidade de U e x. As estruturas de esparsidade de L e U, por sua vez, são funções da estrutura de esparsidade de A e do algoritmo de ordenação.

A chave para a eficiência do método está no conceito de *path* de fatoração, que é definido como uma lista ordenada de colunas de L para *FF* ou linhas de U para *FB*. Um *path*

é executado em ordem para frente para *FF* e em ordem reversa para *FB*. O algoritmo abaixo é utilizado para determinação do *path* de fatoração.

ALGORITMO:

1 - Faça k = local do menor número dos elementos não zero em ***b***
2 - Execute operações de *forward solution* definida pelas colunas k de ***L*** sobre ***b***
3 - Pesquise para frente em ***b*** a partir de k pelo próximo maior número do elemento não zero. Faça k = este número
4 - Se k = N, fim. Senão retorne para o passo 2.

Assim, de acordo com os autores, tudo que é necessário para a exploração dos vetores esparsos é que os elementos não zero sejam topologicamente agrupados ao invés de randomicamente espalhados.

5.6 - Método da Substituição Esparsa *Forward* e *Back* - ALVARADO, MONG & ENNS (1985)

Utilizando-se da idéia central dos métodos de Vetores Esparsos para a determinação de conjuntos de interesse, a técnica proposta pelos autores chega às soluções através do uso das substituições esparsas *forward* e *back*. Até então, a substituição esparsa *back* não era citada na literatura, razão pelo interesse em seu uso.

Tomando o vetor ***b*** das injeções de corrente, percebe-se que o mesmo induz a um problema de substituição esparsa *forward* e o vetor de saída - esparso - induz a um problema de substituição esparsa *back*. Ambas são muito similares, havendo duas maneiras de acesso aos elementos:

- A primeira procura por valores diferentes de zero, no vetor $\boldsymbol{b}$, no decorrer da substituição, pulando as colunas onde há zeros;
- A segunda não procura, prevê as posições onde os não zeros serão posicionados.

A substituição esparsa *back* está associada à segunda maneira.

A implementação dos algoritmos relativos às substituições esparsas *forward* e *back* originou um programa com uma interface bastante amigável, permitindo o acesso interativo às rotinas através do monitor do computador e execução de macros, levando o usuário à sociabilidade com a máquina, sendo possível, até mesmo, efetuar-se simulações de diferentes cenários.

5.7 - Método da Refatoração Parcial da Matriz - CHAN & BRANDWAJN (1986)

Os métodos de refatoração parcial já eram utilizados nesta época, porém com pouca expressão em sistemas de potência dada sua baixa eficiência. No entanto, a união do conhecimento destes métodos com a observação de que a alteração de uma linha/coluna da matriz afeta somente um pequeno subconjunto de linhas/colunas subseqüentes e, ainda, as informações recém-adquiridas sobre vetores esparsos, levaram os autores a apresentarem dois novos métodos de refatoração parcial: *PR1* e *PR2*, considerados tão eficientes para aplicações em sistemas de energia que, segundo a apreciação dos autores BRIGHT, DEMAREE, HUGHES & TINNEY (p. 200): "*se estas técnicas não fossem conhecidas, o fluxo de potência ótimo seria completamente diferente e, certamente, muito menos eficiente*".

Tomando como exemplo a rede de 20 nós da figura 5.7 e sua matriz representada na figura 5.8, se for considerada uma modificação de elementos na linha/coluna 2, os fatores da matriz afetados podem ser encontrados diretamente da fatoração simbólica - figura 5.11. Observa-se que os fatores modificados estão localizados nas linhas/colunas 11, 12, 15, 17, 18, 19 e 20. Os outros fatores da matriz, abaixo da linha/coluna 2, não são afetados.

As linhas/colunas afetadas podem também ser encontradas através do *path graph* da matriz - figura 5.9. O *path* para o nó 2 tem exatamente as linhas/colunas encontradas previamente na fatoração simbólica. Quando mais do que uma linha/coluna é modificada, as linhas/colunas afetadas são dadas pela união dos *path singleton*. O algoritmo para encontrar os *path's* para um ou mais nós é o mesmo dos vetores esparsos.

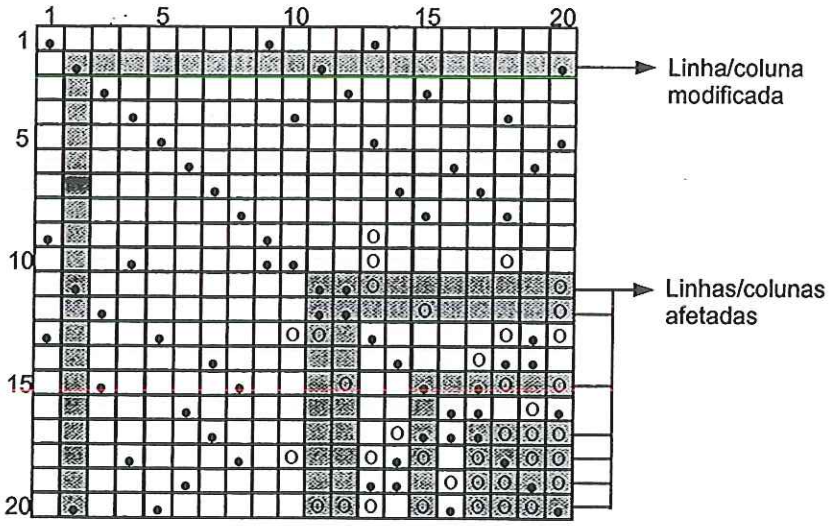


FIGURA 5.11 - Matriz que apresenta as modificações na linha/coluna 2

As linhas/colunas atualizadas pelos novos métodos de refatoração são direcionadas pelos *path's* associados com as entradas da matriz modificada. Chama-se matriz modificada

àquela que, em virtude da formulação do problema, teve alguns de seus elementos alterados, originando o sistema:

$$A'x = b \tag{5.17}$$

Tal matriz modificada é representada por:

$$A' = A + M\,\Delta A\,M^t \tag{5.18}$$

sendo ΔA uma matriz mxm que contém as modificações de A (obs.: n é a dimensão de A e $m \leq n$) e M é uma matriz de conexão nxm.

Para solução deste tipo de sistema existem vários procedimentos específicos chamados métodos de compensação (ALSAÇ, STOTT & TINNEY - 1983), cujo objetivo básico é compensar as mudanças em A sem modificar os fatores LDU. Estes são bem adaptados para aplicações em que:

- o número de mudanças é pequeno;
- as modificações não são permanentes;
- as equações modificadas não precisam ser resolvidas repetidamente.

Levando isto em consideração, os métodos de refatoração parcial propostos são mais eficientes se estas condições não aparecerem reunidas.

Assim sendo, o método *PR1* funciona da seguinte maneira: refatora todas as linhas/colunas que compõe o *path*. É orientado por três passos:

1. Identifica todas as linhas/colunas que serão modificadas;
2. Encontra os *path's* destas linhas/colunas;
3. Lê novamente e refatora as linhas/colunas do *path* usando o algoritmo de fatoração original.

No método *PR2*, o *path* de fatoração é usado para identificar as linhas/colunas que serão atualizadas. As modificações na matriz são como em 5.18. O algoritmo está especificado a seguir e, inicialmente, ***L(.)***, ***D(.)*** e ***U(.)*** contêm os fatores ***LDU*** da matriz; ***M(.)*** e ***N(.)*** contêm os elementos dos vetores M e M^t; e $B = \Delta A$.

ALGORITMO:

```
Encontre os path's para os não-zeros nos vetores M(.) e N(.)
Para I = elementos do path
     D(I) = D(I) + B*M(I)*N(I)
     C1 = B*M(I)
     C2 = B*N(I)
     Para I = todos os elementos não zeros fora da diagonal
          J = coluna
          K = Ukj
          M(J) = M(J) - M(I)*L(K)
          N(J) = N(J) - N(I)*U(K)
          U(K) = U(K) + C1*N(J)/D(I)
          L(K) = L(K) + C2*M(J)/D(I)
     Fim
     B = B - C1 * C2/D(I)
Fim
```

Analisando os dois métodos, verifica-se que os mecanismos dos dois são inteiramente diferentes: *PR1* literalmente refatora as linhas/colunas afetadas quase exatamente como o normal e merece o nome de refatoração parcial; *PR2* modifica os fatores das linhas/colunas diretamente e é normalmente, e mais apropriadamente, chamado de atualização dos fatores.

A programação destes novos métodos é completemente especial. O desafio foi desenvolver um esquema de programação que não se desgastasse em operações aritméticas. Uma implementação pobre perderia muito do potencial das técnicas.

Vale observar que o melhor entre os dois métodos depende das necessidades da aplicação.

5.8 - Melhoria na Técnica de Matriz/Vetor Esparso - Aplicações nos Cálculos do Fluxo de Potência On-Line - RISTANOVIC', BJERLOGRLIC'& BABIC' (1989)

A principal proposta dos autores, neste trabalho, é apontar melhorias tanto nas técnicas de esparsidade (mais especificamente relacionadas à refatoração parcial), quanto em intervenções na programação do fluxo de potência on-line, que podem agilizar significativamente os programas para fluxo de potência AC, fluxo de potência ótimo baseado em Newton e algumas outras aplicações.

Considerando, por exemplo, o fluxo de potência on-line, as seguintes necessidades devem ser satisfeitas em termos de programação: confiabilidade de convergência, alta velocidade, eficiência no tratamento dos limites impostos e ajuste dos *tap's* ULTC. O método sugerido, *PMR* - refatoração parcial da matriz - visa, então, atender estas necessidades utilizando-se dos conceitos de vetores esparsos.

O grande enfoque do mesmo está na ordenação, para que a aplicação da refatoração parcial possa ser mais eficiente. Uma junção das técnicas de ordenação - TINNEY 2 (1967),

GÓMEZ & FRANQUELO (1988a) e BETANCOURT (1988) - deu origem a um esquema mais conveniente para a aplicação enfocada. Para tanto, formalizou-se o conceito a seguir: o tamanho do nó do *path* de fatoração é igual ao número de operações com ponto flutuante para a fatoração deste nó pelos nós precedentes no mesmo *path*. Abaixo aparece um algoritmo para a ordenação do método.

ALGORITMO:

```
1 - Faça k = 1
2 - O pivô é o nó com nt(i) = 0 e nz(i) mínimo.
    Se não há nós, vá para 3
    Se há um, vá para 5
    Se há mais que um, vá para 4
3 - O Pivô é o nó PVT com o mínimo grau neste passo de
    eliminação
    Se há somente um nó nestas condições, vá para 5
    Se há mais do que um, vá para 4
4 - O pivô é o nó com nv(i) mínimo
5 - Eliminar o nó pivô. Para todo nó adjacente não eliminado,
    calcule o tamanho do nó de acordo com a fórmula 5.15 e
    faça nz(j) = nz(j) - 1 para cada par de nós não
    eliminados m e n não adjacentes, mas adjacentes a i. Faça
    nz(m) = nz(m) + 1, nz(n) = nz(n) + 1.
    Se o nó pivô é do tipo PVT, então para todos os nós não
    eliminados adjacentes a ele, faça nt(j) = nt(j) - 1.
6 - Faça k = k + 1
    Se k < n vá para 2
```

Sendo:

- nz(i) = número de nós não eliminados adjacentes a i
- nt(i) = número de nós PVT não eliminados adjacentes a i
- nv(i) = tamanho do nó i (inicialmente zero)
- PVT = nó ao lado do tap do transformador

$$L_m = max\{ L_j + nz(j) * [nz(j) + 3], L_m \} \qquad (5.19)$$

Devido à simetria da matriz A, a pesquisa durante a fatoração ordenada pode ser dividida ao meio. Desta maneira, 5% do tempo de processamento podem ser eliminados, comparado com o procedimento padrão. Usando suas características estruturais, pode-se executar a fatoração simbólica e numérica simultaneamente, conseguindo uma redução de 20 a 30% do tempo de processamento.

O *PMR* é executado por re-leitura, isto é, as colunas e linhas afetadas no *path* de fatoração são lidas novamente com os elementos originais e refatoradas. Entretanto, durante a refatoração parcial, as colunas e linhas em questão são afetadas não somente pelo mesmo *path*, mas por um grande número de colunas e linhas fora deste *path*, aumentando a pesquisa e o número de operações. Para evitar isto, o *PMR* é tratado com correção reversa tornando-se *RCPMR*, que é então executado sem pesquisa e re-leitura das linhas e colunas do *path* de fatoração.

A eficiência do *RCPMR* é significativamente melhorada quando associada com o esquema de ordenação proposto. Do ponto de vista matemático, o tamanho do *path* de fatoração define o número de operações necessárias para a refatoração parcial da matriz. Quando a ordenação é aplicada, o tamanho do *path* no *path graph* é diminuído e o mesmo torna-se extremamente radial.

Uma das mais importantes vantagens do *RCPMR* é que não há problemas com erros de aproximação para um grande número de mudanças simultâneas, como há no método de atualização dos fatores da matriz (CHAN & BRANDWAJN, 1986).

Na implementação dos métodos - *PMR* e *RCPMR* - o esquema de armazenamento empregado para a matriz é o de listas encadeadas para os elementos fora da diagonal e

vetores separados para os elementos da diagonal, como proposto por TINNEY, POWELL & WALKER (1975) e ZOLLENKOPF (1971).

5.9 - Fatores Inversos da Matriz Esparsa - ENNS, TINNEY & ALVARADO (1990)

O método sugerido pelos autores enfoca a importância das soluções diretas principalmente para uma classe de problemas estáticos, como análise de contingência, e dinâmicos, como estabilidade transitória e eletromagnetismo transitório por integração implícita.

Assim, ao invés de usar os fatores triangulares da matriz para calcular as soluções diretas, como é feito no processamento serial, a sugestão é usar a inversa desses fatores triangulares. Diferentemente da inversa da matriz esparsa, que é cheia, as inversas dos fatores triangulares são esparsas, embora menos esparsos que os próprios fatores. Apesar disso, a esparsidade pode ser melhorada pelo particionamento das inversas dos fatores triangulares.

Já é conhecido que a eq. (2.3) pode ser fatorada sob a forma:

$$A = L.D.L^t \tag{5.20}$$

Para a matriz esparsa A, o cálculo de L é precedido por uma ordenação dos eixos (linhas e colunas da matriz ou nós da rede) para minimizar o nº de elementos não zero. Define-se, então, a matriz:

$$W = L^{-1} \tag{5.21}$$

A i-ésima coluna de W é uma solução "*forward*" para um vetor unitário i (um vetor com 1's na posição i e zero, caso contrário. Portanto, W é triangular inferior e esparsa.

Foi comprovado, experimentalmente, que para uma variedade de sistemas de 324 a 2199 nós, a densidade de elementos não zero em W, variou de 2 a 7%, usando ordenação de mínimo grau. Isto sugere o uso da matriz W para solução de (5.20), como segue:

$$z = Wb \quad , \quad y = D^{-1}z \quad , \quad x = W^t y \tag{5.22}$$

Fazendo uso da matriz W, acima definida, determina-se um novo esquema de ordenação conhecido como ESQUEMA W. Tal esquema é uma modificação do Esquema TINNEY 3, no qual, a cada passo o nó escolhido introduz uma quantidade mínima de elementos adicionais em W ao invés de introduzir em L. Isto é feito pela criação da estrutura simbólica de W simultaneamente com a de L, durante o progresso da ordenação.

O algoritmo do ESQUEMA W é mencionado abaixo:

ALGORITMO:

1 - Inicialização: para cada nó, calcular os *fill-ins* em W que resultariam se o nó fosse eliminado primeiro. Armazene em W-COUNTS. Classifique os nós em ordem crescente de W-COUNTS (isto equivale ao Esquema 1 aplicado à matriz W, que é o mesmo aplicado à L, pois neste estágio inicial as estruturas de L e W são exatamente as mesmas.

2 - Se todos os nós estão ordenados - PARE.
Caso contrário, selecione o nó de menor W-COUNT e, simbolicamente, elimine-os. Adicione *fill-ins* às estruturas de L e W.

3 - Recalcule os W-COUNTS, para nós cujas conexões são mudadas no passo 2. Reclassifique os W-COUNTS e vá para o passo 2.

O esforço computacional para o ESQUEMA W é equivalente ao Esquema TINNEY 3. O recálculo e reordenação somente dos nós afetados pela seleção prévia e eliminação são a chave de uma implementação eficiente.

A ordenação ESQUEMA W é eficaz tanto para processamento serial quanto paralelo. Em comparação com o Esquema 2, cujo objetivo é minimizar o nº de elementos não zero em L para soluções com vetores cheios, o ESQUEMA W é melhor, pois se presta para soluções repetidas com vetores esparsos.

Para sistemas com algumas centenas de nós, o ESQUEMA W trabalha satisfatoriamente na produção da matriz esparsa. Por outro lado, para redes muito grandes, o número de elementos não zero torna-se excessivo em W, razão pela qual foi introduzido o ESQUEMA W PARTICIONADO. A estrutura de uma partição W é determinada pela união das matrizes W_i adjacentes. Estas diferem da matriz identidade somente nas colunas correspondentes aos seus índices. A regra de partição usada é muito simples: quando o número de elementos não zero em uma partição atinge um limite pré-determinado, inicia-se um novo fator. A modificação é efetiva na redução do número de elementos não zero, pois a maioria dos *fill-ins* ocorre precipitadamente em relação ao final da ordenação. Cada fator é iniciado com uma matriz identidade e somente poucos *fill-ins* são produzidos pelos primeiros nós selecionados. Por exemplo, se W_i, W_j e W_k são combinados em uma partição, então a partição terá elementos somente abaixo da diagonal nas colunas i, j e k em adição à diagonal unitária.

A tabela 5.2 apresenta uma comparação entre os Esquemas: 2, W e W PARTICIONADO. "*O*" significa o total de elementos na matriz original, "*L*" apresenta o total de não zeros no fator *L* e "*W*" significa o total de não zeros nos fatores inversos, para cada sistema apresentado. Constam também, desta tabela, observações relativas às partições como: número de nós para cada sistema, número de elementos por partição e o total de partições.

Verifica-se, então, que para grandes problemas de redes de potência o método proposto, aliado ao ESQUEMA W particionado de ordenação, é bastante benéfico para soluções repetidas com vetores esparsos.

TABELA 5.2 - Comparação entre os Esquemas 2, W e W PARTICIONADO e Observações sobre as Partições

Sistema	Matriz	Esquema 2	Esquema W	Esquema W Particionado	Observações sobre partições
284 bus	*O* *L* *W*	514 1.153 6.532	514 1.243 5.176	514 1.236 2.258	230 nós e 1500 elementos na 1ª de 2 partições
535 bus	*O* *L* *W*	680 914 7.277	680 987 4.939	680 944 1.800	493 nós e 1500 elementos na 1ª de 2 partições
600 nós tridiagonal	*O* *L* *W*	600 600 cheia	600 1.677 4.530	600 1.273 2.059	493 nós e 1500 elementos na 1ª de 2 partições
5.838 bus	*O* *L* *W*	9.827 28.620 *	9.827 29.914 437.588	9.827 30.883 52.259	Partições de 4285, 5367 e 5725 com 16000 elementos nas primeiras 3 de 4 partições

(*) muitos elementos - excederam a capacidade de armazenamento

5.10 - Métodos Particionados da Inversa da Matriz Esparsa - ALVARADO, YU & BETANCOURT (1990)

O método abordado anteriormente (Fatores Inversos da Matriz Esparsa) deu o fundamento para o uso das inversas particionadas dos fatores de A. Entretanto, o método não foi considerado prático devido ao número de *fill-ins* nas matrizes W . Embora tenha sido apresentada a idéia das partições W para reduzir o número deles, não foram feitos outros esforços para sua redução a zero. Assim, o método aqui abordado, elaborado sobre a idéia das matrizes W particionadas e vetores esparsos, utilizando-se de componentes fatoradas das inversas de L e U, tem o intuito de apresentar "zero" elementos adicionais.

Como introduzido anteriormente, um *path* de fatoração exibe as relações de precedência das operações sobre as linhas e colunas da matriz. *Lenght* de um nó i, no *path* de fatoração, é definido como o máximo de todos os nós que o precedem no *path graph*. Assim, tomando como base o exemplo do sistema de 20 nós da figura 5.7 e sua matriz associada - figura 5.8 - a figura 5.12 ilustra o *path graph* com o *lenght* de cada nó indicado. Observa-se que o nó 20 é trivial, razão pela qual não é mostrado.

A estrutura de W está contida no *path graph*, ou seja, as entradas não zero na coluna i de W são dadas pelos nós que são incluídos no *path* do nó i. Também as entradas não zero na linha i de W são dadas pelos nós no qual o *path* do nó i está incluído.

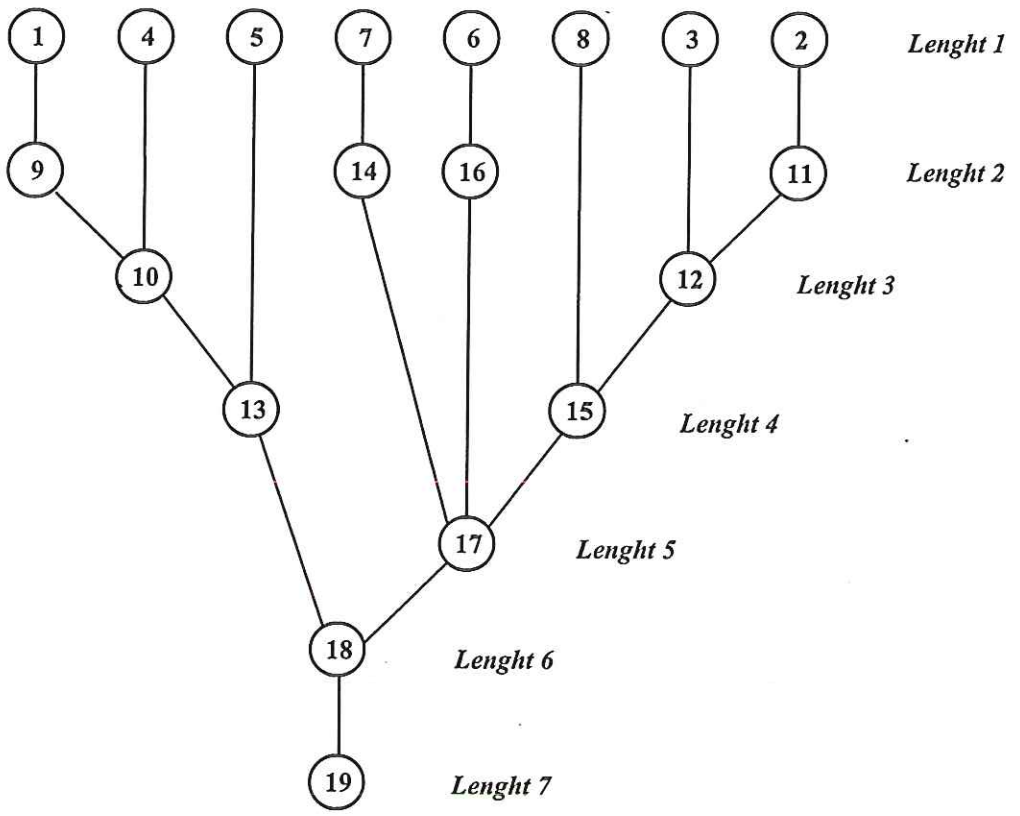


FIGURA 5.12 - *Path* de fatoração ilustrando o conceito de *lenght*

Um acesso à partição W sem produção de novos elementos pode ser baseado no *path graph*. Se todos os nós de um nível são numerados antes que todos os nós do próximo nível, particionando a matriz de acordo com os níveis, resulta em não acréscimo de elementos em W. Portanto, o número de partições em W pode ser feito igual ao máximo *lenght* no *path graph*. Todos os nós do mesmo nível estariam na mesma partição. Nenhum *fill-in* seria criado neste caso, pois os nós dentro de uma partição não seriam predecessores dos outros. Assim, o conceito de *lenght* é equivalente ao conceito de partição.

O objetivo de se utilizar particionamento é reduzir o número de *fill-in* em W. Para tanto, a metodologia proposta utiliza-se de três algoritmos:

- PA1 - particionar a matriz e não produzir *fill-in* em W. Os nós consecutivos são agrupados na mesma partição sem criar qualquer *fill-in* em W, enquanto cada nó é de um ramo diferente no *path graph*;
- PA2 - reduzir o número de partições enquanto ainda não adiciona *fill-in*. Aqui, mais do que um nó do mesmo ramo, é permitido na partição desde que a inclusão do nó não resulte em *fill-in* em W;
- PA3 - permitir poucos *fill-ins* durante a formação de W para reduzir o número de partições.

Para o exemplo da rede de 20 nós, o número de *fill-ins* resultante, se W não é particionado, é 42. Se os algoritmos acima são aplicados, observa-se uma notável redução, como verificado na tabela 5.3.

TABELA 5.3 - Número de Partições e *fill-ins* para os casos *PA1, PA2* e *PA3*

PA1	Nós	1 - 8	9	10 - 11	12 - 14	15 - 16	17	18	19
	W-fills	0	0	0	0	0	0	0	0
PA2	Nós	1 - 8	9	10 - 11	12 - 14	15 - 16	17 - 19		
	W-fills	0	0	0	0	0	0		
PA3	Nós	1 - 9	10 - 14	15 - 19					
	W-fills	1	3	2					

Exemplos e testes foram realizados para sistemas de 118 a 1993 nós. Para cada sistema foram comparadas muitas combinações de algoritmos, sendo considerados os seguintes relativos à ordenação: Tinney - Esquemas 2 e 3; Gómez e Franquelo - 1, 2 e 3 (GF1, GF2 e GF3); Betancourt - MDML e MLMD.

Para cada um destes algoritmos foram obtidos e analisados os itens a seguir:

- o número de elementos da matriz fatorada L;
- o número de partições de acordo com *PA1* e *PA2*;
- o número de partições de acordo com *PA3*, sendo permitidos 1%, 2%, 5%, 10% e 20% de *fill-ins*. Para cada caso, uma quantidade de partições é apresentada.

Os resultados destes itens para o sistema exemplo de 118 nós (179 elementos originais fora da diagonal) estão relacionados na tabela 5.4. Observa-se que o número de cima indica as partições e o de baixo o número de *fill-ins* em W.

TABELA 5.4 - Partições e número de *fill-ins* nas matrizes *W* depois da ordenação

	Esq. 2	Esq. 3	GF1	GF2	GF3	MDML	MLMD
Não zeros em L	265	269	265	264	285	265	303
PA1	57	64	48	19	19	50	12
PA2	51	57	41	13	12	43	6
PA3	22	24	12	6	7	17	6
1% (2)	31	38	29	10	8	29	5
PA3	16	18	12	6	7	13	6
2% (4)	48	52	33	18	15	37	13
PA3	9	12	8	5	5	8	5
5% (9)	69	84	51	37	32	57	20
PA3	6	8	6	4	4	6	4
10% (18)	88	115	79	54	51	80	53
PA3	5	5	4	3	3	4	3
20% (36)	135	160	110	99	80	108	79

5.11 - Técnicas de Aproximação do Vetor Esparso - BACHER, EJEBE & TINNEY (1991)

As técnicas de aproximação do vetor esparso (ASV) aparecem como complementos aos métodos de vetores esparsos, os quais fornecem sua conceituação básica. Para grandes problemas são potencialmente mais vantajosas, pois caracterizam-se pela agilidade na apresentação das soluções. Tal agilidade advém de "pulos" sobre operações que representariam alterações suficientemente pequenas, de acordo com alguns critérios apropriados e que não podem ser desconsiderados.

Tanto as técnicas de vetores esparsos quanto as de aproximação podem ser usadas para vetores estritamente esparsos ou pseudo-esparsos. O aparecimento de um ou de outro está relacionado aos efeitos de mudanças localizadas nos parâmetros da rede e injeções.

Dessa forma, as técnicas de aproximação podem ser derivadas matematicamente, mas a interpretação da rede requer considerações adicionais:

- as variáveis da solução são correções incrementais;
- a fatoração da matriz da rede nodal é análoga à redução da rede na ordem de esparsidade direta;
- uma operação de solução *forward* sobre a coluna k de fatores converte a injeção incremental do nó k nas injeções equivalentes dos nós adjacentes que seguem k na ordenação. Se a injeção do nó k é relativamente pequena, o erro resultante da propagação de seus efeitos para outros nós é também pequeno;
- a substituição *back* é análoga a reconstituir a redução da rede nó por nó. A analogia assume duas formas dependendo se a substituição é por linhas ou por

colunas. Na substituição *back* por colunas, as operações para cada coluna k propaga o efeito da solução completa para a variável k das variáveis dos nós adjacentes ao nó k que o precedem na ordenação. Se a variação da solução para um nó é pequena, o erro resultante da não propagação dos efeitos para outros nós é também pequeno.

Pular as operações que teriam pequenos efeitos acelera a atenuação destes e leva a mais pulos. Os elementos do vetor que teriam sido pequenos sem os pulos, são até mesmo menores, e alguns, que são inicialmente zero, assim permanecem.

Cada pulo depende de um simples número chamado de "valor de erro" (ε). A determinação deste é fundamental para o bom desempenho das técnicas. Se o valor é enorme, o montante de pulos é muito grande; se é muito grande, a perda da precisão pode ser inaceitável; se é muito pequeno, o esforço é perdido.

Quatro novas técnicas são sugeridas, sendo três aproximações de outras já introduzidas pelos métodos de vetores esparsos e a quarta inteiramente nova.

- a primeira técnica, chamada *SKIP FORWARD (SF)*, é uma aproximação da *fast forward (FF)*, na qual as operações que têm relativamente pequenos efeitos são puladas. Para transformar a rotina *FF* em *SF*, basta apenas acrescentar um teste condicional no ponto em que o índice k é incrementado para a próxima coluna no *path* da *FF*, isto é:

```
Se (ABS (y(k)) < ε) então
Execute a substituição SF para a coluna k
```

y(k) = k-ésimo elemento do vetor *y*

ε = valor do erro

Em geral, quando *SF* alcança k, o valor original de y(k) terá sido mudado pelas operações de substituição *forward* precedentes.

- a segunda técnica é a *SKIP BACK* (*SB*). Trata-se de uma aproximação da *fast back (FB)*. É usada somente quando é precedida de uma substituição *forward*. Ela pula a substituição *back* para linhas correspondentes aos locais do vetor que contém zeros. Um único teste é inserido na rotina *FB* para transformá-la em *SF*.

```
Se (ABS (x(k)) = 0) então
Execute a substituição SB para a linha k
```

x(k) = k-ésimo elemento do vetor solução *x*

SB é proveitosa somente para o segmento do *path* de *FB* que coincide com o *path* de *FF*.

- a terceira técnica denominada *SKIP FACTOR UPDATE* (*SFU*) é uma aproximação da *fast update* (*FU*), na qual as operações que têm pequenos efeitos são puladas. A conversão da rotina *FU* para *SFU* é feita através do acréscimo de uma linha de teste no ponto em que o índice k é incrementado para a próxima linha do *path* de *FF* para uma mudança de rank-1 na matriz.

```
Se (ABS (f(k)) < ε) então
     Pule a linha k, atualizando operações na
     linha k
```

f(k) = k-ésimo elemento de um vetor esparso especial usado para *FU*

ε = valor do erro

Do ponto de vista das técnicas de ASV, o vetor f é análogo ao vetor y da *SF*.

- a quarta técnica, *SKIP BACK BY COLUMNS* (*SBC*), não tem correlação com os métodos de vetores esparsos. É uma aproximação da substituição *back*, por colunas, que pula as operações cujos efeitos sobre a solução são relativamente pequenos. Pode ser usada seguindo *FF*, *SF* ou substituição *forward* cheia. A *SBC* não requer um *path*, pois ela faz a varredura de todos os nós do último para o primeiro. Uma rotina padrão para substituição *back* por colunas pode ser convertida para *SBC* com a inserção de um teste no local onde o índice da coluna k é decrementado para a próxima coluna.

```
Se (ABS (x(k)) < ε) então
     Execute a SBC para a coluna k
```

x(k) = elemento parcialmente processado do vetor solução

ε = valor do erro

Quando o processo alcança a coluna k, a solução para x(k) está completa.

Assim, o determinante na escolha da técnica de ASV a ser aplicada é a característica do problema a ser resolvido. De acordo com a necessidade, qualquer uma delas ou combinações delas podem ser utilizadas. Elas são consideradas simples, fáceis de serem implementadas e ainda podem ser substituídas diretamente por sua contraparte SV em algumas aplicações. No entanto, para obter e explorar seu grande potencial, às vezes é necessário adaptá-las fazendo modificações nos algoritmos em que são usadas.

6 - IMPLEMENTAÇÃO E TESTES

Este capítulo apresenta dados sobre a implementação de duas das técnicas abordadas: Bi-fatoração e Eliminação Esparsa e Simétrica de Gauss, que foram comparadas entre si e com o pacote da *Harwell Subroutine Library* - MA28. Estas técnicas foram escolhidas por serem as mais utilizadas na área de sistemas estáticos de energia e, mais particularmente, no Laboratório de Otimização em Sistemas Elétricos de Potência - LOSEP, do Departamento de Engenharia Elétrica da Escola de Engenharia de São Carlos. É importante observar que, apesar de terem sido apresentadas no histórico diversas técnicas de processamento em paralelo, as três técnicas comparadas são essencialmente voltadas para processamento serial.

A técnica da Eliminação Esparsa e Simétrica de Gauss somente resolve problemas com matrizes de coeficientes estritamente simétricas, enquanto que os métodos da Bi-fatoração e MA28 apresentam maior maleabilidade. Dessa maneira, por tratar-se de uma

comparação entre as três técnicas, os testes foram realizados para sistemas com exatamente as mesmas características, isto é, sistemas estritamente simétricos.

6.1 - Armazenamento da Matriz

A título de exemplo, consideremos a rede elétrica de seis nós, representada na figura 6.1, onde são dadas as condutâncias de todos os ramos e a intensidade das injeções de corrente.

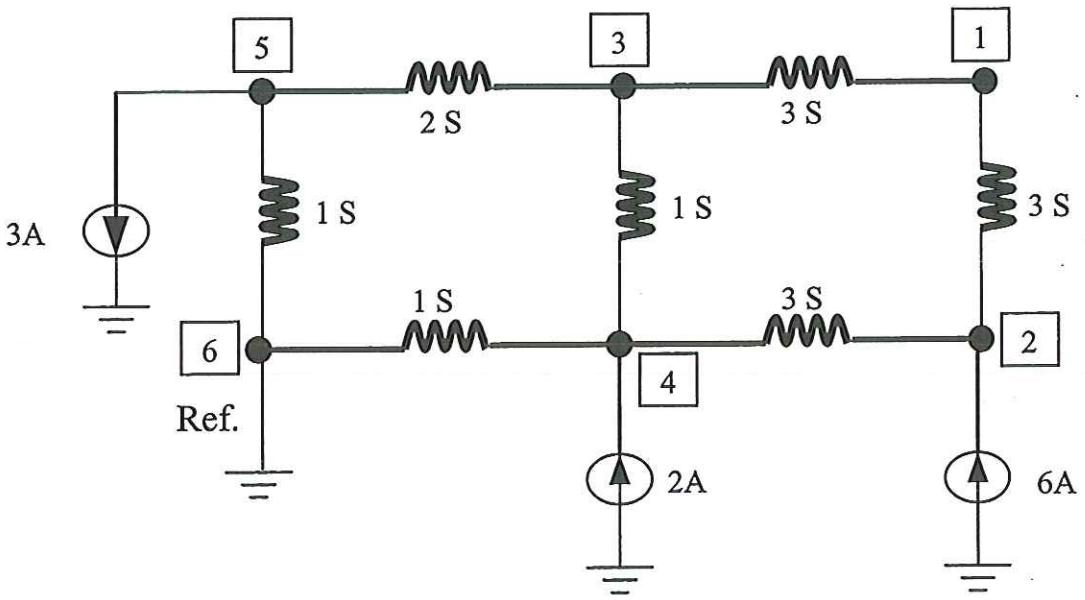


Figura 6.1 - Rede exemplo de seis nós

O sistema linear que representará esta rede, ou seja, $I = YE$ (Y = matriz admitância nodal), observando que *6* é o nó de referência, assume a forma da eq. (6.1).

$$\begin{bmatrix} 6 & -3 & -3 & 0 & 0 \\ -3 & 6 & 0 & -3 & 0 \\ -3 & 0 & 6 & -1 & -2 \\ 0 & -3 & -1 & 5 & 0 \\ 0 & 0 & -2 & 0 & 3 \end{bmatrix} . \begin{bmatrix} E1 \\ E2 \\ E3 \\ E4 \\ E5 \end{bmatrix} = \begin{bmatrix} 0 \\ 6 \\ 0 \\ 2 \\ -3 \end{bmatrix} \tag{6.1}$$

Para efeitos didáticos, teremos em mente a eq. (6.1), porém, a matriz em si não é armazenada explicitamente, sendo que as posições de seus coeficientes são originadas diretamente de um arquivo de dados denominado topologia da rede. Este arquivo é constituído das descrições dos nós e das ligações, sendo que a estrutura de armazenamento final será definida a partir da leitura deste arquivo. O número "9999" indica que não existem mais registros a serem lidos. Para a rede da figura 6.1, o arquivo seria montado de acordo com a figura 6.2:

```
  01                 }
  02                 }
  03                 } Dados dos Nós
  04                 }
  05                 }
9999
  01      02         }
  01      03         } Dados das
  02      04         } Ligações
  03      04         }
  03      05         }
9999                 }
```

Figura 6.2 - Arquivo de dados de uma rede elétrica de 6 nós

Os arquivos de dados são utilizados para as três técnicas abordadas, e a leitura dos mesmos origina as diferentes estruturas de armazenamento, em formato vetorial, usadas em cada uma das técnicas.

A Figura 6.3 apresenta os vetores determinados pelo método da Bi-fatoração, sendo que o significado de cada vetor é especificado a seguir:

- *LCOL*: armazena as posições iniciais das colunas individuais dos elementos que estão em *CE*;
- *NOZE*: contém o número de elementos não zero em cada coluna;
- *NSEQ*: contém, inicialmente, variáveis de 1 a *n* em seqüência ascendente, sendo que no final da simulação, contém a seqüência de pivoteamento, como resultado da estratégia de ordenação empregada;
- *ITAG*: armazena os índices das linhas dos elementos contidos em *CE*;
- *LNXT*: contém o local do próximo elemento não zero em *CE*, em ordem ascendente;
- *CE*: armazena os elementos não zero da matriz *A*.

LCOL	*NOZE*	*NSEQ*
1	3	1
4	3	2
7	4	3
11	3	4
14	2	5

	ITAG	*LNXT*	*CE*
1	1	2	6
2	2	3	-3
3	3	0	-3
4	1	5	-3
5	2	6	6
6	4	0	-3
7	1	8	-3
8	3	9	6
9	4	10	-1
10	5	0	-2
11	2	12	-3
12	3	13	-1
13	4	0	5
14	3	15	-2
15	5	0	3

Figura 6.3 - Esquema de armazenamento do método da Bi-fatoração

O esquema de armazenamento do pacote MA28 também possui suas particularidades. A Figura 6.4 mostra as estruturas vetoriais que o mesmo utiliza, sendo que cada vetor possui os relacionamentos indicados abaixo.

- *A*: contém os elementos diferentes de zero da matriz esparsa;
- *IRN*: cada posição contém o índice da linha do elemento diferente de zero de *A*;
- *ICN*: cada posição contém o índice da coluna do elemento diferente de zero de *A*.

	A	*IRN*	*ICN*
1	6	1	1
2	-3	2	1
3	-3	3	1
4	-3	1	2
5	6	2	2
6	-3	4	2
7	-3	1	3
8	6	3	3
9	-1	4	3
10	-2	5	3
11	-3	2	4
12	-1	3	4
13	5	4	4
14	-2	3	5
15	3	5	5

Figura 6.4 - Esquema de armazenamento segundo pacote MA28

Finalmente, as Figuras 6.5 e 6.6 indicam as estruturas usadas pela Eliminação Esparsa e Simétrica de Gauss relativas às matrizes A e U, respectivamente.

Como visto anteriormente, o esquema de armazenamento empregado pela técnica é denominado "Esquema de Armazenamento sem Compressão", utilizando-se dos seguintes vetores:

- *A*: contém as entradas diferentes de zero do triângulo superior da matriz *A*;
- *JA*: contém os índices das colunas que correspondem às entradas diferentes de zero do vetor *A*;
- *IA*: armazena *n+1* apontadores que indicam as linhas de não zeros em *A* e índices em *JA*, isto é, *A(IA(I))* é a primeira entrada armazenada na i-ésima linha e *A(IA(I+1) – 1)* é a última. Portanto, o tamanho da i-ésima linha é dado por *IA(I+1) – IA(I)*.
- *U*: contém as entradas diferentes de zero do triângulo superior da matriz *U*;
- *JU*: contém os índices das colunas que correspondem às entradas diferentes de zero do vetor *U*;
- *IU*: armazena *n+1* apontadores que indicam as linhas de não zeros em *U* e índices em *JU*.

A	6	-3	-3	6	-3	6	-1	-2	5	3
K	1	2	3	4	5	6	7	8	9	10

JA	1	2	3	2	4	3	4	5	4	5

IA	1	4	6	9	10	11

Figura 6.5 - Esquema de armazenamento sem compressão da matriz *A*

Os vetores referentes à matriz *U* são construídos à semelhança dos relacionados a matriz *A*. Como observado anteriormente, e pela própria definição da matriz *U*, os elementos da diagonal principal são todos iguais a 1 e ocorrem na primeira posição de cada linha; então, para economia de espaço de armazenamento, estes não precisam ser armazenados.

U	a_{12}	a_{13}	a_{14}	a_{24}	a_{34}	a_{35}
k	1	2	3	4	5	6

JU	2	3	4	4	4	5

IU	1	4	5	6	6	6

Figura 6.6 - Esquema de armazenamento sem compressão da matriz *U*

6.2 - Testes e Comparações

A implementação das três rotinas foi realizada através da linguagem *Compaq Visual Fortran* versão 6.1, em um microcomputador *INTEL Pentium III 600E MHZ*, com *Cache* (L2) de 256 Kb, 128 Mb de memória principal padrão PC100 SDRAM, Placa-mãe *INTEL* CA810E (*chipset* 810E), barramento de sistema de 133 *MHZ* e controladora/aceleradora gráfica 2D/3D (on-board). O sistema operacional é o *Microsoft Windows ME*.

Os testes foram aplicados a sistemas de 14, 30, 57 e 118 barras, cuja matriz admitância *Y* possui percentagem de esparsidade de 27,6%, 12,4%, 4,19% e 2,14%, respectivamente. Foi utilizado o pacote MA28 e os programas, relativos às técnicas da Bi-fatoração e Eliminação Esparsa e Simétrica de Gauss, foram desenvolvidos.

Apesar das diferenças metodológicas, a estrutura dos três programas foi didaticamente padronizada, constando das etapas descritas abaixo:

1. Leitura do Arquivo de Dados
2. Topologia da Rede
3. Montagem dos Vetores relacionados a cada técnica
4. Ordenação
5. Fatoração Simbólica (Simulação)
6. Fatoração Numérica (Redução)
7. Solução
8. Saída dos resultados em um arquivo de saída formatado para tal.

As comparações foram relacionadas ao tempo de CPU durante a execução dos programas. Este tempo foi acessado diretamente do sistema através da chamada da função intrínseca do *Fortran* - *GETTIM*. Esta função foi inserida no início das rotinas de interesse, isto é, após a leitura dos arquivos de entrada e topologia da rede, e armazenou hora, minuto, segundos e décimos de segundo inciais, finalizando com mais uma chamada à função *GETTIM* no final da rotina de solução, adquirindo hora, minuto, segundos e décimos de segundo finais.

Os tempos inicial e final foram convertidos para centésimos de segundo e ao final foi encontrada a diferença entre ambos, sendo seguida de sua respectiva conversão, para a obtenção do tempo correto.

Observou-se, entretanto, que o tempo de processamento gira em torno de milésimos de segundo e microssegundos, não sendo registrado pelo relógio interno da máquina, razão pela qual foi empregado um procedimento *DELAY*, isto é, foi inserido um laço de repetição das rotinas 3, 4, 5, 6 e 7, de 1000 unidades, para que se pudesse ter acesso a um tempo maior. Portanto, o tempo total em segundos, que se encontra na tabela 6.1, refere-se a uma proporção do tempo registrado. A figura 6.7 mostra os dados da tabela 6.1

dispostos em um gráfico de colunas para melhor visualização do desempenho entre as três técnicas.

Tabela 6.1 - Tempos de processamento (em segundos) para os quatro sistemas em relação às três técnicas

	14	30	57	118
Bi-fatoração	0,00011	0,00022	0,00066	0,00132
MA28	0,00039	0,00088	0,00181	0,00329
Elim. Esparsa e Simétrica de Gauss	0,00005	0,00021	0,00094	0,00253

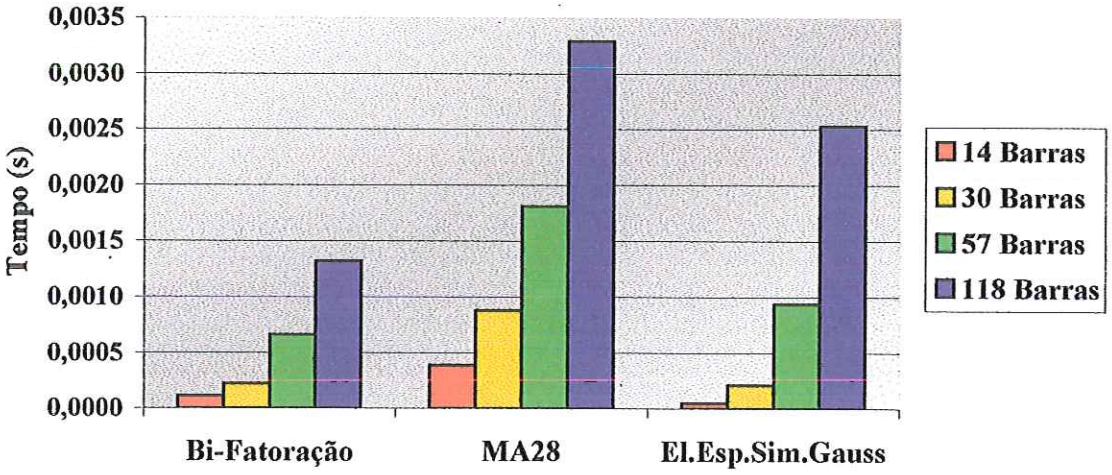


Figura 6.7 - Comparação entre os tempos de processamento (em segundos) para as três técnicas de esparsidade

Observou-se, portanto, nos testes de tempo de processamento, que das três técnicas, a que teve melhor desempenho médio foi a Bi-fatoração. Entretanto, para os dois

primeiros sistemas (14 e 30 barras), a Eliminação Esparsa e Simétrica de Gauss, apresentou melhores resultados que a Bi-fatoração; porém, quando a densidade da matriz começou a aumentar, seu desempenho tornou-se um pouco pior. Também pôde-se observar, que, indiscutivelmente, os piores resultados foram do pacote MA28, que é, das três técnicas abordadas, a mais genérica.

As figuras 6.8 e 6.9 mostram o melhor desempenho da técnica da Eliminação Esparsa e Simétrica de Gauss, para os sistemas de 14 e 30 barras, seguida pela técnica da Bi-fatoração.

Sistema de 14 barras

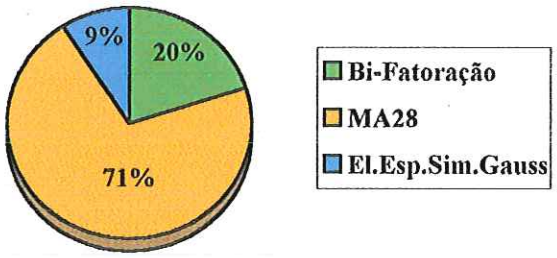


Figura 6.8 - Desempenho das técnicas relativo ao sistema de 14 barras

Sistema de 30 barras

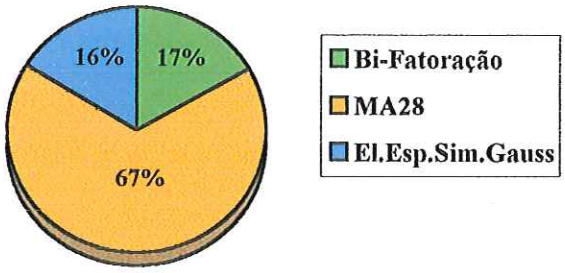


Figura 6.9 - Desempenho das técnicas relativo ao sistema de 30 barras

As figuras 6.10 e 6.11 representam as outras comparações entre as técnicas, para os sistemas de 57 e 118 barras, e confirmam o melhor desempenho da técnica da Bi-fatoração seguida pela Eliminação Esparsa e Simétrica de Gauss. O MA28 teve o pior rendimento para todos os casos.

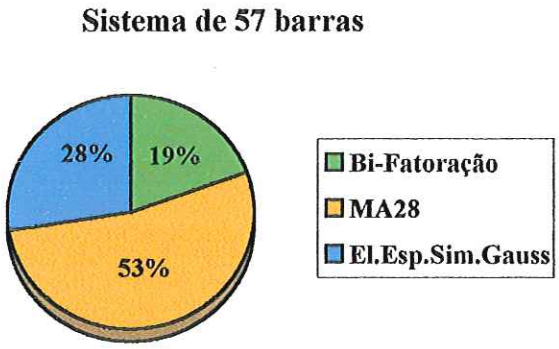


Figura 6.10 - Desempenho das técnicas relativo ao sistema de 57 barras

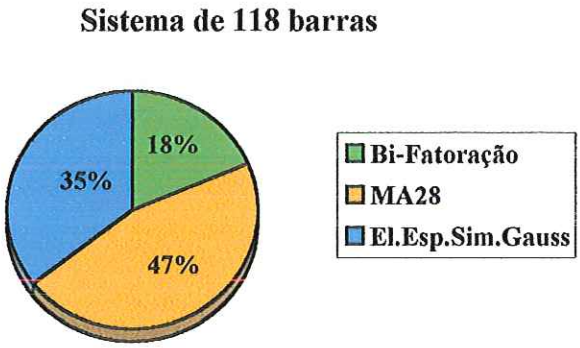


Figura 6.11 - Desempenho das técnicas relativo ao sistema de 118 barras

Dessa maneira, conclui-se que os tempos obtidos representam a associação entre o tamanho do sistema e a técnica adotada.

7 - CONCLUSÕES

A abordagem dada a este trabalho foi a de pesquisar diversas Técnicas de Esparsidade utilizadas para a solução eficiente dos problemas de sistemas estáticos de energia elétrica.

Sabe-se que a resolução eficaz da rede elétrica depende do modelo adotado e está diretamente relacionada com a técnica de esparsidade que o trata. Estas técnicas constituem, portanto, um apanhado de metodologias específicas para o manuseio dos grandes sistemas lineares esparsos envolvidos e objetivam a maximização da eficiência numérica e computacional na obtenção das soluções.

A compilação de uma extensa revisão bibliográfica, enfocando trabalhos realizados na área de sistemas estáticos de energia, foi fundamental para o seu desenvolvimento, pois,

além de fornecer o posicionamento histórico, também apresentou uma ampla revisão do desenvolvimento teórico, relativos às técnicas e também aos modelos de ordenação das matrizes, que aparecem associados.

Complementando o levantamento bibliográfico, foi realizada a implementação de duas destas técnicas - Bi-fatoração e Eliminação Esparsa e Simétrica de Gauss - tendo seus desempenhos comparados entre si e com o pacote da *Harwell Subroutine Library* - MA28, para sistemas de 14, 30, 57 e 118 barras.

Os resultados destas comparações apontaram a técnica da Bi-fatoração como a mais eficiente, na média, tendo os melhores tempos de resposta para os maiores sistemas testados, sendo seguida pela Eliminação Esparsa e Simétrica de Gauss.

Assim, pode-se dizer que a principal contribuição deste trabalho está na organização de uma lista de técnicas de esparsidade e de ordenação, com destaque aos principais atributos de cada uma delas e aos seus algoritmos, alicerçada sobre uma fundamentação teórica básica sobre álgebra matricial, que servirá como ponto de apoio à decisão, aos especialistas da área. Também contribuiu no aspecto de apontar, entre as três técnicas mais empregadas na área, o melhor desempenho da Bi-fatoração para grandes sistemas.

A pesquisa desenvolvida leva a uma seqüência de trabalhos futuros, descritos a seguir:

- Implementação e testes aplicados ao problema de FC e FCO, de outras técnicas paralelas, como: Vetores Esparsos; Substituição Esparsa *Forward* e *Back*;

Métodos Particionados da Inversa da Matriz Esparsa e Aproximação do Vetor Esparso;

- Implementação das técnicas de Refatoração Parcial;
- Associação de técnicas de ordenação recentes às técnicas de esparsidade utilizadas mais freqüentemente;
- Avaliação da viabilidade dos novos pacotes MA da *HSL*-2000.

REFERÊNCIAS BIBLIOGRÁFICAS

ALSAÇ, O.; STOTT, B.; TINNEY, W.F. (1983). Sparsity-oriented Compensation Methods for modified Network Solutions. *IEEE Transactions on Power Apparatus and Systems,* v.PAS-102, n.5, p.1050-60.

ALVARADO, F. L.; YU, D. C.; BETANCOURT, R. (1990). Partitioned Sparse A^{-1} Methods. *IEEE Transactions on Power Systems,* v.05, n.2, p.452-57.

ALVARADO, F. L.; MONG, S. K.; ENNS, M. K. (1985). A fault program with Macros, Monitors and direct compensation in mutual groups. *IEEE Transactions on Power Apparatus and Systems,* v.PAS-104, n.5, p1109-20.

ALVES, A. B.; ASADA, E. N.; MONTICELLI, A. (1999). Critical evaluation of direct and iterative methods for solving Ax=b systems in Power Flow calculations and Contingency Analysis. *IEEE Transactions on Power Systems,* v.14, n.2, p.702-8.

ARAUJO, M.; MEDINA, E.; APONTE, E. (1999) . Spectral Statistics and dynamics of Levy Matrices. *PHYS. Rev. E.,* v.60, n.4, p.3580-88, Parte A.

ARORA, J. S.; LI, G. Y. (1993). Constrained Conjugate Directions methods for design Optimization of large systems. *AIAA J.*, v.31, n.2, p.388-95.

BACHER, R.; EJEBE, G. C.; TINNEY, W. F. (1991). Approximate Sparse Vector techniques for power network solutions. *IEEE Transactions on Power Systems,* v.06, n.1, p.20-28.

BERRY, R.D. (1971). An Optimal Ordering of eletronic circuit equations for a sparse matrix solution. *IEEE Transactions on Circuit Theory,* v.CT-18, p.139-45.

BETANCOURT, R. (1988). An eficient Heuristic ordering algorithm for partial matrix refactorization. *IEEE Transactions on Power Systems,* v.03, n.3, p.1181-87.

BETTERIDGE, T. (1974). An analytic storage allocation model. *Acta Informatica,* v.3 , p. 101-22.

BRITANAK, V.; RAO, K. R. (1999). The fast generalized discrete Fourier transforms: a unified approach to the discrete sinusoidal transforms computation. *Signal Process,* v.79, n.2, p.135-50.

CANNING, F. X. (1992). Sparse Approximation for solving Integral-Equations with Oscillatory Kernels. *SIAM J. SCI. STAT. COMPUT.*, v.13, n.1, p.71-87.

CARAKUSHANSKY, M. S.; LA PENHA, G. M. S. M. (1976). *Introdução à Álgebra Linear*. São Paulo, McGraw Hill.

CARVER, M. B.; MACEWEN, S. R. (1981). On the use of Sparse-Matrix approximation to the Jacobian in integrating large sets of Ordinary Differential-Equations. *SIAM J. SCI. STAT. COMPUT.*, v.2, n.1, p.51-64.

CHEN, T. Y.; DEMMEL, J. W. (2000). Balancing sparse matrices for computing eigenvalues. *Linear Algebra Applicattion,* v.309, n.1-3, p. 261-87.

COOK, J.; GILBERT, J. (1988). Parallel Neural Network simulation using Sparse-Matrix techniques. *Microproc. Microprog.,* v.24, n.1-5, p.621-26.

COSTA, G. R. M.; COSTA, C. E. U.; SOUZA, A. M. (2000). Comparative studies of optimization methods for the optimal power flow problem. *Electric Power Systems Research*, v. 56, p. 249-54.

CURTIS, A. R.; REID, J. K. (1971). The solution of sparse unsymmetric systems of linear equations. *Journal of the Institute of Mathematics and its Applications,* v.8, p.118-24.

DONGARRA, J. J.; DU CROZ, J.; DUFF, I. S.; HAMMARLING, S. (1990). A set of Level 3 basic linear algebra subprograms. *ACM Transactions on Mathematical Software,* v.16, n.1, p.1-17.

DRUD, A. (1978). Optimization code for Non-linear Econometric-Models based on Sparse-Matrix techniques and Reduced Gradients. *Ann. Econ. Soc. Meas*, v.6, n.5, p.563-80.

DUFF, I. S.; REID, J. K. (1974). A comparison of Sparsity orderings for obtaining a Pivotal sequence in Gaussian Elimination. *Journal of the Institute of Mathematics and its Applications,* v.14, p.281-91.

DUFF, I. S.; REID, J. K. (1975). A comparison of some methods for the solution of sparse overdetermined systems of linear equations. *Journal of the Institute of Mathematics and its Applications,* v.17, p.267-80.

DUFF, I. S. (1977). MA28 - A set of Fortran subroutines for sparse unsymmetric linear equations. Report AERE R8730, HMSO, London.

DUFF, I. S. (1977). A survey of Sparse Matrix Research. *Proceedings of the IEEE*, v.06, n.4, p.500-35.

DUFF, I. S.; REID, J. K. (1979). Some design features of Sparse Matrix Code. *ACM Transactions on Mathematical Software,* v.05, n.1, p.19-35.

DUFF, I. S.; REID, J. K. (1984). MA27 - A set of Fortran subroutines for solving sparse symmetric sets of linear equations. Report AERE R10533, HMSO, London.

DUFF, I. S.; ERISMAN, A. M.; REID, J. K. (1986). *Direct Methods for Sparse Matrix*. New York, Oxford University Press.

DUFF, I. S.; GRIMES, R. G.; LEWIS, J. G. (1992). User's Guide for the *Harwell-Boeing* sparse matrix collection (Release 1). Report RAL-92-086, Rutherford Appleton Laboratory, Oxfordshire.

DUFF, I. S.; REID, J. K. (1993). MA48 - A Fortran code for direct solution of sparse unsymmetric linear systems of equations. Report RAL-93-072, Rutherford Appleton Laboratory, Oxfordshire.

DUFF, I. S.; REID, J. K. (1995). MA47, A Fortran code for direct solution of indefinite symmetric linear systems of equations. Report RAL-95-001, Rutherford Appleton Laboratory, Oxfordshire.

DUFF, I. S.; REID, J. K. (1996a). Exploiting zeros on the diagonal in the direct solution of indefinite Sparse Symmetric Linear Systems. *ACM Transactions on Mathematical Software,* v.22, n.2, p.227-57.

DUFF, I. S.; REID, J. K. (1996b). The design of MA48: A Code for the Direct Solution of Sparse Unsymmetric Linear Systems of Equations. *ACM Transactions on Mathematical Software,* v.22, n.2, p.187-226.

EISENSTAT, S. C.; SCHULTZ, M. H.; SHERMAN, A. H. (1981). Algorithms and Data Structures for Sparse Symmetric Gaussian Elimination. *SIAM J. SCI. STAT. COMPUT.*, v.2, n.2, p.225-37.

EISENSTAT, S. C.; GURSKY, M. C.; SCHULTZ, M. H.; SHERMAN, A. H. (1982). YALE SPARSE MATRIX PACKAGE I: the symmetric codes. *International Journal for Numerical Methods in Engineering.*, v.18, p.1145-51.

ENNS, M. K.; TINNEY, W. F.; ALVARADO, F. L. (1990). Sparse Matrix Inverse Factors. *IEEE Transactions on Power Systems,* v.05, n.2, p.466-73.

ERISMAN, A. M.; SPIES, G. E. (1972). Exploiting problem characteristics in the sparse matrix approach to frequency domain analysis. *IEEE Transactions Circuit Theory,* v.CT-19, p.260-69.

ERISMAN, A. M.; TINNEY, W. F. (1975). On computing certain elements of the Inverse of a Sparse Matrix. *Communications of the ACM*, v.18, n.3, p.177-79.

EVES, H. (1997). *Introdução à História da Matemática*. Trad. Por Hygino H. Domingues. Campinas, Editora da UNICAMP. Cap.13, p.519-584: As primeira décadas do século XIX e a Libertação da Geometria e da Álgebra.

FARRCHER, P.; GROTE, P. (1971). Development of a hybrid simulation for extreme automobile manuevers. In: *Proc. Summer Computer Conference*, Boston.

FREITAS, F. D.; COSTA, A. S. (1999). Computationally efficient optimal control methods applied to power systems. *IEEE Transactions on Power Systems,* v.14, n.3, p.1036-43.

GAMM, A. Z.; KRUMM, L. A.; SHER, I. A. (1965). General Principles of calculating the steady-state operation of an electric system with division into subsystems. *IZV. Akad. Nauk. SSSR, Energ. Transp.*, n.6, p.7-15.

GENTLEMAN, W. M. (1974). Regression problems and the QR decomposition. *Bull. Inst. Math. Appl.*, v.10, p. 195-97.

GILL, P. E.; MURRAY, W.; SAUNDERS, M. A.; et al. (1984). Sparse-Matrix methods in Optimization. . *SIAM J. SCI. STAT. COMPUT.*, v.5, n.3, p.562-89.

GOOI, H. B.; WANG, Y. Q. (1998). Efficient ordering algorithms for sparse matrix/vector methods. *Int. J. Elec. Power Systems,* v.20,n.1,p.53-59.

GOLUB, G. H.; VAN LOAN, C. F. (1996). *Matrix Computations*. 3.ed. Baltimore, Johns Hopkins.

GÓMEZ, A.; FRANQUELO, L. G. (1988a). Node ordering algorithms for Sparse Vector method improvement. *IEEE Transactions on Power Systems,* v.03, n.1, p.73-79.

GÓMEZ, A.; FRANQUELO, L. G. (1988b). An efficient ordering algorithm to improve Sparse Vector methods. *IEEE Transactions on Power Systems,* v.03, n.4, p.1538-44.

GRAHAM, M. F.; JENNINGS, G. (1979). Efficient Sparse-Matrix implementation for Direct methods in Reservoir Simulation. *J. Petrol. Technol.,* v.31, n.1, p.69.

GUSTAVSON, F. G. (1972). Some Basic Techniques for solving sparse systems of linear equations. *Sparse Matrices and their Applications*, D.J. Rose and R.A. Willoughby, eds, Plenum Press, p.41-52.

HACHTEL, G. D.; BRAYTON, R. K.; GUSTAVSON, F. G. (1971). The Sparse Tableau approach to network analysis and design. *IEEE Transactions on Circuit Theory,* v.CT-18, p.101-13.

HARARY, F. (1970). Sparse Matrix and Graph Theory. In: *Proc. Conf. Oxford*. London: Academic Press.

HASHIMOTO, M. (1970). A method for solving large sparse matrix equations reduced from Fredholm integral equations of the second kind. *J. Assoc. Comput. Mach,* v.17, p.629-36.

HENNESSY, D.; BUCHANAN, B.; SUBRAMANIAN, D.; et al. (2000). Statistical methods for the objective design of screening procedures for macromolecular crystallization. *ACTA Crystallogr.*, v.D-56, p.817-27, Parte 7.

HONG, Y. Y.; WANG, F. M. (2000). Development of three-phase Newton optimal power flow for studying imbalance/security in transmission systems. *Electr. Power Systems Res.*, v.55, n.1, p.39-48.

HSIEH, H. Y.; GHAUSI, M.S. (1972). On Optimal-Pivoting Algorithms in Sparse Matrix. *IEEE Transactions on Circuit Theory,* v.CT-19, n.1, p.93-96.

HSIEH, H. Y. (1974). Pivoting-Order Computation Method for Large Random Sparse Systems. *IEEE Transactions on Circuits and Systems,* v.CAS-21, n.2, p.225-30.

JACOBSON, M. Z.; LU, R.; TURCO, R. P.; et al. (1996). Development and application of a new air pollution modeling system .1. Gas-phase simulations. *Atmos. Environ.,* v.30, n.12, p.1939-63.

JENNINGS, A.; ORR, D. R. L. (1971). Application of the simultaneous iteration methods to undamped vibration problems. *Int. J. Numer. Methods Eng.* ,v.3, p.13-24.

JHA, R.; HERATH, S.; MUSIAKE, K. (2000). River network solution for a distributed hydrological model and applications. *Hydrol. Process.,* v.14, n.3, p. 575-92.

JOHNSEN, T. L.; ROY, J. R. (1974). On systems of linear equations of the form $A^TAx=b$ error analysis and certain consequences for structural applications. *Comput. Meth. Appl. Mech. Eng.*, v.3, p. 357-74.

JONES, M. T.; PLASSMANN, P. E. (2000). Unstructured mesh computations on networks of workstations. *Comput-Aided Civ. Inf.,* v.15, n.3, p.196-208.

KAVEH, A. (1991). Graphs and Structures. *Comput. Struct.,* v.40, n.4, p.893-901.

LEE, C. L.; CHEN, N. (1993). Sparse Vector method improvements via minimum inverse fill-in ordering. *IEEE Transactions on Power Systems,* v.8, n.1, p.239-45.

LEIMBACH, K. R.; ZELLER, C. (1982). Advantages of using a node-oriented Saprse-Matrix solver in nuclear-plant dynamica anlyses. *Nucl. Eng. Des.,* v.70, n.1, p.85-100.

LINIGER, W. M.; WILLOUGHBY, R. A. (1970). Efficient integration methods for stiff systems of ordinary differential equations. *SIAM J. Numer. Anal.*, v.7, p.47-66.

MAASS, P.; PEREVERZEV, S. V.; RAMLAU, R.; et al. (2001). An adaptative discretization for Tikhonov-Phillips regularization with a posteriori parameter selection. *Numer. Math.,* v.87, n.3, p. 485-502.

MARKOWITZ, H. M. (1957). The elimination form of the inverse and its application to linear programming. *Management Science*, v.3, p.255-69.

MARTELLI, A.; MONTANARI, U. (1972). Nonserial dynamic programming: On the optimal strategy of variable elimination for the rectangular lattice. *J. Math. Anal. Appl.*, v.40, p.226-42.

McLAY, R. T.; SWIFT, S.; CAREY, G. F. (1996). Maximizing sparse matrix-vector product performance on RISC based MIMD computers. *J. Parallel. Distr. Com.,* v.37, n.2, p.146-58.

MEIJSTER, A.; WUBS, F. (2000). Towards an implementation of a multilevel ILU preconditioner on shared-memory computers. *Lect. Notes Comput. SC.* , v.1823, p.109-18.

MISLEVY, R. J.; BEATON, A. E.; KAPLAN, B.; et al. (1992). Estimating Population characteristics from Sparse-Matrix Samples of item Responses. *J. Educ. Meas,* v.29, n.2, p.133-61.

MONTICELLI, A. (1983). *Fluxo de carga em redes de energia elétrica.* São Paulo, Edgard Blucher Ltda. Cap.3, p.29-51.

MOROZOWSKI FILHO, M. (1981) . *Matrizes esparsas em redes de potência:* técnicas de operação. Santa Catarina, LTC.

MUHLBACHER, J. R. (1980). Case-Study on programming in a paged environment: an implementation of Gustavson Fast algorithm for Sparse-Matrix multiplication. *Angew Inform.,* n.2, p.67-71.

NEMETH, K.; COULAUD, O.; MONARD, G.; et al. (2000). Linear scaling algorithm for the coordinate transformation problem of molecular geometry optimization. *J. Chem. Phys.,* v.113, n.14, p.5598-5603.

OGBUOBIRI, G. C.; TINNEY, W. F.; WALKER, J. W. (1970). Sparsity-Direct decomposition for Gaussian Eliminationon Matrices. *IEEE Transactions on Power Apparatus and Systems,* v.PAS-89, n.1, p.141-50.

PACE, R. K. (1997). Performing large spatial regressions and autoregressions. *Econ. Lett,* v.54, n.3, p. 283-91.

PARK, S.; KIM, W. J.; SEOL, T.; et al. (2000). LPABO: a program for interior point methods for linear programming. *Asia Pac. J. Oper. Res.,* v.17, n.1, p. 81-100.

PETIT, L.; BEIDEN, S. V.; TEMMERMAN, W. M.; et al. (2000). Screened real-space Korringa-Kohn-Rostoker description of the relativistic and magnetic properties of transition metals. *J. Phys-Condens. Mat.,* v.12, n.39, p.8439-54.

RADHAKRISHNAN, K.; CHEW, W. C. (1999). Full-wave analysis of multicondutor transmission lines on anisotropic inhomogeneous substrates. *IEEE T Microw Theory,* v.47, n.9, p.1764-70, Parte 2.

RISTANOVIC', P.; BJELOGRLIC', M.; BABIC', B. S. (1989). Improvements in Sparse Matrix/Vector Technique Applications for on-line Load Flow Calculation. *IEEE Transactions on Power Systems,* v. 4, n.1, p.190-96.

ROGERS, A. (1971). *Matrix Methods in Urban and Regional Analysis*. San Francisco, CA: Holden-Day.

ROOS, I. C.; HARARY, F. (1959). A description of streingthening and weakening members of a group. *Sociometry,* v. 22, p.139-47.

SATO, N.; TINNEY, W. F. (1963). Techniques for Exploiting the Sparsity of the Network Admittance Matrix. *IEEE Transactions on Power Apparatus and Systems,* v.12, n.82, p.944-50.

SCHNABEL, R. B.; ESKOW, E. (1990). A new modified Cholesky Factorization. . *SIAM J. SCI. STAT. COMPUT.*, v.11, n.6, p.1136-58.

SHERMAN, M. C.; BRANDWAJN, V. (1986). Partial Matrix Refactorization. *IEEE Transactions on Power Systems,* v. PWRS-1, n.1, p.193-200.

SHIPLEY, R. B.; COLEMAN, D. A. (1959). A New Direct Matrix Inversion Method. *Transactions of the American Institute of Electrical Engineers, Part I, Communication and Eletronics*, n.78, p.568-72.

SIGITANI, T.; IIGUNI, Y.; MAEDA, H. (2000). Progressive cross-section display of 3D medical images. *Med. Biol. Eng. Comput.,* v.38, n.2, p.140-49.

SOMMERFELD, J. T.; SONDHI, D. K.; SPURLOCK, J. M.; et al. (1981). Using of Sparse-Matrix methods in material balance calculations. *Chem. Eng. J. Bioch. Eng.,* v.22, n.2, p.89-94.

SOUZA, A. M. (1998). O Método Primal Dual Barreira Logarítmica aplicado ao problema de Fluxo de Carga Ótimo. São Carlos. 93p. Dissertação (Mestrado) - Escola de Engenharia de São Carlos, Universidade de São Paulo.

STEVENSON JR, W. D. (1986). *Elementos de Análise de Sistemas de Potência*. São Paulo, McGraw Hill.

STEWART, G. W. (1973) . *Introduction to Matrix Computations.* New York, Academic Press.

STOICA, P.; MOSES, R. L.; SODERSTROM, T.; et al. (1991). Optimal High-Order Yule-Walker estimation of Sinusoidal Frequences. *IEEE T Signal Proces.,* v.39, n.6, p.1360-68.

SUN, D.I.; ASHLEY, B.; HUGUES,A.; TINNEY, W.F. (1984). Optimal Power Flow by Newton Approach. *IEEE Transactions on Power Apparatus and Systems,* v. pas 103, n. 10, p. 2864-75.

TAKAHASHI; KAZUHIRA; FAGAN; JOHN and CHIN MO-SHING. (1973). Formation of a Sparse Bus Impedance matrix and its application to short circuit study. 8th *PICA Conference Proceedings*, Minneapolis, p.4-6.

TAOKA, H.; ABE, S. (1986). Fast Solution Method of Sparse Network Equations with an Array Processor. *IEEE Transactions on Power Systems,* v. PWRS-1, n.4, p.215-21.

TIER, B.; SMITH, S. P. (1989). Use of Sparse-Matrix absorption in animal Breeding. *Genet. Sel. Evol.,* v.21, n.4, p.457-66.

TINNEY, W. F.; HART, C. E. (1967). Power Flow solution by Newton's Method. *IEEE Transactions on Power Apparatus and Systems,* v.11, n.86, p.1449-60.

TINNEY, W. F.; WALKER, J. W. (1967). Direct Solutions of Sparse Network Equations by Optimally Ordered Triangular Factorization. *Proceedings of the Institute of Electrical and Eletronics Engineers,* v.(11), n.55, p.1801-9.

TINNEY, W. F.; POWELL, W. L.; PETERSON, N. M. (1974). Sparsity-Oriented Network Reduction. *IEEE Transactions on Power Apparatus and Systems,* v.PA-93, n.1, p.6.

TINNEY, W. F.; POWELL, W. L.; WALKER, J. W. (1975). Programming of Sparsity-Directed Ordering Schemes. *Proceedings PSCC*, Cambridge, U.K.

TINNEY, W. F.; BRANDWAJN, V.; CHAN, S. M. (1985). Sparse Vector Methods. *IEEE Transactions on Power Apparatus and Systems,* v. PAS-104, n.2, p.295-301.

VAN NESS, J. E.; GRIFFIN, J. H. (1961). Elimination Methods for Load-Flow Studies. *Transaction of the American Institute of Electrical Engineers, Part III, Power Apparatus and Systems*, v.6, n. 80, p.299-304.

VARONA, L.; MISZTAL, I. (1999). Prediction of parental dominance combinations for planned matings, methodology and simulation results. *J. Dairy SCI,* v.82, n.10, p.2186-91.

VEMPATI, N.; SLUTSKER, I. W.; TINNEY, W.F. (1992). Orthogonal Sparse Vector Methods. *IEEE Transactions on Power Systems,* v.7, n.2, p.926-32.

WANG, J.; YU, J. D.; YONG, Y. K.; et al. (2000). A new theory for electroded piezoelectric plates and its finite element application for the forced vibrations of quartz crystal resonators. *Int. J. Solids Struct.,* v.37, n.40, p.5653-73.

WANG, Q. L.; KIM, K.; YOON, C. S. (2000). Numerical model for thermal hydraulic analysis in cable-in-conduit-conductors. *KSME Int. J.,* v.14, n.9, p.985-96.

WANG, Y. Q.; GOOI, H. B. (1997). New ordering Methods for Sparse Matrix inversion via Diagonalization. *IEEE Transactions on Power Apparatus and Systems,* v.12, n.3, p.1298-1305.

WATKINS, D. S. (1991). *Fundamentals of Matrix Computations*. New York. John Wiley & Sons.

WEDEKING, J. E.; McKAY, D. B. (2000). Purification, crystallization and X-ray diffraction analysis of small ribozymes. *Method Enzymol,* v.317, p.149-68.

WOO, E. J.; HUA, P.; WEBSTER, J. G.; et al. (1994). Finite-Element method in Electrical-Impedance tomography. *Med. Biol. Eng. Comput.,* v.32, n.5, p.530-36.

ZHANG, S. M.; TINNEY, W. F. (1995). Partial Refactorization with unrestricted topology changes. *IEEE Transactions on Power Apparatus and Systems,* v.10, n.3, p.1631-68.

ZOLLENKOPF, K. (1971). Bi-Factorisation - Basic Computational Algorithm and Programming Techniques. In: *Large Sparse sets of Linear Equations.* REID, J.K., *Academic Press,* p. 75-79.

ZHONG, W. T.; SHAO, Z. J.; ZHANG, Y. Y.; et al. (2000). Applying analytical derivatinve and sparse matrix techniques to large-scale process optimization problems. *Chinese J. Chem. Eng.,* v.8, n.3, p.212-17.