18 AL 17 DE NOVIEMBRE DEL 2000
PUNTA DEL ESTE - URUGUAY
XXIX JORNADAS SUDAMERICANAS DE INGENIERIA ESTRUCTURAL
1950 2000
JUBILEO
Prof. Julio Kicaldoni

O USO DO PROCESSAMENTO PARALELO EM PROBLEMAS DO MEF

Valério S. Almeida, João B. de Paiva

Departamento de Engenharia de Estruturas
Escola de Engenharia da Universidade de São Paulo
Caixa Postal 359 – CEP 13560-970 – São Carlos, SP, Brasil
e-mail: valerio@sc.usp.br, web page: http:// www.set.eesc.sc.usp.br
e-mail: paiva@sc.usp.br, web page: http:// www.set.eesc.sc.usp.br

SUMÁRIO

A crescente aplicação dos métodos numéricos discretos para análise de estruturas no campo da engenharia tem sido feita já há algumas décadas. As modelagens até então feita experimentalmente tem sido, a cada dia, substituídas por modelos numéricos teorizados e calibrados adequadamente. Esse uso volumoso dos métodos numéricos, destacando o Método dos Elementos Finitos (MEF) e o Método dos Elementos de Contorno (MEC), tem-se dado, principalmente, pelo impacto tecnológico dos computadores junto com a facilidade – em geral - que tais métodos apresentam para a sua sistematização em códigos computacionais.

Com o impacto provocado por computadores de arquitetura paralela à análise numérica, foram desenvolvidos procedimentos numéricos para este novo paradigma computacional voltados à resolução de sistemas de equações lineares.

Neste trabalho, então, apresenta-se um método que tem se destacado para a resolução do sistema em paralelo: *o método dos gradientes conjugados*. Sendo que foi aperfeiçoado um otimizador conhecido para melhorar sua perfomance: a Decomposição Incompleta de Cholesky. Desta maneira, mostra-se o exemplo - analisados pelo MEF - de treliça tridimensional em computação paralelo.

1 INTRODUÇÃO

A resolução do sistema linear do tipo

$$[K]\cdot\{U\}=\{F\} \qquad (1)$$

gerada pelo MEF, onde [K] é a matriz de rigidez simétrica e positiva definida, {U} o vetor de deslocamentos e {F} é o vetor das ações, pode ser resolvido pelo dois métodos conhecidos na literatura: método direto e iterativo.

Conforme [1], no início das aplicações do MEF, metade da década de 50, os algoritmos iterativos eram comumente utilizados. A utilização destes métodos requerem iterativamente a realização de um produto matriz-vetor, sendo que nesta matriz armazena-se apenas os valores diferentes de zero e sem alterá-los ao longo do método. Assim, para sistema esparsos - que são característicos do MEF - torna-se uma grande vantagem sua utilização, pois, este fator era imprescindível para a época em virtude de que os computadores tinham baixa capacidade de armazenamento de dados.

Mas, como desvantagem destes métodos iterativos, destaca-se a dificuldade de se estimar o tempo de convergência do sistema, já que a convergência do método
é fator dependente do número de condição (κ) [1]da matriz [K]. Isto faz com que, muitas vezes, este método alcance convergência em um número elevado de iterações (*ite*), sendo que isto torna o método ineficiente.

Para diminuir o número de iterações do método, pode-se, então, aplicar as técnicas de pré-condicionamento sobre o sistema originário dado pela equação (1). Este procedimento aplicado ao método é conhecido como métodos iterativos com pré-condicionadores. Estas técnicas aceleram a convergência do sistema, mas são procedimentos que dependem do tipo de problema a ser tratado, portanto, específicos e, portanto, é necessário se estudar um determinado pré-condicionador para acelerar cada conjunto de problemas.

Os métodos diretos têm como vantagens a estabilidade e por saber-se, *a priori*, o número de operações necessárias para a sua utilização. Este valor é da ordem de $O(\frac{1}{2}\cdot n\cdot m^2)$, onde *n e m* são, respectivamente, o número de graus de liberdade e o comprimento de semibanda. A estabilidade do método é garantida, pois, a convergência do método independe do número de condição e assim, não precisam de estudos complementares para sua aplicação.

Como maior desvantagem para o método direto, sabe-se que os elementos da matriz armazenada [K], são alterados ao longo do algoritmo. Assim, a esparsidade é perdida ao longo das operações, portanto, tendo-se que armazená-la por inteiro, ou seja, com dimensão $(n\cdot m)$. Acrescenta-se também que o número de operações necessárias para o método varia quadraticamente em (m), ou seja, sistemas onde o comprimento de banda é alto torna-se ineficiente seu uso.

Assim, pelos motivos apresentados anteriormente, é que no início das aplicações do MEF, em função dos computadores terem pequena capacidade de memória, fator este limitante, optou-se pela utilização dos métodos iterativos.

Mas, com o desenvolvimento dos componentes microeletrônicos, dando-se ênfase a maior capacidade de armazenagem de dados, a partir da metade da década de 60, com as desvantagens oferecidas pelos métodos iterativos, pela boa estabilidade apresentada nos métodos diretos, estes últimos começaram a ser empregados em substituição aos métodos iterativos para resolução dos problemas advindos do MEF.

Ressalta-se, finalmente, que isto só ocorreu porque os modelos estruturais implementados na época, geravam sistemas lineares que eram possíveis de serem armazenados nos computadores existentes e pelo tipo de arquitetura destas máquinas, basicamente seqüencial ou vetorial, os métodos diretos ofereciam maiores vantagens em comparação aos métodos iterativos.

2 RESOLUÇÃO DO SISTEMA LINEAR EM PROCESSAMENTO PARALELO

Com o início da utilização do processamento paralelo aplicado ao campo da engenharia, principalmente para a resolução de um sistema linear, os métodos iterativos começaram a se destacar.

Isto pelo fato que estes métodos possuem características adequadas a este tipo de arquitetura, como boa versatilidade para se armazenar os sistemas esparsos e o procedimento onde se faz, a cada iteração, o produto matriz-vetor, o qual possui vantagens para paralelização sem muita perda de eficiência. Além de que estes métodos realizam muitas operações sobre vetores, fator também que se beneficia para tais computadores pelo sistema de *pipeline*.

1 O numero de condição é um fator determinado pelo quociente entre o maior e o menor autovalor da matriz.

A boa versatilidade no tratamento de sistemas esparsos significa que para sistemas muito grandes, pode-se ao invés de se armazenar toda a matriz de rigidez em banda de dimensão $(n \cdot m)$, guarda-se - em vetores – apenas as posições diferentes de zero, já que a matriz [K] não irá ser alterada.

Em relação ao produto matriz-vetor, a vantagem oferecida pelo método se apresenta, principalmente, para aplicação em sistemas de arquiteturas de memória distribuída, já que pode-se dividir os dados da matriz entre os diversos processadores e se realizar o produto sob estes diferentes dados em cada processador local, tendo-se que apenas após este produto, atualizar globalmente o vetor resultante desta operação, trocando-se mensagens. Alguns trabalhos que comentam as vantagens de se aplicar os métodos iterativos para análise em processamento em paralelo são [2, 3, 4, 5]

A grande vantagem da implementação do método direto em computadores com memória compartilhada se dá no fato de que os diversos processadores podem realizar todas as instruções do método sob diferentes dados. Tem-se que apenas estabelecer pontos do código computacional onde deve-se colocar comandos de sincronização entre estes processadores. Evita-se, assim, o problema já tratado no capítulo 2, de *racing condition*. Destacam-se alguns trabalhos onde é aplicado o método direto para análise em computadores paralelos com memória compartilhada: [6, 7, 8, 9].

A grande desvantagem para paralelização do método direto aparece para implementação em computadores de memória distribuída. Neste tipo de arquitetura, os dados do sistema (1) são divididos entre os diversos processadores e durante a execução do algoritmo, em determinado momento, um ou vários processadores não realizam instruções com os seus dados armazenados, ou seja, ficam ociosos, não aproveitando, então, o potencial total do paralelismo neste método.

Um procedimento bastante adotado para otimizar resolução do sistema linear em processamento paralelo pelo método direto, tanto para computadores com memória compartilhada para se evitar o problema do *racing condition*, como para máquinas com memória distribuída, para se superar a questão da ociosidade, é utilizando as técnicas de subestruturação e decomposição em domínio.

O emprego de uma destas técnicas acarreta numa sensível redução do número de variáveis globais do sistema a ser resolvido, lembrando-se que em cada processador após a resolução dos pontos de contorno (*Shur Complement*), obtêm-se as incógnitas de cada subestrutura alocada em cada processador. Diminui-se, com isso, o tempo de sincronização para implementação em computadores de memória compartilhada e menor ociosidade dos processadores para as máquinas com memória distribuída.

Citam-se assim alguns trabalhos onde é aplicado estas técnicas: [8, 9, 6, 10, 11, 12, 13]. Dentre estes trabalhos, um que deve ser destacado é o realizado por [13], em que fazem a análise de uma estrutura tridimensional sob o efeito de temperatura, necessitando-se de 8 Gbytes para armazenar a matriz de rigidez. A técnica utilizada foi a técnica da decomposição em domínio em computadores com memória distribuída. Armazena-se, assim, as sub-regiões da estrutura nos diferentes processadores, ao todo são 22 nós, condensando os graus de liberdade nos nós do contorno entre cada sub-região. Desta forma, reduz o sistema e para resolvê-lo, utiliza-se um algoritmo híbrido direto-iterativo.

Para o trabalho aqui apresentado, a resolução do sistema foi realizada via um método iterativo denominado de método dos gradientes conjugados. Este método tem-se destacado para aplicação em processamento paralelo graças a sua facilidade de implementação e pela sua boa convergência quando se utiliza um pré-condicionador adequado. Assim, o próximo item mostrará em detalhes o método.

Ressalta-se que existem muitos outros métodos iterativos que oferecem vantagens para paralelização. Cita-se os trabalhos de [1, 14, 15], os quais apresentam em detalhes outros métodos iterativos, como: Gauss-Seidel, Jacobi e Quase-Newton (BFGS).

2.1 MÉTODO DOS GRADIENTES CONJUGADOS

O Método dos Gradientes Conjugados é aplicado sobre a matriz [K], sendo esta simétrica e positiva definida. A simetria é atendida tal que para um valor da matriz [K], por exemplo, k_{ij} tem-se que $k_{ji} = k_{ij}$, para qualquer i e j.

Para uma matriz [K] ser considerada positiva definida, tem-se que para todo vetor {U} não nulo, é válido $\{U\}^T[K]\{U\} > 0$. Geometricamente, isto significa que existe um ponto crítico $\{U_k\}$ na equação

$$\Phi(U) = \frac{1}{2}\{U\}^T \cdot [K] \cdot \{U\} - \{U\}^T \cdot \{F\} \qquad (2)$$

O Método dos Gradientes Conjugados (GC) é baseado na estratégia do método da *descida íngreme*. Nessa estratégia, é escolhido um ponto $\{U_0\}$, como valor inicial, e mediante uma série de aproximação $\{U_1\}$, $\{U_2\}$, $\{U_3\}$...,

$\{U_n\}$ que é realizada até encontrar um ponto numericamente próximo do mínimo da equação (1).

Um problema que surge na estratégia da descida íngreme, é que a convergência pode ser lenta, pois a estratégia não impede que em determinado passo, chegue-se a uma mesma direção que um outro passo anterior, isto causa uma oscilação no mesmo ponto que não é o de mínimo.

Como avanço, no Método dos Gradientes Conjugados (GC) é usado a estratégia de sempre ir buscando direções ortogonais $\{p_0\}$, $\{p_1\}$, $\{p_2\}$,....., $\{p_{n-1}\}$ às direções já calculadas no passo anterior. Para cada uma dessas direções, se encontrará uma das coordenadas de {U}, com isso, após *n* passos, que é a dimensão de {U}, o processo terminará e a solução procurada é encontrada.

A aproximação do vetor {U} fica neste processo:

$$\{u_{j+1}\} = \{u_j\} + \alpha_j \{p_j\} \qquad (3)$$

e a garantia para atingir a ortogonalidade das direções de $\{p_i\}$ e $\{p_{i+1}\}$ de forma que sejam K- ortogonais é:

$$\{p_i^T\} \cdot [K] \cdot \{p_j\} = 0 \text{ , para } i \neq j \qquad (4)$$

e o pseudo- algoritmo do Método dos Gradientes Conjugados pode ser visto em [16].

Os teoremas (1) e (2), que têm suas respectivas demonstrações em [2, 17], complementam os critérios de convergência do método:

Teorema 1: Se as direções de busca $\{p_0\}$, $\{p_1\}$, $\{p_2\}$,....., $\{p_{n-1}\}$ são K-conjugados (K-ortogonais) e o escalar α_j é escolhido da forma que: $\alpha_j = -\dfrac{\{p_j^T\} \cdot [K] \cdot \{r_j\}}{\{p_j^T\} \cdot [K] \cdot \{p_j\}}$ então o processo termina em, no máximo, *n* passos, onde *n* é a dimensão da matriz quadrada [K] do sistema da expressão (1).

Teorema 2: Seja [K] simétrica e definida positiva, então o algoritmo dos gradientes conjugados produz uma seqüência de vetores $\{U_0\}$, $\{U_1\}$, $\{U_j\}$..., com a seguinte propriedade:

$$\left\|\{U\} - \{U_j\}\right\|_K \leq 2 \cdot \left\|\{U\} - \{U_0\}\right\|_K \cdot \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^{\kappa} \qquad (5)$$

onde $\kappa = \dfrac{\lambda_{\max}}{\lambda_{\min}}$ e $\lambda_{máx}$ e $\lambda_{mín}$ são, respectivamente, o maior e o menor autovalor de [K] e κ é conhecido como número de condição.

Pela equação (5) dois fatores influenciam na convergência do método dos gradientes conjugados: a aproximação inicial $\{U_0\}$ e o número de condição (κ).

Em relação a aproximação inicial, na literatura é comentado que este fator é de pouca importância para a eficiência do método. Entretanto, em [18] é apresentado um estudo de otimização onde se obtêm uma aproximação inicial para o vetor $\{U_0\}$. Nos últimos anos, a pesquisa na área de inteligência artificial tem dado atenção a este item, onde a estratégia é criar bancos de dados de deformações e/ou de esforços de determinados modelos estruturais e, assim, aplicar algoritmos especiais conhecidos como meta-algoritmos para se otimizar a aplicação deste método.

Nota-se em (5) que a influência da variação do número de condição (κ) na convergência do método, é mais significativo do que a aproximação inicial $\{U_0\}$. Assim, quanto maior é a diferença entre os autovalores da matriz [K], mais demorada é a convergência do método. Dessa forma, o principal empecilho para a utilização deste método é justamente a necessidade de se ter um número de condição (κ) de [K] o mais próximo do valor unitário.

Nos últimos cinco anos extensos estudos têm sido dedicados à busca de técnicas matemáticas para se obter um número de condição (κ) o mais próximo do valor um.

A idéia básica empregada é de ao invés de se resolver o sistema $[K] \cdot \{U\} = \{F\}$, resolve-se um sistema do tipo $[M] \cdot [K] \cdot \{U\} = [M] \cdot \{F\}$, tal que a matriz obtida do produto ($[M] \cdot [K]$) seja o mais próximo possível da matriz

identidade, onde sabe-se que $\kappa_{identidade}=1$. Esta matriz [M] é conhecida como pré-condicionador, que nada mais é do que um acelerador de convergência para o método.

Ressalta-se que esta análise numérica de otimização da resolução de sistemas lineares via Método dos Gradientes Conjugados com o uso da técnica de pré-condicionamento, é a área que deve ser dada mais atenção para implementação no MEF, ou em qualquer outra aplicação com este método de resolução de sistemas. Sem o seu adequado uso em computação paralela, não se obtêm ganho de desempenho de tais arquiteturas, pois como o método é sensível ao condicionamento da matriz, em geral, a convergência sem a sua utilização é lenta.

2.2 PACOTE PIM

Para a implementação em paralelo do método dos Gradientes Conjugados (GC) utilizou-se o pacote de subrotinas PIM, ver [19], que foi desenvolvido para a resolução em ambiente paralelo de sistemas lineares simétricos e não-simétricos. Este pacote oferece além do método dos GC, outras alternativas de métodos iterativos como, por exemplo, o Bi-Gradiente Conjugado (Bi-GC), Gradiente Conjugado Quadrado (QGC), a versão estabilizada do Bi-Gradiente Conjugado (Bi-CGSTAB), o Resíduo Mínimo Generalizado (GMRES), dentre outros.

O pacote PIM, tem o intuito de dar total liberdade ao usuário no que se refere ao armazenamento da matriz, seu acesso e particionamento, bem como oferecer portabilidade aos diversos tipos de ambientes de programação em computadores paralelos.

Para se atingir estes dois fatores citados anteriormente, três pontos que são de responsabilidade do usuário para serem implementados no método são:

- O produto matriz-vetor;
- A implementação com o pré-condicionador (se houver);
- Produtos internos dos escalares e critério de parada.

2.2.1 PRODUTO MATRIZ-VETOR

No que se refere ao produto matriz-vetor, o pacote PIM deixa por conta do usuário sua implementação. Assim, para manter a portabilidade do pacote, o usuário deve conhecer as características especiais (se houver) de sua matriz, seja ela, por exemplo tridiagonal, pentadiagonal, cheia ou em banda, e desenvolver um procedimento heurístico ou não heurístico para utilizá-lo em processamento paralelo.

No que diz respeito ao armazenamento da matriz de rigidez negligenciando-se os valores fora do espectro do comprimento de semibanda, emprega-se o procedimento convencional para a montagem da matriz, onde gera-se seus valores começando pela diagonal principal e os alocam à partir da primeira coluna de cada linha, transladando-se estas colunas.

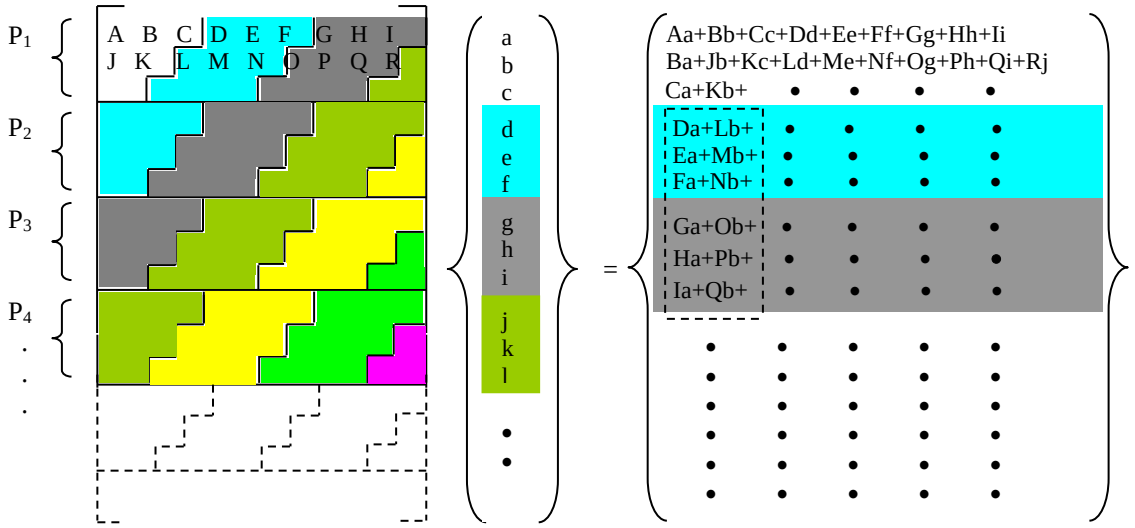
Para se realizar o produto matriz-vetor de um sistema em banda é necessário efetuar o produto de cada valor da matriz duas vezes (menos para os valores da primeira coluna que representam a diagonal principal), ou seja:

- Para a linha *i*, o valor $[K_{ij}]$ multiplica o valor do vetor $\{U_j\}$, armazenando este valor na linha *i (se* $i \neq j$*)*;
- Para a linha *j*, o valor $[K_{ij}]$ multiplica o valor do vetor $\{U_i\}$ e armazena-o em *j (se* $i \neq j$*)*.

Na figura (2), apresenta-se esquematicamente o procedimento em forma de pseudocódigo da metodologia empregada para se fazer o produto matriz-vetor com a matriz em semibanda. Resumidamente, tal procedimento é explicado como:

- Para cada processador em cada passo, gera-se um vetor auxiliar (**{Aux}**) que contenha as características do deslocamento (**{U}**), que é referente aos valores das posições armazenadas em sua memória local e as posições que estão nos processadores adjacentes que influenciam em seu produto local. Desta forma é necessário fazer troca de mensagem destes valores não conhecidos por cada processador, obviamente o último processador não precisa receber estes dados, pois os valores necessários já estão armazenados em sua própria memória;

- Para cada processador, obtêm-se os valores do vetor (**{V}**) resultante do produto local, os quais foram gerados pelo próprio processo. Cria-se um outro vetor auxiliar (**{S}**) que armazena os valores de um produto local, advindo da simetria, mas, que devem ser conhecidos pelos processadores adjacentes;



LEGENDA:

→ valores que devem ser conhecidos pelo processador 2

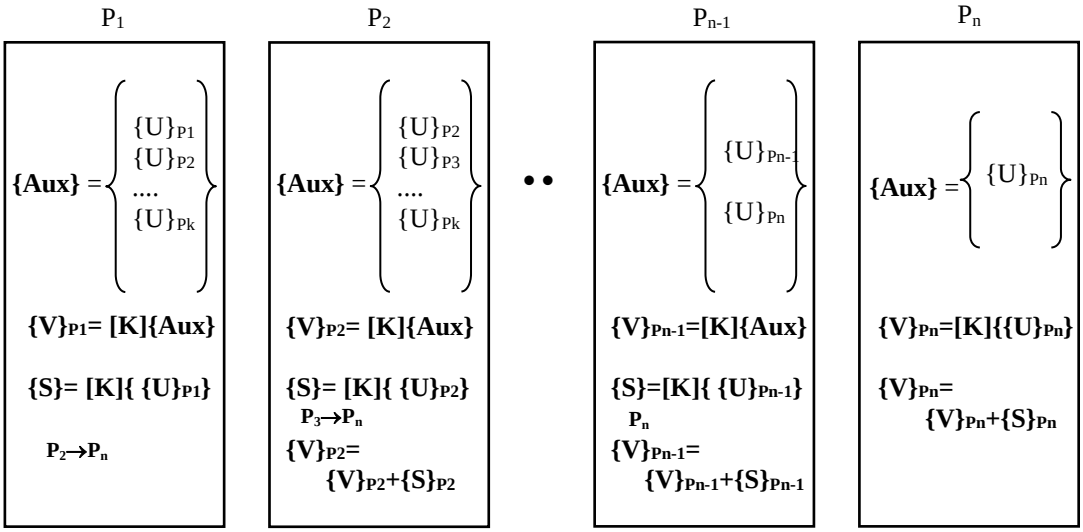
→ valores que devem ser conhecidos pelo processador 3

→ valores que devem ser conhecidos pelo processador 4

→ envio dos subprodutos

FIGURA 1 – Produto matriz-vetor para multicomputadores

- Finalmente, em cada processador - menos o primeiro – somam-se os seus valores resultantes locais com os produtos gerados pela simetria, ou seja o vetor (**{S}**), que foram obtidos pelos outros processadores adjacentes.



$P_i \rightarrow P_j$: processador i envia para os processadores (i+1) até j

FIGURA 2 – Procedimento esquemático do algoritmo para o produto matriz-vetor

2.2.2 PRODUTO INTERNO

Conforme apresentado pelo algoritmo da figura (4.2), verifica-se a necessidade de se realizar um produto interno do tipo $\alpha = \{u^T\} \cdot \{v\} = \sum_{i=1}^{n} \{u_i\} \cdot \{v_i\}$, onde $\{u\}$ e $\{v\}$ são vetores distribuídos dentro dos diversos processadores. Sem perda de generalidade, assume-se que cada processador armazena n/p elementos, onde n é o total de graus de liberdade da estrutura e p é número de processadores.

Este procedimento computacional em paralelo pode ser dividido em três partes:

1. A computação local de cada processador é dada por $\beta_j = \sum_{i=1}^{n/p} \{u_i\} \cdot \{v_i\}$ para cada processador;
2. A acumulação dos valores de β_j é feita através da seguinte forma apresentada na figura (4.5), com um exemplo com 7 processadores em grupo:

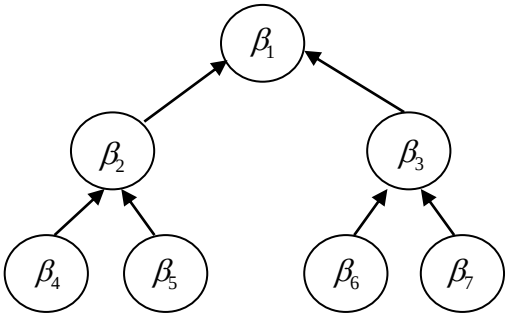


FIGURA 3 – Modelo de transferência de dados em árvore

e no final de tal procedimento, obtêm-se:

$\alpha = \sum_{j=1}^{p} \beta_j$ e armazena-se no processo pai.

3. Transmite-se, assim, para todos os processadores o valor de α .

3 ALGORITMO DO MÉTODO GC COM PRÉ-CONDICIONAMENTO

A adaptação do método dos gradientes conjugados puro, considerando-se o efeito do pré-condicionamento, pode ser feita de modo que, ao invés de se aplicar o pré-condicionador na equação (1), realiza-se a contribuição da matriz pré-condicionadora ao vetor resíduo $\{r_p\}$, ou seja:

$$\{r_p\} = \{F\} - [K] \cdot \{U\} \tag{6}$$

e aplicando a matriz pré-condicionadora, tem-se:

$$\{\tilde{r}_p\} = [M]^{-1} \cdot \{F\} - [M]^{-1} \cdot [K] \cdot \{U\} \tag{7}$$

Ou

$$\{\tilde{r}_p\} = [M]^{-1} \cdot \{r_p\} \tag{8}$$

3.1 DECOMPOSIÇÃO INCOMPLETA DE CHOLESKY (IC)

Em [20] é primeiramente apresentado o método da decomposição (ou também fatorização) incompleta de Cholesky para aplicação com o método dos gradientes conjugados. No trabalho destes pesquisadores, é discutido a possibilidade do uso desta técnica para resoluções de sistemas lineares simétricos e positivo-definidos para o caso onde a matriz é do tipo M-matriz[2]. Em [21], é mostrado que a matriz [K] gerada pelo MEF, mesmo não tendo as características de M-matriz, pode estender esta técnica apresentada em [20], e obter bons resultados de convergência para estes problemas.

3.1.1 MONTAGEM DO PRÉ-CONDICIONADOR INCOMPLETO DE CHOLESKY

O método da decomposição incompleta de Cholesky parte da idéia de se obter a matriz pré-condicionadora pela fatorização de [K] em uma decomposição de Cholesky, ou seja, fatorando a matriz [K] sob a forma:

2 [K]=(kij) é uma M-matriz se kij ≤ 0 p/ i ≠ j, [K] sendo não-singular e [K-1] ≥ [0], sendo que estas matrizes são geradas freqüentemente pela discretização de equações diferenciais elípticas e parabólicas.

$$[K] = [L] \cdot [L^T] \quad (9)$$

onde [L] é uma matriz triangular inferior de dimensão ($n \cdot n$).

Porém, fatorar a matriz [K] em [L] faz com que esta matriz triangular não seja tão esparsa dentro do campo dado pelo comprimento em banda quanto a matriz [K]. Isto é um "gargalo" para a resolução numérica do método, já que há a necessidade de se armazenar estas matrizes cheias, ou seja, $(n \cdot m)$, acarretando mais uso de memória e maior quantidade de dados a serem manipulados em seu processamento.

Desta forma, perante este "gargalo", o método da decomposição *incompleta* se baseia em se fazer uma fatorização de [L] de [K], de maneira que *alguns elementos são negligenciados* na matriz [L] em *posições apropriadas*. Estas posições são escolhidas de modo que se possa manter a esparsidade de [K] sobre as matrizes triangulares. Por isso, o método é denominado de *decomposição incompleta de Cholesky.*

Nestas matrizes triangulares, os valores negligenciados podem ocorrer em lugares arbitrários[3] (mas fora da diagonal principal) e as posições (i,j) não nulas serão indicadas por um conjunto P que é descrito a seguir:

$$P \subset P_N \equiv \{ (i , j) \mid i \neq j, 1 \leq i \leq n, 1 \leq j \leq n \} \quad (10)$$

onde P_N contêm todos os pares de índices de entradas da matriz.

Desta forma, pode-se escolher um campo de preenchimento da matriz [L], indicado por P, onde o par (i,j) indicará as posições de todos os valores diferentes de zero.

3.1.2 VARIAÇÕES DOS ESPECTROS SOBRE L

Para o presente trabalho, desenvolve-se um algoritmo em que o espectro de influência sobre as posições não nulas na matriz [L] são variáveis de entrada do problema. Pode-se, assim, ser escolhido o comprimento de influência das diagonais a partir da diagonal principal, como também a partir da última diagonal não nula.

Como os valores de m_1 e m_2 podem não ser iguais, neste trabalho não poderia ser utilizado a abreviatura apresentada em [20]. Por isso, criou-se uma referência específica para a abreviação do método para se incluir os parâmetros m_1 e m_2.Como exemplo, considere um campo de P tal que $m_1 = 4$ e $m_2 = 3$. Assim, o conjunto P será denominado de $P_{m_1}^{m_2}$ e será dado por:

$$P_4^3 \equiv \{ (i , j) \mid |i - j| \neq 0, 1, 2,3,m-3,m-2,m-1 \} \quad (11)$$

A notação para a utilização do método dos gradientes conjugados com este pré-condicionador com influência genérica, será indicada por ICCG(m_1,m_2) e, para este último exemplo, tem-se ICCG(4,3).

3.1.3 PARALELIZAÇÃO DO MÉTODO IC

O método da decomposição incompleta de Cholesky, para implementação em ambiente em paralelo, não apresenta características vantajosas para este tipo de ambiente. Isto ocorre tanto para a montagem da matriz triangular inferior [L] como para se obter o vetor pré-condicionado.

No tocante a montagem da matriz [L], averigua-se que esta técnica é própria para se realizar em programação seqüencial, já que a técnica em si é feita percorrendo-se *todas as linhas da matriz [K]* e, então, a partir de cada linha, que chamaremos de linha-base, altera-se as demais linhas e colunas abaixo desta linha–base, procedendo assim para cada variação da linha-base. Assim, para os diversos processadores que têm domínio sobre as linhas que estão abaixo da linha-base, estes tem que esperar o processador que tem controle da linha-base realizar a instrução sobre sua linha. Em seguida, este processador envia esta linha para os processadores inferiores. Só então, estes podem realizar sua instrução.

Percebe-se, também, que se o (s) processador (es) tem (têm) domínio sobre as linhas que estão acima da linha-base, este (s) processador (es) tem que ficar ocioso (s) esperando o final do processo. A seguir, mostra-se de forma esquemática o procedimento discutido.

3 Não tão arbitrários assim, pois dependendo da escolha destas posições, a matriz incompleta [L] pode não ser positiva-definida.

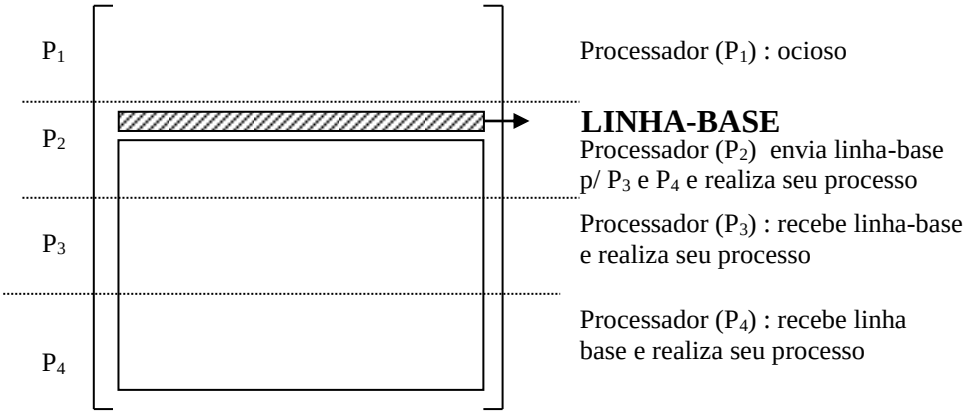


FIGURA 4 – Obtenção da matriz [L] em programação paralela

A resolução do sistema apresentado na equação (5.14) possui também características próprias para programação seqüencial. Pois, para se obter o vetor resposta {r}, realiza-se, respectivamente, uma substituição direta e uma retrosubstituição sobre o sistema.

Isto ocorre porque a matriz de rigidez e os vetores utilizados foram divididos em sub-blocos e cada sub-bloco armazenado em um processador diferente. Assim, cada processador tem que aguardar a realização das instruções sobre o processador adjacente, seja referente ao bloco superior para o caso da substituição direta ou sobre o bloco inferior para o caso da retrosubstituição. Esquematicamente, na figura (5), é apresentado o procedimento para resolução do sistema, sendo que (*np)* representa o número de processadores:

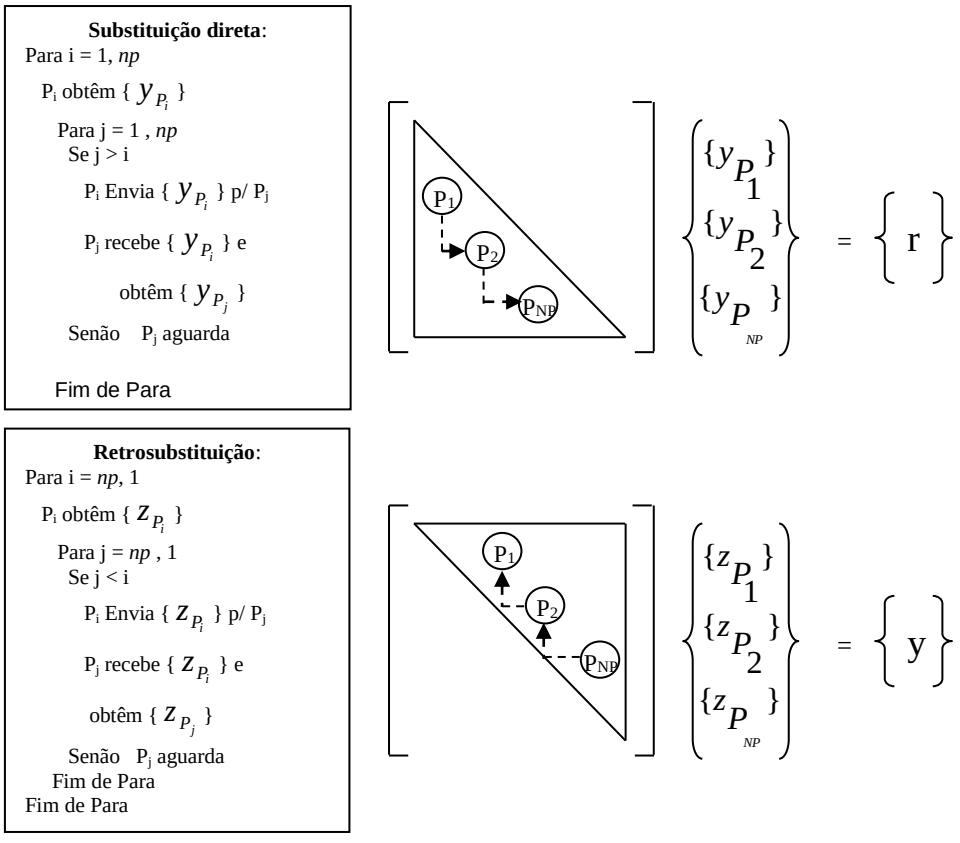


FIGURA 5 – Esquema para obter o vetor {z} pré-condicionado em paralelo

4 MEDIDA DE DESEMPENHO

A medida usual de desempenho no processamento de programas ou de trechos paralelos é o fator conhecido como *Speed-up*. Este eqüivale a relação do quociente entre o tempo necessário para sua execução em programação sequencial e o tempo em processamento paralelo com *n* processadores.

Assim, esta relação é expressa por:

$$Speed-up = \frac{\text{Tempo com 1 processador}}{\text{Tempo com n processadores}} \qquad (12)$$

Esta relação mostra que para valores acima de 1 tem-se um ganho na execução do processamento com *n* processadores. Aliada à medida de *speed-up*, tem-se a medida de eficiência do processo, a qual é dada pela relação

$$Eficiência = \frac{100 \cdot (Speed-up)}{\text{número de processadores}} \quad (\%) \qquad (13)$$

O limite superior do *speed-up* é dado pela relação

$$Speed-up \leq \frac{1}{s + \frac{r}{n}} \qquad (14)$$

de tal modo que r representa a fração em tempo do processo que é paralelizável dentro programa e s é dado por s = r-1.

Assim, por exemplo, onde o tempo de análise em paralelo é bem superior que o tempo sequencial, como no caso da resolução do sistema com o emprego do método dos gradientes conjugados, pode-se, sem perda de generalidade, afirmar que r = 1, ou seja, 100% do programa é paralelizável. Portanto, nestes casos o *Speed-up* é limitado por

$$(Speed-up) \leq n \qquad (15)$$

5 APLICAÇÃO NUMÉRICA

Apresenta-se aqui os resultados de *speed-up*, eficiência e do tempo de processamento de cada rede variando-se os pré-condicionadores e o número de processadores. Na figura (6) é mostrado o modelo esquemático da treliça gerada. Para este modelo variou-se o tamanho da rede com a utilização de um gerador próprio desenvolvido. Nesta figura também é apresentado as características geométricas e físicas do material. Este exemplo foi extraído de [22], bem como as características adotadas, sendo que aplicou-se uma carga concentrada no nó central da estrutura, no banzo superior, de valor indicado na tabela da figura (6).

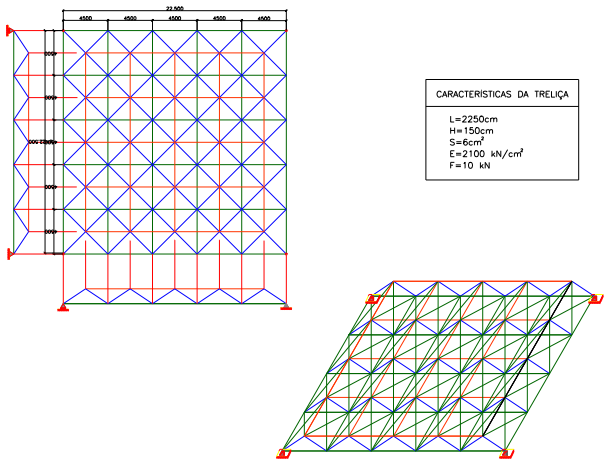


FIGURA 6 - Planta e perspectiva do modelo de treliça

Rede A: 19208 elementos e 4901 nós
Rede B: 63368 elementos e 16021 nós
Rede C: 95048 elementos e 23981 nós

A seguir, nas figuras (7) a (10), são apresentados os gráficos de *speed-up* por número de processadores e de

eficiência por número de processadores. O pré-condicionador ICCG(1,0) é classicamente conhecido como POLY(0).

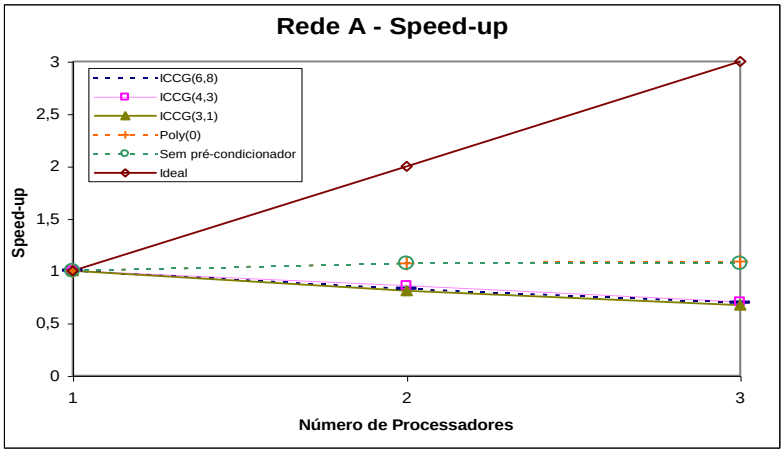


FIGURA 7 –Speed-up da rede A

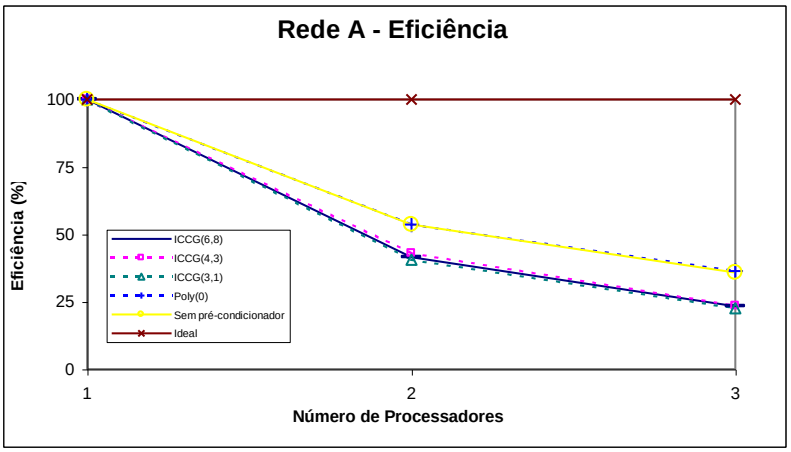


FIGURA 8 –Eficiência da rede A

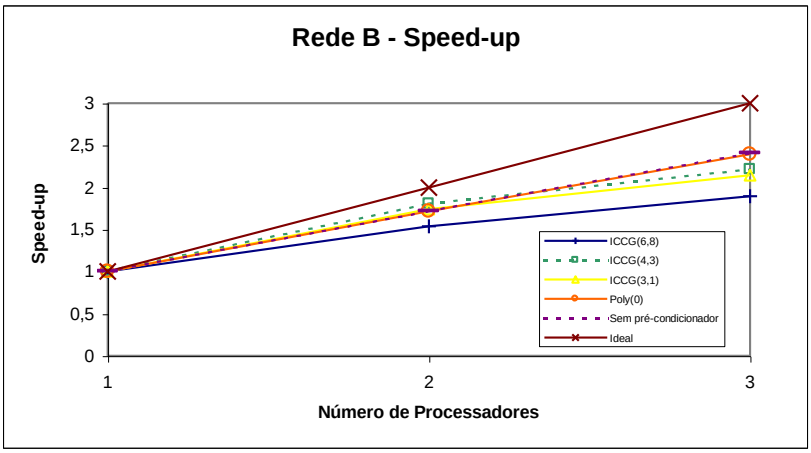


FIGURA 9 –Speed-up da rede B

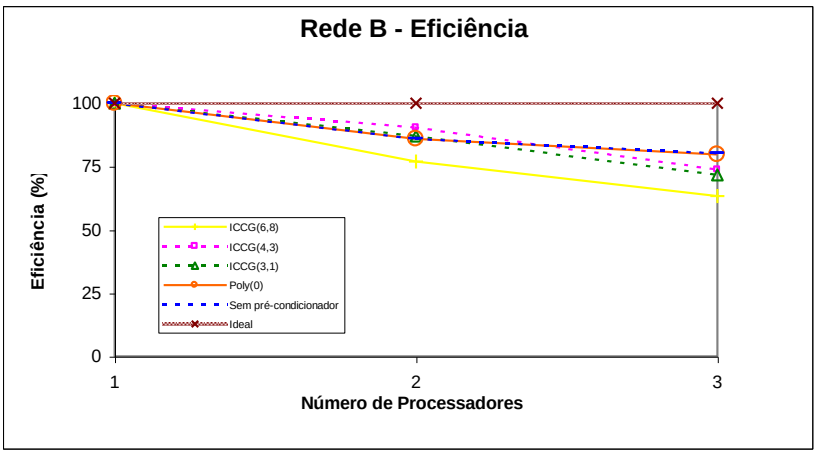


FIGURA 10 – Eficiência da rede B

Os valores de *speed-up* e de eficiência encontrados para a treliça demonstram, como era de se esperar, que o uso do paralelismo só se torna significativo quando tem-se um número de equações do sistema relativamente grande, ou seja, a partir do modelo B.

Pelos valores encontrados para o modelo B, tanto para *speed-up* como para eficiência, nota-se que com a maior quantidade de trocas de mensagens e também com a maior dependência entre os diversos processadores, que ocorre com o uso dos pré-condicionadores do tipo ICCG, a performance do sistema diminui a medida que aumenta-se o número de processadores, no caso de 1, 2 e 3.

Por outro lado, um outro fator que, também, deve ser considerado na performance do programa é o seu tempo comparativo de convergência para os diversos pré-condicionadores sob os diversos processadores. Já que os medidores de desempenho são parâmetros de comparação de eficiência do paralelismo no código computacional, mas *não mostrando a performance relativa de tem*po entre os diferentes pré-condicionadores.

Por exemplo, conforme a figura (9), o valor do *speed-up* para a rede B *com o pré-condicionador* ICCG(6,8) foi de *1,9* enquanto que *sem o uso de pré-condicionador* obteve-se um *speed-up* de *2,4*, ambos com 3 processadores. Por outro lado, o *tempo* requerido para resolver o sistema *com o pré-condicionador* ICCG(6,8) foi em torno de *5000* segundos e o tempo *sem pré-condicionador* foi da ordem de *15000 segundos.*

Assim, nas figuras (11) a (13) são apresentados os tempos de resolução das diversas redes sob os diversos aceleradores de convergência.

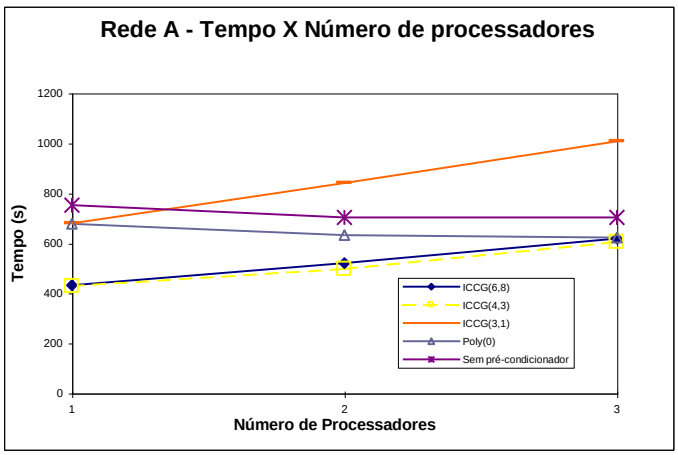


FIGURA 11 –Convergência em termos de tempo da rede A

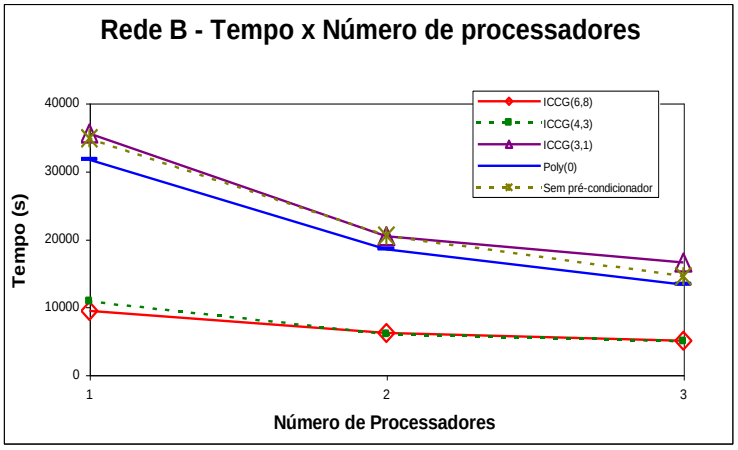


FIGURA 12 –Convergência em termos de tempo da rede B

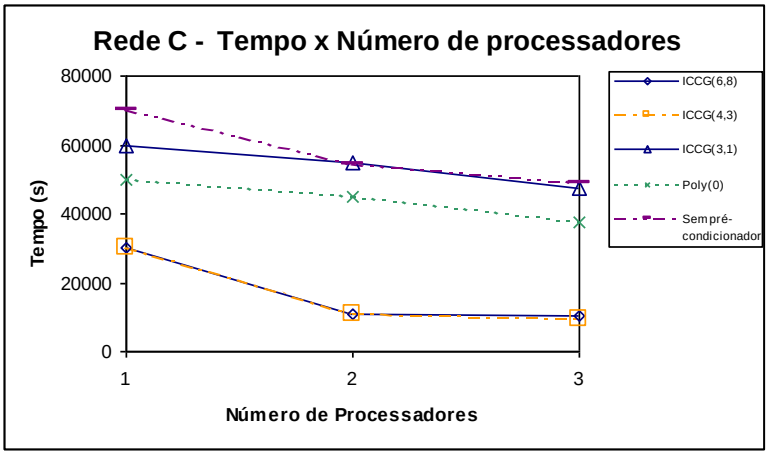


FIGURA 13 –Convergência em termos de tempo da rede C

Pelo apresentado pelas figuras (12) e (13), o tempo de resposta com os pré-condicionadores ICCG(6,8) e ICCG(4,3), em função da alta taxa de convergência destes pré-condicionadores, mesmo estes perdendo eficiência paralelismo, se tornaram melhores pré-condicionadores do que com o uso do POLY(0) para este modelo, com este número de processadores.

Para confirmar a eficiência em termos de tempo dos pré-condicionadores ICCG(6,8) e ICCG(4,3) com, no máximo, 3 processadores sob o modelo apresentado, na figura (14) é plotado para as diversas redes deste modelo, uma curva associando o número de graus de liberdade *versus* o tempo de processamento, sendo que isto foi feito para três processadores.

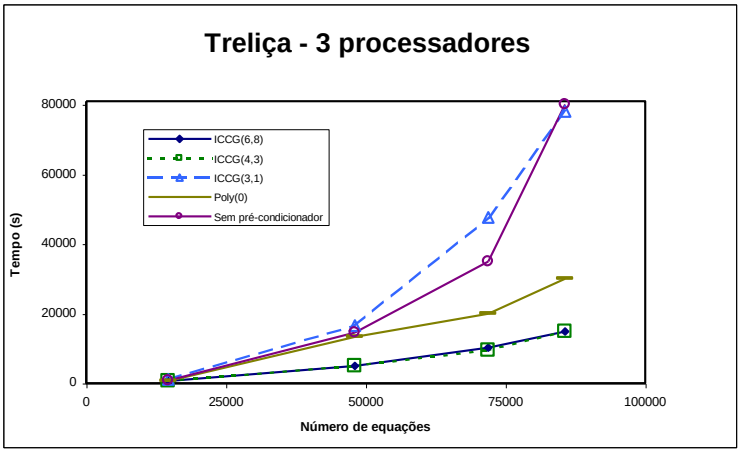


FIGURA 14 –Curva (Tempo x N° de equações) para a treliça

O último ponto da curva da figura (14) foi realizado em uma rede com 113288 elementos e 28561 nós, ou seja, com 85683 incógnitas no sistema linear, levando a formação de uma matriz de rigidez com 85683 linhas e 720 colunas (em semibanda) ou aproximadamente 500 Mbytes de memória RAM.

6 CONSIDERAÇÕES FINAIS

O presente trabalho teve o intuito de adaptar um código computacional advindo do método dos elementos finitos aplicado na análise de uma estrutura em multicomputadores. Em virtude da arquitetura deste tipo de máquina, teve-se que modificar alguns procedimentos numéricos já consagrados pelo método convencional aplicado ao MEF. Destaca-se, assim, a alteração para a montagem e a resolução do sistema linear.

Convêm ressaltar que o ponto de "gargalo" para a otimização do método em processamento paralelo ocorre justamente na resolução do sistema. Assim, para se adaptar o MEF para este tipo de arquitetura, é necessário dar ênfase, principalmente, a esta fase, sendo os demais procedimentos, de certa forma, corriqueiros na paralelização. A importância do estudo da resolução do sistema em multicomputadores se torna mais evidenciada quando se aplica a problemas envolvendo não-linearidades, pois nestes processos, em geral, para cada incremento de passo, é necessário resolver um novo sistema.

Para a montagem do sistema linear, utilizou-se o procedimento proposto por [7], o qual fora desenvolvido para computadores de memória compartilhada. Porém, este se adaptou plenamente aos multicomputadores. Esta alteração, chamada em seu trabalho de abordagem nodal, consiste em montar a matriz de rigidez linha por linha e não pela forma convencional elemento por elemento. Assim, para multicomputadores, isto é aproveitado de forma eficiente, pois divide-se a matriz de rigidez dentre os diversos processadores e então, paralelamente, e mais importante ainda, de maneira independente.

A grande vantagem oferecida por esta técnica, comparada com a técnica da decomposição em domínio, é que para esta última é necessário aplicar um pré-processador para se redividir a rede, de tal modo a se obter um balanceamento equilibrado dentre os vários processadores. Na técnica aqui empregada, o balanceamento de carga é obtido naturalmente.

Para a resolução do sistema linear, utilizou-se um método iterativo denominado de gradiente conjugado. Esta técnica adequou-se bem na análise em multicomputadores, pois, para cada passo, é necessário realizar um produto matriz-vetor, sendo este procedimento bem adequado na realização em paralelo. Implementou-se este produto levando-se em consideração a simetria e a esparsidade características do sistema. Com isso, aumentou-se a dependência entre os diversos processadores, mas procurando-se não perder o potencial oferecido pelo produto sob diversos processadores, e em virtude dos resultados encontrados, este resultado demonstrou ser eficiente.

Além disso, implementou-se também a soma global de um escalar, que é originado do produto interno local de cada processador, advindo tanto para a procura da direção que resolva o sistema quanto para o escalar originado pela norma utilizada. O grande obstáculo que existe na utilização do MGC é a dependência para convergência do método em relação ao número de condição da matriz de rigidez. Assim, o método pode se tornar eficiente ou ineficiente conforme as características intrínsecas deste sistema.

Para solucionar este problema, é comum aplicar o pré-condicionamento sobre o sistema, a fim de tornar o método mais eficaz. Assim, neste trabalho, foi proposto para processamento em paralelo uma adaptação do pré-condicionador advindo da decomposição incompleta de Cholesky (IC); onde foi generalizado o espectro de influência para se considerar qualquer combinação de diagonais extras na montagem de [L].

Para o exemplo da treliça tridimensional, percebe-se que o tempo de convergência com o uso do $ICCG(4,3)$ é um pouco inferior ao obtido pelo *ICCG(6,8)*. O segundo mostra uma convergência em termos de iteração um pouco menor que o primeiro. Porém, em virtude da quantidade de operações serem maior no *ICCG(6,8)*, em geral, o *ICCG(4,3)* mostrou-se mais adequado para grandes sistemas. Ambos, quando comparados com POLY(0), e sem o pré-condicionador, apresentaram valores da ordem de $\frac{1}{3}$ inferiores aos dois últimos. Vê-se também que a eficiência destes dois primeiros foi de 63% para 3 processadores.

7 AGRADECIMENTOS

O trabalho presente foi financiado pela FAPESP: Fundação de Amparo à Pesquisa do Estado de São Paulo. Desta forma, os autores agradecem o apoio da instituição para a confecção do trabalho.

8 BIBLIOGRAFIA

[1] K. J Bathe, *Finite element procedures in engineering analysis*, Englewood Cliffs, Prentice-Hall (1982).

[2] J. A Cuminato e M. Jr Meneguette, *Discretização de equações diferenciais parciais: Técnicas de Diferenças finitas*. Instituto de Ciência e Matemática de São Carlos, Universidade de São Paulo (1998).

[3] A. del Grosso e G. Righetti, "Finite element techniques and artificial intelligence on parallel machines", *Computers & Structures*, v. 30, n.04, p. 999-1007 (1988).

[4] K. H. Law, "*A parallel finite element solution method"*. Comput. Struct, v. 23, n. 6, p. 845-858 (1986).

[5] S. Bitzarakis, M. Papadrakakis e A. Kotsopulos, "Parallel solution techniques in computational structural mechanics", *Comput. Methods Appl. Mech. Engr*., v. 148, p. 75-104 (1997).

[6] Adeli H. e O. Kamal, "*Concurrent Analysis of large structures-I ". Algorithms. Comput. Struct.*, v. 42, n. 03, p. 413-424 (1992).

[7] M. N Rezende, *Processamento paralelo em análise estrutural*, São Carlos. 112p. Tese (Doutorado) – Escola de Engenharia de São Carlos, Universidade de São Paulo (1995).

[8] D. Zois, "*Parallel processing techniques for FE analysis: stiffiness, loads and stresses evaluation"*. Comput. Struct,. v. 28, p. 247-260 (1988a).

[9] D. Zois, "*Parallel processing techniques for FE analysis: system solution"*. Comput. Struct, v. 28, p. 261-274 (1988b).

[10] A. R. M. Rao; K Loganathan e K. N. V. Raman, "*Studies on two concurrent solvers for finite element analysis*." Advances in Engineering Software. v. 18, p. 161-166 (1993).

[11] K. H. Law e D. R. Mackay, "*A parallel row-oriented sparse solution method for finite element structural analysis"*. International Journal for numerical methods in engineering, v. 36, p. 2895-2919 (1993).

[12] B. H. V. Topping e A. I. Khan, "*Parallel Finite Element Computations.",* 1ª ed., Saxe-Coburg Publications, Edinburgh (1996).

[13] Y. Jiang, "*Parallel Computation of Polymer Melt Flow through an extruder*. Polyflow", s.a., Place de l'Université 16, B-1348 Louvain-la-Neuve, Belgium (1997).

[14] J. W. Demmel, "*Berkeley Lecture Notes on Numerical Linear Algebra".* Mathematics Department and Computer Science Division, University of California (1993).

[15] M. Cimerman, *Resolução de Sistemas Lineares Via Métodos Iterativos com Pré-Condicionadores – Aplicação em Problemas de Engenharia de estruturas.* São Paulo. 111p. Dissertação (Mestrado) – Escola Politécnica, Universidade de São Paulo (1996).

[16] Valério S Almeida, *Uma adaptação do MEF para análise em multicomputadores: aplicações em alguns modelos estruturais*, São Carlos. 126p. Dissertação (Mestrado) – Escola de Engenharia de São Carlos, Universidade de São Paulo (1999).

[17] D. G. Luenberger, "*Optimization by vector space methods".* John Wiley,New York (1969).

[18] L. A. Jr Schmt e Y. C. Lai, "Structural optimization based on preconditioned conjugate gradient analysis methods". International Journal for numerical methods in engineering, v. 37, p. 943-964 (1994).

[19] R. Cunha e T. Hopkins, "*PIM 1.1: The Parallel Iterative Methods package for Systems of Linear Equations User's Guide (Fortran 77 version)".* Computing Laboratory, University of Kent at Canterbury, United Kingdom (1998*).*

[20] J. A Meijerink e H. van der Vorst, "An iterative solution method for linear systems of wich the coefficient matrix is a symmetric M-matrix", *Mat. Comp*., v.31 (137), p.148-162 (1977).

[21] T. A Manteuffel, "An incomplete factorization technique for positive definitive linear systems", *Mat. Comp*., v.34 (150), p.473-497 (1980).

[22] A. S. C. Souza, "*Contribuição ao estudo das estruturas metálicas espaciais".* São Carlos. 147p. Dissertação (Mestrado) – Escola de Engenharia de São Carlos, Universidade de São Paulo (1998).