Estudo de Pré-condicionadores para o Método GMRES usando Estruturas de Dados por Arestas

Seminário de Qualificação ao Doutorado

Lucia Catabriga

Maio, 98
Orientador: Alvaro Luiz Gayoso de Azeredo Coutinho

Índice

Capítulo 1

Introdução

Computadores paralelos tem sido amplamente usados em simulações de grande porte de problemas de mecânica dos fluidos via método dos elementos finitos. Vários exemplos podem ser encontrados em [1]. Tais problemas requerem frequentemente a solução de sistemas não lineares de equações envolvendo milhões de variáveis. Nestes casos, técnicas iterativas de atualização baseadas em espaços de Krylov [2] são usadas para resolver sistemas de equações oriundos das discretizações de elementos finitos. As principais características para o sucesso das técnicas baseadas nos espaços de Krylov são implementações eficientes do produto matriz-vetor e a escolha de bons pré-condicionadores, levando-se em consideração o ambiente computacional paralelo.

Para o método dos elementos finitos, implementações globais do produto matriz-vetor são facilmente paralelizáveis em diferentes arquiteturas computacionais, sendo o produto realizado a nível de cada elemento e em seguida executado o agrupamento global. Estas implementações usam muita memória, necessitando de mais operações que o produto executado diretamente com a matriz global, pois a matriz dos elementos possui muitos coeficientes globais não-nulos sobrepostos. Uma alternativa para diminuir o tamanho da memória utilizada é empregar implementações livres de matrizes [3]. Isto acarreta a existência de muitas operações repetitivas ocorrendo acréscimos no tempo de processamento o que aumenta o custo computacional desta alternativa. Recentemente têm sido propostos vários pré-condicionadores paralelos elemento-por-elemento, usando a mesma estrutura de dados do produto matriz-vetor [4] [5].

Um importante passo alcançado foi a idéia de empregar grupos de elementos para definir pré-condicionadores mais eficientes, tal como o pré-condicionador CEBE (*Cluster Element-By-Element*) [6] e o pré-condicionador CC (*Cluster Companion*) [7]. Recentemente Kalro e Tezduyar [8] desenvolveram a utilização de técnicas de matrizes esparsas em conjunto com pré-condicionadores globalmente definidos, utilizando pré-condicionadores do tipo CEBE com fatoração incompleta (ILU0). Estes apresentam resultados satisfatórios, obtendo uma notável economia de memória quando comparados com a técnica convencional elemento-por-elemento.

Por outro lado, inspirado no método dos volumes finitos, estruturas de dados baseadas nas arestas tem sido empregadas para soluções explícitas de elementos finitos para fluidos compressíveis em malhas não estruturadas composta de triângulos e tetraedros [9] [10]. As vantagens dos esquemas baseados nas arestas em relação aos esquemas convencionais são uma considerável redução nas operações de endereçamento indireto e na memória utilizada [11]. Permitem ainda implementações diretas dos esquemas de estabilização baseado nas linhas de corrente (*upwind*), no contexto do método dos elementos finitos. Para reduzir o número de endereçamentos indiretos baseados nas arestas, Lohner, em [12], propõe novas estruturas de dados, conhecidas por: *stars*, *superedges* e *chains*. Tais estruturas podem ser consideradas como agrupamentos da estrutura de dados aresta simples. Dentre elas, as super-arestas (*superedges*) são as mais apropriadas para implementação em computadores paralelos. Porém, em recente implementação paralela de solução da equação de Euler bidimensional, foi observado uma melhora insignificante quando usado super-arestas comparado com arestas simples [13].

A estrutura de dados baseada nas arestas também pode ser vista como uma representação do grafo nodal da malha composta por triângulos e tetraedros. Pode-se dizer então, que a representação por arestas é uma estrutura de dados alternativa para calcular o produto matriz-vetor global, necessário nas técnicas iterativas baseadas nos espaços de Krylov. O desempenho dos primeiros experimentos efetuados por Venkatakrishnam e Mavriplis [14] na solução implícita por volumes finitos baseados nas arestas da equação de Navier-Stokes são promissores. Também foi verificado que, em soluções paralelas por elementos finitos de vários problemas tridimensionais de escoamento potencial para configurações industriais, as estruturas de dados por aresta são mais eficientes que a estrutura de dados baseada nos elementos [15]. Além disso, observou-se que o esquema por super-arestas foi claramente melhor.

O objetivo deste trabalho é desenvolver pré-condicionadores usando estruturas de dados baseada nas arestas para soluções implícitas do método dos elementos finitos. Vários pré-condicionadores são definidos a partir de decomposições das matrizes das estruturas das arestas e das super-arestas. Os pré-condicionadores baseados nas arestas são comparados com pré-condicionadores baseados nos elementos, usando decomposições similares para as matrizes dos elementos.

Neste texto, os pré-condicionadores desenvolvidos são aplicados em problemas de advecção-difusão bidimensionais em malhas estruturadas e não estruturadas. O método iterativo escolhido para solucionar os sistemas lineares resultantes foi o Método do Resíduo Mínimo Generalizado (GMRES), pois este possui o melhor comportamento dentre as técnicas iterativas baseadas nos espaços de Krylov para solucionar sistemas não-simétricos de grande porte. O código computacional foi desenvolvido em linguagem FORTRAN77 a partir da implementação já existente, desenvolvida por Martins em [20], para problemas de escoamento potencial. As implementações visam tirar o maior proveito possível das facilidades para processamento vetorial e paralelo em ambiente de arquitetura PVP (*Parallel Vector Processors*) com memória compartilhada, sendo usado o Cray J90.

O capítulo 2 traz a formulação por elementos finitos da equação de advecção e difusão em duas dimensões usando estruturas de dados por arestas. As contribuições de Galerkin e as correções de Petrov-Galerkin são obtidas para cada parcela da equação em questão. É mostrado um resumo de todas as contribuições armazenadas em matrizes de arestas de ordem 2. No final do capítulo 2 é feito um estudo comparativo de esquemas de armazenamento para matrizes esparsas. No capítulo seguinte, são definidos novos pré-condicionadores baseados nas arestas para o método GMRES, enfatizando o uso das estruturas de dados baseadas nas arestas. São definidas várias decomposições para as matrizes das arestas e super-arestas. No capítulo 4 é analizado o desempenho dos vários pré-condionadores desenvolvidos. Testes são realizados em alguns problemas de advecção e difusão bidimensionais, sendo o tempo de CPU e a memória utilizada os principais parâmetros de comparação. No último capítulo são descritas as conclusões observadas.

Capítulo 2

Formulação de problemas não simétricos usando estrutura de dados por arestas

Neste capítulo é apresentada a formulação por elementos finitos da equação de advecção e difusão em duas dimensões usando a formulação SUPG (*Streamline Upwind Petrov Galerkin*) e estrutura de dados por arestas.

2.1 Equação de Advecção - Difusão em 2D

Seja Ω uma região limitada de $\Re^2$, com contorno Γ suave por partes, sendo $\mathbf{x} = \{x\ y\}^T = \{x_i\}$, um ponto de $\overline{\Omega} = \Omega \cup \Gamma$ e $\mathbf{n} = \{n_x\ n_y\}^T = \{n_i\}$ o vetor normal externo de Γ. Γ_g e Γ_h são subconjuntos de Γ satisfazendo às condições: $\Gamma_g \cup \Gamma_h = \Gamma$ e $\Gamma_g \cap \Gamma_h = \emptyset$.

Considere uma discretização de Ω em subdomínios Ω^e, $e = 1, \ldots, nel$, sendo nel o número de elementos. Denota-se por Γ^e o contorno de Ω^e e $\bar{\Omega}^e = \Omega^e \cup \Gamma^e$, assumindo-se que $\bigcup_e \bar{\Omega}^e = \Omega$ e $\bigcap_e \Omega^e = \emptyset$. Finalmente, considerando as leis de conservação resultantes do balanço da quantidade arbitrária u com seus fluxos σ_i^a e

σ_i^d no interior de um volume de controle, tem-se:

$$u_{,t} + \sigma^a_{i,i} + \sigma^d_{i,i} = f \tag{2.1}$$

onde $\sigma_i^a = \beta_i u$ é o fluxo advectivo; $\sigma_i^d = -k_{ij}u_{,j}$ é o fluxo difusivo; f é o termo de fonte; β_i é o campo de velocidade do fluido e k_{ij} é a difusividade volumétrica, dada por:

$$\mathbf{k} = \begin{bmatrix} k_x & 0 \\ 0 & k_y \end{bmatrix} \tag{2.2}$$

Em coordenadas cartesianas a equação (2.1) apresenta a forma:

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x_i}(\beta_i u) - \frac{\partial}{\partial x_i}(k_{ij}\frac{\partial u}{\partial x_j}) = f \tag{2.3}$$

Considerando o campo de velocidade $\boldsymbol{\beta} = \{\beta_x \ \beta_y\}^T$ com divergência nula, ou seja, $div\boldsymbol{\beta} = 0$, tem-se:

$$\frac{\partial u}{\partial t} + \beta_x\frac{\partial u}{\partial x} + \beta_y\frac{\partial u}{\partial y} - \frac{\partial}{\partial x}(k_x\frac{\partial u}{\partial x}) - \frac{\partial}{\partial y}(k_y\frac{\partial u}{\partial y}) = f \tag{2.4}$$

ou

$$u_{,t} + \boldsymbol{\beta}.\nabla u - \nabla.(\mathbf{k}\nabla u) = f \tag{2.5}$$

Resumindo, deseja-se resolver um problema de valor inicial que consiste em encontrar $u(\mathbf{x}, t)$ satisfazendo (2.1) em Ω, tal que:

$$\begin{aligned} & u = g \quad \text{em} \quad \Gamma_g, \\ & -\sigma_i^d n_i = -k_{ij}\tfrac{\partial u}{\partial x_j} n_j = -\mathbf{k}\tfrac{\partial u}{\partial \mathbf{n}} = h \quad \text{em} \quad \Gamma_h, \\ & u(\mathbf{x}, 0) = u_0 \end{aligned} \tag{2.6}$$

sendo g e h funções de $\mathbf{x}$ e t, e u_0 função apenas de $\mathbf{x}$.

Equações deste tipo tem sido amplamente estudadas ao longo dos últimos 20 anos. A formulação padrão de Galerkin não gera bons resultados devido às imprecisões e instabilidades geradas pelo uso de funções-teste usuais. A formulação Petrov-Galerkin [16] [17] utilizando funções testes especiais tem apresentado resultados excelentes para o tratamento de problemas com advecção dominante.

2.2 Formulação SUPG

No método tradicional de Galerkin as funções-teste consideradas são contínuas ao longo dos contornos entre os elementos. A formulação SUPG porém, define uma função-teste descontínua na forma:

$$\tilde{w} = w + p \tag{2.7}$$

onde w é a função teste contínua e p é a contribuição descontínua no contorno (*streamline upwind*), sendo w e p suaves no interior dos elementos. Estudos realizados por Brooks e Hughes em [16] demonstram que:

$$\tilde{w} = w + \frac{\tau}{||\boldsymbol{\beta}||}\boldsymbol{\beta}\nabla w \tag{2.8}$$

onde $\boldsymbol{\beta}$ é o campo de velocidade do fluido definido em (2.1); $\tau = \frac{\alpha h}{2}$, sendo h o parâmetro de malha e α é denominado parâmetro de *upwind* que depende do número de Peclet do elemento. Codina, em [18], considera uma relação assintótica para α, ou seja,

$$\alpha = min(\frac{Pe}{3}, 1) \tag{2.9}$$

onde

$$Pe = \frac{||\boldsymbol{\beta}||h}{\kappa} \tag{2.10}$$

sendo

$$\kappa = \frac{\boldsymbol{\beta}^T}{||\boldsymbol{\beta}||}\mathbf{k}\frac{\boldsymbol{\beta}}{||\boldsymbol{\beta}||} \tag{2.11}$$

$\boldsymbol{\beta}$ e $\mathbf{k}$ são dados respectivamente por (2.1) e (2.2). Portanto, a formulação fraca para a discretização SUPG de (2.5) é:

$$\sum_e \int_{\Omega_e} \tilde{w}(u_{,t} + \boldsymbol{\beta}\nabla u - \nabla(\mathbf{k}\nabla u))d\Omega = \sum_e \int_{\Omega_e} \tilde{w} f d\Omega \tag{2.12}$$

A equação (2.12) pode ser representada por três contribuições distintas:

$$\boxed{\text{Galerkin (G)}} + \boxed{\text{Petrov-Galerkin (PG)}} = \boxed{\text{Fonte (F)}} \tag{2.13}$$

$$\underbrace{\int_{\Omega} w(u_{,t} + \boldsymbol{\beta}\nabla u - \nabla(\mathbf{k}\nabla u))d\Omega}_{G} +$$
$$\underbrace{\sum_e \int_{\Omega_e} p(u_{,t} + \boldsymbol{\beta}\nabla u - \nabla(\mathbf{k}\nabla u))d\Omega}_{PG} = \tag{2.14}$$
$$\underbrace{\int_{\Omega}(w+p)fd\Omega}_{F}$$

Aplicando integração por partes e usando (2.12) para cada contribuição, obtêm-se:

$$\text{(G)}: \int_{\Omega} wu_{,t}\, d\Omega + \int_{\Omega} w\boldsymbol{\beta}\nabla u - \int_{\Omega} \nabla w.\mathbf{k}\nabla u d\Omega \tag{2.15}$$

$$\text{(PG)}: \sum_e \frac{\tau}{\|\boldsymbol{\beta}\|}(\int_{\Omega_e} \boldsymbol{\beta}\nabla w u_{,t}\, d\Omega + \int_{\Omega_e} \boldsymbol{\beta}\nabla w\boldsymbol{\beta}\nabla u d\Omega + \int_{\Omega_e} \boldsymbol{\beta}\nabla w\nabla(\mathbf{k}\nabla u)d\Omega) \tag{2.16}$$

$$\text{(F)}: \int_{\Omega}(w + \frac{\tau}{\|\boldsymbol{\beta}\|}\boldsymbol{\beta}\nabla w)fd\Omega \tag{2.17}$$

onde cada parcela recebe denominações relativas à sua representação física, ou seja,

$$\begin{array}{ll} \int_{\Omega} wu_{,t}\, d\Omega \text{ e } \sum_e \frac{\tau}{\|\boldsymbol{\beta}\|}\int_{\Omega_e} \boldsymbol{\beta}\nabla w u_{,t}\, d\Omega & \text{(termos de "massa")} \\ \int_{\Omega} w\boldsymbol{\beta}\nabla u \text{ e } \sum_e \frac{\tau}{\|\boldsymbol{\beta}\|}\int_{\Omega_e} \boldsymbol{\beta}\nabla w\boldsymbol{\beta}\nabla u d\Omega & \text{(termos de advecção)} \\ \int_{\Omega} \nabla w.\mathbf{k}\nabla u d\Omega \text{ e } \sum_e \frac{\tau}{\|\boldsymbol{\beta}\|}\int_{\Omega_e} \boldsymbol{\beta}\nabla w\nabla(\mathbf{k}\nabla u)d\Omega & \text{(termos de difusão)} \end{array} \tag{2.18}$$

A aproximação por elementos finitos é feita considerando $u(\mathbf{x},t) = u(\mathbf{x})u(t)$ e usando interpolações do tipo:

$$u(\mathbf{x}) \cong \sum_{i=1}^{nnos} \Phi_i(\mathbf{x})u_i \tag{2.19}$$

sendo $nnos$ o número de nós da malha. Φ_i é a função de interpolação usual [19] relativa ao nó i. Decorrente dessa aproximação, pode-se representar a equação (2.12) na forma aproximada:

$$\mathbf{MU}_{,t} + \mathbf{KU} = \mathbf{F} \tag{2.20}$$

onde $\mathbf{M}$ é denominada Matriz de Massa Consistente, $\mathbf{K}$ Matriz de Rigidez, $\mathbf{F}$ Termo de Fonte e $\mathbf{U} = \{u_1\ u_2\ \dots\ u_{nnos}\}^T$. Note que cada matriz apresenta as parcelas de Galerkin e Petrov-Galerkin. Essas matrizes e o vetor de termos fonte podem ser obtidos a partir das contribuições individuais dos elementos. Assim, têm-se:

$$\begin{aligned} &\mathbf{M} = \textstyle\sum_e (\mathbf{M}_g^e + \mathbf{M}_{pg}^e) \\ &\mathbf{K} = \textstyle\sum_e (\mathbf{K}_g^e + \mathbf{K}_{pg}^e) \\ &\mathbf{F} = \textstyle\sum_e (\mathbf{F}_g^e + \mathbf{F}_{pg}^e) \end{aligned} \tag{2.21}$$

No decorrer desse estudo serão considerados apenas elementos triangulares lineares. Neste caso, o parâmetro de malha h será avaliado simplesmente como $\sqrt{A}$, onde A é a área do elemento. Assim, em um elemento genérico e, a aproximação (2.19) pode ser representada por:

$$u^e(\mathbf{x}) \cong \sum_{i=1}^{3} N_i(\mathbf{x}) u_i \tag{2.22}$$

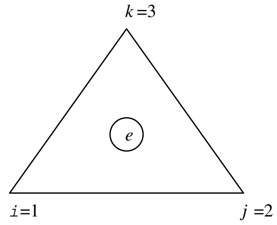


Figura 2.1: Elemento triangular linear padrão

onde N_1, N_2 e N_3 são respectivamente as funções de interpolação usuais [19] relativas aos nós 1, 2 e 3 do elemento e conforme a Figura 2.1. Assim, pode-se dizer que:

$$u^e(\mathbf{x}) \cong \mathbf{N}(\mathbf{x}) \mathbf{U}^e \tag{2.23}$$

onde $\mathbf{N}(\mathbf{x}) = \{N_1(\mathbf{x}), N_2(\mathbf{x}), N_3(\mathbf{x})\}$ e $\mathbf{U}^e = \{u_1, u_2, u_3\}^T$, relativos respectivamente aos nós 1, 2 e 3 do elemento e.

2.3 Estrutura de Dados por Arestas

A partir das matrizes dos elementos oriundas da discretização de Petrov-Galerkin (2.15), (2.16) e (2.17) define-se um desmembramento dos coeficientes gerando as contribuições de cada aresta, que pode ser representado por:

$$\underbrace{\begin{bmatrix} \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \end{bmatrix}}_{\text{elemento } e} = \underbrace{\begin{bmatrix} \times & \times & 0 \\ \times & \times & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{\text{aresta } ij} = \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & \times & \times \\ 0 & \times & \times \end{bmatrix}}_{\text{aresta } jk} = \underbrace{\begin{bmatrix} \times & 0 & \times \\ 0 & 0 & 0 \\ \times & 0 & \times \end{bmatrix}}_{\text{aresta } ki} \tag{2.24}$$

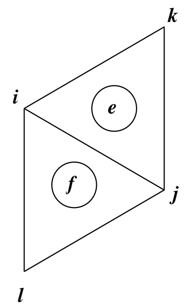


Figura 2.2: Elementos adjacentes a aresta s

Porém, pode-se dizer que sobre a aresta ij incidem informações dos elementos adjacentes a ela, observe a Figura 2.2. Assim, todas as contribuições referentes à aresta s (formada pelos nós i e j), estão presentes nos elementos e e f:

$$\underbrace{\begin{bmatrix} \circ & \circ \\ \circ & \circ \end{bmatrix}}_{\text{aresta } s} = \underbrace{\begin{bmatrix} \times & \times \\ \times & \times \end{bmatrix}}_{\text{elemento } e} + \underbrace{\begin{bmatrix} \times & \times \\ \times & \times \end{bmatrix}}_{\text{elemento } f} \tag{2.25}$$

A estrutura de dados convencional para uma malha triangular linear, define a relação entre nós e elementos onde cada elemento e está associado aos nós i, j e k da malha, de acordo com a Figura 2.1. Na estrutura de dados baseada nas arestas, cada aresta s está associada aos nós i, j, k e l, conforme a Figura 2.2. A mudança de estrutura de dados é feita considerando que os dados referente à malha de elementos já exista de forma convencional. As informações geométricas e topológicas da malha são manipuladas de forma a gerar a nova estrutura baseada nas arestas dos elementos. Para tal, é usado uma técnica usual de busca em tabelas conhecida por *Hash Table*, descrita com detalhes para esta aplicação em [20]. Assim, as matrizes definidas em (2.21), podem ser redefinidas por:

$$
\begin{aligned}
&\mathbf{M} = \textstyle\sum_s(\mathbf{M}_g^s + \mathbf{M}_{pg}^s) \\
&\mathbf{K} = \textstyle\sum_s(\mathbf{K}_g^s + \mathbf{K}_{pg}^s) \\
&\mathbf{F} = \textstyle\sum_s(\mathbf{F}_g^s + \mathbf{F}_{pg}^s)
\end{aligned}
\tag{2.26}
$$

2.4 Matrizes das arestas

A partir das parcelas de Galerkin e de Petrov-Galerkin das matrizes dos elementos descritas em (2.21) é possivel obter parcelas similares para as matrizes das arestas, que aparecem em (2.24). Em todos os casos são consideradas as funções de interpolação descritas em (2.22). Todas as integrais podem ser resolvidas sem uso de aproximações numéricas devido à simplicidade das funções de interpolação escolhidas.

2.4.1 Matriz de Massa Consistente de Galerkin

Aplicando aproximações de elementos finitos a cada elemento na parcela da massa consistente de Galerkin, descrita em (2.15), obtem-se a matriz:

$$\mathbf{M}_g^e = \int_{\Omega_e} \mathbf{N}^T \mathbf{N} d\Omega \tag{2.27}$$

devido as particularidades, já mencionadas de $\mathbf{N}$, tem-se:

$$\mathbf{M}_g^e = \frac{A^e t}{12} \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \tag{2.28}$$

onde A^e é a área do elemento e e t a espessura. Assim, a contribuição do elemento e na matriz da aresta s, conforme a Figura 2.2, é dada por:

$$\mathbf{M}_g^s = \frac{A^e t}{12} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \tag{2.29}$$

A matriz total da aresta s é dada por uma soma das contribuições relativas aos elementos e e f:

$$\mathbf{M}_g^s = \frac{(A^e + A^f)t}{12} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \tag{2.30}$$

2.4.2 Matriz de Difusão de Galerkin

Devido às aproximações impostas pelo método de elementos finitos, a matriz referente a parcela de difusão de Galerkin (2.15), para um dado elemento, pode ser representada por:

$$\mathbf{T}_g^e = \int_{\Omega_e} \mathbf{B}^T \mathbf{k} \mathbf{B} d\Omega \tag{2.31}$$

onde:

$$\mathbf{B} = \frac{1}{2A^e} \begin{bmatrix} \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial x} & \frac{\partial N_3}{\partial x} \\ \frac{\partial N_1}{\partial y} & \frac{\partial N_2}{\partial y} & \frac{\partial N_3}{\partial y} \end{bmatrix} = \frac{1}{2A^e} \begin{bmatrix} y_{23} & y_{31} & y_{12} \\ x_{32} & x_{13} & x_{21} \end{bmatrix} = \frac{1}{2A^e} \begin{bmatrix} b_1 & b_2 & b_3 \\ a_1 & a_2 & a_3 \end{bmatrix}$$

O integrando $\mathbf{B}^T\mathbf{k}\mathbf{B}$ é constante, portanto:

$$\mathbf{T}_g^e = \frac{1}{4A^e} \begin{bmatrix} k_x b_1 b_1 + k_y a_1 a_1 & k_x b_1 b_2 + k_y a_1 a_2 & k_x b_1 b_3 + k_y a_1 a_3 \\ k_x b_2 b_1 + k_y a_2 a_1 & k_x b_2 b_2 + k_y a_2 a_2 & k_x b_2 b_3 + k_y a_2 a_3 \\ k_x b_3 b_1 + k_y a_3 a_1 & k_x b_3 b_2 + k_y a_3 a_2 & k_x b_3 b_3 + k_y a_3 a_3 \end{bmatrix} \tag{2.32}$$

Na matriz descrita em (2.32) observa-se a propriedade de deficiência de posto, ou seja, observa-se que:

$$\begin{aligned} k_x b_1 b_1 + k_y a_1 a_1 &= -(k_x b_1 b_2 + k_y a_1 a_2) - (k_x b_1 b_3 + k_y a_1 a_3) \\ k_x b_2 b_2 + k_y a_2 a_2 &= -(k_x b_2 b_1 + k_y a_2 a_1) - (k_x b_2 b_3 + k_y a_2 a_3) \\ k_x b_3 b_3 + k_y a_3 a_3 &= -(k_x b_3 b_1 + k_y a_3 a_1) - (k_x b_3 b_2 + k_y a_3 a_2) \end{aligned} \tag{2.33}$$

define-se então:

$$t_{ij} = -\frac{1}{4A^e}(k_x b_i b_j + k_y a_i a_j), \quad i, j = 1, 2, 3 \;\; \text{e} \;\; i \neq j \tag{2.34}$$

Observando que $t_{ij} = t_{ji}$, é possível escrever a matriz de difusão de Galerkin com apenas três coeficientes distintos:

$$\mathbf{T}_g^e = \begin{bmatrix} t_{12}+t_{13} & -t_{12} & -t_{13} \\ -t_{12} & t_{12}+t_{23} & -t_{23} \\ -t_{13} & -t_{23} & t_{13}+t_{23} \end{bmatrix} \tag{2.35}$$

Portanto a matriz da aresta s pode ser representada por:

$$\mathbf{T}_g^s = \begin{bmatrix} t_{ij}^e + t_{ij}^f & -(t_{ij}^e + t_{ij}^f) \\ -(t_{ij}^e + t_{ij}^f) & t_{ij}^e + t_{ij}^f \end{bmatrix} \tag{2.36}$$

com t_{ij} definido em (2.34).

2.4.3 Matriz de Advecção de Galerkin

As aproximações do método dos elementos finitos aplicadas a parcela de advecção, em (2.15), define a matriz de advecção de Galerkin, em cada elemento, por:

$$\mathbf{C}_g^e = \int_{\Omega_e} \mathbf{N}^T \boldsymbol{\beta}^T \mathbf{B} d\Omega \tag{2.37}$$

considerando o campo de velocidade constante no elemento obtem-se:

$$\mathbf{C}_g^e = \frac{1}{6} \begin{bmatrix} \beta_x b_1 + \beta_y a_1 & \beta_x b_2 + \beta_y a_2 & \beta_x b_3 + \beta_y a_3 \\ \beta_x b_1 + \beta_y a_1 & \beta_x b_2 + \beta_y a_2 & \beta_x b_3 + \beta_y a_3 \\ \beta_x b_1 + \beta_y a_1 & \beta_x b_2 + \beta_y a_2 & \beta_x b_3 + \beta_y a_3 \end{bmatrix} \tag{2.38}$$

O fluxo de fluido ao longo do elemento e pode ser representado por

$$\boldsymbol{\sigma}^a = \mathbf{C}_g^e \mathbf{U}^e \tag{2.39}$$

assim

$$\begin{aligned} \sigma_1^a &= \tfrac{1}{6}(\beta_x b_1 + \beta_y a_1)u_1 + \tfrac{1}{6}(\beta_x b_2 + \beta_y a_2)u_2 + \tfrac{1}{6}(\beta_x b_3 + \beta_y a_3)u_3 \\ \sigma_2^a &= \sigma_1^a \\ \sigma_3^a &= \sigma_1^a \end{aligned} \tag{2.40}$$

sabendo que:

$$-(\beta_x b_1 + \beta_y a_1) = (\beta_x b_2 + \beta_y a_2) + (\beta_x b_3 + \beta_y a_3) \tag{2.41}$$

e fazendo

$$\begin{aligned} c_{12} &= -\tfrac{1}{6}(\beta_x b_2 + \beta_y a_2) \\ c_{13} &= -\tfrac{1}{6}(\beta_x b_3 + \beta_y a_3) \end{aligned} \tag{2.42}$$

tem-se:

$$\sigma_1^a = c_{12}(u_1 - u_2) + c_{13}(u_1 - u_3) \tag{2.43}$$

da mesma forma tem-se que

$$-(\beta_x b_2 + \beta_y a_2) = (\beta_x b_1 + \beta_y a_1) + (\beta_x b_3 + \beta_y a_3) \tag{2.44}$$

e fazendo

$$\begin{aligned} c_{21} &= -\tfrac{1}{6}(\beta_x b_1 + \beta_y a_1) \\ c_{23} &= -\tfrac{1}{6}(\beta_x b_3 + \beta_y a_3) \end{aligned} \tag{2.45}$$

tem-se:

$$\sigma_2^a = c_{21}(u_2 - u_1) + c_{23}(u_2 - u_3) \tag{2.46}$$

Finalmente

$$-(\beta_x b_3 + \beta_y a_3) = (\beta_x b_1 + \beta_y a_1) + (\beta_x b_2 + \beta_y a_2) \tag{2.47}$$

$$\begin{aligned} c_{31} &= -\tfrac{1}{6}(\beta_x b_1 + \beta_y a_1) \\ c_{32} &= -\tfrac{1}{6}(\beta_x b_2 + \beta_y a_2) \end{aligned} \tag{2.48}$$

$$\sigma_3^a = c_{31}(u_3 - u_1) + c_{32}(u_3 - u_2) \tag{2.49}$$

portanto a equação (2.39) pode ser escrita por:

$$\begin{bmatrix} \sigma_1^a \\ \\ \sigma_2^a \\ \\ \sigma_3^a \end{bmatrix} = \begin{bmatrix} c_{12}+c_{13} & -c_{12} & -c_{13} \\ \\ -c_{21} & c_{21}+c_{23} & -c_{23} \\ \\ -c_{31} & -c_{32} & c_{31}+c_{32} \end{bmatrix} \begin{bmatrix} u_1 \\ \\ u_2 \\ \\ u_3 \end{bmatrix} \tag{2.50}$$

Observa-se a não simetria de $\mathbf{C}_g^e$, porém devido às propriedades observadas no fluxo resultante da equação pode-se representar (2.47) usando apenas 6 coeficientes, dados por:

$$\begin{array}{l} c_{ij} = -\frac{1}{6}(\beta_x b_j + \beta_y a_j) \\ c_{ji} = -\frac{1}{6}(\beta_x b_i + \beta_y a_i) \end{array} \quad , \quad i,j = 1,2,3 \quad i \neq j \tag{2.51}$$

Assim a matriz da aresta s pode ser representada por:

$$\mathbf{C}_g^s = \begin{bmatrix} c_{ij}^e + c_{ij}^f & -(c_{ij}^e + c_{ij}^f) \\ \\ -(c_{ji}^e + c_{ji}^f) & c_{ji}^e + c_{ji}^f \end{bmatrix} \tag{2.52}$$

2.4.4 Correção SUPG na matriz de Difusão

A partir da parcela de difusão existente na expressão (2.16) obtem-se para cada elemento:

$$\mathbf{T}_{pg}^e = \frac{\tau}{||\boldsymbol{\beta}||} \int_{\Omega_e} \mathbf{B}^T \boldsymbol{\beta} \nabla(\mathbf{k}\nabla\mathbf{B}) d\Omega \tag{2.53}$$

O integrando da expressão (2.53) depende das derivadas de segunda ordem das funções de interpolação. Neste caso consideram-se funções lineares, portanto a contribuição SUPG do termo de difusão será nula, ou seja,

$$\mathbf{T}_{pg}^e = 0 \tag{2.54}$$

2.4.5 Correção SUPG na matriz de Advecção

A matriz de correção SUPG da parcela de advecção presente (2.16), para cada elemento, obtida após aproximações de elementos finitos, é dada por:

$$\mathbf{C}^e_{pg} = \frac{\tau}{||\boldsymbol{\beta}||} \int_{\Omega_e} \mathbf{B}^T \boldsymbol{\beta}\boldsymbol{\beta}^T \mathbf{B} d\Omega \tag{2.55}$$

Considerando a velocidade constante no elemento e fazendo:

$$\begin{aligned} \beta_{xx} &= \beta_x \beta_x \\ \beta_{xy} &= \beta_x \beta_y \\ \beta_{yy} &= \beta_y \beta_y \end{aligned} \tag{2.56}$$

chega-se a:

$$\begin{aligned} (\mathbf{C}^e_{pg})_{11} &= \tfrac{\tau}{4A^e||\boldsymbol{\beta}||}(\beta_{xx} b_1 b_1 + 2\beta_{xy} b_1 a_1 + \beta_{yy} a_1 a_1) \\ (\mathbf{C}^e_{pg})_{12} &= \tfrac{\tau}{4A^e||\boldsymbol{\beta}||}(\beta_{xx} b_1 b_2 + \beta_{xy} b_1 a_2 + \beta_{xy} a_1 b_2 + \beta_{yy} a_1 a_2) \\ (\mathbf{C}^e_{pg})_{13} &= \tfrac{\tau}{4A^e||\boldsymbol{\beta}||}(\beta_{xx} b_1 b_3 + \beta_{xy} b_1 a_3 + \beta_{xy} a_1 b_3 + \beta_{yy} a_1 a_3) \end{aligned} \tag{2.57}$$

Também neste caso, tem-se a propriedade $(\mathbf{C}^e_{pg})_{11} = -((\mathbf{C}^e_{pg})_{12} + (\mathbf{C}^e_{pg})_{13})$, ocorrendo de forma similar para as outras linhas da matriz. Assim, obtem-se a matriz de correção de advecção de forma semelhante à matriz de difusão de Galerkin:

$$\mathbf{C}^e_{pg} = \begin{bmatrix} t^{pg}_{12} + t^{pg}_{13} & -t^{pg}_{12} & -t^{pg}_{13} \\ -t^{pg}_{12} & t^{pg}_{12} + t^{pg}_{23} & -t^{pg}_{23} \\ -t^{pg}_{13} & -t^{pg}_{23} & t^{pg}_{13} + t^{pg}_{23} \end{bmatrix} \tag{2.58}$$

onde cada termo é definido por:

$$t^{pg}_{ij} = \frac{\tau}{4A^e||\boldsymbol{\beta}||}(\beta_{xx} b_i b_j + \beta_{xy} b_i a_j + \beta_{xy} a_i b_j + \beta_{yy} a_i a_j) \quad i,j = 1,2,3 \;\text{ e }\; i \neq j \tag{2.59}$$

A correção SUPG da matriz de advecção é simétrica. Portanto a matriz da aresta s, relativa a Figura 2.2, também será simétrica, sendo representada por apenas um coeficiente para cada elemento adjacente:

$$\mathbf{C}^s_{pg} = \begin{bmatrix} ((t^{pg}_{ij})^e + (t^{pg}_{ij})^f) & -((t^{pg}_{ij})^e + (t^{pg}_{ij})^f) \\ -((t^{pg}_{ij})^e + (t^{pg}_{ij})^f) & ((t^{pg}_{ij})^e + (t^{pg}_{ij})^f) \end{bmatrix} \tag{2.60}$$

2.4.6 Correção SUPG na matriz de Massa

Após aplicar aproximações do método de elementos finitos à parcela de massa presente na expressão (2.16), obtem-se para cada elemento:

$$\mathbf{M}^e_{pg} = \frac{\tau}{||\boldsymbol{\beta}||}\int_{\Omega_e} \mathbf{B}^T \boldsymbol{\beta} \mathbf{N} d\Omega \tag{2.61}$$

Assim, considerando constante o campo de velocidades no elemento tem-se:

$$\mathbf{M}^e_{pg} = \frac{\tau}{6||\boldsymbol{\beta}||}\begin{bmatrix} \beta_x b_1 + \beta_y a_1 & \beta_x b_1 + \beta_y a_1 & \beta_x b_1 + \beta_y a_1 \\ \beta_x b_2 + \beta_y a_2 & \beta_x b_2 + \beta_y a_2 & \beta_x b_2 + \beta_y a_2 \\ \beta_x b_3 + \beta_y a_3 & \beta_x b_3 + \beta_y a_3 & \beta_x b_3 + \beta_y a_3 \end{bmatrix} \tag{2.62}$$

Verifica-se que $\mathbf{M}^e_{pg}$ é igual a $(\mathbf{C}^e_g)^T$ a menos de uma constante. Portanto, pode-se representar (2.62) por:

$$\mathbf{M}^e_{pg} = \frac{\tau}{6||\boldsymbol{\beta}||}\begin{bmatrix} c_{12} + c_{13} & -c_{21} & -c_{31} \\ -c_{12} & c_{21} + c_{23} & -c_{32} \\ -c_{13} & -c_{23} & c_{31} + c_{32} \end{bmatrix} \tag{2.63}$$

Da mesma forma como mostrado anteriormente, a matriz da aresta pode ser escrita por:

$$\mathbf{M}^s_{pg} = \begin{bmatrix} c^e_{ij} + c^f_{ij} & -(c^e_{ji} + c^f_{ji}) \\ -(c^e_{ij} + c^f_{ij}) & c^e_{ji} + c^f_{ji} \end{bmatrix} \tag{2.64}$$

onde

$$\begin{array}{l} c_{ij} = -\frac{\tau}{6||\boldsymbol{\beta}||}(\beta_x b_j + \beta_y a_j) \\ c_{ji} = -\frac{\tau}{6||\boldsymbol{\beta}||}(\beta_x b_i + \beta_y a_i) \end{array} \quad , \quad i,j = 1,2,3 \quad i \neq j \tag{2.65}$$

2.4.7 Resumo das matrizes das arestas

A Matriz de Massa Consistente na aresta s pode ser representada por:

$$\mathbf{M}^s = \mathbf{M}_g^s + \mathbf{M}_{pg}^s = \frac{(A^e + A^f)t}{12}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} + \begin{bmatrix} c_{ij}^s & -c_{ji}^s \\ -c_{ij}^s & c_{ji}^s \end{bmatrix} \tag{2.66}$$

sendo os coeficientes definidos em (2.30) e (2.65). Observe que a matriz $\mathbf{M}^s$ pode ser totalmente representada por apenas três coeficientes:

$$\begin{aligned} m(s,1) &= c_{ij}^s \\ m(s,2) &= c_{ji}^s \\ m(s,3) &= \tfrac{(A^e+A^f)t}{12} \end{aligned} \tag{2.67}$$

A Matriz de rigidez da aresta s pode ser representada por:

$$\mathbf{K}^s = \mathbf{T}_g^s + \mathbf{C}_g^s + \mathbf{C}_{pg}^s = \begin{bmatrix} t_{ij}^s & -t_{ij}^s \\ -t_{ij}^s & t_{ij}^s \end{bmatrix} + \begin{bmatrix} c_{ij}^s & -c_{ij}^s \\ -c_{ji}^s & c_{ji}^s \end{bmatrix} + \begin{bmatrix} (t_{ij}^{pg})^s & -(t_{ij}^{pg})^s \\ -(t_{ij}^{pg})^s & (t_{ij}^{pg})^s \end{bmatrix} \tag{2.68}$$

sendo os coeficientes de $\mathbf{T}_g^s$, $\mathbf{C}_g^s$ e $\mathbf{C}_{pg}^s$ definidos respectivamente em (2.34), (2.51) e (2.59). Observe que a matriz $\mathbf{K}^s$ pode ser totalmente representada por apenas dois coeficientes distintos:

$$\begin{aligned} k(s,1) &= t_{ij}^s + c_{ij}^s + (t_{ij}^{pg})^s \\ k(s,2) &= t_{ij}^s + c_{ji}^s + (t_{ij}^{pg})^s \end{aligned} \tag{2.69}$$

Portanto, a matriz da aresta pode ser representada por :

$$\mathbf{K}^s = \begin{bmatrix} k(s,1) & -k(s,1) \\ -k(s,2) & k(s,2) \end{bmatrix} \tag{2.70}$$

Este trabalho trata apenas de aplicações estacionárias para problemas de advecção - difusão. Portanto, o problema descrito em (2.1) resume-se em encontrar a solução do sistema linear:

$$\mathbf{KU} = \mathbf{F} \tag{2.71}$$

onde

$$\mathbf{K} = \sum_{s=1}^{nedges} \mathbf{K}^s \tag{2.72}$$

$$\mathbf{F} = \sum_{s=1}^{nedges} \mathbf{F}^s \tag{2.73}$$

sendo $nedges$ o número de arestas da malha e $\mathbf{K}^s$ descrita por (2.70).

O próximo capítulo trata da solução do sistema linear resultante via métodos iterativos usando estrutura de dados por aresta. Para facilitar a abordagem troca-se o nome dos elementos envolvidos, $\mathbf{K}$, $\mathbf{U}$ e $\mathbf{F}$ pela denominação usual de sistemas lineares, $\mathbf{A}$, $\mathbf{x}$ e $\mathbf{b}$, respectivamente.

2.5 Estudo Comparativo de Estruturas de Dados para Matrizes Esparsas

Este item tem por objetivo comparar estratégias para representação de matrizes esparsas enfatizando esquemas de armazenamento. Um breve estudo de grafos e representação de malhas por grafos é introduzido.

2.5.1 Representação de uma matriz esparsa por grafos

Formulações de elementos finitos em geometria complexa geram matrizes não-estruturadas. Neste caso, os elementos não nulos ocorrem em posições arbitrárias, sendo necessário o conhecimento de dados da malha sempre que for preciso acessá-los. A eficiência dos métodos iterativos está fortemente ligada a esse fato, pois manipula somente os coeficientes não nulos da matriz. A Teoria dos Grafos é uma importante ferramenta para representar estrutura de matrizes esparsas, sendo usada nas técnicas de paralelismo por diversos algoritmos de solução de sistemas lineares

Um grafo não direcionado $G = (V, E)$ poder ser definido pelos conjuntos:

- Um conjunto de vértices:

  $V = \{v_1, v_2, \ldots, v_n\}$

- Um conjunto de arcos E que consiste do par (v_i, v_j), onde v_i e v_j são elementos de V, isto é, $E \subseteq V \times V$.

Uma matriz esparsa de ordem n, pode ser representada por um grafo $G = (V, E)$, sendo as n variáveis representadas pelos n vértices em V. Os arcos representam a relação binária estabelecida pelas equações da seguinte forma:

Existe um arco do nó i para o nó j quando $a_{ij} \neq 0$

Este arco representará então, as relações binárias:

Equação i envolve a variável j
Equação j envolve a variável i

2.5.2 Representação de Grafos para Malhas de Elementos Finitos

Existe uma associação intrínseca entre a mariz esparsa originada pelo método dos elementos finitos e a topologia da malha. Estas propriedades topológicas podem ser exploradas através da associação de grafos, malhas e matrizes esparsas. Três tipos de associações são mais utilizados: grafo nodal, grafo dual e grafo de comunicação [21]. O grafo nodal tem como vértices os nós da malha e a ligação dos vértices são definidas pelas arestas. Enquanto o grafo dual tem por vértices os elementos e a ligação entre vértices ocorre quando há arestas comuns a dois elementos adjacentes. O grafo de comunicação também tem por vértices os elementos, porém a ligação entre eles é definida por todos os elementos que tem pelo menos um nó comum. A Figura 2.3 mostra um exemplo de malha de elementos finitos e os três tipos de grafos correspondentes.

O grafo nodal apresenta boas caracteristicas para armazenar a matriz esparsa envolvida. Observa-se ainda, para os elementos triangulares, que o grafo nodal equivale à malha de elementos. Para este tipo de malha, o uso de estrutura de dados por aresta mostra-se extremamente conveniente, pois ao tratar cada aresta individualmente existe uma associação direta entre os valores dos coeficientes da matriz e seus posicionamentos. O mesmo não ocorre ao se tratar o problema pelo enfoque dos elementos, uma vez que estes se associam indiretamente à matriz através de submatrizes das matrizes de cada elemento,sendo que, percorrer os elementos é equivalente a percorrer o grafo dual. Os grafos dual e de comunicação são amplamente utilizados para problemas que envolvam decomposição de domínio [21].

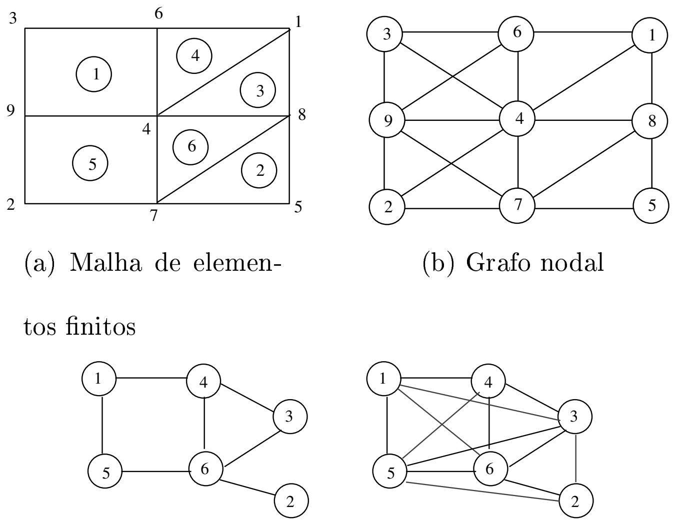
2.5.3 Armazenamento de uma matriz esparsa

Várias técnicas tem sido definidas com o objetivo principal de armazenar somente os elementos diferentes de zero. Neste item é feita uma comparação entre técnicas que usam estrutura de dados diferentes, visando explorar a otimalidade com relação a memória utilizada. Considera-se uma matriz esparsa A de ordem n com um número total de elementos diferentes de zero igual a N_z.

Estrutura de dados usando o formato CSR (*Compressed Sparse Row*)

Na estrutura CSR são necessários 3 vetores sendo:

- Um vetor real AA contendo os valores reais $a_{ij} \neq 0$ armazenados linha por linha, da linha 1 até a linha n. AA tem N_z elementos.
- Um vetor de inteiros JA contendo o índice da coluna de cada elemento a_{ij} armazenado em AA, com N_z elementos.
- um vetor IA de inteiros contendo os ponteiros onde começam cada linha em AA e JA. Ou seja, o elemento $IA(i)$ contém a posição de AA e JA onde a i-ésima linha começa. O tamanho de IA é $n+1$, com $IA(n+1)$ contendo o número $IA(1)+N_z$, isto é, o endereço em AA e JA do começo de um número de linha fictício $n+1$.



(a) Malha de elementos finitos

(b) Grafo nodal

(c) Grafo Dual

(d) Grafo de Comunicação

Figura 2.3: Diferenças entre tipos de grafos para uma malha de elementos finitos

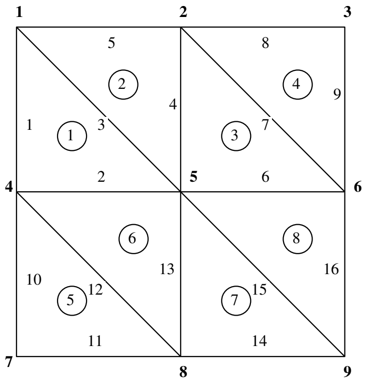


Figura 2.4: Malha de elementos triangulares lineares com 8 elementos, 9 nós e 16 arestas

A Figura 2.4 representa uma malha de elementos triangulares. A matriz esparsa correspondente possui 41 coeficientes diferentes de zero:

$$A = \begin{bmatrix} \otimes & \times & & \times & \times & & & & \\ \times & \otimes & \times & & \times & \times & & & \\ & \times & \otimes & & & \times & & & \\ \times & & & \otimes & \times & & \times & \times & \\ \times & \times & & \times & \otimes & \times & & \times & \times \\ & \times & \times & & \times & \otimes & & & \times \\ & & & \times & & & \otimes & \times & \\ & & & \times & \times & & \times & \otimes & \times \\ & & & & \times & \times & & \times & \otimes \end{bmatrix}$$

O armazenamento do tipo CSR para esse exemplo pode ser representado por:

$$\begin{array}{lllllllllllllll} AA = [& \overbrace{\otimes}^{1} & \times & \times & \times & \overbrace{\times}^{5} & \otimes & \times & \times & \ldots & \overbrace{\times}^{38} & \times & \times & \times] \\ JA = [& 1 & 2 & 4 & 5 & 1 & 2 & 3 & 5 & \ldots & 5 & 6 & 8 & 9] \\ IA = [& 1 & 5\ldots & 38 & 42] \end{array}$$

Para este caso são necessárias 92 posições de memória.

Estrutura de dados usando o formato MSR (*Modified Sparse Row*)

Uma variação comum do esquema CSR explora o fato da diagonal principal de A ser usualmente de elementos diferentes de zero e ser acessada com maior frequência que os demais elementos. O formato MSR utiliza somente 2 vetores:

- Um vetor AA de reais, contendo os elementos não nulos de A. A diagonal de A ocupa as n primeiras posições de AA. A posição $(n+1)$ do vetor AA não é usada para coeficientes de A, mas pode ser usada para alocar alguma informação adicional de A. A partir da posição $(n+2)$ os coeficientes não nulos são armazenados por linha.

- Um vetor de inteiros JA, contendo nas $(n+1)$ primeiras posições os ponteiros de começo de cada linha de AA. Nas posições restantes estão as colunas correspondentes de cada elemento em AA, ou seja, para cada elemento $AA(k)$, o inteiro $JA(k)$ representa o índice da coluna na matriz A. No esquema a seguir encontra-se o armazenamento do tipo MSR para a malha ilustrada na Figura 2.4:

$$AA = [\ \otimes \quad \otimes \quad \ldots \quad \otimes \quad * \quad \overbrace{\times}^{11} \quad \times \quad \overbrace{\times}^{14} \quad \ldots \quad \overbrace{\times}^{40} \quad \times \quad \times]$$

$$JA = [\ 11 \quad 14 \quad \ldots \quad 40 \quad 42 \quad 4 \quad 5 \quad 1 \quad \ldots \quad 5 \quad 6 \quad 8]$$

Neste exemplo são necessárias 84 posições de memória.

Estrutura de Dados Elemento por Elemento

Neste caso os coeficientes da matriz esparsa A são armazenados a nível de cada elemento. Considerando elementos triangulares lineares, tem-se em cada elemento uma matriz de ordem 3, ou seja, 9 coeficientes. Para uma malha de elementos finitos com nel elementos, todos os coeficientes não nulos da matriz esparsa A podem ser armazenados na matriz de reais AA, com 9 colunas e nel linhas. Além disso, é necessário um arranjo de inteiros $lm(nel, 3)$, contendo as incidências de cada elemento. Em geral, a diagonal de A é acessada com frequência, sendo guardada separadamente em um vetor $diagA$ de ordem n. Para o exemplo de malha ilustrado na Figura 2.4, tem-se o seguinte esquema de armazenamento:

$$AA = \begin{bmatrix} \times & \times & \times & \dots & \dots & \times \\ \times & \times & \times & \dots & \dots & \times \\ \times & \times & \times & \dots & \dots & \times \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ \times & \times & \times & \dots & \dots & \times \end{bmatrix}, \text{ com 8 linhas e 9 colunas}$$

$$lm = \begin{bmatrix} 1 & 4 & 5 \\ 1 & 5 & 2 \\ 2 & 5 & 6 \\ \vdots & \vdots & \vdots \\ 5 & 9 & 6 \end{bmatrix}, \text{ com 8 linhas e 3 colunas}$$

$$diagA = [\otimes \quad \otimes \quad \otimes \quad \otimes \quad \otimes \quad \otimes \quad \otimes \quad \otimes \quad \otimes]$$

São necessárias 72 posições de memória para AA, 24 posições de memória para lm e 9 posições para $diagA$, totalizando 105 posições de memória.

Estrutura de Dados Aresta-por-Aresta

Os coeficientes da matriz esparsa A são agora armazenados a nível de cada aresta da malha. Como já foi visto nas expressões (2.69) e (2.70), tem-se em cada aresta, uma matriz de ordem 2, com apenas 2 coeficientes distintos, correspondentes aos termos de fora da diagonal. Para um malha com $nedges$ arestas, os coeficientes de A são armazenados na matriz de inteiros AA com 2 colunas e $nedges$ linhas. Sendo necessário um arranjo auxiliar $lm(negdes, 2)$ contendo as incidências de cada aresta e um vetor de n posições para armazenar $diagA$. Para o exemplo de malha ilustrado na Figura 2.4, tem-se o esquema de armazenamento:

$$AA = \begin{bmatrix} \times & \times \\ \times & \times \\ \vdots & \vdots \\ \times & \times \end{bmatrix}, \text{ com 16 linhas e 2 colunas}$$

$$lm = \begin{bmatrix} 1 & 4 \\ 4 & 5 \\ 5 & 1 \\ \vdots & \vdots \\ 9 & 6 \end{bmatrix}, \text{ com 16 linhas e 2 colunas}$$

$$diagA = [\otimes \quad \otimes \quad \otimes \quad \otimes \quad \otimes \quad \otimes \quad \otimes \quad \otimes \quad \otimes]$$

São necessárias 32 posições de memória para AA, 32 posições de memória para lm e 9 posições para $diagA$, totalizando 73 posições de memória.

Observa-se que para a malha descrita na Figura 2.4 o esquema por aresta é mais vantajoso que os demais com relação à memória utilizada. Porém, Martins, em [20], demonstra que em geral a estratégia por aresta é mais econômica que a estratégia por elemento para qualquer tipo de malha, estruturada ou não.

Capítulo 3

Solução de Sistemas Não-Simétricos via Estrutura de Dados por Arestas

Neste capítulo é apresentada a solução de sistemas não simétricos pelo método iterativo GMRES (Resíduo Mínimo Generalizado) [2], usando estrutura de dados por arestas obtida da formulação descrita no capítulo 2.

Seja o sistema

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{3.1}$$

onde $\mathbf{x}^T = \{x_1, \ldots, x_{neq}\}^T$ é o vetor de variáveis nodais, sendo neq o número de incógnitas; $\mathbf{A}$ é a matriz não simétrica $neq \times neq$; $\mathbf{b}$ é o vetor de termos independentes também de ordem neq.

A matriz $\mathbf{A}$ e o vetor $\mathbf{b}$ são construídos a partir da montagem das contribuições das arestas da malha de elementos triangulares lineares, isto é,

$$\mathbf{A} = \sum_{s=1}^{nedges} \mathbf{A}_s \tag{3.2}$$

$$\mathbf{b} = \sum_{s=1}^{nedges} \mathbf{b}_s \tag{3.3}$$

onde $\mathbf{A}_s$ e $\mathbf{b}_s$ são descritos a nível de cada aresta da malha.

3.1 Método GMRES

O método iterativo GMRES [2], [4] tem por objetivo básico minimizar a norma residual do sistema (3.1). Considerando $\mathbf{x}_0$ a solução inicial, uma solução aproximada apresenta a forma $\mathbf{x}_0 + \mathbf{z}$ onde $\mathbf{z}$ é um vetor do espaço de Krylov $K = span\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, \ldots, \mathbf{A}^{k-1}\mathbf{r}_0\}$, sendo $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$ e k é a dimensão de K. O método GMRES determina $\mathbf{z}$ tal que a norma do resíduo seja mínima, isto é,

$$\mathbf{x}_0 + \mathbf{z} \ \text{ é solução aproximada de (3.1) se } \ \|\mathbf{b} - \mathbf{A}(\mathbf{x}_0 + \mathbf{z})\| \ \text{ é mínima.}$$

onde $\|.\|$ denomina norma euclidiana.

Uma base ortonormal de K pode ser obtida pelo Processo Modificado de Gram-Schmidt [4]:

$\mathbf{u}_1 = \frac{\mathbf{r}_0}{\|\mathbf{r}_0\|}$

`Para` $i = 1, \ldots, k$ `faça`

    $\tilde{\mathbf{u}}_{i+1} = \mathbf{A}\mathbf{u}_i$

    `Para` $j = 1, \ldots, i$ `faça`

        $\beta_{i+1,j} = (\tilde{\mathbf{u}}_{i+1}, \mathbf{u}_j)$

        $\tilde{\mathbf{u}}_{i+1} \leftarrow \tilde{\mathbf{u}}_{i+1} - \beta_{i+1,j}\mathbf{u}_j$

    `Fim do` j

    $\mathbf{u}_{i+1} = \frac{\tilde{\mathbf{u}}_{i+1}}{\|\tilde{\mathbf{u}}_{i+1}\|}$

`Fim do` i

Seja $\mathbf{U}_k = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k]$. Pode-se então, mostrar que $\mathbf{AU}_k = \mathbf{U}_{k+1}\mathbf{H}_k$ onde $\mathbf{H}_k$ é a matriz superior de Hessenberg de ordem $(k+1) \times k$,

$$\mathbf{H}_k = \begin{bmatrix} \beta_{2,1} & \beta_{3,1} & \ldots & \beta_{k,1} & \beta_{k+1,1} \\ \|\tilde{\mathbf{u}}_2\| & \beta_{3,2} & \ldots & \beta_{k,2} & \beta_{k+1,2} \\ 0 & \|\tilde{\mathbf{u}}_3\| & \ddots & \vdots & \vdots \\ \vdots & 0 & \ddots & \beta_{k,k-1} & \vdots \\ \vdots & \vdots & \ddots & \|\tilde{\mathbf{u}}_k\| & \beta_{k+1,k} \\ 0 & 0 & \vdots & 0 & \|\tilde{\mathbf{u}}_{k+1}\| \end{bmatrix} \tag{3.4}$$

Seja $\mathbf{z} = \sum_{j=1}^{k} y_j u_j$ e $\mathbf{e} = \{\|\mathbf{r}_0\|, 0, \ldots, 0\}^T$ com $k+1$ termos. Observe que $\mathbf{r}_0 = \mathbf{U}_{k+1}\mathbf{e}$, portanto, tem-se:

$$\begin{aligned} \|\mathbf{b} - \mathbf{A}(x_0 + z)\| &= \|\mathbf{r}_0 - \mathbf{A}(\textstyle\sum_{j=1}^{k} y_j u_j)\| \\ &= \|\mathbf{r}_0 - A\mathbf{U}_k\mathbf{y}\| \\ &= \|U_{k+1}(\mathbf{e} - \mathbf{H}_k\mathbf{y})\| \\ &= \|\mathbf{e} - \mathbf{H}_k\mathbf{y}\| \end{aligned} \tag{3.5}$$

Assim o problema de minimização em questão pode ser reescrito por:

$$min_{\mathbf{z}\in K}\|\mathbf{b} - \mathbf{A}(x_0 + z)\| = min_{\mathbf{y}\in\Re^k}\|\mathbf{e} - \mathbf{H}_k\mathbf{y}\| \tag{3.6}$$

A matriz $\mathbf{H}_k$ tem um estrutura quase triangular superior, necessitando-se somente eliminar a diagonal contendo $\|\tilde{\mathbf{u}}_2\|, \|\tilde{\mathbf{u}}_3\|, \ldots, \|\tilde{\mathbf{u}}_{k+1}\|$. A transformação de $\mathbf{H}_k$ em uma matriz triangular superior pode ser feita pelo algoritmo QR, usando em cada passo o processo de rotações de Givens para obter um novo problema de minimização, maiores detalhes podem ser vistos em [4]. Assim, obtem-se:

$$min_{\mathbf{y}\in\Re^k}\|\bar{\mathbf{e}} - \bar{\mathbf{H}}_k\mathbf{y}\| \tag{3.7}$$

onde $\bar{\mathbf{H}}_k$ é uma matriz triangular superior. O sistema pode então, ser resolvido facilmente por retrosubstituição e um novo resíduo é calculado com a aproximação encontrada. Se a convergência não for atingida dentro de um ciclo (formado pelo número de vetores na base de Krylov), a solução mais recente será usada para iniciar a próxima iteração. Resumindo, uma iteração do Método GMRES consiste de:

- formação de uma base ortogonal para o espaço de Krylov pelo processo modificado de Gram-Schmidt
- triangularização da Matriz de Hessenberg pelo algoritmo QR
- Retrosubstituição para obter a solução aproximada.

Com estas considerações descreve-se um algoritmo padrão do método GMRES. Para tal define-se os parâmetros:

ε_{tol} $\rightarrow$ tolerância

k $\rightarrow$ número de vetores de Krylov

l_{max} $\rightarrow$ número máximo de iterações

`Dados` $\mathbf{A}$, $\mathbf{b}$, k, ε_{tol} `e` l_{max}

(Inicializações)

$\varepsilon \leftarrow \varepsilon_{tol} b$

$\mathbf{x} = 0$

(ciclo GMRES)

`Para` $l = 1, \ldots, l_{max}$ `faça`

    $\mathbf{u}_1 = \mathbf{b} - \mathbf{Ax}$

    $\bar{e} = \|\mathbf{u}_1\|$

    $\mathbf{u}_1 = \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|}$

    (iterações do GMRES)

    `Para` $i = 1, \ldots, k$ `faça`

        $\mathbf{u}_{i+1} = \mathbf{Au}_i$

        (Ortogonalização modificada de Gram-Schmidt)

        `Para` $j = 1, \ldots, i$ `faça`

            $\beta_{i+1,j} = (\mathbf{u}_{i+1}, \mathbf{u}_j)$

            $\mathbf{u}_{i+1} \leftarrow \mathbf{u}_{i+1} - \beta_{i+1,j}\mathbf{u}_j$

        `Fim do` j

        (Fim da ortogonalização modificada de Gram-Schmidt)

        $\bar{\mathbf{h}}^{(i)} = \{\beta_{i+1,1}, \ldots, \beta_{i+1,i}, \|\mathbf{u}_{i+1}\|\}^T$

        $\mathbf{u}_{i+1} = \frac{\mathbf{u}_{i+1}}{\|\mathbf{u}_{i+1}\|}$

(Algoritmo QR)

`Para` $j = 1, \ldots, i-1$ `faça`

$$\begin{Bmatrix} \bar{h}_j^{(i)} \\ \bar{h}_{j+1}^{(i)} \end{Bmatrix} \leftarrow \begin{bmatrix} c_j & s_j \\ -s_j & c_j \end{bmatrix} \begin{Bmatrix} \bar{h}_j^{(i)} \\ \bar{h}_{j+1}^{(i)} \end{Bmatrix}$$

`Fim do` j

$r = \sqrt{(\bar{h}_i^{(i)})^2 + (\bar{h}_{i+1}^{(i)})^2}$

$c_i = \frac{\bar{h}_i^{(i)}}{r}$

$s_i = \frac{\bar{h}_{i+1}^{(i)}}{r}$

$\bar{h}_i^{(i)} \leftarrow r$

$\bar{h}_{i+1}^{(i)} \leftarrow 0$

$\bar{e}_{i+1} = -s_i \bar{e}_i$

$\bar{e}_i = c_i \bar{e}_i$

(Fim do algoritmo QR)

`Teste de convergência:  Se` $|\bar{e}_{i+1}| \leq \varepsilon_{tol}$, `fim do` i

(Fim da iteração GMRES)

`Resolver` $\mathbf{y}$:

$$\begin{bmatrix} \bar{h}_1^{(1)} & \ldots & \bar{h}_1^{(i-1)} & \bar{h}_1^{(i)} \\ 0 & \ddots & \vdots & \vdots \\ \vdots & \ddots & \bar{h}_{i-1}^{(i-1)} & \bar{h}_{i-1}^{(i)} \\ 0 & \ldots & 0 & \bar{h}_i^{(i)} \end{bmatrix} \begin{Bmatrix} y_1 \\ \vdots \\ y_{i-1} \\ y_i \end{Bmatrix} = \begin{Bmatrix} \bar{e}_1 \\ \vdots \\ \bar{e}_{i-1} \\ \bar{e}_i \end{Bmatrix}$$

`Cálculo da Solução:`

$$\mathbf{x} \leftarrow \mathbf{x} + \sum_{j=1}^{i} y_j u_j$$

`Teste de Convergência:  Se` $|\bar{e}_{i+1}| \leq \varepsilon$, `fim do`$l$

(Fim do ciclo GMRES)

`x é a solução aproximada`

3.2 Multiplicação matriz-vetor aresta-por-aresta

Em vários passos do algoritmo GMRES nota-se a necessidade de executar a multiplicação matriz-vetor. A matriz global $\mathbf{A}$ nunca é explicitamente formada, portanto a operação em questão pode ser feita usando a estratégia aresta-por-aresta:

$$\mathbf{Ap} = \sum_{s=1}^{nedges} \mathbf{A}_s \mathbf{p}_s \tag{3.8}$$

onde $nedges$ é o número de arestas da malha, $\mathbf{A}_s$ é a matriz da aresta, descrita no capítulo 2, e $\mathbf{p}_s$ são as componentes de $\mathbf{p}$ restritas aos graus de liberdade da aresta. Porém, é possível otimizar ainda mais a operação fazendo um desmembramento da diagonal principal de $\mathbf{A}$, [22]:

$$\mathbf{Ap} = diag(\mathbf{A})\mathbf{p} + \sum_{s=1}^{nedges} (\mathbf{A}_s - diag\mathbf{A}_s)\mathbf{p}_s \tag{3.9}$$

Note que este procedimento diminui o número de endereçamentos indiretos necessários à operação. A implementação da multiplicação matriz-vetor aresta-por-aresta em linhas gerais segue o seguinte algoritmo:

`Para cada aresta` s `faça`

> `localizar` $\mathbf{p}_s$ `em` $\mathbf{p}$
>
> `calcular` $\mathbf{v}_s = (\mathbf{A}_s - diag\mathbf{A}_s)\mathbf{p}_s$
>
> `espalhar + acumular` $\mathbf{v} = \mathbf{v} + \mathbf{v}_s$

`fim do` s

$\mathbf{v} = diag(\mathbf{A})\mathbf{p} + \mathbf{v}$

É importante notar que este algoritmo apresenta um ótimo potencial de vetorização e paralelização, uma vez que os passos `localizar` e `calcular` podem ser executados independentemente para cada aresta. Porém o passo `espalhar + acumular` envolve operação de acesso a arranjos globais. Para arestas adjacentes as operações de acesso são executadas em endereços comuns de $\mathbf{v}$. Portanto, operações concorrentes de `espalhar + acumular` são possíveis somente dentro de grupos não adjacentes de arestas. Estes grupos são construídos a partir de um pré-processamento de arestas por um algoritmo de coloração de malha do tipo guloso [2]. As operações `localizar` e `espalhar` são efetuadas através de um arranjo $\mathrm{lm}(nedges, nd)$, onde nd é o número de graus de liberdade por arestas. Este arranjo define o mapeamento entre os graus de liberdade locais da aresta e os graus de liberdade globais do problema. Em geral é uma estrutura de dados presente na maioria das implementações do método dos elementos finitos [19]. A seguir, apresenta-se o algoritmo de multiplicação matriz-vetor na forma aresta-por-aresta:

`para` $s = 1, \ldots, nedges$ `faça`

(localizar as componentes globais de 'p')

$neq1 = lm(s, 1)$

$neq2 = lm(s, 2)$

$p1 = p(neq1)$

$p2 = p(neq2)$

(multiplicação matriz-vetor aresta por aresta)

$ap1 = k(s, 1) * p2$

$ap2 = k(s, 2) * p1$

(espalhar e acumular 'ap' global)

$ap(neq1) = ap(neq1) - ap1$

$ap(neq2) = ap(neq2) - ap2$

`fim do` s

Observa-se que em cada aresta são necessárias 4 *flop* (operações de ponto flutuante) e 6 i/a (endereçamentos indiretos).

3.3 Multiplicação matriz-vetor através de super-arestas

A quantidade de *flop* na operação matriz-vetor é inferior no esquema aresta, comparado com o esquema elemento, porém o número de endereçamentos indiretos necessários à operação de multiplicação matriz-vetor é equivalente nos dois esquemas [20]. É possível reduzir o número de endereçamentos indiretos agrupando arestas que possuam nós comuns, pois dados referentes a um nó que pertença a mais de uma aresta adjacente podem ser acessados de uma única vez. Assim, uma vez que a aresta é acessada na memória, ela deve permanecer e ser utilizada ao máximo, nos registradores. Existem vários tipos de agrupamentos propostos por Lohner em [12], porém aquele que apresenta melhor desempenho considerando custo-benefício são as super-arestas, segundo estudos realizados por Martins em [20]. Para o caso de implementações em 2-D são usados grupos com 3 arestas denominados *superedge3*.

O produto matriz-vetor para *superedge3* é calculado por:

$$\mathbf{Ap} = \sum_{s=1}^{nedges,3} \mathbf{A}_s \mathbf{p}_s + \mathbf{A}_{s+1} \mathbf{p}_{s+1} + \mathbf{A}_{s+2} \mathbf{p}_{s+2} \qquad (3.10)$$

onde o laço é executado sobre o total de arestas da malha, *nedges* de 3 em 3 arestas, que são os *seperedges3*. $\mathbf{A}_{s+i}$ são as matrizes de cada aresta formadora do grupo e $\mathbf{p}_{s+i}$ são as componentes de $\mathbf{p}$ restritas aos graus de liberdade de cada aresta formadora do grupo. Como no caso de arestas simples o produto será executado fazendo um desmembramento da diagonal:

$$\mathbf{Ap} = diag\mathbf{Ap} + \sum_{s=1}^{nedges,3} \widehat{\mathbf{A}}_s\mathbf{p}_s + \widehat{\mathbf{A}}_{s+1}\mathbf{p}_{s+1} + \widehat{\mathbf{A}}_{s+2}\mathbf{p}_{s+2} \tag{3.11}$$

onde:

$$\widehat{\mathbf{A}}_s = \mathbf{A}_s - diag\mathbf{A}_s$$

$$\widehat{\mathbf{A}}_{s+1} = \mathbf{A}_{s+1} - diag\mathbf{A}_{s+1}$$

$$\widehat{\mathbf{A}}_{s+2} = \mathbf{A}_{s+2} - diag\mathbf{A}_{s+2}$$

Para cada *superedge3* a multiplicação matriz-vetor executa os seguintes passos:

`localizar` $\mathbf{p}_s,\ \mathbf{p}_{s+1},\ \mathbf{p}_{s+2} \rightarrow \mathbf{p}_{grupo}$ em $\mathbf{p}$

`calcular` $\mathbf{Ap}_{grupo} = \widehat{\mathbf{A}}_s\mathbf{p}_s + \widehat{\mathbf{A}}_{s+1}\mathbf{p}_{s+1} + \widehat{\mathbf{A}}_{s+2}\mathbf{p}_{s+2}$

`espalhar` $\mathbf{Ap}_{grupo}$ em $\mathbf{Ap}$

As operações `localizar` e `espalhar` são efetuadas através do arranjo $lm(nedges, nd)$, já detalhado para o esquema de aresta simples. Estas operações são responsáveis pela ocorrência dos endereçamentos indiretos. Nesta estratégia de agrupamento de 3 arestas, em vez de 6 operações de endereçamentos indiretos para as 3 arestas consideradas em separado, são realizadas apenas 3, visto que o grupo possui 3 nós. A seguir apresenta-se o algoritmo matriz-vetor para o *superedge3*:

```
para s = 1, nedges, 3
    neq1 = lm(s,1)
    neq2 = lm(s,2)
    neq3 = lm(s+1,2)
    p1 = p(neq1)
    p2 = p(neq2)
    p3 = p(neq3)
    ap1 = k(s,1) * p2 + k(s+2,2) * p3
    ap2 = k(s,2) * p1 + k(s+1,1) * p3
    ap3 = k(s+1,2) * p2 + k(s+2,1) * p1
    ap(neq1) = ap(neq1) - ap1
    ap(neq2) = ap(neq2) - ap2
    ap(neq3) = ap(neq3) - ap3
fim do s
```

Neste esquema são necessários 9 endereçamentos indiretos e 9 *flop*. Observa-se que há um ganho de otimalidade com relação ao esquema de arestas simples, pois o total de acessos a arestas é feito de 3 em 3.

3.4 Pré-condicionadores para o método GMRES

A fragilidade dos métodos iterativos com relação à rapidez de convergência é amplamente conhecida, portanto técnicas que permitam acelerar a convergência tem sido desenvolvidas. Tais técnicas são caracterizadas pelo uso de pré-condicionadores. Grosseiramente falando, um pré-condicionamento é uma maneira explícita ou implícita de modificar um sistema original de forma que se possa resolvê-lo "facilmente" por um dado método iterativo. Por exemplo, o escalonamento de todas as linhas de um sistema linear feita pelos coeficientes da diagonal principal é uma forma explícita de pré-condicionamento. Outros exemplos poderiam ser:

$$\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{b} \tag{3.12}$$

$$\mathbf{M}^{-1}\mathbf{A}\mathbf{N}^{-1}\mathbf{N}\mathbf{x} = \mathbf{M}^{-1}\mathbf{b} \tag{3.13}$$

$$\mathbf{A}\mathbf{N}^{-1}\mathbf{N}\mathbf{x} = \mathbf{b} \tag{3.14}$$

onde (3.12) é conhecido como pré-condicionador à esquerda, (3.13) é conhecido por pré-condicionador à esquerda e direita e finalmente (3.14) é conhecido como pré-condicionador à direita. $\mathbf{M}$ e $\mathbf{N}$ podem representar simples decomposições ou complicadas transformações de $\mathbf{A}$. Em geral, a eficiência do pré-condicionador é inversamente proporcional à complexidade da transformação, pois levando em consideração que o maior objetivo é diminuir o tempo de execução, os pré-condicionadores com decomposições simples acabam sendo mais vantajosos.

Segundo Saad em [2], encontrar um bom pré-condicionador para resolver um sistema linear é frequentemente visto como uma combinação de arte e ciência. Resultados teóricos são raros e algumas técnicas oferecem resultados surpreendentes para algumas aplicações, outras nem tanto. Neste trabalho, apresentam-se técnicas simples de decomposição de $\mathbf{A}$, similares àquelas desenvolvidas em [4], [6] e [7], porém usando estrutura de dados por arestas. A eficiência da operação matriz-vetor, mais uma vez é a maior preocupação, pois nela baseia-se a transformação do sistema. Definem-se dois estágios para os pré-condicionadores desenvolvidos:

- normalização dos elementos de $\mathbf{A}$, através dos coeficientes da diagonal principal
- transformação do sistema usando decomposições simples de $\mathbf{A}$

3.4.1 Primeiro Estágio

Introduz-se uma transformação de escala nos coeficientes do sistema, que pode ser vista como um pré-precondicionamento. O novo sistema a ser resolvido é:

$$\tilde{\mathbf{A}}\tilde{\mathbf{x}} = \tilde{\mathbf{b}} \tag{3.15}$$

onde $\tilde{\mathbf{A}} = \mathbf{W}^{-1/2}\mathbf{A}\mathbf{W}^{-1/2}$, sendo $\mathbf{W} = diag\mathbf{A}$; $\tilde{\mathbf{b}} = \mathbf{W}^{-1/2}\mathbf{b}$ e $\tilde{\mathbf{x}} = \mathbf{W}^{-1/2}\mathbf{x}$. Observe que a nível de cada aresta tem-se:

$$\mathbf{W}_s = \begin{bmatrix} k(s,1) & 0 \\ 0 & k(s,2) \end{bmatrix} \tag{3.16}$$

$$\tilde{\mathbf{A}}_s = \begin{bmatrix} 1 & -\tilde{k}(s,1) \\ -\tilde{k}(s,2) & 1 \end{bmatrix} \tag{3.17}$$

onde

$$\tilde{k}(s,1) = k(s,1)(k(s,1))^{-1/2}(k(s,2))^{-1/2}$$

$$\tilde{k}(s,2) = k(s,2)(k(s,1))^{-1/2}(k(s,2))^{-1/2}$$

Portanto, a multiplicação matriz-vetor (3.9) pode ser representada por:

$$\tilde{\mathbf{A}}\mathbf{p} = \mathbf{p} + \sum_{s=1}^{nedges} (\tilde{\mathbf{A}}_s - diag\tilde{\mathbf{A}}_s)\mathbf{p}_s \tag{3.18}$$

pois $diag\tilde{\mathbf{A}} = \mathbf{I}$.

3.4.2 Segundo Estágio

Neste estágio apresentam-se decomposições conhecidas da matriz dos coeficientes $\tilde{\mathbf{A}}$. O sistema (3.15) sofrerá transformações originadas pela decomposição incompleta da matriz de Gauss-Seidel ou pela fatoração incompleta LU [4], adaptadas para a estrutura de dados por arestas. Dos tipos de transformações citadas em (3.12), (3.13) e (3.14) serão consideradas decomposições onde os fatores são aplicados à direita e à esquerda da matriz dos coeficientes e onde todos os fatores são aplicados à esquerda da matriz dos coeficientes. No primeiro caso deseja-se resolver o sistema:

$$\bar{\mathbf{A}}\bar{\mathbf{x}} = \bar{\mathbf{b}} \tag{3.19}$$

onde $\bar{\mathbf{A}} = \mathbf{L}^{-1}\tilde{\mathbf{A}}\mathbf{U}^{-1}$, $\bar{\mathbf{x}} = \mathbf{U}\tilde{\mathbf{x}}$ e $\bar{\mathbf{b}} = \mathbf{L}^{-1}\mathbf{U}^{-1}\tilde{\mathbf{b}}$. As matrizes $\mathbf{L}$ e $\mathbf{U}$ são definidas por:

$$\mathbf{L} = \prod_{s=1}^{nedges} \mathbf{L}_s \tag{3.20}$$

$$\mathbf{U} = \prod_{s=nedges}^{1} \mathbf{U}_s \tag{3.21}$$

onde as matrizes $\mathbf{L}_s$ e $\mathbf{U}_s$ são os fatores incompletos do tipo Gauss-Seidel ou do tipo LU da matriz A_s que posteriomente serão tratadas com detalhes.

O pré-condicionador agirá no algoritmo GMRES, em cada passo iterativo, no momento em que for necessário executar a operação matriz-vetor. Assim, o produto $\mathbf{v} = \bar{\mathbf{A}}\mathbf{p}$ apresenta os seguintes passos:

$$\begin{aligned} \mathbf{v} &\leftarrow \mathbf{U}^{-1}\mathbf{p} \\ \mathbf{v} &\leftarrow \tilde{\mathbf{A}}\mathbf{v} \\ \mathbf{v} &\leftarrow \mathbf{L}^{-1}\mathbf{v} \end{aligned} \tag{3.22}$$

O produto $\tilde{\mathbf{A}}\mathbf{v}$ será executado por (3.9) no caso de arestas simples e por (3.11) no caso de *superedges3*. É importante notar que as operações $\mathbf{U}^{-1}\mathbf{p}$ e $\mathbf{L}^{-1}\mathbf{v}$ são resolvidas usando idéias básicas de retrosubstituição e substituição dos sistemas triangulares, ou seja, não há necessidade do cálculo de inversas.

No caso de pré-condicionadores à esquerda deseja-se resolver o sistema:

$$\bar{\mathbf{A}}\tilde{\mathbf{x}} = \bar{\mathbf{b}} \tag{3.23}$$

onde $\bar{\mathbf{A}} = \mathbf{L}^{-1}\mathbf{U}^{-1}\tilde{\mathbf{A}}$, e $\bar{\mathbf{b}} = \mathbf{L}^{-1}\mathbf{U}^{-1}\tilde{\mathbf{b}}$. As matrizes $\mathbf{L}$ e $\mathbf{U}$ são definidas por (3.20) e (3.21). Como já citado, a ação dos pré-condicionadores está localizada principalmente na operação de multiplicação matriz-vetor. Porém, neste caso, o produto $\mathbf{v} = \bar{\mathbf{A}}\mathbf{p}$ apresenta os passos:

$$\begin{aligned} \mathbf{v} &\leftarrow \tilde{\mathbf{A}}\mathbf{p} \\ \mathbf{v} &\leftarrow \mathbf{U}^{-1}\mathbf{v} \\ \mathbf{v} &\leftarrow \mathbf{L}^{-1}\mathbf{v} \end{aligned} \tag{3.24}$$

sendo que as mesmas observações feitas em (3.22) são válidas para este caso. A seguir, definimos com detalhes os fatores das decomposições citadas.

Pré-condicionador Gauss-Seidel aresta-por-aresta

Para a decomposição incompleta de Gauss-Seidel cada matriz $\mathbf{A}_s$ será decomposta por:

$$\mathbf{L}_s + \mathbf{U}_s = \tilde{\mathbf{A}}_s - diag\tilde{\mathbf{A}}_s + 2\mathbf{I} \tag{3.25}$$

portanto

$$\tilde{\mathbf{L}}_s = \begin{bmatrix} 1 & 0 \\ -\tilde{k}(s,2) & 1 \end{bmatrix} \tag{3.26}$$

$$\tilde{\mathbf{U}}_s = \begin{bmatrix} 1 & -\tilde{k}(s,1) \\ 0 & 1 \end{bmatrix} \tag{3.27}$$

sendo $\tilde{k}(s,1)$ e $\tilde{k}(s,2)$ definidos em (3.17) . Observe que não há necessidade de armazenar $\mathbf{L}_s$ e $\mathbf{U}_s$ pois seus coeficientes são os próprios termos de $\mathbf{A}_s$.

Pré-condicionador Gauss-Seidel *superedge3*

Para este caso, imagine a matriz formada pelo grupo $\tilde{\mathbf{A}}_s$, $\tilde{\mathbf{A}}_{s+1}$, $\tilde{\mathbf{A}}_{s+2}$:

$$\tilde{\mathbf{A}}_{gr} = \begin{bmatrix} \tilde{k}(s,1)+\tilde{k}(s+2,2) & -\tilde{k}(s,1) & -\tilde{k}(s+2,2) \\ -\tilde{k}(s,2) & \tilde{k}(s,2)+\tilde{k}(s+1,1) & -\tilde{k}(s+1,1) \\ -\tilde{k}(s+2,1) & -\tilde{k}(s+1,2) & \tilde{k}(s+1,2)+\tilde{k}(s+2,1) \end{bmatrix} \tag{3.28}$$

sendo $\tilde{k}(s+i,j)$ definidas de forma similar a (3.17). A decomposição deste grupo será então definida por:

$$\mathbf{L}_{gr} + \mathbf{U}_{gr} = \tilde{\mathbf{A}}_{gr} - diag\tilde{\mathbf{A}}_s + 2\mathbf{I} \tag{3.29}$$

e portanto

$$\mathbf{L}_{gr} = \begin{bmatrix} 1 & 0 & 0 \\ -\tilde{k}(s,2) & 1 & 0 \\ -\tilde{k}(s+2,1) & -\tilde{k}(s+1,2) & 1 \end{bmatrix} \tag{3.30}$$

$$\mathbf{U}_{gr} = \begin{bmatrix} 1 & -\tilde{k}(s,1) & -\tilde{k}(s+2,2) \\ 0 & 1 & -\tilde{k}(s+1,1) \\ 0 & 0 & 1 \end{bmatrix} \tag{3.31}$$

Também neste caso não há necessidade de armazenar as matrizes $\mathbf{L}_{gr}$, $\mathbf{U}_{gr}$. Nem mesmo a matriz $\tilde{\mathbf{A}}_{gr}$ será armazenada como já foi descrito no produto matriz-vetor por *superedge3*. Portanto, no momento em que for necessário executar operações do tipo $\mathbf{L}_{gr}^{-1}\mathbf{v}_{gr}$ ou do tipo $\mathbf{U}_{gr}^{-1}\mathbf{v}_{gr}$, a retrosubstituição ou a substituição devem ser executadas com os coeficientes $\tilde{\mathbf{A}}_s$, $\tilde{\mathbf{A}}_{s+1}$, $\tilde{\mathbf{A}}_{s+2}$ nas posições correspondentes de $\tilde{\mathbf{A}}_{gr}$.

Pré-condicionador *LU* aresta-por-aresta

Define-se para este caso a decomposição incompleta LU de cada matriz $\tilde{\mathbf{A}}_s$:

$$\mathbf{L}_s\mathbf{U}_s = \tilde{\mathbf{A}}_s - diag\tilde{\mathbf{A}}_s + \mathbf{I} \tag{3.32}$$

sendo os coeficientes de $\mathbf{L}_s$ e $\mathbf{U}_s$ obtidos através de operações elementares sobre as linhas de $\tilde{\mathbf{A}}_s - diag\tilde{\mathbf{A}}_s + \mathbf{I}$. Neste caso, tem-se a desvantagem de ser necessário armazenar os coeficientes da decomposição em cada aresta s, para tal define-se a matriz:

$$\mathbf{B}_s = \begin{bmatrix} b(s,1,1) & b(s,1,2) \\ b(s,2,1) & b(s,2,2) \end{bmatrix} \tag{3.33}$$

onde as operações necessárias para a decomposição são:

$$b(s,1,1) = u(s,1,1) = l(s,1,1) = 1.0$$

$$b(s,2,1) = l(s,2,1) = -\tilde{k}(s,2)$$

$$b(s,1,2) = u(s,1,2) = -\tilde{k}(s,1)$$

$$b(s,2,2) = u(s,2,2) = l(s,2,2) = \sqrt{1.0 - \tilde{k}(s,2)\tilde{k}(s,1)}$$

sendo que:

$$\tilde{\mathbf{L}}_s = \begin{bmatrix} b(s,1,1) & 0 \\ b(s,2,1) & b(s,2,2) \end{bmatrix} \tag{3.34}$$

$$\tilde{\mathbf{U}}_s = \begin{bmatrix} b(s,1,1) & b(s,1,2) \\ 0 & b(s,2,2) \end{bmatrix} \tag{3.35}$$

Pré-condiconador *LU superedge*3

Considerando a matriz $\tilde{\mathbf{A}}_{gr}$ descrita em (3.29) tem-se:

$$\mathbf{L}_{gr}\mathbf{U}_{gr} = \tilde{\mathbf{A}}_{gr} - diag\tilde{\mathbf{A}}_s + \mathbf{I} \tag{3.36}$$

sendo que as matrizes $\mathbf{L}_{gr}$ e $\mathbf{U}_{gr}$ são obtidas através de operações elementares. A matriz $\tilde{\mathbf{A}}_{gr} - diag\tilde{\mathbf{A}}_{gr} + \mathbf{I}$ não necessita ser armazenada, mas as matrizes $\mathbf{L}_{gr}$ e $\mathbf{U}_{gr}$ são armazenadas na matriz de decomposição:

$$\mathbf{B}_{gr} = \begin{bmatrix} b(gr,1,1) & b(gr,1,2) & b(gr,1,3) \\ b(gr,2,1) & b(gr,2,2) & b(gr,2,3) \\ b(gr,3,1) & b(gr,3,2) & b(gr,3,3) \end{bmatrix} \tag{3.37}$$

definida de forma similar ao caso aresta-por-aresta, ou seja, para $1 \leq i, j \leq 3$ tem-se:

$$\begin{aligned} & u(gr,1,1) = l(gr,1,1) = b(gr,1,1) \\ & u(gr,i,j) = b(gr,i,j) \quad \text{para} \quad i < j \\ & l(gr,i,j) = b(gr,i,j) \quad \text{para} \quad i > j \end{aligned} \tag{3.38}$$

3.4.3 Pré-condicionador Diagonal

Considera-se ainda o pré-condicionador diagonal à esquerda, que consiste em pré-multiplicar a inversa da diagonal montada da matriz dos coeficientes ao sistema (3.1). Assim, obtem-se:

$$\bar{\mathbf{A}}\mathbf{x} = \bar{\mathbf{b}} \tag{3.39}$$

onde $\bar{\mathbf{A}} = (diag\mathbf{A})^{-1}\mathbf{A}$ e $\bar{\mathbf{b}} = (diag\mathbf{A})^{-1}\mathbf{b}$. A ação do pré-condicionador localiza-se na operação de multiplicação matriz-vetor $\mathbf{v} = \mathbf{Ap}$, que neste caso apresenta os passos:

$$\begin{aligned} \mathbf{v} & \leftarrow \mathbf{Ap} \\ \mathbf{v} & \leftarrow (diag\mathbf{A})^{-1}\mathbf{v} \end{aligned} \tag{3.40}$$

Com certeza, este pré-condicionador é menos eficaz que os demais já descritos. Porém, possui a particularidade de ser o mesmo para os três tipos de estrutura de dados elemento-por-elemento, aresta-por-aresta e *superedge3*. Este fato ocorre, pois a operação (3.40) é executada com a diagonal assemblada, sendo portanto, ideal para efetuar comparações entre as estruturas de dados empregadas.

Capítulo 4

Experimentos Numéricos

Neste capítulo descreve-se o desempenho dos pré-condicionadores definidos no capítulo 3 com ênfase nos aspectos de vetorização e paralelismo. A maioria dos exemplos apresentados foram executados no computador CRAY J90. Os parâmetros básicos para as medidas de desempenho nesta máquina são o número de operações de ponto flutuante por segundo (*flop/s*) e o ganho em tempo de CPU (*speed-up*). São usadas as ferramentas de medida de desempenho disponíveis no CRAY J90. Para uma CPU, onde a vetorização é o objetivo, utiliza-se o *hardware performance monitor (hpm)* e o *profiling tool (perfview)*. Para os exemplos em paralelo são usadas 4 CPU's, com extrapolações para 16 CPU's realizadas pela ferramenta *atexpert*.

4.1 Convecção dominante em um domínio quadrado

Este exemplo trata do estudo da convecção pura de um escalar em um domínio, $\Omega = [0,1] \times [0,1]$, conforme a Figura 4.1.

Com condições de contorno:

$$\begin{cases} y = 0.0 & \Rightarrow \quad u = 0.0 \\ x = 0.0 \text{ e } 0.0 < y < 0.25 & \Rightarrow \quad u = 0.0 \\ x = 0.0 \text{ e } 0.25 < y < 1.0 & \Rightarrow \quad u = 1.0 \end{cases} \tag{4.1}$$

O campo de velocidade do fluido $\boldsymbol{\beta}$, faz um ângulo de 45° com o eixo x, e tem módulo unitário, $\|\boldsymbol{\beta}\| = 1$. O domínio Ω é discretizado em 4 malhas de elementos triangulares, respectivamente com 64×64, 128×128, 256×256 e 512×512

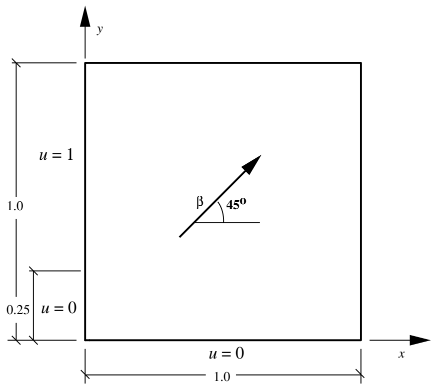


Figura 4.1: convecção pura de um domínio quadrado

células, sendo cada célula subdividida em 4 elementos. O número de arestas resultantes para cada discretização é respectivamente 24704, 98560, 393728 e 1573888. A percentagem de super-arestas para este tipo de malha é pequena, sendo $50,07\%$ de super-arestas e $49,93\%$ de arestas simples. Entretanto, deve-se salientar que não se procurou otimizar o número de super-arestas. Porém, para casos de malhas não estruturadas pode-se obter melhores resultados, como será visto na seção 4.3. São considerados 5 vetores na base de Krylov e tolerância de 10^{-6} para o método GMRES.

Na Tabela 4.1 observa-se a comparação do tempo de CPU necessário para convergência do Pré-condicionador Diagonal usando as estruturas de dados: elemento, aresta e *superedge3*. O tempo de CPU para a estratégia aresta, é menor quando comparado com a estratégia elemento, sendo que a estrutura de dados *superedge3* apresenta o menor tempo dentre as três estratégias. Este bom desempenho fica mais evidente à medida que a malha fica mais refinada. Observa-se ainda, que a taxa de operações de ponto flutuante (*flop/s*) na estrutura de dados por aresta diminui quando comparada com a estratégia elemento. Porém, tem-se uma recuperação no caso *superedge3*.

Nas Tabelas 4.2 e 4.3 estão respectivamente os resultados com os pré-condicionadores Gauss-Seidel à Direita e Esquerda e Gauss-Seidel à Esquerda, para as três estruturas de dados. Nota-se que o tempo necessário para resolver o sistema linear é grande quando comparado com o tempo total, ou seja, o tempo necessário para montagem de qualquer das estrutura de dados não é significativo. Com relação ao tempo de CPU, nota-se que as estratégias baseadas nas arestas apresentam menor tempo à medida que a malha fica mais refinada, sendo o tempo do *superedge3* menor. Observa-se ainda, que na maioria dos casos o número de iterações é maior para o pré-condicionador *superedge3*. A taxa de *Mflop* ainda é melhor para a estratégia elemento, porém há uma recuperação para o caso do *superedge3* comparado com o caso aresta.

malha	*tempo total*	*GMRES(5)*	*iterações*	*Mflop/s*
Elemento por Elemento				
64x64	2.918	2.266	345	61.8
128x128	18.592	16.801	702	77.1
256x256	135.069	128.771	1387	83.1
512x512	1019.196	994.946	2696	85.4
Aresta por Aresta				
64x64	3.521	2.026	345	34.6
128x128	18.505	14.678	702	51.3
256x256	124.070	111.057	1389	59.4
512x512	905.612	855.638	2701	62.8
Superedge3				
64x64	3.138	1.921	344	40.6
128x128	16.493	13.835	704	60.4
256x256	114.417	106.022	1391	67.5
512x512	840.093	809.110	2702	70.8

Tabela 4.1: Desempenho do Pré-Condicionador Diagonal

As Figuras 4.2 e 4.3 mostram o *speed-up* paralelo dos pré-condicionadores Gauss-Seidel estudados. Nos gráficos aparecem o caso da estrutura de dados *superedge3*, mas o mesmo comportamento acontece no caso elemento e aresta simples. Como já mencionado, são usadas 4 CPU's, sendo que os gráficos mostram o resultado da extrapolação para 16 CPU's. As principais rotinas que atuam nos pré-condicionadores são multiplicação matriz-vetor (*matvec*), retrosubstituição (*bkds*) e substituição (*fwds*). Seus desempenhos paralelos são comparados com o caso ideal, representado

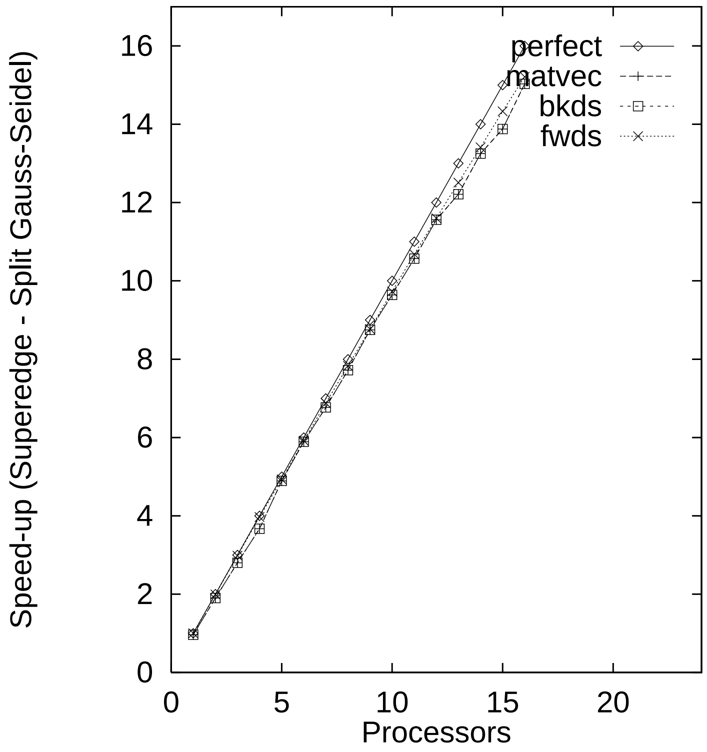


Figura 4.2: Desempenho Paralelo do Pré-condicionador Gauss-Seidel à Direita e Esqueda

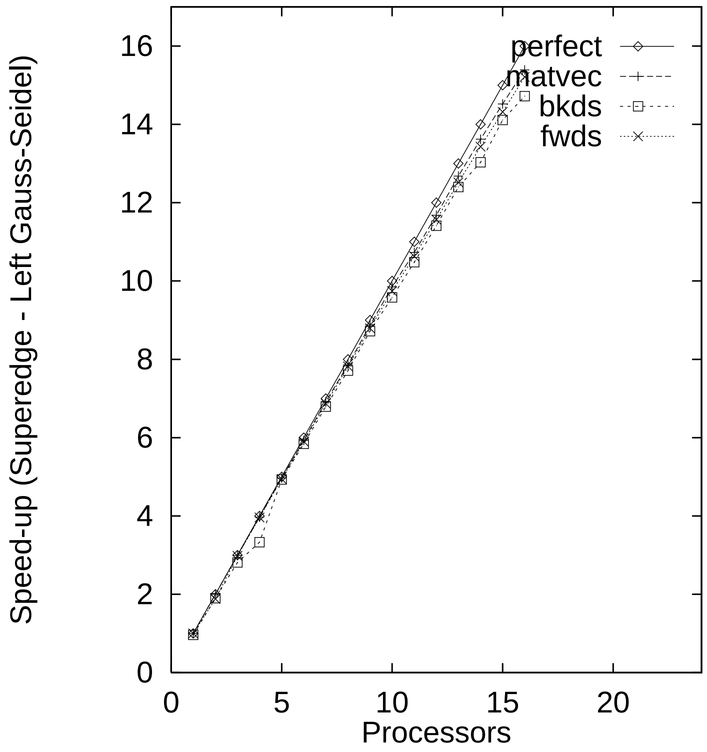


Figura 4.3: Desempenho Paralelo do Pré-condicionador Gauss-Seidel à Esqueda

malha	*tempo total*	*GMRES(5)*	*iterações*	*Mflop/s*
Elemento por Elemento				
64x64	2.045	1.392	127	49.0
128x128	11.378	9.589	231	61.9
256x256	81.513	75.212	465	68.4
512x512	651.520	627.269	976	71.3
Aresta por Aresta				
64x64	2.698	1.201	132	22.5
128x128	11.824	7.999	245	35.4
256x256	75.602	62.506	496	43.1
512x512	558.375	507.928	1017	46.9
Superedge3				
64x64	2.344	1.123	131	27.0
128x128	10.143	7.454	244	42.8
256x256	68.675	60.202	512	50.8
512x512	518.702	487.595	1050	54.1

Tabela 4.2: Desempenho do Pré-Condicionador Gauss-Seidel à Direita e Esquerda

pela reta *perfect*. Observa-se que estas rotinas são praticamente 100% paralelas.

O desempenho do pré-condicionador LU à Esquerda com relação ao tempo de CPU encontra-se na Tabela 4.4. Todas as observações feitas para os pré-condicionadores do tipo Gauss-Seidel são válidas para esse caso. Porém, nota-se que o pré-condicionador LU apresenta tempo de CPU menor em todos os tipos de estrutura de dados (elemento, aresta e super-aresta), quando comparados com os pré-condicionadores do tipo Gauss-Seidel. Com relação a taxa de *Mflop* observa-se

malha	*tempo total*	*GMRES(5)*	*iterações*	*Mflop/s*
Elemento por Elemento				
64x64	1.868	1.217	111	47.3
128x128	10.090	8.282	200	60.6
256x256	70.127	63.687	405	69.3
512x512	551.338	527.185	838	72.4
Aresta por Aresta				
64x64	2.662	1.146	127	22.1
128x128	11.352	7.440	233	35.2
256x256	71.343	57.876	467	43.1
512x512	525.376	474.114	964	47.3
Superedge3				
64x64	2.259	1.041	127	27.2
128x128	9.684	6.990	236	43.5
256x256	64.341	55.967	489	51.8
512x512	479.867	448.243	991	55.2

Tabela 4.3: Desempenho do Pré-Condicionador Gauss-Seidel à Esquerda

que há uma recuperação mais acentuada para as super-arestas se comparado com os pré-condicionadores do tipo Gauss-Seidel.

A Figura 4.4 mostra o desempenho paralelo do pré-condionador LU à Esquerda. Nota-se um comportamento similar ao caso dos pré-condicionadores do tipo Gauss-Seidel, ou seja, as rotinas *matvec*, *bkds* e *fwds* apresentam praticamente 100% de paralelismo.

malha	*tempo total*	*GMRES(5)*	*iterações*	*Mflop/s*
Elemento por Elemento				
64x64	2.098	1.448	110	48.4
128x128	12.352	10.548	201	57.9
256x256	87.825	81.456	400	63.5
512x512	691.862	667.385	835	66.7
Aresta por Aresta				
64x64	2.149	1.272	129	31.8
128x128	12.493	8.517	238	37.4
256x256	78.801	65.362	478	45.8
512x512	592.441	540.680	983	49.1
Superedge3				
64x64	1.703	1.131	126	40.3
128x128	10.045	7.379	233	46.3
256x256	66.705	58.322	479	54.8
512x512	500.006	468.891	974	58.3

Tabela 4.4: Desempenho do Pré-Condicionador LU à Esquerda

Pode-se concluir então, que o pré-condicionador LU à Esquerda com estrutura de dados por super-aresta apresenta o melhor desempenho com relação à vetorização e paralelismo.

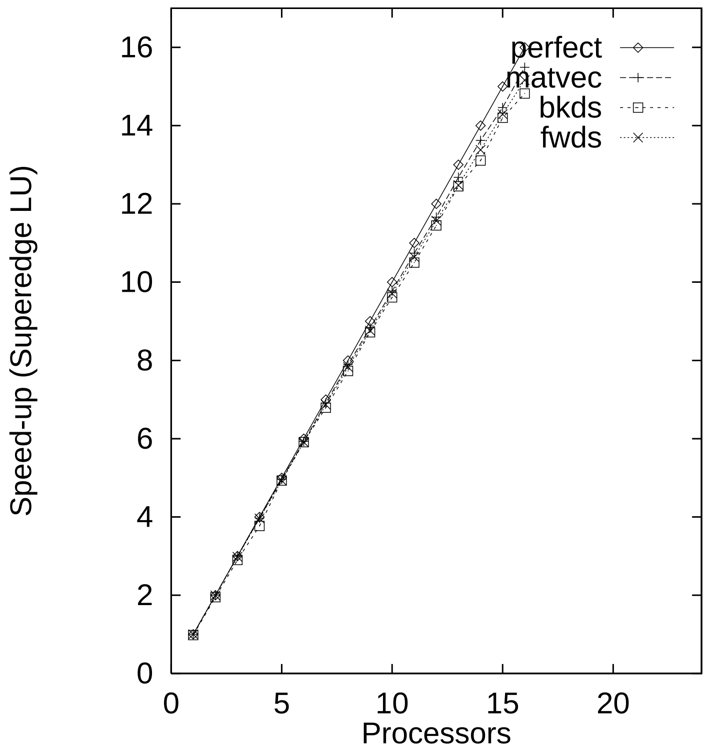


Figura 4.4: Desempenho Paralelo do Pré-condicionador LU à Esqueda

4.2 Convecção - Difusão em um domínio quadrado

Como no exemplo tratado na seção 4.1, tem-se um domínio $\Omega = [0,1] \times [0,1]$, conforme a Figura 4.1. O campo de velocidade também tem módulo unitário e faz ângulo de $45°$ com o eixo x, porém neste caso a difusividade não é desprezível, sendo:

$$\mathbf{k} = \begin{bmatrix} 1.0 & 0 \\ 0 & 1.0 \end{bmatrix}$$

em todos os pontos de Ω.

Para analisar o desempenho dos pré-condicionadores define-se uma discretização de Ω em 128×128 células, sendo cada célula subdividida em 4 elementos triangulares lineares. Também neste caso não foram feitos esforços para otimizar o número de super-arestas, mantendo-se as mesmas porcentagens do exemplo anterior. Considera-se 25 vetores na base de Krylov para o método GMRES e tolerância igual a 10^{-6}.

A Tabela 4.5 mostra os resultados referentes aos pré-condicionadores Diagonal, Gauss-Seidel à Esquerda e Direita (GS-EsqDir), Gauss-Seidel à Esquerda (GS-

Pré-condicionador	*tempo total*	*GMRES(25)*	*iterações*	*Mflop/s*
Elemento por Elemento				
Diagonal	99.409	97.612	2766	87.5
GS-EsqDir	70.849	69.051	1403	78.0
GS-Esquerda	50.402	48.620	1017	79.4
LU-Esquerda	59.547	57.761	1050	76.8
Aresta por Aresta				
Diagonal	104.496	100.559	3181	76.2
GS-EsqDir	55.506	51.562	1264	62.7
GS-Esquerda	48.829	44.899	1097	72.1
LU-Esquerda	57.955	54.027	1251	64.1
Superedge3				
Diagonal	73.321	70.659	2323	81.1
GS-EsqDir	46.455	43.765	1115	67.5
GS-Esquerda	57.647	54.982	1438	70.0
LU-Esquerda	59.402	56.724	1406	70.6

Tabela 4.5: Desempenho dos Pré-Condicionadores para a malha 128×128 considerando difusão

Esquerda) e LU à Esquerda (LU-Esquerda), para os três tipos de estrutura de dados abordada. Em linhas gerais, nota-se que, com relação ao tempo de CPU, as estratégias por aresta apresentam melhor desempenho que a estratégia por elemento. O pré-condicionador Diagonal via elemento necessita de um tempo de CPU sensivelmente menor que o pré-condicionador por aresta, porém o pré-condicionador diagonal por super-arestas apresenta o menor tempo dentre os três. O pré-condicionador Gauss-Seidel à Esquerda e Direita com o uso da estratégia por *Superedge3* apresenta o melhor desempenho com relação ao tempo de CPU, dentre todos analisados. A taxa de *Mflop* apresenta resultados similares aos já relatados no exemplo anterior, ou seja, ainda é melhor para a estratégia elemento, porém há uma recuperação para o caso de super-arestas comparado com o caso aresta.

4.3 Dispersão de Poluente na Baía de Guanabara

Neste último exemplo, considera-se um passo de tempo da simulação da dispersão de poluentes na baía de Guanabara. É um problema com convecção e difusão, sendo a velocidade e a difusividade conhecida em todos os elementos da malha. A Figura 4.5 mostra a malha não estruturada discretizda por 34317 elementos triagulares e 18481 nós, gerada empregando-se o algoritmo de Delaunay [23]. Tal malha possui, 52800 arestas, sendo possível agrupar 78, 82% em super-arestas. Para este exemplo, é usada a máquina SGI-Indy. A Tabela 4.6 mostra o desempenho dos pré-condicionadores diagonal e LU à Esquerda para as estratégias elemento e super-arestas, considerando 25 vetores na base de Krylov e tolerância de 10^{-6} para o método GMRES. Observa-se que a estratégia super-aresta apresenta melhor desempenho.

Neste problema, a memória necessária para armazenar os coeficientes do pré-condicionador LU à Esquerda, usando a estratégia de super-arestas, é de 124 *Mwords* e se fosse usada a estratégia arestas simples, seriam necessárias apenas 45 *Mwords*. Porém para o pré-condicionador LU à Esquerda utilizando a estratégia por elemento são necessárias 308 *Mwords* para armazenar os coeficientes. Observa-se portanto, que o esquema baseado nas arestas reduz a menos da metade a área de armazenamento do pré-condicionador LU à Esquerda.

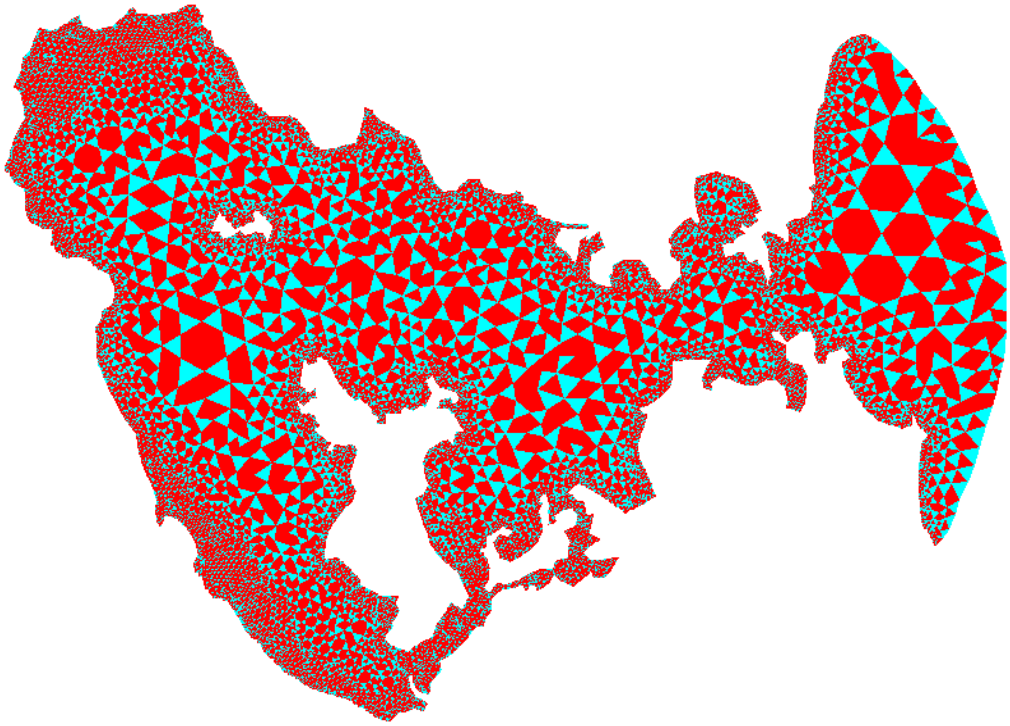


Figura 4.5: Discretização da Baía de Guanabara em *superedge3*

.

	Diagonal		LU	
	Iteração	Tempo(s)	Iteração	Tempo(s)
Elemento	312	81.4	80	36.2
Superedge3	311	68.4	89	29.2

Tabela 4.6: Desempenho dos Pré-Condicionadores Diagonal e LU para a dispersão de poluentes na Baia de Guanabara

Capítulo 5

Conclusões

Neste texto foi apresentado um estudo de pré-condicionadores paralelos, usando estrutura de dados por aresta, para a solução de sistemas lineares não-simétricos, oriundos de formulações de elementos finitos. Estes pré-condicionadores são globalmente definidos, pois existe uma correspondência direta entre os coeficientes das arestas e os coeficientes da matriz esparsa global.

Os resultados obtidos com o uso dos pré-condicionadores desenvolvidos são promissores. Para problemas bidimensionais com convecção dominante o pré-condicionador LU com super-aresta apresenta o melhor desempenho. Porém quando considera-se problemas com convecção e difusão a ação do pré-condicionador não é tão eficaz, sendo o pré-condicionador Gauss-Seidel à Esquerda e Direita com super-aresta sensivelmente melhor.

É importante ressaltar que, em todos os experimentos realizados as estratégias usando arestas apresentam melhor desempenho que a estratégia convencional por elemento, necessitando de menos memória.

Referências Bibliográficas

[1] T. Tezduyar.*Advanced flow simulation and modeling*, (to appear in), Lecture Notes of the Short Course on Finite Element Methods in Flow Simulation, Tokyo, Japan , (1998).

[2] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing, Boston, 1996.

[3] T.J.R. Hughes Z. Johan and F. Shakib, *A globally convergent matrix-free algorithm for implicit time marching schemes arising in finite element analisys of fluids*, Comput. Methods Appl. Mech. Engrg., 87, 281–304 (1991).

[4] F. Shakib, T. J. R. Hughes and Z. Johan, *A multi-element group preconditioned GMRES algorithm for nonsymmetric systems arising in finite element analysis*, Comp. Meth. Appl. Mech and Engng., 65 (1989), pp. 415-456.

[5] T. Tezduyar and J. Liou, *Grouped element-by-element iteration schemes for incompressible flow simulations*, Computer Physics Communications, 53, pp. 441–453 (1989).

[6] J. Liou and T. Tezduyar, *Clustered element-by-element computations for fluid flow* (in Horst D. Simon, editor), Parallel Computational Fluid Dynamics–Implementation and Results, Chapter 9, The MIT Press, Cambridge, Massachussetts, pages 167–187 (1992).

[7] T. Tezduyar M. Behr S.K. Aliabadi S. Mittal and S. Ray, *A new mixed preconditioning method for finite element computations*, Comput. Methods Appl. Mech. Engrg., 99, pp. 27–42 (1992).

[8] V. Kalro and T. Tezduyar, *Parallel iterative computational methods for 3D finite element flow simulations*, (to appear in) Computer Assisted Mechanics and Engineering, (1997).

[9] J. Peraire, K. Morgan and J. Peiro, *Unstructured grid methods for compressible flows*, Special Course on Unstructured Grid Methods for Advection Dominated Flows, 787, 5.1–5.39 (1992).

[10] J.D. Baum, H. Luo and R. Lohner, *Edge–based finite element scheme for Euler equations*, AIAA J., 32, 1183–1190 (1994).

[11] P.R.M. Lyra, J. Peraire, K. Morgan and J. Peiro, *TVD algorithms for the solution of the compressible Euler equations on unstructured meshes*, Int. J. Num. Meth. Fluids, 19, 849–863 (1994).

[12] R. Lohner, *Edges, stars, superedges and chains*, Comp. Meth. Appl. Mech and Engng., 111 (1994), pp. 255-263.

[13] P.R.M. Lyra, R.B. Willmersdorf, M.A.D. Martins, and A.L.G.A. Coutinho, *Parallel implementation of edge-based finite element schemes for compressible flows on unstructured meshes*, (submitted to VECPAR'98), Porto, Portugal, (1998).

[14] V. Venkatakrishnan and D. J. Mavriplis, *Implicit solvers on ustructured meshes*, J. Comp. Physics, 105, 83–91 (1993).

[15] M.A.D. Martins, A.L.G.A. Coutinho and J.L.D Alves, *Parallel iterative solution of finite element systems of equations employing edge-based data structures*, SIAM Conf. on Parallel Processing for Scientific Computing vCD-ROM, (1997).

[16] A. N. Brooks and T. J. R. Hughes, *Streamline Upwind/Petrov-Galerkin Formulations for Convections Dominated Flows with Particular Emphasis on the Incompressible Navier-Stokes equations*, Comp. Meth. Appl. Mech. and Engng., 32, pp. 199-259 (1982).

[17] O. C. Zienkiewicz and R. L. Taylor, *The Finite Element Method, Volume 2 - Solid and Fluid Mechanics Dynamics and Non-Linearity*, Fourth Edition, McGraw-Hill Book Company, London, 1991.

[18] R. Codina, E. Oñate and M. Cervera, *The Intrinsic Time for the Streamline Upwind/Petrov-Galerkin Formulation Using Quadratic Elements*, Comp. Meth. Appl. Mech. and Engng., 94, pp. 239-262,1992.

[19] T. J. R. Hughes, *The Finite Element Method - Linear Static and Dynamic Finite Element Analysis*, Prentice-Hall, Inc., 1987.

[20] M. A. D. Martins, *Solução Iterativa em Paralelo de Sistemas de Equações do Método dos Elementos Finitos Empregando Estrutura de Dados por Arestas*, Tese de Mestrado, COPPE/CIVIL, 1996.

[21] S.-H. Hsieh G. H. Paulino and J. F. Abel, *Recursive Spectral Algotithms for Automatic Domain Partitioning in Parallel Finite Element Analysis*, Comp. Meth. Appl. Mech. and Engng., 121 (1995), pp. 137-162.

[22] Hughes, T. J. R., Ferencz, R. and Hallquist, J. O., *Large-scale vectorized implicit calculations in solid mechanics on a CRAY X-MP/48 utilizing EBE preconditioned conjugate gradients*, Comp. Meth. Appl. Mech and Engng., 61 (1987), pp. 21-248.

[23] Ribeiro, F. L. B., Landau, L., Conceição, A. M., *Generation of Unstructured Meshes for Numerical Simulation of Oil Reservoirs*, Proc. 9th Int. Symposium on Offshore Engineering, Edt. by F. L. L. Carneiro *et al*, Wiley and Sons, pp.529-530, 1995.