SOLUÇÃO DE SISTEMAS LINEARES COM MATRIZES E VETORES ESPARSOS EM COMPUTADORES VETORIAIS

Alessandro Ruiz Basso **Carlos R. Minussi** **Antonio Padilha**

UNESP - Departamento de Engenharia Elétrica

Av. Brasil, 56 15385-000 Ilha Solteira - Brasil

Resumo: Este trabalho apresenta uma metodologia para a resolução de sistemas de equações lineares esparsas em computadores vetoriais. A nova metodologia é capaz de explorar a esparsidade da matriz e dos vetores do sistema. A implementação foi realizada em um computador CRAY Y-MP 2E/232 e são mostrados resultados considerando Sistemas de Energia Elétrica com 118, 320, 725 e 1729 barras. Na análise de desempenho foram consideradas também três metodologias publicadas anteriormente, sendo que a proposta neste trabalho apresenta superioridade.

Palavras-Chave: Sistemas de Energia Elétrica; Sistemas Lineares Esparsos; Vetores Esparsos; Processamento Vetorial.

Abstract: This paper presents a methodology for solving a set of linear sparse equations on vector computers. The new methodology is able to exploit the matrix and vector sparsities. The implementation was made on a CRAY Y-MP 2E/232 computer and the results were taken from electric power systems with 118, 320, 725 and 1729 buses. The proposed methodology was compared with three previous methods and the results show the superior performance of the new one.

Keywords: Electric Power Systems; Sparse Linear Systems; Sparse Vectors; Vector Processing.

1 - INTRODUÇÃO

Algoritmos associados com programas de simulação de Sistemas de Energia Elétrica, por exemplo, estabilidade transitória e fluxo de potência, geralmente requerem soluções repetidas de grandes conjuntos de equações lineares esparsas da forma $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$.

Resolver um sistema linear esparso empregando técnicas de decomposição ***LDU*** envolve três etapas (Monticelli, 1983): *ordenação* – que corresponde a obtenção da ordem de pivoteamento; *fatoração* – que consiste em obter os fatores triangulares através de eliminação de Gauss; e *solução direta* – que compreende a solução do problema. Simulações como, por exemplo, estabilidade transitória, usando simulação passo a passo, exigem que a ordenação seja realizada apenas uma vez e a fatoração poucas vezes, enquanto que a solução direta pode ser necessária centenas de vezes.

Obter um bom desempenho com processamento vetorial e/ou paralelo no estágio de solução deste conjunto de equações lineares esparsas tem sido um desafio para os pesquisadores em sistemas de energia elétrica. A necessidade de muitos cálculos em algumas aplicações e a tendência de redução de custos destes sistemas computacionais têm, ultimamente, estimulado muitas pesquisas neste campo. Muitos trabalhos foram publicados como tentativa de resolver este problema utilizando computadores paralelos (Morelato *et alii*, 1994; Lin e Van Ness, 1994; Padilha e Morelato, 1992), e outros utilizando computadores vetoriais (Granelli *et alii*, 1993; Huang e Lu, 1994; Aykanat *et alii*, 1995; Padilha e Basso, 1995; Vuong *et alii*, 1996).

Este trabalho discute resultados obtidos previamente (Padilha e Basso, 1995) e apresenta uma nova metodologia que explora o método de vetores esparsos (Tinney *et alii*, 1985). A motivação deste trabalho decorre do fato de que em muitas simulações de sistemas de energia elétrica o vetor $\boldsymbol{b}$ é esparso e o vetor $\boldsymbol{x}$ também será esparso caso deseje-se apenas um subconjunto da solução.

Todas as metodologias para o processamento vetorial, apresentadas anteriormente, não exibem habilidade para evitar cálculos desnecessários quando o vetor $\boldsymbol{b}$ e ou $\boldsymbol{x}$ são esparsos.

A nova metodologia procura aproveitar as principais técnicas propostas na literatura especializada e apresenta a capacidade de realizar apenas os cálculos necessários para a solução do sistema de equações lineares quando pelo menos um dos vetores é esparso.

2 - FORMULAÇÃO GERAL

As equações que descrevem as redes elétricas podem ser, geralmente, expressas da seguinte forma: $\boldsymbol{Ax} = \boldsymbol{b}$; em que $\boldsymbol{A}$ é uma matriz esparsa, $\boldsymbol{x}$ é um vetor desconhecido e $\boldsymbol{b}$ é o vetor

Submetido em 06/08/96
1a. revisão em 03/12/96 2a. revisão em 30/06/97
Artigo aceito sob recomendação do Ed. Cons. Prof. Dr. Liu Hsu

independente dado. A matriz A e os vetores x e b podem ser reais ou complexos.

Para resolver o sistema $Ax = b$, através de substituições, fatora-se a matriz A em $A = LDU$, sendo L, D e U matrizes triangular inferior, diagonal e triangular superior, respectivamente. A ordenação da matriz deve ser feita antes da fatoração. Os métodos de ordenação da matriz esparsa de redes de energia elétrica têm o objetivo de reduzir o aparecimento de elementos adicionais (*fill-ins*) e/ou gerar pequeno GCF (Grafo dos Caminhos de Fatoração) (Tinney *et alii*, 1985), ou, ainda, gerar menos elementos na matriz L^{-1}.

A etapa de solução pode ser realizada utilizando-se fatores diretos ou inversos. Quando utilizam-se fatores diretos tem-se:

(a) $Lz = b$ SF (Substituição *Forward*);

(b) $Dy = z$ SD (Substituição Diagonal);

(c) $Ux = y$ SB (Substituição *Backward*).

A solução do sistema também pode ser calculada utilizando-se matrizes de fatores inversos: $W = L^{-1}$ e $W^U = U^{-1}$. Logo tem-se:

(d) $z = W b$;

(e) $y = D^{-1} z$;

(f) $x = W^U y$.

A diferença com relação a solução anterior é que agora são usadas operações matriciais.

A principal idéia de usar fatores inversos (também conhecidos como método da matriz W) é tentar eliminar as relações de precedência das operações elementares (multiplicação e adição) existentes durante o processo de substituição. Esta abordagem pode ser vantajosa para uso em computadores paralelos e vetoriais. Porém, elevado número de elementos adicionais (*W-fills*), que são gerados na formação de W e W^U, pode tornar o uso dos fatores inversos impróprio. Pensando em evitar o aparecimento de elevado número de *W-fills* e manter algumas propriedades da matriz W, foram propostos alguns esquemas de particionamento (Alvarado *et alii*, 1990):

- Algoritmo PA1. Faz-se o particionamento agrupando-se nós que possuam a mesma profundidade no GCF. Nenhuma operação aritmética extra é necessária e nenhum novo elemento é gerado.

- Algoritmo PA2. Usa o princípio do PA1 mais uma função para avaliar nós de profundidades diferentes que poderiam ser agrupados sem gerar novos elementos. Nenhum elemento adicional é gerado, mas são necessárias algumas operações de multiplicação e adição para se obter as partições.

- Algoritmo PA3. Usa todos os critérios de PA2 mais uma função que permite a geração de uma porcentagem pré-estabelecida de novos elementos. Novos elementos surgem e muitos cálculos adicionais podem ser necessários dependendo da porcentagem permitida de novos elementos.

3 - PROCESSAMENTO VETORIAL

Entende-se como modo vetorial e escalar a resolução do problema utilizando processamento vetorial e escalar, respectivamente.

Considerando o esquema de particionamento PA1 e a ordenação MLMD (*Minimun Length Minimum Degree*) (Betancourt, 1988), a figura 1 mostra a matriz W correspondente ao sistema de 20 nós apresentado na referência Tinney *et alii* (1985). Neste caso, a diferença entre as matrizes W e L é apenas o sinal dos elementos não-nulos fora da diagonal. Portanto, nenhuma operação de ponto flutuante extra precisou ser realizada.

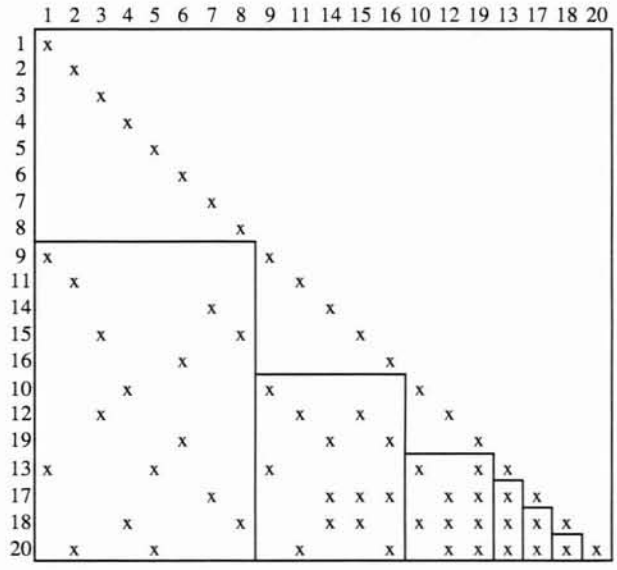

Figura 1. Particionamento da matriz **W** ou **L.**

Para que o processamento vetorial apresente um desempenho satisfatório, é necessário trabalhar com vetores longos, os quais podem ser obtidos através do armazenamento dos elementos das partições de W em pseudocolunas, como mostrado no apêndice 1.

Logo, a solução do problema, adaptada para o processamento vetorial, dá-se pelo particionamento da matriz W e armazenamento das partições através de pseudocolunas. Esta técnica apresenta um bom desempenho vetorial para as primeiras partições da matriz W, pois estas são partições grandes e, desta forma, obtêm-se vetores (pseudocolunas) longos. À medida que as partições da matriz W tornam-se pequenas, verifica-se que o desempenho do processamento vetorial diminui, tendo em vista que os vetores (pseudocolunas) tornam-se pequenos.

O apêndice 2 (tabela 8) mostra como as partições diminuem com o aumento da profundidade no GCF para vários sistemas.

Os esquemas de particionamento PA2 e PA3 tentam aumentar o tamanho de todas as partições. Contudo, para o processamento vetorial este aumento é de fundamental importância para as últimas partições. Pois as últimas partições contêm poucas colunas e isto significa que são geradas pseudocolunas com

poucos elementos. Uma opção mais interessante é a utilização do algoritmo PA1 para as primeiras partições e usar uma matriz cheia que reúne os nós das pequenas partições (Padilha e Morelato, 1992).

4 - METODOLOGIAS COM VETORES CHEIOS

São apresentadas, a seguir, três metodologias para a solução do sistema $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$, considerando-se que $\boldsymbol{A}$ é uma matriz esparsa e que $\boldsymbol{x}$ e $\boldsymbol{b}$ são vetores cheios. As metodologias 1, 2 e 3 foram propostas por Granelli *et alii* (1993), Huang e Lu (1994) e Padilha e Basso (1995), respectivamente. Todas foram implementadas contemplando-se as melhorias propostas por Aykanat *et alii* (1995).

Por simplicidade de representação, considera-se que a matriz $\boldsymbol{A}$ é simétrica e, portanto, $\boldsymbol{W}^{U} = \boldsymbol{W}^{T}$.

Metodologia 1. Consiste na construção das partições da matriz $\boldsymbol{W}$, usando o particionamento PA1. O sistema é, então, resolvido por partições com pseudocolunas na SF e SB:

$$\boldsymbol{x} = \boldsymbol{W}_1^{T} \cdot \boldsymbol{W}_2^{T} \cdots \boldsymbol{W}_n^{T} \cdot \boldsymbol{D}^{-1} \cdot \boldsymbol{W}_n \cdots \boldsymbol{W}_2 \cdot \boldsymbol{W}_1 \cdot \boldsymbol{b} \qquad (1)$$

sendo que: $\boldsymbol{W}_1$, $\boldsymbol{W}_2$, e $\boldsymbol{W}_n$ representam as operações correspondentes às partições 1,2 e n, respectivamente.

Metodologia 2. Consiste na utilização da Metodologia 1 para as primeiras partições (grandes), também armazenadas por pseudocolunas. As restantes são agrupadas em $\boldsymbol{W}_{LP}$. O vetor $\boldsymbol{x}$ é dado, então, por:

$$\boldsymbol{x} = \boldsymbol{W}_1^{T} \cdot \boldsymbol{W}_2^{T} \cdots \boldsymbol{W}_p^{T} \cdot \boldsymbol{W}_{LP}^{T} \cdot \boldsymbol{D}^{-1} \cdot \boldsymbol{W}_{LP} \cdot \boldsymbol{W}_p \cdots \boldsymbol{W}_2 \cdot \boldsymbol{W}_1 \cdot \boldsymbol{b} \qquad (2)$$

sendo:

$$\boldsymbol{W}_{LP} = \boldsymbol{W}_{p+1} \cdot \boldsymbol{W}_{p+2} \cdots \boldsymbol{W}_n \, .$$

Metodologia 3. Semelhante à Metodologia 2, porém, utiliza-se um tratamento diferente para as partições pequenas (últimas partições). Neste caso, as últimas partições são reunidas em uma matriz cheia, o que permite explorar diretamente o processamento vetorial. O vetor solução $\boldsymbol{x}$ vale:

$$\boldsymbol{x} = \boldsymbol{W}_1^{T} \cdot \boldsymbol{W}_2^{T} \cdots \boldsymbol{W}_p^{T} \cdot \boldsymbol{D}^{-1} \cdot \boldsymbol{A}_{LP}^{-1} \cdot \boldsymbol{W}_p \cdots \boldsymbol{W}_2 \cdot \boldsymbol{W}_1 \cdot \boldsymbol{b} \qquad (3)$$

sendo:

$$\boldsymbol{D} = \boldsymbol{D}_1 \cdots \boldsymbol{D}_p,$$

$$\boldsymbol{A}_{LP}^{-1} = \boldsymbol{W}_{p+1}^{T} \cdot \boldsymbol{W}_{p+2}^{T} \cdots \boldsymbol{W}_n^{T} \cdot \boldsymbol{D}_n^{-1} \cdots \boldsymbol{D}_{p+1}^{-1} \cdot \boldsymbol{W}_n \cdots \boldsymbol{W}_{p+1} \, .$$

5 - METODOLOGIA COM VETORES ESPARSOS

O objetivo do método de vetores esparsos é obter uma economia computacional em refatorações de matrizes e nas soluções diretas, calculando somente as operações necessárias nas substituições *forward/backward*. As substituições assim definidas são chamadas *fast-forward* e *fast-backward*.

A substituição *fast-forward* calcula somente um subconjunto de colunas da matriz $\boldsymbol{L}$ / $\boldsymbol{W}$ (quando o vetor $\boldsymbol{b}$ é esparso), e *fast-backward* necessita calcular somente um subconjunto de linhas da matriz $\boldsymbol{U}$ / $\boldsymbol{W}^{U}$ (quando um subconjunto de $\boldsymbol{x}$ é requerido) para resolver um sistema na forma $\boldsymbol{LDU}\,\boldsymbol{x} = \boldsymbol{b}$ / $\boldsymbol{x} = \boldsymbol{W}^{U}\boldsymbol{D}^{-1}\boldsymbol{W}\,\boldsymbol{b}$. As colunas e linhas necessárias podem ser vistas no GCF. A figura 2 mostra o GCF com relação a matriz $\boldsymbol{L}$ da figura 1, e ilustra os caminhos para as substituições *fast-forward* ou *fast-backward*.

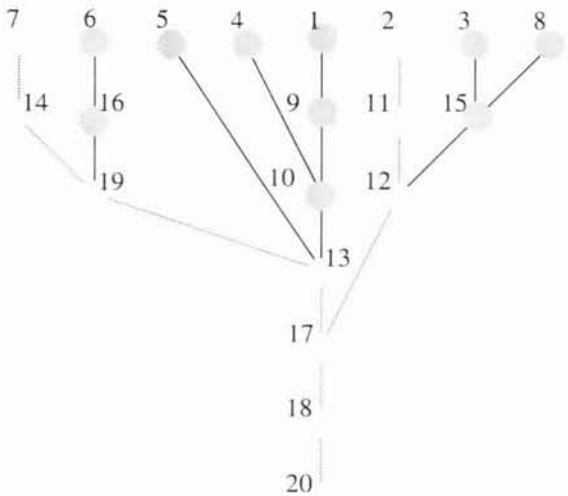

Figura 2. Grafo dos Caminhos de Fatoração (GCF) para a matriz $\boldsymbol{L/W}$. As linhas pontilhadas mostram os caminhos necessários para a *fast-foward*, considerando-se que o vetor $\boldsymbol{b}$ possui elementos não-nulos somente nas posições 2 e 7. Os mesmos caminhos são necessários para a *fast-backward* se o subconjunto desejado de $\boldsymbol{x}$ compreende apenas as posições 2 e 7.

As substituições *fast-forward* e *fast-backward* podem ser empregadas em muitas simulações. Por exemplo, em estudos de estabilidade transitória, a rede elétrica pode ser representada como $\boldsymbol{I} = \boldsymbol{Y}\,\boldsymbol{V}$, em que $\boldsymbol{I}$ é o vetor de injeção de correntes, $\boldsymbol{Y}$ é a matriz de admitância nodal e $\boldsymbol{V}$ é o vetor de tensões desconhecidas. Quando as cargas são modeladas como impedâncias constantes, o vetor $\boldsymbol{I}$ é muito esparso porque existem injeções de corrente somente nas barras de geração. O vetor $\boldsymbol{V}$ pode ter a mesma esparsidade de $\boldsymbol{I}$ quando o usuário desejar conhecer somente as tensões nas barras de geração. O vetor $\boldsymbol{V}$ pode ser cheio se o usuário necessitar das tensões em todas as barras.

A nova metodologia, explorando vetores esparsos, usa somente os elementos não-nulos das colunas/linhas necessárias de W/W^U para formar pseudocolunas. As linhas pontilhadas, na figura 2, mostram as colunas as quais devem ser usadas para formar as pseudocolunas, para a substituição *fast-forward,* considerando que somente as posições 2 e 7 do vetor b possuem elementos não-nulos.

A metodologia considera que os últimos nós do GCF podem ser reunidos para formar uma matriz inversa cheia (chamada última partição – A_{LP}^{-1}) de tal modo que as substituições *forward,* diagonal e *backward,* correspondentes desta última partição, são realizadas simultaneamente (Padilha e Morelato, 1992). O restante da matriz é armazenada em pseudocolunas (matriz-W).

A parte da matriz armazenada em pseudocolunas não requer nenhuma operação de ponto flutuante, quando o algoritmo de particionamento PA1 é utilizado. Ressalta-se que a matriz inversa cheia A_{LP}^{-1} requer cálculos para a sua formação. A figura 3 mostra as pseudocolunas e partições considerando os caminhos necessários da figura 2. As linhas pontilhadas mostram os cálculos necessários para resolver *fast-forward*, diagonal, última partição (A_{LP}^{-1}) e *fast-backward.* Nota-se que os elementos das pseudocolunas devem ser armazenados compactamente para obtenção de eficiência durante a vetorização.

O processo de solução da *fast-forward* (*fast-backward*) pode ser realizado em tantos *do-loops* quantos forem os números de pseudocolunas. A seguir é ilustrada a programação do algoritmo da *fast-forward*:

```
do i = 1, número de pseudocolunas
   do j = {elementos da pseudocoluna i}
      b(ilj) = b(ilj) + W(j)*b(icj)
   end do
end do
```

sendo:

ilj = índice da linha do elemento j;
icj = índice da coluna do elemento j.

A substituição *fast-backward* é realizada de maneira semelhante, porém, iniciando o processo pela última pseudocoluna e finalizando na primeira. Deve-se destacar que as pseudocolunas utilizadas na *fast-forward* são diferentes das usadas na *fast-backward*, como mostrado na figura 3. A solução da última partição (A_{LP}^{-1}) deve ser realizada antes da *fast-backward* e corresponde a multiplicação de um matriz cheia por um vetor. O número de colunas agrupado em A_{LP}^{-1} é determinado de maneira heurística como descrito por Padilha e Morelato, 1992.

6 - ANÁLISE DOS RESULTADOS

Foram utilizados os mesmos níveis de otimização durante a compilação das metodologias (compilador Fortran 77). As diferenças entre as metodologias no modo escalar e vetorial são dadas pelo uso de diretivas de compilação dentro do próprio programa, as quais desabilitam a vetorização no modo escalar, ou informam ao compilador que não existe dependência de dados no laço subseqüente, para o caso vetorial.

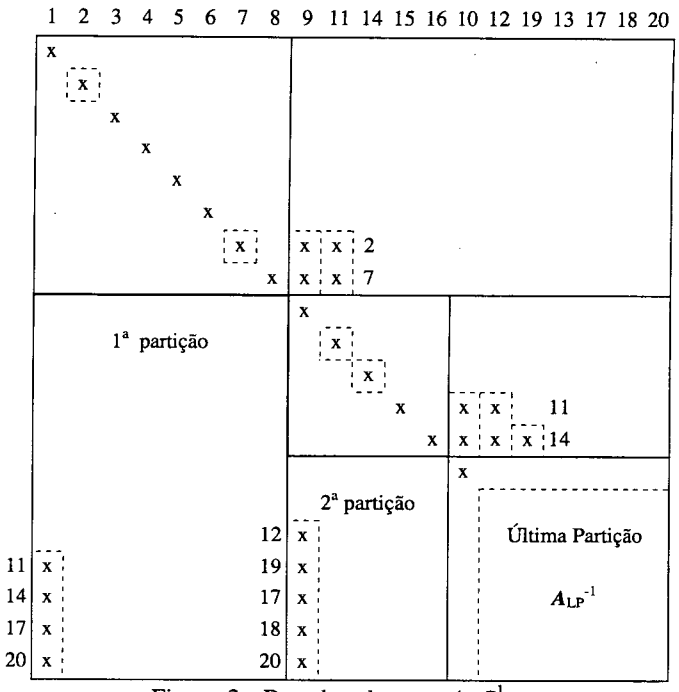

Figura 3 - Pseudocolunas e A_{LP}^{-1}.

Com o objetivo de analisar o desempenho da metodologia proposta e das anteriores, foram considerados dois tipos de problemas:

1) $I = Y V$

em que Y, I e V são, respectivamente, matriz de admitância esparsa complexa, vetor de correntes e vetor de tensões, que é desconhecido;

2) $P = B' \theta$

sendo que B', P e θ são, respectivamente, matriz esparsa real, vetor de potência e vetor dos ângulos das tensões, que é desconhecido.

O desempenho da nova metodologia (denominada metodologia 4) depende da esparsidade dos vetores e também da esparsidade da matriz. Neste trabalho, foi considerado um caso especial onde os vetores I e P contêm elementos diferentes de zero somente nas posições associadas às barras geradoras. Foi também considerada a mesma esparsidade para os vetores V e θ. Este é um caso que acontece em algumas simulações de estabilidade transitória, como descrito na seção 5. Os dados dos sistemas estudados (118, 320, 725 e 1729 barras) encontram-se no apêndice 2.

As quatro metodologias foram implementadas no computador CRAY Y–MP 2E/232 da Universidade Federal do Rio Grande do Sul. A tabela 1 mostra os tempos de CPU gastos na etapa de solução de $I = YV$ usando-se as ordenações MDML (*Minimum Degree Minimum Length*) (Betancourt, 1988) e MLMD. Na tabela 1 *np* denota o número de partições em que os elementos foram armazenados em pseudocolunas. As metodologias 1, 2 e 3 não apresentam habilidade de explorar a esparsidade dos vetores, enquanto que a metodologia 4 explora a esparsidade dos vetores. Caso os vetores não forem esparsos, o desempenho da metodologia 4 será idêntico ao da metodologia 3.

Tabela 1. Tempo de CPU (seg x 10^{-4}) para o sistema de 118 nós.

np	MDML			MLMD		
	Metod. 2	Metod. 3	Metod. 4	Metod. 2	Metod. 3	Metod. 4
0	3.62	3.86	1.72	3.62	3.86	1.85
1	2.02	1.66	1.16	2.09	1.53	1.10
2	1.45	0.85	0.70	1.49	0.83	0.64
3	1.23	0.77	0.61	1.27	0.81	0.62
4	1.14	0.75	0.63	1.21	0.89	0.70
5	1.09	0.81	0.65	1.30	1.04	0.84
6	1.08	0.88	0.70	1.37	1.18	0.98
7	1.13	0.95	0.78	1.45	1.31	1.08
8	1.13	1.01	0.84	1.51	1.41	1.17
9	1.17	1.12	0.92	1.58	1.50	1.25
10	1.23	1.19	0.98	1.63	1.60	1.31
11	1.22	1.22	1.01	1.65	1.62	1.36
12	1.23	1.24	1.04	1.67	1.66	1.40
13	-	-	-	1.67	1.69	1.42

Pela análise do tempo de CPU, pode-se verificar que o algoritmo MDML apresenta o melhor resultado e a metodologia 4 tem um desempenho superior.

Para se analisar eficiência das metodologias foi utilizado um ganho definido como sendo o tempo computacional do melhor algoritmo escalar dividido pelo tempo computacional de cada uma das referidas metodologias. Isto permite comparar as vantagens do uso de computadores vetoriais em relação a computadores que usam processamento escalar.

Primeiramente foi considerado o algoritmo da metodologia 1 no modo escalar. A implementação da metodologia 1, em computadores com processamento escalar, tem-se mostrado a mais eficiente, superando até uma tradicional rotina (Zollenkopf, 1971). Isto permitirá uma comparação entre soluções com e sem processamento vetorial, e ainda, entre as metodologias que não exploram vetores esparsos e a proposta que usa vetores esparsos. Assim, tem-se:

$$g1 = \frac{\text{tempo de CPU da melhor metodologia usando vetores cheios no modo escalar}}{\text{tempo de CPU de cada metodologia no modo vetorial}}$$

As tabelas 2 e 3 mostram os ganhos das quatro metodologias para problemas que usam operações aritméticas reais ($\boldsymbol{P = B'\,\theta}$) e para problemas que usam operações complexas ($\boldsymbol{I = Y\,V}$). A ordenação utilizada é a MDML. ME indica a metodologia aplicada e *np* tem o mesmo significado como definido na tabela 1.

Tabela 2. Ganho g1 na solução de $\boldsymbol{P = B'\,\theta}$.

ME	Sistemas							
	118		320		725		1729	
	g1	np	g1	np	g1	np	g1	np
1	2.61	12	2.30	29	2.96	32	3.69	42
2	3.04	5	3.50	8	3.95	11	5.02	15
3	5.53	2	6.55	4	6.23	6	6.96	10
4	7.19	3	10.89	4	11.08	6	14.73	10

Tabela 3. Ganho g1 na solução de $\boldsymbol{I = Y\,V}$.

ME	Sistemas							
	118		320		725		1729	
	g1	np	g1	np	g1	np	g1	np
1	2.54	12	2.14	29	2.80	32	3.37	42
2	2.92	6	3.01	11	3.44	12	4.16	15
3	4.07	4	4.48	6	4.39	10	5.06	15
4	5.13	3	7.02	7	7.80	9	10.47	13

Observa-se que o melhor desempenho da metodologia proposta é um ganho de aproximadamente 14 e 10 para resolver $\boldsymbol{P = B'\,\theta}$ e $\boldsymbol{I = Y\,V}$, respectivamente, usando o sistema de 1729 barras.

Quando um problema apresenta esparsidade de vetores, a melhor solução no modo escalar é uma que explora esta esparsidade. Assim, agora define-se o ganho novamente como sendo:

$$g2 = \frac{\text{tempo de CPU da melhor metodologia usando vetores esparsos no modo escalar}}{\text{tempo de CPU de cada metodologia no modo vetorial}}$$

A melhor solução no modo escalar explorando a esparsidade dos vetores corresponde à metodologia 1 utilizando vetores esparsos. As tabelas 4 e 5 mostram o ganho g2 para problemas com matrizes reais e complexas. Nota-se que o ganho da metodologia proposta diminui consideravelmente em relação as tabelas 2 e 3. Contudo, é evidente a necessidade do uso da metodologia que explora a esparsidade dos vetores (metodologia 4) pelo que se observa com relação as metodologias 1, 2 e 3.

Tabela 4. Ganho g2 na solução de $\boldsymbol{P = B'\,\theta}$.

ME	Sistemas							
	118		320		725		1729	
	g2	np	g2	np	g2	np	g2	np
1	1.70	12	1.27	29	1.43	32	1.42	42
2	1.98	5	1.94	8	1.88	11	1.94	15
3	3.60	2	3.65	4	3.01	6	2.69	10
4	4.68	3	6.03	4	5.39	6	5.70	10

Tabela 5. Ganho g2 na solução de $I = Y V$.

ME	Sistemas							
	118		320		725		1729	
	g2	np	g2	np	g2	np	g2	np
1	1.58	12	1.11	29	1.24	32	1.21	42
2	1.82	6	1.56	11	1.51	12	1.49	15
3	2.54	4	2.32	6	1.93	10	1.81	15
4	3.20	3	3.64	7	3.46	9	3.76	13

Deve ser destacado que nos cálculos dos ganhos g1 e g2 não foram considerados os tempos empregados na organização dos elementos em pseudocolunas e no cálculo da matriz inversa A_{LP}^{-1}.

7 - CONCLUSÃO

Foi apresentada uma nova metodologia para solução de equações lineares esparsas para uso em computadores vetoriais. A metodologia procura aproveitar as melhores técnicas propostas anteriormente, para solução com processamento vetorial, e explora o método de vetores esparsos. Soluções que exploram vetores esparsos podem ser empregadas em muitas simulações de Sistemas de Energia Elétrica.

O método proposto foi implementado em um computador CRAY Y–MP 2E/232, cujos resultados mostram a superioridade da nova proposta. Um ganho máximo de aproximadamente 10 (14) foi obtido na realização de uma solução com números complexos (reais). Já a melhor metodologia anterior apresenta ganhos de aproximadamente 5 (7). A maneira de calcular estes ganhos (g1) permitiu verificar o desempenho da metodologia proposta em relação as anteriores. Permite, também, comparar como se comportam as metodologias implementadas neste trabalho em relação a implementações de outras publicações, uma vez que estas últimas não usam a esparsidade dos vetores. O ganho apresentado pela metodologia em relação a melhor solução escalar (ganho g2) pode ser de aproximadamente 3 (6) para realizar uma solução com números complexos (reais).

Os ganhos mostraram também que o tempo de processamento de alguns problemas, que requerem repetidas soluções usando a mesma matriz, pode ser diminuido várias vezes dependendo exatamente do número de vezes em que é necessário realizar essa solução.

A metodologia apresentada usou o método da matriz ***W***, mas ela pode ser utilizada também com a metodologia DBSA (*Dependency-Based Substitution Algorithm*), proposta por Vuong *et alii* (1996). Esta referência mostra que o ganho da metodologia DBSA é um pouco superior à matriz ***W*** em computadores vetoriais.

Futuros trabalhos devem considerar a solução do problema de cálculo de estabilidade transitória e, então, poderão ser analisados detalhes como: entrada e saída de dados, ordenação, fatoração, arranjo da matriz em pseudocolunas, formação da matriz inversa A_{LP}^{-1}, etc.

REFERÊNCIAS BIBLIOGRÁFICAS

Alvarado, F. L.; Yu, D.C. and Betancourt, R. (1990). Partitioned sparse A^{-1} methods. *IEEE Transactions on Power Systems*, Vol.5, No. 2, pp. 452-459.

Aykanat, C.; Ozgu, O. and Guven, N. (1995). Algorithms for Efficient Vectorization of Repeated Sparse Power System Network Computations. *IEEE Transactions on Power Systems, Vol. 10,* No. 1, pp. 448–456.

Betancourt R (1988). An Efficient Heuristic Ordering Algorithm For Partial Matrix Refactorization. *IEEE Transactions on Power Systems*, Vol. 3, No. 3, pp. 1181–1187.

Granelli G. P., Pasini G. L. and Marannino P (1993). A W-matrix Based Fast Decoupled Load Flow for Contingency Studies on Vector Computers. *IEEE Transactions on Power Systems*, Vol. 8, No. 3; pp. 946–956.

Huang, H. S. and Lu, C. N (1994). Efficient Storage Scheme and Algorithms for W-matrix Vector Multiplication on Vector Computers. *IEEE Transactions on Power Systems*, Vol.9, No. 2; pp. 1083–1094.

Lin, S.L. and Van Ness J.E (1994). Parallel Solution of Sparse Algebraic Equations. *IEEE Transactions on Power Systems*, Vol.9, No. 2. pp. 743–799.

Monticelli, A. (1983). *Fluxo de Carga em Redes de Energia Elétrica.* Edgar Blucher, Rio de Janeiro – RJ.

Morelato, A; Amaro,M. and Kokai,Y (1994). Combining Direct and Inverse Factors for Solving Sparse Network Equations in Parallel. *IEEE Transactions on Power Systems*, Vol. 9, No. 4, pp. 1942–1948.

Padilha, A. and Morelato, A (1992). A W-matrix Methodology for Solving Sparse Network Equations on Multiprocessor Computer. *IEEE Transactions on Power Systems*, Vol.7, No. 3, pp. 1023–1030.

Padilha, A. and Basso, A. R. (1995). A Hibrid Method for the Solution of a Sparse Power System Matrices on Vector Computers. *IEEE - 38th Midwest Symposium on Circuits and Systems*, Vol.2, 925–928.

Tinney, W.F., Brandwajn, V. and Chan, S. M (1985). Sparse Vector Methods. *IEEE Transactions on Power Apparatus and Systems*, Vol.104, No. 2; 295–301.

Vuong, G.T., Chahine, R., Granelli, G.P. and Montagna, M. (1996). M. Dependency-Based Algorithms For Vector Processing of Sparse Matrix Forward/Backward Substitutions. *IEEE Transactions on Power Systems*, Vol. 11, No. 1, 198–205.

Zollenkopf, K (1971). Bi-Factorisation Basic Computational Algorithm and Programming Techniques. *J. K. Reid (ed.) Large Sparse Sets of Linear Equations*, Academic Press, pp. 75–96.

AGRADECIMENTOS

Os Autores agradecem ao Centro Nacional de Supercomputação da Universidade Federal do Rio Grande do Sul que possibilitou o uso do computador CRAY Y–MP 2E/232 e à FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo) pelo suporte financeiro – Proc. nº 95/9074-7.

APÊNDICE 1

Armazenamento da Matriz *W*

Considerando-se a matriz ***W*** particionada, conforme mostra-se na figura 1, pode-se armazenar os elementos de cada partição de tal forma que favoreça o processamento vetorial. As figuras e 5 mostram como são formadas as pseudocolunas e as pseudolinhas para a 1ª partição da matriz ***W*** da figura 1.

As operações, no processo de substituição, com os elementos de uma pseudocoluna na SF ou SB são independentes e podem ser realizadas todas de uma só vez, por exemplo, usando processamento vetorial. Já as operações com os elementos das pseudolinhas apresentam problemas e não podem ser realizadas todas de uma vez, como monstram as tarefas #3 e #8 da tabela 6.

Tabela 6. Operações na SF com os elementos da 1ª pseudolinha.

Tarefa	Operação	Tarefa	Operação
#1	$b_9 = b_9 + W_{9,1} * b_1$	#5	$b_{13} = b_{13} + W_{13,5} * b_5$
#2	$b_{11} = b_{11} + W_{11,2} * b_2$	#6	$b_{16} = b_{16} + W_{16,6} * b_6$
#3	$b_{15} = b_{15} + W_{15,3} * b_3$	#7	$b_{14} = b_{14} + W_{14,7} * b_7$
#4	$b_{10} = b_{10} + W_{10,4} * b_4$	#8	$b_{15} = b_{15} + W_{15,8} * b_8$

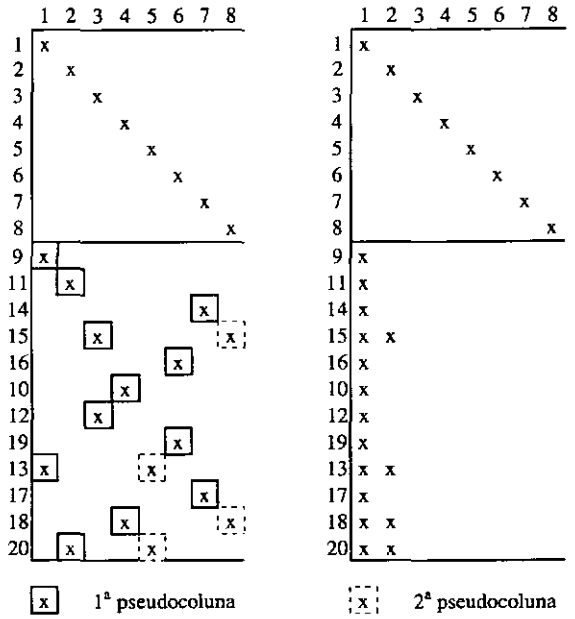

Figura 4. Pseudocolunas.

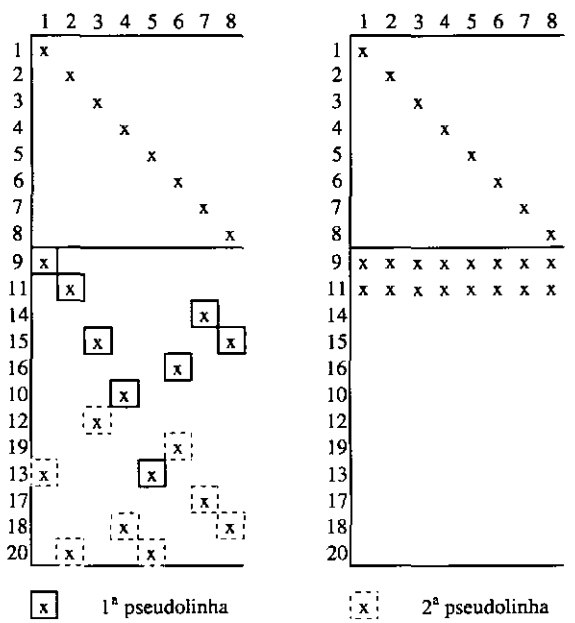

Figura 5. Pseudolinhas.

APÊNDICE 2

Sistemas Elétricos Utilizados

Neste trabalho usou-se quatro diferentes Sistemas de Energia Elétrica: sistema IEEE-118 barras, e três configurações do sistema interligado sul-sudeste brasileiro, conforme ilustrado na tabela 7.

Para se ter uma idéia de como as partições diminuem com o aumento da profundidade no GCF, apresenta-se na tabela 8 o número de nós em cada partição, para vários sistemas, utilizando a ordenação MDML, desde os nós folhas (primeira partição) até o nó raiz (última partição). Nesta tabela encontram-se, também, o número de *fill-ins* e o número total de elementos por partição (matriz ***L***, incluindo a diagonal).

Tabela 7. Características dos Sistemas Utilizados.

	Sistemas			
	IEEE-118	BR-320	BR-725	BR-1729
Barras	118	320	725	1729
Linhas	176	470	935	2154
Geradores	53	44	92	156

Tabela 8. Número de elementos e *fill-ins* por partição.

Part.	Sistemas											
	118			320			725			1729		
	nº nós	nº "fill-ins"	nº elem.	nº nós	nº "fill-ins"	nº elem.	nº nós	nº "fill-ins"	nº elem.	nº nós	nº "fill-ins"	nº elem.
1	52	0	154	153	0	423	361	0	886	875	0	2121
2	27	21	86	57	50	198	148	76	415	367	181	997
3	11	9	38	32	52	132	72	62	217	167	146	485
4	8	13	31	16	35	72	34	33	110	97	98	309
5	5	9	17	11	23	52	26	44	98	60	102	221
6	3	6	12	9	31	45	15	39	67	37	75	145
7	3	8	14	5	17	25	10	33	55	25	60	106
8	2	6	8	3	12	15	8	31	44	17	52	85
9	2	5	9	3	12	18	7	32	44	12	38	58
10	2	6	8	2	7	9	6	30	39	8	25	40
11	1	2	3	2	7	10	4	13	24	6	23	33
12	1	1	2	2	10	13	3	10	16	5	22	29
13	1	0	1	2	11	14	2	9	11	5	29	36
14				2	12	14	2	9	12	3	12	16
15				2	14	17	2	7	12	3	10	17
16				2	14	18	2	9	12	2	8	12
17				2	16	19	2	12	14	2	11	14
18				2	18	20	2	12	15	2	14	18
19				2	18	21	2	9	16	2	13	16
20				1	10	11	2	17	21	2	12	14
21				1	9	10	2	22	25	2	13	15
22				1	8	9	2	20	23	2	13	15
23				1	7	8	1	10	11	2	16	18
24				1	5	7	1	9	10	2	14	16
25				1	3	6	1	7	9	2	16	20
26				1	4	5	1	6	8	2	29	31
27				1	3	4	1	6	7	2	27	29
28				1	2	3	1	5	6	2	27	31
29				1	1	2	1	4	5	2	25	30
30				1	0	1	1	3	4	1	13	14
31							1	2	3	1	12	13
32							1	1	2	1	10	12
33							1	0	1	1	9	11
34										1	8	10
35										1	8	9
36										1	7	8
37										1	6	7
38										1	5	6
39										1	4	5
40										1	3	4
41										1	2	3
42										1	1	2
43										1	0	1
Total	118	86	383	320	411	1201	725	582	2242	1729	1199	5082