Exacta
ISSN: 1678-5428
exacta@uninove.br
Universidade Nove de Julho
Brasil

Pereira, Fabio Henrique; Ikuyo Nabeta, Sílvio
Uma nova abordagem baseada em wavelets para o método Multigrid Algébrico: Parte II - Algoritmo Paralelo
Exacta, vol. 5, núm. 2, julho-dezembro, 2007, pp. 304-309
Universidade Nove de Julho
São Paulo, Brasil

Uma nova abordagem baseada em *wavelets* para o método Multigrid Algébrico: Parte II – Algoritmo Paralelo

Fabio Henrique Pereira
Pós-doutorando do Laboratório de Eletromagnetismo Aplicado, Departamento de Engenharia de Energia e Automação Elétricas da Escola politécnica da Universidade de São Paulo [LMAG-PEA-EPUSP].
São Paulo – SP [Brasil]
fabio@pea.usp.br
fabio.pea@gmail.com

Sílvio Ikuyo Nabeta
Prof. Dr. do Laboratório de Eletromagnetismo Aplicado, Departamento de Engenharia de Energia e Automação Elétricas da Escola politécnica da Universidade de São Paulo [LMAG-PEA-EPUSP].
São Paulo – SP [Brasil]
nabeta@pea.usp.br

Neste trabalho, apresenta-se um novo algoritmo paralelo para o Método Multigrid Algébrico com base em *Wavelets* (PWAMG). Uma variação do algoritmo padrão de paralelização da transformada *wavelet* discreta é usada na construção de uma hierarquia de matrizes e de operadores de transferência para o método Multigrid Algébrico. O PWAMG é testado como método iterativo paralelo para a equação de Poisson bidimensional discretizada por malhas de diferenças finitas com diferentes números de nós. Resultados obtidos do programa paralelo são comparados com a versão seqüencial do mesmo método.

Palavras-chave: Memória distribuída. Multigrid algébrico. Programação paralela transformada. *Wavelet* discreta.

1 Introdução

O método multigrid algébrico (AMG) é um dos mais eficientes algoritmos para resolver sistemas grandes e esparsos de equações algébricas lineares, oriundos da aplicação do Método dos Elementos Finitos (MEF) ou das Diferenças Finitas (DF), em problemas com geometrias complexas e malhas não-estruturadas. Além de exigir apenas informações a respeito da matriz de coeficientes do sistema, o AMG apresenta ótima propriedade de escalabilidade, especialmente desejável no contexto de problemas tridimensionais e da computação paralela. Para um problema com n incógnitas, o número de V-ciclos necessários para convergência é idealmente independente do tamanho do problema. Além disso, o custo computacional da etapa de montagem e de cada V-ciclo é, no caso ideal, linearmente proporcional a n (HAASE, KUHN, REITZINGER, 2002; HENSON, YANG, 2002; KRECHEL, STÜBEN, 2001; STERCK, YANG, HEYS, 2005).

O termo escalabilidade é utilizado para indicar a capacidade de um algoritmo paralelo em resolver problemas cada vez maiores, com um aumento proporcional no custo computacional. Em outras palavras, a escalabilidade está associada ao crescimento do problema, sem que isso, no entanto, prejudique significativamente o desempenho computacional do método.

Por possuir tais características, existe grande interesse em encontrar formas efetivas de aplicar o AMG em problemas extremamente grandes, envolvendo milhões de incógnitas, o que necessariamente implica programar esse método por meio de arquiteturas computacionais com memória distribuída, como é o caso de um agrupamento de computadores pessoais, o chamado *cluster*.

Sobre esse aspecto, a principal dificuldade diz respeito ao processo de seleção das incógnitas para o subsistema. Enquanto outras etapas do AMG podem ser paralelizadas rapidamente, a de seleção das incógnitas é naturalmente seqüencial. Essa dificuldade tem motivado pesquisas com o objetivo de desenvolver procedimentos alternativos de seleção das incógnitas, e que apresentem características mais favoráveis à paralelização (CHANG, HUANG, 2002; CLEARY et al., 1998).

Neste trabalho propõe-se uma contribuição nesse aspecto, apresentando um algoritmo de paralelização do método multigrid algébrico com base em *wavelet*. Essa nova abordagem para o método multigrid algébrico (WAMG) foi proposta inicialmente por este autor, em Pereira, Verardi, Nabeta (2006), e é apresentada, do ponto de vista seqüencial, em Pereira, Nabeta (2007).

Uma das características mais importantes do WAMG é que o procedimento padrão de seleção das incógnitas, crucial na abordagem padrão e dificilmente paralelizável, não é mais realizado, o que facilita, sobremaneira, sua paralelização. Além disso, a versão em paralelo do WAMG (PWAMG) pode ser usada como um *black box solver* ou pré-condicionador, ao contrário de algumas abordagens de paralelização do AMG, tendo como base os métodos de Decomposição de Domínio que, apesar de se mostrarem eficientes, requerem informações geométricas do problema (KRECHEL, STÜBEN, 2001).

A paralelização do WAMG proposta neste estudo baseia-se em uma variação do algoritmo de paralelização da Transformada Wavelet Discreta (do inglês DWT) apresentado por Ford, Chen e Ford (2001). Neste trabalho também se propõe uma nova estratégia de paralelização do WAMG fundamentada na distribuição homogênea das linhas da matriz entre os processadores, apropriada ao bom desempenho do método.

2 Paralelização da DWT

A Transformada Wavelet Discreta é realizada usando filtros digitais passa-banda (passa-baixa e

passa-alta). Esses filtros capturam uma aproximação (coeficientes $c_{k,j}$) do sinal de entrada, bem como os detalhes desse sinal (coeficientes $d_{k,j}$) em cada nível k do processo de decomposição.

Definindo a transformada *wavelet* discreta por seus coeficientes de filtros h_1, h_2, K, h_D, em que D é conhecida como a ordem da transformada (FORD, CHEN, FORD, 2001), a DWT com k níveis de um vetor **u**

$$\mathbf{u} = \mathbf{u}_0 = (c_{0,0}, c_{0,1}, \ldots, c_{0,n})^T \quad (1)$$

com n = $2^m p$, $p \in N$ e $m \geq k$, pode ser realizada usando as relações:

$$c_{k,l} = \sum_{n} h_{n-2l} c_{0,n}$$

e

$$d_{k,l} = \sum_{n} g_{n-2l} c_{0,n}$$

ou ainda,

$$c_{k,l} = \sum_{q=1}^{D} h_q c_{0,\langle q+2l \rangle}$$

$$d_{k,l} = \sum_{q=1}^{D} g_q c_{0,\langle q+2l \rangle} \quad (2)$$

na qual $\langle q + 2l \rangle$ é o resto da divisão de $(q + 2l)$ por n.

Em cada nível de decomposição, o número de elementos a ser transformado é dividido por dois. Assim, no i-ésimo nível, o vetor u_i tem a forma

$$\mathrm{u}_i = (c_{i,0}, \ldots, c_{i,2^{(n-i)}p-1}, d_{i,0}, \ldots, d_{i,2^{(n-i)}p-1}, \ldots, d_{1,0}, \ldots, d_{1,2^{(n-i)}p-1})^T$$

Esse processo, para um vetor com oito elementos e três níveis de decomposição, é ilustrado em (3), produzindo uma aproximação do vetor **u** em cada nível.

$$\begin{array}{cccccccc} c_{0,0} & c_{0,1} & c_{0,2} & c_{0,3} & c_{0,4} & c_{0,5} & c_{0,6} & c_{0,7} \\ & & & & \downarrow & & & \\ c_{1,0} & c_{1,1} & c_{1,2} & c_{1,3} & d_{1,0} & d_{1,1} & d_{1,2} & d_{1,3} \\ & & & & \downarrow & & & \\ c_{2,0} & c_{2,1} & d_{2,0} & d_{2,1} & d_{1,0} & d_{1,1} & d_{1,2} & d_{1,3} \\ & & & & \downarrow & & & \\ c_{3,0} & d_{3,0} & d_{2,0} & d_{2,1} & d_{1,0} & d_{1,1} & d_{1,2} & d_{1,3} \end{array} \quad (3)$$

A DWT bidimensional de uma matriz A é realizada, aplicando-se o procedimento acima nas linhas e colunas da matriz. Nesse processo, a divisão da matriz entre os processadores é o ponto principal da paralelização. Alguns trabalhos têm mostrado que a má distribuição da matriz pode comprometer o desempenho do método (FORD, CHEN, FORD, 2001).

A forma mais direta de paralelizar a DWT bidimensional é dividir as linhas da matriz entre os processadores. Assim, cada processador aplica a transformação nas linhas nele armazenadas, sem necessidade de trocar qualquer informação com outro processador. Na transformação das colunas, cada um transforma uma parte dos elementos de cada coluna correspondente às linhas que ele armazena. Isso é feito usando as relações de recorrência apresentadas em (2).

Como resultado, obtém-se uma nova matriz em cada nível do processo, também dividida por linhas entre os processadores. Essas matrizes são, então, usadas para formar a hierarquia exigida pelo método multigrid algébrico (veja Pereira e Nabeta, 2007, para mais detalhes).

A necessidade de comunicação nesse processo dependerá da distribuição das linhas da matriz entre os processadores e do comprimento do filtro utilizado. Essas questões são formalizadas a seguir.

2.1 Distribuição e comunicação

Com relação à distribuição, é suficiente considerar um vetor coluna **v** dividido entre p processadores para analisar os cálculos realizados

por um processador p_j na aplicação do i-ésimo nível de uma DWT nesse vetor. Suponha, inicialmente, que p_j armazene em sua memória N_j elementos de **v**,

$$c_{0,q_j},\ c_{0,q_j}+1,\ \text{K}\ ,\ c_{0,q_j}+N_j-1 \tag{4}$$

Após i - 1 níveis, os elementos disponíveis para transformação em p_j são:

$$c_{i-1,\frac{q_j}{2^{i-1}}},\ c_{i-1,\frac{q_j}{2^{i-1}}+1},\ \ldots,\ c_{i-1,\frac{q_j}{2^{i-1}}+\frac{N_j}{2^{i-1}}-1} \tag{5}$$

Assim, de acordo com (2), não haverá comunicação no i-ésimo nível da transformação em p_j, se:

$$q+2l \le \frac{q_j+N_j}{2^{i-1}}-1,\ q=0,\ \ldots,\ D-1 \tag{6}$$

Além disso, para $q = D - 1$, tem-se que:

$$l \le \frac{q_j+N_j}{2^i}-\frac{D}{2} \tag{7}$$

Conseqüentemente, é possível estabelecer valores de D para os quais não ocorre comunicação entre os processadores.

Observe que o maior valor de $q + 2l$ usado nos cálculos no nível i em p_j é igual a:

$$(D-1)+2\left(\frac{q_j+N_j}{2^i}-1\right) \tag{8}$$

Subtraindo (8) de (6), segue que o cálculo de D - 2 elementos em p_j exige comunicação com o processador p_{j+1}, lembrando que D é o comprimento do filtro. Como o número de elementos disponíveis em p_{j+1} no nível i,

$$\frac{N_{j+1}}{2^{k+1}} \tag{9}$$

pode ser menor que D - 2, para grandes valores de D e k, é possível impor uma condição adicional sobre k para evitar a necessidade de comunicação com o processador p_{j+2}. Assim, impõe-se

$$2^{k+1} \le \left(\frac{N_j}{(D-2)},\ j=0,\ \ldots,\ p-1\right) \tag{10}$$

É importante notar que para DWT de ordem 2, com coeficientes de *Haar* ou *Daubechies*-2, nenhuma comunicação entre os processadores é necessária, por isso, a restrição (10) não se aplica. Esse resultado é muito importante no contexto do WAMG, pois os resultados numéricos têm mostrado que o uso de coeficientes de *Haar* produz resultados muito bons, na comparação com os obtidos dos filtros de maior comprimento (PEREIRA, VERARDI, NABETA, 2006; PEREIRA, NABETA, 2007). Além disso, para coeficientes de *Daubechies*-4, a restrição (10) é satisfeita sempre que $N_j = 2^k$, ou seja, sempre que o número de elementos do vetor coluna em cada processador for igual a uma potência inteira de 2.

Como resultado, é possível estabelecer um algoritmo que distribua os elementos da matriz entre os processadores, de forma que uma DWT de ordem 2 com k níveis seja realizada inteiramente em paralelo (FORD, CHEN, FORD, 2001).

3 O método WAMG em paralelo

Como a maior parte das etapas do WAMG, com produtos matriz-vetor, aplicação de suavizadores e de métodos iterativos para resolver o problema, em um nível menos refinado, é facilmente paralelizável, a principal tarefa torna-se a para-

lelização da montagem da hierarquia de matrizes por meio da DWT. Tal processo é feito com base nos conceitos apresentados na seção anterior.

No contexto do WAMG, as dimensões das matrizes nem sempre são iguais a potências inteiras de 2. Dessa forma, o algoritmo mencionado na seção anterior não é conveniente. Nesse caso, supondo o uso de filtros de comprimento 2 e de um processo de decimação por 2, uma possível forma de distribuição da matriz entre os p processadores pode ser a seguinte:

- 2^k linhas para os processadores p_j, $j = 0, \ldots,$ p - 2;
- N - 2^k (p - 1) linhas para o processador p_{p-1}, sendo N a dimensão da matriz e k o valor que minimiza a seguinte expressão:

$$(N - 2^k (p - 1)) - 2^k \tag{11}$$

Nesse procedimento de distribuição da matriz, o valor de k, que indica o número de níveis no método, é limitado pela menor quantidade de linhas enviada para um processador. Esse esquema nem sempre produz bons resultados, pois, dependendo do valor de N e do número p de processadores disponíveis, a diferença entre o número de linhas no último processador e o número de linhas nos demais pode ser muito grande, comprometendo, de forma significativa, o desempenho do método. Uma alternativa simples é completar a dimensão da matriz original A^h, gerando uma nova matriz $\overline{A}^h$ da forma

$$\overline{A}^h{}_{n \cdot n} = \begin{bmatrix} \overline{A}^h{}_{N \cdot N} & \\ & I_{m \cdot m} \end{bmatrix} \tag{12}$$

na qual $n = p2^k$ e $I_{m \cdot m}$ é a matriz identidade de dimensão $m = n - N$. Essa alternativa, no entanto, não é viável se o valor de m for muito grande, pois, apesar de permitir o mesmo número de linhas para todos os processadores, a quantidade de elementos da matriz no último processador é relativamente pequena. Uma alternativa mais eficiente, que faz uma distribuição homogênea da matriz entre os processadores, será apresentada na próxima seção.

3.1 Nova estratégia de distribuição

Como mencionado, a dimensão das matrizes que surgem do MEF não é, na maioria dos casos, igual a uma potência inteira de 2. Assim, a aplicação dos algoritmos tradicionais de paralelização da DWT é, em muitas situações, inadequada ao contexto do WAMG. Nesse caso, uma estratégia mais apropriada é a distribuição homogênea da matriz (mesmo número de linhas) entre os processadores disponíveis, principalmente porque o balanceamento de carga é um ponto crítico da eficiente execução do programa paralelo.

Esta seção propõe uma nova estratégia de distribuição da matriz, buscando obter paralelização eficiente do WAMG para um número p qualquer de processadores e para matrizes de qualquer dimensão n.

A estratégia proposta consiste em dividir, de forma homogênea, as linhas da matriz entre os processadores disponíveis. O artifício principal dessa abordagem é empregado no momento da construção da hierarquia de matrizes em cada processador ou, mais precisamente, quando cada um deles aplica a transformada discreta *wavelet* unidimensional nas colunas da matriz. Como cada coluna da matriz é dividida igualmente entre os processadores, cada um armazenará n_l / p elementos da coluna no nível l, denotando por n_l a dimensão da matriz nesse nível. Assim, os processadores irão envolver $\max_k 2^k < n_l / p$ elementos nos cálculos, em cada nível do processo, mantendo inalterados os demais n_l / p - $\max_k 2^k$ elementos para o próximo nível. Os elementos mantidos em cada nível são transformados nos níveis seguintes,

levando-se em consideração o número de linhas da matriz armazenado em cada processador, no nível correspondente.

Como antes, um exemplo simples pode ser usado para ilustrar essas operações. Para tal, considere um vetor coluna de dimensão $n = 10$ e $p = 2$ processadores. Suponha também o uso de filtros de comprimento 2 e de um processo de decimação por 2. No primeiro nível (l=1), cada processador aplica a transformação em

$$4 \left(= \max_k 2^k < \frac{10}{2} \right)$$

elementos, mantendo o restante para o segundo nível da transformada. O número de linhas da matriz armazenada nos processadores, nesse caso, é igual a 3. Assim, novamente, um elemento é mantido para o próximo nível.

Finalmente, no nível 3, o elemento que havia sido mantido nas etapas anteriores é envolvido na transformação. Esses três níveis da transformada estão ilustrados na Figura 1.

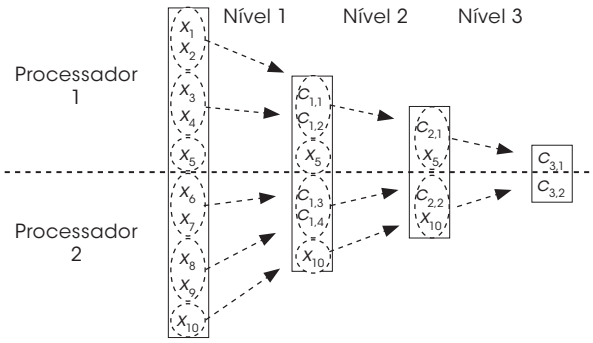

Figura 1: Exemplo de uma transformada Wavelet em paralelo com 3 níveis
Fonte: Os autores.

Uma alternativa viável seria envolver, no nível k da transformação, todos os elementos mantidos do nível anterior. Assim, no exemplo acima, o elemento x_{10} mantido no nível 1 seria utilizado no nível 2, enquanto o elemento c_3^1 permaneceria inalterado para o próximo nível.

É importante observar que não ocorre, como nas abordagens tradicionais, comunicação entre os processadores para filtros de ordem 2. Além disso, a única restrição sobre n e p é que n seja o múltiplo de p. Esse resultado é alcançado pelo procedimento ilustrado em (12), especialmente para baixos valores de p.

4 Resultados numéricos do PWAMG

Esta seção apresenta os resultados da aplicação do método PWAMG na resolução da equação de Poisson em um quadrado unitário com condições de contorno de Dirichlet. Semelhante ao que foi feito em Pereira, Nabeta (2007), considerou-se, para esse problema, uma aproximação por Diferenças Finitas representada pelo estêncil padrão de 5 pontos,

$$L_h^{(5)} = \frac{1}{h^2} \begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix}_h \qquad (13)$$

Em todos os resultados reportados aqui, a seguinte notação foi empregada:

- np: número de processadores utilizados;
- t_s: tempo gasto na fase de solução;
- t_m: tempo gasto na fase de montagem;
- N: número de iterações necessárias para que o vetor resíduo $\mathbf{r}$ satisfaça a relação $\|\mathbf{r}\|/\|\mathbf{r}_0\| \leq 10^{-6}$;
- EQ: dimensão do sistema linear correspondente.

Todos os resultados foram obtidos utilizando o modelo de programação com troca de mensagens (usando MPI-2), em uma arquitetura paralela de *cluster* de PCs (PoPC-*Pile-of-PCs*), conectados por meio de uma rede *Gigabit* e dotada do

sistema operacional Linux Fedora Core 6, como ilustrado esquematicamente na figura 2 e descrito detalhadamente em (PALIN, 2007). Foi utilizado um *cluster* homogêneo com quatro máquinas AMD Athlon(tm) XP 2400+, com 1.99 GHz e 1.0 GB RAM.

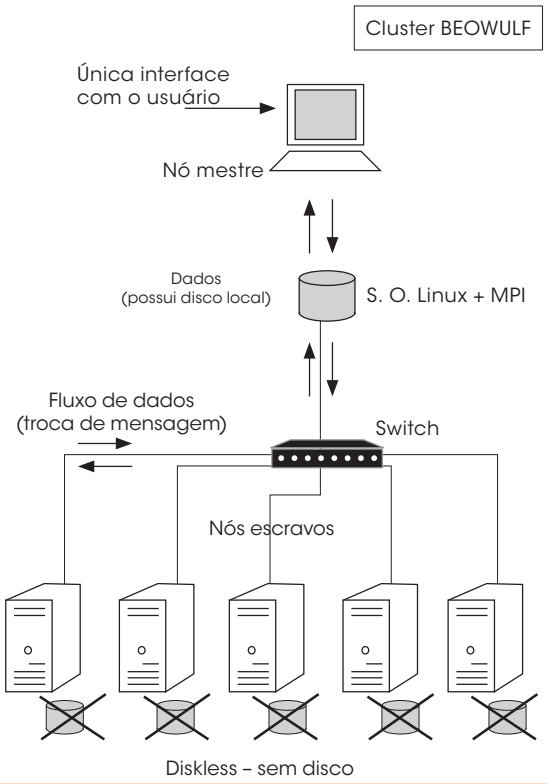

Figura 2: Representação da arquitetura paralela utilizada

Fonte: Os autores.

Apenas filtros de comprimento 2 foram utilizados no PWAMG. Os tempos indicados nas tabelas estão em segundos e são expressos na forma t_1/t_2, sendo t_1 o tempo gasto estritamente no processamento, alcançado por meio do uso da função *clock*() da linguagem C, e t_2, o tempo total gasto na operação, que inclui tempo de processamento e de comunicação entre os processadores, obtido da função *MPI_Wtime*().

Os resultados do PWAMG para o problema teste são apresentados na Tabela 1. Foram testadas algumas discretizações com diferentes números de pontos, com o intuito de observar o comportamento do PWAMG em relação ao aumento da dimensão de um mesmo problema.

Tabela I: Resultados do PWAMG para diferentes discretizações

EQ	*np*	t_m	t_s	*N*
64^2	1	0,04	0,15	3
	2	0,02/0,03	0,06/0,14	3
		0,02/0,02	0,02/0,14	
	4	0,01/0,01	0,04/0,17	3
		0,01/0,01	0,01/0,17	
		0,01/0,01	0,02/0,17	
		0,01/0,01	0,01/0,17	
128^2	1	0,15	0,65	3
	2	0,08/0,11	0,23/0,35	3
		0,08/0,08	0,14/0,35	
	4	0,04/0,09	0,20/0,33	3
		0,03/0,03	0,11/0,32	
		0,03/0,03	0,10/0,32	
		0,03/0,03	0,08/0,32	
256^2	1	0,63	2,70	3
	2	0,32/0,36	1,09/1,46	3
		0,30/0,30	0,77/1,45	
	4	0,17/0,24	0,90/1,36	3
		0,16/0,16	0,49/1,36	
		0,14/0,14	0,55/1,36	
		0,13/0,14	0,52/1,35	
512^2	1	2,57	10,76	3
	2	1,32/1,46	4,58/5,17	3
		1,25/1,25	3,61/5,13	
	4	0,73/0,88	3,42/5,07	3
		0,69/0,69	2,28/5,01	
		0,59/0,59	2,36/4,99	
		0,60/0,60	2,27/4,98	
1024^2	1	10,54	43,68	3
	2	5,38/5,70	18,05/20,71	3
		5,12/5,12	13,76/20,56	
	4	2,95/3,22	13,23/19,83	3
		2,83/2,83	9,60/19,73	
		2,40/2,40	8,34/19,65	
		2,40/2,40	9,24/19,73	

Fonte: Os autores.

5 Considerações finais

Uma vantagem importante do PWAMG, com filtros de ordem 2, é a ausência de comunicação entre os processadores na fase de montagem, o que pode ser constatado, observando-se que os tempos t_1 e t_2, para a etapa de montagem, são os mesmos em praticamente todos os casos. Essa característica do método é particularmente importante quando se deseja resolver sistemas extremamente grandes, nos quais a construção do método é a etapa que apresenta o custo computacional mais elevado. No que diz respeito aos tempos relacionados à etapa de solução, pode-se observar que, nos problemas maiores, os ganhos foram mais significativos. É preciso lembrar, no entanto, que a principal vantagem da programação paralela é a capacidade de resolver problemas cujo tamanho impede a resolução de forma seqüencial.

Duas outras características importantes do PWAMG podem ser observadas nos resultados da Tabela I. A primeira diz respeito à independência da convergência em relação ao espaçamento h entre os pontos da malha, ou seja, o aumento do tamanho do problema não modificou o número de iterações, exatamente como previsto na teoria dos métodos multigrid (TROTTENBERG, OOSTERLEE, SCHULLER, 2001). A segunda característica relaciona-se à escalabilidade do método PWAMG. É possível observar, na tabela, que o aumento do tempo gasto na etapa de solução foi diretamente proporcional à expansão da dimensão dos problemas testados.

Por fim, deve-se destacar que as dimensões das matrizes utilizadas nos testes permitem um bom balanceamento de carga entre os processadores, o que é fundamental para o desempenho adequado do método. Quando a estratégia descrita na seção 3.1 for utilizada, espera-se obter resultados qualitativos similares aos alcançados neste trabalho, na resolução de problemas de qualquer dimensão.

A new wavelet-based Algebraic Multigrid Method: Part II – Parallel Algorithm

In this work, it is presented a new parallel wavelet-based algorithm for the Algebraic Multigrid Method (PWAMG). A variation of the standard parallel implementation of discrete *wavelet* transforms is used in the construction of a hierarchy of matrices and of intergrid transfer operators for Algebraic Multigrid. The PWAMG method has been tested as a parallel solver for the two dimensional Poisson equation, for different numbers of finite difference mesh nodes and comparisons are made with the sequential version of this method.

Key words: Algebraic multigrid method. Discrete wavelet transform. Distributed memory machines. Parallel programming.

Referências

CHANG, Q.; HUANG, Z. Efficient Algebraic Multigrid Algorithms and Their Convergence, *SIAM J. Sci. Comput*. v. 24, n. 2, p. 597-618, 2002.

CLEARY, A. J.; FALGOUT, R. D.; HENSON, V.; JONES, J. E., Coarse-grid selection for parallel algebraic multigrid, In: PROCEEDINGS OF THE 5º INTERNATIONAL SYMPOSIUM ON SOLVING IRREGULARLY STRUCTURED PROBLEMS IN PARALLEL, LECTURE NOTES IN COMPUTER SCIENCE, n. 1457, Springer-Verlag, New York, p. 104-115, 1998.

FORD, J. M.; CHEN, K.; FORD, N.J. Parallel implementation of fast wavelet transforms, In: NUMERICAL ANALYSIS REPORT, n. 39, Manchester University, 2001.

HAASE, G.; KUHN, M.; REITZINGER, S. Parallel algebraic multigrid methods on distributed memory computers, *SIAM J. Sci. Comput*., v. 24, p. 410-427, 2002.

HENSON, V. E.; YANG, U. M. BoomerAMG: a Parallel Algebraic Multigrid Solver and Preconditioner. *Applied Numerical Mathematics*, v. 41, n. 1, p. 155-177, 2002.

KRECHEL, A.; STÜBEN, K. Parallel Algebraic Multigrid Based on Subdomain Blocking, *Parallel Computing*, v. 27, p. 1009-1031, 2001.

PALIN, M. F. *Técnicas de computação paralela para o cálculo de campos eletromagnéticos no regime harmônico*, 2007. Tese de Doutorado – Escola Politécnica da Universidade de São Paulo, São Paulo, 2007.

PEREIRA, F. H.; VERARDI, S. L. L.; NABETA, S. I. A Wavelet-based Algebraic Multigrid preconditioner for sparse linear systems, *Appl. Math. and Comput.*, v. 182, n. 2, p. 1098-1107, 2006.

PEREIRA, F. H.; NABETA, S. I. Uma nova abordagem baseada em *wavelets* para o método Multigrid Algébrico: Parte I – Algoritmo Seqüencial, *Exacta*, v. 5, n. 1, 2007. A aparecer.

STERCK, H.; YANG, U. M.; HEYS, J. J., Reducing complexity in parallel algebraic multigrid preconditioners, *SIAM J. on Matrix Analysis and Appl.*, v. 27, n. 4, p. 1019-1039, 2005.

TROTTENBERG, U.; OOSTERLEE, C. W.; SCHULLER, A. *Multigrid*, Academic Press, London, 2001.

Recebido em 18 set. 2007 / aprovado em 3 out. 2007

Para referenciar este texto

PEREIRA, F. H.; NABETA, S. I. Uma nova abordagem baseada em *wavelets* para o método Multigrid Algébrico: Parte II – Algoritmo Paralelo. *Exacta*, São Paulo, v. 5, n. 2, p. ???-???, jul./dez. 2007.