Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería. Vol. 9, 3, 271–297(1993)

AVALIAÇÃO DE ESTRATÉGIAS COMPUTACIONAIS PARA O MÉTODO DOS ELEMENTOS FINITOS EM COMPUTADORES VETORAIS

ALVARO L.G.A. COUTINHO
JOSE L.D. ALVES
LUIZ LANDAU
NELSON F.F. EBECKEN

Centro de Computação Paralela e
Laboratórios de Métodos Computacionais em Engenharia,
Programa de Engenharia Civil, COPPE/Universidade Federal do Rio de Janeiro,
Cidade Universitária – Centro Tecnologia Bloco B –
Sala 100, Caixa Postal 68506, Rio de Janeiro, RJ 21945,
Brasil, Fax: (5521) 290-6626 - E - mail: coc06001 @ ufrj.bitnet

SUMÁRIO

Este trabalho apresenta uma avaliação de estratégias computacionais iterativas para a solução de sistemas de equações lineares provenientes da aplicação do método dos elementos finitos à problemas tridimensionais de mecânica do contínuo. Emprega-se o método dos gradientes conjugados, acelerado por diversos precondicionadores, diagonal, bloco-diagonal nodal, elemento-por-elemento e multi-nivel. Discute-se també na implementação em computadores vetorais através de técnicas elemento-por-elemento. São apresentados diversos exemplos de interesse prático onde procura-se evidenciar o alto desempenho das estratégias propostas quando comparadas com métodos diretos de solução, também vetorizados.

SUMMARY

This work presents an evaluation of iterative linear equation solvers for finite element systems of equations arising from three-dimensional continuum mechanics problems. The conjugate gradient method is employed, accelerated by several preconditioners, diagonal, nodal block diagonal, element-by-element and multi-level. The implementation on vector computers of element-by-element techniques is also addressed. Some examples of practical interest are presented, showing the high performance achieved by the proposed strategies, when compared to a vectorized direct skyline solver.

Recibido: Junio 1992

 ISSN 0213–1315

INTRODUÇÃO

A formulação cinemática do método dos elementos finitos para a solução de problemas da mecânica do contínuo (lineares ou não-lineares, estáticos ou dinâmicos), conduz à um sistema de equações algébrico da forma (HUGHES, 1987a),

$$\mathbf{A}\,\mathbf{x} = \mathbf{b} \tag{1}$$

onde $\mathbf{A}$ é uma matriz esparsa, usualmente simétrica positiva definida, $\mathbf{x}$ é o vetor de incógnitas nodais e $\mathbf{b}$ o vetor de termos independentes (leva em consideração a presença de forças de volume, de superfície e das condições de contorno). A Equação (1) é resolvida geralmente por um método direto de solução baseado no processo de eliminação de Gauss (GOLUB e VAN LOAN, 1989), onde a matriz $\mathbf{A}$ pode ser armazenada em banda, perfil ou esparso (CAREY e ODEN, 1984). Entretanto, as estratégias de solução direta podem se tornar inviáveis para a análise de problemas envolvendo um número grande de equações devido principalmente ao enorme esforço computacional e demanda de memória.

Na Tabela I encontram-se estimativas do esforço computacional e da demanda de memória de métodos diretos de solução (AXELSSON e BARKER, 1984). Deve-se notar que n é o número de equações e m a largura de banda da matriz $\mathbf{A}$.

TAREFA	ESFORÇO COMPUTACIONAL (FLOPS)	MEMÓRIA
Fatoração	$\frac{1}{2}\,nm^2$	nm
Retro-substituição	$2\,nm$	nm

Tabela I. Estimativa de Custos de Métodos Diretos de Solução (flops = 1 operação em ponto flutuante por segundo).

Os dados da Tabela I são particularmente importantes em problemas tridimensionais onde a largura de banda pode ser da mesma ordem de grandeza do número de equações.

Métodos iterativos de solução requerem comparativamente menos memória, no entanto o número de operações de ponto flutuante para atingir a solução não é conhecido a priori, e depende de características próprias do sistema de equações. Por outro lado, estes métodos são adequados às novas arquiteturas de computadores onde estão disponíveis facilidades de vetorização e eventualmente paralelização das diferentes tarefas do processo de solução. Sendo assim, este trabalho apresenta alguns resultados do desenvolvimento e aplicação de métodos iterativos para a solução de grandes problemas discretizados pelo método dos elementos finitos utilizando as facilidades de

vetorização disponíveis no computador IBM 3090/VF instalado no Centro de Pesquisas da PETROBRÁS S.A. (CENPES).

Este trabalho é organizado da seguinte forma: na seção seguinte apresenta-se o método iterativo dos Gradientes Conjugados (MGC), utilizado neste trabalho. A penúltima seção reúne algumas considerações sobre o ambiente computacional e as técnicas de vetorização utilizadas. Finalmente, a última seção apresenta algumas aplicações numéricas que evidenciem a eficiência computacional das estratégias adotadas.

O MÉTODO DOS GRANDES CONJUGADOS PRE-CONDICIONADO

O Método dos Gradientes Conjugados foi inicialmente apresentado por HESTENES e STIEFEL (1952) e explora o fato de que a solução do sistema de equações (1), equivale à minimização do funcional quadrático,

$$Q = \frac{1}{2}\,\mathbf{x}^t\,\mathbf{A}\,\mathbf{x} - \mathbf{x}^t\,\mathbf{b} \tag{2}$$

Assim, processos iterativos utilizados para minimizar o funcional Q, conduzem à solução do sistema de equações algébricas associado a este funcional (GOLUB e VAN LOAN, 1989). Nos algoritmos de solução que empregam o Método Iterativo dos Gradientes Conjugados a matriz dos coeficientes atua unicamente como operador linear fornecendo uma maneira extremamente elegante de explorar a esparsidade desta matriz. A eficiência do método está intrinsicamente relacionada com as características próprias do operador **A** (AXELSSON e BARKER, 1984), isto é, seus autovalores. Como a matriz **A** é positiva definida, as superfícies representadas pela equação (2) são elipsóides. Métodos numéricos para minimização de um funcional tendem a ser mais eficientes quando as superfícies se aproxima da forma esférica. Neste caso a razão entre o maior e o menor autovalores do espectro, λ_n e λ_1 respectivamente,

$$k(\mathbf{A}) = \frac{\lambda_n}{\lambda_1}, \tag{3}$$

é denominada razão espectral ou número de condicionamento espectral, se aproxima da unidade. Caso contrário, quando a razão espectral se apresenta muito maior que a unidade, torna-se necessário a utilização de uma técnica de pré-condicionamento que tem como efeito resultar em um funcional onde as superfícies na vizinhança da solução sejam significativamente menos excêntricas que a original. Pode-se dizer então que para matrizes simétricas positivas definidas a taxa de convergência de um esquema iterativo pode ser avaliada através da razão espectral.

Logo, com o objetivo de atingir uma convergência mais rápida, o Método dos Gradientes Conjugados pode ser aplicado a um sistema equivalente ao original, mas que tenha sido pré-condicionado. Pré-multiplicando-se a equação original do sistema pela matriz não singular **B**,

$$\mathbf{B}^{-1}\mathbf{A}\,\mathbf{x} = \mathbf{B}^{-1}\mathbf{b} \tag{4}$$

e aplicando-se o Método do Gradientes Conjugados ao sistema pré-condicionado acima, obtém-se o algoritmo dos Gradientes Conjugados Pré-condicionado, cujos principais passos são apresentados na tabela a seguir:

Tabela II.2.1 - Algoritmo dos Gradientes Conjugados Pré-condicionado

Passo 1: Inicializar

$$
\begin{aligned}
\mathbf{k} &= o \\
\mathbf{x}_0 &= \text{Dado (usualmente } \mathbf{x}_0 = 0) \\
\mathbf{r}_0 &= \mathbf{b} - \mathbf{A}\,\mathbf{x}_0 \\
\mathbf{p}_0 &= \mathbf{z}_0 = \mathbf{B}^{-1}\mathbf{r}_0
\end{aligned}
$$

Passo 2: Atualizar solução e resíduo

$$
\begin{aligned}
\alpha_k &= \frac{\mathbf{r}_k^t \cdot \mathbf{z}_k}{\mathbf{p}_k^t \cdot \mathbf{A}\,\mathbf{p}_k} \\
\mathbf{x}_{k+1} &= \mathbf{x}_k + \alpha_k\,\mathbf{p}_k \\
\mathbf{r}_{k+1} &= \mathbf{r}_k - \alpha_k\,\mathbf{A}\,\mathbf{P}_k
\end{aligned}
$$

Passo 3: Verificar convergência

$$
\text{Se } \left(\mathbf{r}_{k+1}^t \cdot \mathbf{r}_{k+1}\right)^{\frac{1}{2}} \leq \delta \left(\mathbf{r}_0^t \cdot \mathbf{r}_0\right)^{\frac{1}{2}} \rightarrow \text{fim}
$$

Passo 4: Atualizar direção conjugada

$$
\begin{aligned}
\mathbf{z}_{k+1} &= \mathbf{B}^{-1}\mathbf{r}_{k+1} \\
\beta_k &= \frac{\mathbf{r}_{k+1}^t \cdot \mathbf{z}_k}{\mathbf{r}_k \cdot \mathbf{z}_k} \\
\mathbf{p}_{k+1} &= \mathbf{z}_{k+1} + \beta_k\,\mathbf{p}_k \\
k &= k + 1
\end{aligned}
$$

vá para o passo 2

Em geral, uma matriz de pré-condicionamento $\mathbf{B}$ para ser eficiente, deve atender alguns requisitos básicos: seus componentes devem ser facilmente determinados além de não necessitar armazenamento excessivo em relação à matriz $\mathbf{A}$; a solução do sistema $\mathbf{B}\mathbf{z}_k = \mathbf{r}_k$ deve ser mais eficiente que a solução do sistema $\mathbf{A}\,\mathbf{x} = \mathbf{b}$, já que o mesmo será resolvido a cada iteração do algoritmo. Finalmente, deve originar um número de condicionamento espectral reduzido em relação a $k(\mathbf{A})$.

Pode-se mostrar, segundo GOLUB e VAN LOAN (1989), que a taxa de convergência do Método dos Gradientes Conjugados Pré-condicionado é proporcional ao quociente,

$$\frac{\sqrt{k(\hat{\mathbf{A}})} - 1}{\sqrt{k(\hat{\mathbf{A}})} + 1}, \tag{5}$$

Sendo $\hat{\mathbf{A}} = \mathbf{B}^{-1}\mathbf{A}$

TÉCNICAS DE PRÉ-CONDICIONAMENTO

Pré-Condicionador Diagonal

Definindo-se como matriz **B** a diagonal da matriz dos coeficientes **A**

$$\mathbf{B}_D^{-1} = \left[A_{ii}^{-1}\right] \quad i = 1, \ldots, n \tag{6}$$

temos o Pré-condicionamento Diagonal, que devido à sua simplicidade de implementação, baixo número de operações envolvidas e efeito favorável de escalonamento de variáveis de grandezas dimensionais diferentes é muito utilizado, embora na maioria dos casos não tenha um efeito muito satisfatório na redução de $k(\hat{\mathbf{A}})$.

Pré-Condicionador Bloco-Diagonal Nodal

De forma a definir precisamente este pré-condicionador é necessário especificar corretamente uma estrutura de dados. Seja o arranjo `ID (NNODE, NDOF)` onde `NNODE` é o número de pontos nodais da malha e `NDOF` o número de graus de liberdade por nó. Este arranjo mapeia graus de liberdade globais (HUGHES, 1987a). Com isso, pode-se definir o pré-condicionador bloco-diagonal nodal da seguinte forma,

$$\mathbf{B}_{\mathtt{BD}}^{-1} = \begin{cases} \mathbf{A}_{ij}^{-1} \text{ se } k = l \\ \\ \mathbf{0} \quad \text{se } k \neq l \end{cases} \qquad 1 \leq k, l \leq \mathtt{NNODE} \tag{7}$$

onde $\mathbf{A}_{ij}^{-1}$ é a inversa da submatriz de **A** correspondente aos nós `I = ID` (k, i), `J = ID` (l, j), $1 \leq i, j \leq$ `NDOF`. A matriz $\mathbf{B}_{\mathtt{BD}}$ é simétrica, positiva definida e requer uma área de armazenamento de 1/2 `NDOF`×(`NDOF`+1)×`NNODE`. Cada submatriz $\mathbf{A}_{ij}$ é montada a partir de contribuições individuais dos elementos e sua inversa é armazenada para uso posterior durante as iterações do Método dos Gradientes Conjugados.

Pré-Condicionadores Elemento-por-Elemento

Os pré-condiconadores Elemento-por-Elemento, ou EPE, foram projetados para manter a estrutura de dados típica do método dos elementos finitos. Eles foram apresentados inicialmente por HUGHES et al, (1983) e se baseiam na aproximação da inversa de **A** por um produto de matrizes. Além disso, de forma a preservar o posto das matrizes que compõem o produto, os pré-condicionadores EPE são aplicados ao sistema transformado de equações,

$$\overline{\mathbf{A}}\overline{\mathbf{x}} = \overline{\mathbf{b}} \tag{8}$$

onde

$$\overline{\mathbf{A}} = \left(\mathbf{D}^{-1}\right)^T \mathbf{A}\,\mathbf{D}^{-1} \tag{9}$$

$$\overline{\mathbf{x}} = \mathbf{D}\,\mathbf{x} \tag{10}$$

$$\overline{\mathbf{b}} = \mathbf{D}^{-1}\,\mathbf{b} \tag{11}$$

sendo $D_{ii} = A_{ii}^{1/2}$, $i = 1, 2, \ldots n$. Este escalonamento diagonal pode também ser interpretado com um pré-precondicionamento do sistema de equações original. Desta forma, os pré-condicionadores EPE utilizados são definidos pelo produto,

$$\mathbf{B}^{-1}_{\mathrm{EPE}} = \prod_{e=1}^{\mathrm{Nel}} (\mathbf{L}^e) \times \prod_{e=\mathrm{Nel}}^{1} (U^e) \tag{12}$$

onde Nel é o número de elementos da malha. O duplo produto é para preservar a simetria do pré-condicionador. As matrizes $\mathbf{L}^e$ e U^e são obtidas da partição Gauss-Seidel de uma matriz de elemento regularizada da seguinte forma (WINGET E HUGHES, 1985, HUGHES et al, 1987b),

$$\mathbf{L}^e + U^e = \overline{\mathbf{A}}^e - \mathrm{diag}\,(\overline{\mathbf{A}})^e + \mathbf{I} \tag{13}$$

Com esta escolha, a solução do sistema de equações auxiliar do MGC, $\mathbf{z} = \mathbf{B}^{-1}\mathbf{r}$, é efetuada em dois passos. O primeiro na ordem crescente dos elementos e o segundo na ordem inversa, de acordo com a Equação (12). Além disso, deve-se notar que este pré-condicionador não necessita de nenhuma área adicional de memória para armazenar a matriz $\mathbf{B}^{-1}_{\mathrm{EPE}}$.

Pré-Condicionadores Multi-Nível

a) Generalidades

Um dos problemas relacionados com os pré-condicionadores expostos anteriormente é que a ordem do número de condicionamento espectral, $k(\mathbf{A})$, não diminui à medida que o parâmetro de malha também diminui. Em outras palavras, se refinarmos uma dada malha e a solucionarmos através do MGC com quaisquer dos pré-condicionadores expostos anteriormente, o número de iterações necessárias para atingir a convergência será maior do que aquele obtido para a malha inicial. Uma das maneiras de se ultrapassar esta dificultade é a utilização de estratégias multi-nível de solução. Estas se baseiam no fato de que frequentemente um determinado problema é solucionado primeiramente em uma malha grossa e por refinamento (uniforme ou adaptativo), constrói-se uma sequência de malhas,

$$\{\mathbf{A}_k\}_{k=1}^{l} \tag{14}$$

onde $\mathbf{A}_1$ é matriz de coeficientes correspondente à malha inicial (grossa) e $\mathbf{A}_e$ à malha fina, onde se deseja computar a solução. Os fundamentos matemáticos dos

métodos multi-nível podem ser encontrados, por exemplo, em HACKBUSH, (1985) e McCORMICK, (1987). O propósito dessas técnicas é desenvolver um método iterativo no qual a quantidade de trabalho computacional seja diretamente proporcional ao número de incógnitas do sistema de equações. A Tabela II apresenta uma estimativa do trabalho computacional (AXELSSON e BARKER, 1984) de métodos de solução direta tipo Gauss, métodos MGC (com e sem pré-condicionamento) e métodos multinível, para problemas de elementos finitos em duas e três dimensões.

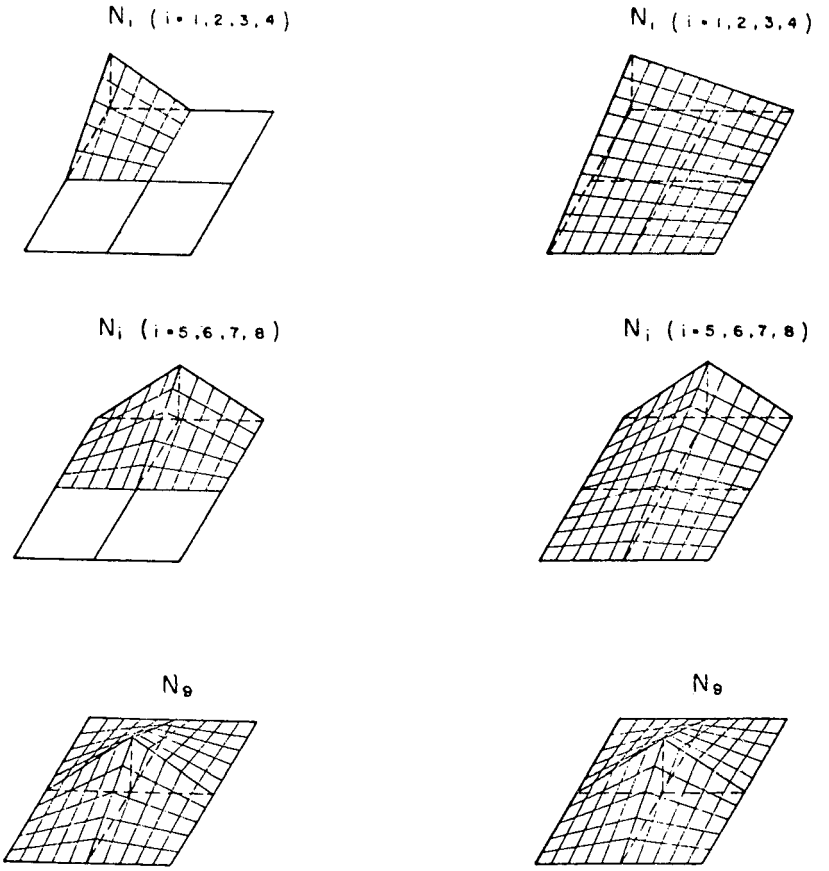
Problemas / Método	Bidimensionais	Tridimensionais
Direto	$0(n^2)$	$0(n^{2.33})$
MGC	$0(n^{1.5})$	$0(n^{1.33})$
MGC c/ pré-cond.	$0(n^{1.25})$	$0(n^{1.17})$
Multi-nível	$0(n)$	$0(n)$

Tabela II. Comparação de Estimativas de Trabalho Computacional.

Conforme foi verificada na prática, por exemplo em PARSONS e HALL, (1990a, 1990b), a obtenção de proporcionalidade entre esforço computacional e número de incógnitas depende de fatores relacionados à implementação, e à plataforma computacional utilizada nos experimentos. Entretanto, a técnica multi-nível parece ser de fato um solucionador muito eficiente quando comparado a outros métodos, tanto diretos quanto iterativos.

Um outro ponto importante é que ao invés de utilizar-se a base nodal padrão para gerar as matrizes de elementos finitos, $\mathbf{A}_k$, $k = 1, 2, \ldots, l$, pode-se usar bases hierárquicas. A Figura 1 apresenta as bases nodais e as bases hierárquicas para elementos quadriláteros.

O aspecto principal das bases hierárquicas é o fato de que quando uma malha correspondente a uma certo nível é refinada, a discretização resultante preserva as informações constantes das discretizações anteriores. Além disso, as matrizes resultantes das bases hierárquicas são melhores condicionadas do que aquelas computadas com a base nodal. Por exemplo, para malhas uniformes em duas dimensões com um tamanho médio de elemento h, o número de condicionamento espectral das matrizes obtidas com bases hierárquicas é proporcional a $|\log h|^2$, enquanto que para bases nodais a estimativa é de h^{-2}, (BANK et al, 1988, 1989 e YSERENTANT, 1986).



(a) Base Nodal (b) Base Hierárquica

Figura 1. Funções de interpolação bidimensionais

b) Propriedades Geométricas das Bases Hierárquicas

Nesta seção apresenta-se a relação entre as bases hierárquicas de diferentes malhas, considerando o fato de que estas são refinamentos sucessivos de uma malha grossa, conforme mostrado na Figura 2.

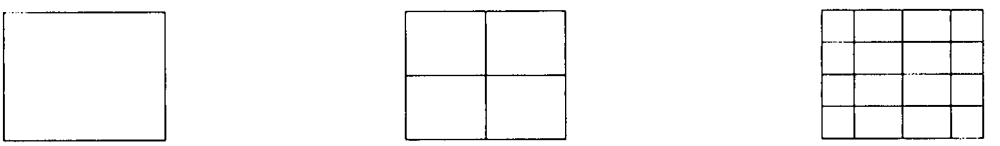


Figura 2. Sequência de Malhas.

Denotando-se por φ_i^1 a i-ésima função de interpolação da malha 1, e como o espaço de aproximação da malha 1 inclui o espaço da malha 0, as funções de interpolação do nível 0 podem ser escritas como uma combinação linear das funções de interpolação da malha 1. Por exemplo, considere um elemento da malha 0 e os quatro elementos produzidos por refinamento uniforme destes, que pertenecem à malha 1. Usando o esquema de numeração da Figura 3, são válidas as seguintes relações

$$\varphi_1^0 = \varphi_1^1 + \frac{1}{2}(\varphi_5^1 + \varphi_8^1) + \frac{1}{4}\varphi_9^1 \tag{15}$$

$$\varphi_2^0 = \varphi_2^1 + \frac{1}{2}(\varphi_5^1 + \varphi_6^1) + \frac{1}{4}\varphi_9^1 \tag{16}$$

$$\varphi_3^0 = \varphi_3^1 + \frac{1}{2}(\varphi_6^1 + \varphi_7^1) + \frac{1}{4}\varphi_9^1 \tag{17}$$

$$\varphi_4^0 = \varphi_4^1 + \frac{1}{2}(\varphi_7^1 + \varphi_8^1) + \frac{1}{4}\varphi_9^1 \tag{18}$$

Apesar destas relações serem válidas somente para um elemento, estas podem ser facilmente extendidas para todo o dominio. Relações similares podem ser construídas para outros tipos de elementos:

Triângulos

$$\varphi_1^0 = \varphi_1^1 + \frac{1}{2}(\varphi_4^1 + \varphi_6^1) \tag{19}$$

$$\varphi_2^0 = \varphi_2^1 + \frac{1}{2}(\varphi_4^1 + \varphi_5^1) \tag{20}$$

$$\varphi_3^0 = \varphi_3^1 + \frac{1}{2}(\varphi_5^1 + \varphi_6^1) \tag{21}$$

Hexaedros

$$\varphi_1^0 = \varphi_1^1 + \frac{1}{2}\varphi_{12}^1 + \frac{1}{2}\varphi_9^1 + \frac{1}{2}\varphi_{17}^1 + \frac{1}{4}(\varphi_{21}^1 + \varphi_{23}^1 + \varphi_{22}^1) + \frac{1}{8}\varphi_{27}^1 \tag{22}$$

$$\varphi_2^0 = \varphi_2^1 + \ldots \tag{23}$$

De forma geral as relações entre funções de interpolação de malhas grossas e funções de interpolação de malhas finas podem ser escritas como,

$$\varphi_i^0 = \sum_j X_{i,j}\ \varphi_j^1 \tag{24}$$

onde $X_{i,j}$ é o peso entre a função φ_i^0 e φ_j^1. Se $i = j$, então $X_{i,j} = 1$, por exemplo. Apesar destas relações serem complexas, sua implementação pode se consideravelmente simplificada pela utilização de uma estrutura de dados apropriada. Estas podem ser encontradas em DEVLOO (1985), para elementos quadriláteros, RIBEIRO (1991) para triângulos e DEVLOO (1991) para hexaedros.

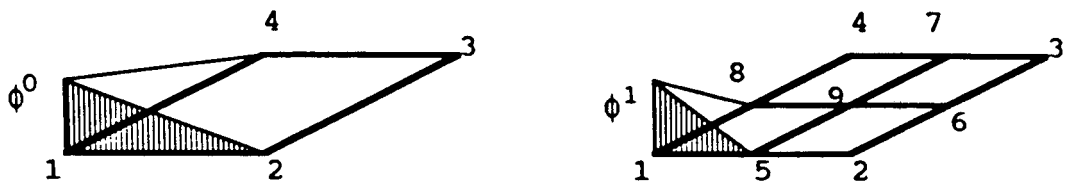


Figura 3. Elemento Quadrilátero em Duas Dimensões.

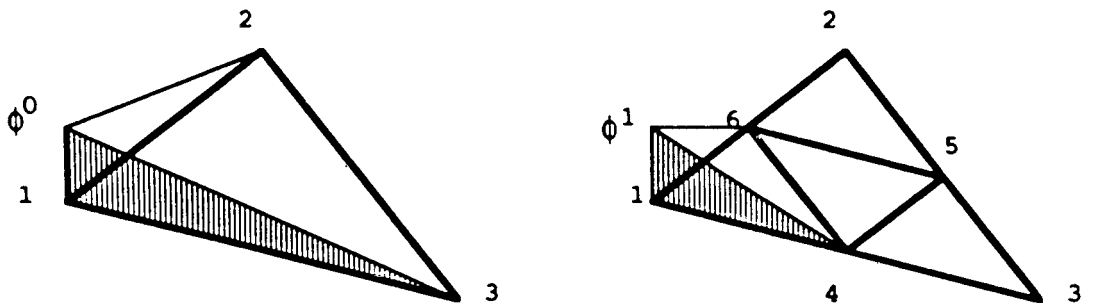
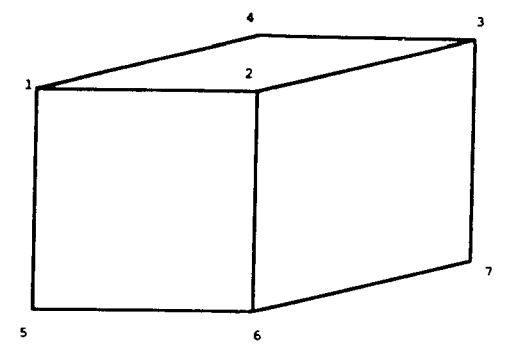


Figura 4. Elementos Triangulares.



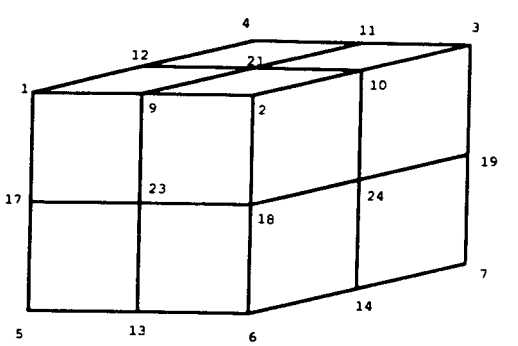


Figura 5. Elementos Sólidos.

c) Pré-Condicionadores Multi-Nível Hierárquicos

Sejam os nós (e equações) correspondentes a uma malha grossa numerados primeiramente e em seguida os nós correspondentes á malha fina. O sistema de equações resultantes para o nível fino pode ser particionado como,

$$\begin{bmatrix} \mathbf{A}_{00} & \mathbf{A}_{01} \\ \mathbf{A}_{10} & \mathbf{A}_{11} \end{bmatrix} \begin{Bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \end{Bmatrix} + \begin{Bmatrix} \mathbf{b}_0 \\ \mathbf{b}_1 \end{Bmatrix} \tag{25}$$

onde os subscritos 0 e 1 correspondem respectivamente ao nível grosseiro (nível 0) e fino (nível 1). Este sistema de equações se relaciona ao sistema de equações hierárquico,

$$\begin{bmatrix} \hat{\mathbf{A}}_{00} & \hat{\mathbf{A}}_{01} \\ \hat{\mathbf{A}}_{10} & \hat{\mathbf{A}}_{11} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{x}}_0 \\ \hat{\mathbf{x}}_1 \end{bmatrix} + \begin{Bmatrix} \hat{\mathbf{b}}_0 \\ \hat{\mathbf{b}}_1 \end{Bmatrix} \tag{26}$$

através de uma matriz não-singular **S**, tal que as seguintes relações são válidas,

$$\hat{\mathbf{A}} = \mathbf{S}^t \, \mathbf{A} \, \mathbf{S} \tag{27}$$

$$\hat{\mathbf{b}} = \mathbf{S}^t \, \mathbf{b} \tag{28}$$

$$\mathbf{x} = \mathbf{S} \, \hat{\mathbf{x}} \tag{29}$$

A matriz **S** transforma a representação de qualquer função de elementos finitos na base hierárquica em sua representação na base nodal (e vice-versa). Ela pode ser entendida também como a representação matricial das relações geométricas apresentadas anteriormente. Desta forma, a matriz **S** pode ser escrita como,

$$\mathbf{S} = \begin{bmatrix} \mathbf{I}_0 & \mathbf{0} \\ \mathbf{W} & \mathbf{I}_1 \end{bmatrix} \tag{30}$$

onde $\mathbf{I}_0$ e $\mathbf{I}_1$ são as matrizes identidade correspondentes de forma exclusiva aos graus de liberdade dos níveis 0 e 1. A submatriz **W** possui termos não nulos somente para as posições definidas na seção anterior. Em outras palavras, a submatriz **W** contém os pesos definidos na relação (24).

Uma vez que na base hierárquica as matrizes correspondentes aos níveis inferiores não são alteradas, ou seja,

$$\hat{\mathbf{A}}_{00} = \mathbf{A}_{00} \tag{31}$$

pode-se definir os seguintes pré-condicionadores bloco-diagonal para MGC,

$$\mathbf{B}_{(\ \mathbf{LDL,BD})} = \begin{bmatrix} (\mathrm{LDL}^t)_{00} & \mathbf{0} \\ \mathbf{0} & (\mathbf{B}_{\mathbf{BD}})_{11} \end{bmatrix} \tag{32}$$

ou

$$\mathbf{B}_{(\mathbf{IC,BD})} = \begin{bmatrix} \mathrm{IC}(\mathbf{A}_{00}) & \mathbf{0} \\ \mathbf{0} & (\mathbf{B}_{\mathbf{BD}})_{11} \end{bmatrix} \tag{33}$$

onde $(\mathbf{LDL}^t)_{00}$ representa a fatoração de Crout da matriz $\mathbf{A}_{00}$, a qual é armazenada em perfil, $\mathrm{IC}(\mathbf{A}_{00})$ é a fatoração incompleta de Crout da matriz $\mathbf{A}_{00}$, também armazenada da mesma forma. O termo $(\mathbf{B}_{\mathbf{BD}})_{11}$ representa um pré-condicionador bloco-diagonal nodal correspondente aos nós do nível 1.

Desta forma, a solução do sistema de equações auxiliar do MGC, $\mathbf{z} = \mathbf{B}^{-1}\mathbf{r}$, é efetuada através dos seguintes passos:

(i) Obter a representação do resíduo na base nodal com respeito à base hierárquica

$$\mathbf{r} = \mathbf{S}^t\,\mathbf{r} = \mathbf{S}^t \begin{Bmatrix} \mathbf{r}_0 \\ \mathbf{r}_1 \end{Bmatrix} \tag{34}$$

(ii) Resolva o sistema correspondente à malha grossa

$$(\mathrm{LDL}^t)_{00}\,\hat{\mathbf{z}}_0 = \hat{\mathbf{r}}_0 \tag{35}$$

ou

$$\mathrm{IC}(\mathbf{A}_{00})\,\hat{\mathbf{z}}_0 = \mathbf{r}_0 \tag{36}$$

(iii) Resolva os bloco-diagonais nodais para os nós da malha fina

$$\hat{\mathbf{z}}_1 = (\mathbf{B}_{\mathbf{BD}})_{11}^{-1}\,\hat{\mathbf{r}}_0 \tag{37}$$

(iv) Obtenha a representação do resíduo pré-condicionado com respeito à base nodal

$$\mathbf{z} = \mathbf{S}\,\hat{\mathbf{z}} = \mathbf{S} \begin{Bmatrix} \hat{\mathbf{z}}_0 \\ \\ \hat{\mathbf{z}}_1 \end{Bmatrix} \tag{38}$$

Com esta estratégia, pode-se verificar facilmente que não há necessidade de se computar as matrizes na base hierárquica, sendo apenas necessária a transformação de bases, via matriz **S**. Além disso, este procedimento pode ser facilmente extendido, sem perda de generalidade, para uma sequência de malhas construída por refinamento uniforme ou adaptativo. Entretanto, cabe observar que no caso de duas malhas, conforme demonstrado por AXELSSON e GUSTAFSSON (1983), o método iterativo resultante irá necessitar de um número de iterações para a convergência independente do número de equações da malha fina.

ESTRATÉGIAS DE IMPLEMENTAÇÃO

Ambiente Computacional

O computador IBM 3090 150E/VF instalado no Centro de Computação do CENPES é uma máquina de uso geral que devido às facilidades de processamento vetorial oferece um alto desempenho em aplicações científicas. Existe atualmente um processador vetorial (VF), integrado à unidade central de processamento (CPU), com um ciclo de máquina de 17.2 nanosegundos. A principal característica da arquitetura do IBM 3090 é a hierárquia de memória, organizada em três níveis, memória cache (64 Kb), mémoria central e memória expandida (opcional), além dos dispositivos de paginação convencionais. A memória principal é de 64 Kbytes e existem também 16 canais de entrada/saída (I/O). Maiores referências sobre a instalação podem ser encontradas no Guia de Uso do CENPES, e sobre a arquitetura da máquina, ver TUCKER (1986). O Sistema Operacional MVS/XA (Extended Architecture) permite a utilização do compilador VS FORTRAN que possui facilidades de geração automática de código vetorizável, através de opção de compilação e um espaço endereçável de até 2 GB.

Aspectos de Vetorização do MGC

O algoritmo do MGC dado na Tabela II possui operações com um alto potencial de vetorização. Estas, de acordo com DONGARRA et al (1991), são as seguintes:

(i) Atualização de Vetores: $\mathbf{x} = \mathbf{x} + \alpha\mathbf{p}$

Esta operação, conhecida como SAXPY, na terminologia empregada no BLAS (Basic Linear Algebra Subroutines), DONGARRA et al (1979), corresponde a uma única instrução vetorial. De forma geral, operações desta natureza são executadas de maneira bastante eficiente na maioria dos computadores vetoriais.

(ii) Produtos Internos: $\alpha = \mathbf{r}^t \cdot \mathbf{r}$

Esta operação corresponde a uma única instrução vetorial e os comentários do item anterior são também válidos aqui.

(iii) Produto Matriz-Vetor: $\mathbf{v} = \mathbf{A}\,\mathbf{p}$

De acordo com DONGARRA et al (1991), produtos matriz-vetor podem, em muitas situações, serem computados com uma eficiência comparável às operações anteriores. Isto é tão verdadeiro quanto mais regular for a estrutura da matriz **A**. Entretanto, para matrizes esparsas, como as encontradas no Método dos Elementos Finitos, o desempenho pode ser seriamente comprometido. Considere por exemplo, o seguinte trecho de código, utilizado no programa ITPACKV (KINCAID, 1989), para uma multiplicação matriz-vetor:

```
DO J = 1, MAXNZ
DO I = 1, N
   Y(I) = Y(I) + COEF(I,J) * X (JCOEF(I,J))
ENDDO
ENDDO
```

onde `COEF(I,J)` armazena de forma compacta os coeficientes da matriz esparsa **A** e `JCOEF (I,J)` os ponteiros para as posições corretas para se efetuar o produto. O loop interno do trecho de código acima é vetorizado, sendo formalmente um SAXPY esparso, devido ao endereçamento indireto de X. Esta característica (endereçamento indireto) é reponsável por uma degradação de desempenho significativa, podendo, em máquinas com hierarquia de memória, reduzir em até mesmo 8 vezes a velocidade de execução do loop.

Devido a estas considerações, e ao fato de que no Método dos Elementos Finitos a matriz global **A** é construída através de contribuções individuais dos elementos, torna-se mais eficiente, segundo HAYES e DEVLOO (1986), se avaliar o produto matriz-vetor a nível de elemento. Esta estratégia, apesar de requerer um número maior de operações aritméticas tem sua eficiência relacionada ao fato que efetuando-se as operações a nível de elemento não há a necessidade de introduzir nenhuma estrutura de dados adicional àquelas já presentes no Método dos Elementos Finitos. Além disso, elimina-se a necessidade de armazenamento e tratamento da matriz esparsa **A**. Existem algumas alternativas para a implementação do produto matriz-vetor à nível de elemento, estudiadas por COUTINHO et al (1991). A estratégia adotada no presente trabalho será apresentada em detalhe adiante.

(iv) Determinação da Solução do Sistema $\mathbf{B}\,\mathbf{z} = \mathbf{r}$

Esta operação diz respeito fundamentalmente a forma adotada para o precondicionador **B**. Caso a solução do sistema de equações não seja suficientemente vetorizada, esta pode comprometer seriamente a eficiência global do MGC. Em consequência, é possível que pré-condicionadores mais simples, porém com maior conteúdo de vetorização, possam conduzir a uma maior velocidade de execução, apesar de requererem um número maior de iterações para convergência do que pré-condicionadores mais complexos. Conforme será exposto adiante, os precondicionadores adotados no presente trabalho possuem um grau de vetorização muito alto, à exceção dos pré-condicionadores multi-nível, que no caso dependem da eficiência de um método direto de solução (ver Equações (34)-(38)).

Multiplicação Matriz-Vetor Elemento-por Elemento

No esquema de multiplicação matriz-vetor elemento-por-elemento (EPE) a matriz **A** nunca é explicitamente formada, sendo o produto na verdade calculado como,

$$\mathbf{A}\,\mathbf{p} = \sum_{e=1}^{Nel} \mathbf{A}_e\,\mathbf{p}_e \tag{39}$$

onde Nel é o número de elementos na malha, $\mathbf{A}_e$ é a matriz de elemento e $\mathbf{p}_e$ são as componentes de **p** restritas aos graus de liberdade do elemento. As matrizes de elemento são armazenadas levando em conta sua simetria, em um arranjo `A(NEL,KND)`, onde `KND=0.5ND* (ND+1)` é o número de termos no triângulo superior de $\mathbf{A}_e$ é `ND` é o número de graus de liberdade do elemento.

Desta forma, a multiplicação matriz-vetor é efetuada nos seguintes passos, para cada elemento:

(i) Localize as componentes do arranjo global **p** para os graus de liberdade do elemento, ou seja, restrinja `P(N)` para `VE1(ND)`.

(ii) Compute o produto matriz-vetor e armazene os resultados em um arranjo local, `VE2(ND)`.

(iii) Espalhe o arranjo local `VE2(ND)` para os graus de liberdade globais do problema, ou seja, espalhe `VE2(ND)` para `AP(N)`, acumulando os resultados.

As operações de localização item(i), e espalhamento item(ii) ou globalização, são efetuadas através de um arranjo `LM(NEL,ND)`. Este arranjo, que define o mapeamento entre os graus de liberdade locais do elemento e os graus de liberdade globais do problema é uma estrutura de dados presente na maioria das implementações do método dos Elementos Finitos (HUGHES, 1987a). Se `LM (I,J)` for igual à zero, então a equação global correspondente foi eliminada da análise.

É fácil de se verificar que o item(ii) possui um conteúdo intrínseco de vetorização. Entretanto, para elementos finitos usuais, as dimensções dos arranjos envolvidos são relativamente pequenas quando comparadas ao tamanho ótimo dos vetores para se obter ganhos significativos com a vetorização do código. Este fato pode comprometer o desempenho, devido ao aumento do tempo necessário para o tráfego de informações entre memória e o processador vetorial. Segundo ORTEGA (1988), o tempo por resultado (t_k) de uma operação vetorial pode ser modelado por:

$$t_k = K + s_t/N \tag{40}$$

onde K é a taxa de resultados, ou seja, o intervalo de tempo necessário para que os resultados comecem a deixar o processador vetorial, s_t é o tempo necessário para o tráfego de informações e N o comprimento do vetor. Da relação (40) pode-se observar que é necessário utilizar vetores suficientemente longos para amortizar o efeito de s_t.

Sendo assim, a taxa de vetorização do produto matriz-vetor EPE pode ser significativamente aumentada processando-se os elementos em blocos e indexando os cálculos pelo número de elementos do bloco. Logo, os arranjos locais serão bi-dimensionais, ou seja, `VE1 (NSIZE,ND)`, `VE2 (NSIZE,ND)`, onde `NSIZE` é o número máximo de elementos por bloco. Entretanto, a operação de globalização envolve endereçamento indireto em ambos os lados da declaração de atribuição, isto é

```
DO IND = 1, ND
DO IEL = LFT, LLT
AP (LM(IEL, IND)) = AP (LM(IEL, IND)) + VE2 (IEL, IND)
ENDDO
ENDDO
```

onde `LFT` e `LLT` são respectivamente o número do primeiro e do último elemento de um dado bloco. A construção anterior não pode ser vetorizada pelo compilador IBM VS FORTRAN, uma vez que ela envolve uma dependência de dados, devido ao endereçamento indireto. Caso a vetorização seja forçada por meio de uma diretiva de compilação, serão produzidos resultados errados, uma vez que geralmente os elementos

de um bloco compartilham um ou mais nós (e graus de liberdade).

A solução para esta recursão global, sugerida por HUGHES et al (1987b), é de se agrupar os elementos em blocos internamente disjuntos, onde nenhum elemento possua um nó (grau de liberdade) comum. Com isso, os elementos serão agrupados em NBLOCK blocos de tamanho máximo NSIZE, correspondendo a um padrão ilustrado na Figura 6, para uma malha de elementos quadriláteros em duas dimensões.

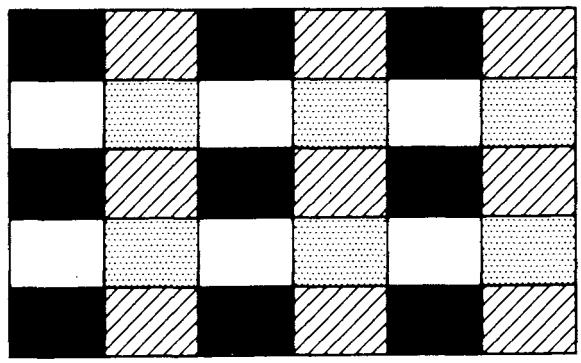


Figura 6. Reordenação dos Elementos em Blocos.

O tamanho ótimo do bloco (NSIZE) para uma máquina IBM 3090/VF modelo E é 128, em função das características do processador vetorial ("Vector Section Size"), como comprovado em um estudo paramétrico anterior (COUTINHO et al, 1988). Caso o número de elementos em um determinado bloco seja inferior à 12, os cálculos são efetuados em modo escalar.

Vetorização da Solução do Sistema de Equações Auxiliar do MGC

a) Pré-Condicionamento Diagonal

Neste caso, a solução do sistema de equações é extremadamente simples, correspondendo a uma única instrução vetorial, conforme o trecho de código,

```
DO I = 1, N
Z(I) = R(I) * DIAG(I)
ENDDO
```

onde o arranjo DIAG(1:N) contém os inversos dos termos da diagonal de A. Deve-se notar que as entradas DIAG(I) são resultado de contribuições individuais dos elementos. Portanto, este arranjo é calculado utilizando a mesma estrutura de blocos de elementos usada anteriormente.

b) Pré-Condicionamento Bloco-Diagonal Nodal

Os blocos-diagonais nodais são computados a partir da contribuição de cada elemento e armazenados em um arranjo `BD(NNOS, KND)` sendo `KND = 0.5*NDOF * (NDOF + 1)` e `NNOS` o número de nós da malha. Uma vez que para elasticidade plana e elasticidade tri-dimensional os blocos-diagonais nodais são respectivamente matrizes 2×2 e 3×3, a inversão dos blocos-diagonais pode ser efetuada facilmente por meio de fórmulas fechadas. Desta forma, a solução do sistema de equações é computada para cada bloco-diagonal em um único *Loop*, totalmente vetorizado, sobre todos os nós da malha.

c) Pré-Condicionamento Elemento-por Elemento

A utilização do MGC com pré-condicionamento EPE envolve a transformação do sistema original de equações, conforme expresso nas relações (8)-(11). Do ponto de vista da implementação, estas operações são basicamente as seguintes:

(i) Cálculo da matriz diagonal

Esta operação é efetuada de forma análoga à descrita no item 4.4.a, com a única diferença que são armazenados os inversos das raízes quadradas dos termos da diagonal de **A**. Portanto, esta operação também é completamente vetorizada.

(ii) Regularização das matrizes de elementos: $\overline{\mathbf{A}}_e = (\mathbf{D}_e^{-1})^T \mathbf{A}_e$ e $\mathbf{D}_e^{-1}$

Esta operação é efetuada com a mesma estrutura do produto matriz vetor, observando-se que os termos de $\mathbf{D}^{-1}$ correspondentes aos graus de liberdade do elemento são replicados no vetor local `VE1`. A operação de regularização é então efetuada para cada elemento no bloco. Evidentemente, não é necessário o passo de espalhamento dos resultados.

(iii) Solução do Sistema Auxiliar

A solução do sistema auxiliar $\mathbf{B}_{\text{EPE}}\mathbf{z} = \mathbf{r}$, com $\mathbf{B}_{\text{EPE}}$ dado pela expressão (12), compreende os seguintes passos:

- Solução dos sistemas triangulares inferiores a nível de elementos, ou seja,

$$\prod_{e=1}^{Nel} (\mathbf{L}_e)\,\mathbf{f} = \mathbf{r}, \text{ onde } \mathbf{f} = \prod_{e=Nel}^{1} (\mathbf{u}^e)\,\mathbf{z}$$

Esta operação é efetuada com a mesma estrutura do produto matriz-vetor, sendo portanto vetorizada.

- Solução dos sistemas triangulares superiores a nível de elemento,

$$\prod_{e=Nel}^{1} (\mathbf{u}^e)\,\mathbf{z} = \mathbf{f}$$

Esta operação é formalmente idêntica à anterior, observando-se apenas a ordem inversa das operações, tanto a nível local, durante a solução de $\mathbf{u}^e\ \mathbf{z}_e = \mathbf{f}_e$, quanto a nível global.

(iv) Recuperação da solução: $\mathbf{X} = \mathbf{D}^{-1}\overline{\mathbf{X}}$

Esta operação é equivalente à descrita no item que trata sobre pré-condicionamiento diagonal e portanto é totalmente vetorizada. Desta forma, todo o processo de solução do sistema auxiliar de equações é vetorizável.

Cabe ressaltar que, devido à estrutura de blocos disjuntos, não há recursão alguma no acúmulo das soluções dos sistemas triangulares a nível de elemento, uma vez que os elementos não possuem grau de liberdade em comum. Além disso, não há necessidade do arranjo intermediário $\mathbf{f}$, podendo os resultados das operações serem acumulados diretamente em $\mathbf{z}$, mais uma vez, devido ao fato dos elementos em cada bloco não possuirem graus de liberdade comuns.

d) Pré-Condicionamento Multi-Nível

Para os pré-condicionadores definidos através das expressões (32) e (33), a solução do sistema auxiliar de equações do MGC é efetuada através das relações (34) a (38). Estas são fundamentalmente a solução direta do sistema de equações correspondente à malha grossa, a representação de um vetor na base hierárquica e a solução para os bloco-diagonais nodais. Esta última é formalmente idêntica à descrita no ítem que trata sobre pré-condicionamiento bloco-diagonal nodal. As duas primeiras, podem causar algumas dificultades à vetorização e são examinadas à seguir.

(i) Solução direta do sistema de equações

Neste trabalho é utilizado um solucionador direto para matrizes armazenadas em perfil. A vetorização da fatoração da matriz na forma de Crout (LDL^t), segue as técnicas descritas por LOZUPONE (1989). Para a implementação da fatoração incompleta de Crout, deve-se notar que um ponteiro adicional é necessário para definição dos termos não-nulos em cada coluna ativa. Entretanto, tendo em vista que será efetuada uma retrosubstituição a cada iteração do MGC, em ambos os casos a etapa de obtenção da solução por avante e retrosubstituição foi vetorizada. Na maioria das implementações vetoriais de solucionadores diretos, esta etapa é desprezada, ou seja, não vetorizada, devido ao fato de que a fatoração da matriz é muito mais custosa que a solução.

(ii) Mudança da base nodal para base hierárquica (e vice-versa)

As mudanças de base levam em consideração a estrutura altamente esparsa da matriz $\mathbf{S}$. Os algoritmos são fundamentalmente os mesmos apresentados por YSERENTANT (1986), e envolvem uma recursão, já que os valores das incógnitas correspondentes ao nível K são dependentes do nível $K-1$. Consequentemente, não é possível, com a estrutura de dados utilzada, vetorizar as operações de mudança de base.

APLICAÇÕES NUMÉRICAS

Preliminares

Serão apresentadas a seguir algumas aplicações que visam exemplificar o desempenho das estratégias computacionais descritas anteriormente. A seguinte nomenclatura será utilizada para identificar os diferentes métodos iterativos empregados:

(D-MGC) – Método de Gradientes Conjugados Pré-Condicionado com Pré-Condicionador Diagonal.

(BD-MGC) – Método de Gradientes Conjugados Pré-Condicionado com Pré-Condicionador Bloco-Diagonal Nodal.

(EPE-MGC) – Método de Gradientes Conjugados Pré-Condicionado com Pré-Condicionador Elemento-por Elemento Gauss-Seidel.

(LDL-MGC) – Método de Gradientes Conjugados Pré-Condicionado com Pré-Condicionador Multi-Nível com Fatoração de Crout para o Nível Grosso e Bloco Diagonal Nodal para os demais.

(IC-MGC) – Método de Gradientes Conjugados Pré-Condicionado com Pré-Condicionador Multi-Nível com Fatoração Incompleta de Crout para o Nível Grosso e Bloco-Diagonal Nodal para os demais.

Todas as medições de desempenho foram efetuadas durante períodos de carga computacional normal do sistema em produção, sem recurso a um ambiente dedicado. Dessa forma, o resultado das medições pode ser considerado, bastante representativo para o sistema em questão.

Problema de Boussinesq Tridimensional

Neste problema clássico de elasticidade, uma carga concentrada é aplicada na direção normal à superfície de um semi-espaço homogêneo e isotrópico. O problema foi modelado por meio de malhas uniformes de elementos sólidos isoparamétricos com 8 nós, adotando-se condições de simetria nas três faces ortogonais. As propriedades elásticas do material são: Módulo de Elasticidade, 2.08×10^6 e coeficiente de Poisson, 0.3.

Este problema foi selecionado para investigar o desempenho das estratégias de vetorização apresentadas anteriormente. Sendo assim, utilizou-se o método (D-MGC), considerando uma série de malhas uniformes, com um número crescente de elementos ao longo de cada eixo ortogonal. Adotou-se como tolerância para o critério de parada dado na Tabela II, $\delta = 10^{-3}$. Os dados topológicos para cada malha encontram-se na Tabela IV, onde N é o número de elementos ao longo de cada eixo ortogonal.

A Figura 7 apresenta a malha resultante para N = 24, mostrando também as isocurvas logaritmicas da distribuição de pressões obtida.

Na Tabela V estão sumarizados número de iterações para cada análise, tempo de

N	NÚMERO DE ELEMENTOS	NÚMERO DE EQUAÇÕES	SEMI-LARGURA DE BANDA
15	3375	11520	789
16	4096	13872	889
20	8000	26460	1349
24	13824	45000	1905

Tabela IV. Dados Topológicos para o Problema de Boussinesq Tridimensional.

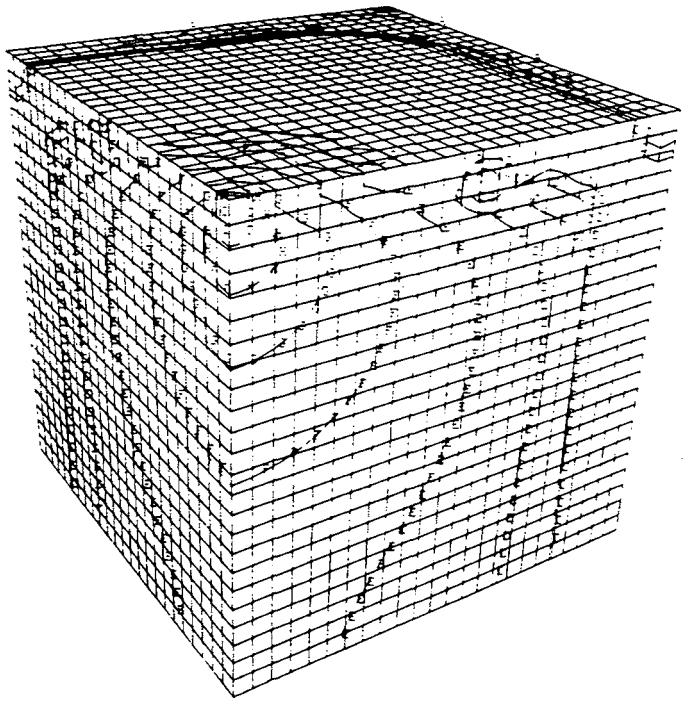


Figura 7. Malha $24 \times 24 \times 24$ para o Problema de Boussinesq Tridimensional.

N	ITERAÇÕES	CPU (s)	CPU VET. (%)	MFLOPS
15	64	12	92	19
16	68	20	95	19
20	83	39	95	19
24	96	81	94	19

Tabela V. Desempenho do (D-MGC) para o Problema de Boussinesq Tridimensional.

processamento em segundos para a solução do sistema de equações, percentagem desse tempo dispendido no processador vetorial e o número de MFLOPS.

O número de MFLOPS para cada análise foi avaliado segundo os critérios sugeridos por DONGARRA et al (1991). Na mesma refêrencia encontra-se um estudo de desempenho semelhante, porém para um problema em que o sistema de equações é penta-diagonal. Os resultados obtidos encontram-se reproduzidos na Tabela VI.

MÁQUINA	DESEMPENHO EM MFLOPS	DESEMPENHO MÁXIMO EM MFLOPS
NEC SX-2	643	1333
FUJITSU VP-200	300	533
HITACHI 5810/20	240	800
CRAY X-MP (1 proc.)	134	235
CYBER 205	106	200
CRAY 2 (1 proc.)	82	480
IBM 3090/VF (1 proc.)	23	108
CONVEX C-210	19.5	50
ALLIANT FX/4 (1 proc.)	1.8	11

Tabela VI. Desempenho do MGC não Pré-Condicionado para Sistema Penta-Diagonal com 10.000 Equações (DONGARRA et al, 1991).

Deve-se notar que para a estrutura regular do problema abordado por DONGARRA a avaliação do produto matriz-vetor é muito simples. Este fato, aliado à utilização de um método não pré-condicionado, conduz a um algoritmo de menor complexidade computacional, favorecendo um alto desempenho em termos de MFLOPS. Sendo assim, as medições apresentadas na Tabela VI podem ser tomadas como um limite superior do desempenho em MFLOPS para o método dos Gradientes Conjugados.

Tendo em vista a complexidade da estratégia elemento-por-elemento aplicada ao (D-MGC) na solução do problema de Boussinesq, pode-se observar que o número de MFLOPS obtido se aproxima bastante do limite apresentado por DONGARRA. Por outro lado, deve-se ter em mente que a utilização de pré-condicionadores mais complexos, examinados à seguir, apesar de resultar em um número de MFLOPS sensivelemente menor implica numa redução considerável do tempo de processamento, que é de fato o objetivo a ser alcançado.

Problema de Boussinesq Axisimétrico

Neste caso, o problema anterior foi modelado por malhas uniformes compostas por $N \times N$ elementos finitos isoparamétricos bilineares como mostrado na Figura 8. A aproximação axisimétrica conduz à matrizes não tão bem condicionadas quanto

à modelagem tridimensional, devido à presença do termo radial nas integrais dos elementos. Sendo assim, o objetivo destas análises é estudar o desempenho do MGC com diferentes pré-condicionadores para dimensões crescentes de discretização.

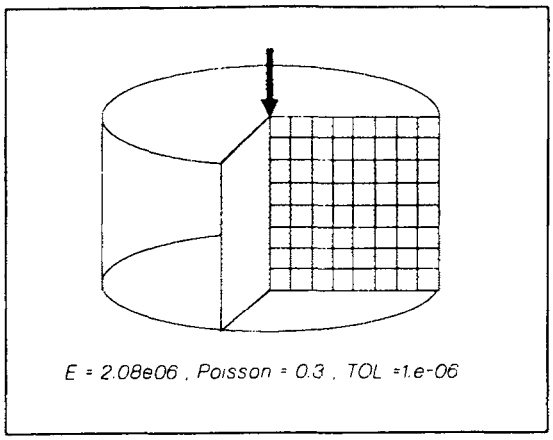


Figura 8. Malha $N \times N$ para o Problema de Boussinesq Axisimétrico.

Os dados relativos às discretizações empregadas e a demanda de memória, em Mbytes, para os diferentes métodos iterativos empregados, assim como para um método direto com armazenamento em perfil, encontram-se sumarizados na Tabela VII.

N	NÚMERO DE EQUAÇÕES	(BD-MGC)	(EPE-MGC)	(LDL-MGC) 2 NÍVEIS	(IC-MGC) 2 NÍVEIS	DIRETO
30	1860	0.4	0.4	0.6	0.5	1.0
60	7320	1.7	1.7	2.7	1.8	7.4
120	29040	6.8	6.5	14.0	8.3	57.6

Tabela VII. Demanda de Mémoria para o Problema de Boussinesq Axisimétrico.

De forma geral, os métodos iterativos requereram significativamente menos mémoria que o método direto. A redução observada varia entre 9 vezes, para o (EPE-MGC), até 4 vezes para o (LDL-MGC). Cabe observar que considerou-se apenas 2 níveis para os pré-condicionadores multinível. Sendo assim, o número de equações para as matrizes correspondentes às malhas grossas para $N = 30, 60$ e 120 é de respectivamente 480, 1860 e 7320. Portanto, nos métodos multinível, a necessidade de se computar a fatoração (completa ou incompleta) da matriz correspondente ao nível grosso aumenta consideravelmente a demanda de memória desses métodos.

N	CUSTO	(BD-MGC)	(EPE-MGC)	(LDL-MGC)	(IC-MGC)	DIRETO
30	CPU	4.8	4.1	3.7	3.7	4.9
	Iters.	213	87	26	34	–
60	CPU	29.5	23.3	16.2	17.3	33.0
	Iters.	422	168	26	51	–
120	CPU	197.8	148.3	84.7	103.3	314.5
	Iters.	838	328	26	99	–

Tabela VIII. Custos Computacionais para o Problema de Boussinesq Axisimétrico.

Os custos computacionais, ou seja, número de iterações para convergência da solução iterativa e tempo de processamento em segundos (CPU), encontram-se dados na Tabela VIII. Como tolerância para verificação de convergência, adotou-se $\delta = 10^{-6}$.

Conforme aumenta a dimensão da malha os métodos iterativos tornam-se cada vez mais eficientes. Além disso, o método (LDL-MGC) requer um número de iterações independente da malha, o que permite fornecer a solução mais rápida em todas as malhas analisadas, chegando a um fator de redução de 3.5, em comparação com a solução direta, para $N = 120$.

Grupo de Estacas

Nesta aplicação são analisados os efeitos de grupo em duas estacas tubulares metálicas idênticas solicitadas lateralmente. A geometria e carregamento da estrutura de fundação é similar àquela encontrada na Plataforma de Bonito, uma jaqueta metálica projetada para uma lâmina de água de 200 m e instalada na Bacia de Campos, Rio de Janeiro, Brasil. O solo adjacente às estacas é considerado como uma única camada de argila, cujos parâmetros elásticos são: módulo de elasticidade, 0.5 MPa e coeficiente de Poisson, 0.4. Os dados geométricos e propriedades elásticas do material das estacas são: comprimento, 45 m; diâmetro externo 2.13 m; espessura, 0.0508 m; módulo de elasticidade 210.88 GPa; coeficiente de Poisson 0.3. Devido à simetria, apenas metade do conjunto solo-estaca foi discretizado. Além disso, uma vez que a interação solo-estaca é um fenômeno localizado em torno das estacas e na região entre elas, a massa de solo foi discretizada até 9 m de cada eixo das estacas, onde prescreveu-se condições de contorno de deslocamentos nulos. Na parte inferior foram restringidos os deslocamentos nodais na direção vertical.

Inicialmente, foi efetuada uma análise com a malha da Figura 8, que compreende 1376 elementos sólidos isoparaméticos trilineares a 1938 pontos nodais, totalizando 4232 graus de liberdade. Após a reordenação da malha pelo algoritmo de Cuthill-McKee reverso (GEORGE e LIU, 1981) a máxima semi-largura de banda obtida foi 2354 e a semi-largura de banda média, 303.

MÉTODO	NÚMERO DE ITERAÇÕES	CPU (s)	MEMÓRIA (Mbytes)
(BD-MCG)	5119	756.8	3.85
(EPE-MCG)	1793	528.3	3.78
Direto	- - - -	58.8	10.45

Tabela IX. Custos Computacionais para Análise Tridimensional do Efeito de Grupo (4232 equações).

Os custos computacionais para os métodos iterativos (BD-MGC), (EPE-MGC) e para o método direto encontram-se na Tabela IX. Neste problema, em todas as análises iterativas foi adotado como critério de parada $\delta = 10^{-3}$.

Devido à presença de materiais com propriedades físicas bastante diferentes e elementos com razão de aspecto elevada (relação entre maior e menor dimensão do elemento), o condicionamento da matriz resultante não é favorável aos métodos iterativos. Portanto, conforme mostrado na Tabela IX, os métodos iterativos foram mais lentos que o método direto. O mau condicionamento do sistema de equações pode ser medido através da relação entre o maior ($D\max$) e o menor ($D\min$) valor dos termos da diagonal da matriz de rigidez, $D\max/D\min = 1.113 \times 10^9$. Entretanto, a demanda de memória favorece as soluções iterativas.

De forma a proporcionar uma melhor modelagem da interação solo-estaca, a malha da Figura 9 foi refinada uniformemente, originando o modelo da Figura 10. Este modelo possui 11008 elementos sólidos, 13204 pontos nodais e 33610 graus de liberdade.

Os gustos computacionais obtidos nesta análise encontram-se na Tabela X, incluindo agora resultados para os pré-condiconadores multinível, que no caso, utilizam 2 níveis, a malha original e a malha refinada. O mesmo critério da parada da análise anterior foi utilizado em todos os métodos iterativos.

Este problema foi solucionado apenas pelo (LDL-MGC). Isto pode ser compreendido notando-se que na análise do problema com a malha grossa, a solução direta foi a mais rápida, e requeria uma quantidade aceitável de área de armazanamento. Portanto, a utilização desta matriz como pré-condicionador para a malha fina tornou a solução com o método pré-condicionado em dois níveis muito eficiente. Em contra partida, não foi possível de se obter a fatoração incompleta da matriz da malha grosseira, o que inviabilizou a solução pelo método (IC-MGC). Os outros métodos iterativos não obtiveram a convergência dentro de um tempo de processamento aceitável e conforme pode ser verificado na Tabela X, ainda se encontravam longe disso. A solução direta deste problema não foi obtida devido à enorme demanda de mémoria, mesmo após a reordenação da malha, que conduziu a um sistema com semi-largura de banda 2101 e semi-largura de banda média 1305, correspondendo ao valor de área de armazenamento indicado na Tabela X.

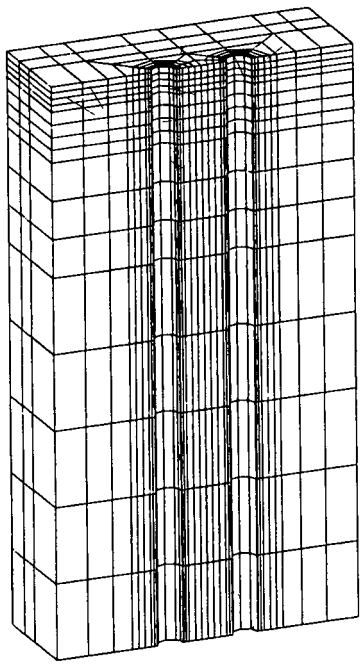


Figura 9. Malha para Análise Tridimensional do Efeito do Grupo (4232 equações).

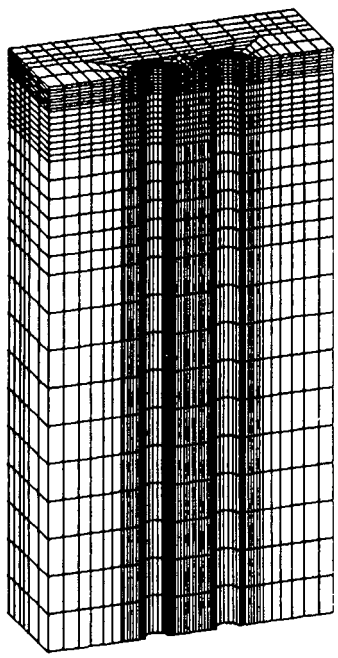


Figura 10. Malha para Análise Tridimensional do Efeito do Grupo (33610 equações).

MÉTODO	NÚMERO DE ITERAÇÕES	CPU (min)	MEMÓRIA (Mbytes)	NORMAL RELATIVA DO RESÍDUO NA ÚLTIMA ITERAÇÃO
(LDL-MGC)	1124	99	40.7	9.970 E-4
(IC-MGC)	–	–	–	–
(EPE-MCG)	3603	120	29.8	2.181 E-1
(BD-MGC)	7098	120	30.5	5.711 E-1
Direto	–	–	351.5	–

Tabela X. Custos Computacionais para Análise Tridimensional do Efeito de Grupo (33610 equações).

CONCLUSÕES

Procurou-se neste trabalho avaliar diversas alternativas de precondicionamento para o método iterativo dos gradientes conjugados, com o objetivo de selecionar aqueles que melhor desempenho deverão ter na solução numérica de grandes problemas

tridimensionais, em ambientes computacionais dotados de processamento vetorial. Verificou-se que, entre as várias alternativas propostas, aquelas que utilizam os pré-condicionadores multi-nível, em especial a que emprega a fatorização de Crout para o nível grosso, são mais eficientes que as que utilizam pré-condicionadores padrão.

Além disso, comparações realizadas entre estes métodos com um solucionador direto vetorizado, mostraram que estes algoritmos são competitivos, justificando a tendência de se utilizar métodos iterativos na análise de problemas que envolvem um grande número de equações.

AGRADECIMENTOS

Este trabalho foi realizado como parte das atividades do Convênio de Cooperação Científica em Engenharia Offshore celebrado entre a Companhia Brasileira de Petróleo, Petrobrás S.A. e a Coordenação dos Programas de Pós-Graduaçõ em Engenharia da Universidade Federal do Rio de Janeiro (COPPE/UFRJ). Os autores gostariam de agradecer ao Dr. H.K. Chang, do Centro de Pesquisa da Petrobrás (CENPES) pela sua cooperação no desenvolvimento desta pesquisa.

REFERÊNCIAS

1. O. Axelsson e V.A. Barker, *"Finite Element Solution of Boundary Value Problems"*, Academic Press, USA, (1984).
2. O. Axelsson e I. Gustafsson, "Preconditioning and Two-Level Multigrid Methods of Arbitrary Degree of Aproximation", *Mathematics of Computation*, **40**, pp. 219–242, (1983).
3. R.H. Bank, T. Dupont e H. Yserentant, "The Hierarchical Basis Multigrid Method", *Numerische Mathematik*, **52**, pp. 427–458, (1988).
4. G.F. Carey e J.T. Oden, *"Finite Elements Computational Aspects"*, Prentice-Hall, Englewood Cliffs, NJ, USA, (1984).
5. A.L.G.A. Coutinho, J.L.D. Alves, L. Landau e N.F.F. Ebecken, "Solução Iterativa de Sistemas de Equações do Método dos Elementos Finitos no Computador IBM 3090", *2o Simpósio Brasileiro de Arquitetura de Computadores e Processamento Paralelo*, SBAC-PP, Águas de Lindóia-SP, Ed. E.W. Bergamini, Soc. Bras. Comp., Vol. **1**, pp. 11.a.6.1.–11.a.6.7., (1988).
6. A.L.G.A. Coutinho, J.L.D. Alves, L. Landau e N.F.F. Ebecken, "A Study of Implementation Schemes for Vectorized Sparse EBE Matrix-Vector Multiplication", *Advances in Engineering Software and Workstations*, Vol. **13**, no. 3, pp. 130–134, (1991).
7. L. Demkowicz, P. Devloo e J.T. Oden, "On an h-Type Mesh Refinement Strategy Based on Minimization of Interpolation Errors", *Computer Methods in Applied Mechanics and Engineering*, Vol. **53**, pp. 67–90, (1985).
8. P.R. Devloo, "A Three-Dimensional Adaptative Finite Element Strategy", *Computer and Structures*, Vol. **38**, pp. 121–130, (1991).
9. J.J. Dongarra, C.J. Moller, J.R. Bunch e G.W. Stewart, "Linpack User's Guide", *SIAM*, Philadelphia, USA, (1978).

10. J.J. Dongarra, I.S. Duff, D.C. Sorensen e H. Van Der Vorst, "Solving Linear Systems on Vector and Shared Memory Computers", *SIAM*, Philadelphia, USA, (1991).
11. A. George e J.W.H. Liu, *"Computer Solution of Large Sparse Positive Definite Systems"*, Prentice-Hall, Englewood, Cliffs, USA, (1981).
12. G.H. Golub e C. Van Loan, *"Matrix Computations"*, 2nd Edition, The Johns Hopkins University Press, Baltimore, USA, (1989).
13. Guia de Uso da Instalação , CENPES/PETROBRÁS S.A., (1991).
14. W. Hackbush, *"Multigrid Methods and Aplications"*, Springer-Verlag, Berlin, (1985).
15. L.J. Hayes e P. Devloo, "A Vectorized Version of a Sparse Matrix-Vector Multiply", *International Journal for Numerical Methods in Engineering*, Vol. **23**, pp. 1043–1056, (1986).
16. M.R. Hestenes e E. Stiefel, "Methods of Conjugate Gradients for Solving Linear Systems", *J. Res. Nat. Bur. Standards Sect. B*, Vol. **49**, pp. 409–436, (1952).
17. T.J.R. Hughes, I. Levit e J. Winget, "An EBE Algorithm for Problems of Structural and Solid Mechanisms", *Computer Methods for Applied Mechanics and Engineering*, Vol. **36**, pp. 215–248, (1983).
18. T.J.R. Hughes, *"The Finite Element Method: Linear Static and Dynamic Finite Element Analysis"*, Prentice-Hall, Englewood Cliffs, USA, (1987 a).
19. T.J.R. Hughes, R.M. Ferencz e J.O. Hallquist, "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, (1987 b).
20. D.R. Kincaid, T.C. Oppe e D.M. Young, "ITPACKV User's Guide", *Center for Numerical Analysis*, University of Texas al Austin, Report CNA-232, (1989).
21. D.F. Lozupone, "A Skyline Solver for Symetric Matrices on the IBM 3090 Vector Multiprocessor", *Application of Supercomputers in Engineering: Algorithms, Computer Systems and User Experience*, Eds. C.A. Brebbia e A. Peters, Elsevier e Computational Mechanics Publications, pp. 53–65, (1989).
22. S.F. McCormick, *"Multigrid Methods"*, SIAM Philadelphia, USA, (1987).
23. I.D. Parssons e J.F. Hall, "The Multigrid Method in Solid Mechanics: Part I – Algorithm Description and Behaviour", *International Journal for Numerical Methods in Engineering*, Vol. **29**, pp. 719–737, (1990 a).
24. I.D. Parsons e J.F. Hall, "The Multigrid Method in Solid Mechanics: Part II – Practical Applications", *International Journal for Numerical Methods in Engineering*, Vol. **29**, pp. 739–753, (1990 b).
25. F.L.B. Ribeiro, "Uma Estratégia h-p de Refinamento para o Método dos Elementos Finitos", *Tese de D.Sc.*, Programa de Engenharia Civil, COPPE/UFRJ, Rio de Janeiro, (1991).
26. S.G. Tucker, "The IBM 3090 System: An Overview", *IBM System Journal*, Vol. **25**, pp. 4–19, (1986).
27. J. Winget e T.J.R. Hughes, "Solution Algorithms for Non Linear Transient Heat Conduction Analysis Employing Element-by-Element Iterative Strategies", *Computer Methods in Applied Mechanics and Engineering*, Vol. **52**, pp. 711–815, (1985).
28. H. Yeserentant, "On the Multi-Level Splitting of Finite Element Spaces for Indefinite Elliptic Boundary Value Problems", *SIAM Journal for Numerical Analysis*, Vol. **23**, pp. 581–595, (1986).