METODOLOGIA PARA SOLUÇÃO DE EQUAÇÕES ALGÉBRICAS ESPARSAS EM COMPUTADORES VETORIAIS

Antonio Padilha
DEE-FEIS-UNESP
Caixa postal 31
15378-000 - Ilha Solteira - SP

RESUMO

Neste trabalho trata-se da solução do problema clássico de um sistema de equações lineares esparsas $Ax = b$. Na análise de sistemas de energia elétrica, este problema surge no cálculo de fluxo de potência, estabilidade transitória, transitórios eletromagnéticos e outros de sistemas de energia elétrica. A utilização de computadores vetoriais é considerada, sendo que analisá-se a utilização da metodologia da matriz de fatores inversos W com particionamento. Recentes artigos propuseram e testaram esta técnica na etapa solução do problema. Analisa-se aqui estes métodos e propõe-se uma nova metodologia para resolver equações algébricas esparsas utilizando processamento vetorial.
Palavras Chaves: Esparsidade, Processamento Vetorial, Fluxo de Potência, Estabilidade Transitória.

METHODOLOGY FOR SOLUTION OF SPARSE ALGEBRAIC EQUATIONS ON VECTOR COMPUTER

ABSTRACT

This paper deals with the classic problem of sparse matrices $Ax = b$. This problem is present in power systems studies such is power flow, transient stability, electromagnetic transient and others. The uses of vector computer have been discussed for utilization of factor inverse matrix methodology with partitioning. Recent papers have presented and discussed the problem solution using this methodology. These methods are analysed and are described, showing one new methodology for solution of sparse algebraic equations.
Keywords: Sparsity, Vector Computing, Power Flow, Transient Stability.

1. INTRODUÇÃO

O desenvolvimento dos computadores de alto desempenho tem estimulado as empresas de energia elétrica a implementarem sistemas computacionais, que possam resolver muito mais rapidamente problemas de análise de sistemas de energia elétrica. Muitos destes problemas, como por exemplo cálculo de estabilidade transitória, requerem muito tempo de simulação para estudo de apenas um caso, nos computadores tradicionais. Nestes problemas aparece a etapa de solução de um conjunto de equações algébricas lineares e esparsas, que devido ao alto grau de esparsidade é resolvido com técnicas de programação de matrizes esparsas. Resolver de forma mais eficientes problemas como estes e' motivo para estudos de novos algoritmos, e para utilização de computadores mais poderesos.

Uma configuração possível de centro de processamento de alto desempenho é a mostrada em [1], que possui um supercomputador (com capacidade de processamento vetorial) e uma rede de estações que utilizam o mesmo sistema operacional. Desenvolvimento como este auxiliam em muito nos problemas de análise de sistemas de energia elétrica. Por outro lado o aparecimento de novos algoritmos para resolver sistemas esparsos de equações algébricas, como o método de vetores esparsos [2], o método de matrizes W [3] e métodos de particionamento de matrizes [4], [5], [6], têm estimulado o interesse em implementar programas cada vez mais eficientes para computadores multiprocessadores [5] [6] e computadores vetoriais [7] [8].

Depois de uma revisão na formulação geral, nos conceitos básicos de matrizes W, e uma análise das mais recentes metodologias para resolver sistemas esparsos de equações algébricas em computadores vetoriais, este trabalho apresenta um algoritmo para solução de equações de redes de energia elétrica.

2. FORMULAÇÃO GERAL

Muitos problemas de sistemas de energia elétrica necessitam resolver um conjunto de equações algébricas que é posto na seguinte forma:

$$Ax = b \qquad (1)$$

onde A é uma matriz esparsa e simétrica, o vetor x é a solução desejada e b é um vetor conhecido.

A matriz A é fatorada como $A = LDL^t$, onde L é uma matriz triangular inferior com diagonal unitária e D é uma matriz diagonal. A solução do sistema é obtida em três passos: substituição "forward" ($Lz = b$), divisão pela diagonal ($Dy = z$) e substituição "backward" ($L^t x = z$). Nestas substituições são feitas operações elementares (multiplicação-adição) em uma certa ordem (relações de precedência). O cálculo da matriz L é precedido de uma ordenação das linhas/colunas da matriz A, geralmente, com objetivo de reduzir o aparecimento de novos elementos diferentes de zero ("fill-ins").

Pode-se também calcular a solução de (1) usando-se matrizes de fatores inversos $W = L^{-1}$ [3]. Assim a solução fica: $z = Wb$, $y = D^{-1}z$ e $x = W^t y$. A diferença em relação a solução anterior é que agora são usadas operações matriciais.

Às vezes pode ser conveniente agrupar algumas operações vizinhas como, por exemplo, as operações correspondentes a uma coluna ou linha da matriz A. Neste caso pode-se expressar L como: $L = L_1 L_2 \cdots L_n$, onde L_i é uma matriz identidade exceto na i-ésima coluna que contém a coluna i de L, e que corresponde ao agrupamento de todas as transformações elementares necessárias para processar a coluna i de A. Consequentemente pode-se expressar W como: $W = L^{-1} = W_n \cdots W_2 W_1$, onde $W_i = L_i^{-1}$ e portanto tem os mesmos elementos fora da diagonal que L_i, apenas com os sinais trocados. Deve ser notado que L_i e W_i correspondem ao processamento de um nó da rede elétrica.

Particionamento da Matriz W

Particionar uma matriz W (ou a matriz L se for o caso) significa agrupar W_i ou W_i^t adjacentes de tal modo a realizar as correspondentes operações em bloco. Cada bloco de colunas (ou linhas) é chamado de uma partição da matriz. Assim:

$$x = W_{1p}^t W_{2p}^t \cdots W_{np}^t D^{-1} W_{np} \cdots W_{2p} W_{1p} b \qquad (2)$$

onde W_{1p} W_{2p} e W_{np} representam as operações correspondentes à partição 1, partição 2 e partição n, respectivamente.

A solução pode então ser obtida em $2np$ etapas seriais (não considerando a diagonal) partindo-se da direita para a esquerda, com paralelismo dentro de cada partição. Alguns trabalhos ([4], [7] e [5]) têm estuuuu metodolgia na etapa de solução para uso de computadores de alto desempenho.

3. ESQUEMAS DE PARTICIONAMENTO

As relações de precedência entre as linhas/colunas das matrizes L ou W podem ser postas em um grafo dos caminhos de fatoração da matriz A [2]. A fim de verificar estas relações considere-se a rede exemplo da Figura 1, que gera o grafo da Figura 2, depois de uma ordenação ML-MD [9]. Esta ordenação leva em consideração a profundidade ("length" ou "depth") de cada nó no grafo de fatoração como primeiro critério de escolha, quando dois nós empatam o desempate é feito considerando-se o número de nós vizinhos. Diferentes ordenações podem produzir diferentes grafos de fatoração e também alterar o número de elementos "fill-ins" em L.

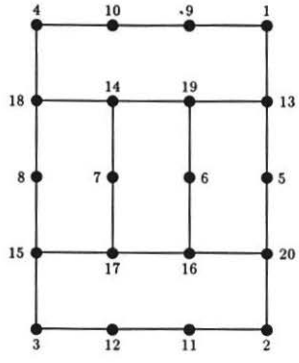


Figura 1 - Rede 20 nós

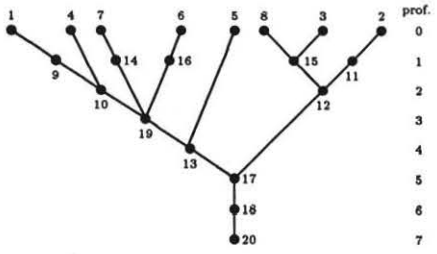


Figura 2 - Árvore dos caminhos de fatoração

A principal idéia da abordagem da matriz W é tentar eliminar as relações de precedência das operações elementares durante o processo de substituição. Nota-se que na "forward" usando W, todas as operações de multiplicação podem ser feitas independentemente uma das outras, mas infelizmente as operações de adição não podem ser feitas simultaneamente. Comparando esta solução com as soluções que usam as operações elementares como tarefa, percebe-se que as operações de multiplicação estão relacionadas com as relações de precedência e as operações de adição com as relações de *não-simultaneidade*. Desta forma usar a inversa da matriz L significa *quebrar* todas as relações de precedência, permanecendo porém as relações de *não-simultaneidade*. O custo da *quebra* das relações de precedência é um trabalho extra, pois a matriz W não é tão esparsa quanto a matriz L. Estes novos elementos não-zeros são "fill-ins" adicionais que, sendo numerosos, podem contrabalançar o ganho obtido com a realização das multiplicações em paralelo [12].

Uma das preocupações ao particionar a matriz W é não gerar muitos "fill-ins" adicionais, o que pode comprometer o desempenho da solução. Neste sentido, a referência [4] propõe três esquemas de particionamento da matriz W que permitem controlar o aparecimento de "fill-ins" adicionais. Estes esquemas são baseados na profundidade dos nós no grafo de fatoração. Se os nós de mesma profundidade são ordenados em sequência, então ao agrupá-los numa partição p, não surgirão "fill-ins" adicionais nesta partição.

Os algoritmos de particionamento descritos em [4] têm como características:

- Algoritmo PA1 - nenhum "fill-in" adicional é permitido e nenhum cálculo extra é feito para obtenção das partições W. Faz-se o particionamento agrupando nós de uma mesma profundidade.
- Algoritmo PA2 - nenhum "fill-in" adicional é permitido e são necessárias algumas multiplicações entre W_i vizinhas para obter as partições W. Este algoritmo usa o mesmo critério de PA1, mais uma função para verificar se as posições onde surgiriam potenciais "fill-ins" já estão preenchidas.
- Algoritmo PA3 - uma porcentagem de "fill-ins" adicionais é permitida por partição, sendo necessário multiplicações entre W_i para obter as partições W. Usa todos os critérios de PA2, e conta o número de "fill-ins" adicionais comparando-o com uma porcentagem pré-determinada.

Os dois primeiros algoritmos têm como vantagem o fato que não criam trabalho extra, mas geram muitas partições, significando muitos passos sequenciais e a maioria deles com pouco trabalho a ser realizado. O algoritmo PA3 pode gerar poucas partições se uma grande porcentagem de trabalho extra for permitida; mesmo assim algumas partições podem ser muito pequenas.

A referência [7] também apresenta um esquema de particionamento, buscando não gerar nenhum "fill-in" adicional. Este esquema é semelhante ao PA2, mas com um critério bastante simples para verificar se ao agrupar nós de profundidades diferentes não surgirão novos elementos. O critério consiste em verificar o grau de cada nó depois de uma ordenação MD (o clássico esquema 2 de Tinney [10]).

Outra proposta apresentada em [5] também usa a idéia de particionar W baseando-se na árvore dos caminhos de fatoração, mas o faz de uma maneira completamente diferente e com outros objetivos. Este esquema consiste em particionar a matriz levando em consideração a profundidade dos nós, e busca *quebrar* relações de precedecência das pequenas partições (últimas), da seguinte forma:

- cada partição é associada a uma profundidade da árvore de fatoração, ou seja, a partição p corresponde a todos os nós de profundidade p;
- o particionamento prossegue até ser atingida uma profundidade escolhida (chamada *profundidade de corte*) e então todos os nós remanescentes são reunidos em uma única partição (chamada de *última partição*).

A escolha da profundidade de corte é heurística e não é crítica, pois o número de folhas por profundidade decresce rapidamente.

A Figura 3 mostra a estrutura da matriz W para a rede 20 nós ordenada pelo esquema ML-MD, com 3 partições obtidas através deste particionamento. Outras ordenações como a MD [10], ou MDMNP [11] geram estruturas diferentes.

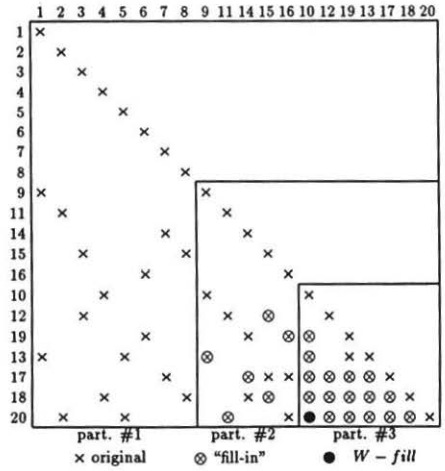


Figura 3 - Matriz W particionada

Como esperado nas duas primeiras partições não surgiu nenhum novo elemento não-zero, enquanto que na última apareceu apenas um "fill-in" adicional. Deve ser notado que todas as informações necessárias para o particionamento podem ser obtidas da árvore de fatoração da rede, e que somente a última partição requer cálculos extras para sua obtenção, como detalhado em [13].

De acordo com este esquema de particionamento pode-se agora escrever a equação (6) como:

$$x = W_1^t W_2^t \cdots W_n^t D_{p-1}^{-1} D_{up}^{-1} W_n \cdots W_2 W_1 b \qquad (3)$$

onde as operações com a diagonal D^{-1} são agora divididas em D_{p-1}^{-1}, referente às primeiras $(p-1)$ partições e D_{up}^{-1}, referente à última partição.

Como as operações correspondentes à última partição não tem relação com as operações de D_{p-1}^{-1}, a equação (8) pode ser reescrita como:

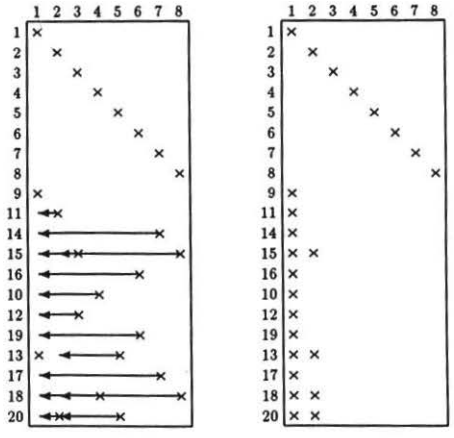
$$x = W_1^t W_2^t \cdots W_{n-1}^t D_{p-1}^{-1} W_{up} W_{n-1} \cdots W_2 W_1 b \quad (4)$$

onde $W_{up} = W_n^t D_{up}^{-1} W_n$ e' calculada de forma que as operações correspondentes a "forward", diagonal e "backward" desta última partição são feitas de uma só vez, sendo portanto calculada a matriz W_{up} que agora é cheia. Nesta matriz aparece adicionais elementos diferentes de zero, o que a primeira vista pode parecer uma desvantagem, mas este trabalho extra é que permite explorar diretamente processamento vetorial na parte menos esparsa da matriz.

3. PROCESSAMENTO VETORIAL

A referência [7] propõe uma solução usando matrizes de fatores inversos W com particionamento, sem gerar nenhum novo elemento diferente de zero (portanto preservado a esparsidade de L). Observa-se que todas as operações de cada coluna podem ser realizadas de forma vetorial, porém cada coluna contém poucos elementos diferentes de zero, acarretando vetores muito curtos e, assim de pouco eficiência para processamento vetorial. Nota-se ainda que todas as operações de multiplicação dentro de uma partição podem ser feitas sem nenhuma relação de precedência e isto, levou a colocar os elementos de cada partição em um longo vetor. Podendo-se então, realizar todas as multiplicações de uma mesma partição usando os recursos de processamento vetorial. As operações de adição continuam sendo feitas de forma escalar devido as relações de não-simultaneidade.

Recente publicação [8] usa matriz W com as mesmas diretrizes de particionamento de [7], mas implementa uma metodologia de armazenamento de cada partição que torna mais eficiente a solução vetorial dentro de cada partição. São criadas pseudocolunas dentro de cada partição, veja Figura 4, obtendo-se colunas densas que são então processadas vetorialmente.



(a) col. originais (b) pseudocolunas

Figura 4 - Formação das pseudocolunas

O número máximo de elementos por linha em cada partição determina o número máximo de pseudocolunas desta partição, sendo que em sistemas reais este número é pequeno, conforme nota-se pelas Tabelas I e II, que mostram estes números para uma ordenação MLMD, com o particionamento de [5]. Nestas tabelas são feitos vários particionamentos com a finalidade de verificar o tamanho da última partição, utilizando o sistema teste IEEE-118 com 118 barras, e o sistema sul-sudeste brasileiro com 1729 barras. O tamanho de cada partição pode ser obsevado na Tabela III, que mostra também que a ordenação MLMD é mais adaqueda para este particionamento de W, por gerar partições maiores no ínicio e criar "fill-ins" nas partições que formarão a última que será uma matriz cheia, de acordo com o particionamento que pretende-se usar. Os números entre colchetes indicam os elementos diferentes de zero originais

na matriz L, e os números entre parenteses indicam os "fill-ins" gerados por cada ordenação.

Tabela I - Sistema com 118 barras.

Passos seriais	Partições 2	3	4	5	6	7
1 (W_1)	4	4	4	4	4	4
2 (W_2)	–	4	4	4	4	4
3 (W_3)	–	–	4	4	4	4
4 (W_4)	–	–	–	3	3	3
5 (W_5)	–	–	–	–	2	2
6 (W_6)	–	–	–	–	–	2
7 (W_{up})	62	35	22	14	10	8
8 (W_6^t)	–	–	–	–	–	8
9 (W_5^t)	–	–	–	–	9	9
10 (W_4^t)	–	–	–	6	6	6
11 (W_3^t)	–	–	6	6	6	6
12 (W_2^t)	–	3	3	3	3	3
13 (W_1^t)	3	3	3	3	3	3
total	69	49	46	47	54	62

Tabela II - Sistema com 1729 barras.

Passos seriais	Partições 4	5	6	7	8	9
1 (W_1)	9	9	9	9	9	9
2 (W_2)	7	7	7	7	7	7
3 (W_3)	5	5	5	5	5	5
4 (W_4)	–	4	4	4	4	4
5 (W_5)	–	–	4	4	4	4
6 (W_6)	–	–	–	4	4	4
7 (W_7)	–	–	–	–	4	4
8 (W_8)	–	–	–	–	–	3
9 (W_{up})	221	143	97	72	56	46
10 (W_8^t)	–	–	–	–	–	17
11 (W_7^t)	–	–	–	–	15	15
12 (W_6^t)	–	–	–	10	10	10
13 (W_5^t)	–	–	10	10	10	10
14 (W_4^t)	–	9	9	9	9	9
15 (W_3^t)	7	7	7	7	7	7
16 (W_2^t)	7	7	7	7	7	7
17 (W_1^t)	6	6	6	6	6	6
total	262	197	165	154	157	167

Tabela III - Distribuição das colunas nas diferentes profundidades.

Prof.	118-barras[179] MD (86)	MLMD (122)	MDMNP (86)	1729-barras[2154] MD (1201)	MLMD (1732)	MDMNP (1211)
0	48	56	52	807	956	875
1	25	27	26	347	380	364
2	11	13	12	184	172	168
3	6	8	8	107	78	98
4	6	4	5	68	46	57
5	5	2	3	45	25	37
6	3	1	3	34	16	26
7	3	1	2	24	10	20
8	3	1	2	18	7	11
9	2	1	2	15	4	8
10	1	1	1	9	4	7
11	1	1	1	8	3	6
12	1	1	1	7	3	5
13	1	1		5	3	3
14	1			3	3	3
15	1			3	2	3
16				3	2	2
17				3	2	2
18				2	1	2
19				2	1	2
20				2	1	2
21				2	1	2
22				2	1	2
23				2	1	2
24				2	1	2
25				2	1	2
26				2	1	2
27				1	1	2
28				1	1	1
⋮				⋮	⋮	⋮
prof. max.	15	13	12	47	30	41

O fato de poder-se criar poucas pseudocolunas nas primeiras particões, por exemplo 4 para a primeira partição do sistema 118 que contém 56 colunas originais, mostra a eficiência desta técnica de armazenamento para o processamento vetorial.

4. METODOLOGIA PROPOSTA

Neste trabalho propõe-se utilizar o particionamento de [5], com a técnica de criar e armazenar pseudocolunas para as primeiras partições, e quando as estas tenderem a ficar com poucas colunas agrupá-las em uma única partição, obtendo-se uma matriz cheia que substituirá as etapas tradicionais: "forward" e "backward" para as colunas agrupadas. Assim a metodologia busca aproveitar o bom desempenho da proposta da referência [8], que foi desenvolvida buscando não criar trabalho adicional e que notadamente é boa para partições grandes, mas cria vetores curtos para as partições com poucas colunas, que são muitas nos problemas de redes elétricas. Neste ponto busca-se explorar a vantagem de outra metodologia feita para multiprocessamento, mas que se adapta perfeitamente bem

para processamento vetorial, que é o agrupamento em uma matriz cheia. Desta forma a solução tem agora os seguintes passos:

- ordenação MLMD;
- fatoração LDU;
- formação da matriz W particionada de acordo com [5], levando em conta as características do computador para escolher o tamanho da última partição (W_{up});
- formar pseudocolunas para as $p-1$ partições e adaptar o vetor b para as pseudocolunas;
- obtenção da solução com as pseudocolunas e a partição W_{up} de forma vetorial.

Na solução de problemas como estabilidade transitória, transitórios eletromagnéticos e outros, os primeiros quatro passos são realizados apenas uma vez para centenas de vezes que se faz o último passo, daí a grande importância de conseguir-se ganho nesta etapa.

5. CONCLUSÕES

Resolver problemas de redes elétricas com técnicas de programação de matrizes esparsas em computadores de alto desempenho, exige mudanças significativas: na forma como tradicionalmente são armazenados os elementos das matrizes; nos algoritmos de ordenação; e principalmente na etapa de solução. Muitas propostas foram apresentadas nos últimos anos para processamento em máquinas com multiprocessadores, e mais recentemente algumas para processamento vetorial. Neste sentido este trabalho buscou aproveitar este desenvolvimento para propor uma nova metodologia que aproveita as vantagens de duas metodologias anteriores, uma feita propriamente para processamente vetorial e outra que foi apresentada para multiprocessamento, mas que se adapta perfeitamente bem para processamento vetorial. Com o uso das vantagens de ambas a metodologia proposta traz ganhos em relação às anteriores, podendo ser de utilidade quando se pensa um utilizar sistemas computacionais eficientes como os já implementados em algumas empresas de energia elétrica, onde destaca-se o sistema mostrado na referência [1], mas que deverá brevemente ser a realidade em todas as empresas preocupadas com eficiência e qualidade no fornecimento de energia elétrica.

Referências

[1] *Vuong, G.T., Chahine, R. & Behling, S.* " Supercomputing for Power System Analysis", **IEEE Computer Application in Power**, Vol. 5, n. 3: 45-49, 1992.

[2] *Tinney, W.F., Brandwajn, V. & Chan, S.M.* "Sparse Vector Methods", **IEEE Transactions on Power Apparatus and Systems**, Vol. 104, n. 2: 295-301, 1985.

[3] *Enns, M.K., Tinney, W.F. & Alvarado, F.L.* "Sparse Matrix Inverse Factors", **IEEE Transactions on Power Systems**, Vol. 5, n. 2: 466-473, 1990.

[4] *Alvarado, F.L., Yu, D.V. & Betancourt, R.* "Partitioned Sparse A^{-1} Methods", **IEEE Transactions on Power Systems**, Vol. 5, n. 2: 452-459, 1990

[5] *Padilha, A. & Morelato, A.* "A W-matrix Methodology for Solving Sparse Network Equations on Multiprocessor Computer, **IEEE Transactios on Power Systems**, Vol.7, n. 3: 1023-1030, 1992.

[6] *Lin, S. & Van Ness J.E.* " Parallel Solution of Sparse Algebraic Equations", **In: Power Industry Computer Applications (PICA)**, Phoenix USA, May - 4th-7th:380-386, 1993.

[7] *Gomes, A. & Betancourt, R.* "Implementation of the Fast Decoupled Load Flow on a Vector Computer", **IEEE Transactions on Power Systems**, Vol. 5, n. 3: 977-983, 1990.

[8] *Granelli G.P., Pasini G.L. & Marannino P.* "A W-matrix Based Fast Decoupled Load Flow for Contingency Studies on Vector Computers", **In: IEEE PES SUMMER MEETING**, Seatle, USA, 1992.

[9] *Betancourt, R.* "An Efficient Heuristic Ordering Algorithm for Partial Matrix Refatorization", **IEEE Transaction on Power Systems**, Vol. 3, n. 3: 1181-1187, 1988.

[10] *Tinney, W. F., & Walker, J. W.* "Direct Solutions of Sparse Network Equations by Optimally Ordered Triangular Factorization", **Proceedings of the IEEE**, Vol. 55: 1801-1809, 1967.

[11] *Gomes A. & Franquelo, L.G.* "An efficient ordering algorithm to improve sparse vector methods".**IEEE Transactions on Power Systems**, Vol. 3, n. 4: 1538-1544, 1988.

[12] *Van Ness, J.E.* Discussão da referência [3], 1990.

[13] *Padilha, A. & Morelato, A.* "Cálculo de Matrizes de Fatores Inversos para Multiprocessamento de Equações de Redes Elétricas", **In: Congresso Brasileiro de Automática, 9**, Vitória - ES pp:598-603, 1992.