Algoritmo Paralelo para o Cálculo de Autovalores Associado à Avaliação da Estabilidade a Pequenas Perturbações

Jorge M. Campagnolo(M.Sc.)[1,*] Nelson Martins(PhD.)[2] José L.R. Pereira(PhD.)[3]
Djalma M. Falcão(PhD.)[1]

RESUMO

A avaliação da estabilidade a pequenas perturbações, em sistemas elétricos de potência de grande porte, é realizada pelo cálculo de autovalores dominantes de matrizes esparsas. Este trabalho apresenta melhorias significativas no algoritmo paralelo, para o cálculo de autovalores dominantes, reportado em [1]. Os autovalores são calculados pelo algoritmo de Iterações Simultâneas, implementado num computador paralelo iPSC/860 da Intel. Resultados são apresentados para um modelo prático de sistema de potência, mostrando grandes " speed-ups" obtidos em computação paralela.

ABSTRACT

Small-signal stability assessment of large power systems is obtained through the partial eigenanalysis of sparse matrices. This paper presents significant improvements to the parallel eigensolution algorithm originally reported in [1]. Eigenvalues are calculated by the Simultaneous Iteration algorithm implemented on a parallel computer Intel iPSC/860. Results are presented for a practical power system model, showing the large speed-ups obtained through parallel computation.

*Afastado do EEL/UFSC para Doutoramento
[1]COPPE-EE/UFRJ, Caixa Postal 68504, CEP 21945-970, Rio de Janeiro, RJ, Tel. (021)260-5010 R51, Fax: (021)290-6626, E-Mail: COE10028@UFRJ.bitnet.
[2]CEPEL, Caixa Postal 2754, CEP 20001-970, Rio de Janeiro, RJ, Tel. (021)598-2167, Fax: (021)260-1340.
[3]UFJF-Fac.Eng, Caixa Postal 422, CEP 36100, Juiz de Fora, MG. Fax: (032)215-6382.

1 Introdução

Técnicas para o cálculo de autovalores dominantes, na análise da estabilidade a pequenas perturbações, de sistemas elétricos de potência de grande porte já estão bem estabelecidas [2, 3, 4, 5, 6, 7]. As equações algébricas e diferenciais que representam o comportamento dinâmico do sistema elétrico de potência são linearizadas num dado ponto de operação e expressadas na sua forma não-reduzida, como mostrado no Apêndice. Isto define um sistema de equações aumentado, que possui uma estrutura bastante esparsa para a matriz Jacobiano.

Os algoritmos presentemente utilizados, para o cálculo de autovalores dominantes em problemas dinâmicos de sistemas elétricos de potência de grande porte, já estão bastante avançados. É, portanto, improvável que ganhos significativos, nos tempos computacionais, sejam obtidos utilizando apenas processamento sequencial.

Os avanços nas arquiteturas paralelas tem permitido meios rápidos para solução de problemas complexos de sistemas elétricos de potência de grande porte [8, 9, 10, 11]. A área de análise da estabilidade a pequenas perturbações, através do cálculo de autovalores/autovetores, é atrativa para a computação paralela, pela intensidade de cálculos necessários na solução do problema.

Referência [1] foi a primeira a apresentar resultados para uma implementação paralela de um algoritmo prático para o cálculo de autovalores, operando com as equações que representam o sistema de potência aumentadas e que foram espectralmente transformadas [2, 3, 4, 5, 6, 7]. Este trabalho apresenta melhorias significativas feitas no algoritmo original [1]. Estas melhorias foram obtidas por: 1) Uso de múltiplos " trial vectors" por processador; 2) Estratégia especial para os Ciclos de Iterações Rápidas; 3) Maior experiência no desenvolvimento de *software* paralelo.

Os resultados apresentados neste trabalho são para um sistema teste, derivado de um modelo de estabilidade dinâmica de grande porte do sistema interligado brasileiro.

O computador paralelo utilizado foi um iPSC/860 da Intel com 8 nós. Os *speed-ups* obtidos pela implementação paralela foram altamente significativos.

2 Algoritmo de Iterações Simultâneas

O Método de Iterações Simultâneas "Lop-sided" [6, 12, 13], determina um subespaço invariante $\mathbf{U} \in \mathbf{C}^{m \times m}$ para uma matriz não-simétrica $\mathbf{A}$, de dimensão $n \times n$, a partir de um conjunto de m vetores iniciais, denominados doravante de *trial vectors*, sendo $m < n$. Este método pode calcular eficientemente um conjunto de autovalores dominantes e seus respectivos autovalores à direita, para a matriz $\mathbf{A}$. O método é uma extensão do método das potências, para o cálculo de múltiplos autovalores/autovetores dominantes [14].

No estudo da estabilidade a pequenas perturbações, em sistemas elétricos de potência, os autovalores de interesse são os instáveis e os pouco amortecidos, que podem ser tornados dominantes através da transformação espectral $(\mathbf{A} - q\mathbf{I})^{-1}$, onde q é uma translação de eixos no plano complexo, que será chamada de *desvio complexo*, e $\mathbf{I}$ matriz identidade.

A matriz $(\mathbf{A} - q\mathbf{I})^{-1}$ possui os mesmos autovetores da matriz $\mathbf{A}$, entretanto, seus autovalores são dados pela expressão (1) [15].

$$\lambda_{ti} = \frac{1}{(\lambda_i - q)} \tag{1}$$

onde: λ_{ti} - autovalor da matriz transformada; λ_i - autovalor da matriz original e q - desvio complexo.

O Método de Iterações Simultâneas "Lop-sided" para matrizes espectralmente tranformadas é conhecido na literatura como Iterações Simultâneas "Lop-sided" Inverso Implícito (ISLSII) [2, 16]. Esta técnica computa os m autovalores de $\mathbf{A}$ que estão mais próximos do desvio complexo q e seus autovetores à direita associados. O algoritmo ISLSII pode ser visto como uma combinação do Método de Iterações Simultâneas e o Método de Iteração Inversa Implícito [2, 6, 12, 13, 14].

Nota Importante: A formulação do Método de Iterações Simultâneas, para entrar em concordância com as referências [6, 12, 13], é apresentada na sua forma direta, entretanto, o algoritmo deste trabalho é, na realidade, aplicado a uma forma implícita de $(\mathbf{A} - q\mathbf{I})^{-1}$, que é altamente esparsa (ver Apêndice). Por razões de brevidade, o algoritmo ISLSII será referido como SI.

2.1 Formulação Matemática

Seja a matriz não-defectiva $\mathbf{A}$ [15]. Agrupa-se seus autovalores, em ordem decrescente de seus valores absolutos, nas matrizes $\mathbf{\Lambda}_a$ e $\mathbf{\Lambda}_b$, sendo $\mathbf{\Lambda}_a = diag\{\lambda_1, ..., \lambda_m\}$ e $\mathbf{\Lambda}_b = diag\{\lambda_{m+1}, ..., \lambda_n\}$, que em forma matricial compacta se apresentam conforme expressão (2) [6, 12, 13, 14].

$$\mathbf{\Lambda} = \begin{bmatrix} \mathbf{\Lambda}_a & 0 \\ 0 & \mathbf{\Lambda}_b \end{bmatrix} \tag{2}$$

A matriz dos autovetores à direita de $\mathbf{A}$ pode ser particionada da mesma forma, conforme mostra a expressão (3).

$$\mathbf{Q} = [\mathbf{Q}_a \; \mathbf{Q}_b] = [\underline{q}_1 \cdots \underline{q}_m | \underline{q}_{m+1} \cdots \underline{q}_n] \tag{3}$$

A partir da equação de definição de autovalor [15], pode-se chegar às expressões (4) e (5).

$$\mathbf{A}\mathbf{Q}_a = \mathbf{Q}_a\mathbf{\Lambda}_a \tag{4}$$

$$\mathbf{A}\mathbf{Q}_b = \mathbf{Q}_b\mathbf{\Lambda}_b \tag{5}$$

O objetivo do método é determinar todos os autovalores de $\mathbf{\Lambda}_a$ e seus autovetores à direita associados.

Inicialmente, estima-se m vetores normalizados e linearmente independentes (*trial vectors*), definidos pelas colunas da matriz $\mathbf{U}$.

$$\mathbf{U} = [\underline{u}_1 \ldots \underline{u}_m] \in \mathbf{C}^{n \times m} \tag{6}$$

Potencializa-se os vetores $(\underline{u}_j, j = 1, ..., m)$ pela pré-multiplicação da matriz $\mathbf{A}$ com a matriz $\mathbf{U}$, resultando m vetores, definidos pelas colunas da matriz $\mathbf{V} = [\underline{v}_1 \ldots \underline{v}_m] \in \mathbf{C}^{n \times m}$, conforme mostra a expressão (7).

$$\mathbf{V} = \mathbf{A}\mathbf{U} \tag{7}$$

As colunas da matriz $\mathbf{U}$ podem ser determinadas como uma combinação linear dos autovetores à direita de $\mathbf{A}$, como mostra a expressão (8).

$$\mathbf{U} = \mathbf{Q}_a\mathbf{C}_a + \mathbf{Q}_b\mathbf{C}_b \tag{8}$$

onde $\mathbf{C}_a \in \mathbf{C}^{m \times m}$ e $\mathbf{C}_b \in \mathbf{C}^{(n-m) \times m}$ são coeficientes matriciais. Substituindo a expressão (8) na expressão (7) chega-se a expressão (9).

$$\mathbf{V} = \mathbf{A}\mathbf{U} = \mathbf{Q}_a\mathbf{\Lambda}_a\mathbf{C}_a + \mathbf{Q}_b\mathbf{\Lambda}_b\mathbf{C}_b \tag{9}$$

Observa-se que o primeiro termo na expressão (9) é mais dominante que na expressão (8). Isso se deve ao fato de que os autovalores em Λ_a são de maiores valores absolutos do que em Λ_b, o que implica que as componentes de $\mathbf{Q}_b$ em $\mathbf{V}$ devem de alguma forma diminuir. O processo iterativo torna os coeficientes $\mathbf{C}_b$ desprezíveis. Para aumentar a dominância do primeiro termo, uma quantidade de potencializações $\mathbf{V} = \mathbf{AU}$ podem ser realizadas . Desta forma, as matrizes $\mathbf{U}$ e $\mathbf{V}$ podem ser aproximadamente determinadas pelas expressões (10) e (11).

$$\mathbf{U} \approx \mathbf{Q}_a \mathbf{C}_a \tag{10}$$

$$\mathbf{V} \approx \mathbf{Q}_a \Lambda_a \mathbf{C}_a \tag{11}$$

Definindo-se as matrizes $\mathbf{G}$ e $\mathbf{H}$ como mostram as expressões (12) e (13).

$$\mathbf{G} = \mathbf{U}^H \mathbf{U} \approx \mathbf{U}^H \mathbf{Q}_a \mathbf{C}_a \tag{12}$$

$$\mathbf{H} = \mathbf{U}^H \mathbf{V} \approx \mathbf{U}^H \mathbf{Q}_a \Lambda_a \mathbf{C}_a \tag{13}$$

sendo que o superescrito $(^H)$ representa o conjugado transposto e assumindo que $\mathbf{U}^H \mathbf{Q}_a$ é não singular, pode-se chegar a expressão (14).

$$\mathbf{G}^{-1}\mathbf{H} \approx \mathbf{C}_a^{-1}(\mathbf{U}^H \mathbf{Q}_a)^{-1} \mathbf{U}^H \mathbf{Q}_a \Lambda_a \mathbf{C}_a = \mathbf{C}_a^{-1} \Lambda_a \mathbf{C}_a \tag{14}$$

Um processo de reorientação, para os vetores potencializados, envolve o cálculo de todos autovalores e seus respectivos autovetores à direita para a matriz de interação $\mathbf{B}$, de ordem $m \times m$, resultante da solução da equação (15).

$$\mathbf{GB} = \mathbf{H} \tag{15}$$

A partir das expressões (14) e (15), chega-se a expressão (16).

$$\mathbf{C}_a \mathbf{B} \approx \Lambda_a \mathbf{C}_a \tag{16}$$

Assim, Λ_a e $\mathbf{C}_a$ aproximam os autovalores e respectivos autovetores à esquerda da matriz $\mathbf{B}$. Se $\mathbf{P}$ for a matriz dos autovetores à direita de $\mathbf{B}$, pode, então, ser aproximadamente calculada pela expressão (17) [15].

$$\mathbf{P} \approx \mathbf{C}_a^{-1} \tag{17}$$

A expressão (18) resulta num conjunto melhorado para autovetores à direita, da matriz $\mathbf{A}$, sendo $\mathbf{W}$ o novo conjunto de vetores estimados. O conjunto de *trial vectors* melhorado é normalizado, para dar sequência ao processo iterativo. Os autovalores calculados para a matriz $\mathbf{B}$ resultam numa nova estimativa para os m autovalores da matriz $\mathbf{A}$. O procedimento acima pode ser repetido até chegar-se ao número de autovalores/autovetores desejados.

$$\mathbf{W} = \mathbf{VP} \approx \mathbf{Q}_a \Lambda_a \tag{18}$$

Com o objetivo de aumentar a taxa de convergência, quando deseja-se determinar m autovalores/autovetores dominantes, s *trial vectors* adicionais, conhecidos na literatura como vetores de guarda, são incorporados [14]. A inclusão dos vetores de guarda ao método aumenta a dimensão da matriz de interação $\mathbf{B}$ para $(m+s) \times (m+s)$. A experiência prática mostrou que o valor de s ideal é em torno de 25% do valor de m.

2.2 Implementação Sequencial do Algoritmo SI

A partir de uma estimativa $\mathbf{U}^{(k)}$ para os m autovetores à direita da matriz $\mathbf{A}$ (*trial vectors*) e um dado desvio complexo q, o ciclo iterativo do Método de Iterações Simultâneas pode ser descrito, para a iteração (k), como:

1. Calcular $\mathbf{V}^{(k)}$, $\mathbf{G}^{(k)}$ e $\mathbf{H}^{(k)}$

$$\mathbf{V}^{(k)} = (\mathbf{A} - q\mathbf{I})^{-1}\mathbf{U}^{(k)} \tag{19}$$

$$\mathbf{G}^{(k)} = \mathbf{U}^{(k)^H}\mathbf{U}^{(k)} \tag{20}$$

$$\mathbf{H}^{(k)} = \mathbf{U}^{(k)^H}\mathbf{V}^{(k)} \tag{21}$$

2. Obter $\mathbf{B}^{(k)}$ a partir da equação

$$\mathbf{G}^{(k)}\mathbf{B}^{(k)} = \mathbf{H}^{(k)} \tag{22}$$

3. Obter os autovalores e autovetores à direita da matriz $\mathbf{B}^{(k)}$

$$\mathbf{B}^{(k)}\mathbf{\Lambda}_{\mathbf{B}}^{(k)} = \mathbf{P}^{(k)}\mathbf{\Lambda}_{\mathbf{B}}^{(k)} \tag{23}$$

4. Obter a nova estimativa $\mathbf{U}^{(k+1)}$

$$\mathbf{W}^{(k+1)} = \mathbf{V}^{(k)}\mathbf{P}^{(k)} \tag{24}$$

$$\underline{\mathbf{u}}_i^{(k+1)} = \frac{\underline{\mathbf{w}}_i^{(k)}}{\|\underline{\mathbf{w}}_i^{(k)}\|_\infty} \quad i = 1, ..., m \tag{25}$$

5. Testar a convergência para os elementos dos vetores $\underline{\mathbf{u}}_i^{(k+1)}$, $i = 1, ..., m$. Se convergiu para o número de autovalores/autovetores desejados, pare. Caso contrário volte ao passo 1.

3 Algoritmo SI Paralelo

O algoritmo SI mostra um potencial natural para paralelização. O maior esforço computacional deste algoritmo está relacionado com as soluções dos vários *trial vectors*. A estratégia de paralelismo óbvia é realizar a solução de cada *trial vector* num diferente processador. Estratégias para paralelização do algoritmo SI tem sido sugeridas em outros campos da engenharia. A maioria destas aplicações, entretanto, referem-se a cálculo de autovalores dominantes de matrizes simétricas [11, 17, 18]. A contribuição de [1] foi a implementação em paralelo do algoritmo SI, aplicado a solução de autovalores/autovetores para matrizes não-simétricas e utilizando um sistema de equações aumentado, associado ao problema da estabilidade a pequenas perturbações [2, 3, 4, 5, 6, 7].

A figura 1 mostra o fluxograma da implementação em paralelo do algoritmo de Iterações Simultâneas, previamente reportado em [1]. Este fluxograma está relacionado com um $i^{ésimo}$ processador genérico. Os blocos desenhados com linhas cheias descrevem a computação numérica realizada pelo processador. A parte desenhada com linhas pontilhadas refere-se a comunicação de dados entre os processadores. A figura 1 é válida para o caso de ter-se somente uma solução de *trial vector* por processador.

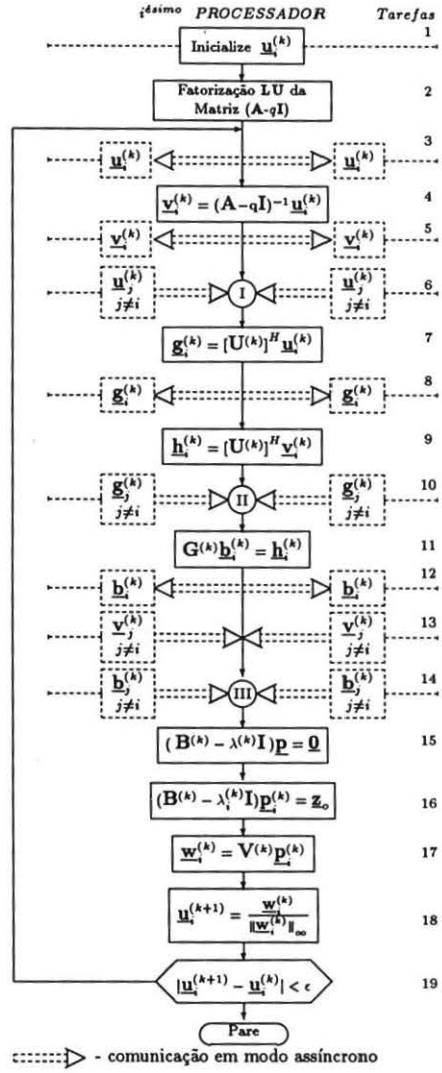

FIGURA 1: Algoritmo SI para um Processador Genérico (i) de um computador paralelo (MIMD).

3.1 Passos do Algoritmo SI Paralelo

1. Inicialize m *trial vectors*, representados pelas colunas da matriz $\mathbf{U}^{(k)}$.
2. Obtenha os fatores $\mathbf{LU}$ para a matriz $(\mathbf{A} - q\mathbf{I})$. Onde q é um desvio complexo e $\mathbf{I}$ matriz identidade.
3. Comunique assincronamente o vetor $\underline{\mathbf{u}}_i^{(k)}$ para o processador j, $j = 1, ..., m$; $j \neq i$.
4. Resolva para $\underline{\mathbf{v}}_i^{(k)}$ dados $(\mathbf{A} - q\mathbf{I})^{-1}$ e $\underline{\mathbf{u}}_i^{(k)}$. Observe que, na realidade, utiliza-se os fatores $\mathbf{LU}$ da matriz $(\mathbf{A} - q\mathbf{I})$.
5. Comunique assincronamente o vetor $\underline{\mathbf{v}}_i^{(k)}$ para o processador j, $j = 1, ..., m$; $j \neq i$.
6. Receba assincronamente o vetor $\underline{\mathbf{u}}_j^{(k)}$ do processador j, $j = 1, ..., m$; $j \neq i$.
7. Obtenha vetor coluna $\underline{\mathbf{g}}_i^{(k)}$ dada a matriz $\mathbf{U}^{(k)^H}$ e o vetor coluna $\underline{\mathbf{u}}_i^{(k)}$.
8. Comunique assincronamente o vetor $\underline{\mathbf{g}}_i^{(k)}$ para o processador j, $j = 1, ..., m$; $j \neq i$.
9. Obtenha vetor coluna $\underline{\mathbf{h}}_i^{(k)}$ da matriz $\mathbf{H}^{(k)}$ dada a matriz $\mathbf{U}^{(k)^H}$ e o vetor coluna $\underline{\mathbf{v}}_i^{(k)}$.
10. Receba assincronamente o vetor $\underline{\mathbf{g}}_j^{(k)}$ do processador j, $j = 1, ..., m$; $j \neq i$.
11. Obtenha a decomposição de Choleski para a matriz $\mathbf{G}^{(k)}$ e determine o vetor coluna $\underline{\mathbf{b}}_i^{(k)}$ da matriz de interação $\mathbf{B}^{(k)}$.
12. Comunique assincronamente o vetor $\underline{\mathbf{b}}_i^{(k)}$ para o processador j, $j = 1, ..., m$; $j \neq i$.
13. Receba assincronamente o vetor $\underline{\mathbf{v}}_j^{(k)}$ do processador j, $j = 1, ..., m$; $j \neq i$.
14. Receba assincronamente o vetor $\underline{\mathbf{b}}_j^{(k)}$ do processador j, $j = 1, ..., m$; $j \neq i$.
15. Obtenha os autovalores da matriz $\mathbf{B}^{(k)}$ utilizando fatoração $\mathbf{QR}$ [19, 20].
16. Obtenha o autovetor à direita $\underline{\mathbf{p}}_i^{(k)}$ correspondente ao autovalor $\lambda_i^{(k)}$ da matriz $\mathbf{B}^{(k)}$, utilizando uma rotina de iteração inversa.
17. Obtenha o vetor coluna $\underline{\mathbf{w}}_i^{(k)}$ a partir da matriz $\mathbf{V}^{(k)}$ e do vetor coluna $\underline{\mathbf{p}}_i^{(k)}$.
18. Obtenha uma atualização do vetor coluna $\underline{\mathbf{u}}_i^{(k+1)}$ para a próxima iteração.
19. Compare os elementos do vetor coluna $\underline{\mathbf{u}}_i^{(k+1)}$ com $\underline{\mathbf{u}}_i^{(k)}$, $i = 1, ..., m$, se convergiu, pare. Caso contrário volte ao passo 3.

3.2 Comentários sobre a Implementação Paralela do Algoritmo SI

Os seguintes comentários estão relacionados com a implementação do Algoritmo SI em computador paralelo [1].

- O código do algoritmo utilizado em processamento sequencial foi significativamente modificado, a fim de permitir eficiente computação paralela e reduzir a comunicação entre processadores.

- O mesmo programa roda em todos os processadores, entretanto, operam com diferentes dados. Em computadores MIMD, como o iPSC/860, isto frequentemente é chamado SPMD ("Single Program Multiple Data") [21]. A maior vantagem do SPMD é a necessidade de somente um programa a ser carregado e copiado nos processadores.
- O cálculo dos autovalores e autovetores da matriz de interação $\mathbf{B}^{(k)}$ é realizado através de rotinas do " EISPACK" [19, 20]. Uma rotina **QR** é utilizada para o cálculo dos autovalores. Cada processador utiliza, então, uma rotina de iteração inversa para o cálculo de somente um autovetor.
- Igual balanço de carga entre processadores é obtido pela solução de cada "trial vector" por processador.
- Soluções de autovetores aumentados são obtidas pela utilização do sistema de equações aumentado, entretanto, somente os valores das variáveis de estado necessitam ser comunicados.
- Após a convergência de um par autovalor/autovetor, o processador associado torna-se relativamente ocioso. Entretanto, a redução da eficiência do algoritmo não é tão intensa, em vista de que após qualquer convergência segue também uma significativa redução das comunicações necessárias.
- O código fonte foi programado em FORTRAN 77 para computação paralela.

4 Informações Adicionais

4.1 Computador Paralelo Utilizado

O computador paralelo do Núcleo de Processamento Paralelo da COPPE/UFRJ é um iPSC/860 com 8 nós da Intel, sendo que cada nó é formado por um microprocessador i860 com 8 MBytes de memória. O i860 é um microprocessador de 64-bit, de tecnologia RISC, e tem performance acima de 80 MFLOPS em precisão simples e 60 MFLOPS em precisão dupla. O iPSC/860 é um computador de memória distribuida, com configuração hipercúbica. Os dados, entre os nós, são transferidos através da comunicação de mensagens. O estabelecimento do caminho, para envio de mensagens, gasta em torno de 65 microsegundos de CPU. Estabelecido o caminho, a capacidade de transferência de dados é, no pico, da ordem de 2.8 MBytes por segundo [21].

4.2 Sistema Teste

O sistema teste deste trabalho é derivado de um modelo prático de estabilidade transitória do sistema brasileiro interligado.

Tal sistema possui 544 barras, 1093 linhas de transmissão, 75 geradores com controladores associados e um elo de corrente contínua. O autovetor aumentado para este sistema possui ordem 3156, sendo 2366 variáveis algébricas e 790 variáveis de estado.

4.3 Avaliação da Performance

A avaliação de desempenho para implementações paralelas normalmente é baseada em dois índices: "*Speed-up (Sp)*" e "*Eficiência (Ep)*" [22]. *Sp* é definido como a razão entre o tempo computacional do algoritmo sequencial em um único processador *(Ts)* e o tempo gasto por p processadores *(Tp)*. A *Eficiência* é definida como sendo a razão entre *Sp* e p.

5 Melhorias Implementadas no Algoritmo SI Paralelo

5.1 Múltiplos "Trial Vectors" por Nó

A implementação do algoritmo SI mostrada no fluxograma da Figura 1 [1], tem todo processador solucionando um único *trial vector*. Após a convergência do $i^{ésimo}$ autovalor/autovetor, o associado $i^{ésimo}$ processador torna-se relativamente ocioso, causando uma redução na eficiência computacional. Adicionalmente, a implementação de um vetor por nó limita o número de *trial vectors* ao número de nós do computador.

Estas limitações foram contornadas pela colocação de múltiplas soluções de *trial vectors* em cada nó do computador paralelo. Estes vetores são colocados nos vários processadores em um modo entrelaçado, permitindo um maior balanço de carga entre os processadores.

Como exemplo, considere um computador paralelo com 4 nós solucionando para 8 *trial vectors*. Assumindo que a convergência dos vetores tem a sequência: $\underline{u}_1$, $\underline{u}_2$, $\underline{u}_3$, $\underline{u}_4$, $\underline{u}_5$, $\underline{u}_6$, $\underline{u}_7$, $\underline{u}_8$ e que o nó 1 resolve para o par ($\underline{u}_1$, $\underline{u}_2$); nó 2 resolve para o par ($\underline{u}_3$, $\underline{u}_4$); o nó 3 resolve para o par ($\underline{u}_5$, $\underline{u}_6$) e o nó 4 resolve para o par ($\underline{u}_7$, $\underline{u}_8$). Note que, neste esquema, o processador 1 torna-se ocioso após a convergência do vetor $\underline{u}_2$. Um esquema melhor é ter: nó 1 resolvendo para ($\underline{u}_1$, $\underline{u}_5$); nó 2 resolvendo para ($\underline{u}_2$, $\underline{u}_6$); nó 3 resolvendo para ($\underline{u}_3$, $\underline{u}_7$) e nó 4 resolvendo para ($\underline{u}_4$, $\underline{u}_8$). Esta solução entrelaçada para os *trial vectors*, entre os nós do computador, faz o nó 1 muito ativo até a convergência de $\underline{u}_5$. O mesmo raciocínio se estende aos demais nós.

5.2 Estratégia para os ciclos de Iterações Rápidas

A taxa de convergência do método é primariamente dependente da velocidade que o primeiro termo da equação (9) se torne dominante. Desta forma a realização de um número de potencializações, antes de dar sequência ao algoritmo, pode acelerar a convergência do algoritmo [13, 14]. Durante as potencializações todo o processo de reorientação não é realizado, de forma a ter uma razoável economia no tempo de execução, advindo, então, o nome de ciclos de iterações rápidas.

Significativos ganhos em tempos de CPU são obtidos pelo emprego dos ciclos de iterações rápidas. A experiência acumulada, em casos de aplicações em Sistemas Elétricos de Potência [7, 23], favorecem $K_f = 3$ como um valor ótimo, onde $K_f - 1$ representa o número de omissões do processo de reorientação.

Existe o perigo do Método de Iterações Simultâneas não chegar a convergência em função da utilização de um valor K_f maior que 3 , pois sucessivas soluções podem tornar linearmente dependentes os *trial vectors*, representados pela colunas da matriz **U** [13, 14].

Ciclos de iterações rápidas são vantajosos tanto em computação sequencial quanto em computação paralela. Entretanto, no segundo caso, existe um benefício extra, devido ao fato de aumentar-se o processamento local de cada processador sem a necessidade de uma comunicação extra entre os mesmos.

A estratégia implementada neste trabalho é empregar diferentes níveis para os ciclos de iterações rápidas para as soluções dos *trial vectors*. Considere, para ilustrar esta estratégia, a performance do algoritmo para uma dada matriz e um dado desvio complexo, quando tendo 8 *trial vectors*. O algoritmo, neste caso, pode divergir para $K_f = 5$ e $K_f = 4$ e converge para $K_f = 3$. Convergência, entretanto, pode ser seguramente realizada, quando utiliza-se $K_f = 5$ para os primeiros 3 vetores e $K_f = 3$ para os 5 restantes. Neste caso, $K_f = 5$ junto com $K_f = 3$ propicia um ganho significativo no tempo de computação.

Esta estratégia gera algum desbalanço de carga entre os processadores do computador paralelo, mas os atrasos extras introduzidos são muito pequenos, devido a utilização de comunicações de modo assíncrono. Os ganhos gerais são significativos, já que existe uma consistente redução no número de iterações globais para a convergência do algoritmo.

A adoção desta original idéia adveio de observações práticas, mostrando que a divergência do algoritmo, pela adoção de um valor alto para K_f, começa pelos autovetores correspondentes aos autovalores mais distantes do desvio complexo definido. Assim, a adoção de valores maiores para K_f, nas soluções de *trial vectors* correspondentes aos autovalores mais próximos ao desvio complexo, viabilizam a convergência do algoritmo ao mesmo tempo que propicia um ganho real nos tempos computacionais. Estes *trial vectors* são representados pelas primeiras colunas da matriz U no algoritmo da Figura 1 e são os primeiros a convergir.

6 Resultados em Processamento Paralelo

Este trabalho apresenta avanços obtidos no algoritmo proposto em [1], e os resultados apresentados para o sistema exemplo são para um desvio complexo ($q = 0 + j3.5$). Os autovalores convergidos são $\lambda_1 = -0.11639 + j3.20177$, $\lambda_2 = -0.09245 + j3.98271$, $\lambda_3 = -0.59110 + j4.69345$, $\lambda_4 = +0.18141 + j4.83230$, $\lambda_5 = -0.60275 + j4.89657$, $\lambda_6 = -0.01047 + j5.20738$, $\lambda_7 = -0.47956 + j5.41204$, $\lambda_8 = -2.05604 + j3.79736$. Os autovalores são numerados de acordo com a ordem natural de convergência do algoritmo SI, para o desvio complexo utilizado. Os resultados são para uma tolerância de 10^{-5}nos elementos dos vetores $\underline{\mathbf{u}}_i^{(k)}$, $i = 1, ...m$.

Figura 2 mostra graficamente os *speed-ups* obtidos com o algoritmo proposto em [1], utilizando 8 processadores e 8 *trial vectors*, para $K_f = 3$.

A estratégia de ter-se vários *trial vectors* por processador foi desenvolvida tendo em mente o aumento da eficiência do algoritmo SI. Os histogramas das Figuras 3 e 4 apresentam medidas da performance do algoritmo, quando da utilização de 8 *trial vectors*. Uma barra com indicação λ_4, por exemplo, relaciona-se com o tempo computacional necessário ou *Eficiência* do algoritmo, para convergência do autovalor λ_4 e seu autovetor associado.

A figura 3 mostra que a *Eficiência* é mais alta para a configuração paralela com 2 nós (4 *trial vectors* por nó), comparada a configuração paralela com 8 nós (1 *trial vector* por nó). Note, entretanto, que a Figura 4 mostra uma redução no tempo computacional com o aumento do número de processadores. Portanto, se for requerido a máxima utilização da máquina, a solução prática é a utilização de 2 processadores, cada um tendo 4 *trial vectors*. Por outro lado, se um tempo mínimo de computação é requerido, utiliza-se 8 processadores do computador paralelo.

Uma significativa redução no tempo computacional foi obtida pela adoção da estratégia para os ciclos de iterações rápidas, descritos na Seção 5.2. Esta estratégia leva a convergência dos autovalores em um menor número de iterações. Estes ganhos computacionais mostraram ser muito consistentes para um largo número de desvios complexos e vários sistemas teste. Os resultados da Figura 5, obtidos quando da utilização de 16 *trial vectors* e 8 processadores, mostram claramente os ganhos advindos com a estratégia proposta.

Convergência frequentemente ocorre para mais de um autovalor/autovetor por iteração. Note que na Figura 5 (linhas cheias) a convergência é sempre aos pares: λ_1 & λ_2, λ_3 & λ_4, λ_5 & λ_6 e λ_7 & λ_8. Observe, entretanto, que λ_3 e λ_4 convergiram em diferentes iterações para a outra opção dos ciclos de iterações rápidas (linhas pontilhadas da Figura 5).

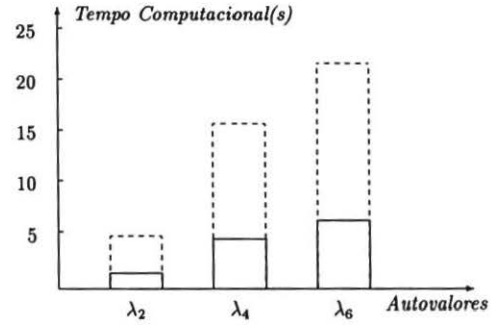

FIGURA 2: Tempo Computacional para convergência de autovalores/autovetores utilizando 8 *trial vectors* ($K_f = 3$).**Convenção**: Linhas Pontilhadas referem-se a resultados de processamento sequencial em um nó do computador paralelo. Linhas cheias referem-se a resultados do computador paralelo com 8 nós.

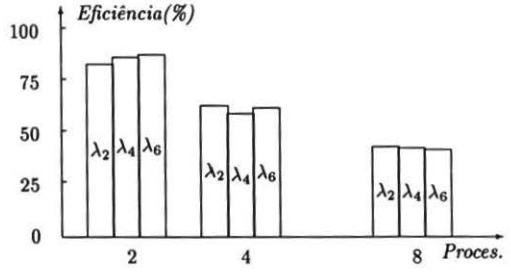

FIGURA 3: Número variável de processadores utilizando 8 *trial vectors* ($K_f = 3$).

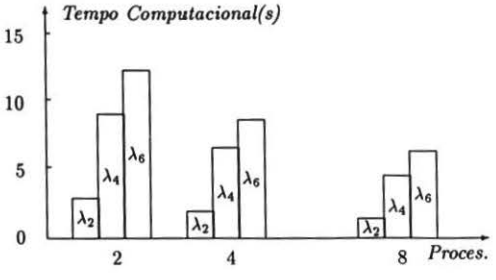

FIGURA 4: Número variável de processadores utilizando *8 trial vectors* ($K_f = 3$).

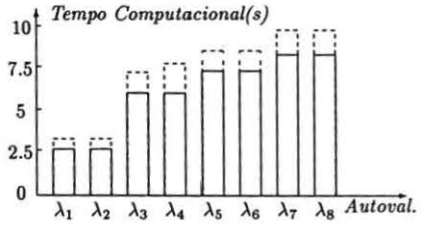

FIGURA 5: Tempo Computacional para convergência de autovalores/autovetores com 8 processadores e 16 *trial vectors*. **Convenção**: Linhas pontilhadas referem-se a $K_f = 3$ para todos *trial vectors*. Linhas cheias referem-se a $K_f = 5$ para os primeiros 8 *trial vectors* e $K_f = 3$ para os restantes.

7 Conclusões

Computadores paralelos estão hoje disponíveis em todas grandes universidades e centros de pesquisa. Eles são uma ferramenta moderna para um grande número problemas que têm necessidade intensiva de CPU. O processamento paralelo introduz aumento de complexidade no *software* e estratégia dos algoritmos. Portanto, a tarefa de converter um algoritmo sequencial em um eficiente procedimento paralelo é sempre um desafio.

O campo de sistemas de potência oferece excelente oportunidade para a computação paralela, devido as grandes dimensões dos problemas e intensos tempos computacionais envolvidos no processo de solução.

Os resultados obtidos pelo algoritmo SI paralelo, juntamente com as melhorias apresentadas, são altamente significativos, pois apresentam importantes reduções no tempo computacional para o cálculo de autovalores/autovetores de sistemas de grande porte.

Das duas melhorias apresentadas, considera-se a estratégia para os ciclos de iterações rápidas como a mais importante. Esta estratégia simples é, pelo conhecimento dos autores, original e levou a um consistente ganho computacional num largo número de casos analisados. Esta estratégia não está restrita à utilização em problemas de sistemas de potência, podendo ser estendida a aplicações gerais do algoritmo de iterações simultâneas.

O modo de comunicação assíncrono do iPSC/860 propiciou importantes ganhos na eficiência do algoritmo, se comparado ao modo de comunicação síncrono. As vantangens do modo de comunicação assíncrono, para o algoritmo SI, tornou-se evidente, após a implementação da estratégia para os ciclos de iterações rápidas.

A *Eficiência* do algoritmo SI paralelo aumentará, com a utilização de novas gerações de computadores paralelos, que possuem menores relações entre a velocidade de processamento e comunicação entre nós [24].

Referências

[1] J.M. Campagnolo, N. Martins, J.L.R. Pereira, L.T.G. Lima, H.J.C.P. Pinto, and D.M. Falcão. Fast small-signal stability assessment using parallel processing. *Paper No. 93 SM 481-2 PWRS,IEEE/PES, Winter Meeting, Vancouver, Canada*, July 1993.

[2] N. Martins. Efficient eigenvalue and frequency response methods applied to power system

small-signal stability studies. *IEEE Trans. on Power Systems*, PWRS–1, No. 1:217–226, February 1986.

[3] N. Martins, L.T.G. Lima, and H.J.C.P. Pinto. Efficient methods for finding transfer function zeros of power systems. *IEEE Trans. on Power Systems*, Vol.7, No. 3:1350–1361, August 1992.

[4] P. Kundur, G.J. Rogers, D.Y. Wong, L. Wang, and M.G. Lauby. A comprehensive computer program package for small signal stability analysis of power systems. *Paper No. 90 WM 007-5, IEEE/PES Winter Meeting, Atlanta, Georgia*, 1990.

[5] L. Wang and A. Semlyen. Application of sparse eigenvalue techniques to the small-signal stability analysis of large power systems. *IEEE Trans. on Power Systems*, PWRS–6, No. 6:635–642, May 1990.

[6] L. Wang. *Eigenvalue Analysis of Large Power Systems.* PhD thesis, University of Toronto, 1991.

[7] L.T.G. Lima. *Estudo Comparativo dos Métodos Iterativos de Cálculo de Autovalores Aplicado ao Problema da Estabilidade a Pequenas Perturbações de Sistemas Elétricos de Potência.* Master's thesis, COPPE/UFRJ, 1991.

[8] An *IEEE* Committee Report by a Task Force of the Computer and Analytical Methods Subcommittee of the Power Systems Engineering Committee. Parallel processing in power systems computation. *Paper No. 91 SM 503-3 PWRS, IEEE/PES, Summer Meeting, San Diego , CA*, July 1991.

[9] J.S. Chai and A. Bose. Bottlenecks in parallel algorithms for power system stability analysis. *Paper No. 92 WM 285-7 PWRS, IEEE/PES, Winter Meeting, New York, NY*, January 1992.

[10] B.N. Datta. Parallel and large-scale matrix computations in control: some ideas. *Linear Algebra and Its Applications*, Vol.121:243–264, 1989.

[11] H.Y. Chang and S. Utku. Applications of parallel processing in structural engineering. *Parallel Processing in Computational Mechanics, Edited by Hojjat Adeli, Marcel Dekker Inc.*, 219–243, 1992.

[12] A. Jennings and W.J. Stewart. Simultaneous iteration for partial eigensolution of real matrices. *J. Inst. Maths. Applics.*, Vol.15:351–361, 1975.

[13] W.J. Stewart and A. Jennings. A simultaneous iteration algorithm for real matrices. *ACM Trans. on Mathematical Software*, Vol.7, No.2:184–198, June 1981.

[14] A. Jennings. *Matrix Computation for Engineers and Scientists.* Wiley, London, 1977.

[15] J.H. Wilkinson. *The Algebraic Eigenvalue Problem.* Clarendon Press - Oxford, 1965.

[16] B. Gao, G.K. Morison, and P. Kundur. Voltage stability evaluation using modal analysis. *Paper No. 91 SM 420-0 PWRS, IEEE/PES, Summer Meeting, San Diego, CA*, July 1991.

[17] S. Utku, H.Y. Chang, M. Salama, and D. Rapp. Simultaneous iterations algorithm for general eigenvalue problems on parallel processors. In *International Conference on Parallel Processing IEEE*, pages 59–66, St. Charles, Illinois, August 1986.

[18] U. Schendel. *Introduction to Numerical Methods for Parallel Computers.* Ellis Horwood Limited - Chichester, 1984.

[19] B.T. Smith et al. *Matrix Eigensystem Routines EISPACK Guide.* Springer-Verlag, Berlin, 1974.

[20] E. Anderson et al. *LAPACK Users' Guide.* Society for Industrial and Applied Mathematics, 1992.

[21] G. Withers. Parallel programming on the *iPSC/860* system. *Proceedings of the fourth ISMM/IASTED International Conference on Parallel and Distributed Computing and Systems*, ISBN: 0-88986-159-5:341–343, October 1991.

[22] J. M. Ortega. *Introduction to Parallel and Vector Solution of Linear Systems.* Plenum Press, New York and London, 1988.

[23] L.H. Bezerra. *Análise da Estabilidade de Sistemas de Potência de Grande Porte.* PhD thesis, PUC/Rio de Janeiro, 1990.

[24] G. Meurant. The evolution of scientific computing on parallel computers. In *I Workshop of High Performance Scientific Computation*, Rio de Janeiro, Brasil, Agosto 1992.

AGRADECIMENTOS

Esta pesquisa foi parcialmente financiada pela FINEP (Contrato no. 5288062200). Os autores agradecem as facilidades e suporte técnico do Núcleo de Computação Paralela da COPPE/UFRJ. Os autores também agradecem ao Dr. Eugenius Kaszkurewicz por importantes discussões sobre o trabalho. O primeiro autor agradece a CAPES e a Universidade Federal de Santa Catarina pelo suporte financeiro e licença para doutoramento.

Os dados do sistema interligado brasileiro foram gentilmente cedidos pelos Engenheiros Xisto Vieira Filho e Paulo Gomes da ELETROBRAS.

APÊNDICE

Modelo para Estudo da Estabilidade a Pequenas Perturbações

O comportamento dinâmico de um sistema elétrico de potência pode ser modelado por um conjunto de equações diferenciais e um conjunto de equações algébricas, que são solucionados simultaneamente.

$$\dot{\underline{x}} = \underline{f}(\underline{x}, \underline{r}) \tag{26}$$

$$\underline{0} = \underline{g}(\underline{x}, \underline{r}) \tag{27}$$

Sendo $\underline{x}$ e $\underline{r}$ vetores de variáves de estado e algébricas respectivamente.

A análise da estabilidade a pequenas perturbações envolve a lineariazação do sistema dinâmico, que representa o sistema elétrico de potência, em um ponto de operação $(\underline{x}_0, \underline{r}_0)$, resultando nas equações lineares:

$$\begin{bmatrix} \Delta\dot{\underline{x}} \\ \underline{0} \end{bmatrix} = \begin{bmatrix} J_1 & J_2 \\ J_3 & J_4 \end{bmatrix} \begin{bmatrix} \Delta\underline{x} \\ \Delta\underline{r} \end{bmatrix} \tag{28}$$

Algoritmos eficientes para obter autovalores/autovetores dominantes para a matriz de estados **A**, de sistemas de grande porte, não necessitam do cálculo específico de **A** [2, 3, 4, 5, 6, 7]. Estes algoritmos são aplicados diretamente na matriz Jacobiano da equação (28), cuja estrutura esparsa é amplamente explorada para reduzir tempo computacional e memória requerida. Esta técnica requer a solução da seguinte equação:

$$\begin{bmatrix} \mathbf{J}_1 - q\mathbf{I} & \mathbf{J}_2 \\ \mathbf{J}_3 & \mathbf{J}_4 \end{bmatrix} \begin{bmatrix} \underline{\mathbf{w}} \\ \underline{\mathbf{v}} \end{bmatrix} = \begin{bmatrix} \underline{\mathbf{z}} \\ \underline{\mathbf{0}} \end{bmatrix} \tag{29}$$

onde $\underline{\mathbf{w}}$ e $\underline{\mathbf{v}}$ são variáveis, q é um desvio complexo especificado, $\underline{\mathbf{0}}$ é um vetor nulo, $\mathbf{I}$ é matriz de identidade e $\underline{\mathbf{z}}$ é um dado vetor complexo. O maior esforço computacional do algoritmo SI envolve as soluções repetidas da equação (29). Vinte e quatro destas soluções são necessárias por iteração do algoritmo SI, considerando $K_f = 3$ (ciclos de iterações rápidas) e $m = 8$ (número de *trial vectors*).