UMA ESTRATÉGIA DE DECOMPOSIÇÃO DE DOMÍNIO EM NÍVEIS APLICADA A PROBLEMAS DE MECÂNICA DOS SÓLIDOS

Rubens M. Sydenstricker
Renato Nascimento Elias
Alvaro L.G.A. Coutinho
Marcos A.D. Martins
(*rubens,renato,alvaro,marcos@nacad.ufrj.br*
Núcleo de Computação de Alto Desempenho,
COPPE/ Universidade Federal do Rio de Janeiro
Caixa Postal 68516, Rio de Janeiro, RJ 21945-970, Brasil
Luiz Landau
landau@lamce.ufrj.br
Laboratório em Métodos Computacionais em Engenharia,
COPPE/ Universidade Federal do Rio de Janeiro
Caixa Postal 68506, Rio de Janeiro, RJ 21945-970, Brasil

Resumo. *O tempo de processamento e capacidade de memória permanecem como pontos críticos em muitas aplicações de engenharia e outras áreas. Em simulações numéricas, tais como as realizadas através do Método dos Elementos Finitos (MEF), a solução de sistemas de equações constitui a tarefa que demanda maior tempo. Muitas estratégias, tais como decomposição de domínio, solução de sistemas de equações por métodos iterativos, pré-condicionamentos, estruturas de dados especiais e processamento paralelo são usadas para diminuir os requerimentos computacionais. Neste trabalho, usamos uma estratégia de decomposição de domínio para reduzir o tempo na solução de sistemas equações que resultam do MEF. A abordagem adotada é muito simples, pois utilizamos o programa Metis 4.0, de distribuição gratuita, para particionar a malha. Adotamos um método misto de solução: nas operações de condensação e recuperação de graus de liberdade, adotamos um método direto do tipo eliminação de Gauss e, para resolver o sistema global de equações, adotamos o método iterativo dos Gradientes Conjugados. O esquema resultante permite que cerca de 80% das equações sejam rapidamente solucionadas através de processos de condensação e recuperação. A eficiência do método dos Gradientes Conjugados para malhas constituídas de super-elementos é estudada e experiências numéricas mostram a efetividade da estratégia.*

Palavras chave: *Elementos finitos, Decomposição de domínio, Computação paralela.*

1. Introdução

A solução de sistemas de equações é a tarefa que consome a maior parte do tempo de processamento na simulação numérica através do método dos elementos finitos, particularmente quando tratamos de domínios com um grande número nós e, conseqüentemente, de equações. A fim de ilustrar o aumento do esforço computacional com número equações, vamos considerar os procedimentos de solução verificados em um método direto, tal como eliminação de Gauss, e em um método iterativo, tal como o dos Gradientes Conjugados.

Na solução de sistemas através do método de eliminação de Gauss, o maior esforço é destinado ao processo de fatoração da matriz de coeficientes (matriz de rigidez). Nesta etapa, a matriz de coeficientes é escalonada até que seja obtida uma matriz de forma triangular. O esforço gasto com a manipulação de cada linha da matriz é proporcional ao tamanho (número de coeficientes) da linha, que por sua vez está diretamente associado ao número de equações. Alem disso, o número de linhas também está diretamente relacionado com o número de equações, de forma que o esforço computacional no processo de fatoração tende a variar quadraticamente com o número de equações.

Na solução de sistemas de equações através do Método dos Gradientes Conjugados, o maior esforço computacional é destinado à multiplicação da matriz de rigidez de cada elemento finito do domínio pelo respectivo vetor de deslocamentos. Esse processo é repetido em um dado número de iterações. O número de elementos da malha tem a mesma ordem do número de nós, que está diretamente relacionado ao número de equações. Assim, se há um aumento do número de equações, também há um aumento do número de elementos, o que acarreta um aumento, de proporções semelhantes, do esforço computacional em cada iteração. Além disso, o número de iterações também aumenta com o número de equações e, mais uma vez, o esforço computacional (e o tempo de processamento) aumenta de forma não linear e desfavorável com o número de equações.

Técnicas de decomposição de domínio (Fragakis e Papadrakakis, 2003; Ellwanger, 1989; Smith, Bjorstad e Gropp, 2004) oferecem a oportunidade de obter uma melhor relação entre o aumento do esforço computacional e do número de incógnitas nodais. Essas técnicas permitem que o domínio seja particionado, e que os nós internos de cada partição sejam eliminados do sistema global de equações através de técnicas de condensação, restando apenas as incógnitas nodais associadas ao contorno das partições. Após a solução do sistema global, as incógnitas nodais internas são recuperadas. Se considerarmos partições de mesmo tamanho, o esforço computacional no processo de condensação e recuperação de incógnitas internas será diretamente proporcional ao tamanho da malha, ou seja, se dobramos o número de partições, esse esforço computacional também dobrará. Assim a técnica é particularmente interessante quando tratamos de problemas de grande escala

Neste trabalho, procuramos desenvolver uma estratégia de decomposição de domínio visando à redução do tempo de processamento em análises de elementos finitos. O estudo é realizado em um problema bidimensional de mecânica dos sólidos, cuja solução é conhecida e independe do grau de refinamento da malha. Assim, mantemos o foco apenas no tempo de processamento e no consumo de memória para armazenamento dos coeficientes de rigidez. É importante observar que a técnica (e o programa computacional desenvolvido) pode ser aplicada a outros tipos de análise (tais como de escoamento de fluidos e transferência de calor) assim como para malhas não estruturadas e de geometria irregular. O exemplo que apresentamos consiste em um simples ensaio que visa ilustrar e avaliar o potencial da técnica proposta.

2. **CONDENSAÇÃO ESTÁTICA**

Considere uma estrutura constituída de um único elemento finito contendo nós internos, ou de um agrupamento de elementos, como mostra a parte esquerda da Figura 1. Então, o sistema de equações [K]{d}={R} do elemento (Figura 1 superior), ou da estrutura (Figura 1 inferior), pode ser particionado como:

$$\begin{bmatrix} [Krr] & [Krc] \\ [Kcr] & [Kcc] \end{bmatrix} \begin{Bmatrix} \{\hat{dr}\} \\ \{\hat{dc}\} \end{Bmatrix} = \begin{Bmatrix} \{Rr\} \\ \{Rc\} \end{Bmatrix} \tag{1}$$

onde $\{\hat{dr}\}$ contem os graus de liberdade dos nós de contorno, $\{\hat{dc}\}$ contém os graus de liberdade dos nós internos, {Rr} contém as forças externas aplicadas no contorno e {Rc} as forças nodais aplicadas nos nós internos.

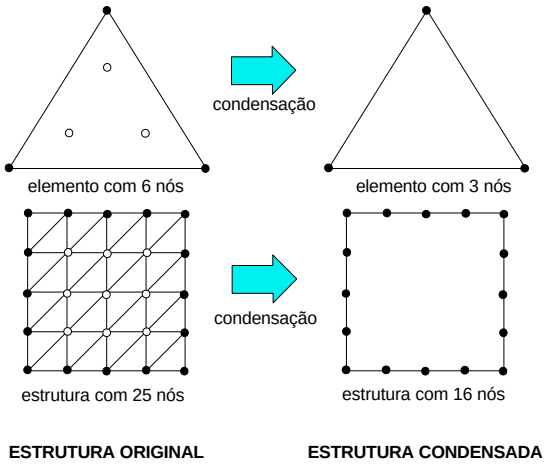


Figura 1 – Condensação estática

Da partição inferior da equação (1) escrevemos:

$$[Kcr]\{\hat{dr}\} + [Kcc]\{\hat{dc}\} = \{Rc\}$$

$$\therefore \quad \{\hat{\mathrm{dc}}\} = -[\mathrm{Kcc}]^{-1}\left([Kcr]\{\hat{dr}\} - \{Rc\}\right) \tag{2}$$

Da partição superior de (1) escrevemos:

$$[Krr]\{\hat{dr}\} + [Krc]\{\hat{dc}\} = \{Rr\} \tag{3}$$

Substituindo (2) em (3) encontramos:

$$\left([Krr] - [Krc][Kcc]^{-1}[Kcr]\right)\{\hat{dr}\} = \{Rr\} - [Krc][Kcc]^{-1}\{Rc\} \tag{4}$$

A equação (4) pode ser escrita como:

$$[K_{cond}]\{\hat{dr}\} = \{R_{cond}\} \tag{5}$$

onde $[K_{cond}]$ e $\{R_{cond}\}$ são respectivamente a matriz de rigidez e o vetor de forças da estrutura condensada, contendo apenas nós de contorno, como mostra o lado direito da Figura 1. Das equações (4) e (5) escrevemos:

$$[K_{cond}] = [Krr] - [Krc][Kcc]^{-1}[Kcr] \tag{6}$$

$$\{R_{cond}\} = \{Rr\} - [Krc][Kcc]^{-1}\{Rc\} \tag{7}$$

A este processo de redução dos graus de liberdade de um elemento ou estrutura denominamos *condensação estática*. Se apenas graus de liberdade internos forem condensados (eliminados do sistema de equação global), então a estrutura resultante pode ser tratada como um único elemento, que pode ser conectado a outros elementos ou agrupamento de elementos, mantendo-se a condição de compatibilidade. O elemento condensado pode ser denominado macro-elemento, tendo a matriz de rigidez $[K_{cond}]$, o vetor de carga $\{R_{cond}\}$ e vetor deslocamento $\{\hat{dr}\}$. Esses deslocamentos são fornecidos pelas respostas de uma análise de elementos finitos que utilize tais macro-elementos, e os graus de liberdade dos nós internos podem então ser determinados a partir da equação (2), num processo conhecido como recuperação dos graus de liberdade condensados.

Neste estudo propomos a utilização da técnica de condensação estática como instrumento de aumento do desempenho computacional. Vamos considerar a condensação dos graus de liberdade internos de partições de uma malha constituída de elementos triangulares com três nós e dois graus de liberdade (g.l.) por nó. Por simplicidade, iremos considerar apenas estruturas sem cargas aplicadas nos nós internos, e com condições de contorno aplicadas apenas nos nós externos (contorno da malha).

Na Figura 2, apresentamos os algoritmos utilizados para condensação estática e recuperação de graus de liberdade condensados. Estes algoritmos são fornecidos por Cook *et al*. (1989). No algoritmo de condensação estática, os dados de entrada são:

número inicial de graus de liberdade : *NSIZE*

número de graus de liberdade a condensar: *NUM*

vetor de cargas, organizado como : $\{RE\}_{NSIZE} = \{RE_{não\ cond.}\}_{NSIZE} = \begin{Bmatrix} \{R_r\}_{NSIZE-NUM} \\ \{R_c\}_{NUM} \end{Bmatrix}$

matriz de rigidez não condensada: $[SE]_{NSIZExNSIZE} = [SE_{não\ cond.}]_{NSIZExNSIZE}$

e os dados de saída são:

$$\{RE\}_{NSIZE} = \begin{Bmatrix} \{R_{cond}\}_{NSIZE-NUM} \\ \{RM\}_{NUM} \end{Bmatrix}$$

$$[SE]_{NSIZExNSIZE} = \begin{bmatrix} [SE_{cond}]_{(NSIZE-NUM)\,x\,(NSIZE-NUM)} & [SE_{lixo}]_{(NSIZE-NUM)xNUM} \\ [SM]_{NUMxNSIZE} \end{bmatrix}$$

onde $[SE_{cond}]$ é a matriz de rigidez do macro-elemento, $\{R_{cond}\}$ é seu vetor de cargas (que não será considerado aqui) e [SM] é uma matriz que será utilizada na fase de recuperação dos g.l.. Fizemos duas alteração neste algoritmo: a) introduzimos a 1ª linha (LL=0), a fim de evitar erros em caso de NUM=0; b) excluímos a 11ª linha (RE(L) = RE(L) - RE(KK) * DUM), pois neste ensaio não iremos considerar cargas fora do contorno da malha.

No algoritmo de recuperação os dados de entrada são:

$SIZE$, NUM

$[SM]_{NUMxNSIZE}$ e $\{RM\}_{NUM}$ $\Rightarrow$ fornecidos pela rotina de condensação (nós não utilizamos {RM})

$$\{DE\} = \begin{Bmatrix} \{\hat{d}_r\}_{NSIZE-NUM} \\ \{d_{lixo}\}_{NUM} \end{Bmatrix} \quad \Rightarrow \text{onde } \{\hat{d}_r\} \text{ contém os g.l. não condensados}$$

e os dados de saída são:

$$\{DE\} = \begin{Bmatrix} \{\hat{d}_r\}_{NSIZE-NUM} \\ \{\hat{d}_c\}_{NUM} \end{Bmatrix} \quad \Rightarrow \text{onde } \{\hat{d}_c\} \text{ contém os g.l. condensados}$$

Neste algoritmo, excluímos a 7ª linha (DE(JJ) = (RM(J) - DUM)/ SM(J,JJ)), visto que não consideramos nós internos carregados.

Algoritmo de condensacão	**Algoritmo de recuperação**
LL = 0 ! => acrescentamos esta linha c Do condensation operations on lower triangle SE DO K=1,NUM LL=NSIZE-K KK=LL+1 DO L=1,LL DUM = SE(KK,L) / SE(KK,KK) DO M =1, L SE(L,M) = SE(L,M) - SE(KK,M) * DUM ENDDO c RE(L) = RE(L) - RE(KK) * DUM ! => excluímos ENDDO ENDDO c Fill in the upper triangle of SE by symmetry DO K = 1, LL DO L = 1, K SE(L,K) = SE(K,L) ENDDO ENDDO	DO J = 1, NUM JJ = NSIZE – NUM + J DUM = 0.0D0 K = JJ – 1 DO L = 1, K DUM = DUM + SM(J,L) * DE(L) c DE(JJ) = (RM(J) - DUM)/ SM(J,JJ) ! => excluí. ENDDO ENDDO

Figura 2 – Algoritmos de condensação e recuperação de g.l. (Cook *et al.* 1989)

2. DESCRIÇÃO DA TÉCNICA PROPOSTA

Nesta seção, procuramos descrever de forma resumida o programa que implementamos, e que pode ser dividido em quatro etapas:

A – partição hierárquica da malha
B – condensação estática em níveis hierárquicos, gerando macro-elementos
C – solução do sistema de equações (malha constituída de macro-elementos)
D – recuperação hierárquica dos graus de liberdade condensados

3.1 Partição hierárquica da malha

Na etapa (A), utilizamos o programa METIS para realizar as partições da malha (veja a seção 4). Embora seja possível fazer uma implementação genérica, onde o número de níveis hierárquicos seja estabelecido pelo usuário (ou calculado automaticamente segundo algum critério de otimização), por simplicidade adotamos quatro níveis hierárquicos, numerados de I a IV. As partições maiores, de nível hierárquico mais elevado, possuem em média, NELmedio elementos. Assim, a malha é inicialmente dividida em NPiv =NEL/NELmedio partições do nível IV, onde NEL é o número total de elementos finitos triangulares da malha. Cada partição do nível IV é então dividida em KFAT_IV partes, dando origem a partes do nível III, que são divididas em KFAT_III partes do nível II, que são divididas em KFAT_II do nível I, que possuem cerca de NELmedio/(KFAT_IV*KFAT_III*KFAT_II) elementos finitos triangulares.

A decisão do número médio de elementos que cada partição do nível IV deve conter é importante, e depende dos fatores KFAT_II a KFAT_IV. Se NELmedio for muito elevado, o tempo de condensação estática pode ser muito elevado, assim como o tamanho das matrizes [SM] que serão geradas em todos os níveis. Por outro lado, se NELmedio for demasiadamente pequeno, problemas podem surgir nos níveis mais baixos, tal como partições com nenhum elemento e partições de geometria inconveniente. Sugerimos que NELmedio e que os KFAT_N sejam tais que, as partições do nível mais baixo (nível I) possuam em média cerca de 31 elementos finitos triangulares. Neste estudo, adotamos NELmedio=2000, e várias combinações de KFAT_N, tais que $(\text{KFAT\_IV*KFAT\_III*KFAT\_II}) \cong 64$, foram experimentadas.

3.2 Condensação estática em níveis hierárquicos, gerando macro-elementos

Os nós internos das partições do nível I são então condensados, dando origem a macro-elementos do nível I (MEs I). Os MEs I são montados (do Inglês, *assembling*) em grupos de KFAT_II, e os nós internos são condensados, dando origem a MEs II, que são montados em grupos de KFAT_III, seus nós internos condensados, dando origem a MEs III, que são montados em grupos de KFAT_IV, seus nós internos condensados dando origem a MEs IV. A cada condensação realizada, a respectiva matriz [SMnível] fornecida pela rotina de condensação (Figura 2) deve ser armazenada. É muito importante observar que a condensação em níveis hierárquicos aqui proposta possibilita uma redução significativa no tempo gasto no processo de condensação, assim como reduz o tamanho das matrizes [SM]. Contudo, ainda assim, o armazenamento em memória RAM de todas as matrizes [SM] tornaria a análise inviável. Por isso, propomos que, após todas as condensações realizadas em cada partição do nível IV, as matrizes [SM] geradas sejam armazenadas em disco, para posterior leitura na fase de recuperação dos graus de liberdade condensados. Como veremos nos resultados numéricos, embora as operações de leitura/gravação dessas matrizes representem uma parcela importante do tempo gasto no processo de condensação/recuperação, ela não compromete os ganhos globais da técnica aqui proposta. A estratégia que adotamos na etapa de condensação é resumida na Figura 3.

3.3 Solução do sistema de equações (malha constituída de macro-elementos)

Uma vez terminado o processo de condensação, passamos a lidar com uma malha constituída por NPiv macro-elementos do nível IV, que possuem apenas nós externos. Estes macro-elementos são tratados como elementos finitos comuns, com a única diferença de que o número de nós por (macro) elemento não é fixo, ou seja, macro-elementos distintos podem

possuir números distintos de nós. Por simplicidade, neste trabalho, o sistema de equações resultante foi resolvido através do Método dos Gradientes Conjugados, utilizando-se uma rotina muito simples, sem pré-condicionamento e sem otimizações.

```
Para cada ME  IV (NPiv vezes)
    Para cada  ME III (KFAT_IV vezes)
        Para cada  ME  II (KFAT_IIII vezes)
            Para cada ME  I (KFAT_II vezes)
                Para cada  triangulo (cerca de NELmedio/( KFAT_IV * KFAT_IV * KFAT_IV ) vezes)
                    Monta a matriz de rigidez do atual triangulo em [SEi]
                Próximo triangulo
                Condensa os g.l. internos de [SEi]
                 Monta [SEi] em [SEii]
            Próximo ME I
            Condensa os g.l. internos de [SEii]
            Monta SEii em [SEiii]
        Próximo ME II
        Condensa os g.l. internos de [SEiii]
        Monta SEiii em [SEiv]
    Próximo ME III
    Condensa os g.l. internos de [SEiv]
    Grava do conjunto das [SMniveis]
Próximo ME IV
```

Figura 3 – Algoritmo para condensação hierárquica

3.4 Recuperação hierárquica dos graus de liberdade condensados

Uma vez calculados os graus de liberdade externos dos macro-elementos do nível IV, os graus de liberdade internos de todos os níveis são recuperados utilizando-se as matrizes [SM] que foram geradas e gravadas em disco durante a condensação. O processo de recuperação dos graus de liberdade condensados está resumido na Figura 4.

```
Para cada ME  IV (NPiv vezes)
    Lê o conjunto de matrizes [SM] associado ao atual ME IV
    Busca os g.l. externos do atual ME IV no vetor de deslocamentos
    Recupera os g.l. internos do atual ME IV
    Distribui  os g.l. internos do atual ME IV no vetor de deslocamentos
    Para cada  ME III (KFAT_IV vezes)
        Busca os g.l. externos do atual ME III no vetor de deslocamentos
        Recupera os g.l. internos do atual ME III
        Distribui  os g.l. internos do atual ME III no vetor de deslocamentos
        Para cada  ME  II (KFAT_IIII vezes)
            Busca os g.l. externos do atual ME II no vetor de deslocamentos
            Recupera os g.l. internos do atual ME II
            Distribui  os g.l. internos do atual ME II no vetor de deslocamentos
            Para cada ME  I (KFAT_II vezes)
                Busca os g.l. externos do atual ME I no vetor de deslocamentos
                Recupera os g.l. internos do atual ME I
                Distribui  os g.l. internos do atual ME I no vetor de deslocamentos
            Proximo ME I
        Proximo ME II
    Proximo ME III
Proximo ME IV
```

Figura 4 – Algoritmo para recuperação hierárquica

4. O PACOTE METIS

Uma das etapas mais difíceis no processo descrito na seção 3 é a partição da malha. Conforme dissemos no item 3.1, aqui nós utilizamos o pacote Metis, que é distribuído gratuitamente por (Karypis G. e Kumar V., 1998) e que se constitui de diversas sub-rotinas para tratamento dos mais diversos problemas que envolvam o particionamento de malhas, grafos e matrizes até mesmo para problemas de grande porte. Dentre as qualidades que a Metis proporciona, podemos citar a sua velocidade e qualidade dos particionamentos gerados independentemente da geometria da malha utilizada. A metodologia dos algoritmos de particionamento da Metis baseia-se na minimização dos nós de interface entre uma partição e outra. Esta característica torna-se particularmente atraente do ponto de vista da computação paralela, onde é desejável o mínimo de comunicação, normalmente associada aos nós de interface, entre as partições de um domínio. No presente estudo, pretendeu-se expandir a funcionalidade desta biblioteca, pois os particionamentos são feitos recursivamente e em níveis hierárquicos. No presente estudo empregou-se a sub-rotina METIS_PartMeshDual para particionamento de malhas por elementos. É importante observar que a qualidade dos macro-elementos construídos a partir da Metis dependerá de critérios para os quais a biblioteca originalmente não foi desenvolvida, mas que, por conseguinte são automaticamente satisfeitos. Por exemplo, ao se minimizar o número de nós que participam de mais de uma partição (nós de interface), automaticamente maximiza-se o número de nós internos às partições. Observou-se que alguns cuidados devem ser tomados para casos onde a "sub-malha" a ser submetida à Metis torna-se pequena demais para produzir-se uma partição válida, desta forma, neste estudo nos restringimos a um máximo de 4 níveis de condensação para que pudesse ser evitado tal cenário.

4.1 Ilustrando o funcionamento do pacote Metis

Considere uma malha quadrada, dividida em 500x500 células com dois elementos triangulares cada, como ilustra a Figura 5. Assim, a malha possui 500 000 elementos, 251 001 nós e 502 002 graus de liberdade.

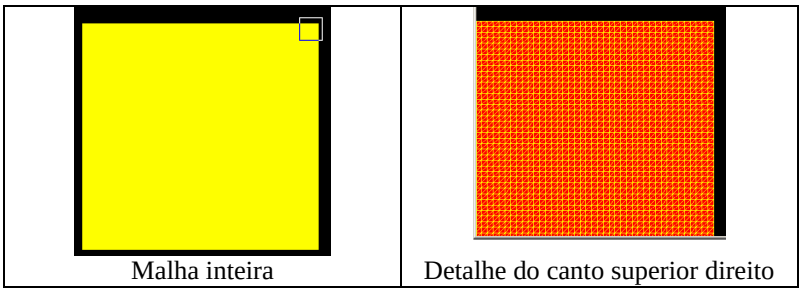


Figura 5 - Malha com 500x500 células

Aqui vamos considerar uma partição em quatro níveis hierárquicos, com:

$$KFAT\_IV = KFAT\_III = KFAT\_II = KFAT\_I = 4$$

As partições de nível IV possuem, em média, 2000 elementos cada. Assim, temos 250 partições do nível IV, como ilustra a Figura 6.

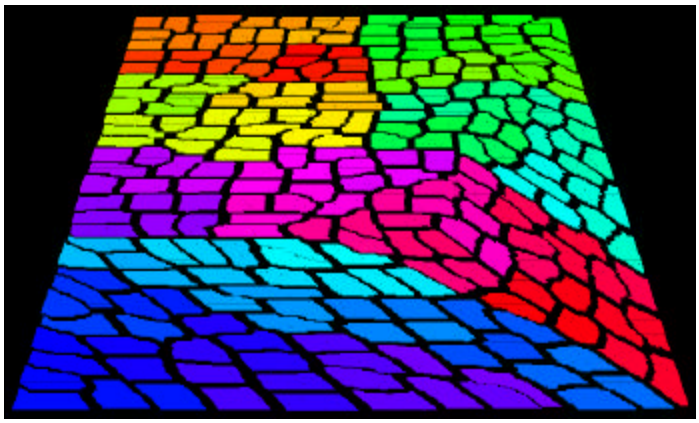


Figura 6 - 250 partições do nível IV, com ~2000 elementos cada

Cada uma das partições do nível IV é dividida em quatro partes, dando origem a 1000 partições do nível III, como ilustra a Figura 7.

Figura 7 - 1000 partições do nível III, com ~500 elementos cada

Cada uma das partições do nível III é dividida em quatro partes, dando origem a 4000 partições do nível II, como ilustra a Figura 8.

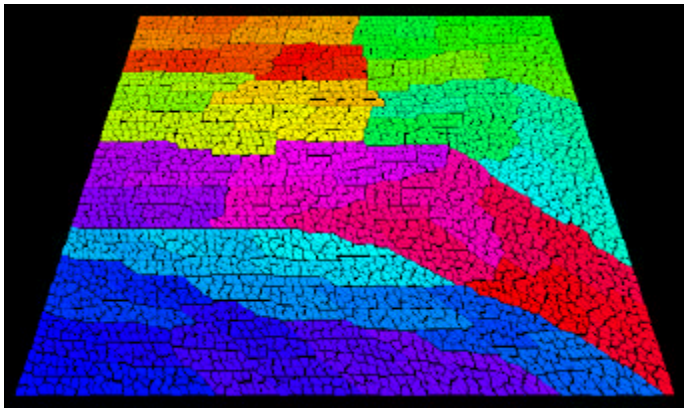


Figura 8 – 4000 partições do nível II com ~125 elementos cada

Finalmente, cada uma das partições do nível II é dividida em quatro partes, dando origem a 16000 partições do nível I, como ilustra a Figura 9.

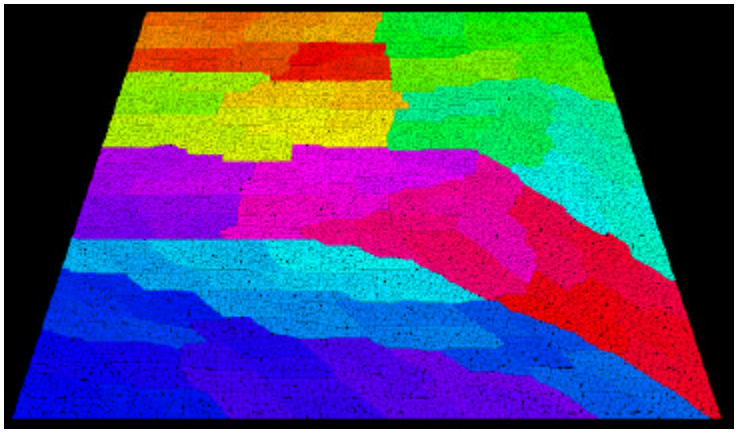


Figura 9 – 16000 partições do nível I com ~31 elementos cada

3. ESTUDO DE CASO

Neste estudo consideramos um painel quadrado, isotrópico elástico, com a base fixa e submetido a uma tensão de tração no topo, como mostra a Figura 10, e que é discretizado através de elementos triangulares lineares.

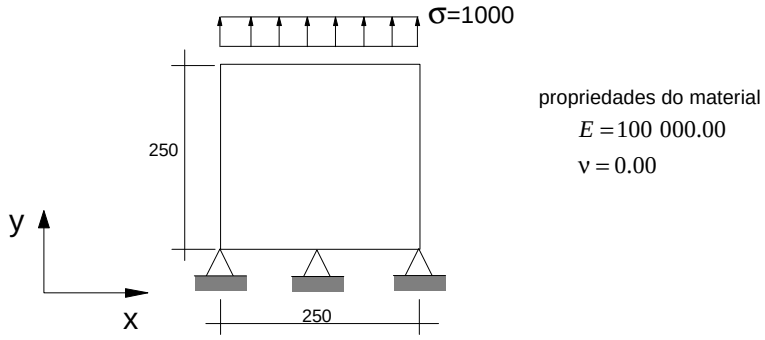


Figura 10 – Painel isotrópico com base fixa e carregamento aplicado no topo

O carregamento aplicado resulta em um estado de tensão constante ($\sigma_y = 1000$; $\sigma_x = \tau_{xy} = 0$) e que é exatamente representado pela análise de elementos finitos para qualquer nível de refinamento de malha. Assim, as respostas esperadas são:

- deslocamentos nulos na direção de x;
- deslocamentos verticais variando linearmente de 0.0 na base, a 2.5 no topo.

Consideramos 11 malhas com NxN células e, para cada malha, resolvemos o problema da forma convencional (obtendo todas as incógnitas nodais através do método dos gradientes conjugados) e também através da estratégia de condensação hierárquica. Para a segunda abordagem, adotamos NELmédio = 2000, KFAT_IV = 2, KFAT_III = 4 e KFAT_II = 8. Na Tabela 1 indicamos o número N de divisões em cada direção e o número total de equações resultantes (NEQ). Também indicamos o número de equações que restam após o processo de condensação hierárquica, e que serão resolvidas através do Método dos Gradientes Conjugados (NEQ_{GC}) considerando-se NPiv macro-elementos do nível IV.

Tabela 1 - Número de equações para as malhas consideradas

N	NEQ	NEQ_{GC}	N	NEQ	NEQ_{GC}
50	5100	398	600	721200	51968
100	20200	1560	700	981400	70734
200	80400	6030	800	1281600	91858
300	180600	13294	900	1621800	115786
400	320800	23232	1000	2002000	143548
500	501000	36638	-	-	-

Para os sistemas de equações resolvidos através do Método dos Gradientes Conjugados (GC), adotamos uma tolerância de 10^{-6} e, como esperado, obtivemos as respostas dentro da tolerância adotada para todas as malhas e para as duas abordagens consideradas. Na abordagem de condensação hierárquica, as matrizes [SM] foram armazenadas em arquivos binários e, para a abordagem convencional, não foi necessária a utilização de disco. Todos os testes foram efetuados em um PC Intel Pentium 4-HT 2.6 GHz e 2Gb de memória RAM

Na Tabela 2 indicamos diversos tempos medidos em nossas análises, como descrevemos a seguir:

T_{div} - tempo gasto no processo de divisão hierárquica da malha

$T_{ass+cond}$ - tempo gasto na etapa de *assembling* e condensação dos graus de liberdade de todos os níveis hierárquicos (inclui o tempo de gravação das matrizes [SM])

T_{rec} - tempo gasto na recuperação dos graus de liberdade condensados (inclui o tempo de leitura das matrizes [SM])

T_{SM} - tempo gasto na gravação das matrizes [SM]

$T_{GC\text{-}m.e.}$ - tempo gasto na solução do sistema que resulta da malha constituída por NPiv macro-elementos do nível IV (solução via GC)

$T_{hier\text{-}tot}$ - tempo total de análise para a abordagem de condensação hierárquica

$T_{GC\text{-}puro}$ - tempo total de análise para o esquema convencional

Tabela 2 - Tempos parciais das análises realizadas (segundos)

	Análise via estratégia de condensação hierárquica						Anál. conv.
N	T_{div}	$T_{ass+cond}$	T_{rec}	T_{SM}	$T_{GC\text{-}m.e.}$	$T_{hier\text{-}tot}$	$T_{GC\text{-}puro}$
50	4.69E-02	1.265625	0.15625	0.75	0.35938	1.875	0.671875
100	0.171875	4.5625	0.578125	2.78125	2.28125	7.70313	4.812500
200	0.71875	18.125	2.28125	11.0938	18.4531	39.9688	41.17188
300	1.625	40.921875	5.078125	25.2344	57.7969	106.359	137.4219
400	3.109375	73.765625	9.25	45.8438	135.703	223.438	321.1719
500	4.890625	115.8125	14.78125	71.5313	267.906	405.922	628.9688
600	7.203125	168.078125	21.8125	103.672	447.469	648.438	1062.062
700	9.65625	230.703125	30.078125	143.047	718.094	993.75	1697.859
800	13.015625	298.078125	36.171875	186.047	1045.02	1398.88	2515.234
900	16.3125	372.28125	46.15625	229.813	1487.5	1930.78	3333.922
1000	20.40625	462.125	55.75	283.563	2186.06	2734.875	4451.484

Na Figura 11, plotamos os valores indicados na Tabela 1 e também o número de incógnitas nodais eliminadas através dos processos hierárquicos (NEQ_{hierq}=NEQ-NEQ_{GC}). No eixo horizontal indicamos o número de divisões em cada direção (N) e, no eixo vertical, indicamos números de equações (NEQ, NEQ_{hierq} e NEQ_{GC}). Notamos que NEQ apresenta uma variação acentuadamente não linear em relação a N, e que essa variação é acompanhada de perto por NEQ_{hierq} , de forma que NEQ_{GC} varia quase linearmente.

Na Figura 12 plotamos os tempos de processamento hierárquico, T_{hierq} (que inclui os tempo de divisão da malha, condensação estática, recuperação dos graus de liberdade condensados e leitura/gravação das matrizes [SM]) e o tempo de solução das NEQ_{GC} equações via GC ($T_{GC\text{-}m.e.}$). Também plotamos os tempo total de análise para a abordagem de condensação hierárquica ($T_{hier\text{-}tot}$) e a convencional ($T_{GC\text{-}puro}$). Aqui, chamamos a atenção para o fato de que T_{hierq} varia linearmente com NEQ, enquanto que $T_{GC\text{-}m.e.}$ e $T_{GC\text{-}puro}$ variam de forma desfavoravelmente não linear. Como conseqüência, que a relação $T_{GC\text{-}m.e.}$ / T_{hierq} aumenta com NEQ. Contrastando as Figuras 11 e 12 e os processos de condensação e GC verificados na abordagem hierárquica, é interessante observar que houve uma inversão das variações lineares (ou quase lineares) e não lineares. A Figura 12 indica também o tempo total de análise para a abordagem convencional ($T_{GC\text{-}puro}$). Notamos que a abordagem que propomos é significativamente vantajosa quando tratamos de grandes sistemas de equação.

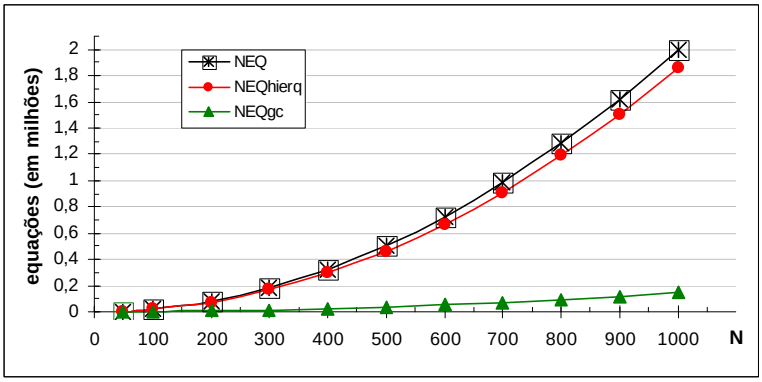


Figura 11 – Número total de equações e equações resolvidas via condensação e GC

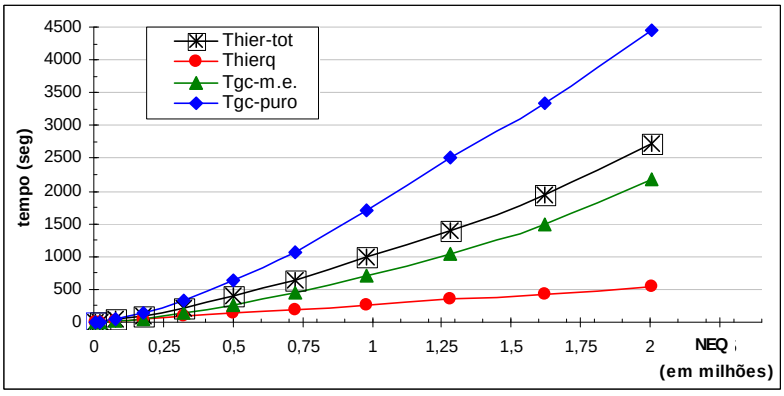


Figura 12 – Tempos de processamento hierárquico e de solução via GC

Na Figura 13, comparamos o volume de equações resolvidas através da condensação estática (NEQ_{hier}) e do GC com macro-elementos ($NEQ_{GC\text{-}m.e.}$). Registramos também a participação destes processos no consumo total de tempo de solução. Podemos observar que,

para todos os níveis de refinamento, cerca de 92% das incógnitas foram eliminadas e posteriormente recuperadas pelo processo de condensação estática, enquanto que para o GC restaram apenas cerca de 8% das equações a resolver. Notamos também que, para as três primeiras malhas, o tempo gasto pelo GC é menor que o gasto pela condensação/recuperação e, a partir da quarta malha, o processo de condensação/recuperação é mais rápido, com o GC consumindo cerca de 80% do tempo de solução para a malha mais refinada. A figura compara também os tempos totais de análise para a solução dos problemas através do processo convencional e da abordagem hierárquica que propomos. Notamos que, para as duas primeiras malhas (M50x50 e M100x100), o processo convencional é mais rápido, e a partir dai passa a ser mais lento, com tempo de processamento cerca de 80% maior para a malha M800x800. É interessante observar que, a partir daí, observa-se um decréscimo da relação $T_{GC\text{-}puro}/T_{hier\text{-}tot}$. Esse comportamento sugere que mais níveis hierárquicos deveriam ter sido utilizados a partir da malha M900x900.

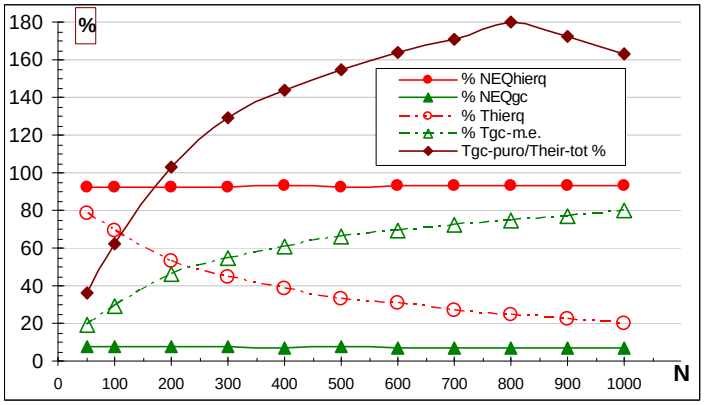


Figura 13 – Participações GC x processos hierárquicos

Analisando os resultados apresentados nas Tabelas 1 e 2, concluímos que o Método dos Gradientes conjugados é mais lento quando resolve sistemas de equações que resultam de malhas de macro-elementos do que malhas compostas por triângulos. Por exemplo, para a malha M300x300, temos que, para a abordagem convencional, o Método dos Gradientes Conjugados resolve cerca 180 mil equações (NEQ=180 600) em cerca de 137 segundos ($T_{GC\text{-}puro}$=137,42), enquanto que para a malha M1000x1000, o GC resolve cerca de 143 mil equações (NEQ_{GC}=143 548) em cerca de 2 mil segundos ($T_{GC\text{-}me}$=2 186,06). A fim de justificar este efeito, realizamos um estudo avaliando o número de operações de soma e multiplicação realizadas em uma solução via GC para elementos triangulares e para macro-elementos com muitos nós. Tal estudo é apresentado no Apêndice A e conclui que, para duas malhas com o mesmo número de incógnitas nodais e constituídas por elementos com número de nós distintos, o GC tende a ser mais rápido para a malha cujos elementos que possuam menor número de nós. Vale observar que a rotina que de multiplicação matriz-vetor que utilizamos neste trabalho é muito simples, não prevendo otimizações. Assim, acreditamos que os resultados possam ser significativamente aprimorados a partir da utilização de rotinas mais adequadas para a realização dos produtos matriz-vetor.

5. CONCLUSÕES

Neste trabalho apresentamos uma técnica de decomposição de domínio visando o aumento do desempenho computacional na solução de sistemas de equações que resultam de análises numéricas via Método dos Elementos Finitos. Os resultados preliminares indicam

que a técnica pode proporcionar significativos ganhos no tempo de processamento, sem apresentar desvantagens no consumo de memória. A técnica é muito favorável ao processamento paralelo, pois as etapas de divisão de malha, condensação estática e recuperação dos graus de liberdade podem ser divididas em tarefas independentes, sem a necessidade de troca de informação entre elas. Como futuras atividades, citamos:

- desenvolvimento de critérios que permitam que o número de níveis hierárquicos, o tamanho de cada partição e o número de subdivisões em cada nível sejam automaticamente estabelecidos;
- implementação da técnica para problemas em três dimensões;
- implementação da técnica para malhas com cargas e prescrições aplicadas em nós internos;
- implementação de uma rotina otimizada para solução via GC.

Agradecimentos

Este trabalho foi desenvolvido com apoio do CPNq, através do projeto CNPq/MCT 522692/95-8. O Dr. Marcos A. D. Martins agradece especialmente o apoio do CNPq através do projeto CT-PETRO/PROSET 500.196/02-8.

REFERÊNCIAS

Cook, R. D., Malkus, D. S. e Plesha, M. E, 1989. Concepts and Applications of Finite Element Analysis, 3ª edição, John Wiley & Sons;

Ellwanger, R.J., 1989, *Análise Estática e Dinâmica com Subestruturação em Múltiplos Níveis*, Tese D.Sc, COPPE/Universidade Federal do Rio de Janeiro.

Fragakis,Y., Papadrakakis, M., 2003, The Mosaic of High Performance Domain Decomposition Methods for Structural Mechanics: Formulation, Interrelation and Numerical Efficiency of Primal and Dual Methods, Comp. Meth Appl. Mech. Eng., v.192, pp 3799-3830.

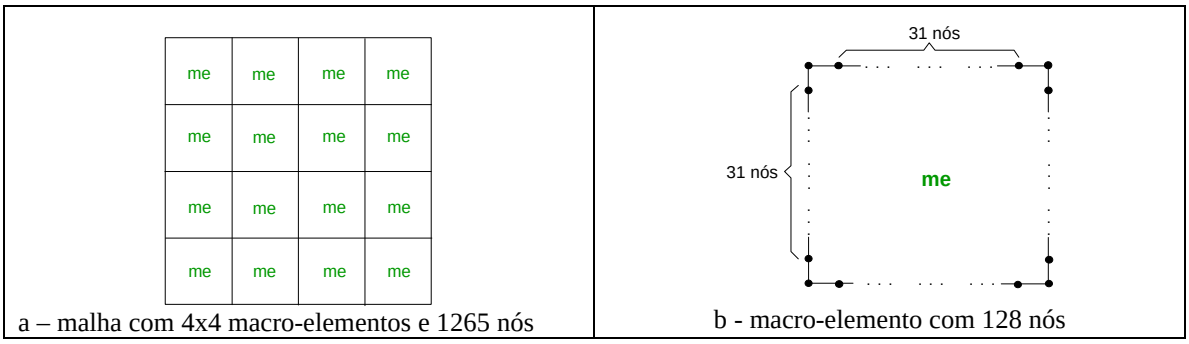
Karypis G. e Kumar V., 1998, *Metis 4.0: Unstructured Graph Partitioning and Sparse Matrix Ordering System*. Technical report, Department of Computer Science, University of Minnesota, Minneapolis; (http://www-users.cs.umn.edu/~karypis/metis).

Smith, B., Bjorstad, P. e Gropp, W., 2004, Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations, Cambridge University Press.

APÊNDICE A

A.1 CASO A – Malha com 1265 nós e 16 macro-elementos

Considere a malha da Figura A.1, com 16 macro-elementos e 1265 nós. Cada macro-elemento possui 128 nós (externos) como detalha a Figura A.1b. Vamos considerar o caso de 2 graus de liberdade por nó, de modo que a malha possui NEQ=2530 equações e o macro-elemento possui ND=256 deslocamentos.



a – malha com 4x4 macro-elementos e 1265 nós

b - macro-elemento com 128 nós

Figura A.1 – malha com 4x4 macro-elementos

Considere o produto da matriz de rigidez de um macro-elemento pelo seu vetor de deslocamento:

$$[Sme]_{256x256}\{Dme\}_{256x1}$$

É fácil concluir que a multiplicação uma linha de [*Sme*] pelo vetor {*Dme*} envolve 256 operações de multiplicação (aqui abreviada como MTP) e 256[1] operações de soma (abreviada como SUM). Então, como a matriz [*Sme*] possui 256 linhas, o esforço computacional para fazer o produto matriz-vetor é:

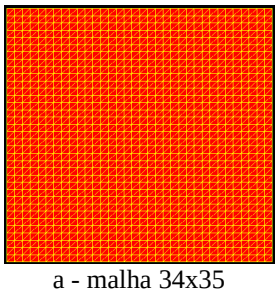
$$ESF_{matvet} = 256\ (256\ MTP + 256\ SUM) = 65\,536\ (MTP + SUM)$$

Assim, como a malha possui 16 macro-elementos, em uma iteração GC o esforço computacional será:

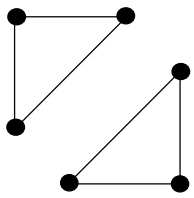
$$ESF_{iteração} = 16*65536\ (MTP + SUM) = 1\,048\,576\ (MTP + SUM)$$

A.2 CASO B – Malha com 1260 nós e 2380 triângulos

Considere a malha da Figura A.2, com 34x35 células contendo 2 triângulos cada uma. Assim, temos 2380 elementos e 1260 nós. Os triângulos possuem 3 nós, como detalha a Figura A.2b. Considerando 2 graus de liberdade por nó, temos NEQ=2520 e ND=6. Podemos considerar que o sistema de equações que resulta desta malha é equivalente ao do exemplo anterior.



a - malha 34x35



b - triangulos de 3 nós

Figura 2 – Malha com 2380 triângulos e 1260 nós

Considere o produto da matriz de rigidez de um triangulo pelo seu vetor de deslocamento:

[1] analiticamente são 255 somas, mas computacionalmente são 256, sendo que o primeiro produto é somado a zero

$$[Se]_{6x6}\{De\}_{6x1}$$

A multiplicação de uma linha de [*Se*] pelo vetor {*De*} envolve 6 operações de multiplicação (MTP) e 6 operações de soma (SUM). Então, como a matriz [*Se*] possui 6 linhas, o esforço computacional para fazer o produto acima é:

$$ESF_{matvet} = 6\ (6\ MTP + 6\ SUM) = 36\ (MTP + SUM)$$

Assim, como a malha possui 2380 elementos, em uma iteração GC o esforço computacional será:

$$ESF_{iteração} = 2\,380 * 36\ (MTP + SUM) = 85\,680\ (MTP + SUM)$$

A.3 Conclusão

Os sistemas de equações dos exemplos podem ser considerados equivalentes, e é de se esperar que resultem em números de iterações GC semelhantes. Contudo, notamos que há uma significativa diferença de esforço computacional por iteração. Assim, concluímos que o GC tende a ser mais lento quando lida com elementos que possuam muitos nós (ND elevado).