UNIVERSIDADE FEDERAL FLUMINENSE

DIEGO NUNES BRANDÃO

Um Refinamento h-Adaptativo de Malhas para o Método dos Elementos Finitos utilizando uma Estrutura de Grafo.

Niterói

2008

UNIVERSIDADE FEDERAL FLUMINENSE

DIEGO NUNES BRANDÃO

Um Refinamento h-Adaptativo de Malhas para o Método dos Elementos Finitos utilizando uma Estrutura de Grafo.

Dissertação submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre em Computação. Aréa de concentração: Modelagem Computacional.

Orientador:
Dsc. Mauricio Kischinhevsky

Niterói

2008

Um Refinamento h-Adaptativo de Malhas para o Método dos Elementos Finitos utilizando uma Estrutura de Grafo.

DIEGO NUNES BRANDÃO

Dissertação submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre em Computação. Aréa de concentração: Modelagem Computacional.

Aprovada por:

Prof. D.Sc. Mauricio Kischinhevsky / IC-UFF (Presidente)

Prof. Ph.D. Carlos Antonio de Moura / IME-UERJ

Prof. D.Sc. Anselmo Antunes Montenegro / IC-UFF

Niterói, 08 de julho de 2008.

A Dona Tereza (*in memorian*)

"Brindo a casa, Brindo a vida

Meus amores, Minha família.

(...)

Tristeza e pesar, sem se entregar

Mal, mal vai passar, mal vou me abalar

(...)"

(Rodrigo Valle e Marcos Lobato)

Agradecimentos

Agradeço à minha família, em especial aos meus pais e avô pelo apoio e dedicação. Não poderia deixar de citar meus irmãos, minhas madrinhas e meu padrinho pelo incentivo. Ao meu orientador Prof Mauricio por todos os conselhos, idéias, desafios, paciência e conhecimento passado durante esses anos de mestrado. Aos professores do Instituto, em especial, aos professores Marco, José Henrique, Anselmo e Regina pelas inúmeras horas empregadas em conversas que possibilitaram a conclusão desse trabalho. Aos professores Isabel e Bazílio pelos ensinamentos durante as aulas na Petrobrás e aos professores da Rural, em especial, Eulina, Marcos, Rosane, Valdomiro, Roque. A professora Anna pelo apoio no CEDERJ.

Agradeço também a Ju, pelos desenhos, carinho, orações e toda atenção que me deram força para conclusão dessa etapa. Aos amigos da escola: Natália, Felipe, Delmo, Fernanda e toda a turma do IOB. Aos irmãos da Rural que me acolheram durante toda a graduação: Ademilton Luiz, Galeto, Felipão, Iran, Bruno, Fabi, Tatá, Marcio, Silvana, Henrique e Nica. Aos grandes amigos que me acompanharam desde a graduação e se tornaram minha família durante esse mestrado: Edgar (Gnomo), Augusto e Ildenir sem esquecer do Kennedy e sua família. Aos que surgiram durante o caminho: Sanderson, Mauricio, Edwin, Elder, Jonny e Carolina, a turma da aplicação.

Não poderia deixar de agradecer aos amigos do laboratório que fizeram as duras horas de trabalho tornarem-se momentos inesquecíveis: Janine, Jacques, Aline, Alexandre, Henrique Bueno, Coppeti, Cristiano, Idalmis, Luciana Brugiollo, Luciana Pessoa, Stenio, Jonivan, Guto, Joven, Familia Toso, Bertini, Renathinha, Adria, Alexandre Antunes, André, Silas, Miriam, Simone, Cris, Warley, Ary e todos os outros que não consegui

mencionar aqui.

Aos funcionários, Angela, Maria, João, Carlinhos, Dul, Rafael e Carlos pelo carinho dedicado e a uma pessoa em especial, Vera pela amizade, compreensão e apoio.

Acima de todas as coisas, a Deus, por me amar, me guiar e colocar tantas pessoas especiais em meu caminho. Muito Obrigado!

Resumo

Estratégias adaptativas podem ser de grande valia no processo de obtenção, por elementos finitos, das soluções computacionais de problemas oriundos de várias áreas da engenharia. Este trabalho apresenta uma implementação *h*-adaptativa do Método dos Elementos Finitos (MEF) baseada em uma estrutura de dados do tipo grafo para conectar os elementos presentes na discretização do domínio. Tal estrutura, inicialmente proposta para uso em discretizações por volumes finitos, é denominada ALG (*Autonomous Leaves Graph*). Nesta representação, uma curva de Hilbert modificada é empregada para definir a ordenação total dos elementos da malha. No caso do MEF, para cada elemento percorrido, os nós da malha geram contribuição que é armazenada em uma lista encadeada. Para a solução dos sistemas lineares resultantes da discretização por elementos finitos, um método iterativo de minimização de funcionais é empregado. Resultados numéricos mostram a viabilidade da integração de tais técnicas, ALG e MEF.

Abstract

In several areas of natural sciences and engineering phenomena are studied with the aid differential equations. Adaptive strategies can improve the efficency of computational processes, since they provide means of increasing the level of detail of the discretized problem only in regions where and when this is needed. This work presents an h-adaptive refinement for use with Finite Element Methods (FEM) which is based on a graph data structure to connect the existing elements of domain discretizations. This structure, named Autonomous Leaves Graph (ALG), was initially proposed for use with Finite Volume discretizations of boundary value problems (BVPs). In this representation, a Modified Hilbert Curve is employed in order to define the total ordering of the mesh. In the case of FEM, for each element swept, the mesh nodes generate a contribution which is stored in a chained list. In order to solve the linear systems arising from the FEM discretization of BVPs, an iterative method based on minimization of functionals is utilized. Numerical results show effectiveness and efficiency of integrating ALG concepts to the FEM framework.

Palavras-chave

1. Refinamento-h
2. ALG
3. Método dos Elementos Finitos
4. Malhas não-conformes
5. Equações Diferenciais Parciais

Glossário

ALG : *Autonomous Leaves Graph*;

MEF : Método dos Elementos Finitos;

CHM : Curva de Hilbert Modificada;

EDP : Equação Diferencial Parcial;

SFC : *Space Filling Curve*;

OO : *Object Oriented*;

CC : Condição de Contorno;

EG : Elemento Geométrico;

GC : Método dos Gradientes Conjugados;

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Métodos que utilizam malha adaptativa podem ser aplicados em uma variedade de áreas da física e engenharia, tais como dinâmica dos fluídos, combustão, transferência de calor, ciência dos materiais, etc. Os fenômenos físicos nessas áreas desenvolvem uma dinâmica natural, em geral dependente do tempo, ocorrendo mudanças rápidas na solução ou na sua derivada em certas regiões do domínio. Como exemplos podem-se citar ondas de choque, camadas limite, zonas de reação em processos de combustão. Nesses casos, a malha deveria conter uma grande quantidade de pontos nessas regiões. Fazer com que todo o domínio tenha a mesma quantidade de pontos implicaria num esforço computacional muito alto na solução do problema. Uma alternativa é o uso de técnicas adaptativas que oferecem certo grau de robustez, eficiência e confiança [1].

Um refinamento computacionalmente eficiente é fundamental para a solução de equações diferenciais devido à quantidade de variáveis envolvidas. Objetivando eficiência computacional, foi proposto em 1978 por Babuska e Rheinboldt [2] a idéia de refinamento das malhas de elementos finitos através da análise do erro de aproximação. Já em 1980 Rheinboldt e Mesztenyi [3] apresentam uma estrutura de dados eficiente que facilitava a utilização do refinamento da malha para elementos quadrangulares e triangulares. Tal estrutura, denominada *quadtree*, é uma árvore de refinamentos, onde o nodo pai representa a malha original e cada novo filho representa um novo refinamento, reproduzindo a idéia de refinamento de forma recursiva. Para garantir a eficiência de tal método, nem

todos os nodos precisam gerar filhos. Em 1985, Oden *et al.* [4] propõem um melhoramento na *quadtree* para elemento quadrilaterais lineares; nos anos seguintes Devloo [5] apresenta novas estruturas para elementos hexaédricos e refinamentos do tipo *h-p* que serão discutidos a diante.

Em 1998 Burgarelli [6] propôs uma estrutura de baixo custo computacional, cuja eficiência foi comprovada em 2006 [1], para representar refinamento de malhas na solução de equações diferenciais, que empregava o Método dos Volumes Finitos (MVF) com volumes quadrangulares. A estrutura consiste em utilizar um grafo de folhas autônomas denominado ALG, sendo a ordenação dos volumes da malha feita através de uma Modificação da Curva de Hilbert. Em 2006, Robaina [7] estendeu tal estrutura para o caso de visualização científica em $3D$. Gonzaga *et al.*, [8], também propuseram uma modificação nessa estrutura para aplicá-la em volumes finitos triangulares, utilizando a curva de Sierpinski na ordenação dos volumes.

Entretanto, para o caso de Elementos Finitos, a aplicação de *quadtree* ou mesmo do ALG depara-se com o problema da subdivisão do elemento resultando que: (i) a uniformidade dos elementos pode não ser satisfatória após o refinamento, ocasionando um mal condicionamento da matriz do sistema de equações; (ii) há perda de continuidade da solução nas interfaces entre elementos refinados e elementos não refinados, o que compromete a convergência do Método dos Elementos Finitos.

Para o caso de malhas com elementos quadrangulares com a divisão adotada neste trabalho, o problema da uniformidade dos elementos não afetará o condicionamento do sistema. Já o problema de aparecimento de nós na interface de elementos de grau de refinamento diferentes é resolvido através de restrições dos nós [9]. Outras soluções seriam o uso de funções de penalidade ou multiplicadores de Lagrange [10, 11], o que aumenta consideravelmente a complexidade do problema. Cabe observar que o problema do aumento da largura de banda do sistema oriundo da aplicação do MEF é contornado facilmente aplicando métodos iterativos e, como será feito aqui, aplicando uma estratégia em que

somente os elementos não-nulos são armazenados.

O objetivo deste trabalho é apresentar a implementação de um procedimento h-adaptativo em elementos finitos utilizando como estrutura de dados o grafo de folhas autônomas proposto em [1]. Para garantir a continuidade entre as interfaces de elementos adjacentes impõem-se restrições entre as funções de forma dos elementos envolvidos. Este trabalho está estruturado da seguinte forma:

O capítulo 2 apresenta uma introdução sobre alguns aspectos do Método dos Elementos Finitos, partindo da formulação do resíduo para problemas de valor de contorno até a construção da matriz de elementos finitos.

No capítulo 3 são apresentadas algumas definições sobre malhas e as estruturas computacionais utilizadas para representá-la, árvores do tipo *quadtree* e o grafo de folhas autônomas proposto em [1]. Além disso, também é discutida a utilização da curva de Hilbert Modificada para a ordenação dos elementos da malha e como aplicá-la ao caso dos Elementos Finitos.

Aspectos sobre a implementação e orientação a objetos são discutidos no capítulo 4, onde são apresentadas as classes e estruturas utilizadas. Nesse capítulo também são discutidas as questões sobre continuidade da solução, onde é apresentado um procedimento geral para garantir a continuidade entre elementos de níveis diferentes e é particularizado o caso para os elementos quadriláteros.

Por fim, no capítulo 5 são apresentados os testes computacionais e análise dos resultados obtidos. Conclusões e trabalhos futuros são discutidos no capítulo 6.

Capítulo 2

Resolução de Equações Diferenciais

Os espaços de elementos finitos foram introduzidos por Courant em [12]. Entretanto somente nos anos 60 tal técnica começou a ser empregada na área de Engenharia [13, 14]. Nas décadas seguintes ocorre a integração do Método dos Elementos Finitos (MEF) dentro do escopo de análise funcional, com a generalização de formulação fraca, demonstração de convergência para problemas elípticos [15] e caracterização de problemas lineares bem postos [16]. Isto possibilitou ao MEF uma forte base teórica e prática com uma vasta área de aplicações (eletromagnetismo, finanças, meios porosos, etc).

Neste capítulo é feita uma breve introdução ao Método dos Elementos Finitos como uma extensão do Método de Galerkin, apresentando os conceitos básicos sobre resíduos, funções de base, estimadores de erro e estratégias adaptativas.

2.1 Método dos Resíduos Ponderados

Considere um operador diferencial ordinário $\pounds$ e a Equação 2.1, onde Ω representa um domínio convexo limitado em $\Re^2$.

$$\pounds u = f(x), x \in \Omega \subset \Re^2 \tag{2.1}$$

As condições de contorno são homogêneas sobre o contorno poligonal $\partial\Omega$.

O Método dos resíduos ponderados consiste em obter uma solução aproximada, u^h, para o problema (2.1) na forma:

$$u^h \approx u_n(x) = \sum_{j=1}^{n} c_j \Phi_j(x) \tag{2.2}$$

de maneira que as funções Φ_j sejam linearmente independentes e que satisfaçam as condições de contorno em $\partial\Omega$ para qualquer coeficiente c_j. Assim o resíduo da aproximação é dado pela Equação 2.3.

$$R(x) = \pounds u_n(x) - f(x) \tag{2.3}$$

Observando a linearidade do operador $\pounds$, a equação acima pode ser reescrita como:

$$R(x) = \sum_{j=1}^{n} c_j \pounds \Phi_j(x) - f(x) \tag{2.4}$$

Agora deve-se escolher os coeficientes c_j de modo que a integral ponderada do resíduo

$$(w_i, R(x)) = \int \sum_{j=1}^{n} w_i c_j \pounds \Phi_j(x) - f(x) \tag{2.5}$$

seja nula sobre cada $i = 1, \ldots, n$. Assim, define-se um conjunto de equações que, aplicadas as igualdades provenientes da Equação 2.4, forma o sistema de equações algébricas 2.6, onde os coeficientes c_j são denominados graus de liberdade do sistema.

$$\left(\sum_{j=1}^{n} \pounds \Phi_j, w_i \right) c_j = (w_i, f), 1 \leq i \leq n \tag{2.6}$$

A escolha das funções de peso,w_i, determina qual o método dos resíduos ponderados a ser usado (Método da Colocação, Galerkin, Ritz, etc). No método de Galerkin as funções de peso e forma são escolhidas no mesmo espaço de aproximação.

2.2 Método de Galerkin

O Método de Aproximação de Galerkin foi proposto em 1913 por Bubnov sendo por isso também conhecido como Método de Bubnov-Galerkin. Esse método procura estabelecer a condição de ortogonalidade entre o erro e as funções Φ_j, para isso faz o espaço das funções de peso ser igual ao das funções Φ_j. Dessa maneira a equação (2.6) torna-se:

$$\left(\sum_{j=1}^{n} \pounds\Phi_j), \Phi_i\right) c_j = (\Phi_i, f) \tag{2.7}$$

com $i = 1, \ldots, n$.

Porém, se o resíduo é ortogonal as funções de forma e estas formam um espaço enumerável completo, obtem-se que, quando n cresce arbitrariamente ($n \to \infty$), o resíduo, ($R(x)$) aproxima-se de zero, ou seja, $R(x) \to 0$. Entretanto, o que é feito, de fato, é; fixado n e dado $\epsilon > 0$ arbitrário obtem-se $|R(x)| < \epsilon$ para n suficientemente grande.

Assim é obtida a convergência do Método de Galerkin. Cabe salientar que a existência e unicidade para o problema de Galerkin são dadas de maneira imediata pela aplicação do Lema de Lax-Milgran [17].

2.3 Método dos Elementos Finitos

Apesar de ser um método interessante para solucionar problemas de equações diferenciais com valor de contorno, o Método de Galerkin tem uma deficiência em relação a como construir as funções Φ_j. Ele não oferece nenhuma sistematização para obtê-las e uma escolha errada de tais funções pode gerar um mau condicionamento do sistema dado pela Equação 2.7, afetando assim a solução a ser obtida [18].

O Método dos Elementos Finitos (MEF) fornece uma técnica geral e sistemática para construir tais funções para as aproximações de Galerkin para problemas de valor de contorno [18]. A idéia fundamental do método consiste em particionar o domínio de interesse em pequenos pedaços interligados apenas por alguns pontos chamados pontos

nodais. Sobre cada elemento a solução é aproximada por uma combinação linear de funções de forma ψ_j^e, que são polinômios de baixo grau definidos por partes. A solução sobre cada elemento pode ser vista como:

$$u_h^e = \sum_{j=1}^{N_e} c_j^e \psi_j^e(x), \qquad e = 1, 2, \ldots, m \tag{2.8}$$

sendo N_e o número de graus de liberdade em um elemento e m o número total de elementos.

As funções de forma ψ_j^e são construídas de modo que ela e possivelmente algumas de suas derivadas tenham o valor 1 ou 0 nos pontos nodais de cada elemento. Para o caso dos elementos quadriláteros essas funções podem ser vistas na Figura 2.1.

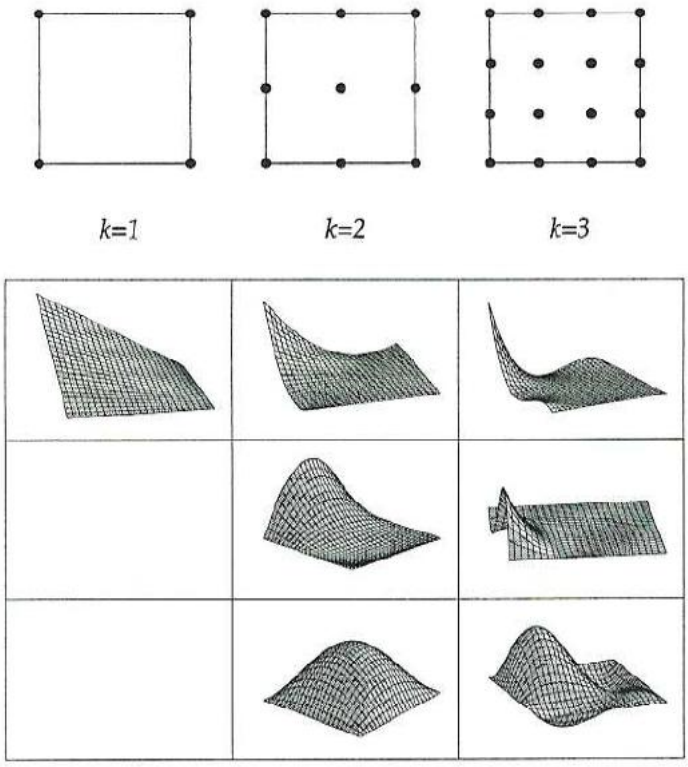


Figura 2.1: Funções de Forma do Elemento Quadrilátero.

Já a construção das funções globais Φ é feita por um processo de união das funções de forma locais ψ_j^e, de modo que as funções Φ e, possivelmente, algumas de suas derivadas se anulem em todos os nós do domínio, com exceção do nó que se está considerando, conforme apresentado na Figura 2.2. Tais funções são ditas de suporte compacto.

Computacionalmente, o MEF consiste em discretizar o domínio original em elementos

e a partir da formulação do problema constroem-se as aproximações sobre cada elemento. A construção da matriz do sistema, também conhecida como matriz de rigidez é feita através da soma das contribuições de cada elemento. Esse fato dá um caráter computacional interessante ao MEF. Após contruído o sistema aplicam-se as condições de contorno formando assim um novo sistema, que pode ser solucionado através de algum técnica de Álgebra Linear Computacional (Métodos Diretos ou Iterativos).

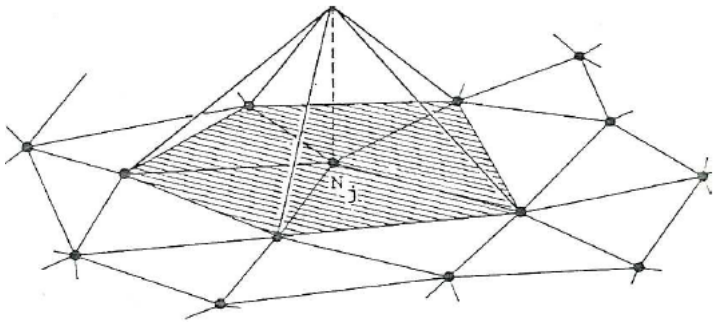


Figura 2.2: Funções de base com suporte compacto.

2.4 Estimadores de Erro

Estimar o erro em uma malha de elementos finitos é imprescindível para a adaptatividade. Através de um estimador define-se qual elemento deverá ser refinado ou qual grupo de elementos devem ser agrupados. Dessa maneira busca-se uma distribuição equitativa do erro por toda a malha. Estimadores de erro *a priori* são os mais utilizados em MEF, entretanto são dependentes de constantes que não podem ser obtidas o que dificulta sua utilização em procedimentos práticos. Já estimativas *a posteriori* são feitas em função do resíduo, permitindo assim sua utilização.

A escolha do estimador de erro a ser empregado depende do problema e da estratégia adaptativa adotada. Existem inúmeros trabalhos na área [19, 20, 21, 22].

O erro em uma solução numérica pode ser definido como:

$$e = u - u^h \tag{2.9}$$

onde u é a solução exata e u^h é a aproximação numérica. Entretanto, esta definição

local não é computacionalmente conveniente, e normas matemáticas são introduzidas para avaliar o erro. Uma das norma mais atrativas, que é facilmente aplicável ao MEF é a norma de energia do erro:

$$||e||^2 = \int_\Omega \nabla e.\nabla e d\Omega \tag{2.10}$$

Como a norma de energia é uma medida absoluta utiliza-se de maneira mais conveniente o erro relativo:

$$\eta = \frac{||e||}{||u||} \tag{2.11}$$

onde $||u||$ é a norma de energia exata.

Um critério simples de atingir a solução com um erro aceitável é [23]:

$$\eta \leq \eta_{max} \tag{2.12}$$

onde η_{max} é o percentual máximo de erro permitido no domínio Ω, e η é dado por:

$$\eta = \frac{||e||}{(||u^h||^2 + ||e||^2)^{\frac{1}{2}}} \tag{2.13}$$

Como mencionado para obter uma malha ótima, o erro deve ser equitativamente distribuído sobre todos os elementos, assim:

$$||e|| = \sqrt{m}||e||_i \tag{2.14}$$

onde m é o número de elementos na malha. O erro pode ser expresso para toda a malha a partir do erro admissível elementar [23]:

$$||e||_i \leq \eta_{max} \left(\frac{(||u^h||^2 + ||e||^2)}{m} \right)^{\frac{1}{2}} \equiv e_m^- \tag{2.15}$$

Assim, se

$$\xi_i = \frac{||e||_i}{e_m^-} > 1 \tag{2.16}$$

significa que o erro no elemento i ainda é maior do que o desejado e assim, ele precisa ser refinado; por outro lado, se esse valor é menor do que 1 significa que o elemento precisa

ser desrefinado.

2.5 Estratégia Adaptativa

Como já mencionado busca-se através da adaptatividade que o erro seja distribuído de maneira uniforme sobre toda a malha. Estratégias adaptativas para elementos finitos dividem-se, basicamente, em três tipos: h, p e r.

1. Método h - nesse método os elementos oriundos da discretização são subdivididos em elementos menores, mantendo-se constante o grau dos polinômios usados. É um método interessante para o tratamento de singularidades [22, 24].

2. Método p - consiste em enriquecer o espaço de solução aumentando-se o grau das funções de interpolação. Apresenta taxas de convergência mais alta que o método h em problemas suaves. Geralmente, o método p é efetivado por meio da introdução de funções hierárquicas, preservando assim as informações quando se passa de um nível de refinamento para o outro. Entretanto, resultados numéricos demonstram que não é um método satisfatório para problemas com presença de singularidade [22].

3. Método r - este esquema reposiciona os pontos nodais da malha de acordo com a distribuição de erros. Tem como vantagem o fato de manter inalterado o número de equações, porém a malha tem que ser rica o suficiente para fornecer um resultado satisfatório. É mais eficiente em problemas dinâmicos, nos quais a distribuição de erros se modifica em função do tempo.

A estratégia escolhida para o presente trabalho é baseada no método h e será detalhada no próximo capítulo, assim como algumas definições geométricas e aspectos computacionais para representá-las.

Capítulo 3

Malhas adaptativas

Neste capítulo serão apresentadas algumas definições sobre malha geométrica, malha computacional e estruturas de dados utilizadas para representá-las. Alguns aspectos sobre refinamento e ordenação da malha também serão abordados. O objetivo é apresentar alguns aspectos que auxilie na concepção da integração entre o MEF e o ALG [1].

3.1 Malha Geométrica

Uma malha é uma estrutura geométrica formada pela discretização do dominío em elementos geométricos (quadriláteros, triângulos, etc). Existem diferentes tipos de malhas (Figura 3.1). Uma malha não-conforme é aquela que os vértices de alguns elementos aparecem na aresta de outro elemento, mas não são vértices deste. As malhas podem ser formadas por diferentes tipos de elementos geométricos. Um elemento geométrico (EG) é uma entidade topológica que, neste trabalho é representado por um quadrilátero. A obtenção de tal quadrilátero se dá mediante uma bijeção contínua f em um elemento de referência geométrico contido em R^2.

Definição 1 *Um quadrilátero Q de referência é expresso por* $Q = \{(x, y) \in R^2 / -1 \leq x, y \leq 1\} \subseteq R^2$

Neste trabalho considera-se a definição geométrica de vértice, a saber:

Definição 2 *Um vértice é um ponto de união entre dois lados contíguos de um polígono.*

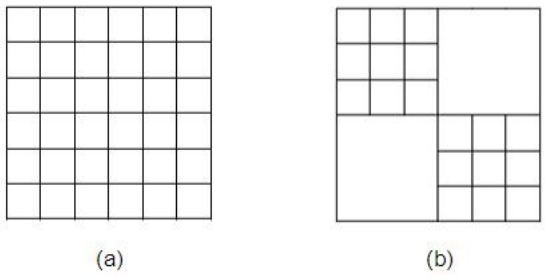


Figura 3.1: Exemplos de tipos de malha: (a) Malha conforme (b) Malha não-conforme.

No contexto de grafos será preterido o termo nodo ou nó, em favor de vértice. Para malhas não-conformes define-se um outro tipo de vértice, denominado vértice restrito.

Definição 3 *Um vértice restrito é um vértice de um polígono que faz parte do lado de um polígono vizinho, não sendo vértice deste.*

Esse tipo de vértice ocorre quando um elemento é dividido durante um refinamento e o seu vizinho não. O aparecimento de vértices restritos apresenta uma dificuldade a mais para o MEF, pois gera não continuidade entre as arestas de elementos distintos [25]. Neste caso as funções de forma associadas a elementos adjacentes podem ter grau diferentes o que gera uma descontinuidade da solução, que pode não representar uma aproximação da solução real do problema. No próximo capítulo será discutida a abordagem para solucionar tal problema.

3.2 Malha Computacional

Denomina-se malha computacional qualquer partição unitária $\wp_u$ do domínio Ω do problema, composta de elementos computacionais. Uma partição unitária é definida pelas duas propriedades a seguir.

- A união dos elementos da partição $\wp_u$ é igual a Ω.
- A interseção dos interiores de dois elementos quaisquer de $\wp_u$ é vazia.

Quando um elemento, e, da partição $\wp_u$ é refinado dando origem a outros elementos, ele é denominado nodo pai. Já os elementos (nodos) criados recebem a denominação de filhos. Em uma malha computacional refinada, pais e sub-elementos (filhos) computacionais não podem coexistir. Ou considera-se um elemento formando parte de uma partição do domínio ou seus sub-elementos obtidos por divisão. No refinamento p (ou h-p) a uma dada malha geométrica refinada podem-se associar várias malhas computacionais, devido ao refinamento das funções de interpolação; porém, para efetuar o cálculo de uma solução aproximada pelo MEF escolhe-se apenas uma malha computacional.

3.3 Estruturas de Dados

3.3.1 *Quadtree*

Uma *quadtree* é uma estrutura de dados hierárquica onde o domínio é um quadrado unitário. Este quadrado unitário é conhecido como raíz da estrutura de dados e pode ser decomposto em 4 novos elementos iguais, esses novos elementos também podem ser subdivididos até que um certo critério de parada seja alcançado. Os novos elementos são denominados filhos do elemento decomposto (nodo pai). Os nós filhos são nomeados de acordo com sua posição no elemento pai {nordeste (NE), sudeste (SE), noroeste (NW) e sudoeste (SW)}, como pode ser observado na Figura 3.2.

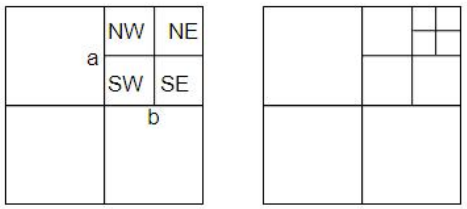


Figura 3.2: Representação de uma *quadtree*.

Um elemento é denominado folha ou terminal se não tem nenhum filho. Dois elementos

são chamados de vizinhos ou adjacentes se possuem uma aresta em comum. O nível de um elemento é igual ao número de decomposições que foram necessárias para obtê-lo. O nível da raiz é zero. Como pode ser visto, as informações relativas a uma *quadtree* lembram a estrutura de uma árvore e, por isso, as informações relativas a ela são armazenadas em uma árvore. Na Figura 3.3 é apresentada uma *quadtree* e sua representação correspondente a forma de uma árvore, observe que, internamente, cada nodo possui um ponteiro para seu pai.

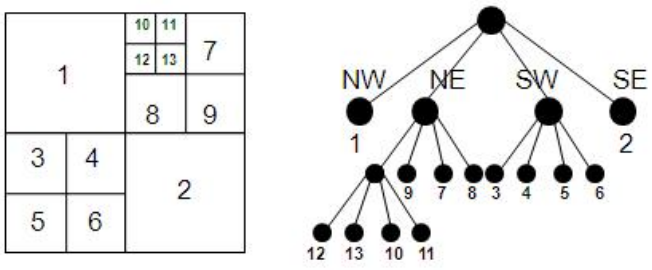


Figura 3.3: Representação de uma malha quadrangular com refinamento não uniforme e sua representação em árvore.

O emprego de uma estrutura de árvore permite representar informações tais como, nível de refinamento, conectividade, ancestral, vizinhos e filhos de um elemento. Com estas informações, outros dados que sejam necessários podem ser obtidos facilmente como, por exemplo, o vizinho de um dado elemento. Existem várias maneiras de armazenar uma *quadtree*, uma das quais é por meio de uma lista encadeada, [25].

Apesar da facilidade mencionada, o custo desse processo, em geral, é proporcional à altura da árvore. Observe que no processo de obtenção da matriz de rigidez é necessário calcular o valor da influência das funções de forma de um elemento vizinho no outro elemento sobre um determinado vértice comum. Assim, para a malha representada na Figura 3.3 é necessário saber a influência do elemento 1 sobre o elemento 12, e vice-versa. Tal processo pode tornar-se custoso, pois a quantidade de refinamentos pode aumentar consideralvemente a altura da *quadtree*. Observe que dessa maneira, verificar a influência entre dois vizinhos de níveis diferentes na árvore implica em percorrer toda a árvore,

aumentando o custo do algoritmo. A solução mais usada para evitar esse processo seria adotar a regra (2 : 1), na qual a diferença de níveis entre nós vizinhos é no máximo 1. Entretanto, em tal regra tem-se a obrigatoriedade de que o refinamento seja propagado por toda a malha sempre que algum elemento não a obedeça. Na próxima seção, é apresentada a estrutura de dados proposta por Burgarelli *et al* [1] que realiza refinamentos locais e, conseqüentemente, não se depara com tal restrição.

3.3.2 ALG (*Autonomous Leaves Graph*)

Como inicialmente proposto por Bugarelli *et al.* [1], considere um domínio bidimensional formado pelo quadrado unitário $\Omega = [0, 1]^2$. Dividindo esse quadrado em quatro quadrados iguais obtêm-se quatro células quadradas com aresta medindo $\frac{l}{2}$.

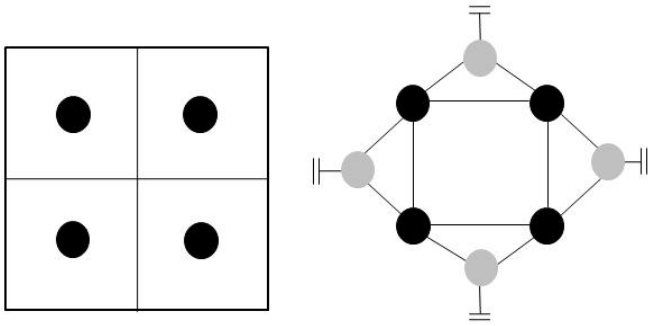


Figura 3.4: Quadrado unitário e estrutura de conexão dos nodos [1].

No centro de cada um desses quadrados é definido um ponto, denominado nodo preto. Informações pertinentes a cada célula são associadas a esses pontos, os quais, por sua vez, representam vértices de um grafo orientado, de acordo com a Figura 3.4. Nesta estrutura definiu-se a vizinhança de maneira análoga às direções geográficas, denominando vizinhos ao norte, sul, leste ou oeste. Além disso, para representar tal vizinhança cada nodo utiliza quatro ponteiros, que na Figura 3.4 são associados às arestas do grafo. Convencionou-se que o quadrado acima e à direita é o quadrado nordeste, e os demais, no sentido horário, são chamados respectivamente de quadrados sudeste, sudoeste e noroeste. Observando que o quadrado noroeste não possui vizinho à direita e nem vizinho acima, criou-se um

novo nodo, denominado nodo branco. Procedendo de maneira análoga aos demais quadrados, acrescentam-se os nodos brancos a toda a estrutura, representando as fronteiras do domínio [1].

Os nodos brancos também apontam para os nodos pretos que os apontam, além de um terceiro ponteiro que aponta para um vértice especial, denominado *vértice terra*.

A profundidade de uma célula, segundo Burgarelli *et al.* [1], pode ser vista como o número de passos realizados para sua obtenção, sendo convencionado que as células de uma malha composta de 4 células iguais estão no nível 1 de profundidade.

Um outra utilidade dos nodos brancos é a conexão dos nodos em diferentes níveis de profundidade. Nodos brancos conectam um nodo preto ou um branco de profundidade n em quatro nodos pretos ou brancos de profundidade $n + 1$.

3.3.2.1 Refinamento

Nesta seção ilustra-se a estrutura de dados obtida após um processo de refinamento do quadrado unitário que foi dividido em $M = 4 + 3k$ células quadradas q_j, $j = 1, ..., M$, onde k indica o nível de refinamento.

Burgarelli *et al.* [1], convencionaram que a malha inicial é composta por quatro células iguais que estão no nível 1 de profundidade no refinamento Figura 3.4. Uma célula de nível n de profundidade ao ser refinada é substituída por 4 células de nível $n + 1$ de profundidade. O nível de profundidade de uma célula está diretamente relacionado com o tamanho do lado da célula. Assim, uma célula de lado $\frac{1}{2^n}$ está no nível de profundidade n.

Sempre que uma célula é refinada, substitui-se o nodo preto correspondente por uma estrutura similar a estrutura inicial da malha. A Figura 3.5 representa a malha inicial (Figura 3.4) com um refinamento do elemento noroeste (NW). Vale ressaltar que a representação dos vértices terra foi suprimida para simplificação da figura.

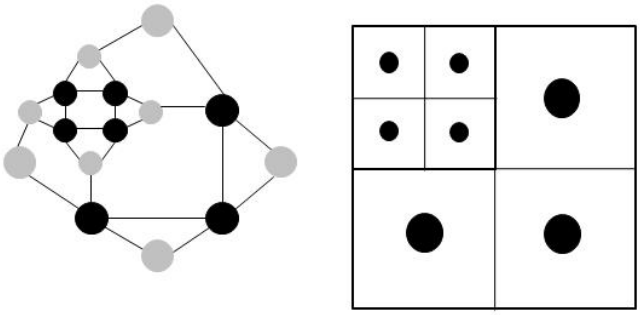


Figura 3.5: ALG e malha refinada [1].

Desta forma, sempre acrescenta-se um *cacho* no lugar de uma célula refinada. Nesse novo cacho cada nova célula, isto é, cada novo nodo preto, armazena uma informação de quem era o nodo pai. Essa observação é de suma importância para o desrefinamento como será visto a seguir. Se após o refinamento nodos brancos de mesma profundidade aparecem ligados, realiza-se uma simplificação da estrutura (Figura 3.6). Essa simplificação possibilita que o algoritmo de busca dos vizinhos de um nodo preto funcione eficientemente.

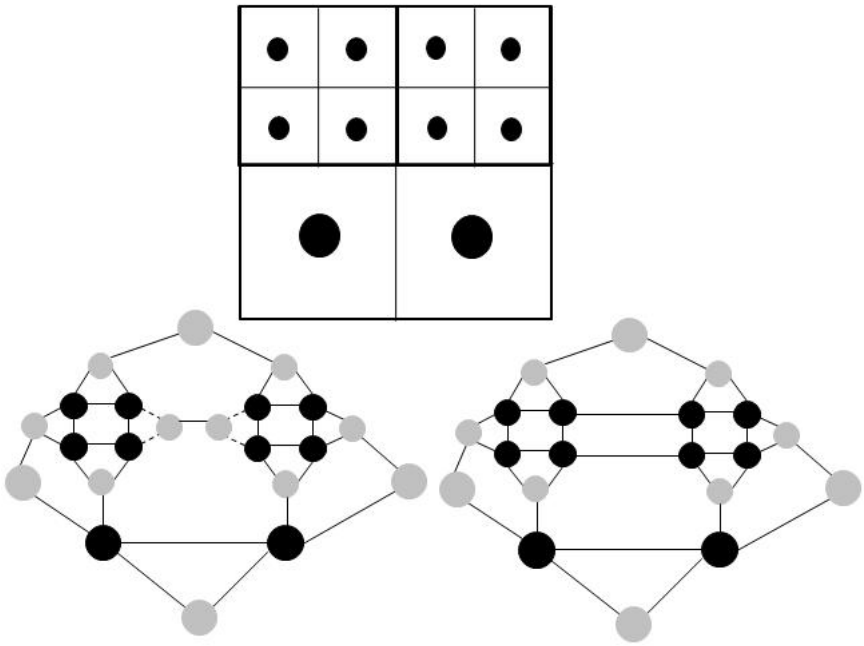


Figura 3.6: Simplificação de nodos brancos de mesmo nível [1].

3.3.2.2 Desrefinamento

O processo de desrefinamento no ALG só pode ser realizado em células que possuem três irmãos, isto é, em um mesmo nível de profundidade. Portanto, tais células pertencem ao mesmo cacho. Isto significa que quatro células de nível $n+1$ são substituídas por uma única célula de nível n (Figura 3.7). Essas células devem ser oriundas do mesmo pai, o que garante a configuração anterior da malha após o desrefinamento, essa observação deve-se à estrutura de geração da CHM [1]. A não obediência de tal restrição impediria a obtenção da CHM no passo anterior ao refinamento.

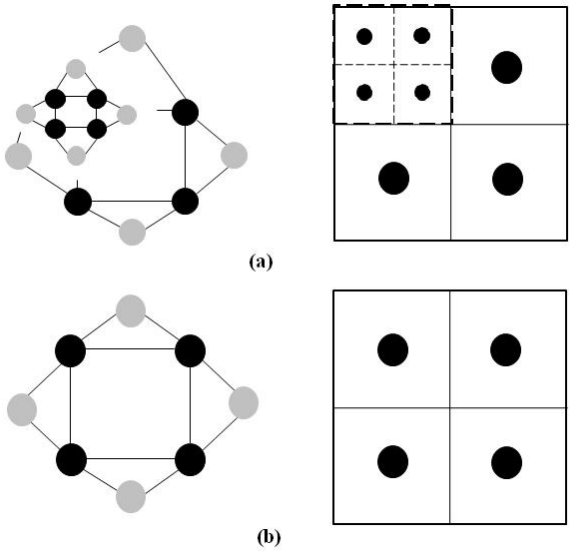


Figura 3.7: Desrefinamento da malha e representação do ALG [1].

Segundo Burgarelli *et al.* [1], as etapas necessárias ao desrefinamento são as seguintes:

1. criação de um novo nodo preto;
2. o nodo preto criado armazena suas coordenadas espaciais e profundidade;
3. ligação do nodo preto com seus vizinhos;
4. ligação dos vértices vizinhos com o nodo preto;
5. ordenação da malha pela curva de Hilbert;
6. liberação da área de memória dos nodos brancos e pretos que deixam de existir.

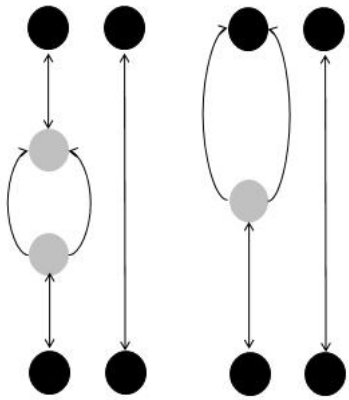


Figura 3.8: Desrefinamento de nodos brancos [1].

O passo (1) se dá pela transformação do vértice noroeste, em relação aos 4 vértices em questão, no vértice resultante do desrefinamento. Assim como no processo de refinamento, o processo de desrefinamento pode acarretar a destruição de nodos brancos, uma vez que nodos brancos de mesmo nível de profundidade não devem coexistir e nodos pretos filhos de mesmo pai devem ser conectados diretamente. A Figura 3.8 apresenta algumas das situações que podem ocorrer durante um desrefinamento e em [1] descrevem-se as metodologias adotadas para todos os casos.

3.4 Ordenação da Malha

A ordenação da malha é fundamental para métodos numéricos que envolvem discretizações de domínio pois afeta diretamente a matriz gerada [1, 3, 10, 24]. Nestes métodos as variáveis do sistema linear estão associadas as discretizações e por isso torna-se necessário alguma ordem nos vértices, nos elementos da malha ou em ambos. O ALG utiliza uma curva de preenchimento de espaço (*space-filling curve*) para garantir a ordenação da malha refinada.

3.4.1 Curva de Hilbert

Uma curva de preenchimento de espaço bidimensional pode ser definida como uma curva contínua tal que seu traço preenche todo o domínio, no caso aqui exposto, todo o quadrado unitário. Mais detalhes podem ser obtidos em [7, 26, 27].

Uma curva de Hilbert é o limite de uma sequência de curvas $H_1, H_2, H_3, ...$ Na Figura 3.9 são ilustrados os três primeiros passos da construção desta sequência. Assim, uma curva de Hilbert de ordem i é dita uma aproximação da curva de Hilbert de preenchimento do espaço. Portanto, as curvas H_i são aproximações e não a curva em si. O processo de obtenção da curva de Hilbert de ordem i se dá pela substituição dos vértices de H_i por curvas de ordem $i-1$ devidamente rotacionadas ou refletidas. O algoritmo de construção da curva de Hilbert pode ser obtido em [1, 7].

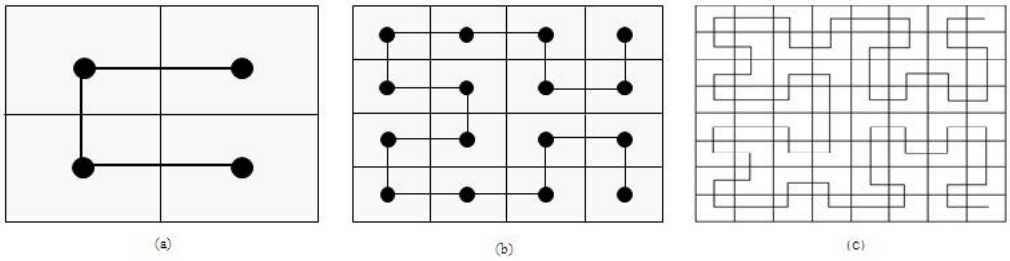


Figura 3.9: Sequência de curvas de Hilbert: (a) Curva de Hilbert H_1. (b) Curva de Hilbert H_2 e (c) Curva de Hilbert H_3 [1].

A estrutura ALG definida anteriormente faz analogia à curva de Hilbert para realizar uma ordenação da malha. Em cada nodo preto existem mais dois ponteiros, que consistem em uma lista duplamente encadeada, que ligam todos os nodos pretos da estrutura. Toda vez que uma célula é refinada um nodo preto é substituído na estrutura por quatro novos nodos pretos. Como se pode perceber, tal modificação é local e isso permite que seja utilizado um algoritmo baseado na construção da curva de Hilbert para que os elementos sejam ordenados [1]. A Figura 3.10 ilustra tal procedimento.

Pode-se observar que a curva de Hilbert apresentada foi modificada para não ficar restrita somente a refinamentos uniformes e, por isso, foi denominada curva de Hilbert

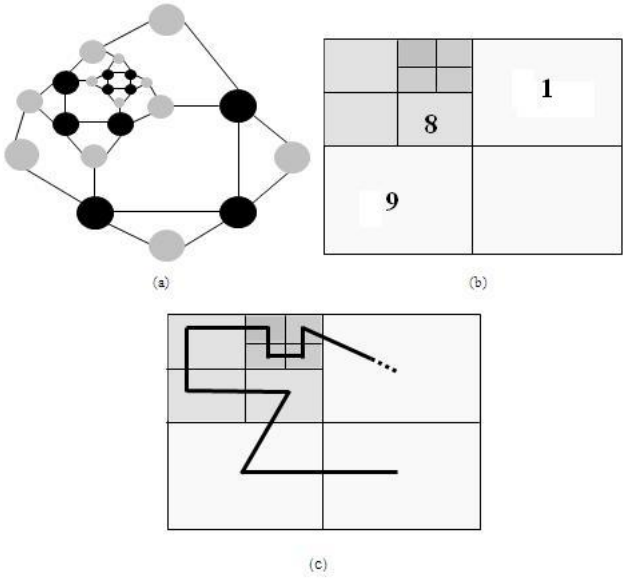


Figura 3.10: Refinamento da malha: (a) A estrutura do ALG. (b) Representação de Malha Refinada Não-estruturada. (c) A curva de Hilbert Modificada

Modificada (Figura 3.11). O ALG aproveita a propriedade de que em cada etapa i da construção da curva de Hilbert, um arco de comprimento muito grande fica contido em uma região de pequena área [7].

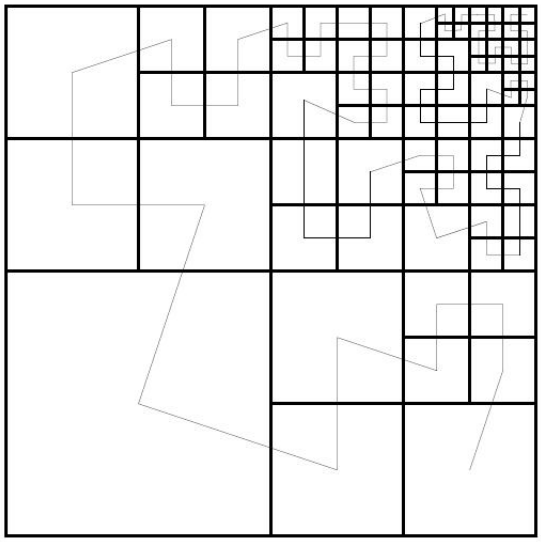


Figura 3.11: Curva de Hilbert Modificada bidimensional [1].

Capítulo 4

FEM-ALG: Resolutor de Equações Diferenciais

O ALG e o FEM foram apresentados distintamente nos capítulos prévios. Aqui será apresentado o programa desenvolvido integrando as duas técnicas. Demonstra-se os problemas de tal integração e como solucioná-los, para tanto são apresentados um algoritmo de Métodos de Elementos Finitos h-adaptativo, um algoritmo para ordenação dos vértices dos elementos e como ele afeta a matriz de rigidez. O capítulo é encerrado apresentando uma solução para o problema de vértices restritos e algumas noções de orientação a objetos na descrição da implementação do FEM-ALG, especificamente algumas classes e métodos.

4.1 Algoritmo de Refinamento

O algoritmo empregado nesse trabalho (Algoritmo 1) apresenta todas as medidas de erro consideradas para uma análise adaptativa de Elementos Finitos [24]. Tal algoritmo busca equalizar os indicadores de erro, ou seja, ele subdivide um elemento nos quais os indicadores de erro são maiores para obter uma distribuição uniforme do erro pela malha (Capítulo 2). Observa-se a característica de refinamento automático intrínseca de métodos adaptativos, onde a partir de um analisador de erros todo o processo do algoritmo de Elementos Finitos é reiniciado.

Algoritmo 1 Algoritmo Refinamento Adaptativo

```
1: geração da malha inicial;
2: enquanto malha inicial ou elementos refinados faça
3:    calculo dos coeficientes da matriz de rigidez;
4:    resolução do sistema linear;
5:    para todo elemento da malha faça
6:       estima erro
7:       se critério 2.16 verdadeiro então
8:          refina elemento
9:       senão
10:         desrefina um cacho
11:      fim se
12:   fim para
13:   se elementos refinados ou elementos desrefinados então
14:      reordena malha
15:   fim se
16: fim enquanto
```

A malha inicial é constituída com quatro elementos. Nesse passo também são definidas as conectividades (elementos, nós) e (nós, elementos). Tais conectividades são de suma importância para o cálculo dos coeficientes da matriz de rigidez (elementos, nós) e para a resolução do sistemas linear (nós, elementos).

A segunda etapa do algoritmo consiste nos processos naturais ao MEF. São calculadas as integrais relativas a Equação 2.8 e, utilizando uma técnica de tratamento de sistemas esparsos, os valores não-nulos são armazenados em uma lista encadeada. Os valores que representariam a diagonal principal da matriz de rigidez são armazenados nessa lista e a partir deles toda a linha da matriz é representada, utilizando uma outra lista Figura 4.1.

O sistema linear originado do MEF tem propriedades de simetria, pois o operador $\pounds$

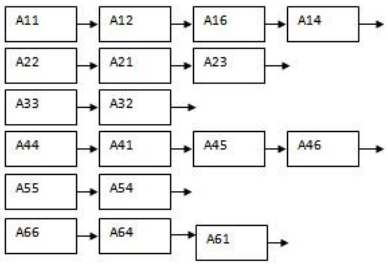


Figura 4.1: Representação da Matriz de Rigidez através de uma lista encadeada.

é simétrico [28]. Experimentos numéricos demonstraram a positividade das matrizes dos problemas abordados no presente trabalho.

4.1.1 Enumeração dos vértices

Algoritmos de enumeração de vértices são fundamentais em elementos finitos pois determinam se a matriz é em banda, faixa ou cheia, etc [29]. Existem inúmeros algoritmos que podem ser empregados na enumeração [29, 30]: Método de Gibbs, Método de Grooms, etc. Na maioria deles, é exigido a utilização de uma matriz que armazena a cada malha gerada, toda sua conectividade. O ALG, assim como a *quadtree*, não necessita utilizar tal matriz.

Originalmente o ALG utiliza a Curva de Hilbert Modificada para percorrer as células da malha [1]. Neste trabalho a CHM também é empregada, entretanto, a cada elemento percorrido uma enumeração dos vértices é estabelecida sendo analisado para isso toda sua vizinhança.

A cada elemento é associado um número, denominado, número de Hilbert que é obtido ao percorrer-se a CHM. Parte-se do príncipio que os vértices já enumerados pertencem a um elemento com número de Hilbert menor do que o elemento atual. Assim a partir de uma enumeração estática inicial, todos os vértices dos elementos que formam a malha são enumerados automaticamente.

No MEF cada (EG) da malha é mapeado em um elemento de referência (ER) através de uma função de mapeamento. Assim, todos os cálculos são feitos no ER e depois são

mapeados na malha.

Para evitar que o cálculo do jacobiano de tal transformação seja negativo, adotou-se o sentido antihorário para a numeração dos vértices do EG (Figura 4.2).

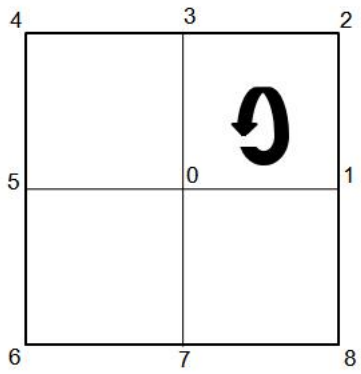
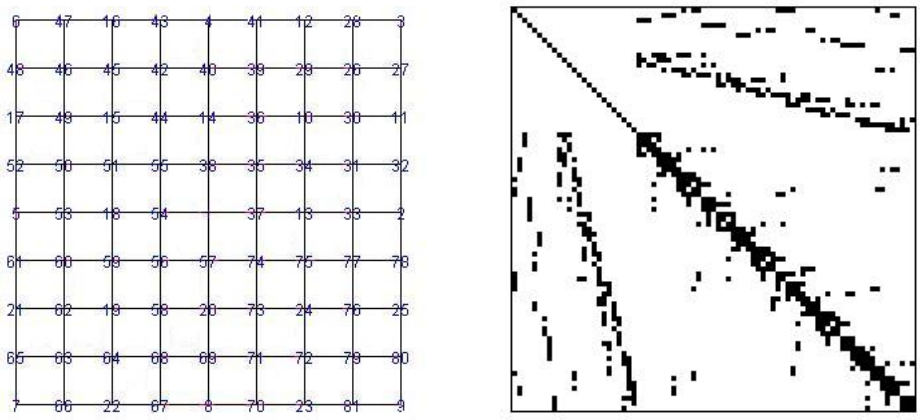


Figura 4.2: Enumeração da Malha no Sentido Anti-horário.

Salienta-se que a numeração é mantida a cada elemento percorrido ou refinado.

Uma desvantagem dessa escolha é que a matriz de rigidez deixa de ter um caráter de faixa como em uma grande parte de programas de elementos finitos, como pode ser observado na Figura 4.3. Entretanto tal exigência só é interessante quando utilizado algum método direto para resolução do sistema linear. Aqui, um método iterativo baseado em minimização de funcionais será utilizado com uma técnica de armazenamento em lista.



(a) Numeração da malha através da CHM

(b) Matriz de Rigidez

Figura 4.3: Efeitos da numeração através da CHM no caráter da matriz de rigidez.

O Algoritmo 2 mostra como realizar a ordenação da malha. Observe que a enumeração

inicial é a do passo anterior, isto é, quando elementos novos aparecem eles herdam um vértice do seu pai. O passo *verifica coordenadas dos vértices e enumera* analisa se os elementos tem níveis iguais, em caso afirmativo basta copiar a enumeração do vizinho. Caso contrário, é necessário verificar qual dos vértices é comum e, somente depois disso, enumerar.

Algoritmo 2 Algoritmo Ordenação(Malha)

```
1: enumeração inicial
2: para todo elemento da malha faça
3:    se vizinho é nodo preto então
4:       se número de Hilbert do vizinho é maior então
5:          verifica coordenadas dos vértices e enumera
6:       fim se
7:    senão
8:       procura vizinhos pretos
9:       se número de Hilbert do vizinho é maior então
10:         verifica coordenadas dos vértices e enumera
11:      fim se
12:   fim se
13: fim para
```

Observe que o passo *Procura vizinhos pretos* envolve duas situações. A célula atual pode estar em um nível maior, neste caso a busca no grafo envolve sempre um único caminho. Observe a Figura 3.10.b. Suponha que a célula atual seja a número 8 (elemento 8). Ao se analisar os vizinhos a leste, verifica-se a existência de um nodo branco (Figura 3.10.a). Como pode se verificar só existe um caminho a ser percorrido.

Caso a célula (elemento) esteja em um nível menor, a busca deve ser feita por dois caminhos diferentes. Deve-se buscar os nodos pretos nas extremidades, os quais representam os elementos que provavelmente têm o mesmo vértice do elemento atual. Novamente

analisando a Figura 3.10.b pode-se perceber que esse é o caso do elemento número 9. O nodo branco, vizinho ao norte do nodo preto representativo do elemento 9, apresenta duas conexões. A idéia é percorrer essas duas conexões até encontar um nodo preto que, caso já tenha tido seus vértices numerados, fornecerá a uma numeração do vértice da célula atual. Após encontrar tal nodo preto, o processo se reinicia pelo outro caminho. Assim, a célula atual tem seus vértices numerados.

4.2 Continuidade sobre a malha refinada

Um problema que pode ser observado é o aparecimento de vértices na interface entre elementos, isto é, cada vez que um elemento é refinado surgem novos elementos os quais por não estarem no mesmo nível de seu vizinho criam vértices na aresta deste, esse novos vértices, como mencionado anteriormente, são denominados vértices de restrição ou vértices restritos. Cabe salientar que funções de forma clássicas como as que são utilizadas aqui geram não conformidades entre os elementos, basta observar a Figura 4.4, onde a função de forma para o elemento A é parabólica enquanto para os elementos B e C ela é linear. Para garantir que a continuidade seja obrigatória é necessário que alguma técnica seja adotada [3, 10, 23, 25, 31].

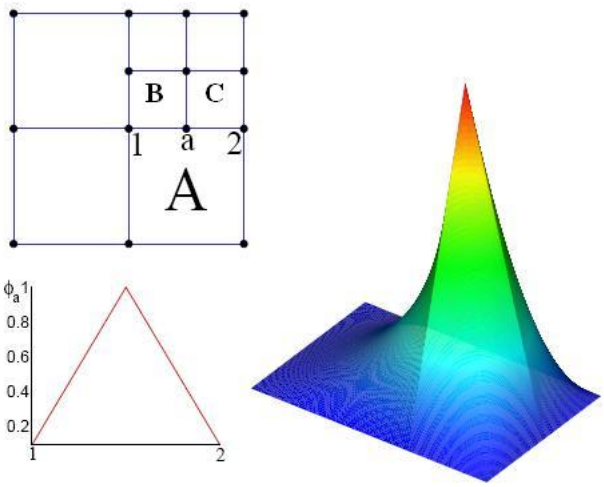


Figura 4.4: Função de forma sobre um nodo restrito.

Como apresentado anteriormente as estratégias de refinamento podem gerar malhas

não-conformes. Uma solução para esse problema foi proposta por [10]: Seja l_p o lado do elemento P, de grau de refinamento maior do que um que lhe é contíguo, G, em que ocorre $l_p \subset l_G$. Seja ψ_i uma função de forma do lado l_p do elemento P e Ψ_j uma função de forma do lado l_G do elemento G. Se ψ_p^i denota a restrição da função ψ_i ao lado l_p e Ψ_p^i denota a restrição da função Ψ_j ao lado l_p contido em l_G tem-se:

$$\psi_p^i = \psi_{i|l_p}; \quad \Psi_p^j = \Psi_{j|l_p}; \quad l_p \subset l_G \tag{4.1}$$

Para se obter a continuidade sobre o lado l_p iguala-se uma combinação das funções ψ_p^i com cada função Ψ_p^j. Considerando o espaço gerado pelas funções ψ_p^i pode-se escrever qualquer função Ψ_p^j definida sobre o lado $l_p \subset l_G$ como uma combinação linear das funções ψ_p^i, $1 \leq i \leq n$:

$$\Psi_p^j = \sum_{k=1}^{n} \beta_{jk}^p \psi_p^k; \quad 1 \leq j \leq n \tag{4.2}$$

Assim, para se obter a continuidade, basta calcular a projeção L^2 sobre os espaço das funções Ψ_p^j associadas à conectividade restrita do elemento P ao lado l_p:

para j = 1, 2, . . . , n:

$$\int_{l_p} \psi_p^k \psi_p^i dxdy = \int_{l_p} \Psi_p^i \psi_p^i dxdy, 1 \leq i, k \leq n \tag{4.3}$$

Seja u_h a solução por elementos finitos. Seja o j-ésimo coeficiente da solução do MEF sobre o elemento *el* denotado por α_j^{el}. Então, os coeficiente multiplicadores das funções do lado l_p do elemento P, são obtidos pela equação:

$$\alpha_k^P = \sum_{j=1}^{n} \alpha_j^G \beta_{jk}^p \ , \tag{4.4}$$

onde os valores β_{jk}^p são obtidos da solução do sistema acima. Assim, u_h é contínua sobre o lado l_p e

$$u_{h|lp} = \sum_{k=1}^{n} \sum_{j=1}^{n} \alpha_j^G \beta_{jk}^p \psi_p^k \tag{4.5}$$

Tal solução é um caso geral para elementos finitos de qualquer ordem de aproximação. Aqui foi utilizado um caso particular, que ocorre para elementos bilineares, assim pode-se observar pela Figura 4.5 as seguintes dependências entre os coeficientes da solução aproximada entre o lado do elemento A e os lados dos elementos B, C e H:

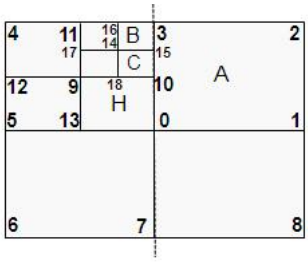


Figura 4.5: Representação de Lados Restritos.

Assim, pode ser aplicada uma interpolação dos valores da solução nesses vértices restritos:

$$\alpha_H^0 = \alpha_A^0 \tag{4.6}$$

$$\alpha_B^3 = \alpha_A^3 \tag{4.7}$$

$$\alpha_C^{10} = \alpha_H^{10} = \frac{1}{2}\alpha_A^0 + \frac{1}{2}\alpha_A^3 \tag{4.8}$$

$$\alpha_C^{15} = \alpha_B^{15} = \frac{1}{4}\alpha_A^0 + \frac{3}{4}\alpha_A^3 \tag{4.9}$$

4.3 Método dos Gradientes Conjugados

Proposto inicialmente por Hestenes e Stiefel para solução de sistemas lineares como método direto. É como método iterativo que o método dos Gradientes Conjugados (GC) tem tido destaque [1, 8, 32, 33].

Considere o sistema linear $Ax = b$, onde A é uma matriz simétrica e positiva-definida

de ordem nXn. O GC constrói duas sequências de vetores.

$$r_k = r_{k-1} - \alpha_k A p_k \tag{4.10}$$

$$p_k = r_{k-1} + \beta_k p_{k-1} \tag{4.11}$$

com $k = 1, 2, ...$

Inicialmente adota-se x_0 como uma estimativa arbitrária e $p_1 = r_0$. Esses vetores (Equação 4.10 e Equação 4.11) satisfazem as condições de ortogonalidade e conjugação [32].

Os escalares α_i e β_i são dados por:

$$\alpha_i = \left(\frac{r_{i-1}^t r_{i-1}}{r_i^t A r_i} \right) \tag{4.12}$$

$$\beta_i = \left(\frac{r_{i-1}^t r_{i-1}}{r_{i-2}^t A r_{i-2}} \right) \tag{4.13}$$

O GC gera uma sequência de vetores (Equação 4.14) que converge para a solução com velocidade dependente do condicionamento da matriz A [1, 32].

$$x_i = x_{i-1} + \alpha_i p^i \tag{4.14}$$

com $i = 1, 2, 3, ...$

O Algoritmo 4.3 implementa o GC de acordo com a proposta de Hestenes e Stiefel. Observe que nessa implementação somente quatro vetores precisam ser armazenados: x, r, p e Ap.

A exigência de continuidade entre os lados de elementos vizinhos é realizada pela maioria dos códigos de elementos finitos através de uma modificação na matriz de rigidez que passa a incorpora as atualizações de continuidade. Neste trabalho é utilizada uma abordagem diferente. A exigência de continuidade é incorporada na resolução do sistema

linear. Assim, no método dos gradientes conjugados é realizada uma pequena modificação, acrescenta-se após linha de atualização da aproximação (linha 13) o trecho do Algoritmo 4.3. Resultados numéricos mostram a eficácia de tal modificação.

Algoritmo 3 Algoritmo do Método dos Gradientes Conjugados

1: estimativa inicial x_0
2: cálculo do resíduo inicial $r_0 = b - Ax_0$
3: $k = 1$
4: $p_1 = r_0$
5: cálculo da direção da busca $\alpha_1 = \frac{(r_0^t r_0)}{p_1^t A p_1}$
6: próximo valor da solução $x_1 = x_0 + \alpha_1 p^1$
7: novo valor do resíduo $r_1 = r_0 - \alpha_1 A p^1$
8: **enquanto** resíduo $r_k \neq 0$ **faça**
9: número de iterações $k = k + 1$
10: $\beta_k = \left(\frac{r_{k-1}^t r_{k-1}}{r_{k-2}^t A r_{k-2}}\right)$
11: $p_k = r_{k-1} + \beta_k p_{k-1}$
12: cálculo da direção de busca $\alpha_k = \left(\frac{r_{k-1}^t r_{k-1}}{r_k^t A r_k}\right)$
13: atualização da aproximação $x_k = x_{k-1} + \alpha_k p^k$
14: atualização do resíduo $r_k = r_{k-1} - \alpha_k A p^k$
15: **fim enquanto**
16: solução $x = x_k$

O vetor de duas posições, γ, no Algoritmo 4.3 armazena as contribuições dos vizinhos no vértice restrito calculadas através da interpolação linear dos resultados. Ele é obtido ao se realizar a ordenação da malha pela CHM.

Algoritmo 4 Atualização da solução nos vértices restritos

para todos os vértices **faça**

se vértice (v_i) é restrito **então**

$x_{k_i} = \gamma_0 * x_{k_j} + \gamma_1 * x_{k_z}$

fim se

fim para

4.4 Programação Orientada a Objetos

Tornou-se crescente o interesse em técnicas orientadas a objetos para análise de elementos finitos desde o início da década de 90. Foi mostrado que o uso de tal paradigma requer menos tempo, códigos fontes menores e melhor gerenciamento em relação a linguagens procedurais [5]. São inúmeros os programas desenvolvidos para análise de elementos desde então, podem-se destacar o PZ [5, 10], o Pyramid [34], dentre outros. Algumas das vantagens de utilizar técnicas orientadas a objetos são [10]:

- Reutilização de código;
- Alto nível de abstração;
- Fácil manutenção e extensão do programa;

Nesse trabalho foi escolhida a linguagem de programação orientada a objetos C++ devido a similaridade com linguagem C, além do fato, da reutilização de código ter permitido que algumas estruturas do ALG proposto em [1] serem utilizadas também para o caso de elementos finitos.

4.4.1 Estrutura de Classes e Métodos

A estrutura do sistema computacional aqui utilizada é idêntica da maioria dos programas de elementos finitos, provendo uma maior familiaridade para pesquisadores em

elementos finitos. Assim o sistema pode ser dividido em três módulos distintos: pré-processador, processador e pós-processador. Como dito anteriormente seu diferencial está no fato da utilização de OO e da estrutura de dados utilizada (ALG). Apesar de poder ser dividido em três módulos, estes estão encapsulados dentro de uma estrutura de classes. O FEM-ALG é composto por cinco classes: Grid, Cell, Element, Node e ReferenceEl (Figura 4.6).

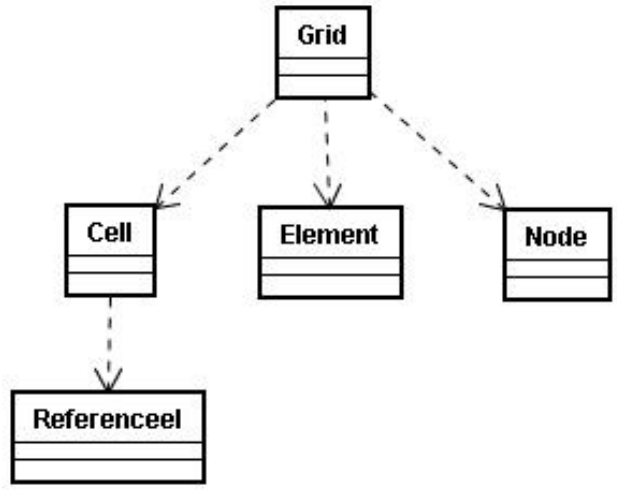


Figura 4.6: Diagrama de Classes do FEM-ALG.

4.4.1.1 Classe Grid

Esta é a principal classe da implementação do FEM-ALG. Nela são controlados a estrutura de dados do ALG, a ordenação da malha, o algoritmo de refinamento adaptativo e os cálculos dos coeficientes da matriz de rigidez. Alguns métodos dessa classe:

- SetDiscretizationMatrix: responsável pelo cálculo dos coeficientes da matriz de rigidez utilizando os métodos da classe Cell e armazenando os resultados em objetos instanciados pela classe Element;
- ErrorAnalysisElementar(Cell *gridCell, double *energyNorm): implementa os cálculos de erros apresentandos no Capítulo 2 Seção 2.4 nas normas L_2 e norma energia (semi-norma H_1);
- RefineGrid(int refinementLevel, double refinementBound, bool drawGrid): verifica se é necessário realizar algum refinamento entre os elementos utilizando o método

ErrorAnalysisElementar;

- InitializeGrid(bool drawGrid): inicializa a malha do problema com quatro elementos finitos utilizando a estrutura do ALG.

- OrderNodesGridsCells(Cell *gridCell, Cell *neighborGridCell,char direction): analisa vizinho do elemento para atribuir valores aos seus vértices obtendo assim uma enumeração na malha.

4.4.1.2 Classe Cell

Essa classe é responsável por implementar os nodos do ALG, armazena as informações referentes a malha geométrica, vizinhos, próximo elemento na curva de Hilbert, etc. Realiza cálculos para a construção das matrizes elementares. Seus principais métodos são:

- LoadElementarMatrix(Cell *gridCell, double **ae, double *be): realiza os cálculos dos coeficientes das matrizes elementares;

- LTransf2d(double **DPSI, double &DETJ , double **DSDX, double *coordx, double *coordy): realiza o mapeamento dos valores oriundos do elemento de referência e os elementos da malha. Utiliza para isso os métodos da classe ReferenceEl (Jacobian, Shape2d e IntegrationRule).

4.4.1.3 Classe Referenceel

Responsável por implementar o elemento de referência. Neste caso, o quadrado biunitário, isto é, $\Omega = [-1, 1]^2$. Basicamente consiste nos métodos responsáveis pela integração numérica e funções de forma.

- IntegrationRule(double **xint, double *wint, int nint): implementa a quadratura gaussiana com 16 pontos de integração;

- Jacobian(): retorna o valor do jacobiano da transformação ou a informação que o jacobiano é nulo;

- Shape2d(double **xint, double *PSI, double **DPSI, int i): calcula as funções de forma do Elemento e suas respectivas derivadas de acordo com o Capítulo 2.

4.4.1.4 Classe Node

Implementa uma lista de vértices da malha. Todos os objetos dessa classe contêm os atributos referentes as coordenadas geométricas do vértice, classificação do tipo (contorno, restrito ou interno), os elementos aos quais pertencem, valor da solução, situação (ativo ou inativo), além de atributos que serão utilizados pelo método dos gradientes conjugados.

- RemoverNode(int id): remove um determinado vértice, que pode ter desaparecido devido ao desrefinamento;

- ApplyBoundaryCondition(): aplica condições de contorno, modifica a lista que armazena os valores dos coeficiente da matriz de rigidez;

- InitializeDiagonalElementsofNodes(): calcula coeficientes da diagonal principal da matriz de rigidez e os armazena na primeira posição da lista encadeada;

- ConjugateGradientNodes(): implementa o método dos gradientes conjugados para a lista encadeada criada com a modificação para o caso de vértices restritos;

- FillDiscretizationGlobalMatrix(Cell *gridCell, double **ae, double *be): cálcula os coeficientes da matriz de rigidez global através da contribuição de cada elemento da malha;

- InterpListaNodes(int id, int coefele0, int coefele1, char direction): encontra os valores dos coeficientes da interpolação linear que serão utilizados para modificar os valores da solução encontrada no GC;

- ActionNode(int id, bool situation): verifica se o vértice já foi criado na lista.

4.4.1.5 Classe Elements

Implementa os elementos da matriz de rigidez. Uma lista contendo a linha da matriz de rigidez referente a cada incógnita da malha. Cada objeto Node criado pela Classe Node contém uma lista de elementos da Classe Elements. Essa classe não possui métodos, sendo formada pelas seguintes variáveis:

- value: armazena os coeficientes da matriz de rigidez;
- column: indica qual coluna da matriz de rigidez está sendo representada por aquele elemento;
- next: próximo elemento de mesma classe, representando algum outro elemento daquela linha;
- cell: indica para cada elemento qual o vértice que ele se refere. Garante que no método dos gradientes conjugados o elemento seja multiplicado pelo valor encontrado para a variável referente aquela coluna.

Capítulo 5

Aspectos Computacionais

Neste capítulo, serão apresentandos os experimentos computacionais para analisar a implementação FEM-ALG. O FEM-ALG foi implementado em C++, utilizando a *Standard Template Library* (STL) e compilado com g++ 3.4.4(mingw especial). A simulação foi realizada em um DELL, com processador CoreDuo de $1.66GHz$ e $512Mbytes$ de memória RAM, usando o sistema operacional Windows XP, versão $2002-SP2$. Para validação utilizou-se a equação de Laplace em dois casos testes. No primeiro foram realizados refinamentos uniformes e mostra-se a convergência da solução numérica a medida que a malha é mais refinada. No segundo caso, utilizou-se a equação de Laplace mantendo sobre controle a quantidade de refinamentos, buscou-se assim manter um número reduzido de vértices restritos para se estudar os efeitos da técnica de restrição de vértices apresentada no capítulo anterior. Para os resultados numéricos foram utilizados diferentes problemas de Poisson com presença de gradientes no domínio. Ressalta-se a importância da equação de Poisson devido sua área de aplicação tanto em eletrostática quanto condução térmica em estados estacionários [32].

5.1 Validação do Método

5.1.1 Caso Teste 1

A validação do método é feita utilizando o problema de Laplace:

$$\nabla u = 0 \Omega = (0,1)^2 \tag{5.1}$$

com $u = g(x) = x_1 + x_2$ sobre $\partial\Omega$.

A solução exata desse problema é $u(x) = x_1 + x_2$. Para que todos elementos da malha fossem refinados adotou-se que $\eta_{max} = 0.0$. O objetivo era verificar a acurácia dos valores encontrados pelo MEF. A malha apresentada na Figura 5.1

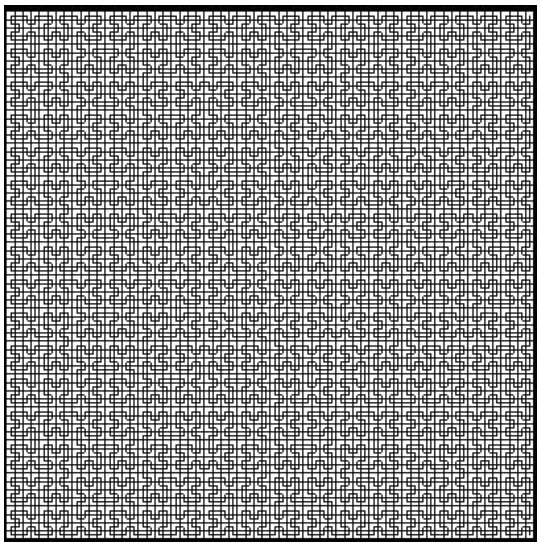


Figura 5.1: Malha do Problema de Laplace com refinamento uniforme contendo 1089 vértices.

A Tabela 5.1 mostra uma comparação entre os erros na norma energia para diferente malhas. Pode-se perceber uma melhor aproximação a medida que aumentam os números de elementos na malha.

Malha	Número de Elementos	Número de Vértices	Erro na Norma Energia
a	16	25	$1.7X10^{-17}$
b	256	289	$4.4X10^{-18}$
c	1024	1089	$2.2X10^{-18}$

Tabela 5.1: Resultados Problema de Laplace.

5.1.2 Caso Teste 2

Abordou-se o mesmo problema de Laplace apresentado no caso teste anterior. Entretanto, admitiu-se um erro relativo máximo de 0.05. A Figura 5.2 mostra o comportamento da malha pra o problema. Nesse caso utilizou-se a regra (2:1) para controlar a quantidade de refinamentos.

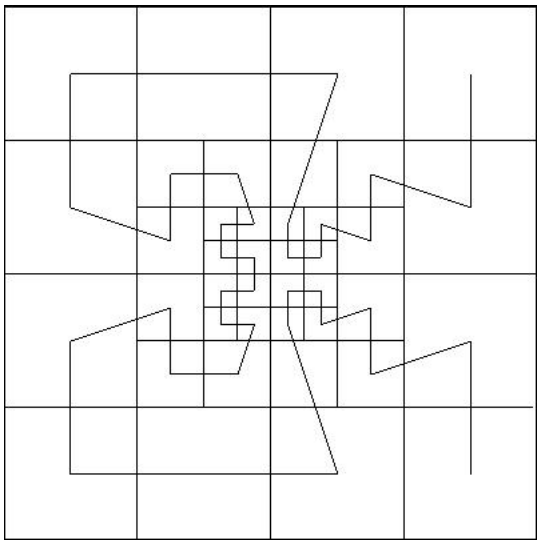


Figura 5.2: Malha do Problema de Laplace com 41 vértices e CHM associada.

A Figura 5.3 mostra o gráfico da quantidade de vértices pela energia do erro na norma energia (semi-norma H_1) mostrando a convergência da solução.

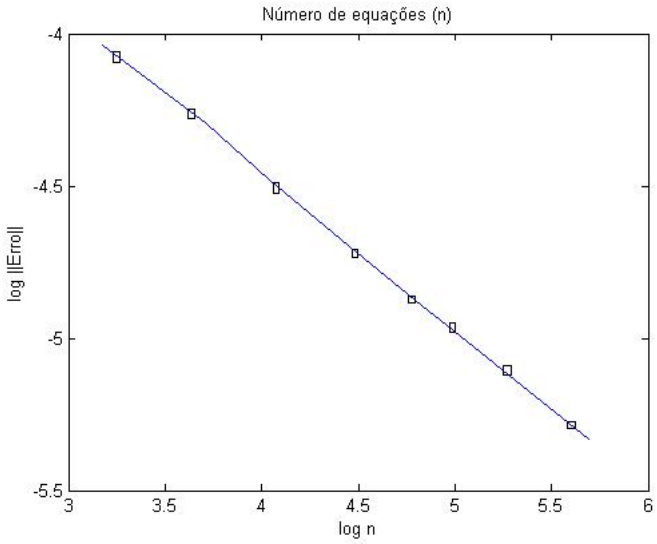


Figura 5.3: Taxa de convergência do refinamento h para o problema de Laplace.

5.2 Testes Computacionais

5.2.1 Problema de Poisson 1

Considere o problema de Poisson em $\Omega = (0,1)^2$ definido pela Equação 5.2 com condição de contorno de Dirichlet.

$$-\nabla^2 u = f(x,y) \tag{5.2}$$

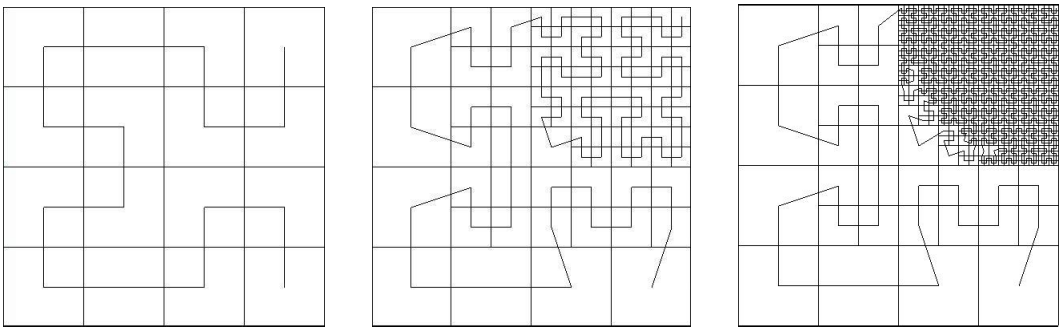
com condição $u = 0$ sobre $\partial\Omega$.

O termo fonte é dado pela Equação 5.3.

$$f(x,y) = -90x^8y^{10}(1-x)(1-y) + 20x^9y^{10}(1-y) - 90x^{10}y^8(1-x)(1-y) + 20y^9x^{10}(1-x) \tag{5.3}$$

A solução analítica desse problema é dada por $u(x) = x^{10}y^{10}(1-x)(1-y)$

Na Figura 5.2.1 a malha inicial e seu processo de refinamentos são demonstrados.



(a) (b) (c)

Figura 5.4: Refinamentos sucessivos da malha do problema de Poisson 1. (a) Malha com 4 elementos; (b) Malha com 43 elementos; (c) Malha com 248 elementos.

Pode-se observar que o FEM-ALG conseguiu detectar a região que apresenta o gradiente próximo ao ponto $(1,1)$.

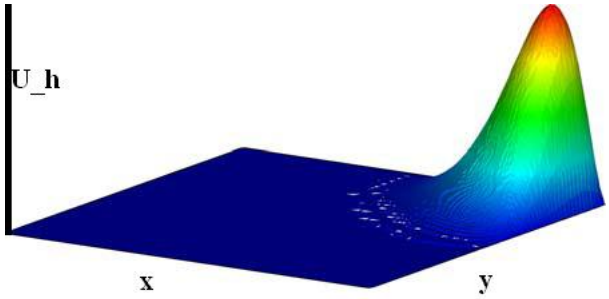


Figura 5.5: Comportamento da solução u_h para o problema de Poisson 1.

5.2.2 Problema de Poisson 2

Considerando o problema definido pelas Equações 5.2 e $u = 0$ sobre todo o contorno($\partial\Omega$). Agora o termo fonte é dado pela Equação 5.4

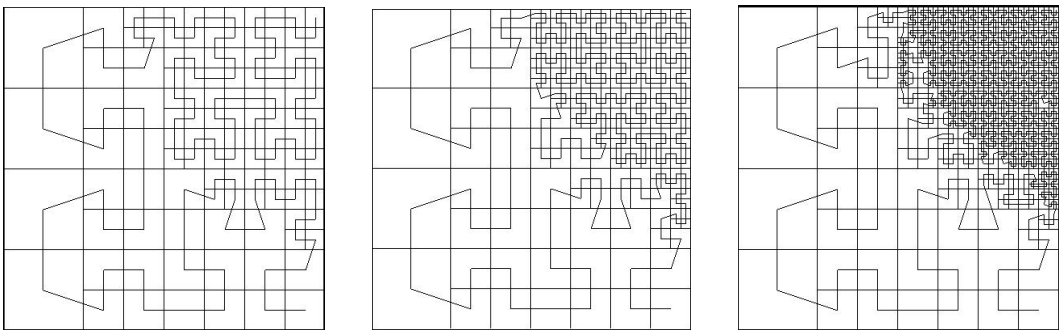
$$\begin{aligned} f(x,y) &= 10(1-x)^2(e^{10x^2}-1) - 40x(1-x)(e^{10x^2}-1) \\ &+10x^2(e^{10x^2}-1) - 400x^3(1-x)e^{10x^2} + 500x^2(1-x)^2e^{10x^2} \\ &+2000x^4(1-x)^2e^{10x^2} + 2(1-y)^2(e^{10y^2}-1) - 8y(1-y)(e^{10y^2}-1) \\ &+100y^2(1-y)^2e^{10y^2} + 2y^2(e^{10y^2}-1) - 80y^3(1-y)e^{10y^2} + 400y^4(1-y)^2e^{10y^2} \end{aligned} \tag{5.4}$$

A Equação 5.5 é a solução analítica desse problema.

A Figura 5.2.2 apresenta as malhas para este problema. Observe que o refinamento é maior próximo a região com presença de gradiente.

$$u(x) = 5x^2(1-x)^2(e^{10x^2}-1)y^2(1-y)^2(e^{10y^2}-1). \tag{5.5}$$

Assim como no problema anterior o FEM-ALG é capaz identificar a presença do e aproximar de maneira precisa.



(a) (b) (c)

Figura 5.6: Refinamentos sucessivos da malha do problema de Poisson 2. (a) Malha com 52 elementos; (b) Malha com 112 elementos; (c) Malha com 335 elementos.

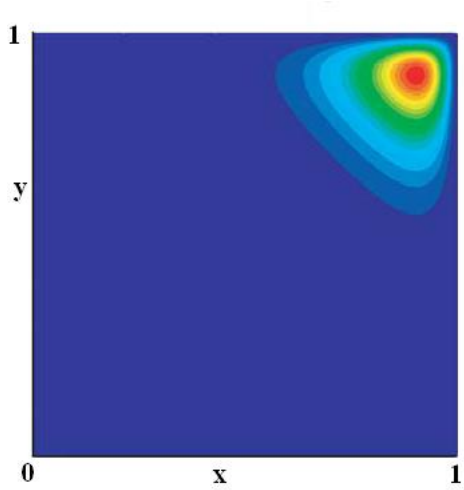


Figura 5.7: Comportamento da solução u_h para o problema de Poisson 2.

Capítulo 6

Conclusão

Neste trabalho foi desenvolvido um aplicativo que integra uma estrutura em grafos (ALG) ao Método dos Elementos Finitos na geração de um método adaptativo para solução de problemas de equações diferenciais, denominado FEM-ALG.

O FEM-ALG é um aplicativo que mantêm a estrutura original de programas de elementos finitos facilitando pesquisas na área. Como se sabe, em métodos numéricos a eficiência de processamento é fundamental. Os resultados dos testes efetuados com o aplicativo mostram que o mesmo é capaz de detectar regiões com presença de gradientes e realiza refinamentos locais com acurácia.

O FEM -ALG é resultado de um estudo detalhado sobre o método dos elementos finitos e malhas adaptativas, focando o melhor tipo de estruturas de dados para representá-las. Além disso, o mesmo implementa uma solução para ordenação dos vértices da malha de elementos finitos através da curva de Hilbert Modificada.

O aparecimento de vértices restritos é solucionado através do cálculo da contribuição dos vértices do lado contíguo a ele, empregando uma interpolação linear. Tal interpolação é incorporada ao método dos Gradientes Conjugados na obtenção da solução sobre os vértices restritos.

Este trabalho apresenta também o diagrama de classes do aplicativo, onde pode ser percebida a simplicidade da estrutura desenvolvida. Esse fato possibilita a integração de

novos módulos, tanto para integração numérica quanto a incorporação de elementos com maiores graus de liberdade.

6.1 Trabalhos Futuros

Uma extensão deste trabalho é a utilização de elementos isoparamétricos para representar domínios de contorno irregulares, além da análise de problemas com singularidade. Dessa maneira, seria possível analisar os efeitos da interpolação quando mais refinamentos são necessários.

Em [23] são propostas funções de forma baseadas em métodos sem malha. Tais funções apresentam alta acurácia quando empregadas a estrutura da *quadtree* no MEF. Torna-se necessário avaliar os efeitos de tais funções sobre a estrutura do ALG.

Burgarelli [6] e Robaina [7] mostram que o ALG apresenta granularidade fina, que sugere sua paralelização. Portanto o mesmo estudo deve ser realizado para o caso do FEM-ALG observando a questão da não conformidade da malha.

Por fim, uma abordagem em malhas triangulares. Esse tipo de malha é interessante para contornos irregulares. Entretanto, o refinamento da malha gera problemas de condicionamento com a diminuição dos ângulos internos dos triângulos [29]. A estrutura de refinamento local do ALG diminuiria os efeitos da redução dos ângulos devido ao refinamento não ser propagado por toda a malha. Portanto, uma implementação de elementos finitos com elementos triangulares utilizando o ALG seria necessária a fim de verificar sua eficiência.

Referências

[1] BURGARELLI, D.; KISCHINHEVSKY, M.; BIEZUNER, R. A new adaptive mesh refinement strategy for numerically solving evolutionary pde´s. *Journal of Computational and Applied Mathematics*, v. 196, p. 115–131, 2006.

[2] BABUSKA, I.; RHEINBOLDT, W. Error estimates for adaptive finite elements computations. *Jounal on Numerical Analysis*, v. 15, n. 4, p. 736–754, Agosto 1978.

[3] RHEINBOLDT, W.; MESZTENYI, C. On a data structure for adaptive finite element mesh refinements. *ACM Transactions on Mathematical Software*, v. 6, n. 2, p. 166–187, Junho 1980.

[4] DEMKOWICS, L.; DEVLOO, P.; ODEN, J. On an h-type mesh refinement strategy base on minimization of interpolation errors. *Comput. Methods Appl. Mech. Engrg.*, v. 35, p. 67–90, 1985.

[5] DEVLOO, P. *An h-p Adaptive Finite Element Method for Stead Compressible Flow.* Tese (Doutorado) — University of Texas at Austin, 1987.

[6] BURGARELLI, D. *Modelagem Computacional e Simulação Numérica Adaptativa de Equações Diferenciais Parciais Evolutivas Aplicadas a um Problema Termoacústico*. Tese — Pontifícia Universidade Católica do Rio de Janeiro, 1998.

[7] ROBAINA, D. *Um visualizador para navegação através de cenários tridimensionais baseado em malhas adaptativas.* Dissertação (Mestrado) — Universidade Federal Fluminense, 2006.

[8] GONZAGA, S.; GUEDES, M.; KISCHINHEVSKY, M. Método dos volumes finitos com refinamento adaptativo de malhas aplicado ao problema da camada limite. *Anais do XXVIII CILAMCE - Congresso Ibero Latino-Americano sobre Métodos Computacionais em Engenharia*, 2007. Porto.

[9] CHEN, Z. *Numerical solution of partial differential equations by the finite element method*. 1st. ed. Nova York: Springer, 2005.

[10] BRAVO, C. *Um sistema h-p adaptativo para o método dos elementos finitos através de funções hierárquicas.* Tese (Doutorado) — Universidade Estadual de Campinas, 2001.

[11] FELIPPA, C. *Introduction to Finite Element Methods*. University of Colorado, 2005. Disponível em: <http://www.colorado.edu/engineering/CAS/courses.d/IFEM.d/>.

[12] COURANT, R. Variational methods for the solution of problems of equilibrium and vibration. *Bull. Amer. Math. Soc.*, v. 49, p. 1–23, 1943.

[13] ARGYRIS, J. Energy theorems and structural analysis, part i: General theory. *Aircraft Engin.*, v. 26, p. 347–356, 1954.

[14] CLOUGH, P. *The Finite Element Method in Plane Stress Analysis.* 2nd. ed. Pittsburgh, PA: In Proc. of the Second ASCE Conference on Electronic Computation, 1960.

[15] CIARLET, P. *The Finite Element Method for Elliptic Problems.* 2nd. ed. Amsterdam: North-Holland, 1978.

[16] BABUSKA, I. Numerical stability in problems of linear algebra. *Jounal on Numerical Analysis*, v. 9, n. 1, p. 53–77, Março 1972.

[17] HUGHES, T. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis.* 2nd. ed. Englewood Cliffs, New Jersey: Prentice-Hall, 1987.

[18] BECKER, E.; CAREY, G.; ODEN, J. *Finite Elements: An Introduction. Volume 1.* 2nd. ed. New Jersey: Prentice-Hall, Englewood Cliffs, 1960.

[19] CHONGBIN, Z.; STEVENS, G. P. A practical error estimator for finite element predicted natural frequencies of membrane vibration problems. *Journal of Sound and Vibration*, v. 195, n. 5, p. 739–756, Junho 1996.

[20] CREUSEA, E.; NICAISEA, S. A posteriori error term estimations of a coupled mixed and standard galerkin method for second order operators. *J. of Comp. and Ap. Mathematics*, v. 213, n. 1, p. 35–55, Janeiro 2008.

[21] CONTI, T. *Aplicação do Método da Expansão em Funções Hierárquicas na solução das Equações de Navier-Stokes em Duas Dimensões para Fluidos Compressíveis em Alta Velocidade.* Tese (Doutorado) — Universidade de São Paulo, 2006.

[22] DUARTE, H. *Estimador de Erro para a Formulação p do Método dos Elementos Finitos Aplicado ao Problema Fluido-Estrutura.* Tese (Doutorado) — Universidade Estadual de Campinas, 2003.

[23] TABARREI, A.; SUKUMAR, N. Adaptive computations on conforming quadtree meshes. *Finite Elements in Analysis and Design*, v. 41, p. 686–702, 2005.

[24] FERNANDO, C. *Um refinamento h-p adaptativo para o método dos elementos finitos.* Tese (Doutorado) — Universidade Federal do Rio de Janeiro, 2001.

[25] DEMKOWICS, L. 2d hp-adaptive finite element package - version 2.0. *TICAM*, v. 02, p. 1–66, 2006.

[26] HILBERT, D. Ueber stetige abbildung einer linie auf ein achenstuck. *Mathematisch Annalen*, p. 459–460, 1891.

[27] WIRTH, N. *Algorithms + data structure = programas.* 2nd. ed. Palo Alto: Prentice-Hall, 1985.

[28] CLAES, J. *Finite Element Methods and Their Applications.* 1st. ed. Suécia: Studentlitteratur, 1987. 278 p.

[29] GEORGE, P. *Automatic Mesh Generation: Application to Finite Element Methods.* 2nd. ed. Washington: John Willey & Sons Ltd., 1991.

[30] GROOMS, H. Algorithm for matrix bandwidth reduction. *Journal of Structural division*, v. 98, p. 203–214, 1972.

[31] TABARREI, A.; SUKUMAR, N. Conforming polygonal finite elements. *International Journal for Numerical Methods in Engineering*, v. 61, p. 2045–2066, Outubro 2004.

[32] BURDEN, R.; FAIRES, J. *Numerical Analysis.* 8th. ed. Youngstown: Thomson, 2004.

[33] BOLZ, J.; FARMER, I.; GRISPUN, E. Sparse matrix solvers on the gpu: conjugate gradients and multigrid. *ACM Transactions on Graphics*, v. 22, p. 917–924, 2003.

[34] NORTON, C. D.; LOU, J. Z.; CWIK, T. A. Finite element mesh adaptation with pyramid. 2*nd Annual JPL IT Symposium*, n. 5, Novembro 2002.