**Pré-condicionadores Multinível para
Equações Elípticas de Segunda Ordem
Discretizadas por Elementos Finitos**

Leonardo J. Borges

DISSERTAÇÃO APRESENTADA
AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
DA
UNIVERSIDADE DE SÃO PAULO
PARA OBTENÇÃO DO GRAU DE MESTRE
EM
MATEMÁTICA APLICADA

Orientador: Prof. Dr. Saulo R. M. Barros

Durante a elaboração deste trabalho, o autor recebeu apoio financeiro do CNPq

-São Paulo, maio de 1994-

aos meus pais

Silvio e Filomena

Agradecimentos

Aos colegas, professores e funcionários: amigos que muito ajudaram e incentivaram no decorrer do meu mestrado.

Ao orientador **Saulo**: pela dedicação, estímulo, confiança e infinita paciência. E ao amigo **Saulo**: pelas longas conversas que rolaram e pela força durante todo o mestrado.

**Pré-condicionadores Multinível para
Equações Elípticas de Segunda Ordem
Discretizadas por Elementos Finitos**

Este exemplar corresponde à redação final
da dissertação apresentada por Leonardo José Borges,
devidamente corrigida e aprovada pela comissão julgadora.

São Paulo, junho de 1994.

Banca examinadora:

Prof. Dr. Saulo Rabello Maciel de Barros (orientador) - IME-USP

Prof. Dr. Marco Antonio Raupp - IME-USP

Prof. Dr. Jörg Friedrich Hubert Blatter - UFRJ

Conteúdo

Introdução	1
1 Preliminares	5
1.1 O método de Galerkin	5
1.2 O método dos Gradientes Conjugados	6
1.3 O método Multigrid	8
2 As Bases Hierárquicas	15
2.1 Interpolação e suas estimativas	15
2.2 Solução do problema variacional	20
2.3 Considerações e generalizações	26
2.4 O método HBMG	30
3 O Método BPX	35
3.1 Projeção e suas estimativas	35
3.2 Aplicação e considerações	42
3.3 Estimativas recentes	48
4 Métodos de Subestrutura ou Decomposição de Domínios	51
4.1 Formulação dos problemas e pré-condicionadores	51
5 O Algoritmo Híbrido	61
5.1 O algoritmo híbrido HBDD	62
5.2 O algoritmo MNBDD	67

6 Experimentos numéricos	71
6.1 Descrição do programa	71
6.2 Resultados numéricos e sua análise	73
Bibliografia	81
Apêndice A: Programas	85

Introdução

A busca de soluções eficientes para sistemas lineares provenientes da resolução numérica de equações diferenciais parciais elípticas discretizadas por elementos finitos é tema de inúmeros trabalhos na literatura. As matrizes obtidas nas discretizações por elementos finitos são comumente muito grandes e esparsas, tornando atraente o uso de métodos iterativos na resolução dos sistemas resultantes. Devido à sua generalidade e facilidade de implementação o *método dos gradientes conjugados* (CG) é um dos mais usados para essa tarefa. No entanto, embora as matrizes sejam esparsas, seu número de condição cresce quadraticamente (ordem $\mathcal{O}(h^{-2})$) com o refinamento do espaçamento h da malha utilizada. Como a velocidade de convergência do método dos gradientes conjugados depende fortemente do número de condição da matriz envolvida, a convergência torna-se extremamente lenta para partições finas. O uso de pré-condicionadores para os sistemas visa remediar este problema. Várias possibilidades foram estudadas na literatura, envolvendo o uso de escalonamento diagonal (Jacobi), symmetric successive overrelaxation (SSOR), decomposição incompleta LU, etc, melhorando o número de condição para $\mathcal{O}(h^{-1})$ (Axelsson e Barker [1], cap. 7).

Mais recentemente, vários pré-condicionadores baseados em estruturas multinível ou em decomposição de domínios foram desenvolvidos, levando a significativas melhoras nos números de condição obtidos. Mais ainda, devido ao grande paralelismo inerente em suas estruturas, tais métodos tornaram-se extremamente atraentes com o advento da computação paralela.

O objetivo deste trabalho é apresentar as técnicas e as idéias presentes nos pré-condicionadores com estruturas multinível, incluindo parte do ferramental utilizado para obtenção de estimativas para seus números de condição. Vários destes métodos apresentam pontos em comum com o método multigrid, que é portanto descrito brevemente no capítulo introdutório, juntamente com as idéias básicas sobre pré-condicionamento do método dos gradientes conjugados.

O primeiro pré-condicionador que apresentamos baseia-se na idéia de *bases hierárquicas* (HB), desenvolvidas por Yserentant [49, 50, 51]. Tais bases surgem naturalmente na construção de elementos finitos quando obtem-se a triangularização final a partir de sucessivos refinamentos sobre uma triangularização inicial mais grossa. Além disso, a representação de uma função elemento finito pertencente à base hierárquica pode ser rapidamente convertida para sua representação em termos da base nodal, o que permite uma implementação simples e eficiente do algoritmo. Como o método HB é definido em termos de malhas, e não em termos dos operadores elípticos, a performance do pré-condicionador pode variar muito em função do operador envolvido. Uma forma de incorporar informações do operador elíptico foi mostrada por Bank, Dupont e Yserentant [4] através do *método multigrid-bases hierárquicas* (HBMG). Xu [48] obtém, para o pré-condicionamento por bases hierárquicas, números de condição com ordem $\mathcal{O}(1)$ para o caso unidimensional, $\mathcal{O}(\ln^2 h^{-1})$ para duas dimensões e $\mathcal{O}(h^{2-d})$ para dimensão $d \geq 3$. Como se vê, o método não é muito eficiente para problemas tri-dimensionais.

Buscando algoritmos também eficientes em dimensões maiores, Bramble, Pasciak e Xu [9] desenvolveram o método BPX (ou *método das bases multinível nodais*), ao qual o terceiro capítulo é dedicado. A estrutura do método é muito próxima a um V-ciclo do método multigrid, mas os operadores de suavização são substituídos por um fator de escala. Utilizando hipóteses comuns à literatura multigrid, Bramble, Pasciak e Xu derivaram estimativas para o número condição da ordem de $\mathcal{O}(\ln^2 h^{-1})$, independentemente da dimensão do problema. Apresentamos ainda um outro caminho para a obtenção destas estimativas, através da interpretação do método BPX como um método aditivo de Schwarz, como sugerido por Dryja e Widlund [24]. Recentemente, utilizando técnicas de teoria da aproximação Oswald [36, 37], Dahmen e Kunoth [18], Bramble e Pasciak [11], Bornemann e Yserentant [7] e Zhang [52], conseguiram obter estimativas de optimalidade ($\mathcal{O}(1)$) para o método BPX. Tais estimativas não serão vistas em detalhe neste trabalho, nos limitamos a indicar os caminhos para obtê-las.

O capítulo seguinte apresenta a idéia de decomposição de domínios. Tais métodos são baseados no fato de que o domínio do problema original pode ser tomado como uma união de dois ou mais subdomínios, nos quais a restrição do operador original pode ter uma forma conveniente. Além disso, proporcionam um caminho para a paralelização; domínios de forma irregular podem ser decompostos em subdomínios de forma regular; e regiões onde o operador requeira algum tratamento diferenciado podem ser isoladas. Métodos como os considerados por Bjorstad e Widlund [6], Bramble, Pasciak e Schatz [12, 13, 14, 15, 16], Dryja e Widlund [23], Mandel [31, 32], baseiam-se no uso do complemento de Schur, isto é, o conjunto de variáveis é dividido em dois subconjuntos: pontos interiores a subdomínios e pontos pertencentes a arestas entre subdomínios. Os problemas a serem resolvidos sobre o interior dos subdomínios são problemas de Dirichlet ou de Neumann independentes. O acoplamento das equações é refletido no complemento de Schur, formado pelas variáveis pertencentes às arestas dos subdomínios. Tais algoritmos tem seu número de condição estimado em $\mathcal{O}((1 + \ln \frac{H}{h})^2)$ (onde H representa o diâmetro dos subdomínios), como em Bjorstad e Widlund [6].

Os métodos apresentados nos capítulos 2, 3 e 4 podem ser combinados, gerando algoritmos denominados híbridos, enfocados no capítulo quinto. Smith e Widlund [43] desenvolveram o método híbrido *bases hierárquicas em decomposição de domínios* (HBDD). Na decomposição do domínio, consideram uma mudança para bases hierárquicas sobre cada aresta formada. Demonstram que o método possui um número de condição menor ou igual ao número apresentado pelas bases hierárquicas, logo menor que a estimativa $\mathcal{O}((1 + \ln \frac{H}{h})^2)$ citada acima (em dimensão dois). Uma grande diferença desse método em relação aos anteriores, como BPS (ou DD1) de Bramble, Pasciak e Schatz [13], consiste na utilização de mudança de base para obtenção de um desacoplamento dos problemas envolvendo cada aresta, aumentando o paralelismo do método. Inspirados no trabalho de Smith e Widlund, Tong, Chan e Kuo [45] desenvolveram o *método bases multinível nodais em decomposição de domínios* (MNBDD) muito similar, onde a mudança de base adotada sobre as arestas corresponde ao método BPX. E, analogamente, o método possui um número de condição menor ou igual ao apresentado pelo método BPX.

Ao longo deste trabalho procuramos ressaltar as várias ligações entre as diferentes

técnicas e ainda, sempre que possível, indicamos fontes na literatura onde as idéias são mais desenvolvidas ou enfocadas de maneira diferente. Assim, acreditamos que este texto sirva como uma boa introdução a este campo de pesquisa.

Não menos importantes que as idéias e a teoria na qual se baseiam os pré-condicionadores multinível, consideramos os aspectos práticos envolvidos em sua implementação. Procuramos apresentar em detalhe os algoritmos, de forma a facilitar a implementação dos mesmos em caso de interesse. Como ilustração, e também para permitir uma análise da performance dos métodos, implementamos os algoritmos das bases hierárquicas e BPX. O último capítulo é dedicado à apresentação e discussão dos resultados numéricos obtidos na solução de uma equação de Poisson. Os programas utilizados estão disponíveis no apêndice.

de uma perspectiva crítica, a análise da literatura brasileira contemporânea revela uma complexa interação entre a tradição e a inovação, a identidade e a diversidade. Este trabalho busca explorar as nuances dessas relações, examinando a produção literária de autores que desafiam as fronteiras estabelecidas e que refletem a pluralidade da sociedade brasileira. Através de uma abordagem interdisciplinar, pretende-se compreender como a literatura atua como um espaço de resistência e transformação social, contribuindo para a construção de uma memória coletiva e para a afirmação de valores democráticos. A análise será realizada sob a luz de teorias literárias e culturais contemporâneas, buscando estabelecer conexões entre o texto literário e o contexto histórico e social em que foi produzido.

1 Preliminares

Neste capítulo introdutório temos uma revisão da formulação variacional para uma equação diferencial parcial elíptica, do método dos gradientes conjugados e também do método multigrid.

1.1 O método de Galerkin

Considerando-se a formulação variacional

$\Omega \subset \mathbb{R}^2$ domínio poligonal limitado ,
 $V = V(\Omega)$ espaço de Hilbert ,
 $a: V \times V \rightarrow \mathbb{R}$ forma bilinear, contínua, V-elíptica (isto é, $\exists c > 0 : a(v, v) \geq c\langle v, v \rangle, \forall v \in V$) ,
 $f: V \rightarrow \mathbb{R}$ funcional linear contínuo .
Determinar $u \in V$ tal que $a(u, v) = f(v), \forall v \in V$.

O teorema de Lax-Milgram (vide Ciarlet [17]) garante existência e unicidade da solução u . Para a resolução da formulação variacional em dimensão finita, temos o *método de Galerkin*:

Seja $V^h \subset V$, subespaço de dimensão finita n .
Determinar $u_h \in V^h$ tal que

$$a(u_h, v_h) = f(v_h), \forall v_h \in V^h \quad (1.1)$$

Visando a obtenção de u^h , seja $\{v_1, \dots, v_n\}$ base de V^h . O problema (1.1) equivale a determinar

$$u_h \in V^h \text{ tal que } a(u_h, v_i) = f(v_i), \forall i = 1, \dots, n .$$

Logo, para $u_h = \sum_{j=1}^n c_j v_j$, temos $a(\sum_{j=1}^n c_j v_j, v_i) = f(v_i) \Rightarrow \sum_{j=1}^n c_j a(v_j, v_i) = f(v_i), i = 1, \dots, n$, o que implica na resolução do sistema linear

$$\begin{pmatrix} a(v_1, v_1) & \cdots & a(v_n, v_1) \\ \vdots & \ddots & \vdots \\ a(v_1, v_n) & \cdots & a(v_n, v_n) \end{pmatrix} \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} f(v_1) \\ \vdots \\ f(v_n) \end{pmatrix} .$$

Observemos que $a(\cdot, \cdot)$ V-elíptica implica em um sistema não singular.

Para a escolha do subespaço V^h , comumente é adotado um processo de refinamento por triangularizações quase-uniformes (indicadas em Ciarlet [17] como regulares). Uma triangularização $\mathcal{T} = \{\cup_i T_i\}$ de um domínio poligonal limitado $\Omega \subset \mathbb{R}^2$ consiste em um conjunto de triângulos $T_i \in \overline{\Omega}$ tais que $\cup_i T_i = \overline{\Omega}$ e a intersecção de dois triângulos distintos resulta em um lado comum, ou um vértice comum, ou é vazia.

Vários métodos multinível são definidos a partir de sucessivos refinamentos aplicados sobre uma triangularização inicial:

$\mathcal{T}_0$ triangularização inicial (malha mais grossa) ,

$\mathcal{T}_1, \dots, \mathcal{T}_J$ triangularizações aninhadas onde um triângulo de $\mathcal{T}_{k+1}$ é um triângulo de $\mathcal{T}_k$ ou um refinamento quase-uniforme de um triângulo de $\mathcal{T}_k$.

Associada ao processo de refinamento acima, temos a seguinte notação

$$\begin{aligned} x_i, i = n_{k-1} + 1, \dots, n_k & \text{ são os vértices criados no refinamento } \mathcal{T}_k \rightarrow \mathcal{T}_{k+1} , \\ \mathcal{J}_k & = \{i : i = 1, \dots, n_k\} \text{ conjunto dos índices referentes a vértices em } \mathcal{T}_k , \\ \mathcal{N}_k & = \{i : i \in \mathcal{J}_k\} \text{ conjunto dos vértices em } \mathcal{T}_k , \\ \mathcal{S}_k & = \{f \in \mathcal{C}^0(\bar{\Omega}) : f \text{ é linear nos triângulos } T \in \mathcal{T}_k \text{ e } f|_{\partial\Omega} = 0\} . \end{aligned}$$

Adota-se $V^h = \mathcal{S}_J \supset \mathcal{S}_{J-1} \supset \dots \supset \mathcal{S}_0$ como o espaço para a formulação de Galerkin.

1.2 O método dos Gradientes Conjugados

Entre os métodos iterativos utilizados na resolução de um sistema linear da forma

$$Au = f , \quad (1.2)$$

temos o *método dos gradientes conjugados*, que é baseado no fato de que, sendo a matriz A simétrica positiva definida, o problema de determinar uma solução para (1.2) é equivalente ao problema de minimizar a forma quadrática

$$F(u) = \frac{1}{2} \langle u, Au \rangle - \langle f, u \rangle . \quad (1.3)$$

O *método steepest descent* utiliza o princípio de minimização acima: o vetor resíduo $r^k = f - Au^k$, que é o negativo do gradiente de $F(u^k)$, define a direção de maior decaimento para a forma (1.3). O método caminha na direção de decaimento através da iteração

$$u^{k+1} = u^k + \alpha_k r^k$$

tomando-se α_k tal que minimize $F(u^k + \alpha_k r^k)$, isto é,

$$\alpha_k = \frac{\langle r^k, r^k \rangle}{\langle r^k, Ar^k \rangle} .$$

Infelizmente, o método steepest descent converge muito lentamente se o número de condição de A é grande.

O método dos gradientes conjugados segue o princípio do método steepest descent pois, dado u^k , toma o vetor direção p^k e a iteração

$$u^{k+1} = u^k + \alpha_k p^k .$$

Entretanto, $p^k = r^k + \beta_k p^{k-1}$, onde β_k é tomado tal que p^k seja o *conjugado* de p^{k-1} , isto é, $\langle p^k, Ap^{k-1} \rangle = 0$. Determinados β_k e p^k , toma-se α_k tal que minimize $F(u^k + \alpha_k p^k)$. Dessa forma, obtem-se

Algoritmo CG

```

escolha  $u^0$ 
 $r^0 = f - Au^0$ 
 $p^0 = r^0$ 
para  $k = 0, \dots, k_{max}$ 
     $\alpha = \langle r^k, r^k \rangle / \langle p^k, Ap^k \rangle$ 
     $u^{k+1} = u^k + \alpha p^k$ 
    teste de convergência
     $r^{k+1} = r^k - \alpha Ap^k$ 
     $\beta = \langle r^{k+1}, r^{k+1} \rangle / \langle r^k, r^k \rangle$ 
     $p^{k+1} = r^{k+1} + \beta p^k$ 
próximo  $k$ 
```

O método dos gradientes conjugados atinge a solução $\bar{u}$ do sistema (1.2) após N iterações. onde N é a ordem de A . Mas isso não é relevante quando N é muito grande, pois seria necessário muito tempo de computação. Torna-se interessante termos uma estimativa sobre o erro após um dado número de iterações. Em Axelsson e Barker [1], sec 1.3, temos uma análise de $p(\varepsilon)$ correspondente ao número máximo de iterações necessárias para a redução do erro $e^k = \bar{u} - u^k$ em um fator $\varepsilon \in (0, 1)$, em relação à norma da energia $\|\cdot\|^2 = \langle \cdot, \cdot \rangle$, isto é,

$$\|e^k\| \leq \varepsilon \|e^0\|, \quad \forall e^0 \in V^h .$$

Para $\kappa = \kappa(A)$ número de condição de A , temos

$$p(\varepsilon) \leq \frac{1}{2} \sqrt{\kappa} \ln \left(\frac{2}{\varepsilon} \right) + 1 . \quad (1.4)$$

O pré-condicionamento da matriz A visa a obtenção de um sistema equivalente à (1.2), com número de condição menor.

Dada M uma matriz simétrica, positiva e definida, da forma $M = EE^T$ (E não singular), o método dos *gradientes conjugados pré-condicionado* (PCG) resolve o sistema

$$\tilde{A}\tilde{u} = \tilde{f} \quad (1.5)$$

onde

$$\begin{aligned} \tilde{A} &= E^{-1}AE^{-T} = E^T M^{-1}AE^{-T} , \\ \tilde{f} &= E^{-1}f , \\ \tilde{u} &= E^T u . \end{aligned}$$

Sendo A simétrica e positiva definida, temos o mesmo para $\hat{A}$. E a mudança

$$E^{-T} \hat{A} E^T = E^{-T} E^{-1} A = M^{-1} A$$

mostra que $\hat{A}$ e $M^{-1} A$ possuem o mesmo espectro. Assim, o número de condição $\kappa(\hat{A})$ está completamente determinado por M e A , mesmo $\hat{A}$ dependendo da particular fatorização de M .

O método pré-condicionado pode ser formulado em duas versões. A primeira produz vetores u^k que aproximam $A^{-1}f$, enquanto a segunda produz aproximações $\hat{u}^k$ para $\hat{A}^{-1}\hat{f}$, que devem ser transformadas para obtermos a aproximação de (1.2). Consideremos a primeira versão: aplicando o método dos gradientes conjugados sobre (1.5), obtemos

Algoritmo PCG

```

escolha  $u^0$ 
 $r^0 = f - Au^0$ 
 $\hat{r}^0 = M^{-1}r^0$ 
 $p^0 = \hat{r}^0$ 
para  $k = 0, \dots, k_{max}$ 
     $\alpha = \langle \hat{r}^k, r^k \rangle / \langle p^k, Ap^k \rangle$ 
     $u^{k+1} = u^k + \alpha p^k$ 
    teste de convergência
     $r^{k+1} = r^k - \alpha Ap^k$ 
     $\hat{r}^{k+1} = M^{-1}r^{k+1}$ 
     $\beta = \langle \hat{r}^{k+1}, r^{k+1} \rangle / \langle r^k, r^k \rangle$ 
     $p^{k+1} = \hat{r}^{k+1} + \beta p^k$ 
próximo  $k$ 
```

Analogamente à (1.4), obtemos

$$p(\varepsilon) \leq \frac{1}{2} \sqrt{\kappa(\hat{A})} \ln \left(\frac{2}{\varepsilon} \right) + 1. \quad (1.6)$$

Assim, se M possui a propriedade $\kappa(\hat{A}) < \kappa(A)$, espera-se que o método PCG convirja mais rapidamente que o método CG sem pré-condicionamento.

É importante observarmos que no algoritmo PCG temos a matriz de pré-condicionamento M , e não a fatorização em E . Logo, podemos adotar a forma $M = ED^{-1}E^T$ mais geral para a fatorização de M , lembrando que o número de condição depende apenas de M e A .

1.3 O método Multigrid

Na resolução de uma equação através de elementos finitos adota-se uma malha bastante fina, para se obter uma boa precisão na aproximação da solução algébrica da equação. Dessa

forma, a matriz gerada possui um número de condição muito grande. Isso pode provocar uma convergência lenta para os métodos iterativos. Adotando-se uma seqüência de refinamentos, formado por malhas cada vez mais finas, pode-se obter um algoritmo com um trabalho computacional muito menor. Esse é o intuito do *método multigrid* (vide Hackbusch [27], por exemplo).

Para um melhor entendimento, consideremos o caso unidimensional do problema de Dirichlet com a formulação de Galerkin

$$A^h u^h = f^h,$$

para elementos finitos associados à malha uniforme $\Omega^h = \{ih : i = 1, \dots, 2^N - 1\}$ de passo $h = \frac{1}{2^N}$ em $\Omega = [0, 1]$. Tomemos $\Omega^{2^J h} \subset \dots \subset \Omega^{4h} \subset \Omega^{2h} \subset \Omega^h$ um conjunto de malhas aninhadas onde $\Omega^{2^k h}$ é uma malha de passo $2^k h$, isto é, consideramos Ω^h como a malha mais fina obtida a partir de sucessivos refinamentos sobre a malha mais grossa $\Omega^{2^J h}$. Para adequar a notação, identifiquemos a malha $\Omega^{2^k h}$ por Ω^{J-k} , ou seja, o conjunto de malhas aninhadas é $\Omega^0 \subset \dots \subset \Omega^{J-1} \subset \Omega^J = \Omega^h$. A figura 1.1 exemplifica Ω^h obtida a partir de dois refinamentos sobre Ω^0 .

Figura 1.1: Obtenção da malha Ω^2 a partir de dois refinamentos sobre a malha Ω^0 .

Sejam as formulações de Galerkin

$$A^k u^k = f^k, \quad k = 0, \dots, J \quad (1.7)$$

associadas aos elementos finitos de cada malha Ω^k . Em particular, para $k = J$, seja u^J uma estimativa para a solução exata $\bar{u}^J$ da formulação de Galerkin (1.7) referente à malha mais fina Ω^J , isto é,

$$A^J u^J = f^J, \quad (1.8)$$

e ainda $e^J = \bar{u}^J - u^J$ o erro cometido nessa estimativa. O método multigrid busca eliminar esse erro através de duas estratégias: *suavização do erro* e *correção do resíduo na malha mais grossa*.

O espaço U^k dos elementos finitos associados à malha Ω^k é gerado pela base $\{w_{i,k} : i = 1, \dots, 2^{N-J+k} - 1\}$, onde $w_{i,k}$ é a função contínua em $[0, 1]$ e linear por partes em Ω^k , assumindo os valores $w_{i,k}(m2^{J-k}h) = \sin\left(m2^{J-k}h \cdot \frac{i\pi}{2^{J-k}}\right)$. A figura 1.2 mostra essas bases para o caso $2^{J-k}h = \frac{1}{8}$. Chamamos *componentes de baixa frequência* às funções $w_{i,k}$, $1 \leq i < 2^{N-J+k-1}$ e *componentes de alta frequência* às funções $w_{i,k}$, $2^{N-J+k-1} \leq i \leq 2^{N-J+k} - 1$ (na

Figura 1.2: Decomposição do espaço U^k em suas componentes de baixa e alta frequência.

figura, $w_{1,k}$, $w_{2,k}$ e $w_{3,k}$ são componentes de baixa frequência e as restantes são componentes de alta frequência). Sendo $\{w_{i,k}\}$ uma base, o erro e^k cometido no nível k (malha Ω^k) pode ser expresso por $e^k = \sum_{i=1}^{2^{N-J+k}-1} \lambda_{i,k} w_{i,k}$. Com o intuito de reduzir o erro $e^J = \sum_{i=1}^{2^N-1} \lambda_{i,J} w_{i,J}$, aplica-se um método iterativo S^J sobre o sistema (1.8) correspondente ao nível J , obtendo-se um novo valor para u^J . Tal método iterativo, ou “smoother”, visa principalmente anular a componente de alta frequência $\sum_{i=2^{N-1}}^{2^N-1} \lambda_{i,J} w_{i,J}$ associada ao erro. Esse procedimento é chamado suavização do erro, ou relaxação. Na prática utilizam-se Gauss-Seidel, damped-Jacobi, e outros. A etapa seguinte visa eliminar a componente de baixa frequência, e é chamada correção do resíduo na malha mais grossa: após aplicada a relaxação, calcula-se o resíduo $r^J = f^J - A^J u^J$ e aplica-se um operador de restrição $I_J^{J-1}: U^J \rightarrow U^{J-1}$ que restringe o resíduo r^J na malha Ω^J à sua aproximação $r^{J-1} = I_J^{J-1} r^J$ na malha mais grossa Ω^{J-1} . Busca-se então a solução da equação

$$A^{J-1} u^{J-1} = r^{J-1} \quad (1.9)$$

sobre a malha Ω^{J-1} de menor dimensão. Tal resolução pode ser exata ou pode-se aplicar recursivamente o mesmo princípio descrito para a equação (1.8), transferindo a resolução para a malha Ω^{J-2} . Obtida a solução u^{J-1} de (1.9), interpola-se a mesma sobre a malha Ω^J de forma a anular a componente de baixa frequência do erro, ou seja,

$$u^J = u^J + I_{J-1}^J u^{J-1}$$

onde $I_{J-1}^J: U^{J-1} \rightarrow U^J$ é um operador de interpolação. Isso completa a correção na malha mais grossa. Opcionalmente, pode-se ainda efetuar mais uma relaxação sobre (1.8).

O uso do resíduo como forma de corrigir o erro é baseado na equação residual

$$A^k e^k = r^k$$

válida para qualquer nível k .

No caso de resolução exata para (1.9), podemos representar o operador multigrid M_J aplicado sobre o erro e^J como

$$M_J e^J = \underbrace{\widehat{S^J}}_{\text{suavização}} \underbrace{(I_J - I_{J-1}^J (A^{J-1})^{-1} I_J^{J-1} A^J)}_{\text{correção na malha mais grossa}} \underbrace{\widehat{S^J}}_{\text{suavização}} e^J$$

onde I_J corresponde à identidade na malha $\Omega^J = \Omega^h$. Para um processo recursivo, adotamos resolução exata apenas na malha mais grossa Ω^0 . E obtemos a representação recursiva para o operador multigrid $M_J = M_J^{J-1}$ como

$$\begin{aligned} M_k^{k-1} &= S^k (I_k - I_{k-1}^k M_{k-1}^{k-2} I_k^{k-1} A^k) S^k, \quad k = J, J-1, \dots, 2 \\ \text{e} \\ M_0^1 &= S^1 (I_1 - I_0^1 (A^0)^{-1} I_1^0 A^1) S^1. \end{aligned} \quad (1.10)$$

Na figura 1.3 temos um esquema para o método (1.10). Essa formulação é conhecida como *V-ciclo* do método multigrid. Algoritmicamente temos

Algoritmo MG v-ciclo

relaxação em $A^J v^J = f^J$, ν_1 vezes, com chute inicial u^J

$$r^J = f^J - A^J u^J$$

$$f^{J-1} = I_{J-1}^{J-1} r^J$$

relaxação em $A^{J-1} v^{J-1} = f^{J-1}$, ν_1 vezes, com chute inicial $u^{J-1} = 0$

$$r^{J-1} = f^{J-1} - A^{J-1} u^{J-1}$$

$$f^{J-2} = I_{J-1}^{J-2} r^{J-1}$$

relaxação em $A^{J-2} v^{J-2} = f^{J-2}$, ν_1 vezes, com chute inicial $u^{J-2} = 0$

$$r^{J-2} = f^{J-2} - A^{J-2} u^{J-2}$$

$$f^{J-3} = I_{J-2}^{J-3} r^{J-2}$$

$\vdots$

resolva $A^0 v^0 = f^0$

$\vdots$

$$u^{J-2} = u^{J-2} + I_{J-3}^{J-2} u^{J-3}$$

relaxação em $A^{J-2} v^{J-2} = f^{J-2}$, ν_2 vezes, com chute inicial u^{J-2}

$$u^{J-1} = u^{J-1} + I_{J-2}^{J-1} u^{J-2}$$

relaxação em $A^{J-1} v^{J-1} = f^{J-1}$, ν_2 vezes, com chute inicial u^{J-1}

$$u^J = u^J + I_{J-1}^J u^{J-1}$$

relaxação em $A^J v^J = f^J$, ν_2 vezes, com chute inicial u^J

Figura 1.3: V-ciclo do método multigrid.

2 As Bases Hierárquicas

O método de bases hierárquicas, apresentado por Yserentant [49] e [51], utiliza uma estrutura multinível, visando um pré-condicionamento baseado na filosofia multigrid. Utilizando-se da hierarquia gerada durante a formação de sucessivos refinamentos da malha, procura-se dividir o espaço dos elementos finitos em subespaços, separando-se altas e baixas frequências a cada nível do refinamento. Isso é obtido substituindo-se as bases nodais (usualmente adotadas) pelas bases hierárquicas. Uma segunda heurística para adotarmos tais bases pode ser encontrada no método unigrid descrito por McCormick e Ruge [35] onde, na análise do operador de suavização (referente ao método multigrid) quando aplicado sobre elementos nodais, observa-se que este atua somente sobre dados locais. Assim, informações em pontos (vértices) distantes de um ponto x_i podem requerer vários passos para afetar a solução nesse ponto x_i . Dessa forma, McCormick e Ruge sugerem uma mudança de base que induz à estrutura das bases hierárquicas.

A matriz de massa obtida através do uso de bases hierárquicas é mais densa que a gerada pelas bases nodais (em Maubach [34] temos alguns exemplos gráficos), mas o seu cômputo e armazenamento não é necessário. No método dos gradientes conjugados basta determinar o vetor obtido pelo produto da matriz (gerada pelas bases hierárquicas) com um dado vetor. Isso pode ser feito utilizando-se a matriz original referente às bases nodais no cálculo do produto, e logo em seguida aplicando-se uma mudança de bases adequada. Em suma, comparando-se com o processo utilizando bases nodais, o trabalho computacional extra consiste apenas da adição do cômputo da mudança de bases.

Para bases nodais o número de condição da matriz de massa tem ordem $\mathcal{O}(h^{-2})$, onde h representa o passo na malha mais fina (vide Axelson e Barker [1] 5.5, por exemplo). Adotadas as bases hierárquicas, obtemos os números de condição $\mathcal{O}(1)$ para o caso unidimensional, $\mathcal{O}(\ln^2 h^{-1})$ para dimensão dois e $\mathcal{O}(h^{2-d})$ para dimensão $d \geq 3$ (conforme Xu [48]), o que mostra que as bases hierárquicas são bem condicionadas para uma ou duas dimensões, mas começam a deteriorar seu número de condição para dimensões maiores que dois.

2.1 Interpolação e suas estimativas

No que segue apresentamos uma análise das bases hierárquicas para um domínio de dimensão dois (como em Yserentant [49]).

Primeiramente definimos as interpolações que levam às bases hierárquicas e, numa segunda etapa, analisaremos suas dependências em relação ao refinamento adotado.

Sejam $\mathcal{I}_k: V^h = \mathcal{S}_J \rightarrow \mathcal{S}_k$, $k = 0, \dots, J$

$$u \mapsto \mathcal{I}_k(u) \text{ tal que } (\mathcal{I}_k u)(x_i) = u(x_i) \quad \forall x_i \in \mathcal{T}_k$$

as interpolações lineares de $u \in V^h$ sobre a triangularização $\mathcal{T}_k$, $k = 0, \dots, J$

Dado $u \in V^h$, temos

$$u = \mathcal{I}_0 u + \sum_{k=1}^J (\mathcal{I}_k u - \mathcal{I}_{k-1} u) \quad (2.1)$$

Pode-se considerar (2.1) como uma decomposição de u em frequências correspondentes a cada nível k do refinamento. Pois $\mathcal{I}_0 u \in \mathcal{S}_0$ é uma função correspondente à triangularização $\mathcal{T}_0$, enquanto $\mathcal{I}_k u - \mathcal{I}_{k-1} u \in \mathcal{S}_k$ anula-se em $\mathcal{N}_{k-1}$, isto é, nos vértices de $\mathcal{T}_{k-1}$. Isto induz a escolha da seguinte decomposição para V^h :

$$V_0 = \mathcal{S}_0 \text{ e } V_k = \text{imagem}(\mathcal{I}_k - \mathcal{I}_{k-1}) = (\mathcal{I}_k - \mathcal{I}_{k-1})(V^h) \subset \mathcal{S}_k$$

Portanto $V_k = \{\psi \in \mathcal{S}_k : \psi(x_i) = 0, x_i \in \mathcal{N}_{k-1}\}, k = 1, \dots, J$, e

$$V^h = V_0 \oplus V_1 \oplus \dots \oplus V_J \quad (2.2)$$

Uma forma para expressarmos tal decomposição, é através da introdução das *bases hierárquicas*. Tomando-se a base nodal de $\mathcal{T}_k$ geradora de $\mathcal{S}_k$, isto é,

$$\{\hat{\psi}_i^{(k)}\}_{i \in \mathcal{J}_k} = \{\hat{\psi}_i^{(k)}, i = 1, \dots, n_k : \hat{\psi}_i^{(k)}(x_l) = \delta_{il}, \forall i, l \in \mathcal{J}_k\}, \quad (2.3)$$

definimos recursivamente:

- para $k = 0$, a base hierárquica $\{\psi_i^{(0)}\}_{i \in \mathcal{J}_k}$ de $\mathcal{T}_0$ é igual à base nodal $\{\hat{\psi}_i^{(0)}\}_{i \in \mathcal{J}_0}$ de $\mathcal{T}_0$, isto é,

$$\{\psi_i^{(0)}\}_{i \in \mathcal{J}_0} = \{\psi_i^{(0)}, i = 1, \dots, n_0 : \psi_i^{(0)} \equiv \hat{\psi}_i^{(0)}\}$$

- para $k = 1, \dots, J$, a base hierárquica $\{\psi_i^{(k)}\}_{i \in \mathcal{J}_k}$ de $\mathcal{T}_k$ é igual à base hierárquica $\{\psi_i^{(k-1)}\}_{i \in \mathcal{J}_{k-1}}$ de $\mathcal{T}_{k-1}$ completada com elementos da base nodal $\{\hat{\psi}_i^{(k)}\}_{i \in \mathcal{J}_k}$ de $\mathcal{T}_k$, isto é,

$$\{\psi_i^{(k)}\}_{i \in \mathcal{J}_k} = \{\psi_i^{(k-1)}\}_{i \in \mathcal{J}_{k-1}} \cup \{\hat{\psi}_i^{(k)}\}_{i \in \mathcal{J}_k \setminus \mathcal{J}_{k-1}}$$

Um exemplo unidimensional é apresentado na figura 2.1. Assim $V_k = \text{span}\{\psi_i^{(k)} : i \in \mathcal{J}_k \setminus \mathcal{J}_{k-1}\} = \text{span}\{\hat{\psi}_i^{(k)} : i = n_{k-1} + 1, \dots, n_k\}$, como na figura 2.6, pag. 25.

Associada à decomposição (2.1) define-se a semi-norma discreta

$$|u|^2 = \sum_{k=1}^J \sum_{x \in \mathcal{N}_k \setminus \mathcal{N}_{k-1}} |(\mathcal{I}_k u - \mathcal{I}_{k-1} u)(x)|^2, \quad u \in V^h = \mathcal{S}_J \quad (2.4)$$

Observemos que tal semi-norma, tomada a formulação hierárquica de u , corresponde a norma euclideana dos coeficientes de u (a menos dos correspondentes à malha mais grossa).

Definimos ainda a norma L_2

$$\|u\|_{0,2;\Omega}^2 = \int_{\Omega} |u(x)|^2 dx, \quad u \in L_2(\Omega),$$

Figura 2.1: Exemplo unidimensional para geração de base (a) nodal e (b) hierárquica.

a semi-norma

$$|u|_{1,2;\Omega}^2 = \sum_{j=1}^2 \|\partial_j u\|_{0,2;\Omega}^2$$

e a norma

$$\|u\|_{1,2;\Omega}^2 = \|u\|_{0,2;\Omega}^2 + |u|_{1,2;\Omega}^2$$

para $u \in W^{1,2}(\Omega)$, onde $W^{1,2}(\Omega)$ é o espaço das distribuições de primeira derivada quadrado integrável.

Figura 2.2: Exemplo de um refinamento (a) regular e (b) irregular sobre um dado triângulo.

Durante o processo de refinamento admitimos dois tipos de atuação sobre os triângulos da malha: um *refinamento regular* sobre um triângulo de $\mathcal{T}_k$ consiste na preservação do mesmo ou em sua divisão em quatro triângulos congruentes (figura 2.2 (a)), chamados *triângulos regulares*; um *refinamento irregular* consiste na divisão do triângulo em dois através da conexão de um vértice com o ponto médio do lado oposto (figura 2.2 (b)), obtendo-se dois triângulos, chamados *triângulos irregulares*.

Num processo de refinamento regular, temos:

Teorema 1 *Existem constantes $c_1, c_2 > 0$ dependendo apenas de uma limitação inferior para os ângulos internos da triangularização, tais que:*

$$\frac{c_1}{(J+1)^2} (|\mathcal{I}_0 u|_{1,2;\Omega}^2 + |u|^2) \leq |u|_{1,2;\Omega}^2 \leq c_2 (|\mathcal{I}_0 u|_{1,2;\Omega}^2 + |u|^2), \quad \forall u \in V^h \quad (2.5)$$

Teorema 2 *Existem constantes $c_1^*, c_2^* > 0$ dependendo apenas do diâmetro do domínio Ω e de uma limitação inferior para os ângulos internos da triangularização, tais que:*

$$\frac{c_1^*}{(J+1)^2} (|\mathcal{I}_0 u|_{1,2;\Omega}^2 + |u|^2) \leq \|u\|_{1,2;\Omega}^2 \leq c_2^* (|\mathcal{I}_0 u|_{1,2;\Omega}^2 + |u|^2), \quad \forall u \in V^h \quad (2.6)$$

Observe que tais constantes *independem do número J de refinamentos e da forma do domínio Ω* . Mais ainda, J é da ordem $|\log_2 h|$ onde h é o passo na malha mais fina (malha de nível J). Como a triangularização inicial $\mathcal{T}_0$ é fixa independentemente do número J de refinamentos efetuados (e portanto $\mathcal{I}_0$ também), podemos interpretar as estimativas (2.5) e (2.6) da seguinte forma: a norma euclideana dos coeficientes de $u \in V^h$ expresso nas bases hierárquicas comporta-se como sua norma de Sobolev em $W^{1,2}(\Omega)$ (com uma restrição apenas ligada à uniformidade do processo de refinamento).

Na demonstração destes resultados Yserentant [49] apresenta uma série de lemas. Inicialmente estimativas sobre triângulos T do refinamento (lema 2.1 a lema 2.3) e, em seguida, extensões sobre Ω . Uma justificativa para a deterioração do método em três dimensões pode ser vista a partir da estimativa apresentada no lema 2.1:

$$\|u\|_{0,\infty;T} \leq c \left(\ln \frac{H}{h} + \frac{1}{4} \right)^{1/2} \|u\|_{1,2;T},$$

válida para o caso dois dimensional (H correspondente ao maior diâmetro da triangularização inicial), e que permite ($\frac{H}{h} \geq 1$) majorações da forma

$$\ln \frac{H}{h} + \frac{1}{4} \leq \log_2 \frac{H}{h} + \frac{1}{4} \approx \log_2 \frac{2^{-k} h_0}{2^{-J} h_0} + \frac{1}{4} = \log_2 2^{J-k} + \frac{1}{4} \leq J - k + 1$$

utilizada nas demonstrações subseqüentes. Tal estimativa usa fortemente a hipótese de dimensão dois (vide, por exemplo, Thomée [46] pag. 75. Veja também Dorfler [20], que procura generalizar as idéias de Yserentant para problemas de ordem maior e para espaços não conformes de elementos finitos).

Merece particular atenção o lema 2.7 de Yserentant [49], que mostra uma propriedade de ortogonalidade para os espaços V_k :

Figura 2.3: Refinamento local obtido a partir de refinamentos regulares e irregulares.

Lema *Existe uma constante $c > 0$ dependendo apenas de uma limitação inferior para os ângulos internos da triangularização, tal que:*

$$D(u, v) \leq c \left(\frac{1}{\sqrt{2}} \right)^{|k-l|} |u|_{1,2;\Omega} |v|_{1,2;\Omega}, \quad \forall u \in V_k, \quad \forall v \in V_l$$

onde $D(u, v) = \sum_{i=1}^2 \int_{\Omega} \partial_i u \partial_i v dx$.

Refinamentos irregulares não garantem uma limitação inferior para os ângulos internos formados depois de vários níveis de refinamento e podem gerar uma dependência em J para as constantes apresentadas nos teoremas 1 e 2. Para evitar tal efeito, assume-se a existência de uma constante $\theta > 0$ que limita inferiormente todos os ângulos internos criados durante o processo de refinamento.

Notemos que é muito importante considerar a possibilidade de refinamentos do tipo irregular, pois eles permitem refinamentos locais, como por exemplo, na figura 2.3.

Procuremos generalizar os teoremas 1 e 2 para também abranger esses refinamentos. Tal generalização é baseada na observação que todo triângulo T pertencente a uma triangularização $\mathcal{T}_k$, $k \leq J$ pode ser expresso como a união de triângulos em $\mathcal{T}_J$ com diâmetros h satisfazendo

$$h \geq \left(\frac{\sin \theta}{2} \right)^{J-k} \text{diam}(T)$$

A partir dessa consideração é possível estimar as limitações inferiores nos teoremas 1 e 2, a menos do termo H correspondente ao maior diâmetro da triangularização inicial (no caso do segundo teorema). Para as limitações superiores é necessário controlar a velocidade do processo de refinamento. Para tal assume-se a existência de constantes $\alpha > 0$ e $0 < q < 1$, tais que para todo triângulo T pertencente à triangularização $\mathcal{T}_k$, $k = 0, \dots, J$ os mesmos triângulos de $\mathcal{T}_J$ que expressam T como uma união, possuam diâmetro h satisfazendo

$$\frac{h}{\text{diam}(T)} \leq \alpha q^{J-k} \quad (2.7)$$

Tal condição é automaticamente satisfeita ao impor-se a seguinte regra: *triângulos obtidos a partir de refinamentos irregulares não podem ser refinados de forma irregular*. Tal regra garante que todo triângulo é semelhante a um triângulo de $\mathcal{T}_0$ (triangularização inicial) ou semelhante a algum refinamento irregular em $\mathcal{T}_0$. Assim, utilizando-se o refinamento irregular da figura 2.3, alternado com um refinamento regular, obtém-se refinamentos locais,

como na figura 2.4. Algoritmos para obtenção de refinamentos com essas características são discutidos, por exemplo, em Rivara [39, 40].

Figura 2.4: Obtenção de uma malha com refinamento local.

Adotando-se tal hipótese, obtemos as generalizações para as desigualdades (2.5) (que mantém-se a mesma) e (2.6), na forma:

$$\frac{c_1^*}{(J+1)^2} (\|\mathcal{I}_0 u\|_{1,2;\Omega}^2 + H^2 |u|^2) \leq \|u\|_{1,2;\Omega}^2 \leq c_2^* (\|\mathcal{I}_0 u\|_{1,2;\Omega}^2 + H^2 |u|^2), \quad \forall u \in V^h.$$

Em Yserentant [51] também encontra-se uma análise sobre a triangularização de forma a garantir que as estruturas de refinamento utilizadas no pacote PLTMG (Bank [2]) preservem as estimativas anteriores.

2.2 Solução do problema variacional

Dadas as estimativas (2.5) e (2.6) para as bases hierárquicas, analisemos o comportamento do pré-condicionador associado à forma bilinear, simétrica e positiva definida (2.8) correspondente à formulação variacional do problema:

$$a(u, v) = f(v), \quad \forall v \in H(\Omega) \subset W^{1,2}(\Omega) \quad (2.8)$$

Para tal análise são necessárias hipóteses de V-elasticidade (coercividade) e continuidade, de modo a garantir certas equivalências entre normas. Isto é, definida a norma da energia

$$|||u|||^2 = a(u, u) \quad (2.9)$$

temos que $\exists \delta, M > 0$ tais que

$$\delta |||u|||_{1,2;\Omega}^2 \leq |||u|||^2 \leq M |||u|||_{1,2;\Omega}^2, \quad \forall u \in H(\Omega). \quad (2.10)$$

Definindo, a partir de (2.4) e (2.9) a norma

$$|||u|||^2 = \|\mathcal{I}_0 u\|^2 + |u|^2, \quad \forall u \in V^h \quad (2.11)$$

obtemos a estimativa

Teorema 3 *Existem constantes $K_1, K_2 > 0$, dependendo apenas de M e δ em (2.10), de uma limitação inferior para os ângulos internos da triangularização final e do diâmetro do domínio Ω (para o caso dos refinamentos não uniformes descritos anteriormente, ainda temos K_2 dependendo de α e q em (2.7)), tais que:*

$$\frac{K_1}{(J+1)^2} |||u|||^2 \leq ||u||^2 \leq K_2 |||u|||^2, \quad \forall u \in V^h.$$

Seja $u \in V^h$ e sua representação em base hierárquica $u = \sum_{i=1}^J u_i \psi_i$. Para suas componentes em $V_1 \oplus \dots \oplus V_J$ (isto é, coeficientes u_i com $i \in \mathcal{J}_J \setminus \mathcal{J}_0$) temos a semi-norma (2.4) expressa como a norma euclidiana (como já foi observado)

$$\begin{aligned} |u|^2 &= \sum_{k=1}^J \sum_{x \in \mathcal{N}_k \setminus \mathcal{N}_{k-1}} |(\mathcal{I}_k u - \mathcal{I}_{k-1} u)(x)|^2 = \sum_{k=1}^J \sum_{x \in \mathcal{N}_k \setminus \mathcal{N}_{k-1}} |(\sum_{i \in \mathcal{J}_k \setminus \mathcal{J}_{k-1}} u_i \psi_i)(x)|^2 = \\ &= \sum_{k=1}^J \sum_{i \in \mathcal{J}_k \setminus \mathcal{J}_{k-1}} |u_i|^2 = \langle u - \mathcal{I}_0 u, u - \mathcal{I}_0 u \rangle \end{aligned}$$

e como os coeficientes u_i , $i \in \mathcal{J}_0$ coincidem tanto para representação hierárquica quanto para nodal, temos

$$||\mathcal{I}_0 u||^2 = a(\mathcal{I}_0 u, \mathcal{I}_0 u) = \langle \tilde{A}_0 \mathcal{I}_0 u, \mathcal{I}_0 u \rangle$$

onde $\tilde{A}_0$ é a matriz de massa correspondente ao problema variacional (2.8) formulado apenas sobre a malha mais grossa $\mathcal{T}_0$ (portanto, sobre o espaço V_0).

Seja A a matriz de massa correspondente ao problema (2.8), adotando-se as bases hierárquicas sobre $\mathcal{S}_J = V^h$ (observe que $A|_{\mathcal{S}_0} = \tilde{A}_0$). Sendo

$$A_0 = \begin{pmatrix} \tilde{A}_0 & 0 \\ 0 & I \end{pmatrix}$$

obtemos, pelo teorema 3,

$$\frac{K_1}{(J+1)^2} \langle A_0 x, x \rangle \leq \langle A x, x \rangle \leq K_2 \langle A_0 x, x \rangle, \quad \forall x \in \mathbb{R}^{n_J}.$$

Logo

$$\begin{aligned} \frac{1}{\mu_1} &= \max_{x \neq 0} \frac{\langle A_0 x, x \rangle}{\langle A x, x \rangle} \leq \frac{(J+1)^2}{K_1}, \\ \mu_2 &= \max_{x \neq 0} \frac{\langle A x, x \rangle}{\langle A_0 x, x \rangle} \leq K_2. \end{aligned}$$

Como A_0 e $\tilde{A}_0$ são simétricas e positivas definidas, existem decomposições de Cholesky tais que

$$A_0 = LL^T, \quad L = \begin{pmatrix} \tilde{L} & 0 \\ 0 & I \end{pmatrix}, \quad \tilde{A}_0 = \tilde{L}\tilde{L}^T.$$

Observando que, dado $x \neq 0$,

$$\frac{\langle A x, x \rangle}{\langle A_0 x, x \rangle} = \frac{\langle A x, x \rangle}{\langle LL^T x, x \rangle} = \frac{\langle AL^{-T}L^T x, L^{-T}L^T x \rangle}{\langle L^T x, L^T x \rangle} \Rightarrow$$

$$\Rightarrow \max_{x \neq 0} \frac{\langle Ax, x \rangle}{\langle A_0 x, x \rangle} = \max_{x \neq 0} \frac{\langle AL^{-T}x, L^{-T}x \rangle}{\langle x, x \rangle} = \max_{x \neq 0} \frac{\langle L^{-1}AL^{-T}x, x \rangle}{\langle x, x \rangle}$$

temos μ_1 e μ_2 , respectivamente, o menor e maior autovalor de $L^{-1}AL^{-T}$. Logo

$$\kappa(L^{-1}AL^{-T}) \leq \frac{K_2}{K_1}(J+1)^2.$$

Observemos que os termos em $A_0 = LL^T$ correspondem apenas à uma correção referente à geometria da triangularização inicial T_0 (ou seja, os blocos $\tilde{L}$ e $\tilde{L}^T$). Logo, não dependem do processo de refinamento e do número de níveis J . Assim

$$\kappa(A) \leq \mathcal{O}((J+1)^2). \quad (2.12)$$

Para o cálculo de produtos internos como $\langle Ax, y \rangle$ no método de gradientes conjugados (CG) é necessário apenas o uso de uma transformação entre a matriz A referente a representação hierárquica e a matriz $\hat{A}$ referente a representação nodal. Tal transformação pode ser obtida através da matriz S que transforma a representação de um elemento em V^h com relação às bases hierárquicas numa representação com relação às bases nodais. Pois, dados $x, y \in \mathbb{R}^n$ dois vetores de coeficientes referentes à representação hierárquica, temos Sx e Sy suas respectivas representações nodais e

$$\langle x, Ay \rangle = \langle Sx, \hat{A}Sy \rangle = \langle x, S^T \hat{A}Sy \rangle$$

o que resulta na representação

$$A = S^T \hat{A} S. \quad (2.13)$$

Tomando a formulação (1.5), pag 7, do método PCG, notamos que basta considerar $E = S^{-T}$ para que a relação $\hat{u} = E^T u$ tenha o significado da mudança $\hat{u} = S^{-1}u$ da base nodal para a base hierárquica. Dessa forma, obtemos

$$A = \tilde{A} = E^{-1} \hat{A} E^{-T} = S^T \hat{A} S$$

como em (2.13), com o pré-condicionador M definido por

$$M^{-1} = E^{-T} E^{-1} = S S^T.$$

O algoritmo para o método das bases hierárquicas pode ser então descrito como:

Algoritmo HB

escolha u^0
 $r^0 = f - \hat{A}u^0$
 $\hat{r}^0 = M^{-1}r^0$
 $p^0 = \hat{r}^0$
 para $k = 0, \dots, k_{max}$
 $\alpha = \langle \hat{r}^k, r^k \rangle / \langle p^k, \hat{A}p^k \rangle$
 $u^{k+1} = u^k + \alpha p^k$
 teste de convergência
 $r^{k+1} = r^k - \alpha \hat{A}p^k$
 $\hat{r}^{k+1} = M^{-1}r^{k+1}$
 $\beta = \langle \hat{r}^{k+1}, r^{k+1} \rangle / \langle r^k, r^k \rangle$
 $p^{k+1} = \hat{r}^{k+1} + \beta p^k$
 próximo k

Pode-se ainda levar em conta as componentes $A_0 = LL^T$ correspondentes à correção na malha mais grossa. Ou seja, tomemos a mudança $\hat{u} = L^T S^{-1}u$ da base nodal para a base hierárquica, onde temos o termo L^T associado à decomposição de A_0 . Isso significa tomarmos $E = S^{-T}L$, obtendo-se

$$A = \hat{A} = E^{-1} \hat{A} E^{-T} = L^{-1} S^T \hat{A} S L^{-T}$$

e

$$M^{-1} = E^{-T} E^{-1} = S L^{-T} L^{-1} S^T = S A_0^{-1} S^T.$$

Figura 2.5: Representação $I1$ e $I2$ para o processo de refinamento

Resta determinarmos o cálculo de Sx e $S^T y$. Como, dado um nível k , os elementos hierárquicos criados nesse nível coincidem com os elementos nodais, a matriz S pode ser determinada como uma composição de operadores $S = S_J S_{J-1} \dots S_1$ onde S_k descreve a influência dos coeficientes referentes aos elementos criados até o nível $k-1$ do refinamento, sobre os elementos do nível k . Dessa forma, a estrutura de S_k fica reduzida a seguinte formulação por blocos

$$S_k = \begin{pmatrix} I & 0 & 0 \\ R & I & 0 \\ 0 & 0 & I \end{pmatrix} \begin{matrix} x_i \in \mathcal{N}_{k-1} \\ x_i \in \mathcal{N}_k \setminus \mathcal{N}_{k-1} \\ x_i \in \mathcal{N}_J \setminus \mathcal{N}_k \end{matrix} \quad (2.14)$$

onde R descreve, por linhas, a influência dos elementos hierárquicos do nível $k - 1$ sobre os elementos nodais do nível k . Uma maneira de obtermos R e também um algoritmo para o cálculo de $S_k x$ ($x \in V^h$) é introduzirmos os vetores $I1$ e $I2$ no processo de refinamento. $I1(i)$ e $I2(i)$ indicam que o nó x_i do nível k foi obtido como ponto médio dos nós $x_{I1(i)}$ e $x_{I2(i)}$ pertencentes a níveis menores que k (veja exemplo na figura 2.5). Tal notação permite estabelecer a relação entre as duas bases :

$$\psi_i = \hat{\psi}_i + \frac{1}{2}\hat{\psi}_{I1(i)} + \frac{1}{2}\hat{\psi}_{I2(i)} \quad (\text{supondo tomados os pontos médios para gerar novos nós}) .$$

Na estrutura de S_k , tal formação do nó x_i e do respectivo elemento ψ_i , reflete-se da seguinte forma

$$S_k = \begin{pmatrix} & I & & 0 & & 0 \\ & \vdots & & \vdots & & \\ \cdots & \frac{1}{2} & \cdots & \frac{1}{2} & \cdots & \\ & \vdots & & \vdots & & \\ & \vdots & & 0 & & \vdots \\ & \vdots & & & 0 & I \\ & \vdots & & & & \vdots \\ I1(i) & & I2(i) & & & \end{pmatrix} \cdots i$$

Dessa forma, a implementação de um algoritmo para S_k , onde $\hat{\psi}_i(x_i) = X(i)$, $i = 1, \dots, n_J$, pode ser expressa por

Cômputo de $S_k x$

para $i \in \mathcal{M}_k = \{i : x_i \in \mathcal{N}_k \setminus \mathcal{N}_{k-1}\}$

$$X(i) = X(i) + (X(I1(i)) + X(I2(i)))/2$$

próximo i

E como $S = S_J \cdots S_1$, temos Sx calculado da seguinte forma

Mudança Sx da base hierárquica para nodal

para $k = 1$ até J

para $i \in \mathcal{M}_k$

$$X(i) = X(i) + (X(I1(i)) + X(I2(i)))/2$$

próximo i

próximo k

Para o caso unidimensional podemos visualizar graficamente tal processo como sucessivas interpolações (figura 2.6).

Analogamente obtemos

Figura 2.6: Interpolações para o caso unidimensional.

Cômputo de $S^T x$, $S^T = S_1^T \cdots S_J^T$
 para $k = J$ até 1, decrescendo
 para $i \in \mathcal{M}_k$
 $X(I1(i)) = X(I1(i)) + X(i)/2$
 $X(I2(i)) = X(I2(i)) + X(i)/2$
 próximo i
 próximo k

Como $S_k^{-1} = \begin{pmatrix} I & 0 & 0 \\ -R & I & 0 \\ 0 & 0 & I \end{pmatrix}$ segue que

Mudança $S^{-1}x$ da base nodal para hierárquica
 para $k = J$ até 1, decrescendo
 para $i \in \mathcal{M}_k$
 $X(i) = X(i) - (X(I1(i)) + X(I2(i)))/2$
 próximo i
 próximo k

e

Cômputo de $S^{-T}x$, $S^{-T} = S_J^{-T} \cdots S_1^{-T}$
 para $k = 1$ até J
 para $i \in \mathcal{M}_k$
 $X(I1(i)) = X(I1(i)) - X(i)/2$
 $X(I2(i)) = X(I2(i)) - X(i)/2$
 próximo i
 próximo k

Dessa forma obtemos um método para o cômputo de $M^{-1} = SA_0^{-1}S^T x$ no algoritmo HB. Tal operação assemelha-se a um V-ciclo do método multigrid, onde a etapa $S^T x$ corresponde à descida da malha mais fina até a malha mais grossa, $A_0^{-1}(S^T x)$ corresponde à correção na malha mais grossa, e $S(S^T x)$ corresponde à subida da malha mais grossa até a malha mais fina.

Como foi observado no final da seção 1.2, um pré-condicionador mais geral é dado por

$$M^{-1} = SD^{-1}S^T. \quad (2.15)$$

Em particular, temos $D = LL^T$, mas veremos mais à frente que D pode ser tomada como uma matriz diagonal de forma a levar em conta o passo h_k associado a cada nível do processo de refinamento. Pela relação (1.6) a estimativa para $p(\varepsilon)$ correspondente ao número máximo de iterações necessárias para a redução do erro em um fator $\varepsilon \in (0, 1)$ é

$$p(\varepsilon) \leq \frac{1}{2} \sqrt{\kappa} \ln \left(\frac{2}{\varepsilon} \right) + 1,$$

onde $\kappa = \kappa(D^{-1}S^T \hat{A} S)$. Lembrando, por (2.12), que o número de condição para bases hierárquicas no caso dois dimensional é $\mathcal{O}((J+1)^2)$, observamos que um limite superior para o número de iterações de PCG pode ser obtido em função do número de refinamentos J

$$C(J+1) \ln \left(\frac{2}{\varepsilon} \right)$$

de forma a reduzir o erro da energia por um fator ε .

Observemos que a mudança $S^T r$ acarreta no acréscimo de $4n_J$ operações e a mudança Sy acrescenta $3n_J$ operações. Assim, a menos do cálculo $D^{-1}v$, temos o acréscimo de $7n_J$ operações sobre o algoritmo CG original.

2.3 Considerações e generalizações

Um segundo caminho para a obtenção das estimativas anteriores é mostrado por Yserentant [51] (vide também Xu [48]) onde a norma (2.11) é redefinida através da norma L_2

$$|||u|||^2 = ||\mathcal{I}_0 u||^2 + \sum_{k=1}^J 4^k ||\mathcal{I}_k u - \mathcal{I}_{k-1} u||_{0,2;\Omega}^2. \quad (2.16)$$

Neste caso, os fatores 4^k decorrem do fato de que um triângulo do nível k tem metade do diâmetro do triângulo pertencente ao nível $k-1$ que o gerou. Logo, tais fatores dependem da estratégia de refinamento e devem ser substituídos, por exemplo, por fatores 9^k no caso dos diâmetros serem reduzidos em um terço. Além disso, utilizando essa formulação de norma, obtemos um paralelo mais próximo à formulação do método BPX (ver capítulo seguinte), onde temos uma decomposição espectral e os fatores do tipo 4^k tem a função de substituir os raios espectrais λ_k em Bramble, Pasciak e Xu [9].

Novamente podemos obter as estimativas do teorema 3 para (2.16). Tomando-se B_H o operador auto-adjunto positivo definido associado à norma (2.16), isto é,

$$|||u|||^2 = (B_H u, u)$$

obtemos pelo teorema a estimativa

$$\kappa(B_H^{-1/2} A B_H^{-1/2}) \leq \frac{K_2}{K_1} (J+1)^2, \quad (2.17)$$

mostrando a conveniência de se usar B_H^{-1} como pré-condicionador. No entanto, seu custo computacional é muito alto, o que nos leva a considerar formas alternativas para um pré-condicionador, essencialmente equivalente a B_H^{-1} , mas mais atraente do ponto de vista computacional. Yserentant ([51], lema 5.1) mostra que tal norma induzida por B_H é equivalente à norma discreta

$$|||u|||^2 = ||\mathcal{I}_0 u||^2 + \sum_{k=1}^J \sum_{x_i \in \mathcal{N}_k \setminus \mathcal{N}_{k-1}} d_i |(\mathcal{I}_k u - \mathcal{I}_{k-1} u)(x_i)|^2 \quad (2.18)$$

definida em S_J , onde

$$d_i = 4^k (1, \psi_i^k), \quad i \in \mathcal{J}_k \setminus \mathcal{J}_{k-1}. \quad (2.19)$$

É possível substituir os pesos (2.19) por outros, por exemplo:

$$d_i = 4^k (\psi_i^k, \psi_i^k),$$

ou

$$d_i = a(\psi_i^k, \psi_i^k). \quad (2.20)$$

Embora o pré-condicionador $M^{-1} = S A_0^{-1} S^T$, na sua formulação original, esteja intimamente ligado ao processo de refinamento, pouca informação contém sobre o operador elíptico em questão (apenas A_0). Na prática, os pesos d_i visam suprir tal deficiência. Greenbaun, Li e Chao [26] tomam $D = \text{diag}(\hat{A})$ em (2.15) incorporando o método de escalonamento diagonal ao pré-condicionador como uma forma de captar variações do operador. Isto equivale ao peso (2.20).

Uma representação para o pré-condicionador gerado ao adotarmos bases hierárquicas pode ser obtida ao descrevermos o processo como um método iterativo linear associado a resolução de (2.8):

$$u \leftarrow u + C(b - Au) \quad (2.21)$$

onde

$A: V^h \rightarrow V^h$ operador auto-adjunto e positivo definido, tal que
 $(Au, v) = a(u, v) \quad \forall v \in V^h$ e
 $b \in V^h$ tal que
 $(b, v) = f(v) \quad \forall v \in V^h$.

Reescrevendo o método em função do resíduo r , obtemos

$$\begin{aligned} r &\leftarrow b - Au \\ u &\leftarrow u + Cr \end{aligned} \quad (2.22)$$

Uma forma de aplicarmos (2.22) é tomando a representação nodal para $u, r \in V^h$. Para $\{\hat{\psi}_i\}_{i=1}^{n_J} = \{\hat{\psi}_i^J\}_{i \in \mathcal{J}_J}$ bases nodais em $\mathcal{T}_J$ temos

$$\begin{aligned} (r, \hat{\psi}_i) &= (b - Au, \hat{\psi}_i) = f(\hat{\psi}_i) - (Au, \hat{\psi}_i) = f(\hat{\psi}_i) - a(u, \hat{\psi}_i) = \\ &= f(\hat{\psi}_i) - a(\sum_{l=1}^{n_J} u(x_l) \hat{\psi}_l, \hat{\psi}_i) = f(\hat{\psi}_i) - \sum_{l=1}^{n_J} u(x_l) a(\hat{\psi}_l, \hat{\psi}_i), \end{aligned}$$

isto é,

$$(r, \hat{\psi}_i) = f(\hat{\psi}_i) - \sum_{l=1}^{n_J} u(x_l) a(\hat{\psi}_l, \hat{\psi}_i), \quad i = 1, \dots, n_J. \quad (2.23)$$

Dessa forma, cada valor $(r, \hat{\psi}_i)$ pode ser calculado como na forma usual para obtenção de resíduos, sem a necessidade da representação explícita da matriz A como um todo.

Para a escolha de C em (2.21), como uma aproximação da inversa de A , a opção imediata seria tomar

$$Cr = B_H^{-1} r$$

em (2.22). Como foi observado, computacionalmente é mais vantajoso adotarmos a norma (2.18). Tomemos H o operador associado a essa norma, isto é,

$$|||u|||^2 = (Hu, u).$$

Dessa forma, podemos definir o pré-condicionador:

$$C = H^{-1} \quad (2.24)$$

A forma final do pré-condicionador é dada pelo seguinte

Lema 1 Para todo $r \in V^h$, temos

$$Cr = A_0^{-1} Q_0 r + \sum_{k=1}^J \sum_{x_i \in \mathcal{N}_k \setminus \mathcal{N}_{k-1}} d_i^{-1}(r, \psi_i) \psi_i, \quad (2.25)$$

onde $Q_0 : V^h \rightarrow V_0$ é dado por $(Q_0 r, v) = (r, v), \forall v \in V_0$.

Demonstração. Fixado $r \in V^h$, para todo $v \in \mathcal{S}_0 = V_0$, e C dado por (2.25), temos

$$(HCr, v) = a(\mathcal{I}_0 A_0^{-1} Q_0 r, v) + a(0, v) = (A_0^{-1} Q_0 r, A_0 v) = (Q_0 r, v) = (r, v).$$

E para todo $v \in V_k$ onde $k = 1, \dots, J$, temos

$$\begin{aligned} (HCr, v) &= \sum_{x_i \in \mathcal{N}_k \setminus \mathcal{N}_{k-1}} d_i \left((\mathcal{I}_k - \mathcal{I}_{k-1}) Cr \right) (x_i) v(x_i) = \\ &= \sum_{x_i \in \mathcal{N}_k \setminus \mathcal{N}_{k-1}} d_i \left(\sum_{x_l \in \mathcal{N}_k \setminus \mathcal{N}_{k-1}} d_l^{-1}(r, \psi_l) \psi_l \right) (x_i) v(x_i) = \\ &= \sum_{x_i \in \mathcal{N}_k \setminus \mathcal{N}_{k-1}} d_i d_i^{-1}(r, \psi_i) \psi_i(x_i) v(x_i) = \sum_{x_i \in \mathcal{N}_k \setminus \mathcal{N}_{k-1}} (r, \psi_i) v(x_i) = \\ &= (r, \sum_{x_i \in \mathcal{N}_k \setminus \mathcal{N}_{k-1}} v(x_i) \psi_i) = (r, v) \end{aligned}$$

Como $V^h = V_0 \oplus \cdots \oplus V_J$, temos $HCr = r$, $\forall r \in V^h$. Logo $C = H^{-1}$ como queríamos em (2.24). $\square$

Uma primeira observação é que podemos obter os valores (r, ψ_i) de (2.25), referentes as bases hierárquicas, a partir do conjunto $\{(r, \hat{\psi}_i^J)\}_{i \in \mathcal{J}_J}$ correspondentes ao resíduo projetado sobre a base nodal. Para isso, basta tomarmos o processo recursivo definido por S^T , pois se $x_i \in \mathcal{N}_k \setminus \mathcal{N}_{k-1}$ foi gerado como ponto médio de $x_{I1(i)} \in \mathcal{N}_l$ e $x_{I2(i)} \in \mathcal{N}_m$ com $m, l < k$, temos de transportar a informação para avaliarmos os valores em termos da base hierárquica

$$(r, \psi_{I1(i)}) = (r, \hat{\psi}_{I1(i)}) + \frac{1}{2}(r, \hat{\psi}_i)$$

$$(r, \psi_{I2(i)}) = (r, \hat{\psi}_{I2(i)}) + \frac{1}{2}(r, \hat{\psi}_i).$$

Outra observação refere-se ao termo $A_0^{-1}Q_0r$. Dada $r \in \mathcal{S}_0$, para a função

$$u_0 = A_0^{-1}Q_0r \in \mathcal{S}_0$$

temos

$$A_0u_0 = Q_0r \Rightarrow a(u_0, v) = (Q_0r, v) = (r, v), \quad \forall v \in \mathcal{S}_0.$$

Assim, para o cálculo de u_0 necessitamos apenas dos valores $\{(r, \psi_i)\}_{i \in \mathcal{J}_0}$ e não do cálculo da projeção Q_0r . O que corresponde à resolução do sistema linear associado ao nível zero (malha mais grossa).

Notamos que a formulação de C apresentada em (2.25) tem a característica de atuar como um método de Jacobi por blocos, onde a divisão dos blocos fica definida pela decomposição $V^h = V_0 \oplus \cdots \oplus V_J$ e o “solver” nos subespaços V_k , $k = 1, \dots, J$ sendo uma matriz diagonal tomando os valores d_i tais que $i \in \mathcal{J}_k \setminus \mathcal{J}_{k-1}$. O pré-condicionador é uma matriz bloco diagonal nas bases hierárquicas. O primeiro bloco é definido pelos elementos nodais de V_0 (isto é, A_0^{-1}) e os outros são diagonais obtidos a partir de $A = S^T \hat{A} S$, como por exemplo, $d_i = a(\psi_i, \psi_i)$. Matricialmente podemos denotar

$$D^{-1}S^T \hat{A} S x = f \tag{2.26}$$

onde D corresponde à matriz bloco diagonal descrita acima. Adotada tal formulação matricial, podemos aplicar o método dos gradientes conjugados (CG) sobre o sistema pré-condicionado (2.26), obtendo um processo de gradientes conjugados pré-condicionado (PCG).

Para a aplicação de C na iteração (2.22) ou em (PCG), os valores $\{(r, \hat{\psi}_i^J)\}_{i \in \mathcal{J}_J}$ já são estabelecidos pela formulação nodal. Logo, o produto interno $(\cdot, \cdot)$ atua no algoritmo final de forma indireta, através do fator de escala d_i . Assim, a escolha desses fatores influi na performance do método.

Baseados na formulação (2.26), para a implementação de PCG, só nos resta definir o algoritmo para o cálculo de Cr , onde r representa o resíduo expresso em bases nodais.

Cálculo de $z = Cr$

$v = S^T r$ (transfere informação da base nodal para base hierárquica)

$y = D^{-1}v$ (resolução na malha mais grossa e elementos diagonais)

$z = Sy$ (mudança da base hierárquica para a base nodal)

E novamente observamos a estrutura de um V-ciclo.

Em análises a posteriori, a resolução na malha mais grossa muitas vezes é substituída por um fator de correção relacionado ao passo h_0 dessa malha. Na verdade, fatores do mesmo tipo são aplicados em todos os níveis do refinamento, visando uma correção relacionada à cada operador A_k (operador associado ao problema restrito a $\mathcal{S}_k$, e portanto ao nível k do processo de refinamento). A grosso modo, temos o conceito de suavizadores, aproximando A_k^{-1} , relativos a cada nível (como na extensão HBMG da seção seguinte). Esses suavizadores podem ser substituídos pelos seu espectros λ_k^{-1} e, num passo seguinte, tais espectros também são substituídos pelo fator h_k^{d-2} , onde h_k representa o passo na malha de nível k , e d é a dimensão de Ω . Uma análise mais detalhada é feita na seção 3.2 referente ao método BPX. Temos assim

$$D^{-1} = \text{bloco diagonal } (h_J^{2-d} I_{n_J - n_{J-1} + 1}, h_{J-1}^{2-d} I_{n_{J-1} - n_{J-2} + 1}, \dots, h_1^{2-d} I_{n_1 - n_0 + 1}, h_0^{2-d} I_{n_0} \text{ ou } A_0^{-1})$$

para o cômputo de $M^{-1} = SD^{-1}S^T$ no algoritmo HB, pag. 22 .

A possível escolha entre resolução completa ou não para a malha mais grossa (equivalentemente A_0^{-1} ou $h_0^{2-d} I_{n_0}$ em D^{-1}), induz à nomenclatura (Elman e Guo [25]): *método das bases hierárquicas completo* (CHB) e *método das bases hierárquicas incompleto* (IHB). Dessa forma, o método IHB pode ser descrito como sendo o método HB (pag. 22) utilizando-se o pré-condicionador M descrito por

pré-condicionamento IHB, retorna $\hat{r} = M^{-1}r$

$$\tilde{r} = r$$

para $k = J$ até 1, decrescendo

$$\tilde{r} = S_k^T \tilde{r}$$

próximo k

$$\tilde{r} = h_J^{2-d} I_{n_J - n_{J-1} + 1} \tilde{r} + \dots + h_0^{2-d} I_{n_0} \tilde{r}$$

para $k = 1$ até J

$$\hat{r} = S_k \tilde{r}$$

próximo k

2.4 O método HBMG

Uma extensão para o método de bases hierárquicas é apresentada no método *multigrid-bases hierárquicas* (HBMG) apresentado por Bank, Dupont e Yserentant [4] e por Bank e Yserentant [5]. Em linhas gerais, ao tomarmos o sistema linear

$$\hat{A}\hat{x} = \hat{b}$$

correspondente à formulação nodal do problema (2.8), obtemos sua solução através do correspondente sistema

$$Ax = b \quad (2.27)$$

expresso em termos das bases hierárquicas. A mudança de bases ocorre de forma implícita, através dos operadores S definidos em (2.13). O novo sistema hierárquico é resolvido utilizando-se PCG. Como pré-condicionador adotamos relaxações associadas ao particionamento da matriz A em blocos induzidos pela separação (2.2) de $V^h = S_J$.

Algoritmicamente, o pré-condicionador é implementado exatamente como um V-ciclo do método multigrid, com operador de suavização Gauss-Seidel. Mas, com a exceção de que o processo de suavização do nível k ocorre apenas sobre vértices correspondentes ao espaço V_k (e não S_k).

Consideremos o sistema (2.27) correspondente à discretização nas bases hierárquicas tomada a decomposição (2.2). Temos

$$\begin{pmatrix} A_{00} & \cdots & A_{0J} \\ \vdots & \ddots & \vdots \\ A_{J0} & \cdots & A_{JJ} \end{pmatrix} \begin{pmatrix} x_0 \\ \vdots \\ x_J \end{pmatrix} = \begin{pmatrix} b_0 \\ \vdots \\ b_J \end{pmatrix}$$

$$\text{onde } A_{kl} = \begin{pmatrix} a(\psi_{n_{l-1}+1}^l, \psi_{n_{k-1}+1}^k) & \cdots & a(\psi_{n_l}^l, \psi_{n_{k-1}+1}^k) \\ \vdots & \ddots & \vdots \\ a(\psi_{n_{l-1}+1}^l, \psi_{n_k}^k) & \cdots & a(\psi_{n_l}^l, \psi_{n_k}^k) \end{pmatrix} \text{ e } b_k = \begin{pmatrix} f(\psi_{n_{k-1}+1}^k) \\ \vdots \\ f(\psi_{n_k}^k) \end{pmatrix}$$

(subentendido que $n_{0-1} = 0$). Tomemos a decomposição

$$A = L + D + L^T$$

de A em suas componentes bloco diagonal inferior, bloco diagonal e bloco diagonal superior, respectivamente. Isto é,

$$L = \begin{pmatrix} 0 & \cdots & \cdots & 0 \\ A_{10} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ A_{J0} & \cdots & A_{JJ-1} & 0 \end{pmatrix}, \quad D = \begin{pmatrix} A_{00} & \cdots & \cdots & 0 \\ 0 & \ddots & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & \cdots & A_{JJ} \end{pmatrix}.$$

Como um método iterativo para a solução de (2.27), podemos tomar Gauss-Seidel simétrico por blocos

$$x^{(i+1/2)} = x^{(i)} + (L + D)^{-T}(b - Ax^{(i)}) \quad (2.28)$$

$$x^{(i+1)} = x^{(i+1/2)} + (L + D)^{-1}(b - Ax^{(i+1/2)}) \quad (2.29)$$

onde (2.29) corresponde à iteração usual por blocos definida ao tomarmos $(L + D)$, e (2.28) iteração aplicada em um sentido reverso, pois $(D + L)^T = D^T + L^T = D + L^T$.

Mais genericamente, podemos substituir a matriz bloco diagonal D em (2.28) e (2.29) por outra matriz $\tilde{D}$ não singular. Obtendo

$$x^{(i+1/2)} = x^{(i)} + (L + \tilde{D})^{-T}(b - Ax^{(i)}) \quad (2.30)$$

$$x^{(i+1)} = x^{(i+1/2)} + (L + \tilde{D})^{-1}(b - Ax^{(i+1/2)}) \quad (2.31)$$

Um resultado inicial, correspondente a uma reformulação para tal iteração é o seguinte

Teorema 4 *Seja $\tilde{D}$ tal que a matriz simétrica*

$$X = \tilde{D} + \tilde{D}^T - D$$

é positiva definida. Então a iteração (2.30) e (2.31) pode ser reescrita como

$$x^{(i+1)} = x^{(i)} + B^{-1}(b - Ax^{(i)}) \quad (2.32)$$

onde

$$B = (L + \tilde{D})^T X^{-1} (L + \tilde{D}) \quad (2.33)$$

é simétrica e positiva definida.

Para a escolha de $\tilde{D}$ consideremos a decomposição

$$D = l + d + l^T$$

$$\text{onde } d = \begin{pmatrix} A_{00} & \cdot & \cdots & 0 \\ \cdot & \text{diag}(A_{11}) & & \cdot \\ \vdots & & \ddots & \vdots \\ 0 & \cdot & \cdots & \text{diag}(A_{JJ}) \end{pmatrix}.$$

Tomemos três casos

caso 1: $\tilde{D} = D$ (isto é, $X = D$)

caso 2: $\tilde{D} = l + d$ (isto é, $X = d$)

caso 3: $\tilde{D} = (l + d)^T d^{-1} (l + d) = D + l^T d^{-1} l$ (isto é, $X = D + 2l^T d^{-1} l$)

Lembrando que globalmente nosso processo atua como um Gauss-Seidel onde cada termo é um bloco, observamos que tais escolhas correspondem ao processo de iteração restrito a cada bloco. No caso 1, temos o cálculo das inversas dos blocos diagonais, logo o processo como um todo corresponde ao método de Gauss-Seidel simétrico por blocos na sua formulação usual. No caso 2, em cada bloco aplicamos um Gauss-Seidel ponto-a-ponto, exceto para o primeiro bloco que é invertido. No caso 3, temos um Gauss-Seidel simétrico ponto-a-ponto, exceto para o primeiro bloco, logo o processo como um todo corresponde a um Gauss-Seidel simétrico ponto-a-ponto exceto pelo primeiro bloco. Mais ainda, para todos os três casos a matriz X é positiva definida, assim vale o teorema 4.

Baseados na formulação (2.32) do método iterativo, novamente analisemos uma majoração para o número de condição $\kappa = \kappa(B^{-1}A)$ para obtermos uma estimativa sobre o método PCG. Lembremos que para μ_1 e μ_2 definidos por

$$\frac{1}{\mu_1} = \max_{x \neq 0} \frac{\langle Bx, x \rangle}{\langle Ax, x \rangle}$$

e

$$\mu_2 = \max_{x \neq 0} \frac{\langle Ax, x \rangle}{\langle Bx, x \rangle}$$

temos

$$\kappa = \frac{\mu_2}{\mu_1}.$$

Um resultado apresentado em Bank, Dupont e Yserentant [4] mostra que μ_2 (o maior autovalor de $B^{-1}A$) é sempre igual a um, nos três casos de escolha para $\tilde{D}$. Logo

$$\kappa = \max_{x \neq 0} \frac{\langle Bx, x \rangle}{\langle Ax, x \rangle} = \frac{1}{\mu_1}$$

e, para o processo iterativo (2.32) temos o raio de convergência

$$\rho(I - B^{-1}A) = 1 - \frac{1}{\kappa} = 1 - \mu_1. \quad (2.34)$$

O comportamento qualitativo do método é o mesmo para as diferentes escolhas de $\tilde{D}$ (conforme Bank, Dupont e Yserentant [4]), obtendo-se nos três casos que:

$$\kappa \leq C(J + 1)^2,$$

onde C é uma constante dependendo apenas da limitação inferior sobre os ângulos internos formados durante o processo de refinamento.

Na implementação do pré-condicionador obtemos basicamente um V-ciclo (vide Hackbusch [27], por exemplo):

V-ciclo para PCG

1. para $k = J$ até 1, decrescendo
 - Suavização sobre os vértices em $\mathcal{J}_k \setminus \mathcal{J}_{k-1}$, correspondente à escolha dos casos 1, 2 ou 3
 - Restrição do resíduo referente ao nível k para o nível $k - 1$, o que corresponde ao operador S_k^T
2. Resolução da equação no nível $k = 0$
3. para $k = 1$ até J
 - Interpolação da correção obtida no nível $k - 1$ para o nível k , o que corresponde ao operador S_k
 - Suavização sobre os vértices em $\mathcal{J}_k \setminus \mathcal{J}_{k-1}$, correspondente à escolha dos casos 1, 2 ou 3

A grande diferença em relação a um V-ciclo do método multigrid, consiste em aplicarmos suavização apenas sobre os vértices $(\mathcal{N}_k \setminus \mathcal{N}_{k-1})$ criados no nível k , e não sobre todos $(\mathcal{N}_k)$. Computacionalmente, expressamos o método como um PCG onde

$$M^{-1} = SB^{-1}S^T$$

com B apresentado em (2.33). Obtemos

Algoritmo HBMG

```

escolha  $u^0$ 
 $r^0 = f - \hat{A}u^0$ 
 $\tilde{r}^0 = SB^{-1}S^T r^0$ 
 $p^0 = \tilde{r}^0$ 
para  $k = 0, \dots, k_{max}$ 
     $\alpha = \langle \tilde{r}^k, r^k \rangle / \langle p^k, \hat{A}p^k \rangle$ 
     $u^{k+1} = u^k + \alpha p^k$ 
    teste de convergência
     $r^{k+1} = r^k - \alpha \hat{A}p^k$ 
     $\tilde{r}^{k+1} = SB^{-1}S^T r^{k+1}$ 
     $\beta = \langle \tilde{r}^{k+1}, r^{k+1} \rangle / \langle r^k, r^k \rangle$ 
     $p^{k+1} = \tilde{r}^{k+1} + \beta p^k$ 
próximo  $k$ 

```


3 O Método BPX

Bramble, Pasciak e Xu [9] apresentam um algoritmo, conhecido como método BPX, que também utiliza uma estrutura multi-nível. Procura dividir o espaço dos elementos finitos em subespaços formados por uma decomposição espectral induzida pelo produto interno usual L_2 e pelo processo de refinamento adotado. Logo, temos um paralelo muito próximo ao método das bases hierárquicas, onde ocorre a decomposição por frequências. Adotando-se BPX como pré-condicionador, podemos expressar o processo como uma mudança de bases. Isso permite previrmos que tal pré-condicionamento também possua a estrutura de um V-ciclo do método multigrid (como nas bases hierárquicas), o que elimina a necessidade do cômputo da matriz obtida na nova base.

Observa-se que o número de condição obtido pelo pré-condicionamento pode ser limitado por $\mathcal{O}(\ln h^{-1})$ para problemas com soluções suaves, por $\mathcal{O}(\ln^2 h^{-1})$ para problemas com singularidades do tipo “crack”, e $\mathcal{O}(\ln^3 h^{-1})$ para problemas com coeficientes descontínuos (conforme Bramble, Pasciak e Xu [9]). Logo, para o caso de dimensão três, temos uma estimativa mais significativa em relação às bases hierárquicas. Recentemente, Oswald [36, 37] estudou os métodos multinível sob a teoria de aproximação. Vários autores vêm seguindo esta linha e demonstram que o método BPX, para problemas com soluções suaves, possui número de condição $\mathcal{O}(1)$.

3.1 Projeção e suas estimativas

A priori, façamos uma análise do caso dois-dimensional, utilizando notação análoga à do capítulo anterior.

Sejam Q_k , $k = 0, \dots, J$, as projeções L_2

$$\begin{aligned} Q_k: V^h &\rightarrow \mathcal{S}_k \\ u &\mapsto Q_k u \text{ tal que } (Q_k u, v) = (u, v) \quad \forall v \in \mathcal{S}_k \end{aligned}$$

definidas sobre os espaços $\mathcal{S}_k$ criados durante o processo de refinamento. Dado $u \in V^h$, temos

$$u = Q_0 u + \sum_{k=1}^J (Q_k u - Q_{k-1} u) \quad (3.1)$$

Podemos considerar (3.1) como uma decomposição espectral de u (lembrando que Q_k são projeções) caso tomemos a seguinte decomposição para V^h

$$V_0 = \text{imagem}(Q_0) \text{ e } V_k = \text{imagem}(Q_k - Q_{k-1}) \subset \mathcal{S}_k, \quad k = 1, \dots, J$$

Logo

$$V^h = V_0 \oplus V_1 \oplus \dots \oplus V_J,$$

e temos uma decomposição ortogonal em relação ao produto interno L_2 . Pois, como $Q_k Q_l = Q_l Q_k = Q_l$ para $l \leq k$, temos $(Q_k - Q_{k-1})(Q_l - Q_{l-1}) = 0$, para $l \neq k$.

Tomemos as projeções (agora em relação ao produto interno $(\cdot, \cdot)_A$)

$$P_k: V^h \rightarrow S_k, \quad k = 0, \dots, J$$

$$u \mapsto P_k u \quad \text{tal que} \quad a(P_k u, v) = a(u, v) \quad \forall v \in S_k$$

e, novamente os operadores

$$A_k: V_k \rightarrow S_k, \quad k = 1, \dots, J$$

$$u \mapsto A_k u \quad \text{tal que} \quad (A_k u, v) = a(u, v), \quad \forall v \in S_k$$

com a notação $A = A_J$. Tais operadores podem ser interpretados como a restrição do problema original aos subespaços S_k . Observemos que para k fixo, dado $u \in V^h$, temos $(A_k P_k u, v) = a(P_k u, v) = a(u, v) = (A u, v) = (Q_k A u, v)$, $\forall v \in S_k$. Logo,

$$A_k P_k \equiv Q_k A \quad (3.2)$$

Uma interpretação para a criação do método BPX pode ser tomada através dos *métodos aditivos de Schwartz* (ASM), apresentados por Dryja e Widlund [24], quando aplicados ao problema (1.1), e indicados em Xu [48] como *decomposição do espaço e equações nos subespaços*:

Sejam

$V^h = V_0 \oplus V_1 \oplus \dots \oplus V_N$, uma decomposição de V^h onde $V_k \subset V^h$ subespaço vetorial,

$\tilde{P}_k: V^h \rightarrow V_k$ projeções definidas pela norma da energia

$$u_h \mapsto \tilde{P}_k u_h \quad \text{tal que} \quad a(\tilde{P}_k u_h, v) = a(u_h, v) \quad \forall v \in V_k$$

e tomemos o problema

$$\text{Determinar } u_h \in V^h \quad \text{tal que} \quad a(\tilde{P}_k u_h, v) = f(v), \quad \forall v \in V_k, k = 0, \dots, N. \quad (3.3)$$

Observemos que, se u_h resolve (1.1) temos, fixado k , $a(\tilde{P}_k u_h, v) = a(u_h, v) = f(v)$, $\forall v \in V_k$. Logo u_h resolve (3.3). Tomemos a formulação

$$\tilde{P} u = g \quad \text{onde} \quad \tilde{P} = \tilde{P}_0 + \dots + \tilde{P}_N$$

$$g = \sum_{i=0}^N g_i \quad \text{tal que} \quad a(g_i, v) = f(v) \quad \forall v \in V_i \quad (3.4)$$

Observe que, nessa formulação, o cálculo de g substitui o cálculo da solução exata u sobre todo V^h . O problema original foi separado em sub-problemas onde os termos g_i são agora calculados sobre os subespaços V_i exata ou aproximadamente, levando-nos a uma solução (exata ou aproximada) de $\tilde{P} u = g$.

Nessa divisão do problema em duas etapas é que distinguimos a criação de um pré-condicionador propriamente dito, e a resolução efetiva ("solver") do problema:

- O processo de decomposição do problema em sub-problemas mais a resolução (exata ou aproximada) desses sub-problemas constitui o pré-condicionador propriamente dito.

Por exemplo, no método das bases hierárquicas (HB) temos o processo de decomposição induzido por interpolações e a resolução do tipo Jacobi. Para HBMG temos a mesma decomposição, mas com a resolução do tipo Gauss Seidel por blocos. No caso de BPX, adotamos a decomposição induzida pelas projeções L_2 como em (3.1).

- O método de resolução da equação $\hat{P}u = g$ pode ser encarado como o método que está sendo pré-condicionado. Por exemplo, gradientes conjugados.

Temos que, se u_h resolve (3.3) então u_h resolve (3.4). Logo, garantida a unicidade de (3.4), obteríamos a equivalência entre (1.1) e (3.4).

Busquemos $\hat{P} = \hat{P}_0 + \dots + \hat{P}_J$ como resultado de um pré-condicionamento. Fixado $k > 0$ e dado $u \in V_k$, temos $(A_k \hat{P}_k u, v) = a(\hat{P}_k u, v) = a(u, v) = (Au, v) = ((Q_k - Q_{k-1})Au, v)$, $\forall v \in V_k$. Portanto

$$\hat{P}_k = A_k^{-1}(Q_k - Q_{k-1})A, \quad k > 0 \quad (3.5)$$

Analogamente $\hat{P}_0 = A_0^{-1}Q_0A$. Assim

$$\hat{P} = \left(A_0^{-1}Q_0 + \sum_{k=1}^J A_k^{-1}(Q_k - Q_{k-1}) \right) A \quad (3.6)$$

O cômputo de (3.6) é muito custoso. Busquemos uma aproximação computacionalmente mais eficiente para tal operador. Baseados na idéia de obtermos uma decomposição espectral para A , tomemos

$$\hat{P} = BA \quad B = \lambda_0^{-1}Q_0 + \sum_{k=1}^J \lambda_k^{-1}(Q_k - Q_{k-1}) = \sum_{k=0}^J \lambda_k^{-1}(Q_k - Q_{k-1}) \quad (3.7)$$

onde λ_k é o raio espectral de A_k e $Q_{-1} = 0$

Observemos que em formulações subseqüentes (Tong, Chan e Kuo [45], por exemplo) o termo A_0^{-1} não é substituído por λ_0^{-1} . Uma consideração para preservar-se A_0^{-1} (como em Pavarino [38] pag 23) é a de que o subespaço V_0 tem uma função muito especial: corresponde à malha mais grossa e proporciona transporte global de informação em cada iteração.

Dessa forma o operador B é tomado como uma aproximação para a inversa de A . E adotemos as projeções P_k que garantem a importante relação (3.2). Devemos estimar o número de condição $\kappa(BA)$. Para tal, seja a hipótese que, para $k = 0, \dots, J$, exista uma constante $C_1 > 0$, independente de J e h , tal que

$$\|(I - Q_{k-1})v\|_2^2 \leq C_1 \lambda_k^{-1} a(v, v) \quad \forall v \in V^h \quad (3.8)$$

onde $\|\cdot\|_2^2 = (\cdot, \cdot)$.

Teorema 5 *Se (3.8) é válida, temos*

$$C_1^{-1}(J+1)^{-1}a(v, v) \leq a(BAv, v) \leq (J+1)a(v, v), \quad \forall v \in V^h, \quad (3.9)$$

onde C_1 independe de J e h .

Demonstração. Dado $v \in V^h$, a segunda desigualdade em (3.9) pode ser obtida sem o uso da hipótese. Pela definição de B , temos

$$\begin{aligned} a(BAv, v) &= \sum_{k=0}^J \lambda_k^{-1} a((Q_k - Q_{k-1})Av, v) = \sum_{k=0}^J \lambda_k^{-1} ((Q_k - Q_{k-1})Av, Av) = \\ &= \sum_{k=0}^J \lambda_k^{-1} ((Q_k - Q_{k-1})Av, (Q_k - Q_{k-1})Av) = \\ &= \sum_{k=0}^J \lambda_k^{-1} \|(Q_k - Q_{k-1})Av\|_2^2 \leq \sum_{k=0}^J \lambda_k^{-1} \|Q_k Av\|_2^2 \end{aligned}$$

Por (3.2) e pela definição de λ_k

$$\begin{aligned} a(BAv, v) &\leq \sum_{k=0}^J \lambda_k^{-1} (Q_k Av, Q_k Av) = \sum_{k=0}^J \lambda_k^{-1} (A_k P_k v, A_k P_k v) \leq \\ &\leq \sum_{k=0}^J (P_k v, A_k P_k v) = \sum_{k=0}^J a(P_k v, P_k v) = \\ &= \sum_{k=0}^J \|P_k v\|_A^2 \leq \|v\|_A^2 \sum_{k=0}^J \|P_k\|_A^2 \end{aligned}$$

Como $\|P_k\|_A = 1$, temos

$$a(BAv, v) \leq (J+1)a(v, v), \quad \forall v \in V^h$$

Para a primeira desigualdade, temos

$$\begin{aligned} a(v, v) &= \sum_{k=0}^J ((Q_k - Q_{k-1})Av, v) = \\ &= \sum_{k=0}^J ((Q_k - Q_{k-1})Av, (I - Q_{k-1})v + Q_{k-1}v) = \\ &= \sum_{k=0}^J \{((Q_k - Q_{k-1})Av, (I - Q_{k-1})v) + ((Q_k - Q_{k-1})Av, Q_{k-1}v)\} \end{aligned}$$

Como

$$\begin{aligned} ((Q_k - Q_{k-1})Av, Q_{k-1}v) &= (Q_k Av, Q_{k-1}v) - (Q_{k-1}Av, Q_{k-1}v) = \\ &= (Av, Q_{k-1}v) - (Av, Q_{k-1}v) = 0, \end{aligned}$$

aplicando a desigualdade de Cauchy-Schwartz e a hipótese (3.8), obtemos

$$\begin{aligned} a(v, v) &\leq \sum_{k=0}^J \|(Q_k - Q_{k-1})Av\|_2 \|(I - Q_{k-1})v\|_2 \leq \\ &\leq C_1^{1/2} \sum_{k=0}^J \lambda_k^{-1/2} a(v, v)^{1/2} \|(Q_k - Q_{k-1})Av\|_2. \end{aligned}$$

Logo

$$a(v, v)^{1/2} \leq C_1^{1/2} \sum_{k=0}^J \lambda_k^{-1/2} \|(Q_k - Q_{k-1})Av\|_2.$$

Como $(\sum_{l=1}^m b_l)^2 \leq m \sum_{l=1}^m b_l^2$, temos

$$a(v, v) \leq C_1(J+1) \sum_{k=0}^J \lambda_k^{-1} \|(Q_k - Q_{k-1})Av\|_2^2 = C_1(J+1)a(BAv, v).$$

□

Assim, a hipótese (3.8) implica $\kappa(BA) \leq C_1(J+1)^2$.

Reescrevendo $B = \sum_{k=0}^J (\lambda_k^{-1} Q_k - \lambda_{k+1}^{-1} Q_{k-1}) = \sum_{k=0}^{J-1} (\lambda_k^{-1} - \lambda_{k+1}^{-1}) Q_k + \lambda_J^{-1} Q_J$, observamos que, se $\{\lambda_k\}_{k=0}^J$ satisfizer a condição de crescimento

$$\lambda_{k+1} \geq \delta \lambda_k, \quad \delta > 1, \quad (3.10)$$

podemos definir

$$\hat{B} = \sum_{k=0}^J \lambda_k^{-1} Q_k$$

e obtemos

$$(1 - \delta^{-1})(\hat{B}v, v) \leq (Bv, v) \leq (\hat{B}v, v), \quad \forall v \in V^h, \quad (3.11)$$

pois

$$\begin{aligned} (Bv, v) &= \sum_{k=0}^{J-1} (\lambda_k^{-1} - \lambda_{k+1}^{-1})(Q_k v, v) + \lambda_J^{-1}(Q_J v, v) \geq \\ &\geq \sum_{k=0}^{J-1} \lambda_k^{-1}(1 - \delta^{-1})(Q_k v, v) + \lambda_J^{-1}(Q_J v, v) \geq \\ &(1 - \delta^{-1})(\sum_{k=0}^{J-1} \lambda_k^{-1}(Q_k v, v) + \lambda_J^{-1}(Q_J v, v)) = (1 - \delta^{-1})(\hat{B}v, v) \end{aligned}$$

e

$$(Bv, v) = \sum_{k=0}^J \lambda_k^{-1}(Q_k v, v) - \sum_{k=0}^J \lambda_k^{-1}(Q_{k-1} v, v) \leq (\hat{B}v, v).$$

Logo, por (3.11), B e $\hat{B}$ são espectralmente equivalentes e, portanto, o número de condição $\kappa(\hat{B}A)$ é da ordem $\mathcal{O}((J+1)^2)$.

Conceitualmente, a passagem de B para $\hat{B}$ apresenta uma diferença em relação ao método das bases hierárquicas, pois aqui utilizamos a condição (3.10).

Uma forma mais geral para o operador $\hat{B}$ é considerada ao substituirmos o termo λ_k^{-1} por um operador $R_k: \mathcal{S}_k \rightarrow \mathcal{S}_k$ simétrico positivo definido, obtendo

$$\mathcal{B} = \sum_{k=0}^J R_k Q_k : \quad (3.12)$$

Tal formulação, apresentada em Bramble, Pasciak e Xu [9] e Xu [48], pode ser tomada como um V-ciclo do método multigrid. Em Bramble e Pasciak [10] é considerada uma formulação do método multigrid baseada na mesma notação e são apresentadas hipóteses para garantia da convergência do método. O operador R_k cumpre o papel da suavização, com a principal diferença que é aplicado sobre o resíduo calculado na malha mais fina e apenas na pré relaxação. Logo, tomemos o método BPX como um V-ciclo multigrid apenas com pré relaxações. Isso nos leva a considerar a seguinte hipótese adotada na análise de métodos multigrid:

$$C_2 \frac{\|u\|_2^2}{\lambda_k} \leq (R_k u, u) \leq C_3 (A_k^{-1} u, u), \quad \forall u \in \mathcal{S}_k \quad (3.13)$$

onde C_2 e C_3 são constantes positivas independentes de J .

A partir de $\mathcal{B}$ definido em (3.12), temos o seguinte corolário.

Corolário 1 *Assumindo as hipóteses (3.8) e (3.13), temos*

$$C_1^{-1} C_2 (J+1)^{-1} a(v, v) \leq a(\mathcal{B} A v, v) \leq C_3 (J+1) a(v, v), \quad \forall v \in V^h. \quad (3.14)$$

Demonstração. Para provar a segunda desigualdade de (3.14), dado $v \in V^h$, temos por (3.13) e (3.2)

$$\begin{aligned} a(BAv, v) &= \sum_{k=0}^J a(R_k Q_k Av, v) = \sum_{k=0}^J a(R_k Q_k Av, P_k v) = \sum_{k=0}^J (R_k A_k P_k v, A_k P_k v) \leq \\ &\leq C_3 \sum_{k=0}^J (P_k v, A_k P_k v) = C_3 \sum_{k=0}^J a(P_k v, P_k v) \leq C_3(J+1)a(v, v) \end{aligned}$$

Pelo teorema 5

$$\begin{aligned} C_1^{-1}(J+1)^{-1}a(v, v) &\leq a(BAv, v) = a(\sum_{k=0}^{J-1}(\lambda_k^{-1} - \lambda_{k+1}^{-1})Q_k Av + \lambda_J^{-1}Q_J Av, v) = \\ &= \sum_{k=0}^{J-1}(\lambda_k^{-1} - \lambda_{k+1}^{-1})(Q_k Av, Av) + \lambda_J^{-1}(Q_J Av, Av) = \\ &= \sum_{k=0}^{J-1}(\lambda_k^{-1} - \lambda_{k+1}^{-1})(Q_k Av, Q_k Av) + \lambda_J^{-1}(Q_J Av, Q_J Av) \leq \\ &\leq \sum_{k=0}^J \lambda_k^{-1} \|Q_k Av\|_2^2 \end{aligned}$$

Logo, por (3.13)

$$\begin{aligned} C_1^{-1}C_2(J+1)^{-1}a(v, v) &\leq \sum_{k=0}^J \frac{C_2}{\lambda_k} \|Q_k Av\|_2^2 \leq \sum_{k=0}^J (R_k Q_k Av, Q_k Av) = \\ &= \sum_{k=0}^J (R_k Q_k Av, Av) = a(\sum_{k=0}^J R_k Q_k Av, v) \end{aligned}$$

o que prova a primeira desigualdade em (3.14). $\square$

Dessa forma, as hipóteses (3.8) e (3.13) garantem a majoração para o número de condição $\kappa(BA) \leq \frac{C_1 C_3}{C_2} (J+1)^2$.

Uma observação é que o operador $R_k = \lambda_k^{-1} I$ original, cumpre (3.13).

Para a estimativa inferior em (3.14) podemos enunciar uma hipótese alternativa também utilizada em métodos multigrid: assumamos que, para $\alpha \in]0, 1]$ fixo, exista uma constante C_4 constante independente de $k = 0, \dots, J$, tal que

$$a((I - P_{k-1})v, v) \leq (C_4 \lambda_k^{-1} \|A_k v\|_2^2)^\alpha a(v, v)^{1-\alpha}, \quad \forall v \in \mathcal{S}_k \quad (3.15)$$

onde $P_{-1} = 0$.

Um comentário associado às hipóteses (3.13) e (3.15) (ou (3.4) (3.3) de Bramble e Pasciak [10], respectivamente) pode ser feito ao tomarmos o método BPX como um multigrid somente com pré-relaxações. Hackbusch [27], cap 6, apresenta uma análise de convergência da iteração de um operador dois-grid. Basicamente é feita a separação do operador em duas componentes: a componente de suavização, e a componente de aproximação constituída por projeção, correção e interpolação). Considerando o caso multi-níveis, como na representação (1.10), pag. 11,

$$M_k = (A_k^{-1} - I_{k-1}^k M_{k-1}^{-1} P_{k-1})(A_k R_k), \quad \text{onde } M_{k-1}^{-1} \approx A_{k-1}^{-1},$$

sobre tais componentes são impostas as seguintes hipóteses:

- *Propriedade de suavização:* visa controlar a norma de $A_k R_k$, ou seja, medir o quanto o “suavizador” R_k aproxima a inversa de A_k (que é a restrição do problema variacional à malha $\mathcal{T}_k$).

- *Propriedade de aproximação*: mede o quanto o operador $I_{k-1}^k M_{k-1}^{-1} P_{k-1}$ utilizado para a correção na malha mais grossa $\mathcal{T}_{k-1}$ aproxima a solução do problema em $\mathcal{T}_k$.

A hipótese (3.13) se relaciona com a *propriedade de suavização* pois,

$$(3.13) \Rightarrow C_2 \lambda_k^{-1} \leq \frac{(R_k u, u)}{(u, u)} \leq C_3 \frac{(A_k^{-1} u, u)}{(u, u)}, \forall u \in \mathcal{S}_k \setminus \{0\} \Rightarrow C_2 \lambda_k^{-1} \leq \rho(R_k) \leq C_3 \rho(A_k^{-1}), \forall k.$$

Lembrando que λ_k é o maior autovalor de A_k , e portanto corresponde às altas frequências na malha $\mathcal{T}_k$, o que essa relação impõe é o quanto o operador de suavização R_k aproxima a ação de A_k^{-1} principalmente nas altas frequências de A_k .

Analogamente, a hipótese (3.15) está associada à *propriedade de aproximação* pois, pela definição de A_k , temos

$$(3.15) \Rightarrow \frac{a((I - P_{k-1})v, v)}{a(v, v)} \leq \left(\frac{C_4}{\lambda_k} \frac{(A_k v, A_k v)}{a(v, v)} \right)^\alpha = \left(\frac{C_4}{\lambda_k} \frac{a(A_k v, v)}{a(v, v)} \right)^\alpha, \forall v \in \mathcal{S}_k \setminus \{0\}$$

o que é “quase”

$$\rho(I - P_{k-1}) \leq \left(\frac{C_4}{\lambda_k} \rho(A_k) \right)^\alpha = C_4^\alpha$$

não o é pois o domínio de $I - P_k$ é $\mathcal{S}_J = V^h$. Logo, tal relação impõe uma condição sobre como atua a projeção $I - P_{k-1}: \mathcal{S}_J \rightarrow \mathcal{S}_{k-1}^{\perp A}$ isto é, limita a ação sobre as componentes pertencentes ao complemento ortogonal de $\mathcal{S}_{k-1}$ (em relação à energia $a(\cdot, \cdot)$). Caso possamos observar que $\mathcal{S}_{k-1}^{\perp A}$ tenha grande relação com as componentes de alta frequência da malha $\mathcal{T}_k$, a hipótese (3.15) indicaria que não estamos gerando muita ação (interferência) sobre as altas frequências no processo de correção na malha mais grossa.

Em métodos de elementos finitos, a hipótese (3.15) pode ser provada utilizando-se regularidade dos problemas elípticos contínuos e propriedades de aproximação dos espaços $\mathcal{S}_k$. Bramble, Pasciak [10] e Bank e Dupont [3] obtêm (3.15) ao assumir a seguinte regularidade elíptica: existe uma constante C , tal que

$$\|u\|_{H^{1+\alpha}(\Omega)} \leq C \|f\|_{H^{\alpha-1}(\Omega)} \quad (3.16)$$

para toda solução u da formulação variacional contínua. Onde $H^s(\Omega)$ são normas de Sobolev de ordem s .

Podemos enunciar o seguinte teorema.

Teorema 6 *Assumindo as hipóteses (3.13) e (3.15), temos*

$$C_2 C_4^{-1} (J+1)^{1-1/\alpha} a(v, v) \leq a(BA v, v) \leq C_3 (J+1) a(v, v), \forall v \in V^h \quad (3.17)$$

Demonstração. Seja $v \in V^h$, podemos reescrever

$$v = \sum_{k=0}^J (P_k - P_{k-1})v .$$

Assim

$$\begin{aligned} a(v, v) &= a\left(\sum_{k=0}^J (P_k - P_{k-1})v, v\right) = \sum_{k=0}^J a((P_k - P_{k-1})v, v) = \\ &= \sum_{k=0}^J a((P_k - P_{k-1})v, P_k v) = \sum_{k=0}^J a((I - P_{k-1})P_k v, P_k v) . \end{aligned}$$

Por (3.15) e como $\|P\|_A = 1$,

$$\begin{aligned} a(v, v) &\leq C_4^\alpha \sum_{k=0}^J \left(\lambda_k^{-1} \|A_k P_k v\|^2\right)^\alpha a(P_k v, P_k v)^{1-\alpha} \leq \\ &\leq C_4^\alpha \sum_{k=0}^J \left(\lambda_k^{-1} \|A_k P_k v\|^2\right)^\alpha a(v, v)^{1-\alpha} , \end{aligned}$$

isto é,

$$a(v, v)^\alpha \leq C_4^\alpha \sum_{k=0}^J \left(\lambda_k^{-1} \|A_k P_k v\|^2\right)^\alpha .$$

Como

$$(3.13) \Rightarrow \lambda_k^{-1} \|u\|^2 \leq C_2^{-1} (R_k u, u) \quad \forall u \in \mathcal{S}_k \Rightarrow \lambda_k^{-1} \|A_k u\|^2 \leq C_2^{-1} (R_k A_k u, A_k u) \quad \forall u \in \mathcal{S}_k ,$$

obtemos

$$a(v, v)^\alpha \leq (C_2^{-1} C_4)^\alpha \sum_{k=0}^J (R_k A_k P_k v, A_k P_k v)^\alpha , \quad \forall v \in V^h .$$

Aplicando a desigualdade de Holder para um seqüência $\{b_k\}$ de números não negativos, obtemos

$$\sum_{k=0}^J b_k^\alpha \leq (J+1)^{1-\alpha} \left(\sum_{k=0}^J b_k \right)^\alpha ,$$

logo

$$\begin{aligned} a(v, v)^\alpha &\leq (C_2^{-1} C_4)^\alpha (J+1)^{1-\alpha} \left(\sum_{k=0}^J (R_k A_k P_k v, A_k P_k v) \right)^\alpha \leq \\ &\leq (C_2^{-1} C_4)^\alpha (J+1)^{1-\alpha} a(\mathcal{B}A v, v)^\alpha . \end{aligned}$$

O que prova a primeira desigualdade em (3.17). A segunda desigualdade já foi provada no corolário 1. $\square$

Tal resultado indica que (3.13) e (3.15) implicam em $\kappa(\mathcal{B}A) \leq \frac{C_3 C_4}{C_2} (J+1)^{1/\alpha}$.

3.2 Aplicação e considerações

Para a aplicação do método resta a escolha de R_k . Consideremos $R_k \equiv \lambda_k^{-1} I$. Para $\{\varphi_l^k\}_{l \in \mathcal{J}_k}$ base ortogonal de $\mathcal{S}_k$, obtemos

$$R_k u = \lambda_k^{-1} u = \lambda_k^{-1} \sum_{l \in \mathcal{J}_k} (u, \varphi_l^k) \varphi_l^k, \quad \forall u \in \mathcal{S}_k .$$

Mas normalmente seria difícil determinar tal base e obter os valores λ_k . Analisando o caso particular $a(u, v) = (\nabla u, \nabla v)$ (isto é, $\Delta u = f$), temos a seguinte estimativa (conforme Xu [48] em 5.8): existem constantes $\gamma_0, \gamma_1 > 0$, independentes de h_k , tais que

$$\gamma_0 h_k^d |v|^2 \leq \langle A_k v, v \rangle \leq \gamma_1 h_k^{d-2} |v|^2, \quad \forall v \in \mathbb{R}^n$$

onde d é a dimensão de Ω . O que implica

$$\gamma_0 h_k^d \leq \lambda_k \leq \gamma_1 h_k^{d-2}$$

e nos induz a tomarmos

$$R_k v = h_k^{2-d} \sum_{l \in \mathcal{J}_k} (v, \varphi_l^k) \varphi_l^k.$$

Como conjunto gerador de V^h tomemos as “bases” multi-nível nodais $\{\hat{\psi}_l^k\}_{l \in \mathcal{J}_k}^{k=0, \dots, J}$ definidas pela união das bases nodais de cada espaço $\mathcal{S}_k$, como na figura 2.1 (a), pag. 17. Num abuso de notação vamos nos referir a este conjunto gerador como base multi-nível nodal. Temos

$$R_k Q_k v = h_k^{2-d} \sum_{l \in \mathcal{J}_k} (Q_k v, \hat{\psi}_l^k) \hat{\psi}_l^k = h_k^{2-d} \sum_{l \in \mathcal{J}_k} (v, \hat{\psi}_l^k) \hat{\psi}_l^k$$

pois $\hat{\psi}_l^k \in \mathcal{S}_k$. Assim, uma tentativa para um pré-condicionador seria

$$M^{-1} v = \sum_{k=0}^J h_k^{2-d} \sum_{l \in \mathcal{J}_k} (v, \hat{\psi}_l^k) \hat{\psi}_l^k \quad (3.18)$$

Ao adotar (3.18), devemos verificar as hipóteses (3.13) e (3.15).

Assumindo um processo de triangularização quase-uniforme, a hipótese (3.13) é resultado de equivalências entre normas (vide Bramble, Pasciak e Xu [9] ou Xu [48]). Já (3.15), como foi comentado, é resultado de regularidade elíptica do tipo (3.16). Caso mantida a consideração feita em (3.7) da não substituição de A_0^{-1} por λ_0^{-1} , teríamos

$$M^{-1} v = A_0^{-1} Q_0 v + \sum_{k=1}^J h_k^{2-d} \sum_{l \in \mathcal{J}_k} (v, \hat{\psi}_l^k) \hat{\psi}_l^k. \quad (3.19)$$

Novamente podemos aplicar o nosso pré-condicionador a partir da formulação nodal do problema. Matricialmente obtemos M^{-1} (vide Tong, Chan e Kuo [45]), a ser aplicado sobre a representação nodal de A , da seguinte forma:

$$M^{-1} = G D^{-1} G^T \quad (3.20)$$

onde

$G: \mathbb{R}^{n_J + \dots + n_0} \rightarrow \mathbb{R}^{n_J}$ corresponde à mudança da base multi-nível nodal para a base nodal,
 $G^T: \mathbb{R}^{n_J} \rightarrow \mathbb{R}^{n_J + \dots + n_0}$ corresponde a sua transposta e
 $D^{-1} = \text{bloco diagonal } (h_J^{2-d} I_{n_J}, h_{J-1}^{2-d} I_{n_{J-1}}, \dots, h_1^{2-d} I_{n_1}, h_0^{2-d} I_{n_0} \text{ ou } A_0^{-1})$

Uma formulação algorítmica para G (vide Elman e Guo [25]) pode ser obtida de forma análoga ao método das bases hierárquicas.

Seja T_k a matriz que representa os coeficientes sobre as bases nodais $\{\hat{\psi}_i^k\}_{i \in \mathcal{J}_k}$ de $\mathcal{S}_k$, $k < J$, em termos dos coeficientes sobre a base nodal $\{\hat{\psi}_l^J\}_{l \in \mathcal{J}_J}$ de $\mathcal{S}_J$:

$$\begin{bmatrix} \hat{\psi}_1^k \\ \vdots \\ \hat{\psi}_{n_k}^k \end{bmatrix} = T_k \begin{bmatrix} \hat{\psi}_1^J \\ \vdots \\ \hat{\psi}_{n_J}^J \end{bmatrix} .$$

Dado $v = \sum_{i=1}^{n_J} v_i \hat{\psi}_i^J$, temos por (3.20)

$$M^{-1} = h_J^{2-d} I_{n_J} + \sum_{k=0}^{J-1} h_k^{2-d} T_k^T T_k \quad (\text{IBPX}) \quad (3.21)$$

ou

$$M^{-1} = h_J^{2-d} I_{n_J} + \sum_{k=1}^{J-1} h_k^{2-d} T_k^T T_k + h_0^{2-d} T_0^T A_0^{-1} T_0 \quad (\text{CBPX}) \quad (3.22)$$

Como no final da seção 2.3, a possível escolha entre resolução completa ou não, na malha mais grossa, será diferenciada pela nomenclatura (Elman e Guo [25]) *método BPX completo* (CBPX) e *método BPX incompleto* (IBPX).

Seja T_k^{k+1} a matriz que representa a base nodal $\{\hat{\psi}_i^k\}_{i \in \mathcal{J}_k}$ de $\mathcal{S}_k$, em termos da base nodal $\{\hat{\psi}_l^{k+1}\}_{l \in \mathcal{J}_{k+1}}$ de $\mathcal{S}_{k+1}$:

$$\begin{bmatrix} \hat{\psi}_1^k \\ \vdots \\ \hat{\psi}_{n_k}^k \end{bmatrix} = T_k^{k+1} \begin{bmatrix} \hat{\psi}_1^{k+1} \\ \vdots \\ \hat{\psi}_{n_{k+1}}^{k+1} \end{bmatrix} . \quad (3.23)$$

Temos

$$T_k = T_k^{k+1} \cdots T_J^{J+1} .$$

Definindo $\hat{S}_k = (T_k^{k+1})^T$, obtemos

$$T_k^T = (T_J^{J+1})^T \cdots (T_k^{k+1})^T = \hat{S}_J \cdots \hat{S}_{k+1}$$

e

$$T_k = \hat{S}_{k+1}^T \cdots \hat{S}_J^T ,$$

podendo reescrever (3.21) e (3.22) respectivamente como

$$M^{-1} = h_J^{2-d} I_{n_J} + \sum_{k=1}^{J-1} h_{k-1}^{2-d} (\hat{S}_J \cdots \hat{S}_k) (\hat{S}_k^T \cdots \hat{S}_J^T) \quad (3.24)$$

$$M^{-1} = h_J^{2-d} I_{n_J} + \sum_{k=2}^{J-1} h_{k-1}^{2-d} (\hat{S}_J \cdots \hat{S}_k) (\hat{S}_k^T \cdots \hat{S}_J^T) + h_0^{2-d} \hat{S}_J \cdots \hat{S}_1 A_0^{-1} \hat{S}_1^T \cdots \hat{S}_J^T \quad (3.25)$$

Tomando $\mathcal{I}_k: S_J \rightarrow S_k$ o operador de interpolação tal que

$$\mathcal{I}_k u(x) = u(x), \quad \forall x \in \mathcal{N}_k,$$

como $\mathcal{I}_{k-1} \in S_{k-1}$ e $\mathcal{I}_{k-1} u(x_i) = u(x_i) = u_i \quad \forall i \in \mathcal{J}_{k-1}$, temos

$$\mathcal{I}_{k-1} u = \sum_{i \in \mathcal{J}_{k-1}} u_i \hat{\psi}_i^{k-1} = (\hat{\psi}_1^{k-1}, \dots, \hat{\psi}_{n_{k-1}}^{k-1}) \begin{bmatrix} u_1 \\ \vdots \\ u_{n_{k-1}} \end{bmatrix}$$

e, por (3.23),

$$\mathcal{I}_{k-1} u = (\hat{\psi}_1^{k-1}, \dots, \hat{\psi}_{n_{k-1}}^{k-1}) (T_{k-1}^k)^T = \begin{bmatrix} u_1 \\ \vdots \\ u_{n_{k-1}} \end{bmatrix} = (\hat{\psi}_1^{k-1}, \dots, \hat{\psi}_{n_{k-1}}^{k-1}) \hat{S}_k \begin{bmatrix} u_1 \\ \vdots \\ u_{n_{k-1}} \end{bmatrix}.$$

Dessa forma, os valores de $\mathcal{I}_{k-1} u$ sobre os vértices de $\mathcal{N}_k$ são dados por

$$\hat{S}_k \begin{bmatrix} u_1 \\ \vdots \\ u_{n_{k-1}} \end{bmatrix}.$$

Como $\mathcal{I}_{k-1} u(x_i) = u_i \quad \forall i \in \mathcal{J}_{k-1}$, $\hat{S}_k$ possui a estrutura

$$\hat{S}_k = \begin{pmatrix} I_{n_0} & & \\ & \ddots & \\ & & I_{n_{k-1}} \\ \hat{R}_0 & \dots & \hat{R}_{k-1} \end{pmatrix}.$$

Logo, os operadores $\hat{R}_l$, $l = 0, \dots, k-1$ coincidem com os operadores de interpolação R apresentados em (2.14), pag 23, do método HB. Estruturalmente $\hat{S}_k$ corresponde a um sub-bloco das matrizes de mudança S_k vistas anteriormente. Analogamente à implementação de um algoritmo para S_k , tomados $\hat{\psi}_i^J(x_i) = X(i)$, $i = 1, \dots, n_J$, obtemos $\hat{S}_k$ expressa por

Cômputo de $\hat{S}_k x$
para $i \in \mathcal{M}_k = \{i : x_i \in \mathcal{N}_k \setminus \mathcal{N}_{k-1}\}$
 $X(i) = (X(I1(i)) + X(I2(i)))/2$
próximo i

E para

$$\hat{S}_k^T = \begin{pmatrix} I_{n_0} & & & R_0^T \\ & \ddots & & \vdots \\ & & I_{n_{k-1}} & R_{k-1}^T \end{pmatrix},$$

temos

Cômputo de $\hat{S}_k^T x$
 para $i \in \mathcal{M}_k$
 $X(I1(i)) = X(I1(i)) + X(i)/2$
 $X(I2(i)) = X(I2(i)) + X(i)/2$
 próximo i

Em suma, à formulação (3.20) correspondem $G = \hat{S}_1 \cdots \hat{S}_J$ e $G^T = \hat{S}_J^T \cdots \hat{S}_1^T$. Reescrevendo (3.24)

$$M^{-1} = h_J^{2-d} I_{n_J} + \hat{S}_J \{ h_{J-1}^{2-d} I_{n_{J-1}} + \hat{S}_{J-1} \{ h_{J-2}^{2-d} I_{n_{J-2}} + \hat{S}_{J-2} \{ \cdots \} \hat{S}_{J-2}^T \} \hat{S}_{J-1}^T \} \hat{S}_J^T$$

obtemos a implementação para o método IBPX. Bastando utilizar a estrutura do método HB (pag. 22), com o pré-condicionador M descrito como

pré-condicionamento IBPX, retorna $w^J = M^{-1}v$
 $v^J = v$
 para $k = J$ até 1, decrescendo
 $v^{k-1} = \hat{S}_k^T v^k$
 próximo k
 $w^0 = h_0^{2-d} v^0$
 para $k = 1$ até J
 $w^k = h_k^{2-d} v^k + \hat{S}_k w^{k-1}$
 próximo k

Observemos que, em se tratando de base multi-nível nodal, os vários termos v^k são distintos, isto é, $(v^J, v^{J-1}, \dots, v^0)^T \in \mathbb{R}^{n_J + \dots + n_0}$. Logo devem ser armazenados em posições distintas de memória. Em contrapartida, os termos w^k podem ser computados sobre um único vetor $w \in \mathbb{R}^{n_J}$, pois esses são formados sucessivamente, partindo-se dos coeficientes $i \in \mathcal{N}_0$ referentes à malha mais grossa, até os coeficientes $i \in \mathcal{N}_J \setminus \mathcal{N}_{J-1}$ referentes apenas à malha mais fina. Reescrevendo em termos de $\hat{S}_k$ e $\hat{S}_k^T$, obtemos a formulação computacional

pré-condicionamento IBPX, retorna $W = M^{-1}V$

para $i \in \mathcal{N}_J$

$$v^J(i) = V(i)$$

próximo i

para $k = J$ até 1, decrescendo

para $i \in \mathcal{M}_k$

$$v^{k-1}(I1(i)) = v^k(I1(i)) + v^k(i)/2$$

$$v^{k-1}(I2(i)) = v^k(I2(i)) + v^k(i)/2$$

próximo i

próximo k

para $i \in \mathcal{N}_0$

$$W(i) = h_0^{2-d} v^0(i)$$

próximo i

para $k = 1$ até J

para $i \in \mathcal{M}_k$

$$W(i) = (W(I1(i)) + W(I2(i)))/2$$

próximo i

para $i \in \mathcal{N}_k$

$$W(i) = h_k^{2-d} v^k(i) + W(i)$$

próximo i

próximo k

Uma segunda linha para introduzir-se o pré-condicionamento por BPX, é apresentada em Yserentant [51] onde é feita uma análise simultânea entre BPX e HB. As estimativas feitas sobre as projeções Q_k baseiam-se em operadores de média

$$M_k: L_2(\Omega) \rightarrow \mathcal{S}_k$$

$$u \mapsto M_k u = \sum_{i \in \mathcal{J}_k} \frac{(u, \hat{\psi}_i^k)}{(1, \hat{\psi}_i^k)} \hat{\psi}_i^k$$

com a diferença que os produtos internos possuem um peso definido pela triangularização inicial. A principal propriedade de M_k é que localmente reproduzem funções constantes, isto é, para todo triângulo $T \in \mathcal{T}_k$, seja $U(T, k) = \cup\{T' \in \mathcal{T}_k : T \cap T' \neq \emptyset\}$ a união dos triângulos $T' \in \mathcal{T}_k$ que intersectam T . Então, para $u|_{U(T, k)} = \alpha$, α constante e $T \cap \partial\Omega = \emptyset$, temos $M_k u|_T = \alpha$.

Obtém-se a estimativa: existem constantes positivas μ_1 e μ_2 , que dependem apenas de uma limitação inferior para os ângulos internos dos triângulos em $\mathcal{T}_k$, tais que

$$\frac{1}{\mu_2} (M_k r, r) \leq (Q_k r, r) \leq \frac{1}{\mu_1} (M_k r, r) \quad \forall r \in V^h$$

e é adotado o pré-condicionador

$$C_X = A_0^{-1} Q_0 + \sum_{k=1}^J 4^{-k} M_k$$

onde M_k faz o papel dos termos R_k da formulação geral. Dessa forma, temos

$$C_X r = A_0^{-1} Q_0 r + \sum_{k=1}^J 4^{-k} \sum_{i \in \mathcal{J}_k} \frac{(r, \hat{\psi}_i^k)}{(1, \hat{\psi}_i^k)} \hat{\psi}_i^k$$

que pode ser comparado à formulação do pré-condicionador HB apresentado no mesmo texto

$$C_H r = A_0^{-1} Q_0 r + \sum_{k=1}^J 4^{-k} \sum_{i \in \mathcal{J}_k \setminus \mathcal{J}_{k-1}} \frac{(r, \hat{\psi}_i^k)}{(1, \hat{\psi}_i^k)} \hat{\psi}_i^k.$$

Novamente cabe a observação feita nas bases hierárquicas sobre a possibilidade de escolha dos pesos d_i^k (no caso acima, $d_i^k = 4^k(1, \hat{\psi}_i^k)$).

3.3 Estimativas recentes

Os trabalhos de Oswald [36, 37] e Dahmen e Kunoth [18] apresentam a equivalência

$$\frac{1}{K_1} |||u|||^2 \leq ||u||^2 \leq K_2 |||u|||^2 \quad (3.26)$$

com constantes K_1 e K_2 independento do número J de níveis de refinamento. Utilizando (3.26) demonstram que o número de condição para o método BPX é $\mathcal{O}(1)$. Tal resultado é baseado no fato que o espaço de Sobolev $W^{1,2}(\Omega)$ coincide com certos espaços de Besov (vide Devore e Popov [19]) e as correspondentes normas são equivalentes. Também utilizam o fato que as propriedades de aproximação, para as projeções do tipo L_2 sobre espaços de elementos finitos, podem ser descritas pelo *módulo de suavização de segunda ordem* envolvido na definição dos espaços de Besov. Especificamente, tomam o operador de diferenças

$$(\Delta_h^l f)(x) = \sum_{m=0}^l \binom{l}{m} (-1)^{l-m} f(x + mk), \quad x, h \in \mathbb{R}^s,$$

e, para $1 \leq p \leq \infty$, o L_p -módulo de suavização de ordem l

$$\omega_l(f, t, \Omega)_p = \sup_{|h|_2 \leq t} \|\Delta_h^l f\|_{p; \Omega_{l,h}}$$

onde $\Omega_{l,h} = \{x \in \Omega : x + mk \in \Omega, m = 0, \dots, l\}$ e $|\cdot|_2$ é a norma euclidiana em $\mathbb{R}^s$. Dada uma representação em multiníveis, como em (3.1),

$$u = Q_0 u + \sum_{i=1}^J (Q_i u - Q_{i-1} u)$$

associada às projeções Q_i e ao refinamento em questão, são formuladas algumas estimativas:

- *estimativa de Berstein*: existe $\gamma > k$ (k ordem do espaço de Sobolev em questão) tal que

$$\omega_{k+1}(u, t, \Omega)_2 \leq c_1 (\min\{1, t2^J\})^\gamma \|u\|_2 \quad \forall u \in \mathcal{S}_J$$

verifica-se para alguma constante c_1 independente de J .

- *estimativa de Jackson*: existe um inteiro $\gamma > k$, tal que

$$\inf_{u \in \mathcal{S}_J} \|f - g\|_2 \leq c_2 2^{-m_l} \|f\|_{l,2} \quad \forall f \in W^{l,2}(\Omega)$$

verifica-se para alguma constante c_2 independente de J .

Cumpridas tais estimativas, o pré-condicionador C^{-1} definido sobre $\mathcal{S}_J$ por

$$(C^{-1}u, u') = (Q_0 u, Q_0 u') + \sum_{i=1}^J 2^{2ik} ((Q_i - Q_{i-1})u, (Q_i - Q_{i-1})u')$$

satisfaz

$$\kappa(C^{1/2} A C^{1/2}) = \mathcal{O}(1), \quad J \rightarrow \infty. \quad (3.27)$$

Essas duas estimativas, comentadas acima, são aplicadas no caso de elementos finitos lineares e refinamentos por triangularizações uniformes. Para o caso não uniforme, a estimativa de Jackson não é obtida, mas supondo um processo de refinamento como na página 19, é possível obter-se (3.27).

Bramble e Pasciak [11] apresentam uma prova diferenciada de Oswald. Utilizam as propriedades das projeções L_2 e $W^{1,2}$ (vide também Xu [48] e Zhang [52]). Na verdade, visam a análise do método multigrid, em formulações como método aditivo ou método multiplicativo. Mas lembremos que o ferramental utilizado na formulação aditiva é basicamente o mesmo utilizado para o método BPX (vide Bramble, Pasciak e Xu [9] e Xu [48]). Bramble e Pasciak assumem hipóteses diferentes das tradicionalmente adotadas em métodos multigrid. A primeira é uma formulação mais fraca para a hipótese de regularidade e aproximação (3.15) com $\alpha = 1$: Seja λ_k denotando o maior autovalor de A_k , existe uma constante $c_0 \geq 1$ satisfazendo

$$A(v, v) \leq c_0 \left[A(P_0 v, v) + \sum_{k=1}^J \frac{\|A_k P_k\|_2^2}{\lambda_k} \right] \quad \forall v \in \mathcal{S}_J.$$

A segunda hipótese, inspirada nas análises de Zhang [52], engloba o operador $T_k = R_k Q_k A_J = R_k A_k P_k$ associado ao espectro do pré-condicionador $\mathcal{B}$ em (3.12) quando aplicado sobre $A = A_J$, isto é,

$$\mathcal{B} A = \sum_{k=0}^J T_k.$$

Tal hipótese avalia o quanto o operador T_k é “pequeno” quando aplicado sobre funções em $\mathcal{S}_l$, $l \leq k$: existem um número $0 < \epsilon < 1$ e uma constante $c_0 > 0$ satisfazendo

$$A(T_k v, v) \leq (c_1 \epsilon^{k-l})^2 A(v, v) \quad \forall v \in \mathcal{S}_l.$$

Somando tais hipótese à hipótese (3.13), Bramble e Pasciak [11] obtêm a limitação uniforme para a formulação aditiva do método multigrid.

Bornermann e Yserentant [7] apresentam outra prova para (3.26) utilizando a condição

$$A(v, w) \leq C\gamma^{|l-k|} \|v\| \cdot \|w\|, \quad v \in \mathcal{S}_l, \quad w \in \mathcal{S}_k, \quad \gamma \in (0, 1)$$

para demonstrar a limitação superior. Essa condição é semelhante à “strengthened Cauchy-Schwarz inequality” assumida em Xu [48]. Adotam o K -método de Peetre da teoria de interpolação, ao invés do modo de suavização como Dahmen e Kunoht [18], para demonstrar a limitação inferior em (3.26), equivalência já indicada em Johnen e Scherer [29]. Na verdade, provam a limitação para a família de triangularizações $\tilde{T}_0, \tilde{T}_1, \dots, \tilde{T}_J$, que contém a família original $\mathcal{T}_0, \mathcal{T}_1, \dots, \mathcal{T}_J$, mas onde todo triângulo é subdividido em quatro triângulos, isto é, um refinamento uniforme. Tomam as respectivas projeções

$$(\tilde{Q}_i u, \tilde{v}) = (u, \tilde{v}) \quad \forall \tilde{v} \in \tilde{\mathcal{S}}_k$$

e utilizam o K -funcional

$$K(t, u) = \inf_{v \in W^{1,2}(\Omega)} \{ \|u - v\|_{0,2}^2 + t^2 |v|_{2,2}^2 \}^{1/2},$$

onde $|\cdot|_{2,2} = |\cdot|_{2,2;\Omega}$ é a semi-norma $W^{1,2}$ dada por

$$|v|_{2,2}^2 = \sum_{m+n=2} \binom{2}{m} \int_{\Omega} |(D_1^m D_2^n v)(x)|^2 dx,$$

para descrever as propriedades de aproximação das projeções $\tilde{Q}_i: L_2 \rightarrow \tilde{\mathcal{S}}_i$. Que implicam na limitação inferior em (3.26), quando estendidas para as projeções Q_i .

4 Métodos de Subestrutura ou Decomposição de Domínios

O particionamento de problemas elípticos é uma idéia muito natural: torna-se atraente para a obtenção de sub-problemas (muitas vezes semelhantes) com menor número de variáveis; permite a criação de processos iterativos e aumenta a gama dos possíveis métodos de resolução a serem adotados. Bjorstad e Widlund [6], Keyes e Gropp [30] e Dryja e Widlund [23] apresentam um apanhado geral dos conceitos envolvidos em *métodos de subestruturas*, também conhecidos como *métodos de decomposição de domínios*. A cada passo de iteração, o problema elíptico discreto original é resolvido em sub-regiões nas quais a original foi dividida. A interação entre diferentes partes é feita pela transferência de dados apropriados através das interfaces criadas pela subdivisão do domínio. Esses dados são gerados em cada iteração a partir do resíduo da equação original ou de uma equação auxiliar. Um método de gradientes conjugados é freqüentemente usado para acelerar a convergência. Em adição aos problemas locais, um problema global, na malha mais grossa, pode ser incorporado ao pré-condicionador para melhorar a velocidade de convergência no caso de muitas sub-regiões. Observa-se um paralelo com a necessidade de um termo global correspondente à solução na malha mais grossa em métodos multigrid.

As considerações a serem feitas consistem em como equacionar o particionamento do problema original (inclusive domínio original) em sub-problemas (subdomínios não sobrepostos), e a forma de levarmos em conta a interação entre esses sub-problemas. Trabalhos iniciais (vide Bjorstad e Widlund [6] para maiores referências) consideram métodos onde os sub-problemas são resolvidos de forma exata e a interação entre os mesmos, que origina um problema auxiliar ligado as variáveis de interface, é computada por um método de gradientes conjugados, por exemplo. O problema correspondente às variáveis de interface é obtido através de uma eliminação de Gauss, obtendo-se uma matriz do tipo complemento de Schur. É interessante notar que tal matriz corresponde a matriz de capacitância, estudada na mesma época (vide, por exemplo, Dryja [22] e [21]).

O uso de pré-condicionadores, ao invés de resolução exata, para os sub-problemas é considerado por Bramble, Pasciak e Schatz [12] e em trabalhos subseqüentes como Mandel [31], [32], [33] e Bramble, Pasciak e Schatz [13] – [16]. Sendo os sub-problemas independentes, e portanto ligados a uma resolução local, oferece-se naturalmente a oportunidade de paralelização no tratamento simultâneo dos mesmos. Um apanhado geral e comparações de alguns desses trabalhos podem ser encontrados em Keyes e Gropp [30].

4.1 Formulação dos problemas e pré-condicionadores

Inicialmente, em Bramble, Pasciak e Schatz [12], é feita a análise de um problema para um domínio formado por duas regiões como, por exemplo, na figura 4.1. O processo de resolução divide-se em três etapas: resolução da restrição do problema às componentes de elementos interiores a uma das duas subestruturas; em seguida, tal informação é utilizada

para a resolução da restrição do problema à outra subestrutura (inclusive fronteira entre as regiões); e, num terceiro passo, temos a correção das componentes da primeira estrutura.

Figura 4.1: Domínios formados por duas sub-regiões.

Definido o problema de Dirichlet para um operador $A(\cdot, \cdot)$ de segunda ordem, uniformemente elíptico em duas dimensões:

$$\begin{aligned} A(V, \phi) &= (f, \phi), \quad \forall \phi \in W_0^{1,2}(\Omega) \\ \bar{\Omega} &= \bar{\Omega}_1 \cup \bar{\Omega}_2 \end{aligned} \quad (4.1)$$

onde

$$A(v, \phi) = \sum_{i,j=1}^2 \int_{\Omega} a_{ij} \frac{\partial v}{\partial x_i} \frac{\partial \phi}{\partial x_j} dx,$$

Ω_1 e Ω_2 não sobrepostos, isto é, $\overset{\circ}{\Omega}_1 \cap \overset{\circ}{\Omega}_2 = \emptyset$,

$\Gamma = \bar{\Omega}_1 \cap \bar{\Omega}_2$ fronteira comum e

$W_0^1(\Omega)$ = espaço das distribuições com suporte compacto em Ω e de primeira derivada quadrado integrável

Seja a respectiva formulação em elementos finitos, obtida a partir de uma triangularização $\mathcal{T}$ quase-uniforme (vide pag. 6):

$\mathcal{S}_h^0 = \mathcal{S}_h^0(\Omega) \subset W_0^1(\Omega)$ espaço das funções lineares em $\mathcal{T}$, contínuas em Ω e com suporte em $\bar{\Omega}$,

$$A: \mathcal{S}_h^0 \times \mathcal{S}_h^0 \rightarrow \mathbb{R}$$

$$A(V, \phi) = A_1(V, \phi) + A_2(V, \phi) \quad \text{onde} \quad A_k(V, \phi) = \sum_{i,j=1}^2 \int_{\Omega_k} a_{ij} \frac{\partial V}{\partial x_i} \frac{\partial \phi}{\partial x_j} dx, \quad k = 1, 2$$

que pode ser tomada como uma decomposição do operador de forma a “separar” o problema sobre Ω_1 e Ω_2 . Temos a formulação

$$A(V, \phi) = (f, \phi), \quad \forall \phi \in \mathcal{S}_h^0.$$

Dada uma solução $V \in \mathcal{S}_h^0$, tomemos sua decomposição, com respeito a Ω_2 , em duas componentes: as geradoras da solução relacionadas ao operador A_2 observado apenas sobre

os elementos interiores de Ω_2 (denotadas V_P); e o complemento ortogonal em relação a A_2 e Ω_2 (denotadas V_H). Isto é

$$V = V_P + V_H, \quad (4.2)$$

onde $V_P \in \mathcal{S}_h^0(\Omega_2)$ é tal que $A_2(V_P, \phi) = A_2(V, \phi)$, $\forall \phi \in \mathcal{S}_h^0(\Omega_2)$. Dessa forma,

$$A_2(V_H, \phi) = 0, \quad \forall \phi \in \mathcal{S}_h^0(\Omega_2),$$

ou seja, $V_H \in \mathcal{S}_h^0(\Omega_2)^{\perp_{A_2}}$. Como $A_1(V_H, \phi) = 0$, $\forall \phi \in \mathcal{S}_h^0(\Omega_2)$, na verdade temos

$$V_H \in \mathcal{S}_h^0(\Omega_2)^{\perp_A},$$

o que sugere a nomenclatura *A-harmônica discreta* para a componente V_H .

Feita a decomposição, consideremos um novo operador $B: \mathcal{S}_h^0 \times \mathcal{S}_h^0 \rightarrow \mathbb{R}$, definido a partir de A

$$B(V, \phi) = A_1(V, \phi) + A_2(V_P, \phi_P) \quad (4.3)$$

onde $\phi = \phi_P + \phi_H$ como em (4.2), e a respectiva formulação para o problema original (4.1) :

$$B(V, \phi) = (f, \phi), \quad \forall \phi \in \mathcal{S}_h^0. \quad (4.4)$$

A escolha do operador B na forma (4.3) pode ser vista como um bom ponto de partida para a criação de um pré-condicionador ao enunciarmos o seguinte teorema, demonstrado em Bramble, Pasciak e Schatz [12], que garante a equivalência espectral entre A e B (observe que $B(V, \phi) = A(V, \phi) - A_2(V_H, \phi_H)$).

Teorema 7 *Existem constantes positivas β_0 e β_1 independentes de h , tais que*

$$\beta_0 B(v, v) \leq A(v, v) \leq \beta_1 B(v, v), \quad \forall v \in \mathcal{S}_h^0.$$

O operador B nos permite a resolução de (4.4) em três etapas, como veremos a seguir. Considerando apenas $\phi \in \mathcal{S}_h^0(\Omega_2)$, o problema (4.4) se reduz a determinar $V_P \in \mathcal{S}_h^0$ tal que

$$A_2(V_P, \phi) = (f, \phi), \quad \forall \phi \in \mathcal{S}_h^0(\Omega_2) \quad (4.5)$$

o que equivale a um problema de Dirichlet em Ω_2 . Obtido V_P em (4.5), passamos a segunda etapa: como $A_2(V_P, \phi_H) = 0$, $\forall \phi_H \in \mathcal{S}_h^0(\Omega_2)^{\perp_A}$, temos

$$B(V, \phi) = A_1(V, \phi) + A_2(V_P, \phi_P) = A_1(V, \phi) + A_2(V_P, \phi).$$

O que nos leva a determinar $V \in \mathcal{S}_h(\Omega_1)$ por

$$A_1(V, \phi) = (f, \phi) - A_2(V_P, \phi), \quad \forall \phi \in \mathcal{S}_h^0 \quad (4.6)$$

o que equivale a um problema de Neumann-Dirichlet em Ω_1 (em particular em Γ). Obtido V , resta na terceira etapa completarmos V_H transportando as informações de V sobre A_2 e Γ .

Matricialmente podemos representar o operador B como num processo de eliminação de Gauss, obtendo-se um complemento de Schur. Tomemos $\mathcal{S}_h^0$ representado por bases nodais particionadas em três subconjuntos $(v_0, v_1, v_2)^T$ correspondentes a Γ , Ω_1 e Ω_2 , respectivamente. Temos a seguinte representação para (4.4)

$$\begin{pmatrix} B_{00} & B_{01} & B_{02} \\ B_{01}^T & B_{11} & 0 \\ B_{02}^T & 0 & B_{22} \end{pmatrix} \begin{pmatrix} v_0 \\ v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \end{pmatrix}. \quad (4.7)$$

O primeiro passo, correspondente à (4.5), é a resolução de

$$B_{22}\bar{v}_2 = b_2,$$

isto é, $\bar{v}_2 = V_P$. Logo, pela terceira equação em (4.7), temos

$$B_{22}v_2 + B_{02}^T v_0 = b_2 \Rightarrow v_2 = \bar{v}_2 - B_{22}^{-1} B_{02}^T v_0 \Rightarrow B_{02}v_2 = B_{02}\bar{v}_2 - B_{02}B_{22}^{-1} B_{02}^T v_0$$

O que corresponde a

$$\begin{pmatrix} B_{00} - B_{02}B_{22}^{-1}B_{02}^T & B_{01} & 0 \\ B_{01}^T & B_{11} & 0 \\ B_{02}^T & 0 & B_{22} \end{pmatrix} \begin{pmatrix} v_0 \\ v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} b_0 - B_{02}B_{22}^{-1}b_2 \\ b_1 \\ b_2 \end{pmatrix}.$$

Dessa forma, o segundo passo, correspondente à (4.6), consiste na resolução do problema de Neumann-Dirichlet

$$\begin{pmatrix} B_{00} - B_{02}B_{22}^{-1}B_{02}^T & B_{01} \\ B_{01}^T & B_{11} \end{pmatrix} \begin{pmatrix} v_0 \\ v_1 \end{pmatrix} = \begin{pmatrix} b_0 - B_{02}B_{22}^{-1}b_2 \\ b_1 \end{pmatrix}.$$

E o terceiro passo, encontrado v_0 , é o cálculo de v_2 (transporte de informações de V sobre V_H em Γ) através de

$$v_2 = \bar{v}_2 - B_{22}^{-1} B_{02}^T v_0.$$

Assim, observamos que o processo de eliminação de Gauss aplicado produz um sistema linear envolvendo um complemento de Schur.

Na série Bramble, Pasciak e Schatz [13] - [16] o mesmo princípio aplica-se a um domínio formado por múltiplas sub-regiões. Aqui o processo de resolução local é aplicado sobre todos os subdomínios e o conceito de complemento ortogonal é tomado somente sobre a fronteira de comunicação Γ , variando-se o método de resolução aproximada para tais componentes (polinômios de Chebyshev, por exemplo, com em Bramble, Pasciak e Schatz [15]). Redefinindo (4.1) para múltiplas sub-regiões:

$$\bar{\Omega} = \cup_{i=1}^{n_r} \bar{\Omega}_i \text{ não sobrepostos, e } \Gamma = \text{fronteira comum} \\ A(V, \phi) = \sum_{i=1}^{n_r} A_i(V, \phi) \text{ onde}$$

$$A_k(V, \phi) = \sum_{i,j=1}^2 \int_{\Omega_k} a_{ij}^k \frac{\partial V}{\partial x_i} \frac{\partial \phi}{\partial x_j} dx, \quad k = 1, \dots, n_r.$$

e tomada uma solução $V \in \mathcal{S}_h^0(\Omega)$, consideramos sua decomposição, em duas componentes: as geradoras da solução relacionadas ao operador A observado apenas sobre os elementos interiores aos subdomínios Ω_i (denotadas V_P); e o complemento ortogonal em relação a A e Γ (denotadas V_H):

$$V = V_P + V_H, \quad (4.8)$$

onde $V_P \in \mathcal{S}_h^0(\Omega_1) \oplus \cdots \oplus \mathcal{S}_h^0(\Omega_{n_r})$ e satisfaz

$$A_i(V_P, \phi) = A_i(V, \phi), \quad \forall \phi \in \mathcal{S}_h^0(\Omega_i), \quad \text{com } i = 1, \dots, n_r. \quad (4.9)$$

Dessa forma,

$$A_i(V_H, \phi) = 0, \quad \forall \phi \in \mathcal{S}_h^0(\Omega_i), \quad i = 1, \dots, n_r, \quad (4.10)$$

isto é,

$$V_H \in [\mathcal{S}_h^0(\Omega_1) \oplus \cdots \oplus \mathcal{S}_h^0(\Omega_{n_r})]^\perp_A \quad (4.11)$$

o que sugere novamente a nomenclatura *A-harmônica discreta* para V_H . Em suma, para cada Ω_i , temos a decomposição (4.8) de V formada por $V_H \in \mathcal{S}_h(\Omega_i)$ satisfazendo (4.10) e assumindo os mesmos valores de V em Ω_i , e por V_P que fica determinado (por (4.9)) em $\partial\Omega_i$ através dos valores que V assume em Ω_i . Feita a decomposição ortogonal, temos por (4.11)

$$A(V, V) = A(V_P, V_P) + A(V_H, V_H). \quad (4.12)$$

O termo $A(V_P, V_P)$ acima refere-se a n_r problemas disjuntos associados ao interior de cada Ω_i , diminuindo a dimensão do problema original e apresentando uma oportunidade de paralelização. A tarefa seguinte consiste em substituímos o termo $A(V_H, V_H)$ em (4.12) de forma conveniente e, em seguida, definirmos um novo operador B .

Em Bramble, Pasciak e Schatz [13] e [16] a escolha de B resulta em sistemas pré-condicionados, de duas e três dimensões, com número de condição da ordem $\mathcal{O}(1 + \ln^2(\frac{d}{h}))$. Onde d é o tamanho dos subdomínios e h o passo na malha mais fina. Em Bramble, Pasciak e Schatz [15] o número de condição obtido, em problemas dois dimensionais, independe de d e h , embora seja necessário um maior trabalho computacional.

Uma primeira aproximação é reduzir o cálculo de A sobre os Ω_i para $\partial\Omega_i$. Isto é sugerido em Bramble, Pasciak e Schatz [13] pela estimativa

$$c|V_H|_{1/2, \partial\Omega_i}^2 \leq A_i(V_H, V_H) \leq C|V_H|_{1/2, \partial\Omega_i}^2, \quad (4.13)$$

para as V_H A_i -harmônicas com valor médio zero em $\partial\Omega_i$, onde

$$|v|_{1/2, \partial\Omega_i}^2 = \int_{\partial\Omega_i} \int_{\partial\Omega_i} \frac{(v(x) - v(y))^2}{|x - y|} ds(x) ds(y) + d^{-1} \langle v, v \rangle_{L^2, \partial\Omega_i},$$

que computacionalmente ainda é indesejável. Tomando-se o operador l linear em $\mathcal{S}_h(\partial\Omega_i)$ que aproxima o operador $-\Delta$ (isto é, $-\frac{\partial^2}{\partial s^2}$ onde s corresponde ao comprimento de arco em $\partial\Omega_i$), da seguinte forma

$$\langle lV, \phi \rangle_{L^2, \partial\Omega_i} = \left\langle \frac{\partial V}{\partial s}, \frac{\partial \phi}{\partial s} \right\rangle_{L^2, \partial\Omega_i}, \quad \forall \phi \in \mathcal{S}_h(\partial\Omega_i)$$

podemos reduzir o resultado (4.13) à estimativa

$$c_1 \langle QV_H, V_H \rangle_\Gamma \leq A(V_H, V_H) \leq C_1 \langle QV_H, V_H \rangle_\Gamma$$

onde

$$\langle QV_H, V_H \rangle_\Gamma = \sum_{i=1}^{n_r} \alpha_i \langle l^{1/2} V_H, V_H \rangle_{L^2, \partial\Omega_i},$$

com α_i fatores de escala. Logo, podemos definir

$$B(V, V) = A(V_P, V_P) + \langle QV_H, V_H \rangle_\Gamma. \quad (4.14)$$

A maior dificuldade para resolução de (4.4) com B definido em (4.14) corresponde ao problema de encontrar $V \in \mathcal{S}_h(\Gamma)$, tal que,

$$\langle QV_H, \phi \rangle_\Gamma = (f, \phi), \quad \forall \phi \in \mathcal{S}_h(\Gamma). \quad (4.15)$$

Novamente podemos tomar uma representação matricial para o operador B , onde temos $(v_B, v_I)^T$ a decomposição correspondente aos elementos em Γ , e aos elementos interiores aos subdomínios Ω_i , respectivamente.

$$\begin{pmatrix} B_{BB} & B_{BI} \\ B_{BI}^T & B_{II} \end{pmatrix} \begin{pmatrix} v_B \\ v_I \end{pmatrix} = \begin{pmatrix} b_B \\ b_I \end{pmatrix}$$

Como primeiro passo, temos a resolução de

$$B_{II} \tilde{v}_I = b_I,$$

o que corresponde à n_r problemas independentes sobre o respectivo Ω_i , o que demonstra uma boa oportunidade de paralelização.

O segundo passo corresponde à resolução de

$$Schur(B)v_B = \tilde{b}_B$$

sobre Γ , onde $Schur(B) = B_{BB} - B_{BI}B_{II}^{-1}B_{BI}^T$ e $\tilde{b}_B = b_B - B_{BI}B_{II}^{-1}b_I$. E finalmente, num terceiro passo, estendemos v_B como A -harmônica sobre os subdomínios

$$v_I = \tilde{v}_I - B_{II}^{-1}B_{BI}^T v_B.$$

Uma outra construção para o operador B faz a distinção entre vértices e arestas em Γ . Bramble, Pasciak e Schatz [13] decompõem $V_H \in \mathcal{S}_h(\Omega_i)$ em $V_H = V_E + V_V$, onde $V_V \in \mathcal{S}_h(\Omega_i)$ é A_i -harmônica discreta, linear sobre cada aresta Γ_{ij} de $\partial\Omega_i$, e assumindo os mesmos valores de V_H (portanto de V) sobre os vértices de Ω_i . Assim, V_E é A_i -harmônica discreta para cada Ω_i , assumindo valor zero sobre todos os vértices dos subdomínios Ω_i . A distinção entre variáveis de vértices e arestas visa obtermos problemas desacoplados para cada aresta, aumentando o grau de paralelismo que pode ser explorado.

Como $V_E = 0$ nos vértices, novamente é utilizada a estimativa (4.13) com a norma $|V_E|_{1/2, \partial\Omega_i}^2$ substituída por $\sum_{\Gamma_{ij}} \alpha_{ij} \langle a^{-1} \tilde{l}_0^{1/2} V_E, V_E \rangle_{\Gamma_{ij}}$ onde $\tilde{l}_0^{1/2}$ é o operador definido em $\mathcal{S}_h^0(\Gamma_{ij})$ por

$$\langle a^{-1} \tilde{l}_0 V_E, \phi \rangle_{\Gamma_{ij}} = \langle a \frac{\partial V_E}{\partial s}, \frac{\partial \phi}{\partial s} \rangle_{\Gamma_{ij}}, \quad \forall \phi \in \mathcal{S}_h^0(\Gamma_{ij}).$$

Além disso, para constantes $0 < c_2 \leq \alpha_{ij} \leq C_2$ independentes de h e d e dos subdomínios Ω_i , temos a estimativa

$$c_2 A_i(V_V, V_V) \leq \sum_{\Gamma_{ij}} \alpha_{ij} (V_V(v_i) - V_V(v_j))^2 \leq C_2 A_i(V_V, V_V)$$

onde v_i e v_j são os vértices da aresta Γ_{ij} .

Baseados nos resultados acima, temos $B(\cdot, \cdot)$ definido por

$$\begin{aligned} B(V, \phi) &= A(V_P, \phi_P) + \sum_{\Gamma_{ij}} \alpha_{ij} \langle a^{-1} \tilde{l}_0^{1/2} V_E, \phi_E \rangle_{\Gamma_{ij}} + \\ &\quad + \sum_{\Gamma_{ij}} \alpha_{ij} (V_V(v_i) - V_V(v_j)) (\phi_V(v_i) - \phi_V(v_j)) \end{aligned} \quad (4.16)$$

e Bramble, Pasciak e Schatz [13] demonstram que seu número de condição cresce no máximo $\mathcal{O}(1 + \ln^2(\frac{d}{h}))$ quando h tende a zero.

Resta a descrição da resolução de (4.4) utilizando-se a nova definição para B . Como veremos, obtêm-se um algoritmo dividido em quatro etapas. A primeira etapa permanece: determinar V_P tal que, restrito a Ω_i , satisfaz

$$A_i(V_P, \phi) = (f, \phi) \quad \forall \phi \in \mathcal{S}_h^0. \quad (4.17)$$

Conhecido V_P , podemos substituí-lo em (4.16)

$$\begin{aligned} \sum_{\Gamma_{ij}} \alpha_{ij} \langle a^{-1} \tilde{l}_0^{1/2} V_E, \phi_E \rangle_{\Gamma_{ij}} + \sum_{\Gamma_{ij}} \alpha_{ij} (V_V(v_i) - V_V(v_j)) (\phi_V(v_i) - \phi_V(v_j)) &= \\ = (f, \phi) - A(V_P, \phi_P) = (f, \phi) - A(V_P, \phi) \end{aligned} \quad (4.18)$$

Lembrando que V_P está definido pelos valores que a solução assume nos pontos interiores aos subdomínios observamos, para cada ϕ , que $(f, \phi) - A_i(V_P, \phi)$ depende apenas dos valores de ϕ sobre as arestas Γ_{ij} . Logo (4.18) representa um conjunto de equações na restrição de $\mathcal{S}_h^0(\Omega)$ em $\cup \Gamma_{ij}$. O primeiro membro em (4.18) possui um termo V_E associado às arestas Γ_{ij} sem os seus vértices v_i e v_j , e um segundo termo associado a esses vértices. Isso sugere, para cada Γ_{ij} , tomarmos ϕ no subespaço de $\mathcal{S}_h^0(\Omega)$ tal que seus elementos anulam-se nos pontos interiores de cada Ω_i e em todas as outras arestas (portanto, nos vértices de Γ_{ij} também). Nesses subespaços temos (4.18) desacoplado nos problemas uni-dimensionais: determinar $V_E \in \mathcal{S}_h^0(\Gamma_{ij})$ tal que

$$\alpha_{ij} \langle a^{-1} \tilde{l}_0^{1/2} V_E, \phi \rangle_{\Gamma_{ij}} = (f, \phi) - A(V_P, \phi) \quad \forall \phi \in \mathcal{S}_h^0(\Gamma_{ij}). \quad (4.19)$$

Obtendo-se a segunda etapa: problemas locais com incógnitas associadas a cada aresta Γ_{ij} e que podem ser tratados em paralelo.

Lembrando que V_V foi tomada linear sobre as arestas Γ_{ij} , consideremos o subespaço de $\mathcal{S}_h^0(\Omega)$ formado pelas funções lineares entre cada vértice das Γ_{ij} e que se anulam nos pontos

interiores aos subdomínios Ω_i . Para cada ϕ pertencente a esse subespaço temos $\phi_E = 0$, pois ϕ_E anula-se nos vértices, é linear sobre as arestas e também A_i -harmônica discreta para todos os Ω_i . Aplicando tal fato à (4.18), obtemos

$$\sum_{\Gamma_{ij}} \alpha_{ij} (V_V(v_i) - V_V(v_j)) (\phi_V(v_i) - \phi_V(v_j)) = (f, \phi) - A(V_P, \phi). \quad (4.20)$$

equação a ser resolvida na terceira etapa do processo. Uma base para tal subespaço é formada tomando-se $\phi^1, \dots, \phi^M$, onde M é o número de vértices interiores à Ω e $\phi^i(v_j) = \delta_{ij}$. Tal escolha produz uma equação (4.20) de diferenças na malha mais grossa, associada a V_V e independente de (4.19). Logo, podemos resolver (4.20) simultaneamente à resolução de (4.19). Como os valores assumidos por V_V nos vértices determinam seus valores assumidos nas arestas, temos $V_H = V_E + V_V$ conhecido sobre todas as arestas Γ_{ij} .

A quarta etapa consiste em determinarmos V_H no interior de cada Ω_i , bastando utilizar a propriedade de que V_H é A_i -harmônica discreta

$$A_i(V_H, \phi) = 0 \quad \forall \phi \in \mathcal{S}_h^0(\Omega). \quad (4.21)$$

Como os valores de V_H já são conhecidos para $\partial\Omega_i$, a equação (4.21) corresponde a problemas de Dirichlet independentes associados à cada subdomínio. Feito isso, basta tomarmos $V = V_P + V_H$ como solução de (4.4).

Podemos resumir o método de decomposição de domínios apresentado, da seguinte forma:

Algoritmo DD1

1. Determine V_P resolvendo os problemas de Dirichlet associados a cada subdomínio. As resoluções pode ser feitas em paralelo.
2. Determine V_E em cada Γ_{ij} resolvendo equações uni-dimensionais em cada Γ_{ij} .
3. Determine V_V em $\cup \Gamma_{ij}$, primeiramente determinando V_V sobre os vértices interiores de Ω através de uma equação de diferenças finitas na malha mais grossa e, em seguida, estendendo V_V como uma função linear por partes sobre as arestas Γ_{ij} . As resoluções nas diferentes arestas Γ_{ij} dos passos 2 e 3 podem ser feitas em paralelo.
4. Determine V_H estendendo os valores de $V_E + V_V$, restritos às arestas Γ_{ij} , como A_i -harmônica discreta sobre os subdomínios. O que equivale à resolução de problemas de Dirichlet associados a cada subdomínio. As resoluções podem ser feitas em paralelo.
5. Tome $V = V_P + V_H$

Novamente a ação do pré-condicionador B pode ser vista como um processo de eliminação de Gauss ao tomarmos sua representação matricial sobre uma base apropriada. Uma formulação para tal base, seguindo as especificações descritas nas etapas do algoritmo, é a seguinte:

- $\{\phi_P^i\}$ conjunto das bases nodais definidas pelos pontos interiores aos subdomínios Ω_i . Esse conjunto forma uma base para $\cup S_h^0(\Omega_i)$.
- $\{\phi_E^i\}$ conjunto das bases nodais definidas pelos pontos pertencentes às arestas Γ_{ij} mas que não são vértices. Esse conjunto forma uma base para $\cup S_h^0(\Gamma_{ij})$.
- $\{\phi_V^i\}$ conjunto das funções associadas aos vértices v_i interiores a Ω , onde ϕ_V^i assume o valor um em v_i e zero sobre os outros vértices v_j , $j \neq i$, e é estendida linearmente sobre as arestas dos subdomínios. Esse conjunto forma uma base para o problema associado à malha mais grossa.

Temos $\{\phi_V^i\} \cup \{\phi_E^i\} \cup \{\phi_P^i\}$ uma base para $S_h^0(\Omega)$.

Resumindo, apresentamos o método de decomposição de domínios em duas formulações: a primeira (4.14) é mais genérica e deixa a cargo do leitor a escolha do “solver” em (4.15); a segunda formulação exemplifica uma escolha. Uma inspiração para o pré-condicionamento (e portanto escolha do “solver” para (4.15)) de tais métodos pode ser encontrada em Mandel [31] (além de [32]) onde temos um princípio correlato com multigrid: um problema global (fronteiras dos subdomínios), mas de dimensão menor que o problema original, é construído para prover transporte de informações a problemas locais (subdomínios). Tal problema global, obtido a partir de um complemento de Schur (feito para separar componentes de fronteira e interiores) é analisado em Mandel [33] a fim de obter-se um pré-condicionamento para o método de gradientes conjugados. Uma possível estimativa pode ser atingida via “strengthened Cauchy inequality” onde consideramos a decomposição do espaço em dois, como acima, e obtemos majoração para o número de condição do pré-condicionador. Em Braes [8], na análise de um método dois-grid, a avaliação dessa desigualdade é obtida adotando-se bases hierárquicas para a malha mais grossa, isto é, sobre as variáveis de fronteira. Tais idéias serão aplicadas no “algoritmo híbrido” proposto por Smith [41] e Smith e Widlund [43] a serem vistos adiante.

5 O Algoritmo Híbrido

O algoritmo híbrido desenvolvido por Smith [41] e Smith e Widlund [43] visa reunir os métodos de decomposição de domínios e das bases hierárquicas apresentados em capítulos anteriores. Nos métodos de subestrutura apresentados em Bjorstad e Widlund [6], Bramble, Pasciak e Schatz [12], [13] - [16] e Dryja [22], [21] temos as variáveis interiores a subdomínios eliminadas inicialmente. Obtém-se um sistema reduzido, correspondente ao complemento de Schur associado as variáveis localizadas nas fronteiras de comunicação entre subdomínios. Tal sistema é resolvido utilizando-se o método dos gradientes conjugados pré-condicionado. A construção do pré-condicionador é um fator importante para a qualidade do método. Uma solução, como foi indicado no final do capítulo anterior, é definirmos, a partir de um problema global associado à malha mais grossa do nosso domínio (vértices das subestruturas), uma formulação de bases hierárquicas como pré-condicionamento para a resolução do sistema. Uma primeira tentativa seria reescrevermos o método de subestruturas em termos das bases hierárquicas. Mas pode-se mostrar (teorema 8, pag 65) que, usando-se tais bases apenas sobre a fronteira entre os subdomínios, obtemos uma estimativa majorada pelo número de condição obtido por Yserentant [49] para o caso de adotá-las sobre todo o domínio. No novo algoritmo, o qual denominaremos *bases hierárquicas em decomposição de domínios* (HBDD), trabalhamos com bases nodais ao eliminarmos variáveis interiores, apenas efetuando mudança para bases hierárquicas nas variáveis associadas às interfaces e vértices. Isso torna o processo mais interessante pois garante a independência entre os subproblemas locais e uma maior exparsidade em cada um desses. O problema restrito à malha mais grossa (vértices dos subdomínios) é resolvido em separado, permitindo a aplicação independente das mudanças de bases (agora uni-dimensionais) associadas a cada aresta da fronteira de comunicação (interface entre dois subdomínios). O algoritmo HBDD é similar ao algoritmo BPS apresentado em Bramble, Pasciak e Schatz [13] (o mesmo algoritmo DD1, apresentado na pag 58). A grande diferença é que HBDD utiliza o pré-condicionamento das bases hierárquicas para cada aresta, enquanto BPS utiliza variantes do pré-condicionamento apresentado por Dryja [21].

Analogamente ao algoritmo híbrido HBDD, Tong, Chan e Kuo [45] apresentam o algoritmo *bases multinível nodais em decomposição de domínios* (MNBDD). Utilizam os mesmos princípios apresentados por Smith e Widlund [43], mas tomando a direção de uma observação feita em Smith [41], sec. 3.1: utilizar o pré-condicionador BPX em lugar das bases hierárquicas (HB). Mais ainda, desenvolvem um algoritmo com uma diferença básica em relação a HBDD: o problema associado às interfaces é tratado como um todo, isto é, a mudança para bases multinível nodais é aplicada para variáveis associadas às arestas e também a vértices. Isso introduz alguma redundância nos vértices. Os autores sugerem tal fato como uma possibilidade para melhor performance.

5.1 O algoritmo híbrido HBDD

Seja a formulação variacional para um problema em duas dimensões, com seu domínio particionado em subestruturas

$$a(V, \phi) = (f, \phi), \quad \forall \phi \in W_0^{1,2}(\Omega) \quad \bar{\Omega} = \cup_{i=1}^{n_r} \bar{\Omega}_i \quad (5.1)$$

onde

$$a(V, \phi) = \sum_{i,j=1}^2 \int_{\Omega} a_{ij} \frac{\partial V}{\partial x_i} \frac{\partial \phi}{\partial x_j} dx,$$

Ω_i , $i = 1, \dots, n_r$ não sobrepostos, isto é, $\cap_{i=1}^{n_r} \bar{\Omega}_i = \emptyset$ e subestruturas triangulares

$W_0^{1,2}(\Omega)$ = espaço das distribuições com suporte compacto em Ω e de primeira derivada quadrado integrável.

No que segue, como em Smith e Widlund [43], consideramos um pré-condicionamento para os problemas uni-dimensionais (associados às variáveis de interface) gerados a partir de um problema em duas dimensões.

Aplicado um processo de refinamento, obtido a partir de uma triangularização quase-uniforme (vide pag. 6)

$$\mathcal{S}_H^0 = \mathcal{S}_H^0(\Omega) \longrightarrow \mathcal{S}_h^0 = \mathcal{S}_h^0(\Omega) \quad (5.2)$$

onde $\mathcal{S}_H^0(\Omega)$ representa o espaço das funções contínuas e lineares sobre a triangularização inicial (malha mais grossa) e com suporte em Ω , e $\mathcal{S}_h^0(\Omega)$ representa o espaço sobre a triangularização final (malha mais fina) do processo. Tomemos $H: \mathcal{S}_h^0(\Omega) \rightarrow \mathcal{S}_H^0(\Omega)$ como a matriz de mudança da base hierárquica para a base nodal relativa ao problema global (como em 2.13, pag 22) e, sobre o espaço de soluções, a decomposição relativa ao método de subestruturas

$$\mathcal{S}_h^0 = \mathcal{S}_{harm} \oplus \mathcal{S}_h^0(\Omega_1) \oplus \dots \oplus \mathcal{S}_h^0(\Omega_{n_r}) \quad (5.3)$$

onde

$$\begin{aligned} \mathcal{S}_h^0(\Omega_i) &= \mathcal{S}_h(\Omega) \cap W_0^1(\Omega_i) \text{ e} \\ \mathcal{S}_{harm} &= \left(\sum_{i=1}^{n_r} \mathcal{S}_h^0(\Omega_i) \right)^{\perp_a}. \end{aligned}$$

Seja A a matriz resultante da formulação nodal do problema (5.1) em $\mathcal{S}_h^0$

$$Ax = g. \quad (5.4)$$

Inicialmente, particionemos $x^T = (x_I, x_B)^T$ e

$$A = \begin{pmatrix} A_{II} & A_{IB} \\ A_{IB}^T & A_{BB} \end{pmatrix} \quad (5.5)$$

nos conjuntos dos pontos interiores e de fronteira, respectivamente. Definidos pontos interiores como os nós pertencentes ao interior de algum subdomínio Ω_i e pontos de fronteira como os nós pertencentes a alguma interface $\Gamma_{ij} = \partial\Omega_i \cup \partial\Omega_j$.

Observemos que, nessa notação, $\mathcal{S}_{harm}$ é caracterizado como o espaço dos elementos $x \in \mathcal{S}_h^0$ tais que

$$A_{II}x_I + A_{IB}x_B = 0. \quad (5.6)$$

Dessa forma, notamos que os elementos de $\mathcal{S}_{harm}$ estão completamente definidos pelos valores que assumem em $\Gamma = \cup \Gamma_{ij}$, isto é, dado x_B obtemos x_I de forma única através da equação

$$x_I = -A_{II}^{-1}A_{IB}x_B.$$

Eliminando-se os pontos interiores, via complemento de Schur, obtemos

$$Schur(A)x_B = \tilde{g}_B \quad (5.7)$$

onde

$$Schur(A) = A_{BB} - A_{IB}^T A_{II}^{-1} A_{IB} \quad \text{e} \\ \tilde{g}_B = g_B - A_{IB}^T A_{II}^{-1} g_I.$$

Tal eliminação corresponde a primeira etapa nos métodos de decomposição de domínios. Nesse equacionamento pode-se observar a caracterização de $\mathcal{S}_{harm}$: dado x_B em Γ , o complemento de Schur toma o respectivo x_I para que $x = (x_I, x_B)^T \in \mathcal{S}_{harm}$, isto é

$$Schur(A)x_B = A_{BB}x_B + A_{IB}^T(-A_{II}^{-1}A_{IB}x_B) = A_{BB}x_B + A_{IB}^T x_I. \quad (5.8)$$

A segunda etapa nos métodos de decomposição de domínios consiste na escolha do pré-condicionador para a resolução do problema (5.8) associado às variáveis de interface. Tomemos o pré-condicionamento HB (como em 2.26, pag 29) restrito à Γ , obtendo

$$D_{BB}^{-1}H_{BB}^T Schur(A)H_{BB}x_B = \tilde{g}_B. \quad (5.9)$$

Façamos uma estimativa para o número de condição do problema (5.9). Inicialmente observemos que utilizando uma mudança de base H invariante sobre Γ (como é o caso para H_{BB}), o complemento de Schur fica definido a partir da mudança de base sobre Γ .

Lema 2 *Seja G uma mudança de base invariante sobre Γ . Então o complemento de Schur associado às incógnitas sobre Γ independe da escolha das bases para $\mathcal{S}_h^0(\Omega_i)$. Mais precisamente*

$$Schur(G^T A G) = G_{BB}^T Schur(A) G_{BB}.$$

Demonstração. Como G é invariante sobre Γ temos

$$\begin{pmatrix} x_I \\ x_B \end{pmatrix} = \begin{pmatrix} G_{II} & G_{IB} \\ 0 & G_{BB} \end{pmatrix} \begin{pmatrix} \hat{x}_I \\ \hat{x}_B \end{pmatrix}$$

a matriz $\hat{A}$ na nova base é representada por

$$\hat{A} = G^T A G = \begin{pmatrix} G_{II}^T & 0 \\ G_{IB}^T & G_{BB}^T \end{pmatrix} \begin{pmatrix} A_{II} & A_{IB} \\ A_{IB}^T & A_{BB} \end{pmatrix} \begin{pmatrix} G_{II} & G_{IB} \\ 0 & G_{BB} \end{pmatrix} =$$

$$= \begin{pmatrix} G_{II}^T A_{II} G_{II} & G_{II}^T (A_{II} G_{IB} + A_{IB} G_{BB}) \\ (G_{IB}^T A_{II} + G_{BB}^T A_{IB}^T) G_{II} & (G_{IB}^T A_{II} + G_{BB}^T A_{IB}^T) G_{IB} + (G_{IB}^T A_{IB} + G_{BB}^T A_{BB}) G_{BB} \end{pmatrix}$$

Logo

$$\begin{aligned} \text{Schur}(\tilde{A}) &= (G_{IB}^T A_{II} + G_{BB}^T A_{IB}^T) G_{IB} + (G_{IB}^T A_{IB} + G_{BB}^T A_{BB}) G_{BB} - \\ &\quad - (G_{IB}^T A_{II} + G_{BB}^T A_{IB}^T) G_{II} G_{II}^{-1} A_{II}^{-1} G_{II}^T G_{II} (A_{II} G_{IB} + A_{IB} G_{BB}) = \\ &= (G_{IB}^T A_{II} + G_{BB}^T A_{IB}^T) G_{IB} + (G_{IB}^T A_{IB} + G_{BB}^T A_{BB}) G_{BB} - \\ &\quad - G_{IB}^T A_{II} G_{IB} - G_{IB}^T A_{IB} G_{BB} - G_{BB}^T A_{IB}^T G_{IB} - G_{BB}^T A_{IB}^T A_{II}^{-1} A_{IB} G_{BB} = \\ &= G_{BB}^T A_{BB} G_{BB} - G_{BB}^T A_{IB}^T A_{II}^{-1} A_{IB} G_{BB} = \\ &= G_{BB}^T (A_{BB} - A_{IB}^T A_{II}^{-1} A_{IB}) G_{BB} = \\ &= G_{BB}^T \text{Schur}(A) G_{BB} \end{aligned}$$

□

Obtemos a seguinte estimativa

Lema 3 Se A é simétrica, positiva definida. Então

$$\kappa(\text{Schur}(A)) \leq \kappa(A).$$

Demonstração. Como A é simétrica, positiva definida, apliquemos o coeficiente de Rayleigh para obtermos λ_M e λ_m seu maior e menor autovalor, respectivamente,

$$\lambda_M = \max_{x \neq 0} \frac{\langle Ax, x \rangle}{\langle x, x \rangle},$$

$$\lambda_m = \min_{x \neq 0} \frac{\langle Ax, x \rangle}{\langle x, x \rangle}.$$

Temos

$$\begin{aligned} \langle Ax, x \rangle &= x^T Ax = (x_I^T \ x_B^T) \begin{pmatrix} A_{II} x_I + A_{IB} x_B \\ A_{IB}^T x_I + A_{BB} x_B \end{pmatrix} = \\ &= x_I^T A_{II} x_I + x_I^T A_{IB} x_B + x_B^T A_{IB}^T x_I + x_B^T A_{BB} x_B \end{aligned}$$

e

$$\langle \text{Schur}(A) x_B, x_B \rangle = x_B^T \text{Schur}(A) x_B = x_B^T A_{BB} x_B - x_B^T A_{IB}^T A_{II}^{-1} A_{IB} x_B.$$

Em particular, para $x_I = 0$, temos

$$\langle Ax, x \rangle = x_B^T A_{BB} x_B \geq \langle \text{Schur}(A) x_B, x_B \rangle \quad \forall x_B.$$

Logo

$$\lambda_M \geq \max_{x_B \neq 0} \frac{\langle \text{Schur}(A) x_B, x_B \rangle}{\langle x_B, x_B \rangle} = \mu_M,$$

onde μ_M é o maior autovalor de $\text{Schur}(A)$. Analogamente, dado $x_B \neq 0$, em particular para $x_I = -A_{II}^{-1} A_{IB} x_B$, temos

$$\begin{aligned}\langle Ax, x \rangle &= x_B^T A_{IB}^T A_{II}^{-T} A_{IB} x_B - x_B^T A_{IB}^T A_{II}^{-T} A_{IB} x_B - x_B^T A_{IB}^T A_{II}^{-1} A_{IB} x_B + x_B^T A_{BB} x_B = \\ &= \langle Schur(A) x_B, x_B \rangle\end{aligned}$$

Logo

$$\lambda_m \leq \min_{x_B \neq 0} \frac{\langle Schur(A) x_B, x_B \rangle}{\langle x_B, x_B \rangle} = \mu_m,$$

onde μ_m é o menor autovalor de $Schur(A)$. Portanto

$$\kappa(Schur(A)) = \frac{\mu_M}{\mu_m} \leq \frac{\lambda_M}{\lambda_m} = \kappa(A).$$

□

Duas observações resultantes da demonstração acima são que $Schur(A)$ é positiva definida, pois A é positiva definida. E, para uma função dada sobre Γ , a sua extensão como harmônica discreta é a extensão de menor energia $a(\cdot, \cdot)$. Pois, dada $\tilde{u}^h \in \mathcal{S}_{harm}$, temos

$$a(\tilde{u}^h, \tilde{u}^h) = \langle A \tilde{u}^h, \tilde{u}^h \rangle = \langle Schur(A) \tilde{u}_B^h, \tilde{u}_B^h \rangle = \min_{v^h|_{\Gamma} = \tilde{u}_B^h} \langle A v^h, v^h \rangle = \min_{v^h|_{\Gamma} = \tilde{u}_B^h} a(v^h, v^h).$$

O que pode ser visto como uma justificativa para a escolha de $\mathcal{S}_{harm}$ na decomposição 5.3: interferir o menos possível sobre os problemas locais nos espaços $\mathcal{S}_h^0(\Omega_i)$. A estimativa final é dada pelo teorema que segue.

Teorema 8 *O número de condição do algoritmo híbrido (5.9) é majorado pelo número de condição do problema original resolvido por bases hierárquicas, isto é*

$$\kappa(D_{BB}^{-1} H_{BB}^T Schur(A) H_{BB}) \leq \kappa(D^{-1} H^T A H). \quad (5.10)$$

Demonstração. D representa a relaxação de Jacobi por blocos usada em conjunto com as bases hierárquicas e é portanto bloco diagonal em relação às variáveis em Γ e diagonal nas variáveis restantes. Assim, D representa uma mudança de base invariante em Γ . Pelo lema 2, temos

$$Schur(D^{-1/2} H^T A H D^{-1/2}) = D_{BB}^{-1/2} Schur(H^T A H) D_{BB}^{-1/2}.$$

Novamente aplicando o lema 2, agora sobre H

$$D_{BB}^{-1/2} Schur(H^T A H) D_{BB}^{-1/2} = D_{BB}^{-1/2} H_{BB}^T Schur(A) H_{BB} D_{BB}^{-1/2}.$$

Logo,

$$\kappa(D_{BB}^{-1/2} H_{BB}^T Schur(A) H_{BB} D_{BB}^{-1/2}) = \kappa(Schur(D^{-1/2} H^T A H D^{-1/2})).$$

Pelo lema 3

$$\kappa(D_{BB}^{-1} H_{BB}^T Schur(A) H_{BB}) \leq \kappa(D^{-1/2} H^T A H D^{-1/2}).$$

□

Dessa forma, ao analisarmos o número de condição do sistema pré-condicionado, notamos que basta tomarmos o pré-condicionamento HB apenas sobre o sistema associado

às variáveis de interface. Isso garante que, mantida a formulação nodal para as variáveis interiores aos subdomínios, temos maior exparsidade nos n_r problemas de Dirichlet a serem resolvidos na primeira etapa do método de decomposição de domínios.

Podemos resumir o algoritmo da seguinte forma:

1. Resolução de n_r problemas de Dirichlet. Podem ser feitas em paralelo.
2. Resolução de $Schur(A)x_B = \hat{g}_B$ por PCG utilizando-se bases hierárquicas restritas às arestas e vértices, isto é,

$$D_{BB}^{-1}H_{BB}Schur(A)H_{BB}^T x_B = \hat{g}_B$$

3. Extensão de x_B como função harmônica sobre os pontos interiores. O que equivale à n_r problemas de Dirichlet que podem ser resolvidos em paralelo.

Explicitando o método PCG (pag 8) e o pré-condicionamento (5.9), temos

Algoritmo HBDD. Retorna $x = (x_I, x_B)^T$

1. $\hat{x}_I = A_{II}^{-1}g_I$
- 2.a. escolha x_B^0
- 2.b. $r^0 = \hat{g}_B - Schur(A)x_B^0$
- 2.c. $\tilde{r}^0 = H_{BB}D_{BB}^{-1}H_{BB}^T r^0$
- 2.d. $p^0 = \tilde{r}^0$
- 2.e. para $k = 0, \dots, k_{max}$
- 2.f. $\alpha = \langle \tilde{r}^k, r^k \rangle / \langle p^k, Schur(A)p^k \rangle$
- 2.g. $x_B^{k+1} = x_B^k + \alpha p^k$
- 2.h. teste de convergência
- 2.i. $r^{k+1} = r^k - \alpha Schur(A)p^k$
- 2.j. $\tilde{r}^{k+1} = H_{BB}D_{BB}^{-1}H_{BB}^T r^{k+1}$
- 2.k. $\beta = \langle \tilde{r}^{k+1}, r^{k+1} \rangle / \langle r^k, r^k \rangle$
- 2.l. $p^{k+1} = \tilde{r}^{k+1} + \beta p^k$
- 2.m. próximo k
- 2.n. $x_B = x_B^{k+1}$
3. $x_I = \hat{x}_I - A_{II}^{-1}A_{IB}x_B$

Uma primeira observação é que embora $Schur(A) = A_{BB} - A_{IB}^T A_{II}^{-1} A_{IB}$ seja geralmente densa, não é necessário armazenar tal matriz ao aplicarmos o método dos gradientes conjugados. Como PCG requer apenas produtos da matriz por vetores, o termo $A_{II}^{-1}v$ é obtido pela resolução dos problemas restritos a cada subdomínio, como no passo 1 do algoritmo. Em segundo lugar, observemos que nos v-ciclos, referentes aos passos 2.c e 2.j, temos os operadores H_{BB} e H_{BB}^T aplicados apenas sobre arestas e vértices. Logo, subida e descida do v-ciclo correspondem a aplicarmos os operadores de restrição H_{BB}^T e interpolação H_{BB} sobre conjuntos uni-dimensionais disjuntos de variáveis (nós interiores a cada aresta), o que permite a paralelização dessa tarefa.

No caso de aplicarmos o pré-condicionador bases hierárquicas com resolução exata na malha mais grossa (CHB), devemos determinar o bloco diagonal em D_{BB} referente à malha mais grossa. O que equivale ao bloco diagonal em $Schur(A)$ restrito aos vértices dos subdomínios.

5.2 O algoritmo MNBDD

O algoritmo *bases multinível nodais em decomposição de domínios* (MNBDD) combina os conceitos de decomposição de domínios e adoção de uma mudança de bases sobre as arestas associadas à decomposição. Tong, Chan e Kuo [45] utilizam basicamente os mesmos resultados de Smith e Widlund [43] descritos na seção anterior.

Tomando-se o problema (5.7) associado às variáveis de interface, apliquemos o pré-condicionamento (3.19), pag. 43, definido no método BPX. Como o problema considerado é de dimensão dois, em particular para (3.20), temos

$$M^{-1} = GD^{-1}G^T$$

onde

$G: \mathbb{R}^{n_J + \dots + n_0} \rightarrow \mathbb{R}^{n_J}$ corresponde à mudança da base multi-nível nodal para a base nodal,
 $G^T: \mathbb{R}^{n_J} \rightarrow \mathbb{R}^{n_J + \dots + n_0}$ corresponde a sua transposta e
 $D^{-1} = \text{bloco diagonal } (h_J^{2-d} I_{n_J}, h_{J-1}^{2-d} I_{n_{J-1}}, \dots, h_1^{2-d} I_{n_1}, h_0^{2-d} I_{n_0} \text{ ou } A_0^{-1})$

O que, em termos práticos, corresponde à aplicarmos a mudança para as bases multi-nível nodais; em seguida, resolvermos o problema associado à malha mais grossa; e finalmente, retornarmos às bases nodais.

Uma primeira observação pode ser feita através do seguinte lema.

Lema 4 *A mudança de bases G mantém o espaço de variáveis sobre Γ invariante, isto é,*

$$G_{BI} = 0 \quad \text{para} \quad G = \begin{pmatrix} G_{II} & G_{IB} \\ G_{BI} & G_{BB} \end{pmatrix}$$

Demonstração. Seja a mudança de bases multinível nodal para a base nodal

$$u \equiv Gv = \sum_{l=0}^J \sum_{i \in \mathcal{I}_l} v_i^l \hat{\psi}_i^l$$

onde $\{\hat{\psi}_i^l : i = 1, \dots, n_l\}$ representa a base nodal do espaço S_l , $l = 0, \dots, J$. Ou seja, $\{\hat{\psi}_i^l\}_{i \in \mathcal{J}_l}^{l=0, \dots, J}$ é a base multinível nodal de V^h , como na seção 3.2.

Consideremos o particionamento $u = (u_I, u_B)^T$ e $v = (v_I, v_B)^T$ para $u \in \mathbb{R}^{n_J}$ e $v \in \mathbb{R}^{n_J + \dots + n_0}$ tomados como vetores de coeficientes sobre suas respectivas bases. Para um ponto $x_j \in \Gamma$, obtemos

$$(u_B)_j = (G_{BB}v_B)_j + (G_{BI}v_I)_j = \sum_{l=0}^J \left\{ \sum_{\substack{i \in \mathcal{J}_l \\ i \in \Gamma}} (v_B)_i^l (\hat{\psi}_i^l)_j + \sum_{\substack{i \in \mathcal{J}_l \\ i \notin \Gamma}} (v_I)_i^l (\hat{\psi}_i^l)_j \right\} \quad (5.11)$$

onde $(\hat{\psi}_i^l)_j$ corresponde ao valor $\hat{\psi}_i^l(x_j)$.

Dado $i \in \mathcal{J}_l$, por (2.3), pag. 16, temos

$$(\hat{\psi}_i^l)_k = \hat{\psi}_i^l(x_k) = \delta_{ik}$$

para x_k representando os vértices da aresta a que x_j pertence. Logo, se $x_i \in \Omega_k$, para algum k (isto é, x_i é ponto interior) temos $\hat{\psi}_i^l$ nula sobre toda a aresta. Assim

$$(\hat{\psi}_i^l)_j = 0, \quad \forall i \in \mathcal{J}_l, \quad i \notin \Gamma,$$

isto é, o segundo termo na somatória (5.11) é nulo. Portanto

$$G_{BI} = 0 \quad \text{e} \quad (u_B)_j = (G_{BB}v_B)_j.$$

□

Dessa forma, o lema 2 da seção anterior nos mostra que

$$\text{Schur}(G^T A G) = G_{BB}^T \text{Schur}(A) G_{BB}.$$

Na verdade, como na demonstração do teorema 8 da seção anterior, sendo D^{-1} bloco diagonal e invariante sobre Γ , podemos também aplicar o lema 2 sobre D^{-1} , obtendo

$$\text{Schur}(D^{-1} G^T A G) = D_{BB}^{-1} G_{BB}^T \text{Schur}(A) G_{BB}.$$

Dessa forma, podemos tomar um pré-condicionador para o problema do complemento de Schur, como

$$M_B^{-1} = G_{BB} D_{BB}^{-1} G_{BB}^T.$$

E definir o novo algoritmo MNBDD como

$$D_{BB}^{-1} G_{BB}^T \text{Schur}(A) G_{BB} x_B = \hat{g}_B. \quad (5.12)$$

Como anteriormente, podemos enunciar o teorema.

Teorema 9 *O número de condição do algoritmo MNBDD (5.12) é majorado pelo número de condição do problema original resolvido pelas bases multinível nodais no método BPX, isto é,*

$$\kappa(D_{BB}^{-1}G_{BB}^T \text{Schur}(A)G_{BB}) \leq \kappa(D^{-1}G^T A G) \quad (5.13)$$

O novo algoritmo MNBDD apresenta vantagens sobre a concepção original do método BPX. A iteração de gradientes conjugados é efetuada sobre um conjunto muito menor de pontos e o número de condição do novo método é majorado pelo número do BPX.

Analogamente ao método HBDD, podemos descrever

Algoritmo MNBDD. Retorna $x = (x_I, x_B)^T$

1. $\hat{x}_I = A_{II}^{-1}g_I$
- 2.a. escolha x_B^0
- 2.b. $r^0 = \hat{g}_B - \text{Schur}(A)x_B^0$
- 2.c. $\tilde{r}^0 = G_{BB}D_{BB}^{-1}G_{BB}^T r^0$
- 2.d. $p^0 = \tilde{r}^0$
- 2.e. para $k = 0, \dots, k_{\max}$
- 2.f. $\alpha = \langle \tilde{r}^k, r^k \rangle / \langle p^k, \text{Schur}(A)p^k \rangle$
- 2.g. $x_B^{k+1} = x_B^k + \alpha p^k$
- 2.h. teste de convergência
- 2.i. $r^{k+1} = r^k - \alpha \text{Schur}(A)p^k$
- 2.j. $\tilde{r}^{k+1} = G_{BB}D_{BB}^{-1}G_{BB}^T r^{k+1}$
- 2.k. $\beta = \langle \tilde{r}^{k+1}, r^{k+1} \rangle / \langle r^k, r^k \rangle$
- 2.l. $p^{k+1} = \tilde{r}^{k+1} + \beta p^k$
- 2.m. proximo k
- 2.n. $x_B = x_B^{k+1}$
3. $x_I = \hat{x}_I - A_{II}^{-1}A_{IB}x_B$

Para cada $i \in \{1, \dots, n\}$, se calcula $\hat{y}_i$ a partir de $\hat{y}_i = \hat{y}_i + \alpha_i$, onde α_i é o passo de atualização. O algoritmo termina quando $\hat{y}_i$ converge para o valor verdadeiro y_i .

Assim, o algoritmo híbrido é capaz de lidar com problemas de otimização não linear, onde a função objetivo é não convexa e o domínio de busca é não convexo.

Para cada $i \in \{1, \dots, n\}$, se calcula $\hat{y}_i$ a partir de $\hat{y}_i = \hat{y}_i + \alpha_i$.

Para cada $i \in \{1, \dots, n\}$, se calcula $\hat{y}_i$ a partir de $\hat{y}_i = \hat{y}_i + \alpha_i$.

Para cada $i \in \{1, \dots, n\}$, se calcula $\hat{y}_i$ a partir de $\hat{y}_i = \hat{y}_i + \alpha_i$.

Para cada $i \in \{1, \dots, n\}$, se calcula $\hat{y}_i$ a partir de $\hat{y}_i = \hat{y}_i + \alpha_i$.

Para cada $i \in \{1, \dots, n\}$, se calcula $\hat{y}_i$ a partir de $\hat{y}_i = \hat{y}_i + \alpha_i$.

Para cada $i \in \{1, \dots, n\}$, se calcula $\hat{y}_i$ a partir de $\hat{y}_i = \hat{y}_i + \alpha_i$.

Para cada $i \in \{1, \dots, n\}$, se calcula $\hat{y}_i$ a partir de $\hat{y}_i = \hat{y}_i + \alpha_i$.

Para cada $i \in \{1, \dots, n\}$, se calcula $\hat{y}_i$ a partir de $\hat{y}_i = \hat{y}_i + \alpha_i$.

Para cada $i \in \{1, \dots, n\}$, se calcula $\hat{y}_i$ a partir de $\hat{y}_i = \hat{y}_i + \alpha_i$.

Para cada $i \in \{1, \dots, n\}$, se calcula $\hat{y}_i$ a partir de $\hat{y}_i = \hat{y}_i + \alpha_i$.

Para cada $i \in \{1, \dots, n\}$, se calcula $\hat{y}_i$ a partir de $\hat{y}_i = \hat{y}_i + \alpha_i$.

6 Experimentos numéricos

Neste capítulo, apresentamos a descrição dos programas utilizados para a implementação dos métodos HB e BPX. Em seguida, temos a análise de resultados numéricos obtidos na solução de uma equação de Poisson.

6.1 Descrição do programa

Para a resolução da equação

$$\begin{aligned} -\Delta u &= f, & \text{em } \Omega = [0, 1] \times [0, 1] \\ u &= g, & \text{em } \partial\Omega \end{aligned} \quad (6.1)$$

tomamos um refinamento regular sobre Ω obtido a partir de uma malha mais grossa com 9 pontos interiores, gerando novos níveis de refinamento com 49, 225, 961, 3969, 16126 e 65025 pontos interiores. Para isso, utilizamos o programa apresentado no apêndice A. A partir de uma malha regular inicial, criada pela rotina **CARREGA_MALHA_GROSSA**, aplicamos uma recursão em **REFINA_UNIFORME** definida pela regra: extraído um nó da pilha de recursão **FILA_IJ**, são criados os nós vizinhos a este e que pertençam ao próximo refinamento (malha seguinte), os quais também são inseridos na pilha para o próximo passo da recursão, isto é, a criação da malha seguinte. A distribuição dos nós na malha é representada na matriz **REFINAMENTO** através do índice associado a cada novo nó criado. A indexação, além de identificar o nó, permite distinguirmos em que nível do refinamento o mesmo foi criado, pois o vetor **DIM_V1_VN** contém na posição **I** o maior índice dos nós criados no nível **I**; também permite a criação dos vetores **I1** e **I2** descritos na figura 2.5, pag. 23. A figura 6.1 exemplifica um passo de refinamento.

Obtida em **REFINAMENTO** a distribuição final dos nós sobre a malha mais fina, a rotina **MONTA_LAPLACIANO** cria o sistema linear associado à formulação de elementos finitos nodais para a equação (6.1). Os dados do problema são fornecidos via rotinas **F** e **G**. Tanto a matriz como os vetores solução e lado direito são armazenados utilizando-se uma estrutura de apontadores sobre o vetor real **VETOR_REAL**.

A etapa seguinte consiste na chamada da rotina **PCG** que aplica o método dos gradientes conjugados pré-condicionado (pag. 8) sobre o sistema linear. O pré-condicionamento do resíduo é aplicado quando chamada a rotina **V_CICLO**. Tal rotina recebe esse nome por ter a estrutura de um V-ciclo (veja os algoritmos das páginas 30 e 47): desce até a malha mais grossa, nível por nível, através de **RESTRICAO**; pode efetuar uma resolução exata, ou não, (CHB ou CBPX) na malha mais grossa pelo método de Choleski; e retorna à malha mais fina, nível por nível, utilizando a rotina **INTERPOLACAO**.

Quando aplicada a resolução exata na malha mais grossa (CHB ou CBPX), a rotina **MONTA_G_LAPLACIANO** carrega na matriz **GROSSA_MATRIZ** a formulação de

Figura 6.1: Exemplificação do processo de refinamento

elementos finitos nodais associada à malha mais grossa. E a rotina **CHOLESK** cria sua decomposição LL^T .

6.2 Resultados numéricos e sua análise

Para a equação (6.1), com

$$\begin{aligned}f &= 10 \sin(3x + y) , \\g &= \sin(3x + y) , \\u &= \sin(3x + y) ,\end{aligned}$$

plotamos, a cada iteração (ciclo) de PCG, a norma discreta da energia do erro entre a aproximação computada na resolução do problema discretizado e os valores assumidos pela solução exata do problema contínuo nos nós da malha mais fina. Foram efetuadas 50 iterações com cada um dos métodos.

para aproximação inicial oscilante entre 10 e -10:

49 pontos interiores

225 pontos interiores

961 pontos interiores

3969 pontos interiores

16129 pontos interiores

65025 pontos interiores

para aproximação inicial constante igual à 1:

49 pontos interiores

225 pontos interiores

961 pontos interiores

3969 pontos interiores

16129 pontos interiores

65025 pontos interiores

Observa-se que o erro na solução iterativa do sistema linear decresce até atingir-se o erro de discretização, correspondente aos patamares constantes nos gráficos. Neste ponto chega-se à convergência do processo iterativo, dentro de uma precisão numérica bem menor que o próprio erro de discretização. Este erro de discretização é reduzido por um fator aproximadamente quatro a cada refinamento da malha (onde h é reduzido à metade), correspondente à ordem dois da discretização utilizada, o que pode ser observado comparando-se os erros finais nos sucessivos gráficos.

É evidente através dos resultados que a velocidade de convergência do método dos gradientes conjugados sem pré-condicionamento deteriora-se rapidamente com o refinamento das malhas. Para as malhas mais finas tal método ainda encontra-se longe da convergência após 50 iterações. O pré-condicionamento através das bases hierárquicas resulta em um ganho significativo em termos de velocidade de convergência, sua deterioração ocorre bem mais lentamente, correspondendo às estimativas da ordem de $\mathcal{O}(\ln^2 h^{-1})$ para seu número de condição. Note-se que o uso da resolução exata na malha mais grossa não traz nenhuma melhora para a performance dos métodos, não se justificando portanto o trabalho computacional extra que acarreta. O método BPX é o mais rápido em todos os casos, sendo trinta iterações suficientes para se atingir convergência em quase todos os experimentos.

Para quantificarmos e acessarmos as diferenças qualitativas entre os métodos, plotamos nos gráficos seguintes o número de iterações necessárias para a redução do erro inicial por um fator 10^{-4} , no caso da aproximação inicial oscilante entre os valores 10 e -10, e uma redução por um fator $5 \cdot 10^{-3}$, no caso da aproximação inicial constante igual a 1. Cada gráfico é acompanhado de uma tabela apresentando o número de iterações e o supremo do resíduo calculado sobre o sistema linear obtido.

para aproximação inicial oscilante entre 10 e -10:

	49		225		961		3969		16129		65025	
CG	16	.10 E-2	29	.82 E-2	50	.45 E-2	87	.44 E-2	149	.30 E-2	242	.22 E-2
IHB	15	.84 E-3	18	.86 E-2	22	.26 E-1	26	.62 E-1	29	.97 E-1	32	.19 E+0
CHB	17	.15 E-2	22	.98 E-2	26	.56 E-1	31	.59 E-1	34	.13 E+0	37	.17 E+0
IBPX	13	.86 E-3	13	.70 E-2	14	.10 E-1	14	.86 E-2	14	.12 E-1	13	.21 E-1
CBPX	13	.17 E-2	13	.89 E-2	14	.75 E-2	14	.14 E-1	14	.13 E-1	14	.10 E-1

para aproximação inicial constante igual à 1:

	49		225		961		3969		16129		65025	
CG	12	.60 E-2	23	.62 E-2	45	.36 E-2	87	.41 E-2	170	.24 E-2	325	.12 E-2
IHB	9	.37 E-2	8	.31 E-1	9	.36 E-1	9	.50 E-1	9	.57 E-1	9	.63 E-1
CHB	10	.65 E-2	10	.23 E-1	12	.22 E-1	13	.21 E-1	13	.65 E-1	13	.11 E+0
IBPX	8	.55 E-2	8	.88 E-2	8	.80 E-2	8	.93 E-2	7	.35 E-1	7	.39 E-1
CBPX	9	.24 E-2	8	.12 E-1	8	.14 E-1	8	.14 E-1	8	.17 E-1	7	.46 E-1

Observemos que o número de iterações do método CG sem pré-condicionamento quase dobra a cada refinamento da malha, de acordo com a ordem $\mathcal{O}(h^{-2})$ do número de condição da matriz em bases nodais (veja expressão (1.4), pag. 7). No caso do método IHB verificamos que o número de iterações cresce ligeiramente com o refinamento das malhas, crescimento este limitado pela ordem $\mathcal{O}(\ln^2 h^{-1})$ do número de condição da matriz em bases hierárquicas. Para comprovar isso, basta lembramos que tal estimativa é equivalente a limitação $\mathcal{O}(J+1)$ associada ao número de refinamentos gerados, e temos a relação $\frac{n^2 \text{ ciclos}}{n^2 \text{ níveis}+1}$ decrescente, o que preserva a limitação.

nível	1	2	3	4	5	6
tabela 1	$\frac{15}{2} = 7.5$	$\frac{18}{3} = 6$	$\frac{22}{4} = 5.5$	$\frac{26}{5} = 5.2$	$\frac{29}{6} \approx 4.8$	$\frac{32}{7} \approx 4.6$
tabela 2	$\frac{9}{2} = 4.5$	$\frac{8}{3} \approx 2.7$	$\frac{9}{4} \approx 2.3$	$\frac{9}{5} = 1.8$	$\frac{6}{6} = 1.5$	$\frac{9}{7} \approx 1.3$

Já os resultados para o método BPX estão de acordo com os recentes resultados obtidos na literatura, mostrando que seu número de condição é de ordem $\mathcal{O}(1)$, o que se reflete no fato de que o número de iterações para a redução do erro por um fator fixo independe do refinamento.

Bibliografia

- [1] O. Axelsson and V. A. Barker. *Finite Element Solution of Boundary Value Problems*. Academic Press, 1984.
- [2] Randolph E. Bank. *PLTMG: A Software Package for Solving Elliptic Partial Differential equations 6.0*. Siam, 1990.
- [3] Randolph E. Bank e Todd F. Dupont. *An Optimal Order Process for Solving Finite Element Equations*. Math. Comp., 36(153), 35–51, 1981.
- [4] Randolph E. Bank, Todd F. Dupont, and Harry Yserentant. *The hierarchical basis multigrid method*. Numer. Math., 52, 427 – 458, 1988.
- [5] Randolph E. Bank e Harry Yserentant. *Some Remarks on the Hierarchical Basis Multigrid Method*. Domain Decomposition Methods (Chan, Glowinski, Periaux e Widlund eds.), Siam, 1989, 140–146.
- [6] Petter E. Bjorstad e Olof B. Widlund. *Iterative Methods for The Solution of Elliptic Problems into Substructures*. SIAM J. Numer. Anal., 26(6), 1097–1120, 1986.
- [7] Folkmar Bornemann e Harry Yserentant. *A Basic Norm Equivalence for The Theory of Multilevel Methods*. Numer. Math. 64, 455 – 476, 1993.
- [8] Dietrich Braess. *The contraction number of a multigrid method for solving the Poisson equation*. Numer. Math., 37, 387 – 404, 1981.
- [9] James H. Bramble, Joseph E. Pasciak, and Jinchao Xu. *Parallel multilevel preconditioners*. Math. Comp., 55, 1 – 22, 1982.
- [10] James H. Bramble, e Joseph E. Pasciak. *New convergence estimatives for multigrid algorithms*. Math. Comp., 49, 311 – 329, 1987.
- [11] James H. Bramble e Joseph E. Pasciak. *New Estimates for Multilevel Algorithms Including The V-cycle*. Math. Comp. 60(202), 447 – 471, 1993.
- [12] James H. Bramble, Joseph E. Pasciak, and Alfred H. Schatz. *An iterative method for elliptic problems on regions partitioned into substructures*. Math. Comp., 46(174), 361–369, 1986.
- [13] James H. Bramble, Joseph E. Pasciak, and Alfred H. Schatz. *The construction of preconditioners for elliptic problems by substructuring, I*. Math. Comp., 47(175), 103–134, 1986.
- [14] James H. Bramble, Joseph E. Pasciak, and Alfred H. Schatz. *The construction of preconditioners for elliptic problems by substructuring, II*. Math. Comp., 49(179), 1–16, 1987.

- [15] James H. Bramble, Joseph E. Pasciak, and Alfred H. Schatz. *The construction of preconditioners for elliptic problems by substructuring, III*. Math. Comp., 51(184), 415–430, 1988.
- [16] James H. Bramble, Joseph E. Pasciak, and Alfred H. Schatz. *The construction of preconditioners for elliptic problems by substructuring, IV*. Math. Comp., 53(183), 1 – 24, 1989.
- [17] Philippe G. Ciarlet. *The Finite Element Method for Elliptic Problems*. North-Holland, 1978.
- [18] Wolfgang Dahmen e Angela Kunoth. *Multilevel Preconditioning*. Numer. Math. 63, 315 – 344, 1992.
- [19] Ronald A. Devore e Vasil A. Popov. *Interpolation on Besov Spaces*. Trans. Amer. Math. Soc. 305, 397 – 414, 1988.
- [20] W. Dörfler. *Hierarchical Bases for Elliptic Problems*. Math. Comp. 58(198), 513 – 529, 1992.
- [21] Maksymilian Dryja. *A Capacitance Matrix Method for Dirichlet Problem on Polygon Region*. Numer. Math. 39, 51–64, 1982.
- [22] Maksymilian Dryja. *A Finite Element - Capacitance Method for Elliptic Problems on Regions Partitioned into Subregions*. Numer. Math. 44, 153–168, 1984.
- [23] Maksymilian Dryja, e Olof B. Widlund. *Some Domain Decomposition Algorithms for Elliptic Problems*. Iterative Methods for Large Linear Systems (Kincaid e Hayes eds.) Academic Press, 1990.
- [24] Maksymilian Dryja, e Olof B. Widlund. *Multilevel additive methods for elliptic finite element problems*. Technical Report 507, Courant Institute, 1991.
- [25] H. C. Elman e Xian-Zhong Guo. *Performance Enhancements and Parallel Algorithms for Two Multilevel Preconditioners*. Siam J. Stat. Comput. 14(4), 890 – 913, 1993.
- [26] Anne Greenbaum, Congming Li e Zheng Chao. *Parallelizing Preconditioned Conjugate Gradient Algorithms*. Comput. Physics Comm. 53, 295 – 309, 1989.
- [27] Wolfgang Hackbusch. *Multi-Grid methods and applications*. Springer-Verlag, 1985.
- [28] Louis A. Hageman e David M. Young. *Applied Iterative Methods*. Academic Press, 1981.
- [29] H. Johnen e K. Scherer. *On The Equivalence of The K-functional and The Moduli of Continuity and some Applications*. Lect. Notes Math. 571, 119 – 140, 1977.
- [30] David E. Keyes e William D. Gropp. *A Comparison of Domain Decomposition Techniques for Elliptic Partial Differential Equations and Their Parallel Implementation*. Siam J. Stat. Comput. 8(2), 166 – 202, 1987.

- [31] Jan Mandel. *Two-level domain decomposition preconditioning for the p-version of finite element in three dimensions*. Internat. J. Numer. Methods Engrg., 29, 1095 – 1108, 1990.
- [32] Jan Mandel. *Iterative solvers by substructuring for the p-version finite element method*. Comput. Meth. Appl. Mech. Engin., 80, 117 – 128, 1990.
- [33] Jan Mandel. *On block diagonal and Schur complement preconditioning*. Numer. Math., 58, 79 – 93, 1990.
- [34] J. Maubach. *On The Sparsity Patterns of Hierarchical Finite Element Matrices*. em: Preconditioned Conjugate Gradient Methods. O. Axelsson e L. Yu Kolotilina (eds). Lect. Notes Math. 1457:79-104 (1990).
- [35] S. F. McCormick, e J. W. Ruge. *Unigrid for multigrid simulation*. Math. Comp. 41, 43 – 62, 1983.
- [36] P. Oswald. *On Function Spaces Related to Finite Element Theory*. Z. Anal. Anwend. 9, 43 – 64, 1990.
- [37] P. Oswald. *On Discrete Normes Estimates Related to Multilevel Preconditoners in The Finite Element Method*. Proceedings International Conference on the Constructive Theory of Functions, Varna, 1991.
- [38] Luca F. Pavarino. *Domain decomposition algorithms for the p-version finite element method for elliptic problems*. tese de PhD, Courant Institute, 1992.
- [39] Maria-Cecilia Rivara. *Design and Data Structure of Fully Adaptative, Multigrid, Finite-Element Software*. ACM Trans. Math. Software 10(3), 242 – 264, 1984.
- [40] Maria-Cecilia Rivara. *A Grid Generator Based on 4-Triangles Conforming Mesh-Refinement Algorithms*. Internat. J. Numer. Methods Engrg. 24, 1343 – 1354, 1987.
- [41] Barry F. Smith. *Domain decomposition algorithms for the partial differential equations of linear elasticity*. tese de PhD, Courant Institute, 1990.
- [42] Barry F. Smith. *A domain decomposition algorithm for elliptic problems in three dimensions*. Technical Report 519, Department of Computer Science, Courant Institute, October 1990.
- [43] Barry F. Smith and Olof B. Widlund. *A domain decomposition algorithm using a hierarchical basis*. SIAM J. Sci. Stat. Comput., 11(6), 1212 – 1220, 1990.
- [44] Charles H. Tong, Tony F. Chan, e C. C. Jay Kuo. *Multilevel Filtering Elliptic Preconditioners*. Siam J. Matrix Anal. Appl. 11(3), 403 – 429, 1990.
- [45] Charles H. Tong, Tony F. Chan, e C. C. Jay Kuo. *A domain decomposition preconditioner based on a change to a multilevel nodal basis*. Siam J. Stat. Comput. 12(6), 1486 – 1495, 1991.

- [46] Vidar Thomée. *Galerkin Finite Element Methods for Parabolic Problems*. Lect. Notes Math. 1054 (1990).
- [47] Richard S. Varga. *Matrix Iterative Analysis*. Prentice-Hall, 1962.
- [48] Jinchao Xu. *Iterative methods by space decompositions and subspace correction*. Siam Review, 34(4), 581 – 613, 1992.
- [49] Harry Yserentant. *On the multi-level splitting of finite element spaces*. Numer. Math., 49:379–412, 1986.
- [50] Harry Yserentant. *Hierarchical bases give conjugate gradient type methods a multigrid speed of convergence*. Appl. Math. Comp., 19:347–358, 1986.
- [51] Harry Yserentant. *Two preconditioners based on the multi-level splitting of finite element spaces*. Numer. Math., 58:163-184 (1990).
- [52] Xuejun Zhang. *Multilevel Schwarz Methods*. Numer. Math. 63, 521 – 539, 1992.

Apêndice A: Programas

A seguir, apresentamos a listagem dos programas utilizados. Tanto para o método HB quanto para o método BPX, o programa é basicamente o mesmo. Alterando-se apenas as rotinas **PCG**, **V_CICLO**, **INTERPOLACAO** e **RESTRICAO**. Optamos por apresentar o código referente ao método BPX. As rotinas específicas para HB são apresentadas na página 98.

O dados são fornecidos através das rotinas **F** e **G** (vide pag. 99)

```

C *** baidu:/home/posemap/leoib/tese/programas/BPX.for,
C Tue Feb 22 16:53:16 1994, edit by leoib ***
C
C BPX com resolucao de cholesk na malha Grossa (2 dimensional)
C
C PROGRAM PGC_BPX
C
C   PARAMETER (DIM_NIVEL = 8)
C   PARAMETER (DIM_RAIZ_MOS = (2**DIM_NIVEL)-1)
C   PARAMETER (DIM_MOS = DIM_RAIZ_MOS**2)
C   PARAMETER (DIM_MATRIZ = 13*DIM_MOS)
C   PARAMETER (DIM_INTEIROS = 8*DIM_MOS)
C   PARAMETER (DIM_NOS_MA_FIMA = DIM_MOS)
C   PARAMETER (DIM_NIVEL_GROSSA = 3)
C   PARAMETER (DIM_NOS_MA_GROSSA = ((2**DIM_NIVEL_GROSSA)-1)**2)
C   PARAMETER (DIM_GROSSA_MATRIZ = DIM_NOS_MA_GROSSA *
C     (DIM_NOS_MA_GROSSA / 2 + 1) + 1)
C
C   DIM_NIVEL = valor maximo para o nivel da malha mais fina
C   DIM_RAIZ_MOS = raiz quadrada do maximo total de nos =
C     = numero maximo em uma dimensao
C   DIM_MOS = numero maximo de nos
C     = DIM_MOS = (numero maximo em uma dimensao)**2
C
C   * as alocacoes abaixo estao definidas em DEFINE_VECTOR
C   DIM_MATRIZ = tamanho da area que armazena a matriz, o lado
C     = direito, o vetor solucao, etc... Configuracao:
C     = solucao = 1*DIM_MOS
C     = lado.direito = 1*DIM_MOS
C     = diagonal = 1*DIM_MOS
C     = porcao diagonal inferior <= 4*DIM_MOS (laplace)
C     = residuo = 1*DIM_MOS
C     = auxiliar 1 = 1*DIM_MOS
C     = auxiliar 2 = 1*DIM_MOS
C     = auxiliar 3 = 1*DIM_MOS
C     = auxiliar 4 = 2*DIM_MOS
C
C     ----- +
C     13*DIM_MOS
C
C   DIM_INTEIROS = tamanho da area que armazena os vetores de
C     = de inteiros Configuracao:
C     = I1 = 1*DIM_MOS
C     = I2 = 1*DIM_MOS
C     = I_MATRIZ = 1*DIM_MOS
C     = TAM_LINHA = 1*DIM_MOS
C     = J_MATRIZ <= 4*DIM_MOS (laplace)
C
C     ----- +
C     8*DIM_MOS
C
C   DIM_NOS_MA_FIMA = total de nos criados na malha mais fina
C   * aproximada por DIM_MOS pois esta associada a inteiros
C   DIM_NIVEL_GROSSA = nivel maximo da malha inicial do processo
C
C
C   * de refinamento
C   DIM_NOS_MA_GROSSA = numero maximo de nos na malha inicial
C   DIM_GROSSA_MATRIZ = armazenamento necessario para a matriz
C     = na malha mais Grossa
C
C
C   DOUBLE PRECISION VECTOR_REAL(1:DIM_MATRIZ), PASSO_H, SUPREMO,
C     $ AUX, SUP_RES, SUP_ERR,
C     $ GROSSA_MATRIZ(1:DIM_GROSSA_MATRIZ), GROSSA_PASSO_H,
C     $ GROSSA_LADO_DIREITO(1:DIM_NOS_MA_GROSSA)
C
C   VECTOR_REAL = area (de tamanho DIM_MATRIZ) reservada para
C     = armazenamento dos vetores reais, a forma de armazenamento
C     = esta descrita na rotina DEFINE_VECTOR e esta associada aos
C     = indices IM_SOL, IM_LD, IM_DIAG, IM_L_MATRIZ e IM_REST
C   GROSSA_MATRIZ = area reservada para o armazenamento da
C     = porcao triangular inferior da matriz simetrica associada
C     = ao problema restrito a malha mais Grossa. Utilizada na
C     = resolucao completa, e portanto fatoracao de Cholesk
C   GROSSA_LADO_DIREITO = lado direito do sistema linear associado
C     = a malha mais Grossa. Utilizado na resolucao completa
C   PASSO_H = passo na malha mais fina. Utilizado para a formacao
C     = do sistema linear
C   GROSSA_PASSO_H = passo da malha mais Grossa. Utilizado quando
C     = feita a resolucao completa
C
C   INTEGER VECTOR_INTEIRO(1:DIM_INTEIROS), DIM_V1_VN(1:DIM_NIVEL),
C     $ NUM_NIVEIS,
C     $ REFINAMENTO(O:DIM_RAIZ_MOS+1,O:DIM_RAIZ_MOS+1), NUM_VAR,
C     $ FILA_IJ(1:DIM_NOS_MA_FIMA,1:2), TAM_FILA, INICIO_FILA,
C     $ DIM_FILA, MALHA_FIMA,MALHA_GROSSA,NUM_VAR,UNID, VARIABEL,
C     $ UNID_MAIS_UM, TAM_L_MATRIZ, I, J, LOC,
C     $ CONHECE_SOL, MAX_IT, PRECOND,
C     $ TAM_GROSSA_MATRIZ, GROSSA_NUM_VAR, GROSSA_UNID_MAIS_UM,
C     $ RESOLVE_GROSSA,
C     $ IM_SOL, IM_LD, IM_DIAG, IM_L_MAT, IM_REST, IM_I1, IM_I2, IM_L_MAT,
C     $ IM_TAM_LIN, IM_J_MAT, IM_REST, IM_VET1, IM_VET2, IM_VET3,
C     $ CONT_IT, IM_VET4
C
C   VECTOR_INTEIRO = area (de tamanho DIM_INTEIROS) reservada para
C     = armazenamento dos vetores inteiros, a forma de armazenamento
C     = esta descrita na rotina DEFINE_VECTOR e esta associada aos
C     = indices IM_I1, IM_I2, IM_I_M, IM_TAM_LIN e IM_J_M
C   IM_SOL = inicio do vetor SOLUCAO em VECTOR_REAL
C   IM_LD = inicio do vetor LADO_DIREITO em VECTOR_REAL
C   IM_DIAG = inicio do vetor DIAGONAL em VECTOR_REAL
C   IM_L_MAT = inicio do vetor L_MATRIZ em VECTOR_REAL
C   IM_REST = primeira posicao livre em VECTOR_REAL
C   IM_I1 = inicio do vetor I1 em VECTOR_INTEIRO
C   IM_I2 = inicio do vetor I2 em VECTOR_INTEIRO
C   IM_J_MAT = inicio do vetor J_MATRIZ em VECTOR_INTEIRO
C   IM_TAM_LIN = inicio do vetor TAM_LIN em VECTOR_INTEIRO
C   IM_J_MAT = inicio do vetor J_MATRIZ em VECTOR_REAL
C   TAM_L_MATRIZ = area ocupada pelo vetor L_MATRIZ

```



```

C      IM_VET1, IM_VET2, IM_VET3, IM_VET_4 = inicio de vetores
C      * auxiliares para calculos em CG ou PCG
C      MALHA_GROSSA = indica o nivel da malha mais grossa no processo
C      * de refinamento. O numero de variaveis (pontos interiores)
C      * tomados na malha mais grossa sera ((2*MALHA_GROSSA)-1)**2
C      MALHA_FINA = indica o nivel da malha mais fina no processo de
C      * refinamento. O numero de variaveis sera
C      * ((2*MALHA_FINA)-1)**2
C      NUM_VAR = numero total de variaveis criadas. Logo,
C      * NUM_VAR = ((2*MALHA_FINA)-1)**2
C      NUM_NIVEIS = numero total de niveis criados durante o
C      * processo de refinamento, onde malha_grossa = nivel 1.
C      * Temos NUM_NIVEIS = MALHA_FINA - MALHA_GROSSA + 1
C      DIM_VI_VN[I] = armazena a dimensao do espaco VI, isto e,
C      * o numero de variaveis criadas ate o nivel I do processo
C      * de refinamento. Logo, DIM_VI_VN[NUM_NIVEIS] = NUM_VAR
C      NUM_VAR_UNID = numero total de variaveis em uma dimensao,
C      * isto e, NUM_VAR_UNID = (2*MALHA_FINA)-1
C      REFINAMENTO = matriz que armazena o resultado final obtido apos
C      * o processo de refinamento. Cada variavel criada
C      * identifica-se por um valor inteiro I. REFINAMENTO[i,j]=1
C      * indica que a variavel I esta na posicao (i,j) da malha
C      * mais fina
C      FILA_IJ = fila utilizada preenchimento de REFINAMENTO. Armazena
C      * pares (i,j) de coordenadas. Sua manutencao esta associada
C      * TAM_FILA, INICIO_FILA e DIM_FILA
C      LOC = funcao inteira utilizada para armazenamento de uma matriz
C      * simetrica em um vetor
C      MAX_IT = numero maximo de iteracoes de PCG (ou CG) permitido
C      PRECOND = 1 indica que sera usado PCG com metodo BPX
C      * <>1 indica que sera usado CG
C      RESOLVE_GROSSA = 1 indica resolucao exata (Cholesky) na malha
C      * mais grossa (CBPX)
C      * <>1 indica IBPX
C
C      REAL TARRAY(2), TEMPO, DTIME
C
C      rotinas para dados iniciais no arquivo "dados.for"
C      EXTERNAL IMPRIME_FUNCODES, CHUTE_INICIAL, F, G
C
C      WRITE(*,*) "***** METODO BPX *****"
C
C      MAX_IT = 50
C      WRITE(*,*) "PRECOND=? (1=pre-condicionado, 0=nao)"
C      READ(*,*) PRECOND
C      IF (PRECOND.EQ.1) THEN
C          WRITE(*,*) "pre-condicionado: PCG com BPX"
C          WRITE(*,*) " RESOLVE_GROSSA=? (1=resolucao exata (cholesky) na
C          * malha grossa, 0=sem resolucao exata)"
C          READ(*,*) RESOLVE_GROSSA
C          IF (RESOLVE_GROSSA.EQ.1) THEN
C              WRITE(*,*) " resolucao exata (CBPX)"
C          ELSE
C              WRITE(*,*) " sem resolucao exata (IBPX)"
C          ENDIF
C      ELSE
C          WRITE(*,*) "nao pre-condicionado: CG"
C          ENDIF
C      WRITE(*,*) "nivel da malha fina="
C      READ(*,*) MALHA_FINA
C      IF (MALHA_FINA.GT.DIM_NIVEL) THEN
C          WRITE(*,*) " malha fina nao suportada"
C          STOP
C      ENDIF
C      WRITE(*,*) "nivel da malha fina=" , MALHA_FINA
C      WRITE(1,*) "nivel da malha fina=" , MALHA_FINA
C      MALHA_GROSSA = 2
C      IF (MALHA_GROSSA.GT.DIM_NIVEL_GROSSA) THEN
C          WRITE(*,*) " malha grossa nao suportada"
C          STOP
C      ENDIF
C      WRITE(*,*) "nivel da malha grossa=" , MALHA_GROSSA
C      WRITE(1,*) "nivel da malha grossa=" , MALHA_GROSSA
C
C      inicia contagem de tempo
C      TEMPO = DTIME(TARRAY)
C
C      MALHA_FINA = indicador do total de nos interiores
C      = (2*malha_fina - 1)**2
C      MALHA_GROSSA = indicador do total de nos na malha mais grossa
C      = (2*malha_grossa - 1)**2
C
C      NUM_NIVEIS = MALHA_FINA - MALHA_GROSSA + 1
C      numero de niveis
C
C      WRITE(*,*) "numero de niveis=" , NUM_NIVEIS
C      I = 2*MALHA_GROSSA - 1
C      I = I+1
C      WRITE(*,*) "com primeiro nivel (1) contendo " , I , " nos"
C
C      NUM_VAR_UNID = 2*MALHA_FINA - 1
C      numero de variaveis em uma dimensao
C
C      UNID_MAIUS_UNID = NUM_VAR_UNID + 1
C      NUM_VAR = NUM_VAR_UNID * NUM_VAR_UNID
C      WRITE(*,*) "total de nos interiores=" , NUM_VAR
C
C      CALL IMPRIME_FUNCODES
C
C      WRITE(*,*) "numero maximo de iteracoes de PCG=" , MAX_IT
C      WRITE(*,*)
C      IF ((PRECOND.EQ.1).AND.(RESOLVE_GROSSA.EQ.1)) THEN

```



```

WRITE (*,*) (VETOR_REAL(IH_VETI+1), I=0, NUM_VAR-1)

C
  calculo do residuo para a solucao encontrada
  CALL CALC_RES(VETOR_REAL(IH_LD), VETOR_REAL(IH_SOL),
    $ VETOR_REAL(IH_RESI), VETOR_REAL(IH_DIAG),
    $ VETOR_REAL(IH_L_MAT), VETOR_REAL(IH_I_MAT),
    $ VETOR_REAL(IH_TAM_LIN), VETOR_REAL(IH_J_MAT),
    $ NUM_VAR, TAM_L_MATRIZ)

  SUP_RES = 0
  DO 47 I = 0, NUM_VAR-1
    AUX = ABS(VETOR_REAL(IH_RESI+1))
    IF (AUX.GT.SUP_RES) SUP_RES = AUX
  CONTINUE

47
  WRITE (*,*)
  WRITE (*,*) "  sup do residuo= ", SUP_RES
  WRITE (*,*)
  STOP
END

C
  fim do programa principal
  #####

C
  INTEGER FUNCTION LOC( J, K)

C
  LOC - funcao inteira onde, dada a posicao (J,K) de uma matriz
  C simetrica, retorna um inteiro que indicando a correspondente
  C posicao num vetor que armazena a restricao triangular inferior
  C da matriz, linha a linha.

  INTEGER J, K

  IF (J.GE.K) THEN
    LOC = (J*(J-1) / 2) + K
  ELSE
    LOC = (K*(K-1) / 2) + J
  ENDIF
  RETURN
END

#####

C
  SUBROUTINE INSETE_FILA( I, J, FILA_IJ, INICIO_FILA, TAM_FILA,
    $ DIM_FILA)

C
  INSETE_FILA - insere o par I,J no fim da fila

  INTEGER I, J, FILA_IJ(1:DIM_FILA,1:2), INICIO_FILA, TAM_FILA,
    $ DIM_FILA

  INTEGER POSICAO

  POSICAO = INICIO_FILA + TAM_FILA
  IF (POSICAO.GT.DIM_FILA) POSICAO = INICIO_FILA - DIM_FILA
  FILA_IJ(POSICAO,1) = I
  FILA_IJ(POSICAO,2) = J
  TAM_FILA = TAM_FILA + 1
  IF (TAM_FILA.GT.DIM_FILA) WRITE(*,*) "OVERFLOW NA FILA"
  RETURN
END

#####

C
  SUBROUTINE REMOVE_FILA( I, J, FILA_IJ, INICIO_FILA, TAM_FILA,
    $ DIM_FILA)

C
  REMOVE_FILA - remove o primeiro par da fila e retorna em I,J

  INTEGER I, J, FILA_IJ(1:DIM_FILA,1:2), INICIO_FILA, TAM_FILA,
    $ DIM_FILA

  IF (TAM_FILA.LE.0) THEN
    WRITE(*,*) "ERRO: FILA VAZIA"
  ELSE
    I = FILA_IJ(INICIO_FILA,1)
    J = FILA_IJ(INICIO_FILA,2)
    INICIO_FILA = INICIO_FILA + 1
    IF (INICIO_FILA.GT.DIM_FILA) INICIO_FILA = 1
    TAM_FILA = TAM_FILA - 1
  ENDIF
  RETURN
END

#####

C
  SUBROUTINE CARREGA_MALHA_GROSSA( REFINAMENTO, VARIABEL,
    $ NUM_NIVEIS, UNID_MAIS_UM, FILA_IJ, INICIO_FILA, TAM_FILA,
    $ DIM_FILA)

C
  CARREGA_MALHA_GROSSA - inicializa a representacao dos nos da
  C malha mais grossa na matriz REFINAMENTO. VARIABEL retorna
  C o total de nos criados na malha mais grossa
  C obs- NAO carrega os formadores li e l2 dos pontos da malha grossa

  INTEGER REFINAMENTO(0:UNID_MAIS_UM,0:UNID_MAIS_UM), VARIABEL,
    $ NUM_NIVEIS, UNID_MAIS_UM, FILA_IJ(1:DIM_FILA,1:2),
    $ INICIO_FILA, TAM_FILA, DIM_FILA

  INTEGER I, J, FIM, INICIO, PASSO

  VARIABEL = -1
  DO 10 I = 0, UNID_MAIS_UM
    REFINAMENTO(0,I) = VARIABEL
    VARIABEL = VARIABEL - 1
  END DO

```



```

C      $      INICIO_FILA, TAM_FILA, DIM_FILA)
C
C      AUX2_REF_UNIF- rotina auxiliar a REFINA-UNIFORME. Computa a
C      formacao de um novo ponto por refinamento
C
C      INTEGER REFINAMENTO(O:UNID_MAIS_UM,O:UNID_MAIS_UM), VARIABEL,
C      $      UNID_MAIS_UM, I1(1:DIM_MOS), I2(1:DIM_MOS), DIM_MOS,
C      $      POS_I, POS_J, PASSO, K, L,
C      $      FILA_IJ(1:DIM_FILA,1:2), INICIO_FILA, TAM_FILA, DIM_FILA
C
C      INTEGER I, J, IAUX, JAUX
C
C      I = POS_I + K*PASSO
C      J = POS_J + L*PASSO
C      IF (REFINAMENTO(I,J).EQ.O) THEN
C          VARIABEL = VARIABEL + 1
C          REFINAMENTO(I,J) = VARIABEL
C          CALL IMSENE_FILA( I, J, FILA_IJ, INICIO_FILA, TAM_FILA, DIM_FILA)
C          IAUX = I - PASSO
C          JAUX = J - PASSO
C          I1(VARIABEL) = REFINAMENTO(IAUX,JAUX)
C          IAUX = I + PASSO
C          JAUX = J + PASSO
C          I2(VARIABEL) = REFINAMENTO(IAUX,JAUX)
C          IF (I1(VARIABEL).LT.O) THEN
C              IAUX = I1(VARIABEL)
C              I2(VARIABEL) = I2(VARIABEL)
C              I1(VARIABEL) = IAUX
C          ENDIF
C      ENDIF
C      RETURN
C      END
C
C      *****
C
C      SUBROUTINE REFINA-UNIFORME( REFINAMENTO, VARIABEL, NUM_NIVEIS,
C      $      UNID_MAIS_UM, I1, I2, DIM_MOS, DIM_V1_VM, DIM_NIVEL,
C      $      FILA_IJ, INICIO_FILA, TAM_FILA, DIM_FILA)
C
C      REFINA-UNIFORME- dada a representacao inicial do nos na malha
C      mais grossa na matriz de representacao REFINAMENTO, cria os
C      refinamentos uniformemente (nivel a nivel). VARIABEL indica o
C      total de nos ja existentes na malha mais grossa. carrega I1, I2
C      e DIM_V1_VM apropriadamente
C
C      INTEGER REFINAMENTO(O:UNID_MAIS_UM,O:UNID_MAIS_UM), VARIABEL,
C      $      NUM_NIVEIS, UNID_MAIS_UM, I1(1:DIM_MOS), I2(1:DIM_MOS),
C      $      DIM_MOS, DIM_V1_VM(1:DIM_NIVEL), DIM_NIVEL,
C      $      FILA_IJ(1:DIM_FILA,1:2), INICIO_FILA, TAM_FILA, DIM_FILA
C
C      INTEGER PASSO, POS_I, POS_J, TAREFAS, CONT_NIVEL
C
C      DIM_V1_VM(1) = VARIABEL
C      CONT_NIVEL = 1
C      PASSO = 2**(NUM_NIVEIS-1)

```

```

10 CONTINUE
C      IF (PASSO.LE.1) GOTO 50
C      while passo >= 1 do
C          PASSO = PASSO / 2
C          TAREFAS = TAM_FILA
C          numero de pontos para gerar refinamentos
C          CONTINUE
C          IF (TAREFAS.LT.1) THEN
C              CONT_NIVEL = CONT_NIVEL+1
C              DIM_V1_VM(CONT_NIVEL) = VARIABEL
C              GOTO 10
C          ENDIF
C          while tarefas >= 1 do
C              CALL REMOVE_FILA(POS_I, POS_J, FILA_IJ, INICIO_FILA, TAM_FILA,
C              $      DIM_FILA)
C              CALL AUX2_REF_UNIF( POS_I, POS_J, -1, 1, PASSO, REFINAMENTO,
C              $      VARIABEL, UNID_MAIS_UM, I1, I2, DIM_MOS, FILA_IJ,
C              $      INICIO_FILA, TAM_FILA, DIM_FILA)
C              CALL AUX1_REF_UNIF( POS_I, POS_J, 0, 1, PASSO, REFINAMENTO,
C              $      VARIABEL, UNID_MAIS_UM, I1, I2, DIM_MOS, FILA_IJ,
C              $      INICIO_FILA, TAM_FILA, DIM_FILA)
C              CALL AUX1_REF_UNIF( POS_I, POS_J, 1, 1, PASSO, REFINAMENTO,
C              $      VARIABEL, UNID_MAIS_UM, I1, I2, DIM_MOS, FILA_IJ,
C              $      INICIO_FILA, TAM_FILA, DIM_FILA)
C              CALL AUX1_REF_UNIF( POS_I, POS_J, -1, 0, PASSO, REFINAMENTO,
C              $      VARIABEL, UNID_MAIS_UM, I1, I2, DIM_MOS, FILA_IJ,
C              $      INICIO_FILA, TAM_FILA, DIM_FILA)
C              CALL AUX1_REF_UNIF( POS_I, POS_J, 1, 0, PASSO, REFINAMENTO,
C              $      VARIABEL, UNID_MAIS_UM, I1, I2, DIM_MOS, FILA_IJ,
C              $      INICIO_FILA, TAM_FILA, DIM_FILA)
C              CALL AUX1_REF_UNIF( POS_I, POS_J, -1, -1, PASSO, REFINAMENTO,
C              $      VARIABEL, UNID_MAIS_UM, I1, I2, DIM_MOS, FILA_IJ,
C              $      INICIO_FILA, TAM_FILA, DIM_FILA)
C              CALL AUX1_REF_UNIF( POS_I, POS_J, 0, -1, PASSO, REFINAMENTO,
C              $      VARIABEL, UNID_MAIS_UM, I1, I2, DIM_MOS, FILA_IJ,
C              $      INICIO_FILA, TAM_FILA, DIM_FILA)
C              CALL AUX1_REF_UNIF( POS_I, POS_J, 1, -1, PASSO, REFINAMENTO,
C              $      VARIABEL, UNID_MAIS_UM, I1, I2, DIM_MOS, FILA_IJ,
C              $      INICIO_FILA, TAM_FILA, DIM_FILA)
C              CALL AUX2_REF_UNIF( POS_I, POS_J, 1, -1, PASSO, REFINAMENTO,
C              $      VARIABEL, UNID_MAIS_UM, I1, I2, DIM_MOS, FILA_IJ,
C              $      INICIO_FILA, TAM_FILA, DIM_FILA)
C          TAREFAS = TAREFAS - 1
C          GOTO 20
C      RETURN
C      END
C
C      *****

```



```

SUBROUTINE MONTA_G_LAPLACIANO(MATRIZ, LADO_DIREITO, DIM_MATRIZ,
$ DIM_NUMS, PASSO_H, REFINAMENTO, RAIZ_NUM_VAR_MAIS_1)
C
C MONTA_G_LAPLACIANO -- dada a matriz REFINAMENTO com a disposicao
C dos nos no grid, retorna em MATRIZ a respectiva discretizacao
C de -laplaciano=F (contorno G e quadrado unitario) por diferencas
C finitas. MATRIZ e' um vetor que contem a representacao da porcao
C triangular inferior, e e' acessado pela funcao LOC.
C
$ DOUBLE PRECISION MATRIZ(1:DIM_MATRIZ), LADO_DIREITO(1:DIM_NUMS),
$ PASSO_H
$ INTEGER REFINAMENTO(O:RAIZ_NUM_VAR_MAIS_1,O:RAIZ_NUM_VAR_MAIS_1),
$ RAIZ_NUM_VAR_MAIS_1, DIM_MATRIZ, DIM_NUMS
C
$ INTEGER I, J, VARIAVEL, VIZINHO, LOC,
$ RAIZ_NUM_VAR
$ DOUBLE PRECISION H_QUAD, F, G, POS_I, POS_J
$ EXTERNAL LOC, F, G
C
C * zera a matriz
DO 5 I = 1, DIM_MATRIZ
$ MATRIZ(I) = 0
5 CONTINUE
$ H_QUAD = PASSO_H * PASSO_H
$ RAIZ_NUM_VAR = RAIZ_NUM_VAR_MAIS_1 - 1
$ POS_I = 0.0
C
C * extremidade horizontal
DO 60 I = 1, RAIZ_NUM_VAR
$ POS_I = POS_I + PASSO_H
$ POS_J = 0.0
60 CONTINUE
C
C * extremidade vertical
DO 50 J = 1, RAIZ_NUM_VAR
$ POS_J = POS_J + PASSO_H
$ VARIAVEL = REFINAMENTO(I,J)
$ MATRIZ(LOC(VARIAVEL,VARIAVEL)) = 4.0
$ LADO_DIREITO(VARIAVEL) = H_QUAD * F(POS_I,POS_J)
C
C * pesquisa vizinho superior (I,J+1)
$ VIZINHO = REFINAMENTO(I,J+1)
$ IF (VIZINHO.GE.VARIAVEL) GOTO 10
10 CONTINUE
C
C * atua so na triang inferior
$ IF (VIZINHO.LT.O) THEN
$ LADO_DIREITO(VARIAVEL)=LADO_DIREITO(VARIAVEL)
$ + G(POS_I,POS_J+PASSO_H)
$ ELSE
$ MATRIZ(LOC(VARIAVEL,VIZINHO)) = -1.0
$ ENDIF
10 CONTINUE
C
C * pesquisa vizinho inferior (I,J-1)
$ VIZINHO = REFINAMENTO(I,J-1)
$ IF (VIZINHO.GE.VARIAVEL) GOTO 20
20 CONTINUE
C
C * atua so na triang inferior
$ IF (VIZINHO.LT.O) THEN
$ LADO_DIREITO(VARIAVEL)=LADO_DIREITO(VARIAVEL)
$ + G(POS_I,POS_J-PASSO_H)
$ ELSE
$ MATRIZ(LOC(VARIAVEL,VIZINHO)) = -1.0
$ ENDIF
20 CONTINUE
C
C * pesquisa vizinho esquerda (I-1,J)
$ VIZINHO = REFINAMENTO(I-1,J)
$ IF (VIZINHO.GE.VARIAVEL) GOTO 30
30 CONTINUE
C
C * pesquisa vizinho direita (I+1,J)
$ VIZINHO = REFINAMENTO(I+1,J)
$ IF (VIZINHO.GE.VARIAVEL) GOTO 40
40 CONTINUE
C
C * atua so na triang inferior
$ IF (VIZINHO.LT.O) THEN
$ LADO_DIREITO(VARIAVEL)=LADO_DIREITO(VARIAVEL)
$ + G(POS_I+PASSO_H,POS_J)
$ ELSE
$ MATRIZ(LOC(VARIAVEL,VIZINHO)) = -1.0
$ ENDIF
40 CONTINUE
50 CONTINUE
60 CONTINUE
RETURN
END
$ #####
SUBROUTINE MONTA_LAPLACIANO(VECTOR_REAL, VECTOR_INTEIRO,DIM_MATRIZ,
$ DIM_INTEIROS, NUM_VAR,
$ TAM_L_MATRIZ, PASSO_H, REFINAMENTO, RAIZ_NUM_VAR_MAIS_1)
C
C MONTA_LAPLACIANO -- dada a matriz REFINAMENTO com a disposicao
C dos nos no grid, retorna em VECTOR_REAL a respectiva discretizacao
C de -laplaciano=F (contorno G e quadrado unitario) por diferencas
C finitas
C
$ DOUBLE PRECISION VECTOR_REAL(2:DIM_MATRIZ+1), PASSO_H
$ INTEGER REFINAMENTO(O:RAIZ_NUM_VAR_MAIS_1,O:RAIZ_NUM_VAR_MAIS_1),
$ RAIZ_NUM_VAR_MAIS_1, DIM_MATRIZ, NUM_VAR, TAM_L_MATRIZ,
$ VECTOR_INTEIRO(2:DIM_INTEIROS+1), DIM_INTEIROS
C
$ INTEGER I, J, VARIAVEL, VIZINHO, RAIZ_NUM_VAR, INICIO_F,
$ INICIO_DIAG, INICIO_L_MAT, AUX, AUX1, AUX2, AUX3,
$ INICIO_TAM_LIN, INICIO_J_MAT, PESQUISA, DIRECAO_I, DIRECAO_J
$ DOUBLE PRECISION H_QUAD, F, G, POS_I, POS_J, POSICAO_I, POSICAO_J
$ EXTERNAL F, G
C
$ inicializa a porcao diagonal estirita inferior como vazia
$ TAM_L_MATRIZ = 0

```



```

C          INICIO = L_MATRIZ(1)
C          CONT = 0
C          * enquanto CONT < FIM
20      IF (CONT.GE.FIM) GOTO 30
C          POS = INICIO+CONT
C          J = J_MATRIZ(POS)
C          * seja AUX = A(1,J) = A(J,1), I > J
C          AUX = L_MATRIZ(POS)
C          * diagonal inferior
C          SOL(1) = SOL(1) + AUX*v(J)
C          * diagonal superior
C          SOL(J) = SOL(J) + AUX*v(1)
C          CONT = CONT + 1
C          * fim do enquanto
C          GOTO 20
30      CONTINUE
C          RETURN
C          END
#####
C          SUBROUTINE CALC_RES( LADO,DIREITO, V, RES, DIAG_MATRIZ,
C          $ L_MATRIZ, I_MATRIZ, TAM_LINHA, J_MATRIZ, NUM_VAR,
C          $ TAM_L_MATRIZ)
C          CALC_RES -- calcula o resíduo LADO,DIREITO-A.V Onde a
C          matriz A corresponde a estrutura DIAG_MATRIZ e L_MATRIZ.
C          O resultado retorna no vetor RES.
C
C          DOUBLE PRECISION LADO,DIREITO(1:NUM_VAR), V(1:NUM_VAR),
C          $ RES(1:NUM_VAR), DIAG_MATRIZ(1:NUM_VAR),
C          $ L_MATRIZ(1:TAM_L_MATRIZ)
C          INTEGER NUM_VAR, TAM_L_MATRIZ, I_MATRIZ(1:NUM_VAR),
C          $ TAM_LINHA(1:NUM_VAR), J_MATRIZ(1:TAM_L_MATRIZ)
C          INTEGER I
C          calcula A.V em RES
C          CALL PROD_MATR_VET(V,RES,DIAG_MATRIZ,L_MATRIZ,I_MATRIZ,
C          $ TAM_LINHA, J_MATRIZ, NUM_VAR, TAM_L_MATRIZ)
C
C          DO 10 I = 1, NUM_VAR
C              RES(1) = LADO,DIREITO(1) - RES(1)
C          CONTINUE
C          RETURN
C          END
#####
C          SUBROUTINE RESTRIUGE(V1, V2, I1, I2, INICIO_NIVEL, FIM_NIVEL,
C          $ DIM1, DIM2, DIM_NOS)
C          RESTRIUGE - aplica o operador St descrito no metodo BPX
C
C          DOUBLE PRECISION V1(1:DIM1), V2(1:DIM2)
C          INTEGER INICIO_NIVEL, FIM_NIVEL, DIM_NOS, I1(1:DIM_NOS),
C          $ I2(1:DIM_NOS), DIM1, DIM2
C          INTEGER I, GERADOR
C          DOUBLE PRECISION AUX
C          DO 5 I = 1, DIM2
C              V2(1) = V1(1)
C          CONTINUE
C          DO 10 I = INICIO_NIVEL, FIM_NIVEL
C              GERADOR = I1(1)
C              IF (GERADOR.GE.O) THEN
C                  AUX = 0.5 * V1(1)
C                  V2(GERADOR) = V2(GERADOR) + AUX
C              ELSE
C                  V2(GERADOR) = V2(GERADOR) + AUX
C              IF (GERADOR.GE.O) V2(GERADOR) = V2(GERADOR) + AUX
C          ENDIF
C          CONTINUE
C          RETURN
C          END
#####
C          SUBROUTINE INTERPOLACAO( V, I1, I2, INICIO_NIVEL,
C          $ FIM_NIVEL, DIM_NOS)
C          INTERPOLACAO - aplica o operador S descrito no metodo BPX
C
C          DOUBLE PRECISION V(1:DIM_NOS)
C          INTEGER INICIO_NIVEL, FIM_NIVEL, DIM_NOS, I1(1:DIM_NOS),
C          $ I2(1:DIM_NOS)
C          INTEGER I, GERADOR
C          DOUBLE PRECISION AUX
C          DO 10 I = INICIO_NIVEL, FIM_NIVEL
C              GERADOR = I1(1)
C              IF (GERADOR.GE.O) THEN
C                  AUX = 0.5 * V(GERADOR)
C                  V(1) = V(1) + V(GERADOR)
C              ELSE
C                  V(1) = V(1) + V(GERADOR)
C              IF (GERADOR.GE.O) V(1) = V(1) + V(GERADOR)
C          ENDIF
C          CONTINUE
C          RETURN
C          END
#####
C          SUBROUTINE CHOLESK( MATRIZ, NUM_VAR, TAM_MATRIZ)

```

```

C      CHOLESK -- aplica o algoritmo de Cholesk sobre MATRIZ, que
C      vai receber L referente a sua representacao Llt
C
      DOUBLE PRECISION MATRIZ(1:TAM_MATRIZ)
      INTEGER NUM_VAR, TAM_MATRIZ
      INTEGER I, J, K, POS, LOC
      DOUBLE PRECISION SOMA, AUX
      EXTERNAL LOC
      DO 40 K=1, NUM_VAR
      DO 20 J=1, K-1
      SOMA = 0
      DO 10 I=1, J-1
      SOMA = SOMA + MATRIZ(LOC(J,I))*MATRIZ(LOC(K,I))
      CONTINUE
      POS = LOC(K,J)
      MATRIZ(POS) = (MATRIZ(POS) - SOMA) / MATRIZ(LOC(J,J))
      CONTINUE
      SOMA = 0
      DO 30 I=1, K-1
      AUX = MATRIZ(LOC(K,I))
      SOMA = SOMA + AUX*AUX
      CONTINUE
      POS = LOC(K,K)
      AUX = MATRIZ(POS) - SOMA
      MATRIZ(POS) = SQRT(AUX)
      CONTINUE
40      RETURN
      END
      *****

C      SUBROUTINE RES-CHOLESK ( MATRIZ, V, LADO_DIREITO, TAM_MATRIZ,
      $      NUM_VAR)
C
C      RES-CHOLESK -- dada MATRIZ com uma decomposicao de Cholesk,
C      retorna em V a resolucao dos dois sistemas triangulares associados
C      a Llt x = b
C
      DOUBLE PRECISION MATRIZ(1:TAM_MATRIZ), V(1:NUM_VAR),
      $      LADO_DIREITO(1:NUM_VAR)
      INTEGER NUM_VAR, TAM_MATRIZ
      DOUBLE PRECISION SOMA
      INTEGER I, J, LOC
      EXTERNAL LOC
      * Ly = b , y -> V
      DO 20 I=1, NUM_VAR
C
C      SOMA = LADO_DIREITO(I)
C      DO 10 J=1, I-1
C      SOMA = SOMA - V(J)*MATRIZ(LOC(I,J))
C      CONTINUE
C      V(I) = SOMA/MATRIZ(LOC(I,I))
C      CONTINUE
C
C      * Lt x = y , x -> V }
      DO 40 I=NUM_VAR, 1, -1
      SOMA = V(I)
      DO 30 J=I+1, NUM_VAR
      SOMA = SOMA - V(J)*MATRIZ(LOC(J,I))
      CONTINUE
      V(I) = SOMA/MATRIZ(LOC(I,I))
      CONTINUE
40      RETURN
      END
      *****

C      SUBROUTINE V-CICLO(W, V, RESOLVE_GROSSA, I1, I2, NUM_NIVEIS,
      $      DIM_V1, VM, DIM_MOS, DIM_V,
      $      DIM_NIVEL, GROSSA_MATRIZ, GROSSA_LADO_DIREITO,
      $      TAM_GROSSA_MATRIZ, GROSSA_NUM_VAR)
C
C      V-CICLO -- aplica o pre-condicionamento S D(-1) St referente a
C      adocao das bases multinivel nodais
C
      DOUBLE PRECISION W(1:DIM_MOS), GROSSA_MATRIZ(1:TAM_GROSSA_MATRIZ),
      $      GROSSA_LADO_DIREITO(1:GROSSA_NUM_VAR), V(1:DIM_V)
      INTEGER NUM_NIVEIS, DIM_MOS, DIM_NIVEL, DIM_V1, VM(1:DIM_NIVEL),
      $      I1(1:DIM_NIVEL), I2(1:DIM_NIVEL), DIM_V,
      $      TAM_GROSSA_MATRIZ, GROSSA_NUM_VAR, RESOLVE_GROSSA
      INTEGER NIVEL_ATUAL, INICIO_NIVEL, FIM_NIVEL, I, DIM1, DIM2,
      $      INIC1, INIC2
C      copia de W em V
      DO 5 I=1, DIM_MOS
      V(I) = W(I)
      CONTINUE
      S
C      mudanca da base nodal para multinivel
      INIC2 = 1
      DIM2 = DIM_V1-VM(NUM_NIVEIS)
      INICIO_NIVEL = DIM2 + 1
C
C      desce do v-ciclo
      DO 10 NIVEL_ATUAL=NUM_NIVEIS, 2, -1
      DIM1 = DIM2
      DIM2 = DIM_V1-VM(NIVEL_ATUAL-1)
      INIC1 = INIC2
      INIC2 = INIC2 + DIM1
      * [INICIO_NIVEL, FIM_NIVEL] indica os vertices criados em NIVEL

```



```

      FIM_NIVEL = INICIO_NIVEL-1
      INICIO_NIVEL = DIM2 + 1

C   * aplica o operador S(nivel) sobre V
      CALL RESTRIKE(V(INICI), V(INIC2), I1, I2, INICIO_NIVEL, FIM_NIVEL,
$      DIM1, DIM2, DIM_NOS)

10  CONTINUE

      INIC2 = INIC2 - 1
      IF (RESOLVE_GROSSA.EQ.1) THEN
        resolucao na malha mais grossa
      C
        FIM_NIVEL = DIM_V1_VN(1)
        DO 15 I = 1, FIM_NIVEL
          GROSSA_LADO_DIREITO(1) = V(INIC2+1)
          W(1) = 0
15      CONTINUE

        CALL RES_CHOLESK(GROSSA_MATRIZ, N, GROSSA_LADO_DIREITO,
$      TAM_GROSSA_MATRIZ, GROSSA_NUM_VAR)

      ELSE
C   * no caso de Bpx incompleto, nao faco nada para dimensao 2
        FIM_NIVEL = DIM_V1_VN(1)
        DO 17 I = 1, FIM_NIVEL
          W(1) = V(INIC2+1)
17      CONTINUE
        ENDIF

C   retorno as bases nodais
      INICI = INIC2
      FIM_NIVEL = DIM_V1_VN(1)

C   subida do v-ciclo
      DO 20 NIVEL_ATUAL = 2, NUM_NIVEIS
        INICIO_NIVEL = FIM_NIVEL + 1
        FIM_NIVEL = DIM_V1_VN(NIVEL_ATUAL)
        INICI = INICI - FIM_NIVEL

C   * aplica o operador S(nivel) sobre V
      CALL INTERPOLACAO(W, I1, I2, INICIO_NIVEL, FIM_NIVEL, DIM_NOS)

20  CONTINUE
      RETURN
      END

C *****
SUBROUTINE PCG( LADO_DIREITO, SOLUCAO, PRECOND, MAX_IT,
$
$   RESOLVE_GROSSA, DIAG_MATRIZ, L_MATRIZ, L_MATRIZ, TAM_LINHA,
$   J_MATRIZ, NUM_VAR, TAM_L_MATRIZ, I1, I2, DIM_V1_VN,
$   NUM_NIVEIS, DIM_NIVEL, RESIDUO, RES_PRECOND, P, AUX_VET,
$   BASE_BPX, GROSSA_MATRIZ, GROSSA_LADO_DIREITO,
$   TAM_GROSSA_MATRIZ, GROSSA_NUM_VAR, CONT_IT)
C
C   PCG -- aplica gradientes conjugados com pre-condicionamento
C   se PRECOND = 1. Caso contrario, aplica gradientes conjugados
C
      DOUBLE PRECISION LADO_DIREITO(1:NUM_VAR), SOLUCAO(1:NUM_VAR),
$      DIAG_MATRIZ(1:NUM_VAR), L_MATRIZ(1:TAM_L_MATRIZ),
$      GROSSA_LADO_DIREITO(1:GROSSA_NUM_VAR)
      INTEGER PRECOND, MAX_IT, NUM_VAR, TAM_L_MATRIZ,
$      I_MATRIZ(1:NUM_VAR), TAM_LINHA(1:NUM_VAR),
$      J_MATRIZ(1:TAM_L_MATRIZ), GROSSA_MATRIZ(1:TAM_GROSSA_MATRIZ),
$      NUM_NIVEIS,
$      DIM_NIVEL, DIM_V1_VN(1:DIM_NIVEL), I1(1:DIM_NIVEL),
$      I2(1:DIM_NIVEL), TAM_GROSSA_MATRIZ,
$      GROSSA_NUM_VAR, RESOLVE_GROSSA, CONT_IT

      DOUBLE PRECISION RESIDUO(1:NUM_VAR), RES_PRECOND(1:NUM_VAR),
$      AUX_VET(1:NUM_VAR), P(1:NUM_VAR), BASE_BPX(1:2*NUM_VAR),
$      ALPHA, BETA, AUX1, AUX2, SUP_ERRO

      INTEGER I

C   calculo do residuo inicial, dado o chute inicial em SOLUCAO
      CALL CALC_RES(LADO_DIREITO, SOLUCAO, RESIDUO, DIAG_MATRIZ, L_MATRIZ,
$      L_MATRIZ, TAM_LINHA, J_MATRIZ, NUM_VAR, TAM_L_MATRIZ)

      DO 10 I = 1, NUM_VAR
        RES_PRECOND(1) = RESIDUO(1)
10      CONTINUE

C   * pre-condicionado (PCG)
      CALL V_CICLO(RES_PRECOND, BASE_BPX, RESOLVE_GROSSA, I1, I2,
$      NUM_NIVEIS, DIM_V1_VN, NUM_VAR, 2*NUM_VAR, DIM_NIVEL,
$      GROSSA_MATRIZ, GROSSA_LADO_DIREITO,
$      TAM_GROSSA_MATRIZ, GROSSA_NUM_VAR)
      ENDIF

C
      AUX2 = 0
      DO 20 I = 1, NUM_VAR
        * copia do residuo em P
        P(1) = RES_PRECOND(1)

C   * AUX2 = < RES_PRECOND, RESIDUO >
        AUX2 = AUX2 + RES_PRECOND(1)*RESIDUO(1)

20      CONTINUE

C   processo iterativo com MAX_IT iteracoes
      CONT_IT = 1
25      CONTINUE

```



```

C      repita ate que CONT_IT > MAX_IT
      WRITE(*,*) "ciclo= ", CONT_IT

C      * ALPHA = < RES_PRECOND, RESIDUO > / < P, MATRIZ.P >
      CALL PROD_MATR_VET(P,AUX_VET,DIAG_MATRIZ,L,MATRIZ,I,MATRIZ,
$      TAM_LINHA,J,MATRIZ,MUM_VAR,TAM_L,MATRIZ)
      AUX1 = AUX2
      AUX2 = 0
      DO 30 I=1,MUM_VAR
        AUX2 = AUX2 + P(I)*AUX_VET(I)
      CONTINUE
      ALPHA = AUX1/AUX2

C      * atualiza a solucao
      DO 40 I=1,MUM_VAR
        SOLUCAO(I) = SOLUCAO(I) + ALPHA*P(I)
      CONTINUE

C      * aqui poderia fazer o teste de convergencia com SUP_ERR0
C      IF ((SUP_ERR0.IT.1.OO-3).AND.(SUP_ERR0.ANTERIOR.IT.1.OO-3))
C      $      RETURN

C      * RESIDUO = RESIDUO - ALPHA*MATRIZ.P
      DO 50 I=1,MUM_VAR
        RESIDUO(I) = RESIDUO(I) - ALPHA*AUX_VET(I)
      CONTINUE

C      DO 60 I=1,MUM_VAR
        RES_PRECOND(I) = RESIDUO(I)
      CONTINUE

C      IF ( PRECOND.EQ.1) THEN
        pre-condicionado

C      $      CALL V_CICLO(RES_PRECOND,BASE_BPX,RESOLVE,GROSSA,1,12,
$      $      MUM_NIVEIS,DIM_V1_VH,MUM_VAR,2*MUM_VAR,DIM_NIVEL,
$      $      GROSSA_MATRIZ,GROSSA_LADO_DIREITO,
$      $      TAM_GROSSA_MATRIZ,GROSSA_MUM_VAR)
      ENDIF

C      * BETA=<RES_PRECOND,RESIDUO>/<RES_PRECOND velho,RESIDUO velho>
      AUX2 = 0
      DO 70 I=1,MUM_VAR
        AUX2 = AUX2 + RES_PRECOND(I)*RESIDUO(I)
      CONTINUE
      BETA = AUX2/AUX1

C      * P = RES_PRECOND + BETA*P
      DO 80 I=1,MUM_VAR
        P(I) = RES_PRECOND(I) + BETA*P(I)
      CONTINUE

C      CONT_IT = CONT_IT + 1
      IF (CONT_IT.LE.MAX_IT) GOTO 25

C      fim do repita

```

```

      RETURN
END

```

```

C
SUBROUTINE RESTRINGE(V, I1, I2, INICIO_NIVEL, FIM_NIVEL, DIM_MOS)
  DOUBLE PRECISION V(1:DIM_MOS)
  INTEGER INICIO_NIVEL, FIM_NIVEL, DIM_MOS, I1(1:DIM_MOS),
    I2(1:DIM_MOS)
  INTEGER I, GERADOR
  DOUBLE PRECISION AUX

  DO 10 I = INICIO_NIVEL, FIM_NIVEL
    GERADOR = I1(I)
    IF (GERADOR.GE.O) THEN
      AUX = 0.5 * V(I)
      V(GERADOR) = V(GERADOR) + AUX
      GERADOR = I2(I)
      IF (GERADOR.GE.O) V(GERADOR) = V(GERADOR) + AUX
    ENDIF
  CONTINUE
  RETURN
END

C
SUBROUTINE INTERPOLACAO(V, I1, I2, INICIO_NIVEL, FIM_NIVEL,
  DIM_MOS)
  DOUBLE PRECISION V(1:DIM_MOS)
  INTEGER INICIO_NIVEL, FIM_NIVEL, DIM_MOS, I1(1:DIM_MOS),
    I2(1:DIM_MOS)
  INTEGER I, GERADOR
  DOUBLE PRECISION AUX

  DO 10 I = INICIO_NIVEL, FIM_NIVEL
    GERADOR = I1(I)
    IF (GERADOR.GE.O) THEN
      AUX = 0.5 * V(GERADOR)
      GERADOR = I2(I)
      IF (GERADOR.GE.O) AUX = AUX + 0.5 * V(GERADOR)
      V(I) = V(I) + AUX
    ENDIF
  CONTINUE
  RETURN
END

C
SUBROUTINE V_CICLO(V, RESOLVE_GROSSA, I1, I2, NUM_NIVEIS,
  DIM_V1_VN, DIM_MOS,
  DIM_NIVEL, GROSSA_MATRIZ, GROSSA_LADO_DIREITO,
  TAM_GROSSA_MATRIZ, GROSSA_NUM_VAR)
  V_CICLO -- aplica o pre-condicionamento S D (-1) St referente a
  adocao das bases hierarquicas

```

```

C
  DOUBLE PRECISION V(1:DIM_MOS), GROSSA_MATRIZ(1:TAM_GROSSA_MATRIZ),
    GROSSA_LADO_DIREITO(1:GROSSA_NUM_VAR)
  INTEGER NUM_NIVEIS, DIM_MOS, DIM_NIVEL, DIM_V1_VN(1:DIM_NIVEL),
    I1(1:DIM_NIVEL), I2(1:DIM_NIVEL),
    TAM_GROSSA_MATRIZ, GROSSA_NUM_VAR, RESOLVE_GROSSA
  INTEGER NIVEL_ATUAL, INICIO_NIVEL, FIM_NIVEL, I

  mudanca da base nodal para hierarquia
  INICIO_NIVEL = DIM_V1_VN(NUM_NIVEIS)+1
  DO 10 NIVEL_ATUAL=NUM_NIVEIS, 2, -1
    * ] INICIO_NIVEL, FIM_NIVEL [ indica os vertices criados em NIVEL
    FIM_NIVEL = INICIO_NIVEL-1
    INICIO_NIVEL = DIM_V1_VN(NIVEL_ATUAL-1)+1
    * aplica o operador S(nivel) sobre V
    CALL RESTRINGE(V, I1, I2, INICIO_NIVEL, FIM_NIVEL, DIM_MOS)
  CONTINUE

  IF (RESOLVE_GROSSA.EQ.1) THEN
    C
    resolucao na malha mais grossa
    * no caso de HB incompleto, nao faco nada para dimensao 2
    FIM_NIVEL = DIM_V1_VN(1)
    DO 15 I = 1, FIM_NIVEL
      GROSSA_LADO_DIREITO(I) = V(I)
      V(I) = 0
    CONTINUE
    CALL RES_CHOLESKY(GROSSA_MATRIZ, V, GROSSA_LADO_DIREITO,
      TAM_GROSSA_MATRIZ, GROSSA_NUM_VAR)
  ENDIF
  C
  retorno as bases nodais
  FIM_NIVEL = DIM_V1_VN(1)
  DO 20 NIVEL_ATUAL=2, NUM_NIVEIS
    INICIO_NIVEL = FIM_NIVEL + 1
    FIM_NIVEL = DIM_V1_VN(NIVEL_ATUAL)
    * aplica o operador S(nivel) sobre V
    CALL INTERPOLACAO(V, I1, I2, INICIO_NIVEL, FIM_NIVEL, DIM_MOS)
  CONTINUE
  RETURN
END

```

```

C      ** farofa:/home/posemap/leobj/lesee/programas/dados.for,
C      edit by leobj ***
C
C*****
C
C      SUBROUTINE IMPRIME_FUNCIONES
C
C      WRITE (*,*) "##"
C      WRITE (*,*) "## funcões dadas:"
C      WRITE (*,*) "##      f = 10.0DO+DSIN(3.0DO*X+Y)"
C      WRITE (*,*) "##      g = DSIN(3.0DO*X+Y)"
C      WRITE (*,*) "##      solucao = DSIN(3.0DO*X+Y)"
C
C      WRITE (*,*) "##"
C
C      WRITE (*,*) "##"
C      WRITE (*,*) "##      f = 0"
C      WRITE (*,*) "##      g = 0"
C      WRITE (*,*) "##      solucao = 0"
C
C      RETURN
C      END
C
C*****
C      DOUBLE PRECISION FUNCTION F(X,Y)
C      DOUBLE PRECISION X, Y, DSIN
C      F = 10.0DO+DSIN(3.0DO*X+Y)
C      F = 0
C      RETURN
C      END
C
C      DOUBLE PRECISION FUNCTION G(X,Y)
C      DOUBLE PRECISION X, Y, DSIN
C      G = DSIN(3.0DO*X+Y)
C      G = 0
C      RETURN
C      END
C*****
C      SUBROUTINE CHUTE_INICIAL ( V, DIM_V)
C
C      DOUBLE PRECISION V(1:DIM_V)
C      INTEGER DIM_V
C
C      CHUTE_INICIAL - inicializa o vetor solucao
C      INTEGER I
C
C      DOUBLE PRECISION AUX

```

```

C      WRITE (*,*) "## chute inicial constante = 1"
C      WRITE (*,*) "## chute inicial oscilante = -10 ou 10"

```

```

C      AUX = 10
C      DO 10 I = 1, DIM_V

```

```

C      V(I) = AUX
C      AUX = -AUX

```

```

C      V(I) = 1

```

```

C      10 CONTINUE
C      RETURN
C      END

```