SOLUÇÃO ITERATIVA DE SISTEMAS DE EQUAÇÕES DO MÉTODO DOS ELEMENTOS FINITOS NO COMPUTADOR IBM 3090

Álvaro L.G. Coutinho, José L.D. Alves, Luiz Landau e Nelson F.F. Ebecken
Programa de Engenharia Civil
COPPE/Universidade Federal do Rio de Janeiro
Caixa Postal 68506
21945 - Rio de Janeiro, Brasil

RESUMO

Este trabalho apresenta uma estratégia de implementação do Método dos Gradientes Conjugados Precondicionado para a solução iterativa de sistemas de equações do Método dos Elementos Finitos, utilizando as facilidades para processamento vetorial disponíveis na arquitetura do computador IBM 3090. Os resultados numéricos obtidos evidenciam a eficiência computacional da estratégia adotada.

ABSTRACT

In this work the implementation of Preconditioned Conjugate Gradients for the solution of finite element equations utilizing the vector facilities of IBM 3090 is presented. The numerical results obtained stress the remarkable computer ef - fectiveness of the present implementation.

1. INTRODUÇÃO

A aplicação do processo de discretização espacial do Método dos Elementos Finitos (MEF) às equações da mecânica dos sólidos, estruturas ou fluidos conduz a um sistema de equações semi-discretas de evolução, em geral não lineares. Utilizando-se uma aproximação no tempo implícita, torna-se necessária a solução de um sistema não trivial de equações. Caso o problema seja linear, o custo da análise é dominado pela solução, em cada instante de tempo, de um sistema de equações algébricas. Em caso contrário,os custos também são dependentes da estratégia empregada no processo iterativo de verificação do equilíbrio, da atualização das matrizes de elemento , do cálculo de resíduos, etc. Tradicionalmente,a solução dos sistemas de equações é efetuada através de um método direto de solução, explorando a esparsidade da matriz resultante |1|.

Entretanto, com o advento de computadores dotados de arquiteturas vetoriais como por exemplo, o CDC Cyber 205, CRAY-1, CRAY-2, IBM 3090 ou paralelas do tipo memória compartilhada, tais como o CRAY X-MP, Alliant FX, ou até mesmo do tipo hipercúbica (JPL/CalTech Mark III,Intel iPSC/2, NCUBE/10) a utilização de métodos iterativos de solução vem sendo cuidadosamente reestudada.

Pesquisas recentes apontam o método dos Gradientes Conjugados Precondicionados e o Método de Lanzcos Precondicionado como os mais apropriados e de melhor desempenho em uma série de diferentes máquinas, como por exemplo o CRAY X-MP |2|, Alliant FX/80 e CRAY-2 |3|, JPL/CalTech MARK III |4|. É interessante observar também que estes métodos, para uma certa classe de problemas bi e tri-dimensionais envolvendo um grande número de equações (> 10000), mostraram-se competitivos diante dos métodos diretos de solução,conforme pode-se verificar em |5,6,7,8|.

Desta forma, este trabalho tem por objetivo a implementação do método dos Gradientes Conjugados Precondicionados (GCP) para a solução iterativa de sistemas de equações do MEF explorando as facilidades de processamento vetorial do computador IBM 3090/VF. A seção seguinte apresenta de forma sucinta o método GCP. A seção 3 procura discutir a implementação deste método de forma a explorar ao máximo as potencialidades da máquina disponível. A seção 4 apresenta os resultados numéricos obtidos em um problema modelo com um número crescente de equações (2500 até 40000). Finalmente, na seção 5 as principais conclusões deste trabalho são discutidas e alguns desenvolvimentos futuros são sugeridos.

2. MÉTODO DOS GRADIENTES CONJUGADOS PRECONDICIONADOS

O método dos Gradientes Conjugados Precondicionados (GCP), sumarizado na Tabela 1) é um algoritmo iterativo de solução de sistemas de equações algébricas, cujos detalhes encontram-se em |9|.

$j=0$
$\underset{\sim}{r}_o = \underset{\sim}{b} - \underset{\sim}{\underset{\sim}{A}}\ \underset{\sim}{u}_o$
$\underset{\sim}{p}_o = \underset{\sim}{z}_o = \underset{\sim}{\underset{\sim}{C}}^{-1}\ \underset{\sim}{r}_o$
para $j = 0,1,2,\ldots$ faça

$a_j = \underset{\sim}{r}_j \cdot \underset{\sim}{z}_j / \underset{\sim}{p}_j \cdot \underset{\sim}{\underset{\sim}{A}}\ \underset{\sim}{p}_j$
$\underset{\sim}{u}_{j+1} = \underset{\sim}{u}_j + a_j\ \underset{\sim}{p}_j$
$\underset{\sim}{r}_{j+1} = \underset{\sim}{r}_j - a_j\ \underset{\sim}{p}_j$
se $||\underset{\sim}{r}_{j+1}|| \leq RTOL\,||\underset{\sim}{r}_o||$ fim
$\underset{\sim}{z}_{j+1} = \underset{\sim}{\underset{\sim}{C}}^{-1}\ \underset{\sim}{r}_{j+1}$
$c_j = \underset{\sim}{r}_{j+1} \cdot \underset{\sim}{z}_{j+1} / \underset{\sim}{r}_j \cdot \underset{\sim}{z}_j$
$\underset{\sim}{p}_{j+1} = \underset{\sim}{z}_{j+1} + c_j\ \underset{\sim}{p}_j$
fim

Tabela 1-Método dos Gradientes Conjugados Precondicionados.

Neste algoritmo, $\underset{\sim}{A}$ é uma matriz simétrica positiva definida em banda de ordem n, construída a partir da contribuição de cada elemento finito; b é um vetor n-dimensional formado a partir das condições de contorno não essenciais; u são as incógnitas nodais; r é o vetor de resíduos (gradiente); p e z as direções de busca; e C é a matriz de precondicionamento. Neste trabalho optou-se pelo precondicionamento de Jacobi, ou seja, C = diag A. O algoritmo GCP necessita de 4 arranjos n-dimensionais (r,p,z,u) além das matrizes A e C. Em relação às operações em ponto flutuante, estas são basicamente produtos escalares, operações com tríades de vetores e um produto matriz-por-vetor, uma vez que a solução $\underset{\sim}{z}_{j+1} = \underset{\sim}{C}^{-1}\ \underset{\sim}{r}_{j+1}$, no caso é trivial. O produto matriz-por-vetor pode ser computado, valendo-se da estrutura local do MEF, no nível de elemento, através da decomposição,

$$\underset{\sim\sim}{Ap}_j = \sum_{i=1}^{NEL} \hat{\underset{\sim}{A}}^i\ \hat{\underset{\sim}{p}}^i_j \qquad (1)$$

onde $\hat{\underset{\sim}{A}}^i$ é a i-ésima matriz de elemento, $\hat{\underset{\sim}{p}}^i_j$ é a componente do vetor $\underset{\sim}{p}_j$ relacionada a este elemento e NEL é o número total de elementos finitos. Esta decomposição caracteriza o esquema de solução elemento-por-elemento (EPE). Para a implementação de esquemas EPE, uma estrutura de dados especial para as operações de agrupamento/espalhamento de informações do nível global para o nível local (elementos), torna-se necessária |6,7|. Considera-se alcançada a convergência quando a norma euclideana relativa dos resíduos for inferior a uma tolerância pré-especificada (RTOL).

3. IMPLEMENTAÇÃO NO COMPUTADOR IBM 3090

O computador IBM 3090 200E/2VF instalado no Centro Computacional da PETROBRÁS tem a sua arquitetura com as seguintes características. Dois processadores vetoriais com um ciclo de 18.5 na no-segundos cada, capazes de executar operações lógicas ou aritméticas em até 128 conjuntos de operandos com uma única instrução de máquina. A arquitetura do processador vetorial provê 63 novas instruções de máquina com 171 novos códigos de operação, sendo 104 códigos para operações de ponto flutuante do tipo real. Uma memória central de 64 Mbytes com uma expansão de memória (Expanded Storage Facility) de 128 Mbytes que permite a transferência de dados para a memória principal em blocos de 4096-bytes com uma taxa de 50 Mbytes/s. Encontram-se disponíveis no computador IBM 3090 dois sistemas operacionais:VM/A para processamento interativo e o MVS /Extended Architeture (MVS/XA), utilizado principalmente, para processamento por lotes. O sistema operacional MVS/XA permite a utilização do compilador VS FORTRAN |10 |, capaz de gerar um código que faz uso tanto da velocidade do processador vetorial (Vector Facility) quanto da facilidade multitarefa (MultiTasking Facility) para distribuir a carga computacional entre os dois processadores disponíveis. Neste trabalho será dada atenção especial à utilização de apenas um processador vetorial do IBM 3090.

O problema maior na vetorização através do uso do compilador VS FORTRAN é evitar estruturas do tipo DO contendo dependências e/ou recorrências. Ou seja, durante o processamento de uma instrução do tipo DO não é possível utilizar-se o resultado de uma operação em um comando de atribuição subseqüente. Outros aspectos de programação podem inibir a vetorização de instrução tipo DO, como operações de entrada e saída, chamadas de subrotinas, referências à função externa que não as intrísecas do compilador, etc . Maiores detalhes sobre estas restrições podem ser encontrados em |11|.

Com respeito à vetorização do algoritmo GCP dado na Tabela 1, pode-se verificar que este é composto basicamentede quatro tipos de operações. Produtos escalares, solução de um sistema de equações trivial, operações em tríades de vetores, e um produto matriz-por-vetor. Para os três primeiros tipos de operação, a vetorização é imediata, pois são basicamente estruturas do tipo DO-para-todas-as-equações, sem nenhuma interrupção. Em relação à operação de produto matriz por-vetor, deve-se observar que na estratégia EPE utilizada, a matriz nunca é formada explicitamente, sendo necessário efetuar este produto a partir das contribuições dos diversos elementos finitos. Além disso, verifica-se que o custo desta operação domina o custo da solução do sistema de equações através do algoritmo GCP . Portanto, de forma a explorar ao máximo as potencialidades do processador vetorial para efetuar esta operação, a estrutura de dados para os cálculos do MEF deve ser convenientemente modificada. Ao invés de computar seqüencialmente a contribuição de cada elemento finito para o produto matriz-por-vetor, é mais eficiente o processamento de grupos de elementos. Entretanto, uma vez que de forma geral um ponto nodal é comum a diversos elementos, estabelece-se uma dependência, que além de inibir a vetorização , resulta na perda da contribuição de alguns elementos, se estes forem processados em sua ordenação natural. Esta dependência pode ser evitada reordenando-se os elementos em uma seqüência tal que não existam nós comuns entre os elementos pertinentes a esta seqüência. Desta forma , os elementos são reordenados em um conjunto de blocos de elementos disjuntos através de um algoritmo de coloração, cujos detalhes se encontram em |12|. A Figura 1a apresenta um exemplo de um domínio retangular simples subdividido em 16 elementos com sua ordenação natural enquanto que a Figura 1b mostra o domínio com os elementos reordenadors indicando o bloco (COR) a que pertencem.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

(a)

1	2	1	2
3	4	3	4
1	2	1	2
3	4	3	4

(b)

Figura 1-Reordenação dos Elementos Finitos para o Processamento Vetorial.

A contribuição de cada bloco de elementos de mesma cor para o produto matriz-por-vetor do algoritmo GCP é computada em paralelo no processador vetorial do IBM 3090. Os conjuntos de blocos de elementos de mesma cor são processados seqüencialmente.

4. RESULTADOS NUMÉRICOS

Selecionou-se como problema modelo a solução da equação de Laplace, $k\nabla^2 u=q$, no domínio $\Omega = |0,10|x|0,10|$ com condições de contorno de Dirichlet e considerando-se respectivamente k=2 e q=2.4. Diversas análises foram efetuadas para um número crescente de graus de liberdade (N), em malhas regulares. Nestas análises discretizou-se o problema por meio de elementos finitos isoparamétricos com 4 pontos nodais, derivados de formulação via método de Galerkin. A Tabela 2 apresenta um sumário das principais características das análises efetuadas, indicando o número de equações, a largura de banda (LB) e o número de iterações (ITER) necessárias, para as dimensões do problema variando entre 2500 e 40000 equações. Estes resultados foram obtidos considerando-se uma tolerância para a norma relativa de resíduos de 10^{-3}.

MALHA	N	LB	ITER
50 x 50	2500	52	64
100 x 100	10000	102	122
150 x 150	22500	152	179
200 x 200	40000	202	234

Tabela 2 - Características dos Problemas Analisados.

Primeiramente, procurou-se determinar o tamanho ótimo dos blocos de elementos de mesma cor. Processaram-se diversas análises para blocos de respectivamente 64, 128 e 256 elementos, identificados como BL(64), BL(128), BL(256). A Figura 2 mostra a taxa de utilização do processador vetorial, calculada como a razão entre o tempo de execução no processador vetorial e o tempo total de execução.

Pode-se notar nesta Figura que a taxa de utilização do processador vetorial é maior para BL (256). A Figura 3 apresenta os tempos de processamento total para cada um dos casos analisados, enquanto que a Figura 4 mostra o custo de cada execução.

Examinando-se as Figuras 3 e 4, verifica-se que os menores tempos de processamento e os menores custos foram obtidos para as análises em que o tamanho do bloco é de 128 elementos finitos.Tal valor é idêntico ao número máximo de conjuntos de operandos executáveis em uma única instrução de máquina na arquitetura do processador vetorial do IBM 3090. Os experimentos numéricos evidenciaram que as potencialidades do processador vetorial devem ser usadas criteriosamente, pois caso contrário podem acarretar em uma perda de eficiência computacional considerável. Para as análises com o tamanho de bloco ideal, observou se que em média, o algoritmo GCP utilizou 57% do tempo total de processamento, o algoritmo de reordenação 7% e a leitura e geração das matri-

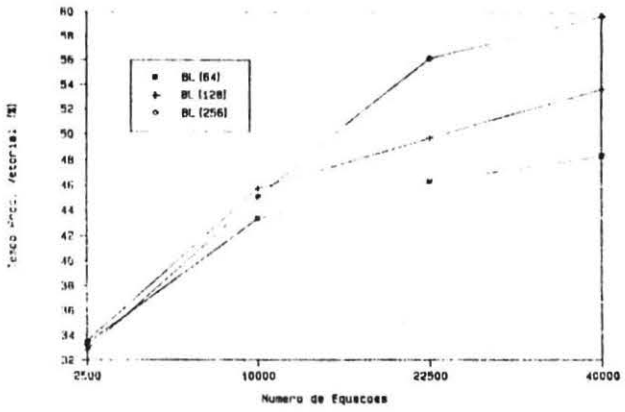

Figura 2-Taxa de Utilização do Processador Vetorial.

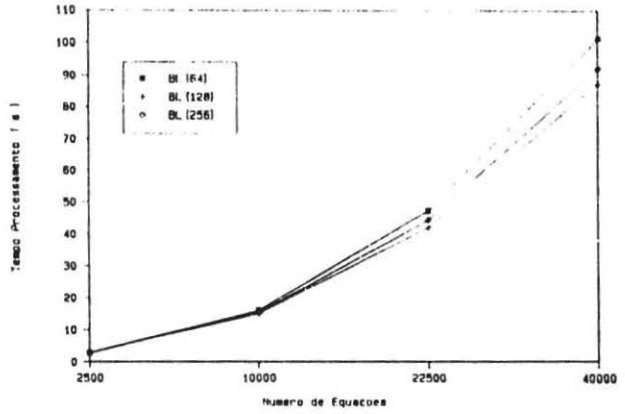

Figura 3 - Tempo de Processamento Total.

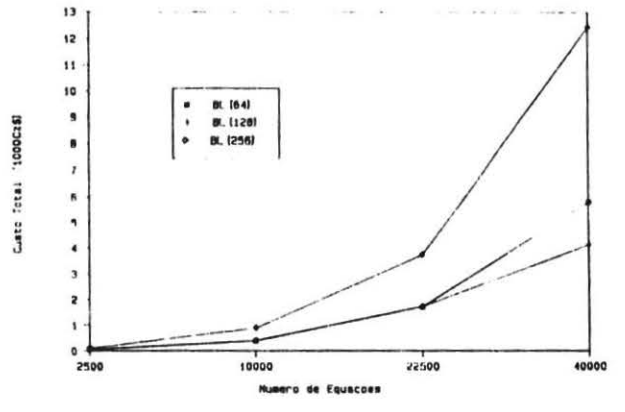

Figura 4 - Custo de Execução.

zes de elemento e do vetor de termos independentes o restante. Conforme esperado, verificou-se que o produto matriz-por-vetor domina o tempo de solução, sendo responsável por uma parcela da ordem de 40% do tempo total de processamento. Tal fração representa aproximadamente 80% do tempo de solução do sistema de equações pelo algoritmo GCP. Cabe ressaltar que nenhum esforço foi efetuado no sentido de vetorização da geração das matrizes de elemento, para permitir comparações.

Em seguida, procurou-se avaliar o ganho obtido com a vetorização no esquema proposto, em relação a um código vetorial gerado pelo compilador VS FORTRAN onde o produto matriz-por-vetor é computado a partir da ordenação natural dos elementos, identificado por NAT. O código NAT foi gerado a partir de um programa fonte otimizado para computadores com arquitetura escalar |6|. A vetorização deste código foi efetuada pelo compilador VS FORTRAN, sem que modificação alguma fosse introduzida no programa fonte original. A Figura 5 apresenta a taxa de utilização do processador vetorial para os códigos BL(128) e NAT.

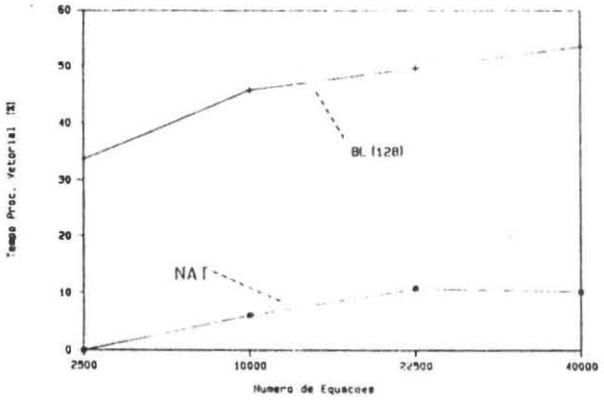

Figura 5 - Taxa de Utilização do Processador Vetorial.

Pode-se observar nesta Figura que o código BL (128) utiliza muito mais eficientemente o processador vetorial do que o código NAT. Este permaneceu em uma taxa de utilização do processador vetorial em torno de 10%, enquanto que o código BL(128) apresentou uma taxa de 54% para 40000 equações. Este acréscimo significativo deve-se exclusivamente à vetorização completa do produto matriz-por-vetor presente no código BL (128).

Finalmente, procurou-se avaliar o ganho no tempo total de processamento e a redução de custo computacional dos códigos vetorizados BL(128) e NAT em relação ao código escalar. As Figuras 6 e 7 apresentam respectivamente estes resultados. O ganho em tempo de processamento foi calculado como a razão entre os tempos totais de processamento do código escalar e do código vetorial, enquanto que a redução de custo como a razão entre custo do código vetorial e o custo do código seqüencial.

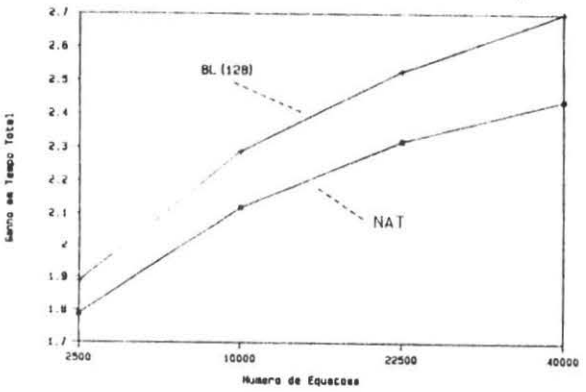

Figura 6 - Ganho em Tempo Total de Processamento.

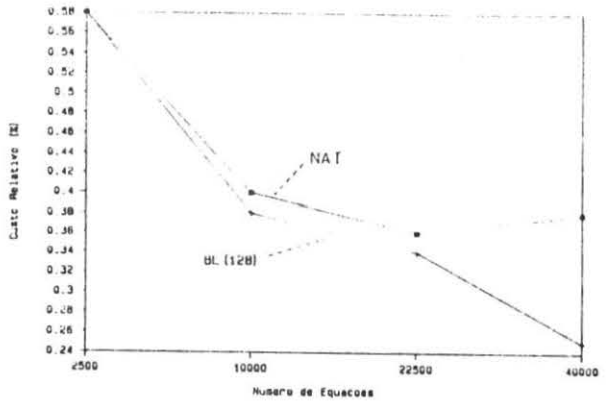

Figura 7 - Redução de Custo Computacional Total.

Dos resultados apresentados nestas Figuras observa-se que o código BL(128) apresentou um ganho em tempo total de processamento máximo de 2.70 e uma correspondente redução do custo computacional de 0.25 enquanto o código NAT obteve respectivamente 2.44 e 0.38. Desta forma, verifica-se que o código NAT obteve um ganho máximo aproximadamente 20% inferior e uma redução de custo máxima 30% superior ao código BL (128). Quanto ao ganho em tempo de processamento para somente o Método GCP, os resultados obtidos encontram-se na Figura 8.

Pode-se observar que o código BL(128) apresentou um desempenho significativamente superior ao código NAT, obtendo um ganho máximo de 4.54. Estes resultados evidenciam a eficiência computacional do esquema de solução adotado.

5. CONCLUSÕES

Este trabalho apresentou uma estratégia de im -

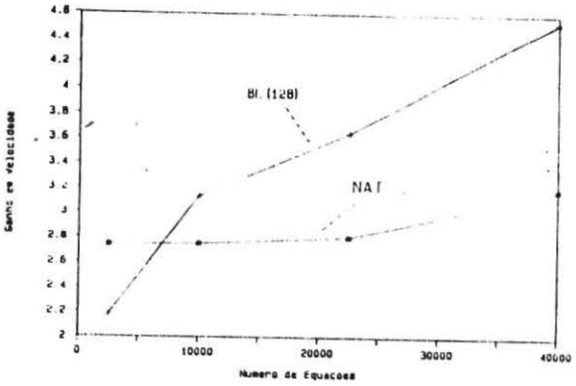

Figura 8 - Ganho em Tempo de Processamento para o Método GCP.

plementação do Método dos Gradientes Conjugados Precondicionado (GCP) para a solução iterativa de sistemas de equações do MEF, através do esquema elemento-por-elemento, valendo-se das facilidades para processamento vetorial disponíveis na arquitetura do computador IBM 3090. Através da utilização de um algoritmo de coloração para reordenar os elementos finitos em blocos disjuntos, conseguiu-se vetorizar completamente o método GCP. Os resultados numéricos obtidos evidenciam a eficiência computacional da estratégia adotada, que resultou numa redução em tempo total de processamento máximo em torno de 2.7 e de 25% no custo computacional em relação ao código escalar. Cabe observar que o algoritmo de coloração empregado representou em média 7% do tempo total de processamento e que este algoritmo pode ser utilizado como um pré-processador, uma vez que depende exclusivamente da topologia da malha de elmentos finitos adotada. Por fim, cabe ressaltar que nenhum esforço foi efetuado no sentido da vetorização do algoritmo para o cálculo das matrizes de elemento. Entretanto, a inclusão destes algoritmos, como por exemplo o descrito em |13|, é imediata e deve representar um fator a mais para se obter um melhor desempenho computacional.

Este trabalho foi realizado sob o Convênio de Cooperação em Engenharia Offshore entre a PETROBRÁS S.A. e a COPPE/UFRJ. A presente pesquisa faz parte do estudo e implementação de Técnicas Computacionais Avançadas para a análise de fundações de plataformas marítimas, desenvolvido no Setor de Desenvolvimento e Métodos do Centro de Pesquisas da Petrobrás (SEDEM/DIPREX/CENPES) |14|.

6. REFERÊNCIAS

| 1| Hughes, T.J.R. (1987) "The finite element method-linear static and dynamic finite element analysis", Prentice-Hall Int., Englewood Cliffs, NJ.

| 2| Hughes, T.J.R.; Ferencz, R.M. e Hallquist, J.O. (1987) "Large-scale vectorized Implicit calculations in solid mechanics on a CRAY X-MP/48 utilizing EBE preconditioned conjugate gradients", Computer Methods in Applied Mechanics and Engineering, Vol. 61, pp. 215-248.

| 3| Fahrat C. e Crivelli L. (1987) "A general approach to nonlinear FE computations on shared memory multiprocessors", Center for Space Structures and controls, University of Colorado, Bouder, USA, Report CU-CSSC - 87-09.

| 4| Nour-Omid, B.; Raefsky, A. e Lyzenga, G. (1987) "Solving finite element equations on concurrent computers",Parallel Computations and Their Impact on Mechanics, Edt. by A.K. Noor, ASME, AMD-Vol. 86, pp. 209-228.

| 5| Winget, J.M. e Hughes, T.J.R. (1985) "Solution algorithms for nonlinear transient heat conduction analysis employing element-by-element iterative strategies", Computer Methods in Applied Mechanics and Engineering, Vol. 52, pp. 711-815.

| 6| Coutinho, A.L.G.A.; Alves, J.L.D; Landau, L. e Ebecken, N.F.F. (1986) "Um procedimento elemento-por-elemento para a solução de grandes sistemas de equações do MEF pelo método dos gradientes conjugados", Proc. VII Congresso Latino-Americano Sobre Métodos Computacionais em Engenharia, São Carlos, SP, Vol. 1, pp. 243-258.

| 7| Coutinho, A.L.G.A.; Alves, J.L.D.; Landau, L.; Lima, E.C.P. e Ebecken, N.F.F. (1987) "On the application of an element-by-element Lanczos solver to large offshore structural engineering problems", Computer and Structures, Vol. 27, pp. 27-37.

| 8| Carey, G.F. e Jiang, B.N. (1986) "Element-by-element linear and nonlinear solution schemes", Communications in Applied Numerical Methods, Vol.2, 145-154.

| 9| Golub, G.H. e Van Loan, C.F. (1983)"Matrix computations", John Hopkins University Press, USA.

|10| IBM VS FORTRAN Version 2 (1986)"Programming Guide", IBM Corporation SC26-4222-0.

|11| Dubrulle, A.A., Scarborough, R.G. e Kolsky, H.G. (1985) "How to write good vectorizable FORTRAN", IBM Palo Alto Scientific Center , Report G320-3478.

|12| Berger, P.; Brouaye, P.; e Syre, J. (1982) "A mesh coloring method for efficient MIMD processing in finite element problems" , Proc. International Conference on Parallel Processing. Edts K.E. Batcher, W.C. Meilander e J.L. Potter, IEEE Computer Society Press, pp. 41-46.

|13| Silvester, D.J. (1988) "Optimizing finite element matrix calculations using the gene

ral technique of element vectorization" , Parallel Computing, Vol. 6, pp. 157-164.

|14| Coutinho, A.L.G.A., Costa, A.M., Alves, J. L.D., Landau, L. e Ebecken, N.F.F. (1988)- "Pile driving simulation and analysis by the finite element method", 3rd International Conference on the Application of Stress-Wave Theory on Piles, Editor Blugt. H. Fllenius, Ottawa, Canada, pp.197-207.