Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería. Vol. 13,2, 123–148(1997)

ANÁLISE COMPARATIVA DE MÉTODOS DIRETOS E ITERATIVOS PARA A SOLUÇÃO DE SISTEMA DE EQUAÇÕES

MARCO LÚCIO BITTENCOURT*
y
RAÚL ANTONINO FEIJÓO**

* *Departamento de Projeto Mecânico (DPM)*
Faculdade de Engenharia Mecânica (FEM)
Universidade Estadual de Campinas (UNICAMP)
Caixa Postal 6051, CEP 13083-971, Campinas/SP - Brasil
Tel: + 55-19-788 8384 Fax: + 55-19-239 3722
E-mail: mlb@dpm.fem.unicamp.br

** *Laboratório Nacional de Computação Científica (LNCC/CNPq)*
Rua Lauro Muller 455, Caixa Postal 56018, CEP 22290-000, Rio de Janeiro/RJ - Brasil
Tel: + 55-21-541 2132 Fax: + 55-21-295 7944
E-mail: feij@server01.lncc.br

SUMÁRIO

Neste trabalho, procura-se analisar a performance de alguns métodos diretos e iterativos, em termos do número de operações e espaço de memória, empregando-se estruturas de dados em colunas ascendentes e esparsa para a matriz do sistema de equações. Apresenta-se uma revisão destes métodos e das classes implementadas em C++. Finalmente, problemas elásticos definidos em domínios bi e tridimensionais são empregados, visando efetuar uma comparação entre os métodos abordados.

COMPARATIVE ANALYSIS OF DIRECT AND ITERATIVE METHODS FOR SOLVING SYSTEMS OF EQUATIONS

SUMMARY

In the present work, the performance of iterative and direct methods is investigated, both in terms of allocated memory and number of operations, using skyline and sparse data structures for the system matrix. A review of these methods along with the classes implemented in C++ is also presented. Finally, two and three-dimensional elastic problems are considered for comparison among the methods.

Recibido: Febrero 1996

 ISSN 0213–1315

INTRODUÇÃO

Em geral, na aplicação do Método de Elementos Finitos (MEF) para a solução de problemas de engenharia, é necessário resolver um sistema de equações da seguinte forma

$$\mathbf{Au} = \mathbf{b} \tag{1}$$

onde a matriz $\mathbf{A}_{N\times N}$ é em geral simétrica e positiva-definida (SPD); $\mathbf{u}_{N\times 1}$ e $\mathbf{b}_{N\times 1}$ são os vetores das incógnitas e termos independentes, respectivamente.

Para resolver o sistema (1), pode-se empregar métodos diretos e iterativos. No primeiro caso, a solução é obtida através de um número determinado de operações, modificando-se os coeficientes da matriz **A**. Já os métodos iterativos, a partir de uma solução inicial $\mathbf{u}^{(0)}$, realizam iterações sucessivas até que se atinja a solução dentro de uma precisão especificada. Ambas as técnicas possuem vantagens e desvantagens[1].

Tradicionalmente, os métodos diretos têm sido empregados para sistemas de ordem moderada devido a maior eficiência computacional dos mesmos. No entanto, a medida que se aumenta o número de equações, os métodos iterativos tornam-se competitivos tanto em relação ao número de operações para a solução de (1) quanto ao espaço de armazenamento para a matriz **A**.

Uma outra técnica empregada na resolução de sistemas de equações são os métodos Multgrid[1,2,3], os quais proporcionam uma maior taxa de convergência dos métodos iterativos. A motivação deste trabalho foi desenvolver um estudo dos métodos diretos e iterativos mais empregados, visando sua comparação com algoritmos Multgrid[2]. Observa-se, por exemplo, que a aplicação de estimadores de erro e técnicas adaptáveis, eleva sensivelmente a ordem do sistema de equações, fazendo com que a eficiência na solução de (1) seja de fundamental importância.

O objetivo deste trabalho é apresentar resultados da aplicação de métodos diretos e iterativos para a solução de sistemas de equações obtidos da aplicação do MEF a problemas elásticos lineares. Estes resultados estão na forma do custo computacional e espaço de memória.

Inicialmente, discutem-se estruturas de dados para o armazenamento da matriz $\mathbf{A}$[1,4,5], implementadas através de classes da linguagem C++[2,6]. Posteriormente, apresenta-se um resumo baseado em[1,2,4,5,7,8] sobre os métodos direto de Gauss, estacionários (Jacobi, Gauss-Seidel, SOR e SSOR) e de aceleração polinomial (Chebyshev e Gradiente Conjugado). Finalmente, aplicam-se estes métodos para problemas elásticos bi e tridimensionais, discutindo os resultados obtidos.

ESTRUTURAS DE DADOS PARA A MATRIZ DO SISTEMA DE EQUAÇÕES

Visando alcançar eficiência na solução do sistema de equações (1), torna-se essencial empregar estruturas de dados adequadas para a matriz **A**. Deve-se procurar armazenar o número mínimo de elementos necessários para a resolução de (1) através de um método direto ou iterativo, reduzindo assim não apenas o espaço de memória, mas também o

número de operações. Neste trabalho, foram adotados os armazenamentos em colunas ascendentes e esparso, estando as estruturas de dados ilustradas na Figura 1 para um exemplo simples.

$$\mathbf{A} = \begin{bmatrix} 10 & 30 & 50 & 0 & 90 \\ & 20 & 0 & 70 & 0 \\ & & 40 & 0 & 0 \\ \text{sim.} & & & 60 & 0 \\ & & & & 80 \end{bmatrix}$$

Colunas ascendentes:

maxa = [1 2 4 7 10 15]

a = [10 20 30 40 0 50 60 0 70 80 0 0 0 90]

Esparsa:

diag = [1 5 7 8 9 10]

a = [10 30 50 90 20 70 40 60 80]

col = [1 2 3 5 2 4 3 4 5]

Figura 1. Estruturas de dados para matrizes em colunas ascendentes e esparsa

No caso de colunas ascendentes, tem-se os vetores **a** e **maxa**, respectivamente, para os elementos da matriz e os indíces de **a** correspondentes aos termos da diagonal principal. Este último vetor contém $N+1$ posições e a diferença entre os elementos $i+1$ e i $(i=1,\ldots,N)$ fornece o número de elementos m_i da coluna i. Por sua vez, a altura da coluna i, denotada por m_i^U, é dada por $m_i^U = m_i - 1$, ou seja, não se considera o elemento da diagonal principal.

Para matriz esparsa, utiliza-se o Armazenamento Comprimido por Linhas[1,4]. Este esquema é bastante simples e armazena estritamente apenas os elementos não-nulos da matriz. São necessários um vetor **a** para os coeficientes, outro denominado **col** com os índices das colunas, além do vetor **diag**, análogo ao **maxa**, com os indíces de **a** dos termos da diagonal principal da matriz. Uma das desvantagens deste formato é o endereçamento indireto para o acesso aos elementos de uma linha.

Observa-se que o uso de métodos iterativos com matrizes esparsas não requer a renumeração ótima das equações. No entanto, para colunas ascendentes de forma geral e métodos diretos em matrizes esparsas é fundamental determinar uma numeração ótima, visando reduzir a demanda de memória e processamento. No caso de colunas ascendentes, empregou-se o algoritmo de Cuthill-Mckee[9,10]. Já para matrizes esparsas, utilizou-se a técnica de grau mínimo[11,12].

Denota-se por m a altura média das colunas para a estrutura em colunas ascendentes ou o número médio de elementos por linha na parte superior da matriz esparsa. Da mesma maneira, m_i^U denota a altura da coluna i ou o número de elementos na matriz esparsa, desconsiderando o termo da diagonal principal. Estas constantes estão relacionadas por $m = \sum_{i=1}^{N} m_i^U/N + 1$.

Verifica-se que para as duas estruturas de dados consideradas, necessitam-se, respectivamente, $N + \sum_{i=1}^{N} m_i^U = Nm$ e $(N+1)/2$ posições de memória com precisão

dupla para os vetores **a** e **maxa/diag**. Para matriz esparsa, tem-se ainda o vetor **col** de números inteiros ocupando a metade do armazenamento de **a**, ou seja, $Nm/2$ posições.

Neste trabalho, uma operação de ponto flutuante (*flop - floating operation*) será entedida como uma multiplicação/divisão seguida geralmente de uma adição/subtração.

MÉTODO DIRETO DE GAUSS

No método de Gauss, as equações do sistema são modificadas em passos sucessivos até que a solução seja encontrada[7]. Para isso, a matriz simétrica **A** é fatorada como

$$\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^T \tag{2}$$

sendo **L** triangular inferior com diagonal unitária e **D** uma matriz diagonal.

Substituindo (2) em (1) e denotando $\mathbf{w} = \mathbf{D}\mathbf{L}^T\mathbf{u}$, chega-se ao sistema triangular inferior $\mathbf{L}\mathbf{w} = \mathbf{b}$ cuja solução é obtida por um processo de substituição a frente. Conhecido o vetor **w**, a solução **u** de (1) é determinada por uma substituição para trás, ou seja, $\mathbf{L}^T\mathbf{u} = \mathbf{D}^{-1}\mathbf{w}$.

A fatoração de **A** é realizada em N passos através das colunas ou linhas, ambas conduzindo ao mesmo resultado. No entanto, a primeira forma é mais conveniente para armazenamento em colunas ascendentes, enquanto a decomposição por linhas é empregada para matrizes esparsas.

Uma forma de estimar o número de operações para a fatoração de **A** e os processos de substituição está dada em[7], considerando o parâmetro m_i^U definido anteriormente. Para a fatoração são necessárias $\sum m_i^U(m_i^U+3)/2$ multiplicações e $\sum m_i^U(m_i^U+1)/2$ adições. Já nas substituições empregam-se $N + 2\sum m_i^U$ multiplicações e $2\sum m_i^U$ adições. Devido a definição anterior de operação de ponto flutuante, tem-se um total de $N + \sum m_i^U(m_i^U+7)/2$ operações para a solução via método de Gauss. No que se refere ao espaço de memória, além da matriz **A**, necessitam-se mais N posições para o vetor **b**.

Na fatoração de **A**, ocorre em geral preenchimento (*fill in*), ou seja, elementos nulos tornam-se diferentes de zero devido à eliminação de termos nas linhas e colunas de **A**. Em colunas ascendentes, o preenchimento está confinado no perfil da matriz, estando todos os elementos alocados no início da fatoração. Já para matrizes esparsas, a fatoração inclui novos termos não previstos inicialmente. Para isso, aplica-se antes da fatoração, o processo denominado Gauss simbólico, o qual determina os termos que se tornarão não-nulos, permitindo, então, alocar espaço para estes novos elementos[2,7].

MÉTODOS ITERATIVOS ESTACIONÁRIOS

A partir de (1), define-se uma expressão geral para os métodos iterativos estacionários ou básicos da seguinte forma[8]

$$\mathbf{u}^{(n+1)} = \mathbf{G}\mathbf{u}^{(n)} + \mathbf{k} \quad n = 0, 1, 2, \ldots \tag{3}$$

onde $\mathbf{G} = \mathbf{I} - \mathbf{Q}^{-1}\mathbf{A}$ e $\mathbf{k} = \mathbf{Q}^{-1}\mathbf{b}$ são, respectivamente, a matriz de iteração do método e um vetor conhecido, sendo **Q** a matriz de partição do método.

Assumindo que $\mathbf{A}$ é não-singular, tem-se que $\bar{\mathbf{u}}$ é solução do sistema associado $(\mathbf{I}-\mathbf{G})\mathbf{u} = \mathbf{k}$, se e somente se $\bar{\mathbf{u}}$ é a única solução de (1), ou seja, $\bar{\mathbf{u}} = \mathbf{A}^{-1}\mathbf{b}$. Um método iterativo na forma (3), cujo sistema relacionado tem solução única $\bar{\mathbf{u}}$ coincidente com a solução de (1), é denominado *completamente consistente*. Sendo $\{\mathbf{u}^{(n)}\}$ a sequência de iterações determinada por (3), esta propriedade implica que se $\mathbf{u}^{(n)} = \bar{\mathbf{u}}$ para algum valor de (n), tem-se que $\mathbf{u}^{(n+1)} = \mathbf{u}^{(n+2)} = \ldots = \bar{\mathbf{u}}$. Além disso, se $\{\mathbf{u}^{(n)}\}$ converge para algum vetor $\hat{\mathbf{u}}$ então $\hat{\mathbf{u}} = \bar{\mathbf{u}}$.

Por outro lado, um método iterativo é *convergente* se para qualquer vetor inicial $\mathbf{u}^{(0)}$, a sequência de aproximações $\mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \ldots$ definida por (3) converge para $\bar{\mathbf{u}}$. A condição necessária e suficiente para a convergência é dada a partir do raio espectral de $\mathbf{G}$, ou seja, $S(\mathbf{G}) < 1$[4,8].

A convergência na iteração n é medida a partir do vetor de erro $\epsilon^{(n)} = \mathbf{u}^{(n)} - \bar{\mathbf{u}}$. Tomando-se uma norma adequada $||\cdot||$, tem-se que $||\epsilon^{(n)}|| \leq ||\mathbf{G}^n||\,||\epsilon^{(0)}||$. Assim, $||\mathbf{G}^n||$ fornece uma medida da redução da norma do erro após n iterações.

No entanto, não é necessário que os métodos definidos por (3) sejam convergentes, de acordo com a definição anterior. Exige-se apenas que os mesmos sejam simétricos. Para isso, deve existir uma matriz não-singular $\mathbf{W}$ tal que a matriz $\mathbf{W}(\mathbf{I}-\mathbf{G})\mathbf{W}^{-1}$ seja SPD. Se o método iterativo (3) é simétrico, então as seguintes propriedades são válidas[8]: os autovalores de $\mathbf{G}$ são reais; o maior autovalor algébrico $M(\mathbf{G})$ é menor que 1; os autovetores de $\mathbf{G}$ definem uma base para o espaço vetorial associado.

A seguir serão apresentados os métodos iterativos de Jacobi, Gauss-Seidel, SOR e SSOR. Para isso, expande-se a matriz $\mathbf{A}$ como

$$\mathbf{A} = \mathbf{D} - \mathbf{C}_L - \mathbf{C}_U \tag{4}$$

onde $\mathbf{D}$, $\mathbf{C}_L$ e $\mathbf{C}_U$ são, respectivamente, as matrizes diagonal, triangular inferior e superior de $\mathbf{A}$.

Método de Jacobi

Dada uma solução inicial $\mathbf{u}^{(0)}$, o método de Jacobi (JAC) consiste em obter para cada passo i a respectiva incógnita u_i $(i = 1, \ldots, N)$. De forma geral, tem-se para a iteração $(n+1)$

$$a_{ii}u_i^{(n+1)} = b_i - \sum_{j=1\,(j\neq i)}^{N} a_{ij}\,u_j^{(n)} \quad i = 1, \ldots, N \tag{5}$$

Observa-se que a ordem na qual as incógnitas são atualizadas é irrelevante. Por isso, este algoritmo é denominado como *método de deslocamentos simultâneos*. Observa-se que a convergência do método de Jacobi só é alcançada para casos onde a matriz $\mathbf{A}$ é fortemente diagonal dominante[4].

Para colunas ascendentes, uma implementação eficiente do método de Jacobi é efetuada considerando o algoritmo ao longo das colunas de $\mathbf{A}$. No entanto, independente do tipo de armazenamento, são necessários, além de $\mathbf{A}$, os vetores $\mathbf{u}^{(n)}$, $\mathbf{u}^{(n+1)}$ e $\mathbf{b}$, resultando em mais $3N$ posições de memória. Por sua vez, o número de operações por iteração é $N(2m-1)$.

Método de Gauss-Seidel

O método de Gauss-Seidel (GS) é análogo ao anterior, diferenciando-se pelo fato que as componentes já atualizadas da aproximação $\mathbf{u}^{(n)}$ são empregadas a medida que são calculadas. Logo

$$a_{ii}u_i^{(n+1)} = b_i - \sum_{j=1}^{i-1} a_{ij}u_j^{(n+1)} - \sum_{j=i+1}^{N} a_{ij}u_j^{(n)} \tag{6}$$

Neste método cada componente $u_i^{(n+1)}$ depende daquelas determinadas anteriormente, sendo a aproximação $\mathbf{u}^{(n+1)}$ função da ordem na qual as equações são examinadas. Desta maneira, o algoritmo GS é conhecido como *método de deslocamentos sucessivos*, enfatizando a dependência no ordenamento das equações. No que se refere à implementação, o espaço de memória e o custo por iteração são os mesmos do método de Jacobi.

Os métodos de Jacobi e Gauss-Seidel são efetivos na redução das amplitudes das componentes de alta frequência da aproximação $\mathbf{u}^{(n)}$, sendo portanto adequados em métodos multigrid[2].

Método de Sobre-relaxação

No método de sobre-relaxação (SOR), emprega-se um parâmetro de relaxação ω, visando acelerar a taxa de convergência das aproximações. Assim, de maneira geral tem-se

$$a_{ii}u_i^{(n+1)} = \omega\left(b_i - \sum_{j=1}^{i-1} a_{ij}u_j^{(n+1)} - \sum_{j=i+1}^{N} a_{ij}u_j^{(n)}\right) + (1-\omega)a_{ii}u_i^{(n)} \quad i = 1,\ldots,N \tag{7}$$

Verifica-se que para $\omega = 1$, obtém-se o método de Gauss-Seidel; para $\omega < 1$ e $\omega > 1$ tem-se, respectivamente, sub e sobre-relaxação. É possível ainda selecionar ω, de tal forma que a taxa de convergência seja sensivelmente superior àquela obtida nos métodos de Jacobi e GS. Para problemas elípticos, toma-se $\omega \approx 1,8$ como valor ótimo. Pode-se empregar algoritmos adaptáveis para determinar ω, utilizando estimativas para a taxa na qual a iteração corrente está convergindo[4,8].

A implementação deste método é análoga ao caso anterior, devendo-se considerar apenas as multiplicações por ω a cada passo, implicando num custo de $2Nm$ operações por iteração.

Método de Sobre-relaxação Simétrico

O método de sobre-relaxação simétrico (SSOR) é definido pelas seguintes equações

$$a_{ii}u_i^{(n+1/2)} = \omega\left(b_i - \sum_{j=1}^{i-1} a_{ij}u_j^{(n+1/2)} - \sum_{j=i+1}^{N} a_{ij}u_j^{(n)}\right) + (1-\omega)a_{ii}u_i^{(n)} \tag{8}$$

$$i = 1,\ldots,N$$

$$a_{ii}u_i^{(n+1)} = \omega\left(b_i - \sum_{j=1}^{i-1} a_{ij}u_j^{(n+1/2)} - \sum_{j=i+1}^{N} a_{ij}u_j^{(n+1)}\right) + (1-\omega)a_{ii}u_i^{(n+1/2)} \qquad (9)$$

$i = N, \ldots, 1$

Inicialmente, calculam-se $u_1^{(n+1/2)}, \ldots, u_N^{(n+1/2)}$ aplicando-se o método SOR a (8). Posteriormente, determinam-se $u_N^{(n+1)}, \ldots, u_1^{(n+1)}$ usando SOR de modo reverso.

Para **A** SPD, demonstra-se que o método SSOR converge para todo $0 < \omega < 2$, sendo a convergência relativamente insensível para valores neste intervalo. A principal aplicação do algortimo SSOR é como pré-condicionador para métodos iterativos com matrizes simétricas. Tomando-se um mesmo valor ótimo para ω, a taxa de convergência do SSOR é inferior ao SOR[4], sendo o custo de cada iteração duas vezes superior, ou seja, $4Nm$.

ACELERAÇÃO POLINOMIAL

Os procedimentos de aceleração polinomial visam aumentar a taxa de convergência dos métodos iterativos básicos discutidos anteriormente. Para isso, determina-se uma nova sequência de aproximações tomando-se combinações lineares dos vetores obtidos nas iterações.

Suponha que o método básico seja completamente consistente, simétrico e descrito por

$$\mathbf{w}^{(n+1)} = \mathbf{G}\mathbf{w}^{(n)} + \mathbf{k} \quad n = 0, 1, 2, \ldots \qquad (10)$$

Para acelerar a convergência das aproximações $\mathbf{w}^{(n)}$, toma-se uma nova sequência $\mathbf{u}^{(n)}$ dada pela seguinte combinação linear

$$\mathbf{u}^{(n)} = \sum_{i=0}^{n} \alpha_{n,i}\, \mathbf{w}^{(i)} \quad \text{com} \quad \sum_{i=0}^{n} \alpha_{n,i} = 1 \quad n = 0, 1, 2 \ldots \qquad (11)$$

A partir daí, demosntra-se que o erro na n-ésima iteração pode ser escrito como

$$\epsilon^{(n)} = Q_n(\mathbf{G})\epsilon^{(0)} \qquad (12)$$

onde $Q_n(\mathbf{G}) = \alpha_{n,0}\mathbf{I} + \alpha_{n,1}\mathbf{G} + \ldots + \alpha_{n,n}\mathbf{G}^n$ é um polinômio matricial. Tomando-se $Q_n(x) = \alpha_{n,0} + \alpha_{n,1}x + \ldots + \alpha_{n,n}x^n$ como o polinômio algébrico associado a $Q_n(\mathbf{G})$ observa-se que $Q_n(1) = 1$.

Devido a forma da equação (12), o procedimento definido por (10) e (11) é denominado *método de aceleração polinomial*. No entanto, através de (11), verifica-se um alto custo computacional em termos de processamento e armazenamento. Para

contornar este problema, basta tomar os polinômios $Q_n(x)$ dados pela seguinte relação de recorrência[8]

$$Q_0(x) = 1 \qquad Q_1(x) = \gamma_1 x - \gamma_1 + 1$$
$$Q_{n+1}(x) = \rho_{n+1}(\gamma_{n+1}x + 1 - \gamma_{n+1})Q_n(x) + (1 - \rho_{n+1})Q_{n-1}(x) \quad n \geq 1$$

onde γ_i e ρ_i são números reais. Além disso, observa-se que $Q_n(1) = 1 \quad \forall n \geq 0$. A partir daí, demonstra-se que as aproximações $\mathbf{u}^{(n+1)}$ podem ser determinadas através da relação com 3 termos

$$\begin{aligned} \mathbf{u}^{(1)} &= \gamma_1(\mathbf{G}\mathbf{u}^{(0)} + \mathbf{k}) + (1 - \gamma_1)\mathbf{u}^{(0)} \\ \mathbf{u}^{(n+1)} &= \rho_{n+1}\{\gamma_{n+1}(\mathbf{G}\mathbf{u}^{(n)} + \mathbf{k}) + (1 - \gamma_{n+1})\mathbf{u}^{(n)}\} + (1 - \rho_{n-1})\mathbf{u}^{(n-1)} \quad n \geq 1 \end{aligned} \tag{13}$$

Existem várias sequências de polinômios $\{Q_n(x)\}$ satisfazendo (13), como por exemplo aquelas associadas aos métodos de Gradiente Conjugado e de Chebyshev. No método de Gradiente Conjugado, procura-se determinar a sequência $\{Q_n(x)\}$ minimizando a norma de energia do erro $||\epsilon^{(n)}||_{\mathbf{A}}$. Por sua vez, no algoritmo de Chebyshev, toma-se $\{Q_n(x)\}$ tal que o raio espectral $S(Q_n(\mathbf{G}))$ seja pequeno. Várias outras técnicas de aceleração não-polinomial dos métodos básicos podem ser empregadas, como por exemplo os algoritmos multigrid[2,3].

ACELERAÇÃO DE CHEBYSHEV

O polinômio de Chebyshev de ordem n é dado pela expressão recursiva[8,13]

$$T_0(w) = 1 \qquad T_1(w) = w \qquad T_{n+1}(w) = 2wT_n(w) - T_{n-1}(w) \tag{14}$$

Para minimizar o raio espectral $S(Q_n(\mathbf{G}))$, definem-se parâmetros de iteração depedentes dos autovalores extremos $m(\mathbf{G})$ e $M(\mathbf{G})$ da matriz de iteração $\mathbf{G}$. Demonstra-se que o único polinômio $P_n(x)$ minimizando $S(Q_n(\mathbf{G}))$ é dado por[8]

$$P_n(x) = \frac{T_n\left(\frac{2x - m(\mathbf{G}) - M(\mathbf{G})}{M(\mathbf{G}) - m(\mathbf{G})}\right)}{T_n\left(\frac{2 - m(\mathbf{G}) - M(\mathbf{G})}{M(\mathbf{G}) - m(\mathbf{G})}\right)} \tag{15}$$

No procedimento ótimo de aceleração de Chebyshev, as aproximações são determinadas pela seguinte relação de recorrência

$$\mathbf{u}^{(n+1)} = \bar{\rho}_{n+1}\{\bar{\gamma}\delta^{(n)} + \mathbf{u}^{(n)} + \mathbf{u}^{(n-1)}\} + \mathbf{u}^{(n-1)} \tag{16}$$

onde $\delta^{(n)} = \mathbf{G}\mathbf{u}^{(n)} + \mathbf{k} - \mathbf{u}^{(n)} = \mathbf{u}^{(n+1)} - \mathbf{u}^{(n)}$ é denominado vetor de pseudo-resíduo, obtido resolvendo-se o sistema $\mathbf{Q}\delta^{(n)} = \mathbf{b} - \mathbf{A}\mathbf{u}^{(n)}$. Na expressão anterior, $\bar{\gamma}$ e $\bar{\rho}_{n+1}$ são dados por

$$\begin{aligned} &\bar{\rho}_1 = 1 \quad \bar{\rho}_2 = \left(1 - \frac{1}{2}\bar{\sigma}^2\right)^{-1} \quad \bar{\rho}_{n+1} = \left(1 - \frac{1}{4}\bar{\sigma}^2\bar{\rho}_n\right)^{-1} \quad n \geq 2 \\ &\bar{\gamma} = \frac{2}{2 - m(\mathbf{G}) - M(\mathbf{G})} \quad \bar{\sigma} = \frac{M(\mathbf{G}) - m(\mathbf{G})}{2 - m(\mathbf{G}) - M(\mathbf{G})} \end{aligned} \tag{17}$$

O termo ótimo é empregado devido a utilização dos autovalores $m(\mathbf{G})$ e $M(\mathbf{G})$ da matriz de iteração $\mathbf{G}$. Como em geral, estas constantes não estão disponíveis, pode-se aplicar um algoritmo adaptável visando melhorar estimativas iniciais para os autovalores extremos de $\mathbf{G}$[8]. Neste trabalho, calculam-se estas estimativas através do algoritmo de Lanczos[13].

Considerou-se a aceleração de Chebyshev aplicada ao método SSOR, denotado por CHSS. A implementação está baseada em[14], necessitando $5N$ posições de memória adicionais para os vetores e $N(4m + 1)$ operações por iteração.

MÉTODO DE GRADIENTE CONJUGADO

Seja o funcional quadrático $f(\mathbf{u}) = \frac{1}{2}(\mathbf{u}, \mathbf{A}\mathbf{u}) - (\mathbf{b}, \mathbf{u}) + c \quad (\mathbf{b} \in \Re^n, \ c \in \Re)$. Como $\mathbf{A}$ é SPD, observa-se que o problema de minimizar f é análogo à solução do sistema de equações (1). Assim, tem-se uma classe de métodos iterativos cujo objetivo é determinar um mínimo local de um funcional quadrático. A solução iterativa deste problema de mínimo possui a seguinte forma geral[1]

$$\mathbf{u}^{(n+1)} = \mathbf{u}^{(n)} + \tau_n \mathbf{d}^{(n)} \quad n = 0, 1, 2, \ldots \tag{18}$$

sendo $\mathbf{d}^{(n)}$ uma direção de busca e τ_n um escalar, selecionado de tal maneira a minimizar $f(\mathbf{u})$ ao longo da linha passando por $\mathbf{u}^{(n)}$ na direção $\mathbf{d}^{(n)}$.

Supondo que $f \in C^1(\Re^n)$, demonstra-se que dentre todas as direções $\mathbf{d}$ num ponto $\mathbf{u}$, aquela onde f decresce mais rapidamente, na vizinhaça de $\mathbf{u}$, é dada pelo gradiente de f[1]

$$\mathbf{d} = \mathbf{g}(\mathbf{u}) = -\nabla f(\mathbf{u}) = \mathbf{b} - \mathbf{A}\mathbf{u} \tag{19}$$

Os procedimentos para a determinação de τ_n são conhecidos como algoritmos de busca. No caso de um funcional quadrático, τ_n pode ser obtido de forma simples, ou seja

$$\tau_n = -\frac{(\mathbf{d}^{(n)}, \mathbf{g}^{(n)})}{(\mathbf{d}^{(n)}, \mathbf{A}\mathbf{d}^{(n)})} \tag{20}$$

O método de Gradiente Conjugado (GC) define a direção de busca como

$$\mathbf{d}^{(n+1)} = -\mathbf{g}^{(n+1)} + \beta_n \mathbf{d}^{(n)} \quad n = 0, 1, 2, \ldots \tag{21}$$

sendo $\mathbf{d}^{(0)} = -\mathbf{g}^{(0)}$ e $\beta_0, \beta_1, \ldots$ constantes determinadas de tal forma a minimizar o erro $||\epsilon^{(n)}||_A$ a cada iteração. Demonstra-se que os parâmetros β_n são dados por[1]

$$\beta_n = \frac{(\mathbf{g}^{(n+1)}, \mathbf{A}\mathbf{d}^{(n)})}{(\mathbf{d}^{(n)}, \mathbf{A}\mathbf{d}^{(n)})} \tag{22}$$

Multiplicando (18) por $\mathbf{A}$, subtraindo $\mathbf{b}$ em ambos os lados da equação e substituindo (19) no resultado, obtém-se a relação

$$\mathbf{g}^{(n+1)} = \mathbf{g}^{(n)} + \tau_n \mathbf{A}\mathbf{d}^{(n)} \tag{23}$$

Os parâmetros β_n e τ_n podem ainda ser reescritos como[1,8]

$$\beta_n = \frac{(\mathbf{g}^{(n+1)}, \mathbf{g}^{(n+1)})}{(\mathbf{g}^{(n)}, \mathbf{g}^{(n)})} \qquad \tau_n = \frac{(\mathbf{g}^{(n)}, \mathbf{g}^{(n)})}{(\mathbf{d}^{(n)}, \mathbf{A}\mathbf{d}^{(n)})} \tag{24}$$

A partir destas equações, a forma computacionalmente mais conveniente do método de gradiente conjugado, com $\mathbf{u}^{(0)}$ dado e $\mathbf{d}^{(0)} = -\mathbf{g}^{(0)}$, pode ser resumida em[1]

$$\begin{aligned}
\tau_n &= \frac{(\mathbf{g}^{(n)}, \mathbf{g}^{(n)})}{(\mathbf{d}^{(n)}, \mathbf{A}\mathbf{d}^{(n)})} \\
\mathbf{u}^{(n+1)} &= \mathbf{u}^{(n)} + \tau_n \mathbf{d}^{(n)} \\
\mathbf{g}^{(n+1)} &= \mathbf{g}^{(n)} + \tau_n \mathbf{A}\mathbf{d}^{(n)} \\
\beta_n &= \frac{(\mathbf{g}^{(n+1)}, \mathbf{g}^{(n+1)})}{(\mathbf{g}^{(n)}, \mathbf{g}^{(n)})} \\
\mathbf{d}^{(n+1)} &= -\mathbf{g}^{(n+1)} + \beta_n \mathbf{d}^{(n)}
\end{aligned} \tag{25}$$

Em relação ao espaço de memória do algoritmo anterior, deve-se armazenar um total de 4 vetores, equivalentes a 4N posições de memória. No que se refere ao número de operações, tem-se o produto $\mathbf{Ad}$, 2 produtos internos e 3 fórmulas de recorrência, respectivamente, com $(2m-1)N$, $2N$ e $3N$ operações, totalizando $N(2m+4)$ por iteração.

Método de Gradiente com Pré-condicionamento

A taxa de convergência do método de Gradiente Conjugado depende do número de condição espectral $\kappa(\mathbf{A})$[1]. O objetivo da técnica de pré-condicionamento é minimizar, ao invés de f, o funcional quadrático $\tilde{f}(\mathbf{v}) = \frac{1}{2}(\mathbf{v}, \tilde{\mathbf{A}}\mathbf{v}) - (\tilde{\mathbf{b}}, \mathbf{v}) + \tilde{c}$ de tal forma que $\kappa(\tilde{\mathbf{A}}) < \kappa(\mathbf{A})$, aumentando assim a taxa de convergência do método.

O funcional $\tilde{f}$ é obtido tomando-se uma matriz não-singular $\mathbf{E}$ e a tranformação $\mathbf{v} = \mathbf{E}^T\mathbf{u}$, a qual substituída na expressão para f resulta em $\tilde{f}$, com $\tilde{\mathbf{A}} = \mathbf{E}^{-1}\mathbf{A}\mathbf{E}^{-T}$, $\tilde{\mathbf{b}} = \mathbf{E}^{-1}\mathbf{b}$ e $\tilde{c} = c$. Observa-se que $\mathbf{C} = \mathbf{E}\mathbf{E}^T$ é positiva-definida, sendo denominada *matriz de pré-condicionamento*.

Pode-se aplicar o algoritmo de GC ao funcional $\tilde{f}$, definindo o método de GC com pré-condicionamento transformado pois a sequência de aproximações $\mathbf{v}^{(n)}$ converge para $\bar{\mathbf{v}} = \tilde{\mathbf{A}}^{-1}\tilde{\mathbf{b}}$. Para se obter a solução do problema original efetua-se a transformação $\mathbf{u} = \mathbf{E}^{-T}\mathbf{v}$.

Logo, a partir de (25), a versão pré-condicionada do método de gradiente não-transformado é dada por[1]

$$\begin{aligned}
\tau_n &= \frac{(\mathbf{g}^{(n)}, \mathbf{g}^{(n)})}{(\mathbf{d}^{(n)}, \mathbf{A}\mathbf{d}^{(n)})} \\
\mathbf{u}^{(n+1)} &= \mathbf{u}^{(n)} + \tau_n \mathbf{d}^{(n)} \\
\mathbf{g}^{(n+1)} &= \mathbf{g}^{(n)} + \tau_n \mathbf{A}\mathbf{d}^{(n)} \\
\mathbf{h}^{(n+1)} &= \mathbf{C}^{-1}\mathbf{g}^{(n+1)} \\
\beta_n &= \frac{(\mathbf{g}^{(n+1)}, \mathbf{h}^{(n+1)})}{(\mathbf{g}^{(n)}, \mathbf{h}^{(n)})} \\
\mathbf{d}^{(n+1)} &= -\mathbf{h}^{(n+1)} + \beta_n \mathbf{d}^{(n)}
\end{aligned} \tag{26}$$

Se $\mathbf{C}$ possui a propriedade tal que $\kappa(\tilde{\mathbf{A}}) < \kappa(\mathbf{A})$, logo a versão pré-condicionada apresenta uma taxa de convergência maior que a versão original (25). Observa-se ainda em (26), a presença da matriz $\mathbf{C}$ e não de $\mathbf{E}$. Assim, como qualquer matriz positiva-definida $\mathbf{C}$ possui várias fatorações do tipo $\mathbf{C} = \mathbf{E}\mathbf{E}^T$, pode-se tomar a matriz de pré-condicionamento na classe de matrizes positiva-definidas.

Uma matriz de pré-condicionamento adequada deve possuir as seguintes propriedades[1]: $\kappa(\tilde{\mathbf{A}}) \ll \kappa(\mathbf{A})$; os termos de $\mathbf{C}$ devem ser determinados de forma rápida; a resolução do sistema $\mathbf{C}\mathbf{h}^{(n+1)} = \mathbf{g}^{(n+1)}$ deve ser mais eficiente que a solução de (1).

Consideram-se como pré-condicionadores a matriz diagonal $\mathbf{D}$, assim como as matrizes de partição dos métodos SSOR e Gauss-Seidel simétrico[15] dadas respectivamente por

$$\mathbf{C} = \left(\frac{1}{\omega}\mathbf{D} - \mathbf{C_L}\right)\left(\frac{1}{\omega}\mathbf{D}\right)^{-1}\left(\frac{1}{\omega}\mathbf{D} - \mathbf{C_L}\right)^T \qquad \mathbf{C} = (2\mathbf{D} - \mathbf{C_L})(2\mathbf{D})^{-1}(2\mathbf{D} - \mathbf{C_L})^T \tag{27}$$

sendo $\mathbf{C}_L$ a matriz triangular inferior de $\mathbf{A}$ indicada em (4).

A partir da versão transformada do método de GC, obtém-se uma variante pré-condicionada bastante eficiente[1]. Neste caso, tem-se $(2m + 7)N$ operações por iteração e $5N$ posições de memória para os vetores auxiliares. Os métodos com pré-condicionamento serão indicados por GCD, GCSS e GCGS respectivamente para os casos de matriz de pré-condicionamento diagonal, SSOR e GS simétrico.

CRITÉRIOS DE CONVERGÊNCIA

Como visto anteriormente, os métodos iterativos produzem uma sequência de aproximações $\{\mathbf{u}^{(n)}\}$ convergindo para a solução $\bar{\mathbf{u}}$ de (1). Assim, torna-se fundamental estabelecer critérios de convergência visando terminar o processo iterativo. Um bom critério de parada deve possuir as seguintes características[4]: identificar quando o erro $\epsilon^{(n)}$ é suficientemente pequeno; parar se o erro não está decrescendo ou se a taxa de decaimento é muito baixa; limitar a quantidade máxima de iterações.

Deseja-se parar as iterações quando a magnitude do vetor $\epsilon^{(n)}$ for pequena. No entanto, é impossível empregar $\epsilon^{(n)}$ como critério de parada. Assim, geralmente procura-se limitar o erro $\epsilon^{(i)}$ em função do resíduo $\mathbf{r}^{(i)}$. Para comparar a performance dos métodos iterativos abordados, adotou-se o critério de convergência baseado na norma relativa do resíduo $||\mathbf{r}^{(n)}||_2$, ou seja

$$||\mathbf{r}^{(n)}||_2 = \frac{||\mathbf{A}\mathbf{u}_{it}^{(n)} - \mathbf{b}||_2}{||\mathbf{b}||_2} < \xi \tag{28}$$

Visando uniformizar os resultados, selecionou-se a precisão ξ, de forma que o erro relativo entre as soluções direta de Gauss $\mathbf{u}_d$ e iterativa $\mathbf{u}_{it}$ fosse no máximo $0,1$ %, ou seja

$$||\mathbf{e}||_2 = \frac{||\mathbf{u}_d - \mathbf{u}_{it}||_2}{||\mathbf{u}_d||_2} < 0,1 \ \% \tag{29}$$

Observa-se que são necessárias $2Nm$ operações para o cálculo do critério de convergência (28), sendo equivalente a uma iteração do método. Para reduzir este custo, pode-se verificar a convergência a cada 5 ou 10 iterações. No entanto, para os métodos de gradiente conjugado (GC, GCD, GCSS, GCGS), o gradiente em cada iteração corresponde ao próprio resíduo. Assim, tem-se o custo do critério apenas para os métodos de Jacobi, GS, SOR e CHSS.

CLASSES IMPLEMENTADAS

Para implementar as estruturas de dados e os algortimos discutidos anteriormente, empregou-se a linguagem C++[6]. Desenvolveram-se as classes Symmetric, SymmetricSkyline e SymmetricSparse, respectivamente para matrizes simétricas, em colunas ascendentes e esparsa[2]. As técnicas de Gauss e iterativas são métodos destas classes. Logo, dado um arquivo com os dados de uma malha, é possível determinar a topologia da matriz, inicializar os seus elementos e selecionar o algoritmo a ser empregado, especificando parâmetros tais como o número máximo de iterações, critério de convergência, precisão, dentre outros.

Estas classes foram incorporadas a um ambiente orientado por objetos com ferramentas para geração de malhas, análise por elementos finitos e visualização de resultados[16].

ESTUDO DE CASOS

A Tabela I resume o número de operações e o espaço de memória dos vetores auxiliares para todos os métodos discutidos anteriormente. O parâmetro N_{it} indica o número de iterações para a convergência no caso dos algoritmos iterativos.

Observa-se que as expressões na Tabela I consideram apenas o núcleo principal dos métodos. A demanda computacional real é superior aquela indicada devido a fatores como gerenciamento de memória, expressões condicionais, sincronismos, dentre outros. No entanto, estas relações são significativas para efeito de uma análise comparativa dos métodos, além do que o cálculo do número de operações a partir destas expressões independe da implementação adotada para os vários métodos. Deve-se ressaltar ainda que não estão contabilizados os custos de renumeração de nós, do procedimento de Gauss simbólico em matrizes esparsas e do cálculo das estimativas dos autovalores extremos através do algoritmo de Lanczos no método CHSS.

Método	Operações	Memória
Gauss	$N + \sum m_i^U (m_i^U + 7)/2$	N
Jacobi	$N_{it}N(2m-1)$	$3N$
GS	$N_{it}N(2m-1)$	$3N$
SOR	$2N_{it}Nm$	$3N$
SSOR	$4N_{it}Nm$	$3N$
CHSS	$N_{it}N(4m+1)$	$5N$
GC	$N_{it}N(2m+4)$	$4N$
GCD	$N_{it}N(2m+5)$	$5N$
GCSS	$N_{it}N(2m+7)$	$5N$
GCGS	$N_{it}N(2m+7)$	$5N$

Tabela I. Número de operações e espaço de memória dos vetores auxiliares para os métodos direto e iterativos considerados

Para avaliar o desempenho dos vários algoritmos apresentados, considerou-se primeiramente os 3 problemas planos ilustrados na Figura 2, tomando-se 4 malhas de elementos triangulares de 6 nós para cada um destes casos. No cilindro vertical, tem-se uma estrutura axissimétrica sem singularidades; a placa com furo é um problema de estado plano de tensão com singularidade moderada; e finalmente o exemplo de fratura, constitui-se novamente num caso de estado plano, mas com uma forte singularidade.

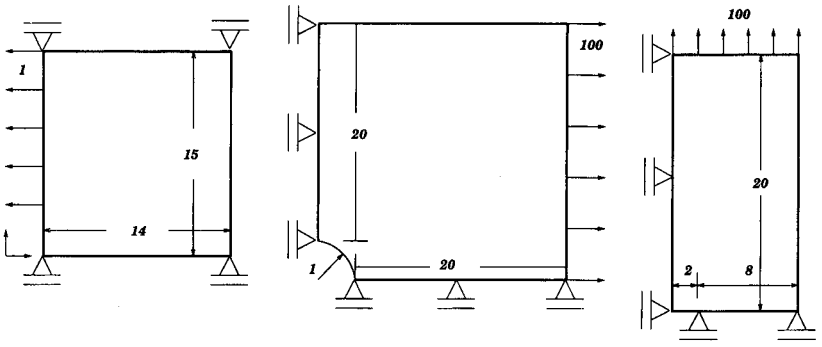

Figura 2. Problemas planos analisados: cilindro vertical, placa com furo e fratura ($E = 21 \times 10^5$, $\nu = 0,3$)

Cilindro vertical								
Nós	Elems	Eqs	m_{sky}	NElems_{sky}	m_{esp}^{dir}	$\text{NElems}_{esp}^{dir}$	m_{esp}^{ite}	$\text{NElems}_{esp}^{ite}$
531	244	1020	69,4	70739	30,7	31327	11,2	11397
1595	760	3116	134,1	417803	47,9	149306	11,5	35931
3069	1482	6036	183,3	1106335	57,7	348535	11,7	70395
5293	2578	10452	242,7	2536176	67,9	709156	11,7	122733

Placa com furo								
Nós	Elems	Eqs	m_{sky}	NElems_{sky}	m_{esp}^{dir}	$\text{NElems}_{esp}^{dir}$	m_{esp}^{ite}	$\text{NElems}_{esp}^{ite}$
533	248	1020	62,3	63581	34,4	35073	11,3	11481
1558	745	3034	111,5	338319	45,9	139123	1,5	35034
3172	1537	6226	153,1	952947	55,6	345996	11,7	72714
5078	2477	10014	214,1	2143473	65,3	654376	11,8	117678

Fratura								
Nós	Elems	Eqs	m_{sky}	NElems_{sky}	m_{esp}^{dir}	$\text{NElems}_{esp}^{dir}$	m_{esp}^{ite}	$\text{NElems}_{esp}^{ite}$
563	260	1078	70,4	75838	31,8	34297	11,2	12060
1538	735	3002	112,3	337056	44,1	132369	11,5	34650
3202	1547	6284	160,0	1005527	59,3	372732	11,7	73209
5216	2539	10282	211,0	2169416	66,2	680468	11,7	120600

Tabela II. Atributos das malhas dos 3 problemas planos

A Tabela II apresenta as principais características de cada uma das malhas: número de nós; número de elementos; número de equações; m_{sky}, NElems_{sky} = altura média das colunas e número de coeficientes da matriz em colunas ascendentes com renumeração pelo algoritmo de Cuthill-Mckee; m_{esp}^{dir}, $\text{NElems}_{esp}^{dir}$ = número médio de elementos por linha e número total de coeficientes na matriz esparsa com renumeração pelo algoritmo

de grau mínimo; m_{esp}^{ite}, $\text{NElems}_{esp}^{ite}$ = número médio de elementos por linha e número total de coeficientes não-nulos na matriz esparsa para os métodos iterativos.

Observa-se que na geração das malhas procurou-se apenas aumentar o número de equações. Assim, não se aplicou nenhum procedimento adaptável visando obter uma malha ótima para a análise de cada um dos problemas. Para os métodos iterativos, tomou-se como solução inicial $\mathbf{u}^{(0)} = \mathbf{0}$.

Aplicaram-se os seguintes métodos na análise destes 3 problemas: Gauss, SOR, CHSS, GC, GCD, GCSS e GCGS. Obteve-se uma grande quantidade de resultados, tais como número de iterações para convergência, número de operações, espaço de memória, normas do resíduo e erro relativo[2]. Neste trabalho, faz-se um resumo de todas as simulações, procurando ressaltar os aspectos mais importantes. Considera-se em detalhes apenas o problema de fratura estando as malhas adotadas para este caso ilustradas na Figura 3.

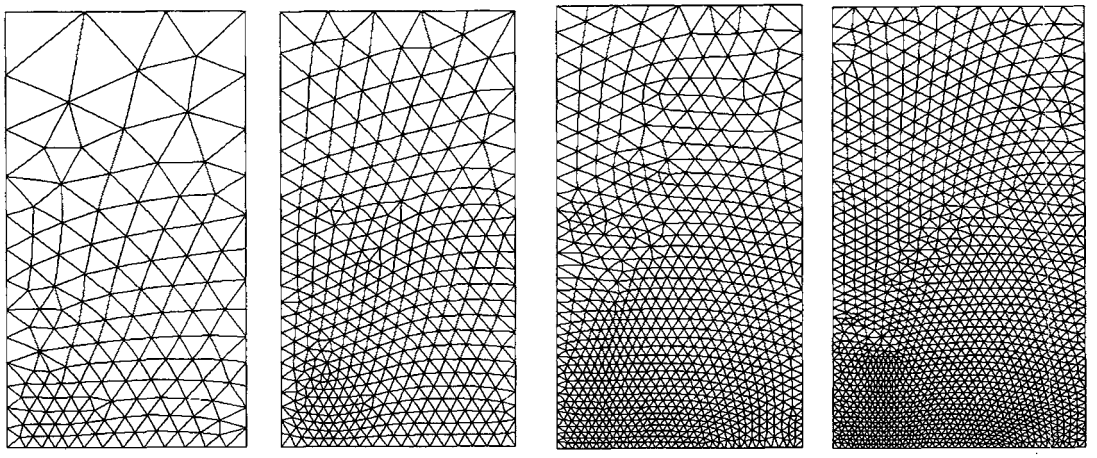

Figura 3. Malhas geradas para o problema de fratura

A Tabela III apresenta, para os métodos iterativos, o número de iterações para convergência em cada uma das 4 malhas dos problemas analisados, empregando-se uma precisão $\xi = 10^{-3}$. Apenas em alguns casos indicados, usaram-se outros valores de ξ, com o objetivo de satisfazer a condição (29).

Na Figura 4, tem-se para o problema de fratura o número de operações em MFlop para as técnicas iterativas e método de Gauss skyline e esparso para as 4 malhas. Foram ajustadas retas da forma $y = Cx^{\alpha}$ em escala log-log para cada um dos métodos, estando os coeficientes angulares α indicados na Tabela III. A Figura 5 ilustra o comportamento do resíduo do critério de convergência para a primeira malha da Figura 3.

Cilindro vertical					
Método	1	2	3	4	α
Gauss - esparsa	–	–	–	–	1,83
Gauss - skyline	–	–	–	–	2,08
SOR	116	337	677	1121	2,00
CHSS	101	175	257	348	1,55
GC	124	193	265	355[1]	1,47
GCD	102	162	187	245	1,38
GCSS	75	111	165	220	1,48
GCGS	57	91	102	132	1,36

Placa com furo					
Método	1	2	3	4	α
Gauss - esparsa	–	–	–	–	1,68
Gauss - skyline	–	–	–	–	2,07
SOR	287[1]	1045[2]	1783[1]	3040[1]	2,03
CHSS	127	228	317	481[1]	1,58
GC	138	247	351	448	1,53
GCD	129	203	285	429	1,52
GCSS	64	142	204	257	1,62
GCGS	73	109	157	236	1,51

Fratura					
Método	1	2	3	4	α
Gauss - esparsa	–	–	–	–	1,78
Gauss - skyline	–	–	–	–	1,96
SOR	369[1]	1139[1]	2242[1]	3704[1]	2,04
CHSS	147	252	373	476	1,54
GC	153	250	378	477	1,53
GCD	140	237	349	447	1,53
GCSS	86	139[3]	207	251[4]	1,50
GCGS	79	131	191	244	1,52

Tabela III. Número de iterações para os métodos iterativos e coeficientes das retas ajustadas: 1-Precisão de 2,5E-4. 2-Precisão de 1,0E-4. 3-Precisão de 5,0E-3. 4-Precisão de 2,5E-3.

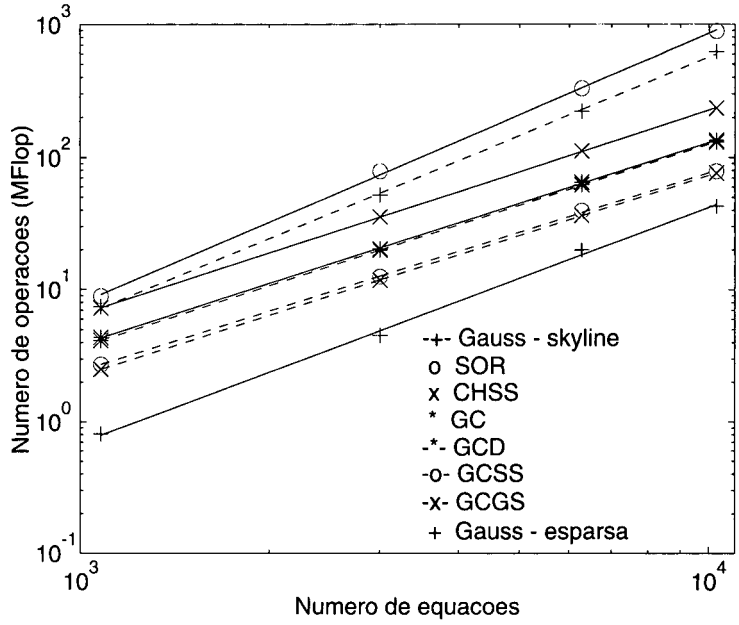

Figura 4. Número de operações (MFlop) para os métodos de Gauss e iterativos no problema de fratura

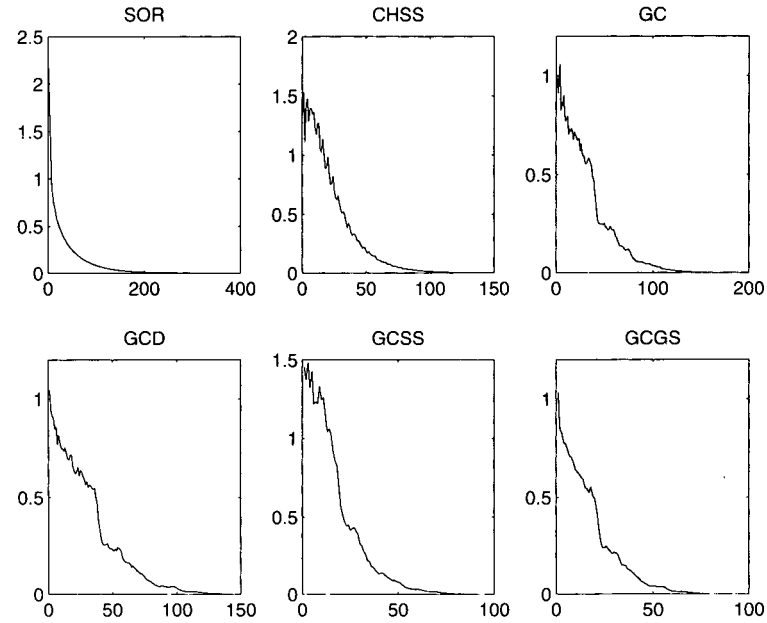

Figura 5. Fratura: comportamento do resíduo $||\mathbf{r}||_2$ do critério de convergência na primeira malha

ANÁLISE DOS RESULTADOS

A partir dos resultados obtidos observa-se que:

- Os termos da matriz do sistema e dos vetores foram tratados com precisão dupla, ocupando, portanto, 8 bytes. Assim, as expressões para o espaço de memória devem ser multiplicadas pelo fator 8/1024, resultando valores em Kbytes.
- Para a primeira malha dos 3 problemas, com cerca de 1000 equações, o método SOR apresenta, em termos do número de iterações, uma performance comparável aos métodos CHSS, GC e GCD. A medida que o número de equações aumenta, verifica-se uma queda progressiva na eficiência deste método.
- O número de iterações para a convergência com o método CHSS é inferior ao GC na maioria dos resultados obtidos. No entanto, o seu custo computacional é superior devido ao fator $4m$ presente na expressão para o cálculo do número de operações.

Como indicado em[8], a performance do algoritmo GC com pré-condicionadores é bastante superior ao CHSS. As estimativas para os autovalores obtidas pelo método de Lanczos permitiram uma redução significativa do número de iterações, comparando-se com o procedimento adaptável dado em[8]. Uma das vantagens da aceleração de Chebyshev, em relação ao método de Gradiente Conjugado, está em não efetuar produtos internos, facilitando a sua paralelização. Os produtos internos constituem-se em pontos de sincronismo em versões paralelas do algoritmo de GC[4].

- No método GCD, o número de iterações é inferior em relação aos procedimentos CHSS e GC. No entanto, a medida que a singularidade torna-se mais forte, verifica-se um equilíbrio nos 3 procedimentos.
- As versões pré-condicionadas do método de Gradiente Conjugado GCSS e GCGS permitiram reduzir significativamente o número de iterações em relação aos demais métodos. Na maioria dos casos, o algoritmo GCGS foi superior ao GCSS.
- Para métodos iterativos, torna-se essencial utilizar um armazenamento em matrizes esparsas. Como pode ser visto na Tabela II, no caso das 3 malhas mais finas, apenas cerca de 5 % dos elementos são não-nulos em relação ao caso de colunas ascendentes. No método de Gauss, também se verifica uma redução no número de elementos, mas não tão substancial devido ao processo de fatoração, onde novos coeficientes não-nulos são gerados.
- Em geral, os métodos iterativos de gradiente conjugado são superiores em relação ao procedimento de Gauss em colunas ascendentes. No entanto, considerando apenas o número de operações, o método de Gauss esparso é superior até um certo número de equações, dependente do problema em estudo. Na Figura 4, extrapolando os resultados, verifica-se o cruzamento entre as retas dos métodos de Gauss e GCGS, indicando um mesmo custo computacional, para 81181 equações.

Empregando o mesmo procedimento para os exemplos de cilindro vertical e placa com furo, foram obtidos gráficos semelhantes à Figura 4, identificando os cruzamentos das retas de Gauss e GCGS, respectivamente, para 6438 e 182310 equações[2].

- O problema de fratura apresenta uma forte singularidade, implicando num maior número de iterações para a convergência em relação a placa com furo. Por sua vez, este problema com singularidade moderada demanda mais iterações que o caso de cilindro vertical. Portanto, como esperado, a presença de singularidades afeta a performance dos métodos iterativos.
- Em relação ao comportamento do resíduo, ilustrado na Figura 5, observa-se um decaimento suave para o método SOR. Para as demais técnicas, o resíduo decresce em patamares ocorrendo maiores instabilidades nas iterações iniciais. De forma geral, tem-se uma queda mais instável do resíduo a medida que a singularidade aumenta.

Das simulações numéricas anteriores, verifica-se que o algoritmo GCGS é superior comparando-se com os demais métodos iterativos. Tomou-se o problema de fratura com dimensões 100 vezes maiores que aquelas indicadas na Figura 2, gerando-se 7 malhas com elementos triangulares de 6 nós, estando as características destas malhas indicadas na Tabela IV. Além disso, analisou-se a viga tridimensional ilustrada na Figura 6 com a Tabela V apresentando os parâmetros das 4 malhas empregadas. O objetivo foi estudar o comportamento dos métodos de Gauss e GCGS a medida que se aumenta o número de equações.

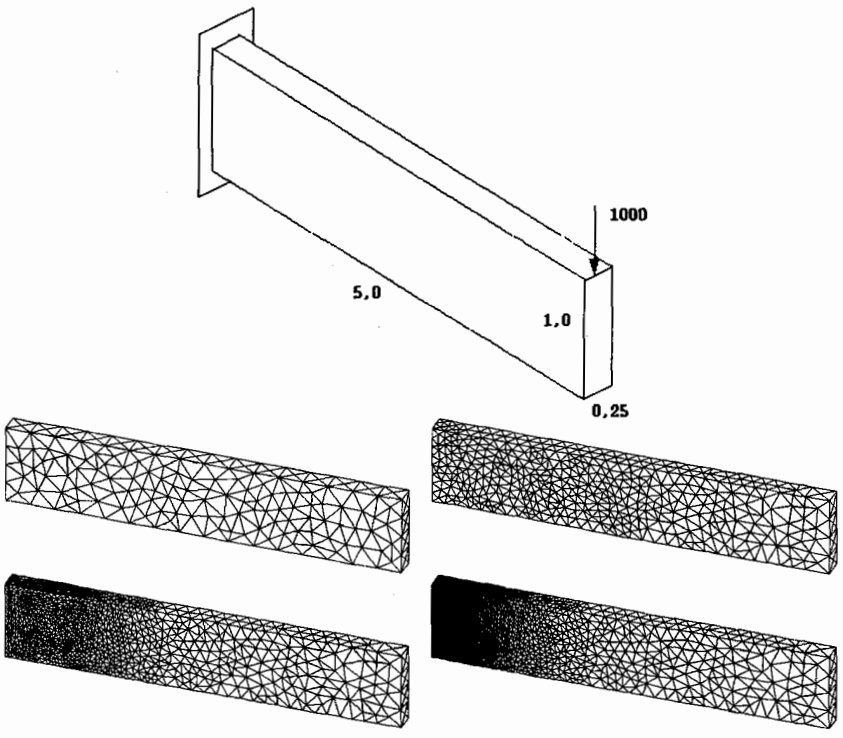

Figura 6. Malhas geradas para a viga tridimensional ($E = 1,0 \times 10^5$, $\nu = 0,3$)

Nós	Elems	Eqs	m_{sky}	NElems_{sky}	m_{esp}^{dir}	$\text{NElems}_{esp}^{dir}$	m_{esp}^{ite}	$\text{NElems}_{esp}^{ite}$
654	303	1254	73,2	91798	34,3	43016	11,2	14076
1817	868	3540	129,0	456687	46,7	165205	11,5	40821
3006	1449	5888	169,6	998440	50,7	298805	11,6	68451
5190	2527	10226	213,1	2179279	60,6	619395	11,7	119958
10304	5055	20390	298,2	6079571	72,8	1483760	11,8	240756
16272	8013	32272	376,1	12136114	81,2	2620189	11,9	382305
27765	13722	55376	503,7	27895204	93,0	5148300	11,9	658151

Tabela IV. Atributos das malhas do problema de fratura modificado para matrizes em colunas ascendentes (*sky*) e esparsa (*esp*); métodos direto (*dir*) e iterativo (*ite*)

Malha	Nós	Elems	Eqs	$\text{NElems}_{esp}^{ite}$	m_{esp}^{ite}	$\text{NElems}_{esp}^{dir}$	m_{esp}^{dir}
1	335	921	963	15444	16,0	38511	40,0
2	1130	3922	3279	57804	17,6	265821	81,1
3	3024	11813	8724	162276	18,6	1415535	162,3
4	8633	37976	24606	482364	19,6	8671554	352,4

Tabela V. Atributos das malhas para a viga tridimensional

Para o problema de fratura modificado, a Tabela VI apresenta o número de iterações do método GCGS em cada uma das malhas, utilizando precisões $\xi = 10^{-2}$ e $\xi = 10^{-3}$. Analogamente, para a viga tridimensional, com $\xi = 10^{-3}$, foram necessárias, respectivamente, 222, 369, 490 e 556 iterações para as malhas adotadas. As Figuras 7 a 12 ilustram para estes dois exemplos o número de operações, o espaço de memória (matriz + vetores auxiliares) e o número médio de elementos por linha/coluna em função do número de equações. Finalmente, a Tabela VII contém os coeficientes angulares das retas das Figuras 7, 8, 10 e 11.

Malha	1	2	3	4	5	6	7
NIT ($\xi = 10^{-2}$)	71	122	159	206	293	374	493
NIT ($\xi = 10^{-3}$)	87	155	189	251	358	459	676

Tabela VI. Número de iterações para GCGS nas 7 malhas do problema de fratura modificado

A partir destes resultados, acrescentam-se as seguintes considerações:

- Para casos bidimensionais, embora os coeficientes angulares das retas de custo computacional dos algoritmos tipo gradiente sejam menores que aqueles do método de Gauss em matriz esparsa, este último apresenta melhor desempenho.

A precisão ξ afeta diretamente a performance da técnica GCGS. Em particular para o caso de fratura, o erro relativo (29) foi inferior a 1 % e 0.1 % para $\xi = 10^{-2}$ e $\xi = 10^{-3}$, respectivamente. Para o critério de convergência adotado, $\xi = 10^{-3}$ parece mais adequado.

- A consideração anterior não se aplica para problemas tridimensionais, onde os coeficientes angulares dos algoritmos iterativos baseados em gradiente mantém-se da mesma ordem dos casos bidimensionais. Já no método de Gauss em matriz esparsa, o coeficiente α alcança o valor de 2,57 para o exemplo estudado. Isto se deve ao crescimento pronunciado do número médio de elementos por linha. Observa-se que neste exemplo, o método GCGS passa a ser superior em termos do número de operações a partir de 6639 equações, como pode ser visto na Figura 10.
- O espaço de memória indicado nas Figuras 8 e 11 considera apenas a matriz e os vetores auxiliares. Não se incluem, por exemplo, estruturas de dados aplicadas ao procedimento de Gauss simbólico em matrizes esparsas, as quais passam a ocupar um espaço considerável a medida que se aumenta o número de equações. Logicamente, este último aspecto depende da implementação do procedimento.

Como esperado, a demanda em termos de memória para matriz esparsa é inferior para GCGS em relação ao método de Gauss. Como esta demanda é função não apenas do número de equações, mas também do número médio de elementos por linha/coluna da matriz, pode-se observar como este parâmetro varia ao longo das malhas nas Figuras 9 e 12.

De forma geral, a seleção de um algoritmo direto ou iterativo para a solução do sistema (1), deve se basear não apenas na performance, mas principalmente no espaço de memória necessário. Observa-se que o número de equações, a partir do qual tem-se um mesmo custo computacional para os procedimentos de Gauss e GCGS, é dependente das características do problema.

Assim, um aspecto fundamental a ser considerado é a demanda em termos de memória, a qual cresce de forma mais pronunciada para o algoritmo de Gauss, não apenas para os elementos do sistema de equações, mas também no procedimento simbólico. Para números crescentes de equações, o método de Gauss necessitará preliminarmente de memória em disco (*swap file*) fazendo cair a sua performance.

Desta forma, a técnica GCGS é competitiva em relação ao método de Gauss, principalmente em problemas tridimensionais, onde se torna superior por demandar um menor espaço de memória. Além disso, as taxas de convergência das técnicas iterativas podem ser sensivelmente melhoradas, considerando procedimentos de aceleração multigrid em malhas não-estruturadas e não-aninhadas[2].

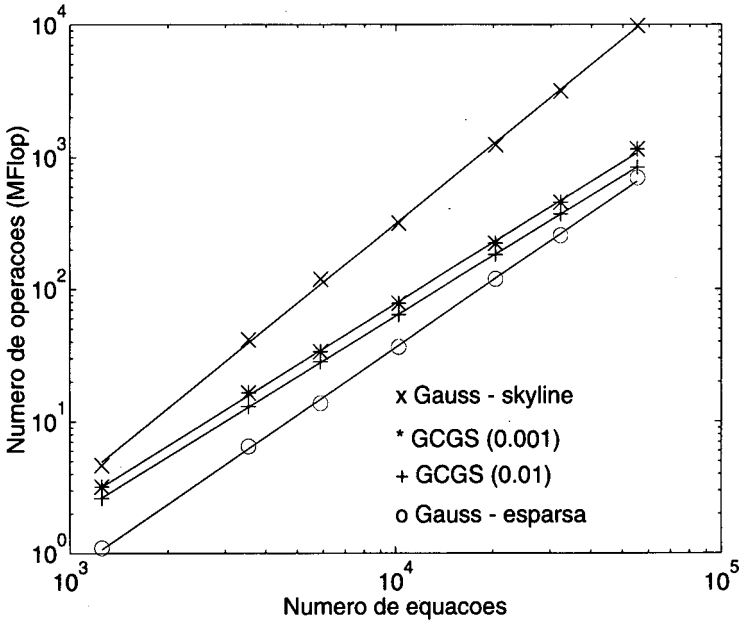

Figura 7. Fratura modificado: número de operações (MFlop) para o método de Gauss esparso e em colunas ascendentes; método iterativo GCGS com precisões $\xi = 10^{-2}$ e $\xi = 10^{-3}$

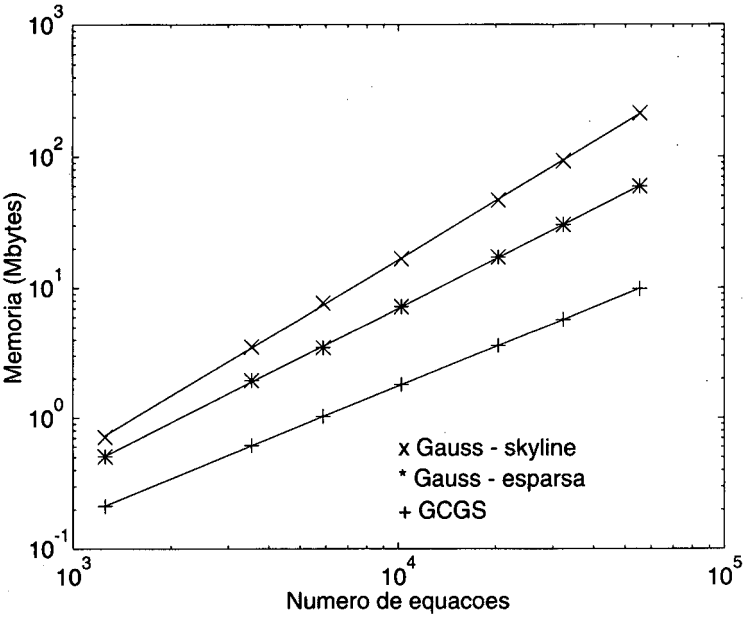

Figura 8. Fratura modificado: espaço de memória (Mbytes) para armazenamento em colunas ascendentes e esparso para o método de Gauss e esparso para o algoritmo GCGS

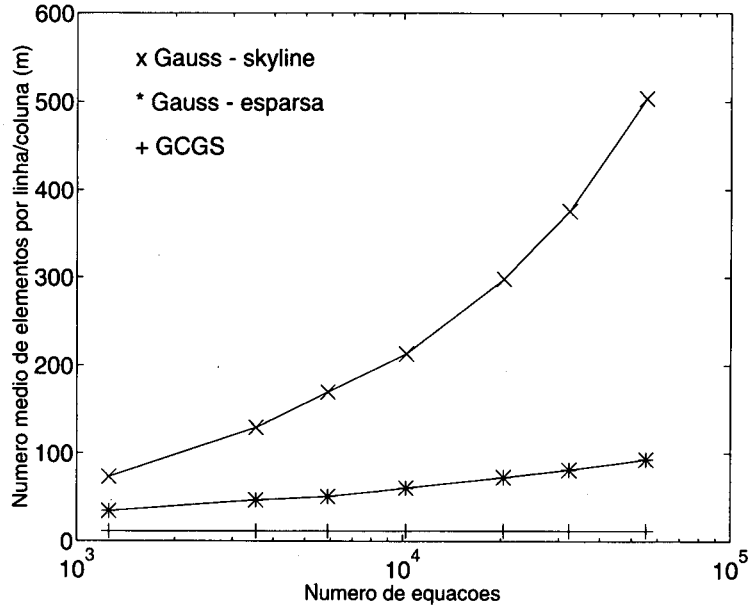

Figura 9. Fratura modificado: número de elementos por linha ou coluna para os métodos de Gauss e GCGS

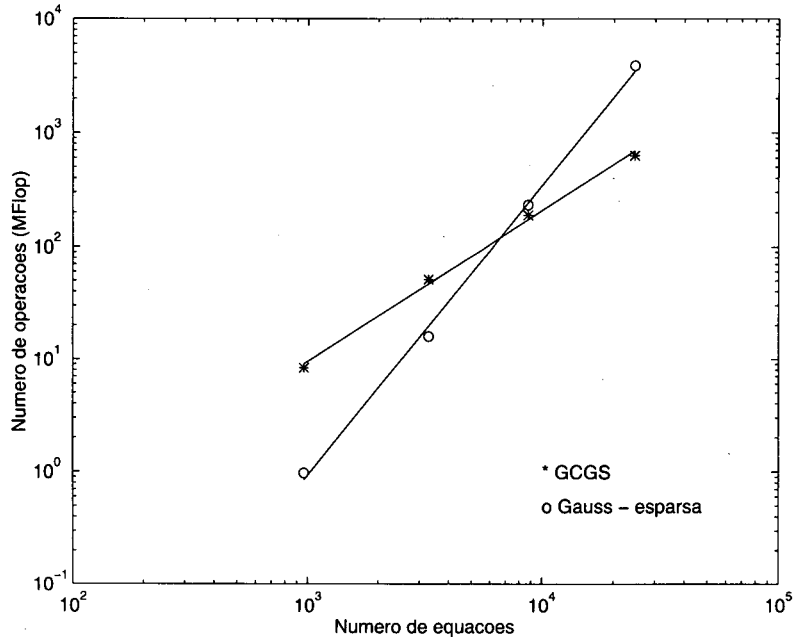

Figura 10. Viga: número de operações (MFlop) para os método s de Gauss e GCGS

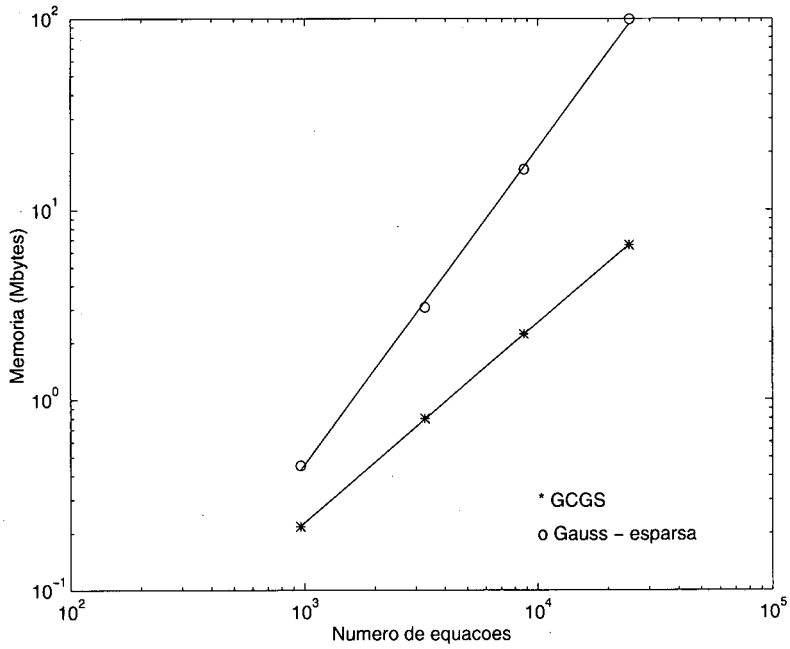

Figura 11. Viga: espaço de memória (Mbytes) para os métodos de Gauss e GCGS

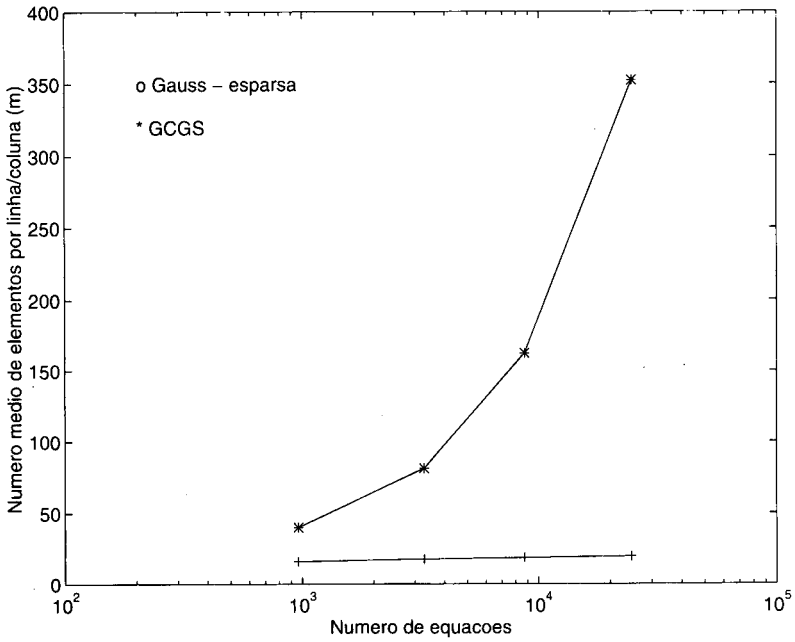

Figura 12. Viga: número de elementos por linha para os métodos de Gauss e GCGS

Fratura modificado		
Método	Operações	Memória
Gauss - skyline	2,00	1,50
Gauss - esparsa	1,70	1,26
GCGS ($\xi = 10^{-2}$)	1,52	1,01
GCGS ($\xi = 10^{-3}$)	1,54	1,01

Viga tridimensional		
Método	Operações	Memória
Gauss - esparsa	2,57	1,67
GCGS ($\xi = 10^{-3}$)	1,34	1,05

Tabela VII. Coeficientes angulares das retas dos gráficos com número de operações e espaço de memória

AGRADECIMENTOS

Os autores agradecem aos seguintes órgãos pelo apoio ao desenvolvimento do trabalho: CNPq, CNPq-RHAE, FAPESP, LNCC, UNICAMP, CENAPAD.

REFERÊNCIAS

1. O. Axelsson e V.A. Barker, "*Finite Element Solution of Boundary Value Problems Theory and Computation*" Academic Press, (1984).
2. M.L. Bittencourt, "Métodos Iterativos e Multigrid Adaptáveis em Malhas Não-estruturadas", Tese de doutorado, DPM/FEM/UNICAMP, (1996).
3. S.F. McCormick (Ed.), "*Multigrid Methods*", SIAM, (1987).
4. R. Barret, "Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods", Edição preliminar (ftp netlib2.cs.utk.edu), (1994).
5. K. Bathe e E.L. Wilson, "*Numerical Methods in Finite Element Analysis*", Prentice Hall, (1976).
6. S.B. Lippman, "*C++ Primer*", Addison-Wesley, (1991).
7. S. Pissanetzky, "*Sparse Matrix Technology*", Academic Press, (1984).
8. L.A. Hageman e D.M. Young, "*Applied Iterative Methods*", Academic Press, (1981).
9. E. Cuthill e J. MCKee, "Reducing the bandwith of sparse symmetric matrices", *Proc. 24th Nat. Conf. Assoc. Comp. Mach, ACM Publ.*, pp. 157 172, (1969).
10. J.W. Liu e A.H. Sherman, "Comparative analysis of the Cuthill-McKee and the reverse Cuthill-McKee ordering algorithms for sparse matrices", *SIAM Journal of Numerical Analysis*, Vol. **13**, pp. 198–213, (1977).
11. A. George e J. W. Liu, "Computer Solution of Large Sparse Positive Definite Systems", Prentice-Hall Series in Computational Mathematics, (1981).

12. V.E. Taylor e B. Nour-Omid, "A study of the factorization fill-in for a parallel implementation of the finite element method", *Int. Journal for Numerical Methods in Engineering*, Vol. **37**, pp. 3809–3823, (1994).
13. G.H. Golub e C.F. Van Loan, "*Matrix Computations*", The Johns Hopkins University Press, (1989).
14. T. Skalický, "LASPack Reference Manual – Version 1.12", Dresden University of Technology, (1995).
15. A. Muller e T.J.R. Hughes, "Precondicionadores elemento-por-elemento y globales. Una Perspectiva", *Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería*, Vol. **2**, 1, pp. 27–41, (1986).
16. A.C.S. Guimarães e R.A. Feijóo, "ACDP: Um Ambiente Computacional para Desenvolvimento de Programas", *Relatório de Pesquisa e Desenvolvimento*, LNCC, No. 027/89, Rio de Janeiro, Brasil.