RICARDO ALEXANDRE REINALDO DE MORAES

UMA IMPLEMENTAÇÃO DO MÉTODO DE ELEMENTOS FINITOS EM GEOMETRIAS NÃO-CONVEXAS

Florianópolis - SC

1999

UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Ricardo Alexandre Reinaldo de Moraes

UMA IMPLEMENTAÇÃO DO MÉTODO DE ELEMENTOS FINITOS EM GEOMETRIAS NÃO-CONVEXAS

Prof. Dr. Daniel Santana de Freitas - Orientador
Prof. Dr. Sérgio Peters - Co-Orientador

Florianópolis, dezembro 1999.

UMA IMPLEMENTAÇÃO DO MÉTODO DE ELEMENTOS FINITOS EM GEOMETRIAS NÃO-CONVEXAS

RICARDO ALEXANDRE REINALDO DE MORAES

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação Área de Concentração (Sistemas de Computação) e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

Prof. Daniel Santana de Freitas, Dr.
Orientador

Prof. Fernando A. Ostuni Gauthier, Dr.
Coordenador

Banca Examinadora:

Prof. Sérgio Pëters, Dr.

Jáuber Cavalcante de Oliveira, Dr.

Luiz Paulo Luna de Oliveira, Dr.

Ricardo Felipe Custódio, Dr.

À minha mãe,
daquele que a ama muito.

Agradecimentos

A Deus, pela força em todos os momentos de incertezas.

À minha família, por todo o apoio durante à minha vida e principalmente nestes dois anos, em especial à minha mãe.

Aos meus orientadores, professor Daniel Santana de Freitas e Sérgio Peters, pela compreensão, pela amizade, pela ajuda sempre constante no desenvolvimento deste trabalho, e pelo seus esforços para que se concretizasse mais um sonho meu.

Aos grandes amigos que fiz aqui em Florianópolis, em especial à Adalto, Alex, Gill, Marie, Nelson, Taís e Sâmela. Valeu, vocês foram e são muito importantes para mim.

Aos meus grandes amigos "lageanos": Alexandre, Beto, Cheila, Dayse, Fabiana, Jussara e Simone.

Aos meus colegas de trabalho na UDESC: Álvaro, Gilberto, Ivânia, Julíbio, Jussara, Marli, Maurício, Oyara, Renata, Rosângela, enfim, a todos aqueles que me apoiaram nesta jornada.

Aos bolsistas do LABCAS, em especial à Andrea e Antônio.

Aos professores do CPGCC, em especial ao professor Jáuber por sua imensa sabedoria e paciência, e ao professor Ricardo Custódio pelo seu apoio e ótimos conselhos recebidos desde o início do mestrado.

À Verinha e Valdete pela simpatia, alegria e competência.

Aos membros da banca examinadora.

À UDESC e à CAPES pelo apoio financeiro.

Resumo

Neste trabalho é utilizado o Método dos Elementos Finitos para a implementação de um código computacional capaz de resolver problemas elípticos, com ênfase na equação de Poisson. A formulação adotada abrange domínios convexos ou não-convexos, sujeitos a qualquer combinação de condições de contorno de Neumann e de Dirichlet e com qualquer configuração para as propriedades físicas e para o carregamento.

Diversos aspectos fundamentais da Análise Numérica aplicada à resolução de EDPs são abordados, tais como a discretização por Elementos Finitos, a resolução de sistemas lineares do tipo banda e a integração numérica por Gauss Legendre, além da análise de erros. Os procedimentos numéricos efetivamente implementados incluem a geração da malha computacional por triangulação de Delaunay, a discretização para elementos triangulares lineares e quadráticos, a montagem do sistema de equações algébricas e respectiva resolução por eliminação gaussiana para sistemas banda, a busca de ordenamento ótimo dos pontos nodais pelo algoritmo de Cuthill-McKee e o cálculo de estimativas de erros *a priori*.

O desempenho e a correção do programa obtido são testados por comparação com resultados apresentados na literatura para a aplicação de outros métodos numéricos de solução de EDPs a problemas conhecidos.

Abstract

This work presents an application of the Finite Element Method to the solution of elliptic boundary value problems with emphasis on Poisson's equation. The problem formulation encompasses 2D convex or non-convex domains, any combination of Neumann and Dirichlet boundary conditions and any functional dependence of the equation data on the independent variables.

Several fundamental aspects of Numerical Analysis applied to the resolution of PDEs are discussed, such as finite element discretization, resolution of banded linear systems, Gauss-Legendre numerical integration and error analysis. Numerical implementation includes mesh generation by Delaunay triangulation, discretization with linear and quadratic elements, assembly of the system of linear algebraic equations and its resolution by Gaussian elimination to banded systems, search for optimal ordering of nodal points using Cuthill-McKee ordering algorithm and the calculation of *a priori* error estimates.

The performance and correctness of the developed code is tested against well-known results from the literature.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1 - Introdução

Diversos problemas, tais como o movimento dos planetas, a catenária (formato de uma corda pendente presa nas extremidades), o estudo da oscilação do pêndulo e outros, foram estudados empiricamente por J. Kepler (1571-1630) e L. da Vinci (1452-1519), pois faltava a eles a fundamentação teórica para modelar tais fenômenos. Com o aparecimento do cálculo diferencial no final do século XVII por obra de I. Newton e G. W. Leibnitz, inúmeros problemas, incluindo os citados acima, puderam ser modelados matematicamente por Equações Diferenciais (EDs). Normalmente um problema real não pode ser representado de maneira exata, em toda a sua complexidade, por uma equação matemática ou um sistema de equações. No entanto, quando se trabalha com as variáveis essenciais do fenômeno observado, o modelo que simula tal situação produz soluções bastante próximas das observadas na realidade.

Atualmente, um vasto número de problemas físicos de interesse científico e tecnológico é modelado por Equações Diferenciais Parciais (EDPs) e neste último século teve-se um grande avanço tecnológico decorrente do desenvolvimento da matemática aplicada à solução de problemas reais. Com o auxílio do computador, tem-se investido em métodos numéricos para construção de soluções aproximadas para estes problemas. Exemplos de aplicações do método podem ser encontrados, por exemplo, no projeto aerodinâmico e estrutural de um carro ou de um avião, simulações de condução de calor, mecânica dos fluidos, e outras. Dentre os principais métodos numéricos estudados, destacam-se os seguintes:

- Diferenças Finitas (MDF);

- Volumes Finitos (MVF);

- Elementos Finitos (MEF)

Estes métodos são derivados do mesmo princípio (método dos resíduos ponderados) e diferem na forma de minimização escolhida. Porém, existem métodos que se adaptam melhor a determinado problema do que outros.

Quando se aplica um método numérico a um problema real descrito por uma ED, é necessário executar alguns passos preliminares, como por exemplo, gerar uma malha.

A geração da malha consiste em dividir o domínio em um número finito de pontos, sobre os quais a ED é integrada, sendo que somente para estes pontos uma solução (aproximada) é obtida. Quanto maior o número de divisões (mais "refinada" a malha), mais próxima fica a solução numérica obtida da solução que seria a exata para o problema (se fosse possível resolvê-lo analiticamente). No MVF são geradas equações algébricas aproximadas, através de balanço das grandezas físicas em nível de volumes elementares através de balanços elementares, obtidos com a integração simples das equações governantes sobre estes volumes. No MDF a aproximação das derivadas envolvidas na ED é feita em termos de variáveis discretas.

O MEF começou a ser amplamente utilizado nos anos 70; até então o método mais difundido era o de Diferenças Finitas. Conforme Braess (1997) apud Oliveira (1998), o método dos elementos finitos é uma das principais ferramentas no tratamento numérico de EDPs elípticas e parabólicas. Este método possui flexibilidade superior aos métodos das diferenças finitas e volumes finitos, pois baseia-se na formulação variacional da equação diferencial. Esta característica variacional permite que seja utilizado em problemas mais complexos. A aplicação do MEF consiste em dividir o domínio de solução (Ω) em elementos com a forma de triângulos, hexaedros, etc., com pontos convenientemente distribuídos chamados de nós, e transformar a ED em um conjunto de equações algébricas estabelecendo a ligação entre os nós. As variáveis da ED são estabelecidas em cada ponto nodal, representada ao longo do domínio, como uma combinação linear de funções de interpolação. Usando técnicas variacionais, ou o método dos resíduos ponderados, a ED é transformada localmente em um conjunto de equações algébricas que representam o comportamento discreto das grandezas físicas envolvidas em um domínio elementar. Estas equações locais são agrupadas, o que corresponde a juntar os diversos elementos até formar o domínio do problema, e passam a formar um sistema global de equações algébricas, onde são impostas as condições de contorno (CC.), que correspondem aos nós do domínio Ω onde são conhecidos os valores da variável da ED. Os valores nodais não conhecidos são determinados resolvendo o sistema de equações. Este método tem sido utilizado com grande sucesso na resolução de problemas nas áreas de Matemática, Física e Engenharia.

Muitas pesquisas foram realizadas simultaneamente em várias partes do mundo, e em diferentes direções, sendo que os métodos variacionais e de resíduos ponderados

foram aplicados à técnica numérica dos elementos finitos. Uma enorme variedade de elementos foi desenvolvida, incluindo elementos curvilíneos e a introdução do conceito de isoparametrização (mapeamento de um elemento mestre em cada um dos elementos da malha). No entanto, somente na década de 70 é que o método foi generalizado para solução de EDPs. Sua aplicação em problemas de estruturas não-lineares e dinâmicas foi amplamente desenvolvida; foi então, utilizado em outras áreas, como mecânica dos sólidos, dos fluidos, termodinâmica, etc.

A teoria matemática que fundamenta o MEF é bastante desenvolvida e o método vem sendo estudado há bastante tempo. Uma grande vantagem do método é que são conhecidos muitos resultados matemáticos que provam a convergência e a estabilidade do método. Segundo Gupta and Meek (1996), há cinco grupos de artigos que podem ser considerados no desenvolvimento do MEF, os quais foram escritos por: Courant (1942), Argyris (1954), Turner *et al.* (1956), Clough (1960) e Zienkiewicz and Cheung (1965). Gupta and Meek (1996) estudaram a fundo os referidos artigos com o objetivo de fazer um histórico do MEF. Eles examinaram as contribuições de cada um deles, levando em conta se as técnicas de discretização são demonstradas de modo que permitam a implementação de qualquer modelo prático, com geometrias complexas e carregamentos, se existem provas da convergência do método e por fim se no artigo são apresentadas rotinas de como os cálculos podem ser automatizados. Constataram o que segue:

Courant (1942) desenvolveu a idéia de minimização de um funcional usando aproximações lineares sobre sub-regiões do domínio, com os valores sendo especificados como pontos discretos, os quais na essência são os pontos nodais. Courant mostra a subdivisão da malha em 1, 2, 3, 5 e 9 aproximações pontuais para resolver a torção de St Venant num quadrado oco de 2 X 2. Ele inicia os estudos explorando o método de Rayleigh-Ritz, utilizando a função de interpolação $\phi = a\,(1 - x)$. Baseado na teoria de elasticidade, ele constatou que esta aproximação não era suficiente e então utilizou uma função com dois coeficientes desconhecidos $\phi = a\,(1 - x)[1 + \alpha\,(x - 3/4)\,y]$. A seguir, Courant trocou esta aproximação e introduziu a idéia de aproximação linear ϕ *sobre áreas triangulares do seu domínio.* Ele valida seus resultados por comparação com outros métodos gerais, como o de Diferenças Finitas, e conclui seus artigo ressaltando a grande vantagem da divisão em elementos triângulares, por permitir a

generalização das malhas e em conseqüência, a aplicação do método em domínios com quatro furos, por exemplo e, obviamente, a sua adaptação para qualquer tipo de domínio. Courant também menciona que mostraria em outros artigos a aplicação do método a problemas que envolvem derivadas de mais alta ordem (provavelmente a equação de Poisson), mas, aparentemente, este trabalho nunca foi publicado. Examinando-se cuidadosamente o seu trabalho, constata-se que ele usou um tipo de elemento finito indefinido no artigo, num procedimento de minimização de energia potencial, mas Courant não escreveu qualquer detalhe matemático sobre a sua aproximação localmente linear para as funções ϕ, e por este motivo não é atribuído a ele a origem do método.

Argyris (1954), em sua série de artigos intitulada "Energy Theorems and Structural Analysis", estabeleceu um significante marco na história da mecânica estrutural. Neste artigo ele desenvolve a teoria matricial de estruturas para elementos discretos e mostra que estes elementos são apenas um caso particular do caso geral (domínio como um todo). Esta inovação conduz à concepção de flexibilidade e rigidez, onde ele consegue dividir uma equação diferencial em duas partes, cada uma representando um fenômeno físico ("flexibility and stiffness"). Então, após uma reformulação das equações, ele aplica a teoria desenvolvendo funções de interpolação "serendipity" para os elementos. Ele determina a matriz de rigidez K_{ij} usando uma matriz 8 X 8, e demonstra como o método pode ser aplicado em outros casos. Ele ainda antecipa a convergência com o refinamento da malha, mas não prova a convergência explicitamente.

O trabalho pioneiro de Turner *et al.* (1956) inicia com a discussão do desmembramento da matriz de rigidez em coordenadas globais. Após uma discussão inicial sobre elementos retangulares, ele comenta que: "The triangle is not only simpler to handle than the rectangle but later it will be used as the basic "building block" for calculation stiffness matrices for plates of arbitrary shape" e então formula e utiliza as funções de interpolação lineares dos triângulos. Esta forma de encontrar estas funções são as utilizadas até hoje, sendo que o procedimento é análogo ao apresentado no capítulo 2 deste trabalho. Portanto, a origem dos elementos triangulares é atribuída a Turner *et al.*. Outro importante aspecto deste artigo é o estudo da convergência do método.

O trabalho de Clough (1960), decorrente da pesquisa realizada para a "Boeing

Company", analisa os coeficientes de flexibilidade relacionados a aspectos da estrutura da asa de um avião para análise dinâmica. Clough atribuiu a Turner os créditos da invenção dos elementos triangulares e chegou a auxiliar no seu trabalho, provando que para diversas geometrias conhecidas a resolução converge à solução analítica. Clough foi quem deu o nome ao método (MEF) e em um de seus artigos ele mostra a diferença entre uma análise contínua e um método matricial.

Finalmente, Zienkiewicz and Cheung (1965) explanaram as técnicas de minimização de um funcional referidas por Courant (1942) e desenvolveram aplicações não estruturais por meio de minimização da energia potencial total, desenvolvendo sistematicamente o método. Neste primeiro artigo, foi feita uma aproximação para os valores nodais em um domínio triangular, o qual obviamente, foi dividido em sub-regiões. Seguindo o artigo de Clough, esta aproximação de Zienkiewicz and Cheugh foi referenciada como o Método dos Elementos Finitos (MEF) e a partir daí foi aberto um imenso campo de análise de problemas pelo MEF.

Atualmente, pode-se observar que o MEF está sendo utilizado para resolver problemas variados e, na maioria dos casos se obtêm resultados satisfatórios. Harari e Hughes (1993) desenvolveram através do MEF uma metodologia geral para resolução de uma EDP elíptica de segunda ordem. Os autores afirmaram que o MEF possui uma excelente base matemática, sendo que a taxa de convergência pode ser determinada já na fase de projeto, podendo-se obter a solução exata do problema com o refinamento da malha. Eles citaram também algumas vantagens do método, como por exemplo super convergência em modelos unidimensionais. Smolinski (1990) resolveu pelo MEF problemas de difusão não estacionária. Ele utilizou integração explícita no tempo em cada nó da malha, ou para um grupo de nós da malha atualizados em cada passo de tempo. O trabalho é aplicável em malhas de EF de qualquer dimensão e compostas de elementos arbitrários. Utilizando a integração explícita o autor conseguiu que pouca memória de computador fosse requerida. Na formulação não foi assumido o tamanho do elemento, a geometria, o tipo ou a propriedade do material. Assim sendo a estabilidade do algoritmo foi garantida para problemas lineares. Daichao, Kennet and Sven (1992) resolveram um sistema onde se tem uma mudança no modo de transmissão de calor (problema de convecção para um problema de condução). Utilizando o conceito de entalpia, foi feita uma reformulação da ED para uma Equação Diferencial Parcial

Quase-Linear. Então, esta equação juntamente com as Condições de Contorno (CC.) iniciais, foi decomposta em dois conjuntos de equações, representando respectivamente, um problema de convecção e difusão. As equações foram resolvidas pelo MEF e os exemplos numéricos demonstram que o método produz bons resultados.

Outro aspecto muito pesquisado no MEF está relacionado a sistemas de reordenamento dos índices nodais da malha, com o objetivo de reduzir o tamanho da banda da matriz de rigidez global ligada ao sistema de equações algébricas a ser resolvido. Segundo Oden e Carey (1984), o algoritmo mais comumente empregado para a minimização do tamanho de banda é o algoritmo Cuthill-Mckee (CM). O algoritmo CM, baseado na teoria de grafos, é considerado como um método de reordenamento direto, já que não é requerido nenhum prévio ordenamento dos vértices, com a vantagem de que o novo ordenamento nunca conduzirá a um tamanho de banda maior do que o original. Derivado do algoritmo CM, tem-se o Reverse Cuthill-Mckee (RCM), que conduz a resultados melhores que o CM em determinados casos, como por exemplo, na modelagem discreta de redes elétricas e fluxos em tubos. Liu and Sherman (1976) compararam o CM e o RCM aplicados a matrizes esparsas, simétricas e positivo-definidas e conseguiram provar que para os métodos de resolução que exploram a largura de banda, os dois são equivalentes. No entanto, em outros casos o RCM é realmente o que produz sempre melhores resultados. Scott and Han (1994) descreveram que os métodos de reordenamento existentes são os algoritmos de redução direta (os quais trabalham por troca de linhas e colunas da matriz), os algoritmos baseados na teoria de grafos e os algoritmos híbridos, os quais foram apresentados por Armstrong (1984). Sendo que nesta última modalidade, inicialmente são renumerados os nós fazendo trocas de linhas e colunas e depois é utilizada a teoria de grafos. Neste artigo eles elaboraram um novo algoritmo de reordenamento (híbrido), o qual é comparado com outros dois importantes algoritmos híbridos da literatura (GPS e NSAS). O algoritmo de Armstrong foi testado em 25 casos e, em geral, levou a resultados melhores que os dois últimos. Koo and Lee (1992) fizeram um esquema de reordenamento utilizando o algoritmo CM para o ordenamento dos elementos finitos e criaram um novo esquema para a numeração dos nós. O esquema introduzido é baseado na teoria de grafos e os resultados são comparados com diversos algoritmos existentes, demonstrando eficiência e confiabillidade. Segundo Koo and Lee (1992), o algoritmo de

Gibs Poole and Stockmeyer (GPS), é conhecido na literatura por ser superior em relação a tempo de processamento e resultados, sendo que o algoritmo por eles proposto não é superior ao GPS. Paulino *et al.* (1994) desenvolveram um algoritmo chamado de Spectral Finite Element Graph Resequencing (SFR), o qual segundo os resultados apresentados na época, tornou-se o mais eficiente algoritmo da literatura em termos de tempo de processamento e resultados, sendo inclusive superior ao GPS. No entanto, observa-se um moderado grau de dificuldade para implementar tal algoritmo.

Também em relação à teoria matemática, O MEF é objeto de pesquisa atualmente. Strouboulis and Haque (1992) apresentaram estimativas de erros e métodos adaptativos em malhas triangulares e quadriláteras. Foram realizadas diversas estimativas para aproximações em EF para PVCs elípticos. As estimativas foram obtidas usando diversos métodos encontrados na literatura, utilizando a Equação de Poisson numa malha de triângulos hierárquicos de ordem p ($1 \leq p \leq 7$). O artigo comparou as diversas estimativas de erros, visando encontrar a melhor para cada caso. Stewart e Hughes (1997) citam que as estimativas de erros se classificam em duas grandes categorias: estimativas a *priori* e a *posteriori*. Este trabalho foi apresentado como um tutorial do MEF nas estimativas de erro com grande atenção na demonstração matemática.

Pode-se encontrar diversos trabalhos na área de Elementos Finitos na própria Universidade Federal de Santa Catarina (UFSC). Tello (1991) usou o MEF em três dimensões para dimensionar sistemas de aterramento em baixas freqüências. A comparação de seus resultados com outras duas metodologias mostrou que o MEF fornece resultados coerentes, justificando seu uso. Rabelo (1992) formulou e avaliou modelos de elementos finitos sólidos ortotrópicos isoparamétricos baseados no modelo de deslocamentos, sendo aplicados à análise de placas espessas e semi-espessas, no contexto do estudo de materiais compostos fibrosos unidos através de técnicas de laminação. Observa-se que a elevação da ordem das funções de interpolação, investigada no trabalho, deveria ser precedida por uma análise de viabilidade e necessidade, aproveitando o fato de que o MEF possui uma forte estrutura matemática, a qual permite verificar, sob determinadas hipóteses, a influência do aumento da ordem das funções de interpolação na convergência do método. Medeiros (1994) estudou vários métodos visando a modelagem de problemas com fronteiras abertas, aplicados em conjunto com o MEF. Este é um trabalho da área de eletromagnetismo e está

relacionado com a resolução das Equações de Maxwell. O autor constata problemas na solução quando o domínio se estende ao infinito e trata de alternativas que visam resolver problemas com domínios abertos. Serafim (1998) implementou uma biblioteca de funções de forma para diversos tipos de elementos finitos, em duas e três dimensões. Foram desenvolvidas as funções de interpolação para os diversos tipos de elementos: lineares, quadrático, cúbico completo, quadrilátero, tetraédricos e hexaédricos.

Os trabalhos apresentados na área de Matemática Computacional do Curso de Pós-Graduação em Ciência da Computação (CPGCC), até o momento, envolvem a utilização do MVF em malhas geradas por diagramas de Voronoi (Cardoso, 1997; Mariani, 1997). O estudo do MEF é uma área que está sendo recentemente estudada no curso de computação, sendo que este método tem sido utilizado, tradicionalmente, na análise de estruturas estáticas e dinâmicas (Rabelo, 1992). Ele também tem sido empregado na solução de problemas de Transferência de Calor e Mecânica dos Fluidos (Adamik *et al.*, 1989).

O objetivo do presente trabalho é estudar alguns dos pontos chave da implementação computacional do MEF na solução de problemas governados por EDPs elípticas, os chamados Problemas a Valores no Contorno ou PVCs. Na verdade, independente do problema, o MEF é essencialmente um método de solução numérica de EDPs. Por isto, é escolhida uma EDP representativa de muitos problemas nas áreas mencionadas acima, a equação de Poisson: $-\nabla^2 u = f$ e é aplicado o MEF para a obtenção de soluções numéricas desta equação, em domínios com formatos quaisquer e sujeitos a qualquer combinação de CC. de Dirichlet e Neumann. Ainda pode-se ter diferentes propriedades físicas no domínio e pontos com cargas concentradas e/ou distribuídas.

O código computacional obtido inclui a geração de malha em domínios computacionais bidimensionais arbitrários, estimativas de erros para as soluções obtidas, com base na teoria dos elementos finitos, um reordenamento dos nós computacionais, de modo que o sistema global de equações algébricas resultante seja esparso e com a menor largura de banda possível, além de outros aspectos considerados intermediários, como por exemplo, a resolução de sistemas lineares, geração de pontos internos ao contorno, integração numérica, etc.

O programa prevê a discretização do domínio em triângulos, que neste caso é realizada através da triangulação de Delaunay (TDEL). Sabe-se que os erros cometidos nas aproximações das soluções de EDPs dependem em muito da forma dos triângulos usados e, em geral, os erros são tanto maiores quanto menos equiláteros forem os triângulos (Resende, *et al.*, 1994). A TDEL foi escolhida por ser o método que produz os triângulos mais equiláteros possíveis. Utiliza-se funções de interpolação lineares ou quadráticas na ligação entre os nós do domínio.

Este trabalho está estruturado em 8 capítulos. No capítulo 1, encontra-se uma revisão bibliográfica sobre vários aspectos relacionados ao Método dos Elementos Finitos. No capítulo 2, apresenta-se a caracterização do problema em estudo. No capítulo 3 é exposta a técnica para geração da malha computacional. No capítulo 4 são descritos os procedimentos numéricos e estimativas de erros, sendo que algumas definições e teoremas estão no apêndice A. No capítulo 5 é realizada a validação do código computacional implementado, que é feita através da comparação dos resultados obtidos pelo MEF com os resultados analíticos de diversos problemas. No capítulo 6 é descrito o código computacional desenvolvido. No capítulo 7 compara-se os resultados do código computacional implementado com outras metodologias de resolução, e no capítulo 8 expõe-se as considerações finais e sugestões para futuros trabalhos a serem desenvolvidos sobre o tema. Em anexo constam os arquivos fonte do programa implementado.

Capítulo 2 – Caracterização do Problema

2.1 Introdução

O MEF destaca-se na resolução de PVCs em 2D e 3D. O código computacional desenvolvido é direcionado principalmente à solução de problemas bidimensionais, mas também poderá ser aplicado à solução de problemas unidimensionais. O caso unidimensional pode ser importante durante o processo de análise de uma solução obtida para o caso 2D ou mesmo na solução de problemas que podem ser reduzidos à forma unidimensional.

Neste capítulo é apresentada a formulação de EF para um problema bidimensional genérico, cuja equação governante é a Equação de Poisson. A Figura 2.1 representa o domínio a ser discretizado.

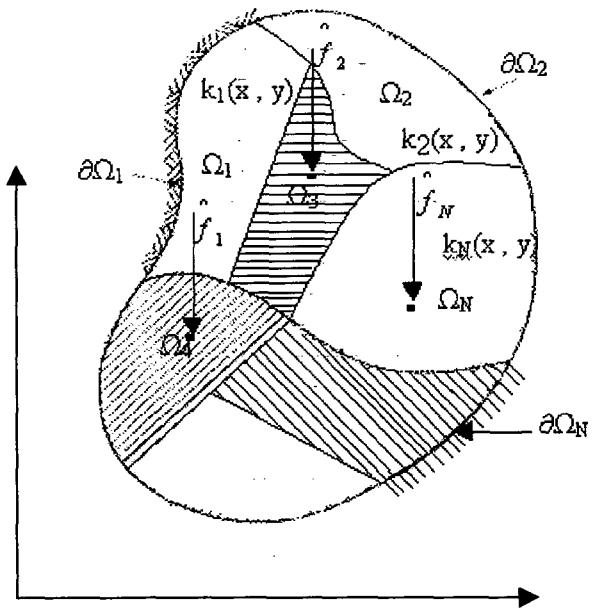


Figura 2.1 Domínio de um PVC bidimensional.

Observa-se na Figura 2.1 as irregularidades previstas no problema em estudo, destacando-se:

1) Contorno é genérico;

2) Cargas concentradas $\hat{f}_i$ em qualquer ponto;

3) Qualquer combinação de CC. de Neumann e Dirichlet ($\partial\Omega_1$, $\partial\Omega_2$... $\partial\Omega_N$);

4) Qualquer número de regiões com propriedades distintas $k_i=k_i(x,y)$ para $(x,y) \in \Omega_i$, i=1,2,...N;

5) Carregamento distribuído f(x, y) qualquer.

A equação governante resolvida pelo código computacional implementado é a equação de Poisson de 2ª. ordem linear, representada pela equação:

$$\frac{-\partial}{\partial x}\left[k_x(x,y)\frac{\partial u}{\partial x}\right]-\frac{\partial}{\partial y}\left[k_y(x,y)\frac{\partial u}{\partial y}\right]=f(x,y) \qquad (2.1)$$

Na primeira parte deste capítulo são discutidas as irregularidades contidas no problema. Em seguida, são definidos alguns conceitos básicos e por fim é apresentada a discretização da Equação de Poisson.

2.2 Irregularidades do Problema

Uma das vantagens do MEF é a facilidade de tratar as irregularidades apresentadas no problema. Neste tópico são analisadas essas irregularidades. Para facilitar o entendimento, consideremos o domínio unidimensional da Figura 2.2.

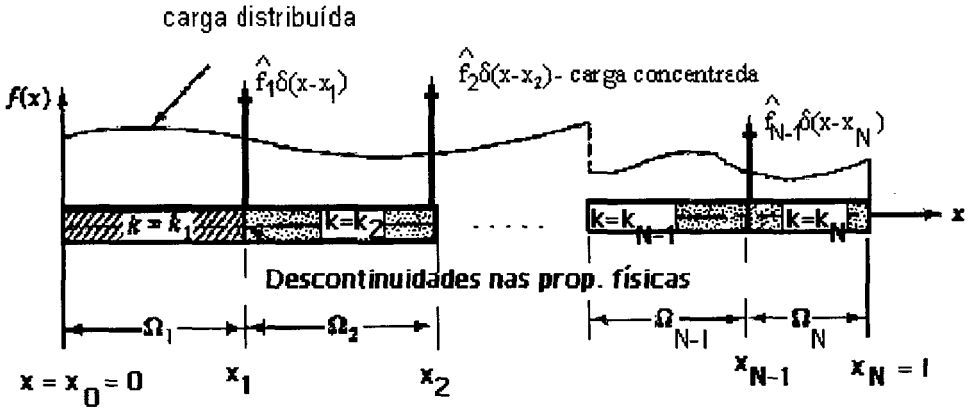


Figura 2.2 Domínio Unidimensional (diversas irregularidades).

Para esta discussão consideremos a equação (2.2) e as condições de contorno representadas pela equação (2.3). A equação (2.3) representa a combinação de condições de contorno de Neumann e Dirichlet.

$$\frac{-d}{dx}\left[k(x)\frac{du}{dx}\right]=f(x) \tag{2.2}$$

$$u(N)=u_N \quad \text{ou} \quad k(N)\frac{du}{dx}(N)=\tau_N \quad , \quad \text{N} = 0..1 \tag{2.3}$$

É importante lembrar que, na maioria das aplicações práticas, não há solução analítica para problemas do tipo acima, devido à não suavidade dos dados (irregulares ou descontínuos), à complexidade do domínio, dos coeficientes ou das condições de contorno.

2.2.1 Descontinuidades nas Propriedades Físicas

Conforme Figuras 2.1 e 2.2, pode-se notar que o domínio pode possuir diversos pontos com valores de k(x) diferentes. No MEF, exatamente nos pontos onde ocorre esta mudança de propriedades físicas, deve-se ter um nó na malha. Sendo que estas descontinuidades são facilmente tratadas, pois o fluxo fica contínuo, mesmo se k(x) é descontínuo.

2.2.2 Descontinuidade nos carregamentos

Na análise das descontinuidades, por exemplo, o degrau na carga em $x=x_{N-2}$ na Figura 2.2, o fluxo também fica contínuo quando o carregamento é descontínuo.

2.2.3 Cargas Concentradas

Tomando-se, por exemplo, o ponto $x = x_2$, onde assume-se que há uma carga

concentrada valendo $\hat{f}_2$, e integrando a equação (2.2) perto deste ponto, tem-se:

$$\int_{x_2-\Delta x}^{x_2+\Delta x}\left[\frac{-d}{dx}\left(k(x)\frac{du}{dx}\right)\right]dx = \int_{x_2-\Delta x}^{x_2+\Delta x} f(x)dx + \hat{f}_2\,\delta(x-x2) \tag{2.4}$$

Integrando a equação (2.4) e fazendo $\Delta x \to 0$, obtem-se:

$$\left[-k(x)\frac{du}{dx}\right]_{x_2^+} - \left[-k(x)\frac{du}{dx}\right]_{x_2^-} = \hat{f}_2 \tag{2.5}$$

Portanto, o valor de $\hat{f}_2$ deverá entrar no sistema no nó correspondente à carga concentrada, como será visto mais adiante.

Em suma, após analisar estas irregularidas, pode-se concluir que a solução da equação (2.2) é dada pela função contínua u=u(x) que satisfaz:

1) A equação diferencial;

$$\frac{-d}{dx}\left[k(x)\frac{du}{dx}\right] = f(x) \quad x \in \Omega_i \text{ , i=1,2,..,N} \tag{2.6}$$

2) Condições de "salto" em todos os pontos de descontinuidade;

$$-\left[\!\left[ku'\right]\!\right] = \hat{f}_i, \quad x = x_i, \quad \text{onde} \quad -\left[\!\left[ku'\right]\!\right] = \lim_{x\to x_i^+}(ku') - \lim_{x\to x_i^-}(ku') \tag{2.7}$$

OBS.: Pode-se concluir que somente nos pontos onde há cargas concentradas, deve-se especificar o valor desta carga, pois nas outras descontinuidades o fluxo é contínuo. Mais adiante será exemplificado como estas cargas se incorporam ao código computacional desenvolvido.

3) Condições de Contorno.

2.3 Conceitos Básicos

Antes de iniciar a discretização da equação governante do problema, são definidos alguns conceitos que são utilizados neste processo.

2.3.1 Método de Resíduos Ponderados (MRP)

Os métodos de Resíduos Ponderados consistem em técnicas gerais de obtenção de soluções de EDPs, onde a solução desconhecida é expandida em um conjunto de funções testes. Estas funções devem satisfazer às CC. e são substituídas na EDP na forma de resíduo. Este resíduo, então, é forçado a se anular no sentido das médias ponderadas, ou seja:

$$\int r(x)v(x)dx = 0 \ , \ \text{i=1,2,...N} \tag{2.8}$$

onde r(x) é função de u(x) e v(x) é o conjunto das funções "teste" ou funções "peso". A seguir a função u(x) é substituída por uma aproximação do tipo:

$$u_N(x) = \sum_{j=1}^{N} \alpha_j \phi_j(x) \tag{2.9}$$

Ou seja, a solução é expandida em um conjunto de "funções tentativas". Por outro lado, as funções "teste" ou "peso" são expandidas em um outro conjunto de funções de base, conforme a equação abaixo:

$$v_N(x) = \sum_{i=1}^{N} \beta_i \Psi_i(x) \tag{2.10}$$

Cada escolha de funções teste corresponde a um critério diferente do MRP. No MEF utiliza-se o MRP de Galerkin, onde utiliza-se as mesmas funções para "teste" e "tentativa", ou seja: $\Psi_i = \phi_j$.

O MRP de Galerkin tem as seguintes características:

1) Minimiza o erro na norma da energia, definida conforme equação (2.11).

$$\|e\|_E := \left[\int_\Omega \left[e(x)'^2 + e(x)^2 \right] dx \right]^{1/2} \tag{2.11}$$

Onde:

$e(x) = u(x) - u_N(x)$;

$u(x)$ = solução analítica;

$u_N(x)$ = solução aproximada.

2) Resulta em matrizes simétricas.

2.3.2 Problemas Elípticos, Parabólicos e Hiperbólicos

A classificação dos problemas em elípticos, parabólicos e hiperbólicos é feita de acordo com uma relação entre os coeficientes da equação que governa o fenômeno (Guarabedian, 1964, p. 57 apud Fletcher, 1997):

$$A\frac{\partial^2 u}{\partial x^2}+B\frac{\partial^2 u}{\partial x\partial y}+C\frac{\partial^2 u}{\partial y^2}+D\frac{\partial u}{\partial x}+E\frac{\partial u}{\partial y}+Fu+G=0 \tag{2.12}$$

onde A, B, C, D, E, F e G são coeficientes constantes. De acordo com estes coeficientes, há três categorias de Equações Diferenciais:

$$\text{elíptica: } B^2-4AC<0 \tag{2.13}$$

$$\text{parabólica: } B^2-4AC=0 \tag{2.14}$$

$$\text{hiperbólica: } B^2-4AC>0 \tag{2.15}$$

Segundo Maliska (1995), problemas de transferência de calor, por exemplo, são governados por sistemas de equações, onde a classificação do sistema é quase sempre mista.

Analisando do ponto de vista numérico, tem que se observar as características das equações, com a finalidade de reduzir o tempo de processamento. Interpretando as vantagens computacionais, é interessante definir, quando possível, os problemas de transferência de calor e mecânica dos fluidos em problemas que permitam a sua solução pelo processo de "marcha" em uma determinada coordenada (tempo ou espaço).

A equação governante que é resolvida é a equação de Poisson, ou seja, uma equação elíptica.

2.3.3 Interpretação Física de EDPs Elípticas

A principal característica dos problemas elípticos é que, ao se considerar um ponto P no interior do domínio (Figura 2.3), esse ponto influencia todos os outros pontos do domínio computacional. Isto implica que, para obter a solução, deve-se considerar o domínio como um todo. Ao contrário da resolução de problemas parabólicos e hiperbólicos, os quais podem ser resolvidos pelo processo de "marcha", a partir das condições iniciais.

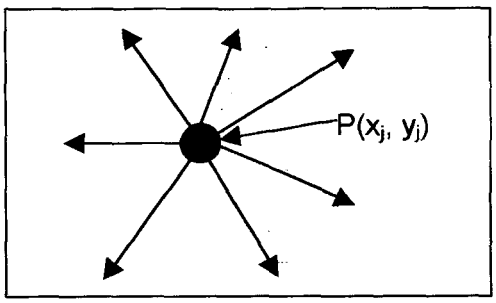


Figura 2.3 Problemas Elípticos.

2.3.4 Teorema da Divergência

Este teorema é utilizado na formulação variacional do problema, onde é necessário transformar uma integral de área em uma integral no contorno.

Se $\vec{F}$ (x,y,z) e $\vec{\nabla}\vec{F}$ são contínuas sobre uma superfície S e no interior do volume V, e se $\hat{n}$ é o vetor unitário perpendicular (externo) a S em um ponto genérico, então:

$$\iiint_V \left(\vec{\nabla}.\vec{F}\right)dV = \iint_S (\hat{n}.\vec{F})\,dS \tag{2.16}$$

O teorema também pode ser escrito como:

$$\int_\Omega \left(\frac{\partial F}{\partial x} + \frac{\partial G}{\partial y}\right)dxdy = \int_{\partial\Omega} \left(F\ \mathrm{n}_x + G\,\mathrm{n}_y\right)dS \tag{2.17}$$

2.4 Discretização da Equação de Poisson

Neste tópico é apresentada a discretização da equação governante, a qual foi apresentada na introdução deste capítulo, relembrando que as irregularidades do problema, bem como as condições de contorno, foram apresentadas na Figura 2.1. Reescrevendo a equação goventante:

$$\frac{-\partial}{\partial x}\left[k_x(x,y)\frac{\partial u}{\partial x}\right]-\frac{\partial}{\partial y}\left[k_y(x,y)\frac{\partial u}{\partial y}\right]=f(x,y) \tag{2.18}$$

A solução do problema é dada pela função u = u(x) que satisfaz à equação (2.18), às condições de contorno e a todas as irregularidades do problema.

No MEF os passos básicos para construção de soluções aproximadas, segundo Becker and Carey (1981), são:

1) Formulação do problema variacional, identificando as funções admissíveis em um subespaço H;
2) Construção da malha e definição de polinômios locais, como funções de base sobre a malha;
3) Construção de uma aproximação do PVC no subespaço H^h de H. Montagem do sistema global de equações algébricas lineares, que representam a solução aproximada nos pontos nodais da malha;
4) Solução do sistema algébrico;
5) Estimativas de erro.

2.4.1 Formulação Variacional

O tratamento clássico de equações diferenciais exige que a solução satisfaça à equação em todos os pontos do domínio. Devido à existência de matrizes com propriedades diferentes, o que causa descontinuidades nas interfaces, ou uma região do domínio que gera uma singularidade, isso torna-se uma exigência muito forte. Então, o

objetivo da formulação variacional é desenvolver um método de aproximação numérica que possa tratar irregularidades no problema, são as chamadas "Formulações Fracas ou Variacionais".

Uma formulação fraca do problema pode ser descrita como:

"Encontre a função u(x) tal que a EDP, juntamente com as CC. e todas as irregularidades do problema, seja satisfeita no sentido das médias ponderadas". Isto significa que "todas as médias ponderadas" de uma certa classe da EDP devem ser satisfeitas e obviamente todas as irregularidades do problema, também devem ser satisfeitas.

$$\int_{\Omega}\left\{-\frac{\partial}{\partial x}\left[k_x(x,y)\frac{\partial u}{\partial x}\right]-\frac{\partial}{\partial y}\left[k_y(x,y)\frac{\partial u}{\partial y}\right]-f(x,y)\right\}v\,dxdy=0 \quad , \quad \forall\, v \in H \tag{2.19}$$

onde:

v := qualquer conjunto de "funções-teste" ou "funções-peso" (ou seja, qualquer função bem comportada que facilite a resolução da integral).

H := o conjunto de todas as funções suficientemente suaves, que servem como função-teste.

Então, no Método dos Elementos Finitos a expressão entre chaves da equação (2.19) é definida como um resíduo, que é dado por:

$$r(x,y)=\left\{-\frac{\partial}{\partial x}\left[k_x(x,y)\frac{\partial u}{\partial x}\right]-\frac{\partial}{\partial y}\left[k_y(x,y)\frac{\partial u}{\partial y}\right]-f(x,y)\right\} \tag{2.20}$$

Multiplicando a equação (2.20) por um conjunto de funções teste (v) e integrando sobre todo o domínio, tem-se:

$$\int_{\Omega}\left\{\frac{-\partial}{\partial x}\left[k_x(x,y)\frac{\partial u}{\partial x}\right]-\frac{\partial}{\partial y}\left[k_y(x,y)\frac{\partial u}{\partial y}\right]-f(x,y)\right\}v\,dxdy \;=0 \tag{2.21}$$

O processo abaixo é equivalente à integração por partes. Este procedimento é necessário para reduzir o primeiro termo da integral em derivada primeira.

$$\frac{\partial}{\partial x}\left[k_x\frac{\partial u}{\partial x}\right]v=\frac{\partial}{\partial x}\left[k_x\frac{\partial u}{\partial x}v\right]-\left[k_x\frac{\partial u}{\partial x}\right]\frac{\partial v}{\partial x} \tag{2.22}$$

$$\frac{\partial}{\partial y}\left[k_y \frac{\partial u}{\partial y}\right]v = \frac{\partial}{\partial y}\left[k_y \frac{\partial u}{\partial y} v\right] - \left[k_y \frac{\partial u}{\partial y}\right]\frac{\partial v}{\partial y} \tag{2.23}$$

Substituindo as equações (2.22) e (2.23) na equação (2.21), vem:

$$\int_\Omega \left\{\frac{\partial v}{\partial x}\left[k_x \frac{\partial u}{\partial x}\right] + \frac{\partial v}{\partial y}\left[k_y \frac{\partial u}{\partial y}\right] - fv\right\} dxdy - \left\{\int_\Omega \left[\frac{\partial}{\partial x}\left(k_x \frac{\partial u}{\partial x} v\right) + \frac{\partial}{\partial y}\left(k_y \frac{\partial u}{\partial y} v\right)\right] dxdy\right\} \tag{2.24}$$

O último termo da equação (2.24) pode ser transformado em uma integral sobre o contorno, usando o teorema da divergência (equação (2.17)).

$$-\left\{\int_\Omega \left[\frac{\partial}{\partial x}\left(k_x \frac{\partial u}{\partial x} v\right) + \frac{\partial}{\partial y}\left(k_y \frac{\partial u}{\partial y} v\right)\right] dxdy\right\} = -\left\{\int_{\partial\Omega} \left[\left(k_x \frac{\partial u}{\partial x}\right) v\, n_x + \left(k_y \frac{\partial u}{\partial y}\right) v\, n_y\right] ds\right\} \tag{2.25}$$

Definindo:

$$\tau_x = -k_x \frac{\partial u}{\partial x} \tag{2.26}$$

$$\tau_y = -k_y \frac{\partial u}{\partial y} \tag{2.27}$$

O que leva a:

$$\tau_n = \tau_x n_x + \tau_y n_y \tag{2.28}$$

Logo é obtida a integral no contorno, ou seja:

$$-\left\{\int_\Omega \left[\frac{\partial}{\partial x}\left(k_x \frac{\partial u}{\partial x} v\right) + \frac{\partial}{\partial y}\left(k_y \frac{\partial u}{\partial y} v\right)\right] dxdy\right\} = \int_{\partial\Omega} \tau_n v ds \tag{2.29}$$

Então, substituindo a equação (2.29) na equação (2.24) , tem-se:

$$\int_\Omega \left\{\frac{\partial v}{\partial x}\left[k_x \frac{\partial u}{\partial x}\right] + \frac{\partial v}{\partial y}\left[k_y \frac{\partial u}{\partial y}\right] - fv\right\} dxdy = -\int_{\partial\Omega} \tau_n v ds \tag{2.30}$$

As funções admissíveis para a equação (2.30), são as funções $v \in H^1(\Omega)$, onde $H^1(\Omega)$ é o espaço de Sobolev de ordem 1, ou seja, o espaço das funções suficientemente suaves de forma que suas primeiras derivadas sejam integráveis ao quadrado sobre Ω, o

que define a norma:

$$\|v\|_1 := \int_\Omega \left[\left(\frac{\partial v}{\partial x} \right)^2 + \left(\frac{\partial v}{\partial y} \right)^2 + v^2 \right] d\Omega < \infty \tag{2.31}$$

Busca-se a solução da equação (2.30), através das aproximações:

$$u(x,y) = \sum_{j=1}^{N} u_j \Psi_j(x,y) \tag{2.32}$$

$$v(x,y) = \sum_{i=1}^{N} v_i \Psi_i(x,y) \tag{2.33}$$

2.5 Aproximações de Galerkin

O problema variacional representado na equação (2.30), pode ser denotado como:

$$\int_\Omega \left[\left(k_x \frac{\partial u}{\partial x} \frac{\partial v}{\partial x} + k_y \frac{\partial u}{\partial y} \frac{\partial v}{\partial y} \right) - fv \right] dxdy = -\int_{\partial\Omega} v \tau_n ds \tag{2.34}$$

E, substituindo as aproximações, conforme equações (2.32) e (2.33), chega-se a:

$$\sum_{j=1}^{N} K_{ij} u_j = F_i \quad , \quad i = 1,2,..N \tag{2.35}$$

Onde:

$$K_{ij} = \int_{\Omega_h} \left[\left(k_x \frac{\partial \Psi_i}{\partial x} \frac{\partial \Psi_j}{\partial x} + k_y \frac{\partial \Psi_i}{\partial y} \frac{\partial \Psi_j}{\partial y} \right) \right] dxdy \tag{2.36}$$

$$F_i = \int_{\Omega_h} f \Psi_i dxdy - \int_{\partial\Omega_h} \tau_n \Psi_i ds \tag{2.37}$$

O MEF fornece uma técnica geral para a construção destas funções de base. Na aplicação deste método, o domínio é dividido ou discretizado em elementos finitos. Sobre cada elemento são identificados certos pontos, que são chamados de nós, sendo que o conjunto de nós e elementos é o que se denomina de malha de elementos finitos.

2.6 Implementação dos Elementos Finitos

A idéia básica da interpolação usada para problemas bidimensionais é representar a solução u_h e as funções teste v_h por polinômios de baixa ordem em sub-regiões do domínio Ω_h, no plano (x, y).

Neste trabalho são utilizados triângulos lineares e quadráticos para a interpolação do domínio.

2.6.1. Triângulos Lineares

Neste caso a interpolação varia linearmente no interior do domínio. Conforme a Figura 2.4, v_h é dado por:

$$v_h^e(x,y) = a_1 + a_2 x + a_3 y \qquad \forall\ (x,y) \in \Omega_e \tag{2.38}$$

Onde, A_e é a área do elemento Ω_e e os coeficientes valem:

$$a_1 = \frac{1}{2A_e}\left[(x_2 y_3 - x_3 y_2)v_1 + (x_3 y_1 - x_1 y_3)v_2 + (x_1 y_2 - x_2 y_1)v_3\right] \tag{2.39}$$

$$a_2 = \frac{1}{2A_e}\left[(y_2 - y_3)v_1 + (y_3 - y_1)v_2 + (y_1 - y_2)v_3\right] \tag{2.40}$$

$$a_3 = \frac{1}{2A_e}\left[(x_3 - x_2)v_1 + (x_1 - x_3)v_2 + (x_2 - x_1)v_3\right] \tag{2.41}$$

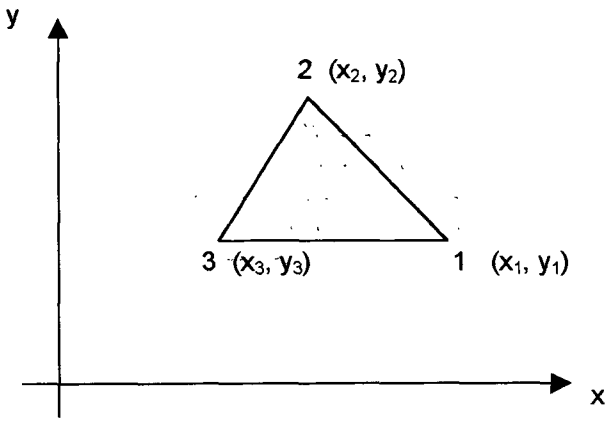


Figura 2.4 Interpolação linear no plano x e y.

A equação (2.38) também pode ser colocada na forma:

$$v_h^e(x,y) = a_1(v_1,v_2,v_3) + a_2(v_1,v_2,v_3)x + a_3(v_1,v_2,v_3)y \tag{2.42}$$

$$v_h^e(x,y) = v_1\Psi_1^e(x,y) + v_2\Psi_2^e(x,y) + v_3\Psi_3^e(x,y) \tag{2.43}$$

Onde os $\Psi_i^e(x,y)$ são as funções de base elementares que aparecem nas equações (2.36) e (2.37):

$$\Psi_1^e(x,y) = \frac{1}{2A_e}\left[(x_2y_3 - x_3y_2) + (y_2 - y_3)x + (x_3 - x_2)y\right] \tag{2.44}$$

$$\Psi_2^e(x,y) = \frac{1}{2A_e}\left[(x_3y_1 - x_1y_3) + (y_3 - y_1)x + (x_1 - x_3)y\right] \tag{2.45}$$

$$\Psi_3^e(x,y) = \frac{1}{2A_e}\left[(x_1y_2 - x_2y_1) + (y_1 - y_2)x + (x_2 - x_1)y\right] \tag{2.46}$$

As funções lineares das equações (2.44) a (2.46) podem ser vizualidas na Figura 2.4. Observa-se que $\Psi_i^e(x_j,y_j)$ é igual a 1 (um) se i = j e 0 (zero) se i ≠ j, para i, j = 1, 2, 3, ou seja:

$$\Psi_i^e(x_j,y_j) = \begin{Bmatrix} 1, & i = j \\ 0, & i \neq j \end{Bmatrix} \tag{2.47}$$

Compondo as funções de forma, obtem-se as funções de base globais (pirâmide),

conforme Figura 2.5.

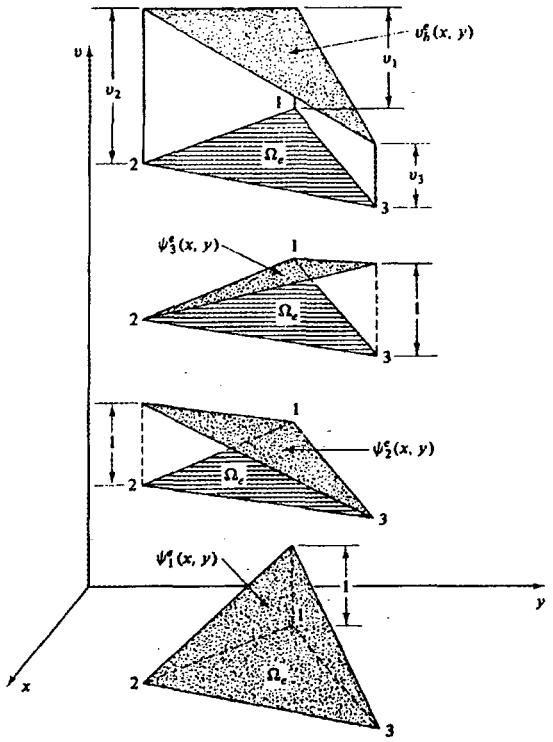


Figura 2.5 Combinação das funções de base linear.
Fonte: Carey and Becker (1981).

2.6.2 Elementos Triangulares de Mais Alta Ordem

Outros triângulos com polinômios de mais alta ordem em x e y podem ser construídos com grande facilidade, utilizando o triângulo de Pascal.

1	Grau 0
x y	Grau 1
x^2 xy y^2	Grau 2
x^3 x^2y xy^2 y^3	Grau 3
x^4 x^3y x^2y^2 xy^3 y^4	Grau 4

Uma das vantagens da utilização de polinômios como funções de base é que, para grau > 0, nota-se que a função tem a primeira derivada parcial quadrado integrável. Para o caso dos triangulos quadráticos, então:

$$v_h^e(x,y) = a_1 + a_2 x + a_3 y + a_4 x^2 + a_5 xy + a_6 y^2 \tag{2.48}$$

Onde os coeficientes também podem ser deduzidos e da mesma forma que para o linear chega-se as funções $\Psi_i^e(x,y)$.

2.7 Matrizes Elementares

Agora que já foram definidas as funções Ψ_i e Ψ_j, pode-se restringir as aproximações u_h e v_h dadas nas equações (2.32) e (2.33) apenas ao elemento Ω_e (elemento mestre), obtendo:

$$u_h^e(x,y) = \sum_{j=1}^{Ne} u_j^e \Psi_j^e(x,y) \tag{2.49}$$

$$v_h^e(x,y) = \sum_{j=1}^{Ne} v_j^e \Psi_j^e(x,y) \tag{2.50}$$

onde N_e é o número de nós em Ω_e.

Substituindo estas aproximações na formulação variacional (equação (2.34)), aplicada apenas a um elemento (Ω_e), tem-se a equação (2.51).

$$\int_{\Omega_e} \left(k_x^e \frac{\partial u_h^e}{\partial x}\frac{\partial v_h^e}{\partial x} + k_y^e \frac{\partial u_h^e}{\partial y}\frac{\partial v_h^e}{\partial y} \right) dxdy = \int_{\Omega_e} f v_h^e dxdy - \int_{\partial\Omega_e} v_h^e \tau_n ds \tag{2.51}$$

E então, chega-se ao sistema:

$$\sum_{j=1}^{Ne} K_{ij}^e u_j^e = f_i^e - \tau_i^e, \quad \text{i=1,2,..}N_e \tag{2.52}$$

Onde:

$$K_{ij}^{\ e} = \int_{\Omega e}\left[k_x^{\ e}\left(\frac{\partial \Psi_i^{\ e}}{\partial x}\frac{\partial \Psi_j^{\ e}}{\partial x}\right) + k_y^{\ e}\left(\frac{\partial \Psi_i^{\ e}}{\partial y}\frac{\partial \Psi_j^{\ e}}{\partial y}\right)\right]dxdy \tag{2.53}$$

$$f_i^{\ e} = \int_{\Omega e} f\Psi_i^{\ e}dxdy \tag{2.54}$$

$$\tau_i^{\ e} = \int_{\partial\Omega e} \tau_n \Psi_i^{\ e} ds \tag{2.55}$$

Expandindo as matrizes e somando sobre todos os E elementos da malha, tem-se no sistema global:

$$\sum_{e=1}^{E}\left(K_{ij}^{\ e}u_j - F_i^{\ e} + \sum i^e\right) = 0 \tag{2.56}$$

Finalmente,

$$\sum_{j=1}^{N} K_{ij}u_j = F_i - S_i \qquad \text{, i=1,2,.. N} \tag{2.57}$$

Onde:

$$K_{ij} = \sum_{e=1}^{E} K_{ij}^{\ e} \tag{2.58}$$

$$F_i = \sum_{e=1}^{E} F_i^{\ e} \tag{2.59}$$

$$S_i = \sum_{e=1}^{E}\sum i^{\ e} \tag{2.60}$$

Ou seja, para se chegar à solução do problema, basta resolver o sistema

$$K_{ij}.u_j = F_i - S_i \tag{2.61}$$

Contudo, pode-se observar que é requerido um alto custo computacional para determinar K_{ij}, F_i e S_i em uma malha com um grande número de nós, recorrendo-se então, às transformações elementares que são tratadas no próximo tópico.

2.8 Transformações Elementares

Até o momento foi tratado problemas em coordenadas cartesianas locais para o sistema abordado neste trabalho. Computacionalmente isso não é eficiente e então recorre-se às transformações. Na aplicação do MEF a um PVC dado, deve-se construir uma malha Ω_h e desenvolver um método geral e sistemático para o cálculo das matrizes elementares em 2D. A idéia geral é ter um elemento mestre, Ω, sobre o qual todos os cálculos são realizados.

Conforme Figura 2.6, todos os cálculos são feitos sobre um elemento mestre $\hat{\Omega}$. A ligação dos resultados obtidos para este elemento com os elementos reais (plano x e y) que efetivamente compõem o domínio é feita com o auxílio de transformações. Esta facilidade em obter as transformações é o que torna o MEF muito versátil.

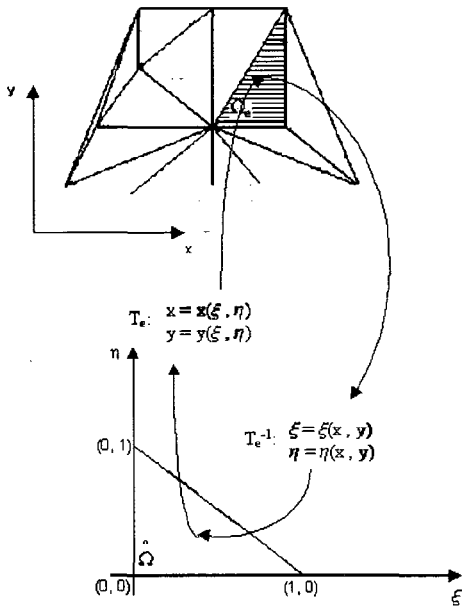


Figura 2.6 Um elemento finito no plano x, y e transformações.

Será considerada a seguinte transformação:

$$T_e : \begin{Bmatrix} x = x(\xi, \eta) \\ y = y(\xi, \eta) \end{Bmatrix} \tag{2.62}$$

Segundo Becker *et al.* (1981), a geração da malha de Elementos Finitos que

contém E elementos é vista como uma sequência de transformações $\{T_1, T_2, \ldots, T_E\}$ na forma da equação (2.62), na qual cada elemento Ω_e é a imagem fixa do elemento mestre sobre uma coordenada mapeada T_e.

Outras propriedades das transformações são detalhadas a seguir. Tais propriedades são fundamentais na ligação entre os resultados das integrações efetuadas sobre o elemento mestre e os resultados sobre os elementos reais.

Primeiramente, deve-se considerar que x e y são contínuos e diferenciáveis em relação a ξ e η, logo:

$$\begin{Bmatrix} dx = \dfrac{\partial x}{\partial \xi} d\xi + \dfrac{\partial x}{\partial \eta} d\eta \\ dy = \dfrac{\partial y}{\partial \xi} d\xi + \dfrac{\partial y}{\partial \eta} d\eta \end{Bmatrix} \tag{2.63}$$

ou, então:

$$\begin{bmatrix} dx \\ dy \end{bmatrix} = \begin{bmatrix} \dfrac{\partial x}{\partial \xi} & \dfrac{\partial x}{\partial \eta} \\ \dfrac{\partial y}{\partial \xi} & \dfrac{\partial y}{\partial \eta} \end{bmatrix} \begin{bmatrix} d\xi \\ d\eta \end{bmatrix} \tag{2.64}$$

Onde o determinante da matriz 2 X 2 dada na equação (2.64) de derivadas parciais é denominado Matriz Jacobiana da transformação $|T_e| = J$

O sistema acima pode ser invertido desde que:

$$|J| = \frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \xi} \neq 0 \quad \text{em } \Omega \tag{2.65}$$

Neste caso,

$$\begin{bmatrix} d\xi \\ d\eta \end{bmatrix} = J^{-1} \begin{bmatrix} dx \\ dy \end{bmatrix} = \frac{1}{|J|} \begin{bmatrix} \dfrac{\partial y}{\partial \eta} & \dfrac{-\partial x}{\partial \eta} \\ \dfrac{-\partial y}{\partial \xi} & \dfrac{\partial x}{\partial \xi} \end{bmatrix} \begin{bmatrix} dx \\ dy \end{bmatrix} \tag{2.66}$$

$$\begin{bmatrix} d\xi \\ d\eta \end{bmatrix} = \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \xi}{\partial y} \\ \frac{\partial \eta}{\partial x} & \frac{\partial \eta}{\partial y} \end{bmatrix} \begin{bmatrix} dx \\ dy \end{bmatrix} \tag{2.67}$$

Comparando as equações (2.66) e (2.67), tem-se:

$$\frac{\partial \xi}{\partial x} = \frac{1}{|J|}\frac{\partial y}{\partial \eta} \tag{2.68}$$

$$\frac{\partial \xi}{\partial y} = -\frac{1}{|J|}\frac{\partial x}{\partial \eta} \tag{2.69}$$

$$\frac{\partial \eta}{\partial x} = -\frac{1}{|J|}\frac{\partial y}{\partial \xi} \tag{2.70}$$

$$\frac{\partial \eta}{\partial y} = \frac{1}{|J|}\frac{\partial x}{\partial \xi} \tag{2.71}$$

2.9 Cálculos a Nível de Elemento

Deve-se escolher um elemento $\hat{\Omega}$ com coordenadas ξ e η e o tipo de elemento a ser utilizado (no caso triangular). As coordenadas do mapeamento são construídas via:

$$T_e : \begin{Bmatrix} x(\xi,\eta) = \sum_{j=1}^{n} x_j \Psi_j(\xi,\eta) \\ y(\xi,\eta) = \sum_{j=1}^{n} y_j \Psi_j(\xi,\eta) \end{Bmatrix} \tag{2.72}$$

Onde n é o número de nós do elemento mestre.

$$x = x(\xi,\eta) \tag{2.73}$$

$$y = y(\xi,\eta) \tag{2.74}$$

Para que o mapeamento seja aceitável, deve-se ter $|J|>0$. Voltando as equações (2.68) a (2.71) e substituindo o mapeamento, vem:

$$\frac{\partial \xi}{\partial x} = \frac{1}{|J|}\sum_{j=1}^{n} y_j \frac{\partial \Psi_j}{\partial \eta} \tag{2.75}$$

$$\frac{\partial \eta}{\partial x} = -\frac{1}{|J|}\sum_{j=1}^{n} y_j \frac{\partial \Psi_j}{\partial \xi} \tag{2.76}$$

$$\frac{\partial \xi}{\partial y} = -\frac{1}{|J|}\sum_{j=1}^{n} x_j \frac{\partial \Psi_j}{\partial \eta} \tag{2.77}$$

$$\frac{\partial \eta}{\partial y} = \frac{1}{|J|}\sum_{j=1}^{n} x_j \frac{\partial \Psi_j}{\partial \xi} \tag{2.78}$$

$$|J| = \left\{\sum_{j=1}^{n} x_j \frac{\partial \Psi_j}{\partial \xi}\right\}\left\{\sum_{j=1}^{n} y_j \frac{\partial \Psi_j}{\partial \eta}\right\} - \left\{\sum_{j=1}^{n} x_j \frac{\partial \Psi_j}{\partial \eta}\right\}\left\{\sum_{j=1}^{n} y_j \frac{\partial \Psi_j}{\partial \xi}\right\} \tag{2.79}$$

A matriz de rigidez está em coordenadas reais x, y dos elementos, conforme equação (2.53).

$$K_{ij}^{\ e} = \int_{\Omega e}\left[k_x^{\ e}\left(\frac{\partial \Psi_i^{\ e}}{\partial x}\frac{\partial \Psi_j^{\ e}}{\partial x}\right) + k_y^{\ e}\left(\frac{\partial \Psi_i^{\ e}}{\partial y}\frac{\partial \Psi_j^{\ e}}{\partial y}\right)\right] dxdy \tag{2.80}$$

Esta integral pode ser calculada de modo equivalente no domínio do elemento mestre, com o auxílio do mapeamento definido na equação (2.72).

Começa-se observando que as funções de forma $\Psi(x, y)$ que aparecem na equação (2.80) podem ser escritas como:

$$\Psi_j^{\ e}(x,y) = \Psi_j^{\ e}(\xi(x,y), \eta(x,y)) \quad j=1,2,\ldots N_e \tag{2.81}$$

Então, aplicando a regra da cadeia:

$$\frac{\partial \Psi_j^{\ e}}{\partial x} = \frac{\partial \Psi_j^{\ e}}{\partial \xi}\frac{\partial \xi}{\partial x} + \frac{\partial \Psi_j^{\ e}}{\partial \eta}\frac{\partial \eta}{\partial x} \tag{2.82}$$

$$\frac{\partial \Psi_j^{\ e}}{\partial y} = \frac{\partial \Psi_j^{\ e}}{\partial \xi}\frac{\partial \xi}{\partial y} + \frac{\partial \Psi_j^{\ e}}{\partial \eta}\frac{\partial \eta}{\partial y} \tag{2.83}$$

Logo, usando as equações 2.75 a 2.78, tem-se:

$$\frac{\partial \Psi_j^{\,e}}{\partial x} = \frac{\partial \Psi_j}{\partial \xi}\frac{1}{|J|}\sum_{k=1}^{N_e} y_k \frac{\partial \Psi_K}{\partial \eta} - \frac{\partial \Psi_j}{\partial \eta}\frac{1}{|J|}\sum_{k=1}^{N_e} y_k \frac{\partial \Psi_K}{\partial \xi} \tag{2.84}$$

$$\frac{\partial \Psi_j^{\,e}}{\partial y} = -\frac{\partial \Psi_j}{\partial \xi}\frac{1}{|J|}\sum_{k=1}^{N_e} x_k \frac{\partial \Psi_K}{\partial \eta} + \frac{\partial \Psi_j}{\partial \eta}\frac{1}{|J|}\sum_{k=1}^{N_e} x_k \frac{\partial \Psi_K}{\partial \xi} \tag{2.85}$$

Onde N_e é o número de nós do elemento mestre (Ω_e). Logo substituindo as equações (2.84) e (2.85) na equação (2.80), tem-se que:

$$K_{ij}^{\,e} = \int_{\Omega e} \left[\begin{array}{l} k_x^{\,e}\left(\frac{\partial \Psi_i}{\partial \xi}\frac{1}{|J|}\sum_{k=1}^{N_e} y_k \frac{\partial \Psi_K}{\partial \eta} - \frac{\partial \Psi_i}{\partial \eta}\frac{1}{|J|}\sum_{k=1}^{N_e} y_k \frac{\partial \Psi_K}{\partial \xi}\right)\left(\frac{\partial \Psi_j}{\partial \xi}\frac{1}{|J|}\sum_{k=1}^{N_e} y_k \frac{\partial \Psi_K}{\partial \eta} - \frac{\partial \Psi_j}{\partial \eta}\frac{1}{|J|}\sum_{k=1}^{N_e} y_k \frac{\partial \Psi_K}{\partial \xi}\right) \\ + k_y^{\,e}\left(-\frac{\partial \Psi_i}{\partial \xi}\frac{1}{|J|}\sum_{k=1}^{N_e} x_k \frac{\partial \Psi_K}{\partial \eta} + \frac{\partial \Psi_i}{\partial \eta}\frac{1}{|J|}\sum_{k=1}^{N_e} x_k \frac{\partial \Psi_K}{\partial \xi}\right)\left(-\frac{\partial \Psi_j}{\partial \xi}\frac{1}{|J|}\sum_{k=1}^{N_e} x_k \frac{\partial \Psi_K}{\partial \eta} + \frac{\partial \Psi_j}{\partial \eta}\frac{1}{|J|}\sum_{k=1}^{N_e} x_k \frac{\partial \Psi_K}{\partial \xi}\right) \end{array} \right] |J| d\xi d\eta \tag{2.86}$$

$$F_i^{\,e} = \int_{\Omega e} [f(\xi,\eta)\Psi_i] |J| d\xi d\eta \tag{2.87}$$

$$S_i^{\,e} = \int_{\partial \Omega e} \tau_n \Psi_i^{\,e} ds \tag{2.88}$$

Os cálculos da equação (2.88) são efetuados nos elementos do contorno. Para um melhor entendimento de como são realizados estes cálculos, observe como são aplicadas as condições de contorno de Neumann, no item 2.11.

Agora têm que ser definidas as funções de interpolação "$\Psi_i^{\,e}$", nas coordenadas ξ e η que são utililizadas na aproximação.

2.9.1 Cálculos Elementares para o Triângulo Linear

As funções de interpolação dadas pelas equações 2.44 a 2.46, quando avaliadas sobre um elemento mestre triangular como o mostrado na Figura 2.7, adquirem a forma:

$$\Psi_1(\xi,\eta) = 1 - \xi - \eta \tag{2.89}$$

$$\Psi_2(\xi,\eta) = \xi \tag{2.90}$$

$$\Psi_3(\xi,\eta) = \eta \tag{2.91}$$

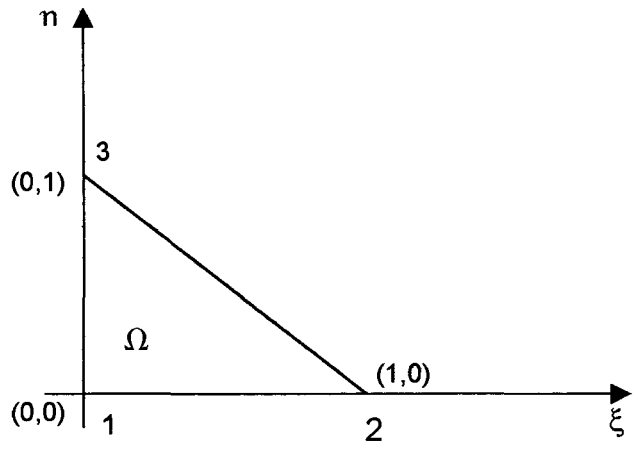


Figura 2.7 Interpolação linear no plano transformado ξ, η.

Ao considerar a equação (2.89), por exemplo, observa-se que ela vale 1 no nó número 1 (vide Figura 2.7) e 0 nos outros dois nós, o mesmo ocorrendo com a equação (2.90) que vale 1 no nó número 2 e 0 nos outros dois nós. Ou seja, as funções das equações (2.89) a (2.91) no plano (ξ, η) são equivalentes às funções representadas pelas equações (2.44) a (2.46) no plano (x, y).

OBS.: Ao utilizar funções de interpolação lineares, pode-se definir facilmente todas as transformações e consegue-se realizar a integração de forma exata (analiticamente), quando se considera k constante em cada elemento. No entanto, não será feita aqui esta demonstração, pois no código computacional desenvolvido é utilizada a integração numérica e também porque para ordens mais altas esta forma de integração não é possível.

2.9.2 Cálculos Elementares para o Triângulo Quadrático

As funções de interpolação quando avaliadas sobre um triângulo quadrático, conforme Figura 2.8, tem os seguintes valores:

$$\Psi_1(\xi,\eta) = 2(1-\xi-\eta)(1-\xi-\eta-1/2) \tag{2.92}$$

$$\Psi_2(\xi,\eta) = 2\xi(\xi-1/2) \tag{2.93}$$

$$\Psi_3(\xi,\eta) = 2\eta(\eta-1/2) \tag{2.94}$$

$$\Psi_4(\xi,\eta) = 4(1-\xi-\eta)\xi \tag{2.95}$$

$$\Psi_5(\xi,\eta) = 4\xi\eta \tag{2.96}$$

$$\Psi_6(\xi,\eta) = 4\eta(1-\xi-\eta) \tag{2.97}$$

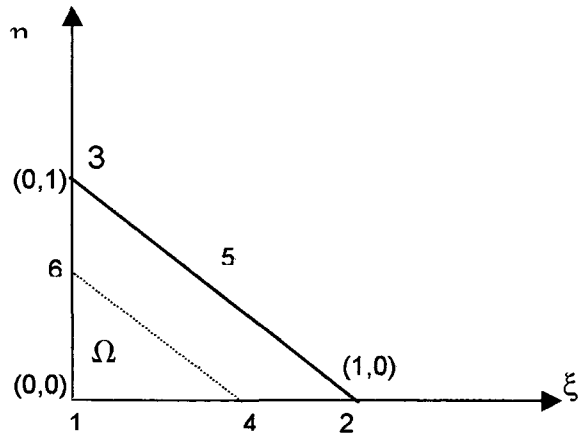


Figura 2.8 Interpolação quadrática no plano transformado ξ, η.

Igualmente ao caso dos triângulos lineares, para o caso quadrático a função de interpolação também vale 1 no seu nó e 0 em todos os demais. Ou seja, a equação (2.94), por exemplo, tem valor 1 no nó número 3 e valor 0 em todos os outros.

2.10 Coeficientes Variáveis

Para os coeficientes variáveis do problema, deve-se fazer uma aproximação desses valores. Uma primeira aproximação consiste em usar a média aritmética da função nos nós da função de interpolação, ou então, fazer:

$$k(x,y) = k_h(x,y) = \sum_{j=1}^{Ne} k_j \Psi_j^{\,e}(x,y) \tag{2.97}$$

$$b(x,y) = b_h(x,y) = \sum_{j=1}^{Ne} b_j \Psi_j^{\,e}(x,y) \tag{2.98}$$

$$f(x,y) = f_h(x,y) = \sum_{j=1}^{Ne} f_j \Psi_j^{\,e}(x,y) \tag{2.99}$$

Quando se tem um grande número de elementos, a média aritmética será uma ótima aproximação para estes coeficientes.

2.11 Condições de Contorno (CC)

Os PVCs ocorrem quando as CC. são aplicadas em diferentes pontos do domínio, tendo diferentes valores da variável independente. As EDPs sujeitas às CC. podem ou não apresentar solução. Dependendo do tipo de CC., o problema apresenta uma solução única ou mais que uma solução. Temos três tipos de condições de contorno:

(i) Condições de Dirichlet (essencial) – Prescreve o valor de uma função para CC;

(ii) Condições de Neumann (natural) – Prescreve o valor de uma derivada para CC;

(iii) Condições de Robin (mista) – Prescreve uma combinação das condições anteriores.

O programa desenvolvido prevê a possibilidade de qualquer combinação de CC. de Dirichlet e Neumann.

As CC. são especificadas após a montagem do sistema global. Com a finalidade de demonstrar como aplicar as CC., consideremos o seguinte sistema N x N, que é a forma que o sistema adquire após terem sido efetuados os cálculos.

$$\begin{bmatrix} k_{11} & k_{12} & 0 & 0 & \dots & 0 & 0 \\ k_{21} & k_{22} & k_{23} & 0 & \dots & 0 & 0 \\ 0 & k_{32} & k_{33} & k_{34} & \dots & 0 & 0 \\ 0 & 0 & k_{43} & k_{44} & \dots & 0 & 0 \\ & & & & \dots & & \\ . & . & & . & \dots & & \\ & & & & \dots & & \\ 0 & 0 & & 0 & \dots & k_{N-1,N-1} & k_{N-1,N} \\ 0 & 0 & & 0 & \dots & k_{N,N-1} & k_{NN} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ . \\ . \\ . \\ u_{N-1} \\ u_N \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \\ . \\ . \\ . \\ F_{N-1} \\ F_N \end{bmatrix} \qquad (2.100)$$

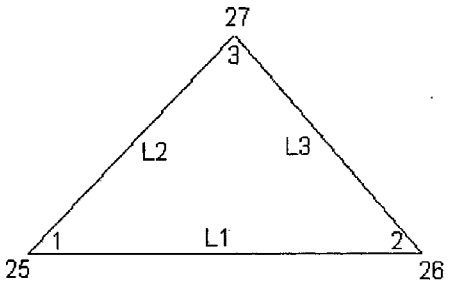
2.11.1 Condições de Contorno de Dirichlet (essenciais)

Neste tipo de condição de contorno, são conhecidos os valores de u_h nos referidos nós sujeitos a esta condição. Por exemplo, consideremos que o nó número **1** e o nó número **N** têm condições de contorno de Dirichlet. Neste exemplo, o valor de u_1 é conhecido, então a linha do sistema global, que se refere à equação algébrica de u_1, pode ser eliminada. Porém, u_1 está multiplicando a coluna 1 do sistema, logo $u_1.k_{21}$ deve ser subtraído de F_2 conforme equação (2.101). O mesmo ocorrendo em u_N.

$$\begin{bmatrix} & & & & & & \\ k_{22} & k_{23} & 0 & \dots & 0 & 0 \\ k_{32} & k_{33} & k_{34} & \dots & 0 & 0 \\ 0 & k_{43} & k_{44} & \dots & 0 & 0 \\ & & & \dots & & \\ . & . & & . & \dots & & \\ & & & \dots & & \\ 0 & & 0 & \dots & k_{N-1,N-2} & k_{N-1,N-1} \end{bmatrix} \begin{bmatrix} u_2 \\ u_3 \\ u_4 \\ . \\ . \\ . \\ u_{N-1} \end{bmatrix} = \begin{bmatrix} F_2 - k_{21}.u_1 \\ F_3 \\ F_4 \\ . \\ . \\ . \\ F_{N-1} - k_{N-1,N}.u_N \end{bmatrix} \qquad (2.101)$$

2.11.2 Condições de Contorno de Neumann (naturais)

Na imposição das CC naturais, efetua-se a integração da equação (2.88), ou seja, é realizada uma integral de linha. Para realizar os cálculos é necessário transformar a equação (2.88) em coordenadas ξ e η. Para isso é necessário identificar qual dos lados



dos triângulos do contorno representa dS. Há três possibilidades de dS, para um melhor entendimento, consideremos que a Figura 2.9 representa um elemento do contorno.

Figura 2.9 Elemento do Contorno.

Se a borda do triângulo que está no contorno for L1, tem-se que $\eta = 0$ e então:

$$\theta_j(\xi) = \Psi_j(\xi,0), \qquad j=1,2,...N_e \tag{2.102}$$

E então, a equação (2.88) é dada por:

$$S_i^e = \int_{\partial\Omega e} \tau_n \Psi_i^e ds = \int_{-1}^{1} \tau_n \theta_j(\xi) |j(\xi)| d\xi \tag{2.103}$$

Ou seja:

$$dS = \underbrace{\left[\left(\frac{\partial x(\xi,0)}{\partial \xi}\right)^2 + \left(\frac{\partial y(\xi,0)}{\partial \xi}\right)^2\right]^{1/2}}_{|j(\xi)|} d\xi \tag{2.104}$$

No entanto, se a borda do triângulo que está no contorno for L2, $\xi = 0$ e:

$$\theta_j(\eta) = \Psi_j(0,\eta), \qquad j=1,2,...N_e \tag{2.105}$$

E então, a equação (2.88) é dada por:

$$S_i^e = \int_{\partial\Omega e} \tau_n \Psi_i^e ds = \int_{-1}^{1} \tau_n \theta_j(\eta) |j(\eta)| d\eta \tag{2.106}$$

Onde:

$$dS = \underbrace{\left[\left(\frac{\partial x(0,\eta)}{\partial \eta}\right)^2 + \left(\frac{\partial y(0,\eta)}{\partial \eta}\right)^2\right]^{1/2}}_{|j(\eta)|} d\eta \tag{2.107}$$

A última possibilidade é que a borda do triângulo que está no contorno seja L3. Neste caso pode-se fazer $\eta = 1 - \xi$

$$\theta_j(\eta) = \Psi_j(\xi, 1-\xi), \qquad j = 1,2,...N_e \tag{2.108}$$

E então, a equação (2.88) é dada por:

$$S_i^e = \int_{\partial\Omega e} \tau_n \Psi_i^e ds = \int_{-1}^{1} \tau_n \theta_j(\xi) |j(\xi)| d\xi \tag{2.109}$$

Onde:

$$dS = \underbrace{\left[\left(\frac{\partial x(0,\eta)}{\partial \eta}\right)^2 + \left(\frac{\partial y(0,\eta)}{\partial \eta}\right)^2\right]^{1/2}}_{|j(\eta)|} d\eta \tag{2.107}$$

Os valores de $x(\xi, \eta)$ e $y(\xi, \eta)$ que aparecem nas equações acima foram definidos na equação (2.72).

2.12 Conclusão

Neste capítulo foi apresentado a técnica de resolução do problema proposto, utilizando o MEF. Vale a pena ressaltar a facilidade de tratar todas as irregularidades do problema, ou seja, a possibilidade de diferentes propriedades físicas contidas num mesmo domínio, a concentração de quaisquer cargas concentradas e obviamente a grande vantagem de utilização do domínio genérico. Maiores detalhes sobre as efetivas implementações, são apresentados no capítulo 5.

Capítulo 3 - Geração da Malha

3.1 Introdução

Um passo importante para a simulação numérica de um problema real é a maneira como o domínio é dividido, sendo que a precisão da solução numérica é muito influenciada pela natureza desta divisão (discretização). Especialmente em domínios complexos, há a necessidade de discretizações não estruturadas, tais como diagramas de Voronoi, Delaunay, entre outras.

A divisão do domínio em um número finito de subdomínios é feita da maneira mais simples possível, como por exemplo, triângulos ou quadriláteros em domínios bidimensionais e tetraedros, pentaedros ou hexaedros em domínios tridimensionais.

As malhas são divididas em dois grandes grupos: malhas estruturadas (Thompson, 1987) e malhas não estruturadas (Lohner e Parikh, 1988). No caso estruturado, o domínio é dividido de acordo com o sistema de coordenadas, gerando polígonos regulares para o caso bidimensional e poliedros para o tridimensional. Cada uma dessas sub-regiões é numerada de maneira sequêncial, ou seja, pode-se sempre de maneira sequêncial determinar os vizinhos de cada elemento. Já para o caso não estruturado, o domínio é geralmente dividido em polígonos irregulares para os casos bidimensionais e poliedros quaisquer nos tridimensionais. As sub-regiões formadas não são numeradas de forma sequencial e a malha gerada não está vinculada ao sistema de coordenadas. Porém, as malhas não-estruturadas oferecem uma grande flexibilidade geométrica do domínio, além de vantagens nos casos de refinamento da malha, pois, como a malha não está vinculada a um sistema de coordenadas, esta permite que novos elementos sejam adicionados ou removidos com grande facilidade, ou então que somente uma região do domínio seja refinada. No entanto, as malhas não estruturadas são menos eficientes computacionalmente, pois, devido à sua geração, é necessário um armazenamento das informações de cada elemento e sua vizinhança.

Segundo Thompson *et al.* (1999) a triangulação de Delaunay (TDEL) é um método popular de geração de malha, que surgiu quando Dirichlet em um paper de 1850

discutiu conceitos básicos de geometria e propôs um método onde, dado um domínio qualquer, este domínio pudesse ser decomposto em um conjunto de polígonos convexos.

Neste trabalho é utilizada a triangulação de Delaunay (TDEL) (Preparata e Shamos, 1988), que é bastante usada na interpolação de dados, modelagem de sólidos, em elementos finitos e outras áreas da análise numérica. Esta triangulação é a que produz os triângulos mais equiláteros possíveis. Baseado neste fato é que foi feita a escolha da referida triangulação, pois sabe-se que os erros cometidos no MEF dependem em muito da forma dos triângulos usados (Resende, 1994) e, em geral, os erros são tanto maiores quanto menos equiláteros forem os triângulos. Como exemplo de utilização da TDEL, tem-se a discretização de um automóvel (vide Figura 3.1).

Neste capítulo são descritos os conceitos básicos da TDEL. O programa de geração da TDEL que está sendo utilizado neste trabalho para domínios não-convexos foi implementado por Silva (1999), utilizando o algoritmo incremental. No entanto, nos problemas onde o domínio é convexo, utiliza-se a própria geração do *software* MatLAB. A descrição dos conceitos contidos neste capítulo são baseados em Resende e Stolfi (1994) e Preparata and Shamos (1988). Na primeira parte são descritas as propriedades fundamentais da TDEL e em seguida são discutidos alguns aspectos do reordenamento dos índices nodais, sendo que as técnicas de reordenamento têm por objetivo reduzir a largura da banda da matriz de rigidez global, acelerando a resolução do sistema de equações algébricas. É aplicado no código computacional o algoritmo de reordenamento dos índices nodais proposto por Cuthil and Mckee (Oden, 1984). O algoritmo e os resultados obtidos são apresentados neste capítulo.

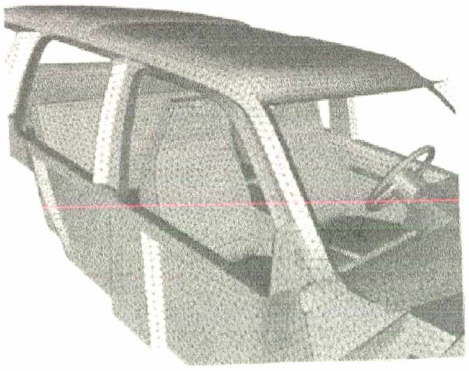


Figura 3.1 Discretização de um automóvel.
Fonte: Thompson *et al. (1999)*

Para maiores informações sobre a implementação da TDEL, referência é feita a Silva (1999), sendo que o fluxograma representado na Figura 3.2 resume o algoritmo naquele trabalho implementado.

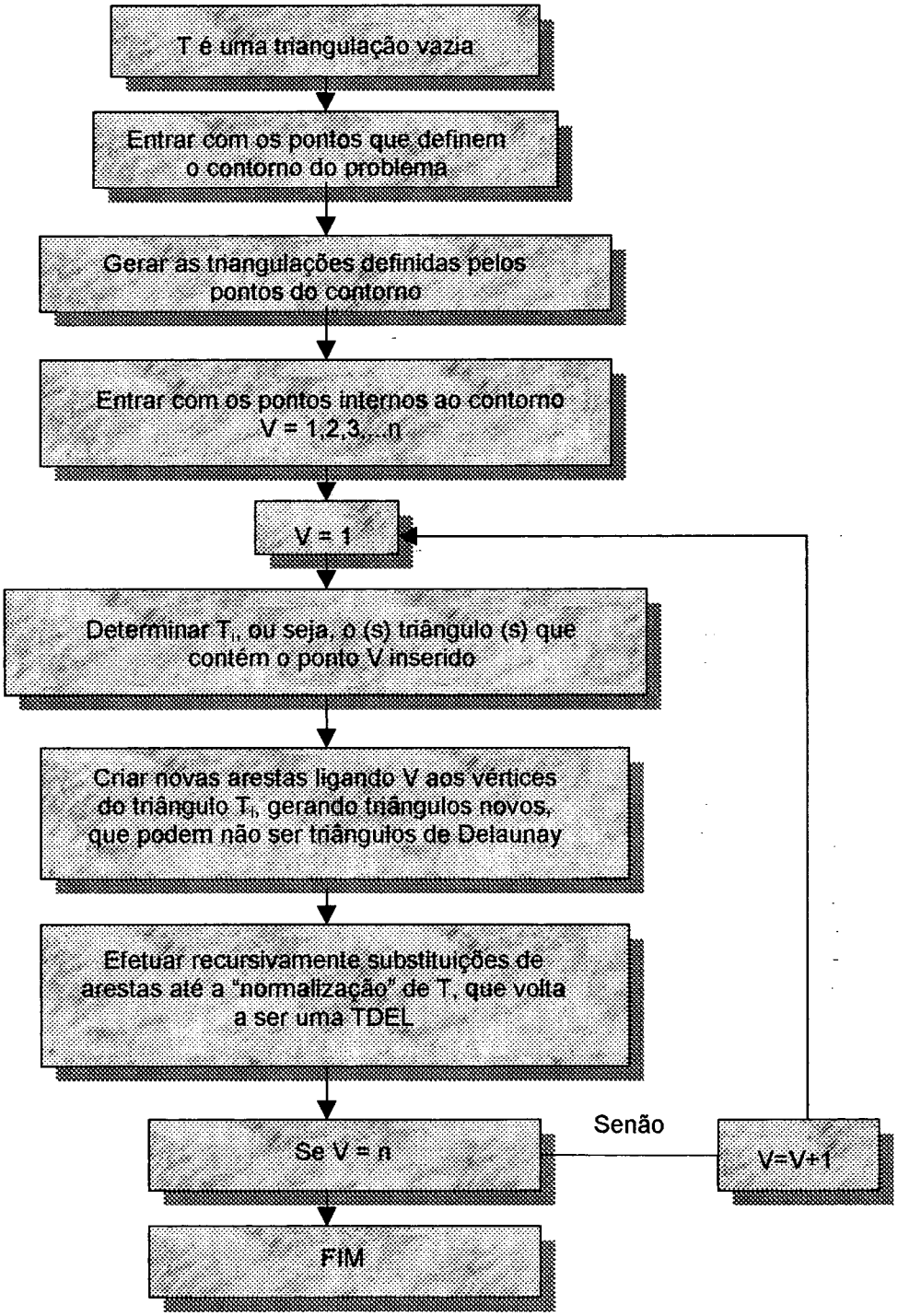


Figura 3.2 Algoritmo proposto para a geração da TDEL.

3.2 Conceitos Fundamentais da TDEL

Conforme pode-se observar na Figura 3.2, na primeira etapa da TDEL são informados os pontos do contorno, no sentido anti-horário. Em seguida é gerada uma primeira triangulação somente com estes pontos, sendo que os vértices são ligados somente nos casos em que o segmento de reta que os une não passa por uma região fora do contorno, conforme ilustrado na Figura 3.3. Em seguida, passa-se para a etapa de inserção de um novo vértice à triangulação. Nos próximos tópicos são apresentados alguns conceitos básicos para o entendimento da TDEL.

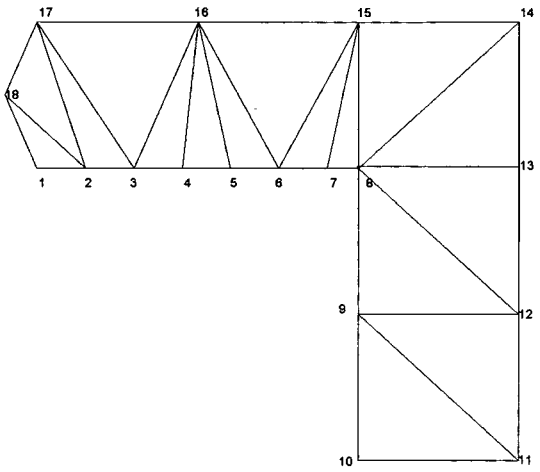


Figura 3.3 Geração da triangulação a partir dos pontos do contorno.

3.2.1 Ligação de um novo vértice

Na primeira etapa do processo de inserção de um novo vértice, é necessário determinar em qual triângulo ele está contido. Este processo é realizado através do teste de sentido.

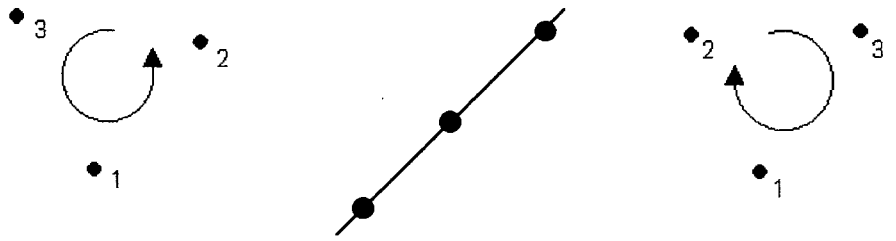
3.2.1.1 Teste de Sentido

Além de encontrar o triângulo em que o novo vértice está contido, o teste de

sentido também tem o objetivo de manter a orientação básica nos processos de caminhamento dos vértices durante a triangulação. Para isso pode-se usar a definição de pontos colineares, que é definido por Rezende e Stolfi (1994) da seguinte forma: "Dizemos que três pontos são colineares se eles pertencem a mesma reta". Em geometria cartesiana, prova-se que três pontos $P_i = (x_i, y_i)$, representados pela função $S(P_1,P_2,P_3)$ são colineares se e somente se:

$$S = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} = 0 \qquad (3.1)$$

Ou seja, eles são colineares se o determinante da matriz S for igual a zero (conforme Figura 3.4b). Conseqüentemente, o sinal de S é utilizado para determinar o sentido dos pontos. Quando $S > 0$, os três pontos na seqüência P_1, P_2, P_3 definem um ciclo no sentido anti-horário (vide Figura 3.4a). Se $S < 0$, os pontos estão em sentido horário, conforme Figura 3.4c.



(a) (b) (c)

Figura 3.4 Testes de sentido.

Observa-se que o intercâmbio de dois pontos troca o sinal de S, ao passo que a "rotação" dos argumentos mantém o sinal:

$$S(A, B, C) = S(B, C, A) = S(C, A, B) = -S(A, C, B) = -S(C, B, A) = -S(B, A, C)$$

Resumindo, nesta etapa deve ser realizado um procedimento de busca exaustiva, fazendo o teste de sentido em todos os novos triângulos formados com o novo ponto. Este teste é realizado em relação às coordenadas (x, y) do novo ponto a ser inserido, que é testado com dois dos três pares de vértices consecutivos de cada triângulo existente, sendo que quando é obtido nos três testes de S (equação (3.1)) o mesmo sinal, significa

que o ponto está contido neste triângulo (ver Figura 3.5).

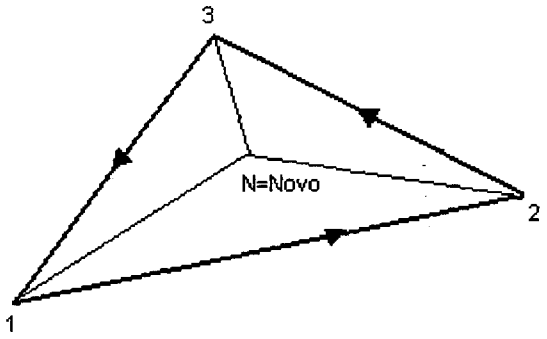


Figura 3.5 Novo vértice inserido.

3.2.2 Proximidade Inicial e Ligação do Novo Vértice

Após identificar em qual triângulo o novo ponto inserido está, procede-se a primeira etapa de modificação da triangulação, sendo que o ponto dado passa a ser um novo vértice e os vértices próximos a ele são ligados por arestas. Essas arestas passam a denotar as relações de vizinhança entre os vértices por elas ligadas (Figura 3.6).

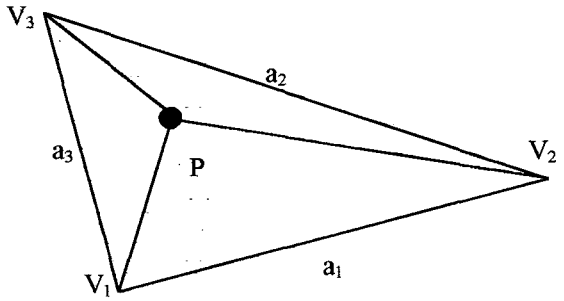


Figura 3.6 Ponto no interior do triângulo.

Dois vértices são considerados próximos quando é possível traçar pelo menos uma circunferência que passe por eles e não contenha qualquer outro vértice em seu interior conforme ilustrado na Figura 3.7. Três vértices mutuamente próximos definem uma circunferência que não possui nenhum outro vértice em seu interior (Figura 3.8). Se formar um ângulo > 90° entre um dos pares de arestas do triângulo, o centro da circunferência está fora do triângulo, conforme Figura 3.8b.

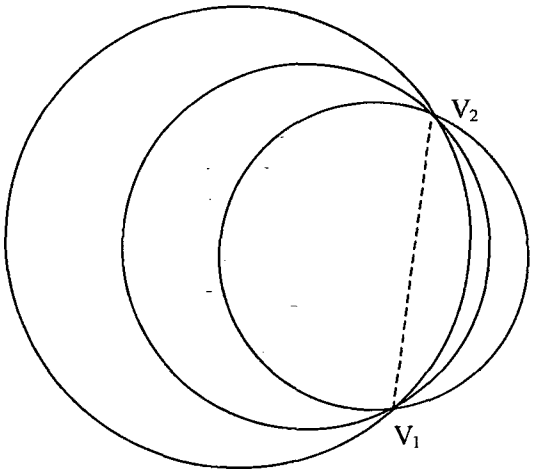


Figura 3.7 Dois vértices próximos.

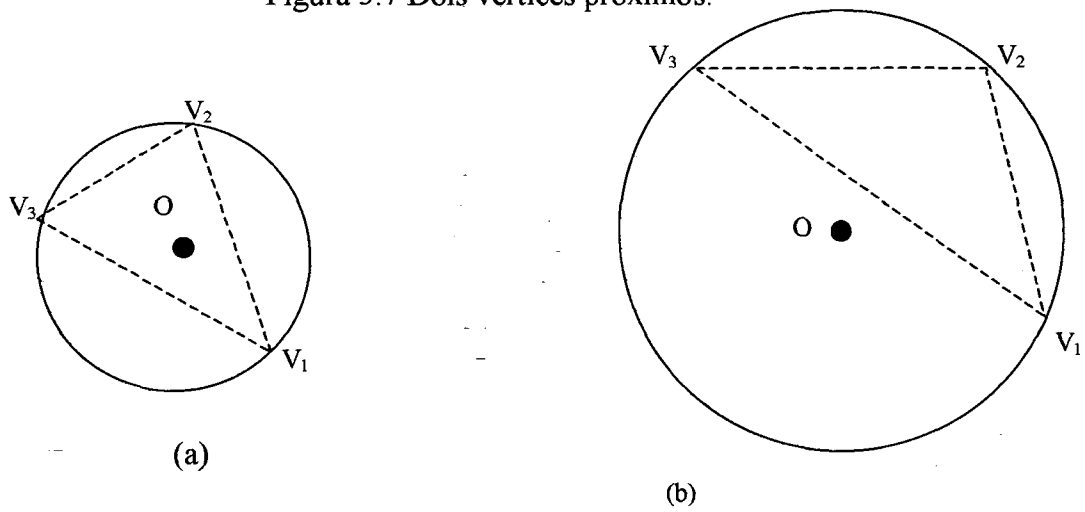


Figura 3.8 Três vértices próximos.

Um novo vértice P colocado no interior de um triângulo existente certamente é próximo aos três vértices deste triângulo. Sempre é possível definir circunferências tangentes à circunferência definida pelos vértices do triângulo existente, que passam por um dos vértices deste triângulo e pelo novo vértice. Essas circunferências estão inteiramente contidas na circunferência maior (vide Figura 3.9).

A inclusão de um novo vértice pode fazer com que outros vértices que eram próximos entre si deixem de sê-lo. Também, o novo vértice pode ser próximo a outros vértices além daqueles determinados inicialmente. Na etapa seguinte, são explicados os meios para restabelecer as relações de proximidades entre os vértices.

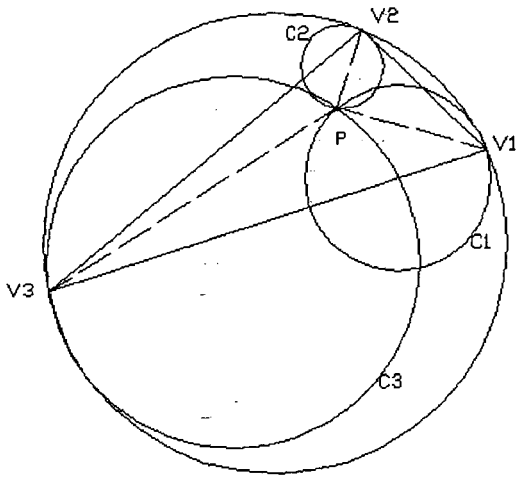


Figura 3.9 Vértices internos e próximos.

3.2.3 Ajustes no mapa

Depois da inclusão do novo vértice e das novas arestas, é necessário verificar se esta inclusão influiu nas proximidades entre os vértices vizinhos. Numa TDEL, as circunferências definidas pelos vértices de cada triângulo nunca contêm vértices em seus interiores. Entretanto, pode ocorrer que o novo vértice esteja situado dentro de uma dessas circunferências. Pode acontecer também que as circunferências definidas pelos novos triângulos acrescentados à triangulação passem a conter outros vértices já existentes na triangulação. Devido a tais situações, deve ser considerado que, embora as arestas acrescentadas expressem a proximidade entre o novo vértice e os três vértices vizinhos imediatos, podem existir outras arestas que deverão ser substituídas para que a triangulação existente volte a ser uma TDEL.

Na Figura 3.6 foi acrescentado um vértice e algumas arestas novas ao domínio. Nesta etapa, o número de arestas é mantido, isto é, ter-se-á apenas substituições de arestas numa etapa de ajustes.

Consideremos, por exemplo, a situação ilustrada na Figura 3.10. Nela, o vétice **P** foi incluído no interior do triângulo TQS. Foram criadas três novas arestas: **PT**, **PQ** e

PS. Como conseqüência, o triângulo TQS deixou de existir, dando lugar aos triângulos PTQ, PQS e PST. Nota-se agora que a circunferência que passa pelos vértices do novo triângulo PQS contém R em seu interior. Assim, será necessário fazer algum ajuste nesta triangulação para que ela volte a ser uma TDEL. Estes ajustes são feitos com base em critérios capazes de determinar se um ponto dado está no interior de uma circunferência definida por outros três pontos. Então, deve-se estabelecer um critério com o propósito de validar triangulações de quatro vértices.

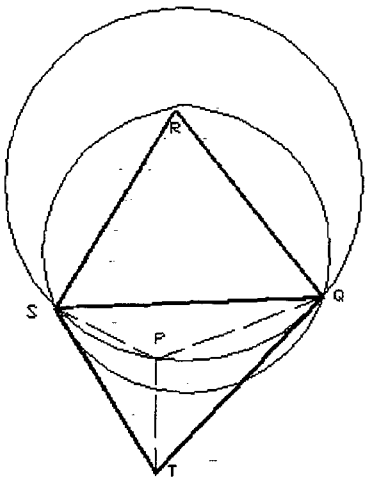


Figura 3.10 Triangulação que requer ajustes.

3.2.4 Critério de interioridade

Considere os pontos Q, R e S definindo a circunferência C em sentido anti-horário (vide Figura 3.11). Se o determinante da equação (3.2) for negativo, o ponto P está no interior da circunferência C.

$$H = \begin{vmatrix} 1 & x_Q & y_Q & (x_Q^2 + y_Q^2)/2 \\ 1 & x_R & y_R & (x_R^2 + y_R^2)/2 \\ 1 & x_S & y_S & (x_S^2 + y_S^2)/2 \\ 1 & x_P & y_P & (x_P^2 + y_P^2)/2 \end{vmatrix} \qquad (3.2)$$

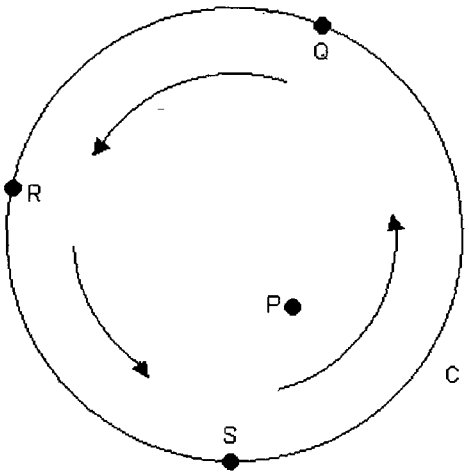


Figura 3.11 Ponto no interior da circunferência.

Então, para o quadrilátero da Figura 3.12, por exemplo, a triângulação de Delaunay é formada com o acréscimo de uma nova aresta entre os vértices Q e S.

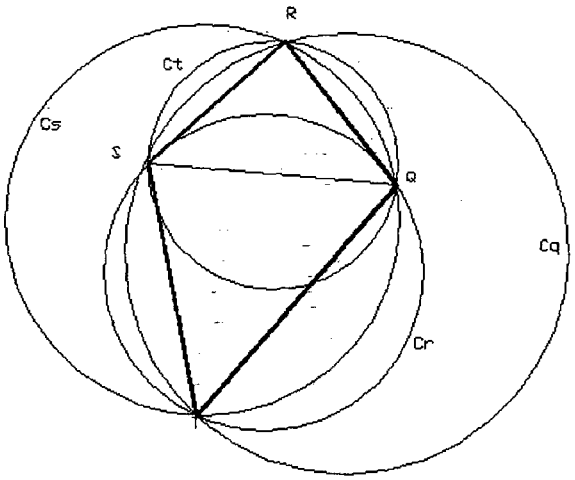
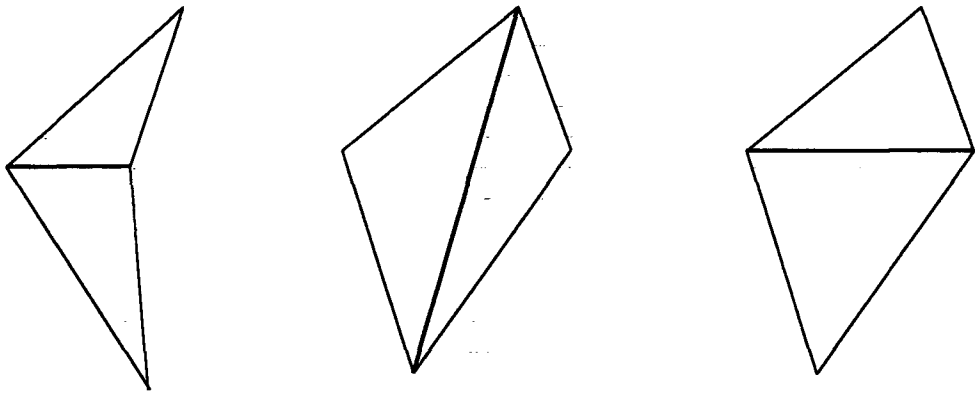


Figura 3.12 QS é uma aresta de Delaunay.

Para adaptar os resultados obtidos acima ao algoritmo, pode-se definir que cada aresta de qualquer triangulação, não pertencente ao contorno, define um quadrilátero com os dois triângulos vizinhos. Esse quadrilátero pode assumir três configurações básicas, como mostra a Figura 3.13. No caso de um quadrilátero não convexo (Figura 3.13a), a aresta válida é sempre a que fica em seu interior. Num quadrilátero convexo

com uma aresta inválida, conforme Figura 3.13b, efetua-se uma substituição desta aresta por outra que une os dois outros vértices, obtendo-se uma aresta válida (vide Figura 3.13c). Ao se tomar uma triangulação qualquer, faz-se sucessivos ajustes em seus quadriláteros até que não existam mais "arestas inválidas", obtendo assim uma TDEL.



(a) (b) (c)

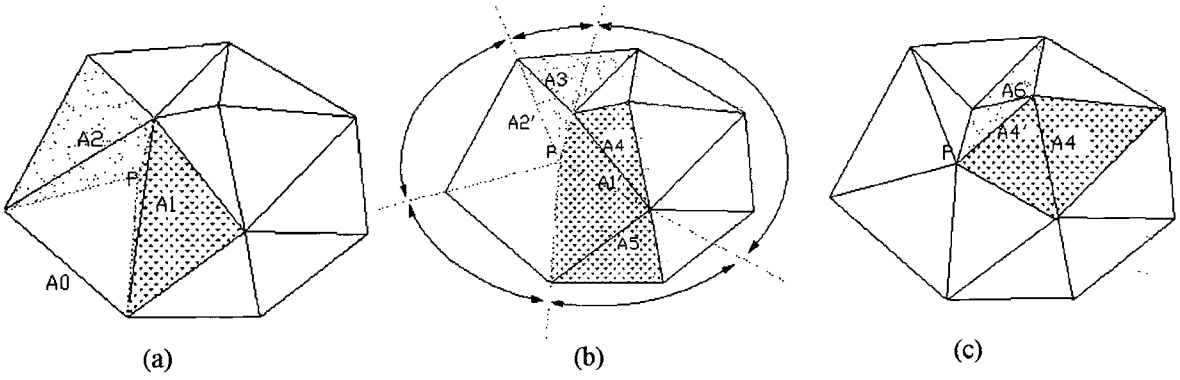
Figura 3.13 Os três quadriláteros possíveis.

Entretanto utilizando a técnica incremental do algoritmo proposto, é possível reduzir o número de quadriláteros a serem testados após a inclusão de cada vértice.

3.2.5 Ajustes na Região Próxima a um Novo Vértice

Agora é possível vizualizar a execução de uma etapa do algoritmo geral. Para tal, é utilizado um exemplo onde ocorrem sucessivas substituições. Consideremos a triangulação da Figura 3.14a, onde um novo vértice P foi incluído e ligado aos vértices próximos (linhas finas). Da lista de vértices próximos se obtém uma lista de arestas que, por sua vez, leva a uma lista inicial de quadriláteros a serem testados e ajustados.

A aresta a_0 não define um quadrilátero, pois está no contorno. As arestas a_1 e a_2 definem quadriláteros formados por dois vizinhos, mas são ambas as arestas inválidas. Na Figura 3.14b, a_1 e a_2 foram substituídas por a_1' e a_2'. Com a substituição por a_2', tem-se um novo quadrilátero definido por a_3' a ser testado (o outro não é necessário testar, pois está no contorno). Na substituição por a_1', tem a_4 e a_5 para testar e ajustar.



(a) (b) (c)

Figura 3.14 Sequência de ajustes após a inclusão de um novo vértice.
Fonte: Labriola (1994).

A aresta a_3 representa um quadrilátero não convexo e a_5 é válida. Com a substituição de a_4 por a_4', ilustrada na Figura 3.14c, surgem mais dois quadriláteros a serem testados. Como as arestas a_6 e a_7 são válidas e não há mais nenhum quadrilátero a ser ajustado, a etapa de inclusão de um vértice termina e a triangulação volta a ser de Delaunay.

Um aspecto interessante a observar durante a execução da etapa descrita acima é que, a cada substituição de aresta dentro de um quadrilátero, a nova aresta passa a ser incidente ao novo vértice, de modo que a região em torno do vértice vai sendo subdividida em "fatias polares", como aparece na Figura 3.14b. Cada uma dessas fatias pode ser tratada de forma independente, sem interferir no tratamento das fatias vizinhas. Assim, as fatias podem ser tratadas em qualquer ordem, fato que permite, por exemplo, o uso de uma pilha para armazenar os quadriláteros a serem tratados.

Na eventualidade do quadrilátero de uma fatia "invadir" a fatia vizinha, como acontece com a_3 na Figura 3.14b, este quadrilátero não será convexo. Assim, a fatia na qual ele se encontra não necessitará de qualquer tratamento subsequente.

Identifica-se cada fatia pela aresta que está "em frente" ao novo vértice, tendo quatro casos a tratar:

- o novo vértice faz parte da periferia do mapa e a fatia correspondente ao exterior do mapa, sem aresta para identificá-la. Nenhum tratamento é necessário;
- a aresta faz parte da periferia do mapa e não define um quadrilátero. Nenhum

tratamento é necessário;

- a aresta define um quadrilátero convexo ou não é válida. Nenhum tratamento é necessário;
- a aresta é inválida. Após sua substituição, a fatia é dividida em duas outras a serem tratadas recursivamente usando esta mesma metodologia.

O tratamento de cada fatia é finito. Uma aresta válida denota o fim da "área de influência" do novo vértice naquela fatia.

3.2.6 Situações especiais

Até aqui, foram descritos os conceitos básicos para o funcionamento do algoritmo considerando que a lista de pontos a ele fornecida não contivesse quaisquer casos de coincidências, colinearidades, ou cocircularidades. Isto facilitou bastante a descrição, embora tais casos ocorram com certa frequência na prática. Uma malha ortogonal retangular, por exemplo, apresenta casos de colinearidade e cocircularidades.

Além disso, a descrição do algoritmo foi feita de modo a lidar com variáveis reais, de precisão ilimitada, sem considerar situações anormais, decorrentes dos arredondamentos que surgem quando se escolhe uma forma limitada para representar essas variáveis reais. Esses fatores devem ser considerados posteriormente de maneira a tornar a implementação do algoritmo estável e robusta, capaz de tratar qualquer lista de pontos a ele fornecida.

3.2.6.1 Erros numéricos

Uma parte do algoritmo merece consideração especial com relação a erros numéricos: é o teste de sentido. Este teste deve ser utilizado com o ponto a ser testado sempre colocado como o terceiro argumento. Além disso, sua implementação deve ser feita de modo a colocar em evidência as coordenadas deste ponto de teste.

Tal precaução faz com que o teste de sentido para um ponto de teste quase

colinear a uma aresta **a** resulte sempre num mesmo valor com sinal oposto ao obtido num teste com a aresta oposta a **a** e o mesmo ponto, isto é, $S(P_1, P_2, P_3) = -S(P_2, P_1, P_3)$, mesmo que o sinal esteja errado. Assim, evita-se a busca interminável por um ponto no mapa.

3.2.6.2 Três vértices colineares

Tanto durante o processo de localização de um ponto dentro do contexto do mapa como durante o processo de ligações e ajustes, podem surgir situações onde a função S zera, denotando colinearidade entre uma aresta e o ponto de teste. Em alguns casos, os tratamentos de uma das condições resultantes na avaliação de S, consistem na substituição de $S > 0$ e $S < 0$ por $S \geq 0$ ou $S \leqslant 0$.

Em outros casos, entretanto, as colinearidades requerem tratamento especial e a função S não é mais adequada para fazer esse tratamento, já que ela só vai dar zero. Assim, definimos a função $C(P_1, P_2, P_3)$ que calcula o produto escalar entre os vetores P_1P_2 e P_1P_3. Com ela é possível saber se P_1 está entre P_2 e P_3 ou não.

3.2.6.3 Quatro vértices cocirculares

O tratamento de cocircularidades não requer maiores cuidados por dois motivos. Em primeiro lugar, um quadrilátero cocircular (Figura 3.15) é válido em suas duas formas; assim, quando H=0, pode-se dispensar a troca da aresta central. Em segundo lugar cada quadrilátero é testado no máximo uma vez durante a inclusão de cada vértice. Assim, não existe o risco do algoritmo entrar numa sequência de ajustes em torno de um mesmo quadrilátero.

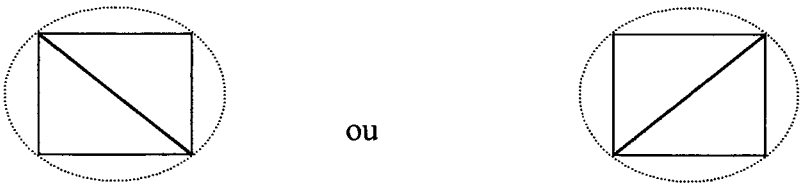


Figura 3.15 Quadrilátero cocircular.

3.3 Reordenamento dos Índices Nodais

Dependendo do método escolhido para a resolução do sistema Ax = b, o reordenamento dos números dos nós tem papel fundamental no esquema de resolução. No código computacional desenvolvido, foi escolhida a eliminação Gaussiana como método de resolução, sendo exploradas as características de banda da matriz A. Portanto, esta é uma importante etapa do programa.

Segundo Carey and Oden (1984), historicamente a primeira implementação da eliminação Gaussiana para explorar a esparsividade da matriz A (considerando o sistema Ax = b) foram os métodos de banda de Martin and Wilkinson (1965), que foi baseada na observação de que os elementos não nulos da matriz são freqüentemente armazenados em um pequeno número de diagonais em volta da diagonal principal. Então diversos algoritmos foram desenvolvidos para explorar esta característica. O MEF é usado para domínios complexos, os quais conduzem a uma variação local do tamanho de banda devido à numeração nodal. Estes casos com uma grande variação no tamanho de banda chamados "Envelope Methods" são estudados com o objetivo de reduzir este problema.

Para obter um sistema Ax = b e resolvê-lo explorando as características de banda e esparsividade é necessário a Permutação e o Ordenamento dos índices nodais. Ou seja, quando a matriz A não concorda com um método particular, é freqüentemente possível renumerar as equações, ou seja, reordenar as linhas e as colunas de A, para obter uma nova matriz que é mais compatível com o método. Usar uma nova matriz é equivalente a resolver a permutação do sistema linear seguinte:

$$(PAQ^T)\,(Qx) = Pb \tag{3.3}$$

Onde P e Q são as matrizes booleanas permutadas, ou seja, a permutação matricial P é a matriz de zeros e uns que satisfaz a:

$$PP^T = P^TP = I \tag{3.4}$$

Quando A é simétrica e positivo-definida (como nos casos que o código computacional resolve), há a vantagem de que P = Q. No entanto, é difícil encontrar

uma boa permutação da matriz P para uma dada matriz A. Entretanto, para cada método de reordenamento são elaborados procedimentos heurísticos que, embora não garantam boas escolhas de P, são bastante eficientes na prática. A maior parte destes procedimentos veêm o problema de encontrar P como um problema num grafo G(A) e modelam estruturas de zeros para A.

O algoritmo Cuthill-Mckee é um algoritmo de reordenamento clássico e é baseado na teoria de grafos. Assim sendo, este algoritmo é aplicado no código computacional desenvolvido. Outro algoritmo muito aplicado é o Reverse Cuthill-Mckee, que não é aplicado ao programa porque Liu and Sherman (1976) provaram que para o caso que está sendo resolvido (matriz esparsa, simétrica e positivo-definida) os dois são equivalentes. No entanto, para um melhor entendimento do algoritmo, alguns conceitos básicos da teoria de grafos são necessários, os quais são citados em seguida, para maiores informações sobre esta teoria, consultem Rabuske (1992).

3.3.1 Definição de Grafo

Um grafo G é definido como sendo um par ordenado (V,E), onde V é um conjunto e E é uma relação binária sobre V. Ou seja, os elementos de V são denominados de vértices ou pontos ou nós, e os pares ordenados de E são denominados arestas ou linhas ou arcos de grafo.

3.3.2 Definição de Pontos Adjacentes

Dois vértices são adjacentes se eles estão ligados por uma aresta.

3.3.3 Definição de Grau do elemento

O grau de um vértice é dado pelo número de vértices adjacentes ao vértice em análise.

3.3.4 Algoritmo Cuthill- Mckee (CM)

Para aplicação do algoritmo, assume-se que os vértices de cada elemento estão em ordem crescente de grau (Adj(v_i)). I é um vetor que contém uma lista de todos os elementos; N é o número de elementos; v é um vetor com os vértices na nova ordem. O algoritmo pode ser visto na Figura 3.16.

```
Enquanto k < N
        Se i <= N
                vi = v(i)
                Para j = 1 até nº. vértices adjacentes de v(i) Faça
                        w = vértice_adjacente (vi, j)
                        Se w ∈ I
                                k = k + 1
                                v(k) = w
                                I = I – {w}
                        Fim
                Fim
                i = i + 1
        Senão
                k = k + 1
                v(k) = vértice de menor grau que ∈ I
                I = I – {v(k)}
        Fim
Fim
```

Figura 3.16 Algoritmo Cuthil-Mckee (CM).

Consideremos, por exemplo, que através da entrada de dados para a resolução de um determinado problema, é gerada uma malha conforme Figura 3.17.

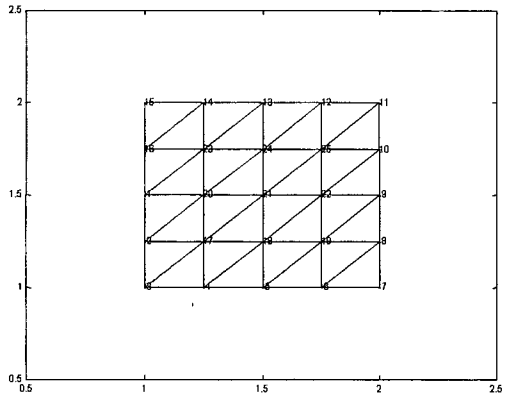


Figura 3.17 Numeração nodal antes da aplicação do CM.

Utilizando os pontos da malha da Figura 3.17 e aplicando o algoritmo CM, é gerado um reordenamento nodal conforme a Figura 3.18.

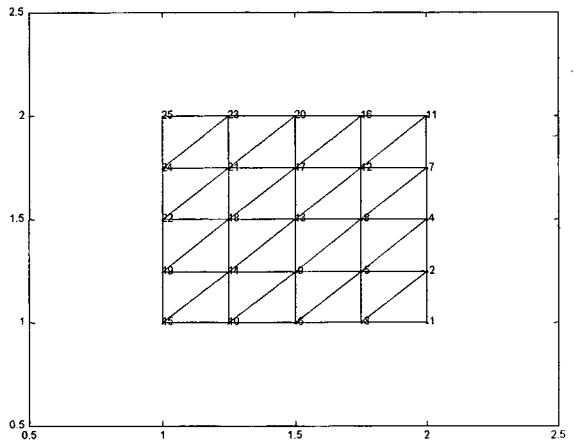


Figura 3.18 Numeração nodal após a aplicação do CM.

Com a finalidade de apresentar resultados numéricos da aplicação do algoritmo CM, foi aplicado o algoritmo ao seguinte problema:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{x}{y} + \frac{y}{x} \tag{3.5}$$

O problema está sujeito as seguintes condições de contorno

$u(x, 1) = x \ln x$ $\quad u(x,2) = x \ln(4x^2)$ $\quad$ para $\quad 1 \leq x \leq 2$ (3.6)

$u(1,y) = y \ln y$ $\quad u(2,y) = 2y \ln(2y)$ $\quad$ para $\quad 1 \leq y \leq 2$ (3.7)

O tempo de processamento obtido para a resolução do sistema de equações algébricas é apresentado na Tabela 3.1, para duas malhas, com 128 e 512 elementos respectivamente:

Tabela 3.1 Tempo de Processamento Algoritmo Cuthill-Mckee.

Número de Elementos	Tempo Processamento (s) Sem Ordenação	Tempo Processamento (s) aplicando o CM
128	0.44	0.44
512	5.55	5.55

Observa-se pelos dados da Tabela 3.1 que não houve um menor tempo de processamento aplicando o algoritmo CM, neste caso. Isso ocorreu devido a que este problema está sujeito apenas à condições de contorno de Dirichlet, sendo que no código computacional os nós com este tipo de condição de contorno são eliminados do sistema global. No entanto, se o problema estivesse sujeito à condições de contorno de Neumann, os nós do contorno não seriam eliminados, aumentando assim a largura de banda e conseqüentemente o tempo de processamento. Um exemplo deste fato pode ser observado na Figura 3.17, onde a largura de banda é determinada pelo triângulo de índices nodais igual a 22, 10 e 25, ou seja, tem-se uma largura de banda igual a 15. Aplicando o algoritmo CM é obtida a malha com a numeração nodal da Figura 3.18, observa-se nesta Figura que o triângulo de índices nodais igual a 8, 7 e 12 é que determina a largura de banda, que será neste caso igual a 5.

3.4 Conclusão

Neste capítulo foi apresentada a fundamentação teórica da geração de malhas, dando ênfase na Triangulação de Delaunay. Foram apresentados todos os conceitos fundamentais para a construção de um algoritmo utilizando esta técnica. Para obtenção de uma visão mais prática sobre a implementação consultem os trabalhos de Preparata and Shamos (1988), Silva (1999) e Silva (1999).

Quanto ao reordenamento dos índices nodais, foram encontrados na literatura diversos trabalhos, os quais foram referenciados no Capítulo 1, sendo que foi implementado somente uma técnica de reordenamento. Esta é uma área importante dentro do MEF, mas não é o objetivo principal deste trabalho. Segundo Liu and Shermann (1976), na aplicação do algoritmo CM, nunca é gerado um envelopamento

(largura de banda) maior do que o original, por esse motivo é que foi apresentado os resultados somente para um problema teste.

Capítulo 4 – Procedimentos Numéricos e Estimativas de Erros

4.1 Introdução

Como em métodos numéricos são realizadas diversas aproximações, em cada etapa de resolução do problema é gerado um erro. O primeiro erro ocorre na aproximação do domínio, onde um domínio real é aproximado por pontos em coordenadas x, y e para a ligação de dois pontos são utilizadas interpolações, no caso deste trabalho, lineares ou quadráticas. Outro erro ocorre na determinação da matriz elementar (K_{ij}^e) representada pela equação (2.86), onde é realizada uma integração numérica, e assim sucessivamente nas etapas da aproximação. O MEF possui uma excelente vantagem que é o contole do erro que se comete em cada aproximação.

Neste capítulo é descrita a metodologia de solução de algumas etapas do problema proposto, começando pela integração numérica e em seguida são discutidos alguns aspectos da resolução de sistemas lineares e por fim é feita uma análise de estimativas de erros, onde é mostrada a teoria matemática conhecida sobre a Equação de Poisson, onde sob certas condições pode-se determinar qual o grau das funções de interpolação que se deve utilizar para um determinado problema.

4.2 Integração Numérica

Para a obtenção dos valores de K_{ij}^e e F_i^e, representadas respectivamente pelas equações (2.86) e (2.87) é necessário efetuar a integração. Devido à complexidade dos termos existentes nestas equações, não se consegue resolvê-las por métodos analíticos, para funções de interpolação de ordem maior que linear. Então, tradicionalmente no MEF é utillizada a integração Gaussiana para a resolução de tais problemas (Raizer, 1987).

A escolha das fórmulas de quadratura para integrandos polinomiais pode ser realizada de forma a conduzir a uma integração exata, sendo que o método fornece

integrais exatas para polinômios de ordem 2*N-1, onde N é o número de pontos de integração.

Com a finalidade de descrever a quadratura gaussiana, consideremos os elementos finitos como sendo quadriláteros. Segundo Carey and Oden (1981), as regras de quadratura para elementos quadriláteros são derivadas das fórmulas de quadratura unidimensional, tratando-se a integração sobre o elemento mestre como uma integral dupla, ou seja, considerando o integrando da equação (2.86) como uma função $G(\xi, \eta)$, tem-se:

$$\iint_{\Omega} G(\xi,\eta)d\xi d\eta \cong \int_{-1}^{1}\left[\int_{-1}^{1} G(\xi,\eta)d\xi\right]d\eta \tag{4.1}$$

Aproximando as duas integrais em relação a ξ e η, pela regra de quadratura unidimensional, tem-se:

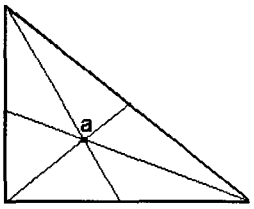
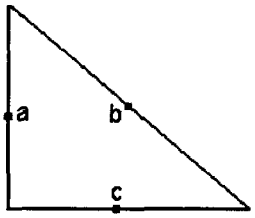
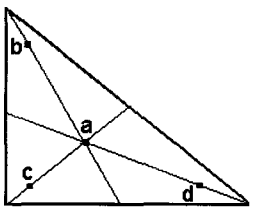
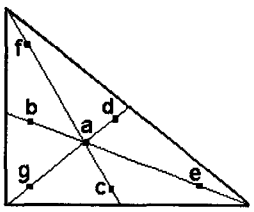
$$\iint_{\Omega} G(\xi,\eta)d\xi d\eta \cong \sum_{m=1}^{N}\left[\sum_{n=1}^{N} G(\xi_n,\eta_m)w_n\right]w_m = \sum_{l=1}^{N^2} G(\xi_l,\eta_l)\overline{w}_l \tag{4.2}$$

Onde w_n e w_m são os pesos da regra da quadratura Gaussiana e ξ_n e η_m são os valores nos pontos de integração. Para resolver a integração utilizando a equação (4.2), basta definir para o elemento mestre os pontos de integração e os respectivos pesos de Gauss.

No código computacional implementado são utilizados triângulos, com a possibilidade de se efetuar a integração utilizando as seguintes ordens de integração: quadrática, cúbica e quíntica, as quais representam respectivamente a utilização de 3, 4, 7 pontos de Gauss. Na Tabela 4.1 são apresentados os pontos de integração com os respectivos pesos para elementos triangulares. Nesta tabela também consta a ordem do erro da aproximação e os pontos para integração até ordem quíntica. Para maiores detalhes sobre como realmente os cálculos são executados, consultem o capítulo 6 (Descrição do Programa).

Tabela 4.1 Formulário de Integração Numérica para Triângulos

Fonte: Zienkiewicz (1994).

Ordem	Figura	Erro	Pontos de Integração	Coordenadas Triangulares	Pesos
Linear		R= $O(h^2)$	a	ζ_1 ζ_2 ζ_3 1/3, 1/3, 1/3	1
Quadrática		R= $O(h^3)$	a b c	ζ_1 ζ_2 ζ_3 1/2, 1/2, 0 0, 1/2, 1/2 1/2, 0, 1/2	1/3 1/3 1/3
Cúbica		R= $O(h^4)$	a b c d	ζ_1 ζ_2 ζ_3 1/3, 1/3, 1/3 0.6, 0.2,0.2 0.2, 0.6, 0.2 0.2, 0.2, 0.6	-27/48 25/48 25/48 25/48
Quíntica		R= $O(h^6)$	a b c d e f g	ζ_1 ζ_2 ζ_3 1/3, 1/3, 1/3 $\alpha_1, \beta_1, \beta_1$ $\beta_1, \alpha_1, \beta_1$ $\beta_1, \beta_1, \alpha_1$ $\alpha_2, \beta_2, \beta_2$ $\beta_2, \alpha_2, \beta_2$ $\beta_2, \beta_2, \alpha_2$	0.2250000000 0.1323941527 0.1323941527 0.1323941527 0.1259391805 0.1259391805 0.1259391805
				Com: $\alpha_1 = 0.0597158717$ $\beta_1 = 0.4701420641$ $\alpha_2 = 0.7974269853$ $\beta_2 = 0.1012865073$	

Na Tabela 4.1 os pontos de integração são dados para ζ_1, ζ_2, ζ_3, onde:

$$\zeta_1 = 1 - \xi - \eta \tag{4.3}$$

$$\zeta_2 = \xi \tag{4.4}$$

$$\zeta_3 = \eta \tag{4.5}$$

Ou seja, os pontos utilizados nas integrações são os referentes e $\zeta 1$ e $\zeta 2$ que indicam diretamente os valores para ξ e η respectivamente.

4.3 Sistemas Lineares

Conforme exposto em capítulos anteriores, após a discretização da ED, é formado um sistema global de equações algébricas, o qual representa a solução do problema em análise. A resolução do sistema de equações algébricas é de grande importância para a solução. Neste processo é requerido um alto custo de processamento, pois os sistemas gerados pelo MEF são geralmente esparsos.

Os métodos usados para resolver os sistemas de equações lineares podem ser diretos ou iterativos. Mariani (1997) apresenta o seguinte quadro de características dos métodos diretos e iterativos.

Características	Métodos Diretos	Métodos Iterativos
Operações	Sequência Recursiva	Sequência Iterativa
Matriz de Coeficiente	É modificada gerando "fill-ins" e erros de arredondamento	É preservada
Número de Operações	Finito	Dependem de um critério de parada
Aplicado em sistemas	De médio porte	De grandes dimensões e esparsos
Melhorar a performance e precisão	Através de ordenações nos elementos da matriz e refinamento iterativo.	Com coeficientes de relaxação apropriados, pré-condicionamento e ordenações das variáveis.

Em função das características apresentadas no quadro acima, tem crescido o

interesse pelos métodos iterativos. Isto ocorre devido ao desenvolvimento crescente de técnicas eficientes de aceleração, conhecidas como pré-condicionadores, que podem melhorar a velocidade de convergência, encontrando a solução em menor número de iterações e tempo de processamento (Angeleri *et al.*, 1989 *apud* Mariani, 1997). No entanto, nos métodos diretos sempre é garantida a convergência da solução. Então, no próximo item é analisada a estrutura da matriz obtida para escolher o melhor método de resolução.

4.3.1 Armazenamento das Matrizes

Na técnica do MEF, para cada elemento da malha é calculada uma matriz, chamada de matriz elementar ou matriz de contribuições. Esta matriz, depende diretamente do tipo de elemento utilizado. Para o caso dos triângulos lineares é uma matriz simétrica de 3 X 3 e, para o caso dos triângulos quadráticos uma matriz simétrica 6 X 6. A matriz de rigidez global é formada pela soma das matrizes elementares. A título de exemplo, consideremos a Figura 4.1, como sendo parte de uma malha.

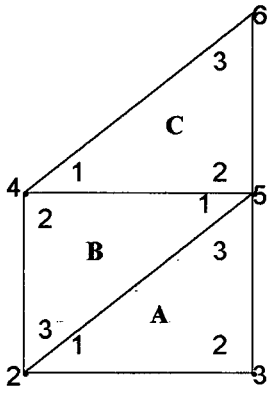


Figura 4.1 Parte de uma malha.

Pode-se observar na Figura 4.1 que os triângulos A, B e C tem os nós numerados da seguinte forma:

Elemento A	Elemento B	Elemento C
1 - 2	1 - 5	1 - 4
2 - 3	2 - 4	2 - 5
3 - 5	3 - 2	3 - 6

Logo, na matriz global, as matrizes de rigidez elementares K_{ij} de cada triângulo (utilizando triângulos lineares), são somadas da seguinte forma:

	1	2	3	4	5	6	7	8	...	N
1										
2		$K_{11}^{A}+K_{33}^{B}$	K_{12}^{A}	K_{32}^{B}	$K_{13}^{A}+K_{31}^{B}$					
3		K_{21}^{A}	K_{22}^{A}		K_{23}^{A}					
4		K_{23}^{B}		$K_{22}^{B}+K_{11}^{C}$	$K_{21}^{B}+K_{12}^{C}$	K_{13}^{C}				
5		$K_{31}^{A}+K_{13}^{B}$	K_{32}^{A}	$K_{12}^{B}+K_{21}^{C}$	$K_{33}^{A}+K_{11}^{B}+K_{22}^{C}$	K_{23}^{C}				
6				K_{31}^{C}	K_{32}^{C}	K_{33}^{C}				
7										
8										
.										
.										
N										

Figura 4.2 Soma das matrizes elementares.

Baseado na estrutura vista na Figura 4.2, nota-se que nos problemas sobre os quais o código computacional desenvolvido é aplicado geram um sistema de equações lineares que possui as seguintes características:

1) Simétrico;

2) Esparso;

3) Positivo-Definido;

4) Banda.

Então, baseado nas características acima e no fato de que o reordenamento reduz a banda da matriz, optou-se por implementar a eliminação gaussiana para a resolução. Ressalta-se que não é a eliminação gaussiana tradicional, sendo que no algoritmo implementado são exploradas as características do sistema para reduzir o tempo computacional nesta etapa. Como pode ser visto na Figura 4.2, a matriz global do sistema é do tipo banda.

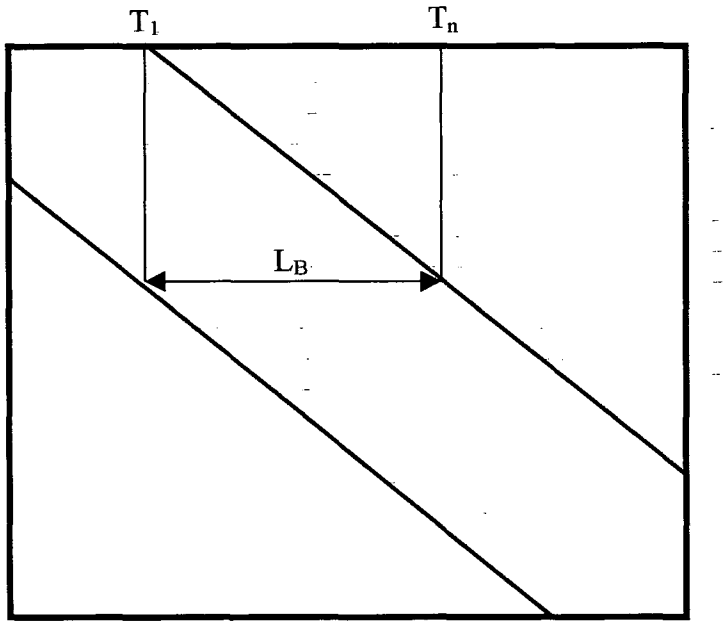


Figura 4.3 Matriz Banda.

A Figura 4.3, representa uma matriz banda, onde L_B é a largura da banda, ou seja, é a maior distância entre as numerações dos nós, para um mesmo elemento, ou, também pode ser definido como a distância entre os mais distantes termos da matriz global, que

podem ser obtidos pela equação (4.6):

$$L_B = T_n - T_L + 1 \qquad (4.6)$$

Como a matriz é simétrica, pode-se trabalhar somente com meia-banda. Esta é a estrutura explorada para a resolução do sistema. Maiores detalhes sobre este tópico são apresentados no Capítulo 6.

4.4 Estimativas de Erros

As estimativas de erros em elementos finitos, em geral, são classificadas em duas grandes categorias:

1) Estimativas *a priori*;

2) Estimativas *a posteriori*.

As estimativas *a priori* são aquelas que podem ser obtidas antes que se saiba a solução do problema em estudo. Este tipo de estimativa é geralmente usada no projeto do método de elementos finitos, ou seja, é usado para determinar a taxa de convergência do método e também para encontrar dependências nos parâmetros do problema, sendo que a taxa de convergência do MEF, tende a zero com o refinamento dos elementos (parâmetro h).

A segunda categoria de estimativas, *a posteriori*, é obtida depois que a solução é conhecida. Estas estimativas são muito úteis para adaptar e controlar o erro da solução. Os dois principais tipos de estimativas *a posteriori* são: a "stress recovery", que também é chamada como "postprocessing", a qual proposta para problemas elípticos lineares por Zienkiewicz and Zhu (1987). E, o segundo tipo é a estimativa baseada no resíduo, que é subdividida em: explícita e implícita.

Neste tópico são apresentados os fundamentos para a realização de estimativas de erros a *priori* e a *posteriori*. Resultados auxiliares são apresentados no Apêndice A, onde se faz referêcia a outros trabalhos com ênfase em apectos matemáticos e estimativas de erros em elementos fintos.

4.4.1 Estimativas a Priori

As estimativas *a priori* indicam a taxa de convergência do MEF em malhas quando h (parâmetro global que caracteriza o tamanho do elemento) tende a zero. Como resultados naturais são obtidos critérios para selecionar as funções de interpolação locais, podendo-se comparar qualitativamente a performance de vários elementos para o problema (Carey and Oden, 1983).

4.4.1.1 Resultados da Teoria Matemática para EDPs Elípticas

O tratamento matemático do MEF é baseado na formulação variacional das EDP's elípticas (Braess, 1997). Neste tópico são apresentadas as definições, teoremas e corolários mais relevantes para a obtenção das estimativas *a priori* do erro.

Definição 1 – Espaços de Sobolev

Defme-se $W^{m,p}(\Omega) := \{u : \Omega \to R$ tal que $D^{\alpha}u \in L_p(\Omega)$, $\forall\, \alpha$ tal que $\alpha \leq m\}$. Então, seja $\Omega \subset R^n$, aberto e não vazio, $n \in N$. Seja $1 \leq p < \infty$ e $m = 1, 2, \ldots$ O espaço de Sobolev $W^{m,p}(\Omega)$ é o conjunto de todas as funções de modo que:

$$D^{\alpha}u \in L_p(\Omega)\ ,\ \forall\, \alpha : |\alpha| \leq m \tag{4.7}$$

Onde $L_p(\Omega)$ é o espaço das funções u tal que $\int_{\Omega} |u(x)|dx < \infty$ e $D^{\alpha}u$ representa as derivadas generalizadas da função. Para m = 0 define-se que:

$$W^{0,p}(\Omega) = L_p(\Omega)\,. \tag{4.8}$$

Teorema 1 - Interpolação

A teoria de interpolação tem papel central nas estimativas de erros, sendo que o

principal resultado desta teoria é o seguinte: Seja $(\hat{\Omega}, \hat{D}, \hat{p})$ um elemento finito para o qual o conjunto de graus de liberdade $\hat{D}$ envolve a especificação de derivadas parciais de ordem $s \geq 0$. Além disso, para m, k ∈ N, tem-se que:

$$W^{k+1,p}(\hat{\Omega}) \rightarrow \overline{C}^{s}(\hat{\Omega}) \tag{4.9}$$

$$W^{k+1,p}(\hat{\Omega}) \rightarrow W^{m,q}(\hat{\Omega}) \tag{4.10}$$

$$P_k(\hat{\Omega}) \subset \hat{P} \subset W^{m,q}(\hat{\Omega}) \tag{4.11}$$

Então, existe uma constante positiva $C = C(\hat{\Omega}, \hat{D}, \hat{P})$, tal que para todo elemento $(\hat{\Omega}_e, \hat{D}_e, \hat{P}_e)$ equivalente por um mapeamento afim a $(\hat{\Omega}, \hat{D}, \hat{P})$ e $\forall v \in W^{k+1,p}(\Omega_e)$, tem-se:

$$\left| v - \prod\nolimits_e v \right|_{m,q,\Omega_e} \leq C(\hat{\Omega}, \hat{D}, \hat{P}) meas\,(\Omega_e)^{1/q-1/p} \frac{h_e^{\,k+1}}{\rho_e^{\,m}} |v|_{k+1,p,\Omega_e} \tag{4.12}$$

Onde $\prod_e v$ é o interpolante em P_e de v e:

$$h_e := dia(\Omega_e) \tag{4.13}$$

$$\rho_e := \sup\{dia(S), S \text{ é uma esfera contida em } \Omega_e\} \tag{4.14}$$

Alguns comentários sobre as inclusões presentes no teorema:

1. A inclusão referenciada na equação (4.9).

Se D_e contém graus de liberdade envolvendo derivadas de ordem s, então deve-se ter as funções de interpolação locais Ψ em $C^s(\Omega_e)$. Pelo teorema de inclusão de Sobolev (Carey and Oden, pp. 14, 1983), para $1 \leq p \leq \infty$, a inclusão $W^{k+1,p}(\hat{\Omega}) \rightarrow C^s(\hat{\Omega})$ é válida quando:

$$(k + 1 - s)p > \mathrm{n} \tag{4.15}$$

Se s = 0, então a equação (4.15) é satisfeita para ∀ k > 1, para n=1 ou n = 2. Para n = 3 e $p \geq 2$, a equação (4.15) também é válida para $k \geq 1$. No entanto, se s = 1 e p = 2, deve-se ter $k \geq 2$ para n = 2 e n = 3, e $k \geq 3$ para n = 4. Note que no caso s = 1, n = 3, p = 1, k = 3 não satisfaz a equação (4.15). Portanto, a equação (4.15) é satisfeita para

todas as aplicações usuais dos elementos finitos, mas sempre deve-se verificá-la para assegurar estimativas do tipo (4.12). Para o caso implementado neste trabalho, o qual utiliza funções de interpolação lineares e quadráticas tem-se que os elementos são C^0, ou seja, são funções continuamente diferenciáveis, portanto, s = 0 e esta condição é facilmente satisfeita.

2. De modo análogo, a validade na inclusão da equação (4.9) deve ser verificada a partir das condições estabelecidas pelo teorema de inclusão de Sobolev.

3. A presença dos parâmetros h_e e ρ_e na equação (4.12) é consistente com o fato de que para h_e fixo, a equação (4.12) sugere que em presença de elementos muito distorcidos (ρ_e muito pequeno), conduzirá a elevados erros de interpolação.

Na equação (4.15), p é a norma, k é a ordem da função de interpolação e n é a dimensão do espaço Euclideano R^n, onde $\Omega \subset R^n$.

Definição 2 – Família de Elementos Regulares

Considere o domínio $\overline{\Omega}$, onde Q_h é uma partição de $\overline{\Omega}$ em E elementos. Define-se a família $F = \left\{(\overline{\Omega}_e, D_e, P_e) : \overline{\Omega}_e \in Q_h, 1 \le e \le E\right\}$ como regular se:

i) F é uma família afim;

ii) $\exists \mu > 0$ tal que $h_e / \rho_e \le \mu$, $\forall \overline{\Omega}_e$ pertencente a família de Q_h.

iii) Os diâmetros h_e aproximam-se de zero.

Definição 3 – Malhas Uniformes

As malhas uniformes são definidas de acordo com a equação (4.16)

$$\sigma := \frac{h}{p} = \frac{\max\limits_{1 \le e \le E}\{h_e\}}{\min\limits_{1 \le e \le E}\{\rho_e\}} \qquad \gamma := \frac{h}{\hat{h}} = \frac{h}{\min\limits_{1 \le e \le E}} \tag{4.16}$$

Sendo que quando $\gamma = 1$, tem-se uma malha uniforme. Quando $\exists \sigma > 0$ tal que $\sigma_0 >$

σ em uma seqüência de refinamentos, diz-se que as malhas são quase uniformes.

Para a obtenção de estimativas de erros confiáveis é necessário que as malhas sejam construídas de forma a atender as definições 2 e 3.

Teorema 2 – Generalizado de Lax-Milgran

Este Teorema foi provado por Babuska and Aziz (1972), sendo um dos principais teoremas para a realização de estimativas *a priori* de erros, o qual tem o seguinte enunciado: Se H e G são dois espaços de Hilbert reais e b(,) uma forma bilinear em H x G, tal que:

i) b(,) é contínuo, ou seja, $\exists$ M > 0 tal que: $b(u,v) \leq M\|u\|_H \|v\|_G$, $\forall$ u $\in$ H , $\forall$ v $\in$ G ($\|.\|_H$ e $\|.\|_G$) são as normas nestes espaços de Hilbert.

ii) b(,) é coercivo, ou seja, $\exists\alpha$ tal que: $\inf_{\substack{u\in H \\ \|u\|_H=1}} \sup_{\substack{v\in G \\ \|v\|_G\leq 1}} |b(u,v)| \geq \alpha > 0$

iii) $\forall$ v $\neq$ 0 em G, $\sup_{u\in H} |b(u, v)| > 0$ $\exists$ u* $\in$ H (único) tal que b(u*,v) = f(v), $\forall$ v $\in$ G, onde f $\in$ G'. Além disso:

$$\|u^*\|_H \leq \frac{1}{\alpha}\|f\|_G \tag{4.17}$$

Com base no Teorema 2, considere o seguinte PVC:

$$Aw = f^* \tag{4.18}$$

$$\gamma_j w = g_j \quad \text{em } \partial\Omega \quad 0 \leq j \leq m-1 \tag{4.19}$$

Onde A é um operador diferencial linear elíptico de ordem 2m. Nas equações (4.18) e (4.19) os dados do problema são f* (termo fonte) e g_j (condições de contorno).

Então, define-se u:= w – w_0 e $\gamma_j w_0 = g_j$ em $\partial\Omega$. O problema agora consiste em determinar u que satisfaz a seguinte equação:

$$Au = f = \mathrm{f}^{*} - \mathrm{A}w_0 \ , \ \mathrm{x} \in \Omega \tag{4.20}$$

$$\gamma_j u = 0 \ \text{ em } \partial\Omega \ \ 0 \le \mathrm{j} \le \mathrm{m} - 1$$

Considerando a forma bilinear definida pela equação abaixo:

$$b(u,v) := \sum_{|\alpha|,|\beta| \le m} \int_{\Omega} a_{\alpha,\beta} D^{\alpha} u D^{\beta} v dx \tag{4.21}$$

onde:

$$a_{\alpha,\beta} \in C^{\infty}(\Omega) \ \text{ e } u, v \in \mathrm{H}_0^{\mathrm{m}}(\Omega) := \left\{ w \in H^{m}(\Omega) \text{ tal que } \gamma_{\mathrm{j}} w = 0 \text{ em } \partial\Omega ; 0 \le \mathrm{j} \le \mathrm{m} - 1 \right\}.$$

Assume-se então, que os $a_{\alpha,\beta}(x)$ são tais que:

$$\exists \alpha, M > 0 \text{ tal que } \mathrm{b(u, v)} \le \mathrm{M} \|\mathrm{u}\|_m \|\mathrm{v}\|_m \tag{4.22}$$

$$\inf_{\|\mathrm{u}\|_{m=1}} \sup_{\|\mathrm{v}\|_{m \le 1}} |\mathrm{b(u, v)}| \ge \alpha > 0 \tag{4.23}$$

$$\sup_{u \in H_0^m(\Omega)} |b(u,v)| > 0 \tag{4.24}$$

Então, via Teorema 2, existe uma solução única para o seguinte problema variacional: Encontre $u \in H_0^m(\Omega)$ tal que:

$$b(u,v) = F(v) \ , \ \forall \mathrm{v} \in \mathrm{H}_0^{\mathrm{m}}(\Omega) \tag{4.25}$$

Onde :

$$\mathrm{F(v)} := \int_{\Omega} f v dx \ , \ \forall \ \mathrm{v} \in \mathrm{H}_0^{\mathrm{m}}(\Omega) \tag{4.26}$$

Note que a equação (4.25) é equivalente a equação (4.18), então:

$$Au := \sum_{|\alpha|,|\beta| \le m} (1)^{|\beta|} D^{\beta} (a_{\alpha,\beta}(x) D^{\alpha} u \tag{4.27}$$

Onde D^{α} e D^{β} representam as derivadas generalizadas. Agora, o problema consiste em encontrar a solução por elementos finitos, ou seja, deve ser determinado $u_h \in \mathrm{H}_0^{\mathrm{h}}(\Omega)$ tal que:

$$b(u_h, v_h) = F(v_h) \ , \ \forall \mathrm{v_h} \in \mathrm{H}_0^{\mathrm{m}}(\Omega) \tag{4.28}$$

onde:

$$H_0^h(\Omega) := \left\{ v_h \in H^h(\Omega) := \gamma_j v_h = 0 \text{ e.q.t.p. em } \partial\Omega ; 0 \leq j \leq m-1 \right\} \tag{4.29}$$

Como conseqüência do Teorema 2, assume-se que este problema possui solução única, então tem-se que os itens de (i) a (iii) do Teorema são verdadeiros.

Corolário 1 – Primeira Estimativa de Erros

Assumindo que as hipóteses dos Teoremas 4 e 5 (Apêndice A) são satisfeitas. Suponha que $f \in H^s$, $s \geq 0$, e $g_j \in H^{pj}(\partial\Omega)$, $p_j \geq 0$, $0 \leq j \leq m-1$. Então, $\exists\, C > 0$ tal que quando $h \to 0$, tem-se que:

$$\|e\|_{m,\Omega} \leq C\left[h^{\mu 1} \|f\|_{s,\Omega} + b^{\mu 2} \sum_{j=0}^{m-1} \|g_j\|_{p_j,\partial\Omega} \right] \tag{4.30}$$

onde:

$$\mu_1 := \min(k+1-m, s+m) \tag{4.31}$$

$$\mu_2 := \min(k+1-m, \min_{0\leq j\leq m-1} p_j + j + \frac{1}{2} - m) \tag{4.32}$$

Com o objetivo de estimar erros em normas com ordem menor que m, tem-se o método de Aubin-Nitsche (1969), sendo que o resultado mais importante é do Teorema 2.

Teorema 2 – Método de Aubin-Nitsche

Assumindo que as hipóteses dos teoremas 4 e 5 (Apêndice A) são satisfeitas e que $g_j = 0$ em $\partial\Omega$, $0 \leq j \leq m-1$. E o problema variacional satisfaça as hipóteses análogas às equações (4.22) a (4.24). Então, $\exists\, C > 0$ tal que quando $h \to 0$ o erro de aproximação por elementos finitos, na norma $H^T(\Omega)$ é estimado via equação (4.33):

$$\|e\|_{\tau,\Omega} \leq Ch^{\gamma}\|u\|_{r,\Omega} \quad , \quad 0 \leq \tau \leq \mathrm{m} \tag{4.34}$$

onde:

$\gamma = v + \mu = \min(2(k+1-m), k+1-\tau, r-\tau)$. Além disso, se $f \in H^s(\Omega)$, s ⩾ 0 e $\|u\|_{\tau,\Omega} \leq C\|f\|_{r-2m,\Omega}$, a equação (4.34) pode ser reescrita como:

$$\|e\|_{\tau,\Omega} \leq Ch^{\eta}\|f\|_{s,\Omega} \quad , \quad 0 \leq \tau \leq \mathrm{m} \tag{4.35}$$

onde:

$$\eta = \min(2(k+1-m), k+1-\tau, s+2m-\tau) \tag{4.36}$$

E finalmente, se $f \in L_2(\Omega)$, então:

$$\|e\|_{0,\Omega} \leq Ch^{k+1}\|f\|_{0,\Omega} \quad , \quad \text{sempre que k} + 1 \leq 2\mathrm{m} \tag{4.37}$$

4.4.1.2 Exemplo de Estimativas a priori de Erros

Neste tópico é apresentado um exemplo visando um entendimento prático de como realizar estimativas *a priori* do erro. Considere a seguinte equação:

$$\begin{aligned} -\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} &= f \quad \text{em} \quad \Omega \\ u &= 0 \quad \text{em} \quad \partial\Omega \end{aligned} \tag{4.38}$$

Onde $\partial\Omega \in \mathrm{C}^2$. Então, deve-se formular o problema contínuo e discreto, para isso define-se o operador b(. , .) associado ao operador $\Delta = -\frac{\partial}{\partial x^2} - \frac{\partial}{\partial y^2}$:

$$b(u,v) := \int_{\Omega} \nabla u \nabla v dx \quad , \quad \mathrm{u,v} \in \mathrm{H}_0^1(\Omega) \tag{4.39}$$

O problema variacional consiste em determinar $u \in H_0^1(\Omega)$ tal que $\mathrm{b(u,v)} = \int_{\Omega} fvdx \quad , \quad \forall \mathrm{v} \in \mathrm{H}_0^1(\Omega)$. Para aplicar o Teorema 2, deve-se mostrar que as equações (4.22) a (4.24) são satisfeitas (as provas encontram-se em Oden

and Carey, 1983).

As condições são verdadeiras, ou seja, pode-se provar que as hipóteses do Teorema de Lax-Milgram (Teorema 2) são válidas. Então, existe uma única solução $u^* \in H_0^1(\Omega)$ para o problema variacional $b(u,v) = F(v)$, $\forall u \in H_0^1(\Omega)$. Em relação a aproximação por elementos finitos, é construído um subespaço $H_0^h(\Omega) \subset H_0^1(\Omega)$ via funções de base polinomiais por partes, então o problema discreto consiste em obter $u_h \in H_0^h(\Omega)$ tal que $\int_\Omega \nabla u_h \nabla v_h dx = \int_\Omega f v_h dx$, $\forall v_h \in H_0^h(\Omega)$. Como o operador b(. , .) é o mesmo, via Teorema 2 existe uma solução única $u_h^* \in H_0^h(\Omega)$. Portanto, de acordo com a equação (A.19 – apêndice A), tem-se:

$$\|e\|_{1,\Omega} \leq C\left(1 + \frac{M}{\alpha_h}\right) h^{\mu} |u^*|_{r,\Omega} \tag{4.40}$$

onde $u^* \in H^r \cap H_0^1$, r > 0 e $\mu := \min(k, r-1)$, sendo que r limita a taxa de convergência. Então, é obtido a seguinte estimativa do erro:

$$\|e\|_{1,\Omega} \leq C h^{\mu 1} \|f\|_{s,\Omega} \tag{4.41}$$

onde $u^* \in H^r$, r ≥ 2, s = r – 2 e $\mu_1 = \min(k, s+1)$.

Via equação (4.34), tem-se a seguinte estimativa:

$$\|e\|_{0,\Omega} \leq C h^{\gamma} \|u^*\|_{r,\Omega} , \quad 0 \leq \tau \leq m \tag{4.42}$$

Finalmente, pela equação (4.35), tem-se:

$$\|e\|_{0,\Omega} \leq C h^{\eta} \|f\|_{s,\Omega} , \quad 0 \leq \tau \leq m \tag{4.43}$$

onde:

$$\eta = \min(2k, k+1, s+2) \tag{4.44}$$

$$\|u^*\|_{r,\Omega} \leq C \|f\|_{r-2,\Omega} \tag{4.45}$$

Portanto se $f \in L_2(\Omega)$ e s = 0 não adianta utilizar k = 2 (funções de interpolação quadráticas), pois a taxa de convergência do erro para os dois casos é (linear e quadrático) $\theta(h^2)$. O mesmo ocorrendo se s = 1, onde é obtido a taxa de convergência

$\theta(h^3)$ para ambos os valores de k.

4.4.2 Estimativas a Posteriori

As estimativas *a priori* do erro são dadas em termos da solução exata do problema e são muito importantes para a determinação da taxa de convergência do problema. No entanto, na prática a solução exata dificilmente é conhecida, e devido a natureza dos dados do problema a solução de um determinado PVC, pode ser menos regular em determinadas regiões do domínio. Nestes casos é necessário aumentar os graus de liberdade dos elementos (utilizar funções de interpolação de mais alta ordem). Um modo eficiente de conseguir uma melhor solução é refinar a malha nestes subdomínios, são os chamados métodos adaptativos (Strouboulis and Haque, 1991).

Para a realização destas estimativas, primeiramente os cálculos são efetuados em uma malha provisória, e então, calcula-se a estimativa *a posteriori* do erro identificando qual a região da malha que induz a grandes erros. Utilizando estas informações, pode-se refinar localmente a malha e se necessário este procedimento é repetido várias vezes (Johnson, 1990).

Consideremos a equação de Poisson com condições de contorno de Dirichlet, e que os triângulos da malha são regulares.

$$-\frac{\partial^2 u}{\partial x^2}-\frac{\partial^2 u}{\partial y^2}=f \tag{4.46}$$

Após a realização dos cálculos sobre todos os elementos, supomos que u_h é a solução em elementos finitos do problema, supomos ainda que Γ_h é o conjunto de todos os elementos, cujas arestas dos triângulos estão sobre o contorno do domínio.

Inserindo a solução aproximada u_h na equação (4.46) é obtido um resíduo. Além disso, u_h difere da solução clássica devido as descontinuidades de u nos elementos do contorno, então:

$$R_{\Omega_e} := R_{\Omega_e}(u_h)=\frac{\partial^2 u_h}{\partial x^2}\frac{\partial^2 u_h}{\partial y^2}+f \quad , \quad \forall\, \Omega_e \in \mathrm{T_h} \tag{4.47}$$

Ou seja, o resíduo é calculado pela equação (4.47) para todos os triângulos (Ω_e) internos ao domínio. Para os triângulos, cujas arestas estão no contorno, esta estimativa é obtida com base na equação (4.63).

$$R_e := R_e(u_h) = \frac{\partial u_h}{\partial n} \quad , \quad \forall \, \mathrm{e} \in \Gamma_\mathrm{h} \tag{4.48}$$

Segundo Johnson (1994), há essencialmente cinco diferentes aproximações para a construção das estimativas *a posteriori*: estimativas residuais (Babuska and Rheinboldt, 1978a), estimativas baseadas num problema local de Neumann, as quais são obtidas usando a norma da energia proposta por Bank and Weiser (1985), estimativas baseadas num problema local de Dirichlet (Babuska and Rheinboldt, 1978b), estimativas baseadas na média (Zienkiewicz and Zhu, 1987) e estimativas hierárquicas.

Para estimar os resíduos locais, usa-se as funções dadas nas equações (4.47) e (4.48):

$$\eta_{\Omega_e,R} := \left\{ h_{\Omega_e}^2 \left\| R_{\Omega_e} \right\|_{0,\Omega_e}^2 + \frac{1}{2} \sum_{e \in \partial\Omega} h_e \left\| R_e \right\|_{0,e}^2 \right\}^{1/2} \quad , \quad \forall \, \Omega_\mathrm{e} \in \mathrm{T_h} \tag{4.49}$$

Fazendo um somatório ao quadrado sobre todos os triângulos, é obtido um resíduo global:

$$\eta_R := \left\{ \sum_{\Omega_e \in T_h} h_{\Omega_e}^2 \left\| R_{\Omega_e} \right\|_{0,\Omega_e}^2 + \sum_{e \in \Gamma_h} h_e \left\| R_e \right\|_{0,e}^2 \right\}^{1/2} \tag{4.50}$$

Com base na estimativa dada pela equação (4.49), pode-se determinar um $\delta > 0$ como sendo uma tolerância, e utilizar estas informações como critérios de refinamento dos elementos. Ou seja, refinar os elementos Ω_e, cujo resíduo dado pela equação (4.49) for maior que δ.

4.5 Conclusão

Neste capítulo foram apresentados alguns procedimentos numéricos e os conceitos fundamentais para a realização de estimativas de erros. O controle de erros e métodos

adaptativos são fundamentais para todas as análises numéricas (Zienkiewicz, 1992). Nas análises dos erros do capítulo 5, são explorados alguns dos resultados que aqui foram apresentados.

Capítulo 5 - Validação

5.1 Introdução

Neste capítulo é feita a validação do código computacional implementado. A validação é feita com base em vários problemas cujas soluções analíticas são conhecidas, as quais são comparadas com a solução numérica. Conforme descrito em capítulos anteriores, o programa proposto tem várias flexibilidades:

- A combinação de quaisquer condições de contorno de Neumann e Dirichlet;
- A resolução em contornos genéricos;
- Propriedades físicas diferentes;
- Cargas distribuídas;
- Cargas Concentradas.

Portanto, cada problema apresentado tem o objetivo de validar um ou mais desses itens, isoladamente ou com várias irregularidades juntas. A equação governante é a descrita pela equação (5.1).

$$\frac{-\partial}{\partial x}\left[k_x(x,y)\frac{\partial u}{\partial x}\right]-\frac{\partial}{\partial y}\left[k_y(x,y)\frac{\partial u}{\partial y}\right]=f(x,y) \tag{5.1}$$

Sabe-se que o MEF minimiza o erro na norma da energia, mas os erros serão apresentados na norma infinito, que é definida por:

$$\|e\|_\infty = \max_{x,y\in\Omega}|e(x,y)| \tag{5.2}$$

Ou seja, é considerado o maior erro, pois assim pode-se garantir com certeza que o erro em todos os pontos do domínio é menor que um determinado valor previamente estabelecido.

Em cada problema são descritos detalhadamente os testes realizados. Define-se que durante a apresentação dos resultados é utilizada a notação $\|erro\|_\infty^{MEFL}$ para representar os erros obtidos na norma infinito utilizando elementos lineares e,

$\|erro\|_{\infty}^{MEFQ}$ para representar os erros utilizando funções quadráticas.

5.2 Problema 1

O primeiro problema analisado é descrito pela equação:

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=\frac{x}{y}+\frac{y}{x} \tag{5.3}$$

Ou seja, é resolvida a equação de Poisson com os seguinte dados:

$$k_x(x,y)=1 \tag{5.4}$$

$$k_y(x,y)=1 \tag{5.5}$$

$$f(x,y)=\frac{x}{y}+\frac{y}{x} \tag{5.6}$$

A equação (5.3) está sujeita às seguintes condições de contorno:

$u(x, 1) = x \ln x$ $\quad u(x,2) = x \ln(4x^2)$ $\quad$ para $\quad 1 \le x \le 2$ $\quad$ (5.7)

$u(1,y) = y \ln y$ $\quad u(2,y) = 2y \ln(2y)$ $\quad$ para $\quad 1 \le y \le 2$ $\quad$ (5.8)

A solução analítica do problema representado pelas equações (5.3), (5.7) e (5.8) é conhecida e é dada por:

$$u(x, y) = xy \ln(xy) \tag{5.9}$$

Na resolução deste problema utiliza-se três malhas computacionais com 32, 128 e 512 elementos, respectivamente, utilizando triângulos lineares e quadráticos. A Figura 5.1 ilustra uma das malhas utilizadas e os resultados obtidos são apresentados na Tabela 5.1.

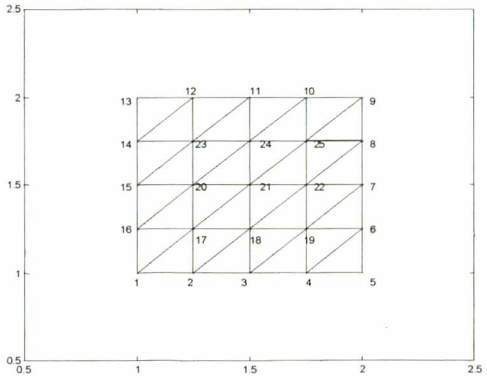


Figura 5.1 Malha com 32 elementos.

Tabela 5.1 Resultados obtidos no Problema 1.

N°. Elementos	h	$\|erro\|_{\infty}^{MEFL}$	$\|erro\|_{\infty}^{MEFQ}$
32	0.1768	3.4138 E-4	5.3511 E-6
128	0.0884	8.8697 E-5	4.0659 E-7
512	0.0442	2.2778 E-5	2.7823 E-8

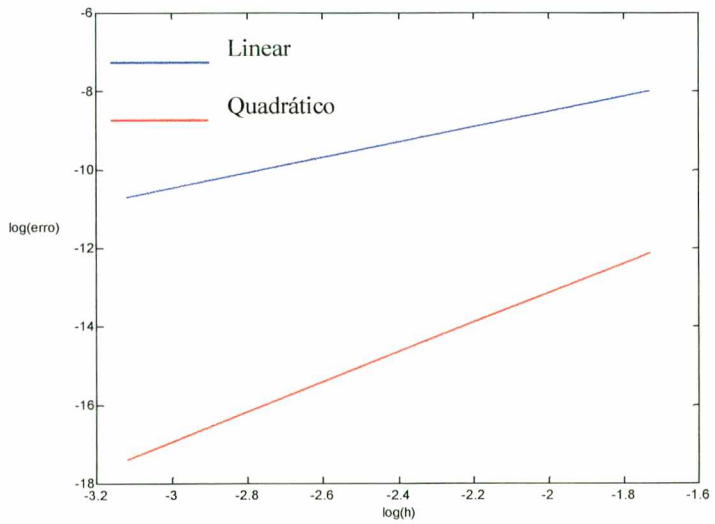


Figura 5.2 Evolução do erro para o Problema 1.

Logo, analisando os resultados apresentados na Tabela 5.1 e Figura 5.2, pode-se verificar que a solução numérica apresentada tende a solução analítica com o refinamento da malha. Este é um problema teste onde k_x e k_y são constantes, f(x,y) é

variável e o problema está sujeito somente à condições de contorno essenciais.

5.3 Problema 2

O segundo problema analisado é descrito pela equação governante representada pela equação:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \tag{5.10}$$

Comparando a equação (5.10) com a equação (5.1), obtém os seguintes dados:

$$k_x(x, y) = 1 \tag{5.11}$$

$$k_y(x, y) = 1 \tag{5.12}$$

$$f(x, y) = 0 \tag{5.13}$$

Este problema é um caso particular de um problema bidimensional que recai em um problema unidimensional, onde a solução analítica é conhecida e é dada por:

$$u(x, y) = 25x + 50 \tag{5.14}$$

O problema é resolvido utilizando três sub-casos, os quais tem objetivos específicos.

5.3.1 Problema 2 - Caso A

No caso A a equação (5.10) está sujeita às seguintes condições de contorno:

$$\frac{\partial u}{\partial y}(x, 0) = 0 \qquad \frac{\partial u}{\partial y}(x, 2) = 0 \qquad \text{para } 0 \le x \le 2 \tag{5.15}$$

$$u(0,y) = 50 \qquad u(2,y) = 100 \qquad \text{para } 1 \le y \le 2 \tag{5.16}$$

O caso A é resolvido utilizando duas malhas computacionais, tendo elas respectivamente 32 e 128 elementos, os resultando obtidos são apresentados na Tabela

5.2.

Tabela 5.2 Resultados obtidos no Problema 2 – Caso A.

Nº. Elementos	h	$\|erro\|_{\infty}^{MEFL}$	$\|erro\|_{\infty}^{MEFQ}$
32	0.3536	4.2633 E-14	1.4211 E-13
64	0.1768	4.2633 E-14	3.8369 E-13

Pelos resultados apresentados na Tabela 5.2 e vizualizados na Figura 5.3, nota-se que, devido à simplicidade deste problema, mesmo com uma malha "grosseira" (h = 0.3536) é obtida uma ótima solução. Também nota-se que a utilização de funções de interpolação quadráticas não melhora a solução, isso devido principalmente aos erros de arredondamento. No entanto, constata-se que a equação que está sendo resolvida, atende a todas às condições estabelecidas no capítulo 4, e assim sendo a taxa de convergência do erro é determinada pela equação (4.43). O erro obtido é da ordem h^2 ($\theta(h^2)$) para as funções de interpolação lineares e quadráticas. Então, para os casos B, C e D é utilizada apenas a malha referente a h = 0.3536.

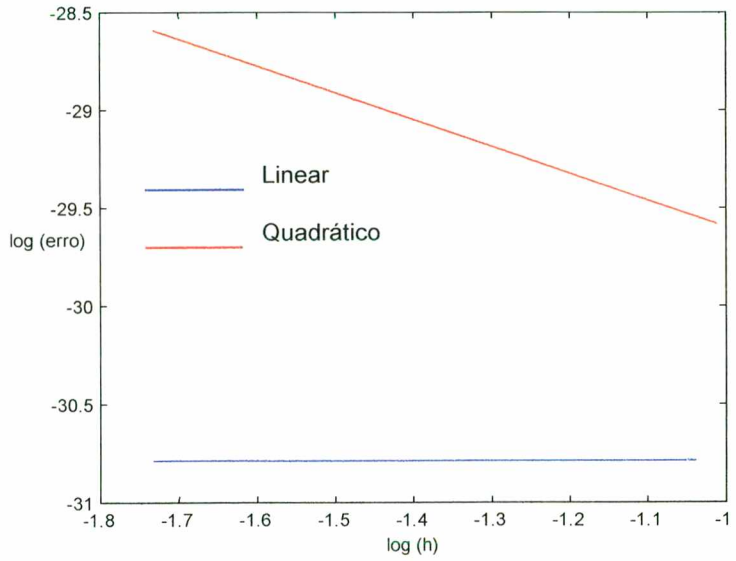


Figura 5.3 Evolução do erro para o Problema 2 – Caso A.

5.3.2 Problema 2 - Caso B

No caso B a equação (5.10) está sujeita às seguintes condições de contorno:

$$\frac{\partial u}{\partial y}(x, 0) = 0 \qquad u(x, 2) = 25x + 50 \qquad \text{para } 0 \le x \le 2 \tag{5.16}$$

$$u(0,y) = 50 \qquad u(2,y) = 100 \qquad \text{para } 1 \le y \le 2 \tag{5.17}$$

A diferença do caso B e do caso A está nas condições de contorno, sendo que no caso B há condições de contorno de Neumann somente em um dos lados do domínio. Os resultados obtidos para este caso são apresentados na Tabela 5.3.

Tabela 5.3 Resultados obtidos no Problema 2 – Caso B

Nº. Elementos	h	$\|erro\|_{\infty}^{MEFL}$	$\|erro\|_{\infty}^{MEFQ}$
32	0.3536	4.2633 E-14	9.9476 E-14

Novamente é obtido um melhor resultado para os elementos lineares, isso devido aos erros de arredondamento cometidos na utilizaçao de funções de interpolação quadrática, uma vez que a solução do problema é linear.

5.3.3 Problema 2 - Caso C

Para o caso C a equação (5.10) tem as seguintes condições de contorno:

$$u(x, 0) = 25x + 50 \qquad u(x, 2) = 25x + 50 \qquad \text{para } 0 \le x \le 2 \tag{5.18}$$

$$u(0,y) = 50 \qquad \frac{\partial u}{\partial x}(2, y) = 25 \qquad \text{para } 0 \le y \le 2 \tag{5.19}$$

A diferença do caso C para os casos A e B também está nas condições de contorno, onde, no problema C, tem-se uma derivada prescrita (CC. de Neumann) em uma aresta diferente dos dois casos anteriores. O erro na norma infinito é apresentado

na Tabela 5.4.

Tabela 5.4 Resultados obtidos no Problema 2 – Caso C.

Nº. Elementos	h	$\|erro\|_{\infty}^{MEFL}$	$\|erro\|_{\infty}^{MEFQ}$
32	0.3536	1.4211×10^{-14}	1.5390×10^{-10}

5.3.4 Problema 2 - Caso D

No caso D a equação (5.10) é resolvida no domínio apresentado pela Figura 5.4. Na mesma Figura pode-se observar a disposição dos pontos internos, bem como a triangulação de Delaunay.

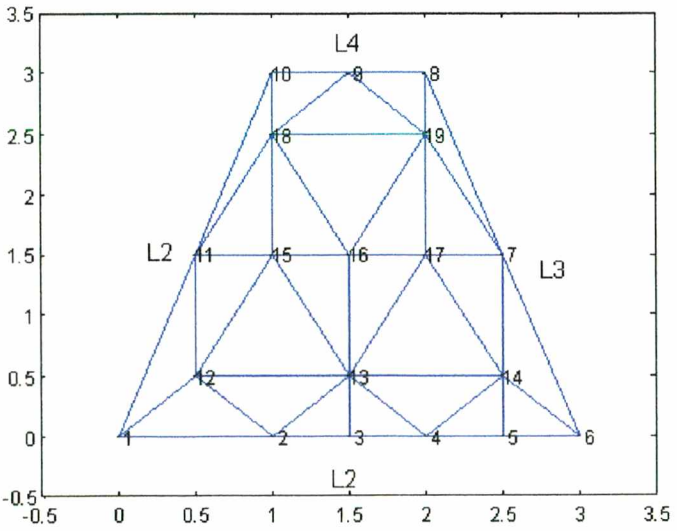


Figura 5.4 Domínio para o Caso D.

Considerando o contorno representado pela Figura 5.4, tem-se as seguintes condições de contorno:

$$u(x,y) = \frac{12.5}{1.5} y + 50 \qquad \text{para } (x,y) \in L1 \qquad (5.20)$$

$$\frac{\partial u}{\partial y}(x,y) = 0 \qquad \text{para (x,y)} \in \text{L2} \qquad (5.21)$$

$$u(x,y) = -\frac{25}{3}y + 125 \qquad \text{para (x,y)} \in \text{L3} \qquad (5.22)$$

$$\frac{\partial u}{\partial y}(x,y) = 0 \qquad \text{para (x,y)} \in \text{L4} \qquad (5.23)$$

Os resultados obtidos são apresentados na Tabela 5.5.

Tabela 5.5 Resultados obtidos no Problema 2 – Caso D.

Nº. Elementos	h	$\|erro\|_{\infty}^{MEFL}$	$\|erro\|_{\infty}^{MEFQ}$
25	1.7678	2.8422 E-14	9.9476 E-14

Analisando os resultados obtidos no problema 2, conclui-se que na resolução de problemas deste tipo não há a necessidade de aumentar a ordem das funções de interpolação.

5.4 Problema 3

O terceiro problema analisado é descrito pela equação governante representada pela equação:

$$\frac{\partial}{\partial x}\left[xy\frac{\partial u}{\partial x}\right] = 4xy^2 \qquad (5.24)$$

Ou seja, é resolvida a equação de Poisson com os seguintes dados:

$$k_x(x,y) = xy \qquad (5.25)$$

$$k_y(x,y) = 0 \qquad (5.26)$$

$$f(x,y) = 4xy^2 \qquad (5.27)$$

A equação (5.24) está sujeita às seguintes condições de contorno:

$u(x, 0) = x^2 + 2$ $\quad u(x,2) = 2x^2 + 2$ $\quad$ para $\quad 0 \leq x \leq 2$ $\quad$ (5.28)

$u(0,y) = y + 2$ $\quad u(2,y) = 4y + 2$ $\quad$ para $\quad 0 \leq y \leq 2$ $\quad$ (5.29)

A solução analítica do problema representado pelas equações (5.24), (5.28) e (5.29) é conhecida e é dada por:

$$u(x, y) = x^2y + 2 \qquad (5.30)$$

Na resolução deste problema utiliza-se três malhas computacionais com 8, 32 e 128 elementos, respectivamente, utilizando triângulos lineares e quadráticos. A Figura 5.5 ilustra uma das malhas utilizadas. Neste caso, objetiva-se mostrar que o programa desenvolvido está preparado para resolver problemas onde as propriedades físicas ($k_x(x,y)$ e $k_y(x,y)$) variam somente em uma direção. Os resultados obtidos são apresentados na Tabela 5.6.

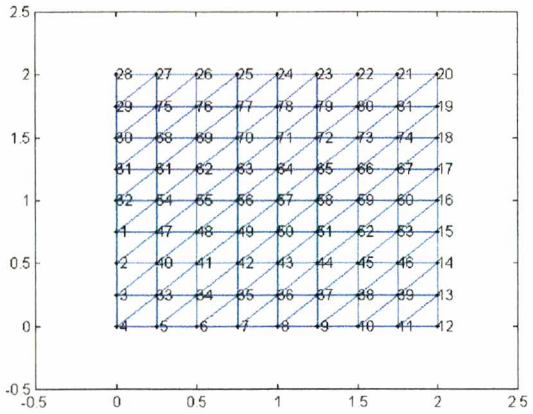


Figura 5.5 Malha 128 elementos – Problema 3.

Tabela 5.6 Resultados obtidos no Problema 3.

Nº. Elementos	h	$\|erro\|_{\infty}^{MEFL}$	$\|erro\|_{\infty}^{MEFQ}$
8	0.3536	2.4083 E-2	2.3615 E-3
32	0.1768	6.8189 E-3	5.2860 E-4
128	0.0884	1.8294 E-3	1.3261 E-4

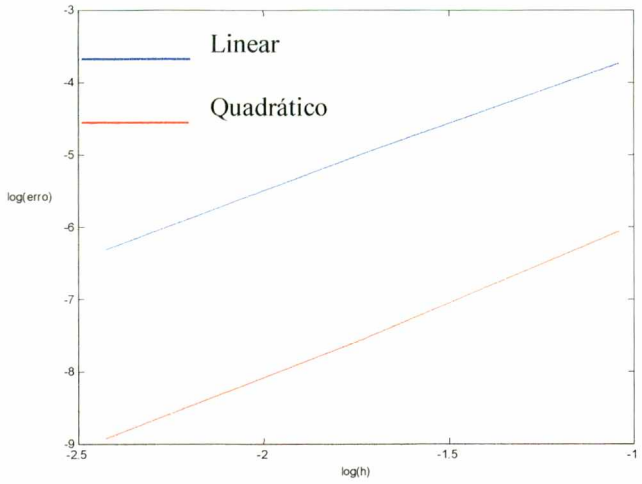


Figura 5.6 Evolução do erro para o Problema 3.

Pela Figura 5.6, observa-se que a taxa de convergência do erro para os casos linear e quadráticos é a mesma ($\theta(h^2)$).

5.5 Problema 4

O quarto problema analisado é descrito pela equação governante representada pela equação:

$$\frac{\partial}{\partial y}\left[xy\frac{\partial u}{\partial y}\right] = x^3 \tag{5.31}$$

Ou seja, é resolvida a equação de Poisson com os seguintes dados:

$$k_x(x,y) = 0 \tag{5.32}$$

$$k_y(x,y) = x\,y \tag{5.33}$$

$$f(x,y) = x^3 \tag{5.34}$$

A equação (5.31) está sujeita às seguintes condições de contorno:

$u(x, 0) = x^2 + 2$ $\quad u(x,2) = 2x^2 + 2$ $\quad$ para $\quad 0 \le x \le 2$ $\quad$ (5.35)

$u(0,y) = y + 2$ $\quad u(2,y) = 4y + 2$ $\quad$ para $\quad 0 \le y \le 2$ $\quad$ (5.36)

A solução analítica do problema representado pelas equações (5.20), (5.24) e (5.25) é conhecida e é dada por:

$$u(x, y) = x^2y + 2 \tag{5.37}$$

Na resolução deste problema utiliza-se três malhas computacionais com 8, 32 e 128 elementos, respectivamente, utilizando triângulos lineares e quadráticos. Neste caso, objetiva-se mostrar que o programa desenvolvido está preparado para resolver problemas onde as propriedades físicas ($k_x(x,y)$ e $k_y(x,y)$ variam somente em uma direção. Os resultados obtidos são apresentados na Tabela 5.7.

Tabela 5.7 Resultados obtidos no Problema 4.

Nº. Elementos	h	$\|erro\|_{\infty}^{MEFL}$	$\|erro\|_{\infty}^{MEFQ}$
8	0.3536	1.0321 E-2	2.0916 E-3
32	0.1768	2.6362 E-3	5.2916 E-4
128	0.0884	6.6259 E-4	1.3293 E-4

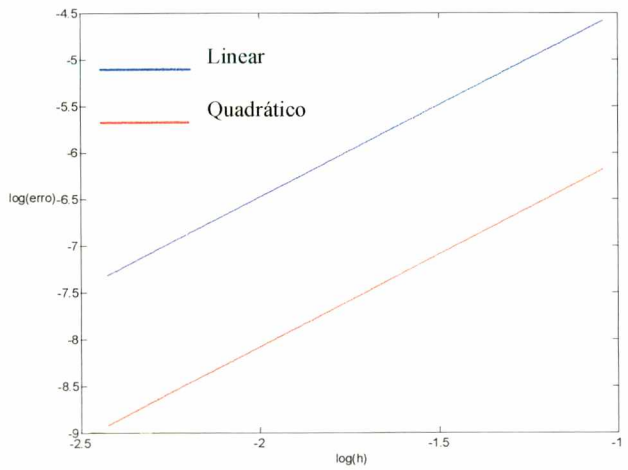


Figura 5.7 Evolução do erro para o Problema 4.

5.6 Problema 5

O quinto problema analisado é descrito pela equação governante representada pela equação:

$$\frac{\partial}{\partial x}\left[y\frac{\partial u}{\partial x}\right]+\frac{\partial}{\partial y}\left[x\frac{\partial u}{\partial y}\right]=\frac{y^2}{x}+\frac{x^2}{y} \tag{5.38}$$

Ou seja, é resolvida a equação de Poisson com os seguintes dados:

$$k_x(x,y)=y \tag{5.39}$$

$$k_y(x,y)=x \tag{5.40}$$

$$f(x,y)=\frac{y^2}{x}+\frac{x^2}{y} \tag{5.41}$$

Na resolução deste problema utiliza-se três malhas computacionais com 32, 64 e 128 elementos, respectivamente, utilizando triângulos lineares e quadráticos. Neste problema as propriedades físicas variam nas duas direções (x e y). Na Tabela 5.8 são apresentados os resultados obtidos. Na Figura 5.8 observa-se o erro obtido nas soluções em escala logarítmica.

A equação (5.38) tem as seguintes condições de contorno:

$u(x, 1) = x \ln x$ $\quad u(x,2) = x \ln(4x^2)$ $\quad$ para $\quad 1 \le x \le 2$ (5.42)

$u(1,y) = y \ln y$ $\quad u(2,y) = 2y \ln(2y)$ $\quad$ para $\quad 1 \le y \le 2$ (5.43)

Tabela 5.8 Resultados obtidos no Problema 5.

Nº. Elementos	h	$\|erro\|_{\infty}^{MEFL}$	$\|erro\|_{\infty}^{MEFQ}$
32	0.1768	2.7899 E-5	2.6566 E-5
64	0.0884	7.9296 E-6	1.8964 E-6
128	0.0442	2.0522 E-6	1.8528 E-7

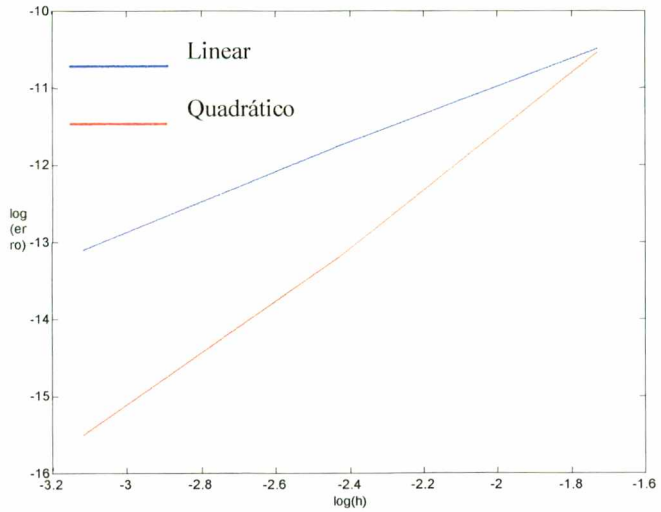


Figura 5.8 Evolução do erro para o Problema 5.

Conforme Tabela 5.8 e Figura 5.8, nota-se um bons resultados na resolução deste problema.

5.6 Problema 6

No problema 6 resolve-se a equação:

$$\frac{\partial}{\partial x}\left[x\frac{\partial u}{\partial x}\right]+\frac{\partial}{\partial y}\left[y\frac{\partial u}{\partial y}\right]=x+y \tag{5.44}$$

Ou seja, tem-se os seguintes dados:

$$k_x(x,y)=x \tag{5.45}$$

$$k_y(x,y)=y \tag{5.46}$$

$$f(x,y)=x+y \tag{5.47}$$

O problema é resolvido utilizando o domínio representado pela Figura 5.9.

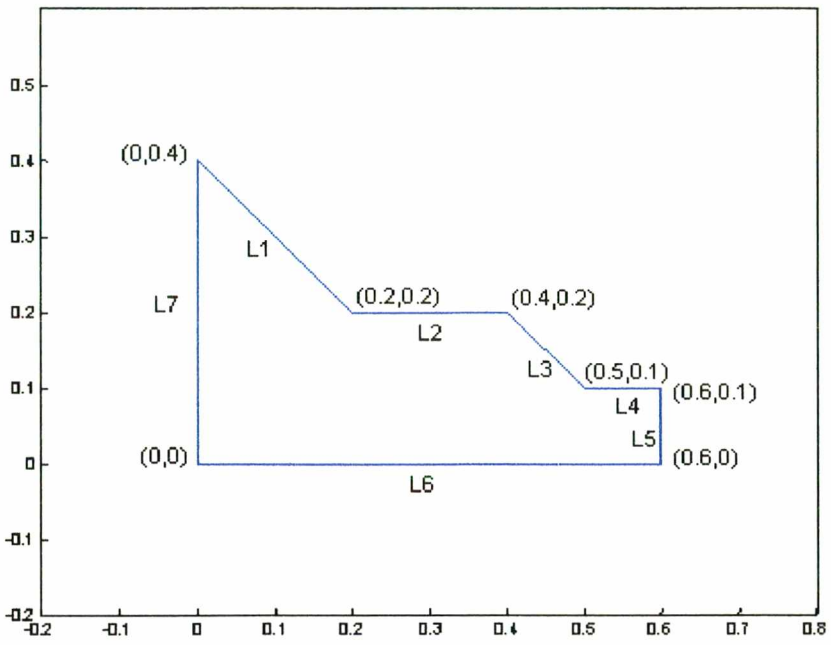


Figura 5.9 Domínio para o problema 6.

No contorno representado pela Figura 5.9, tem-se as seguintes condições de contorno:

$$u(x,y) = 4 \qquad \text{para (x,y)} \in \text{L6 e para (x,y)} \in \text{L7} \qquad (5.48)$$

$$u(x,y) = 0.2\,x + 4 \qquad \text{para (x,y)} \in \text{L2} \qquad (5.49)$$

$$u(x,y) = 0.6\,y + 4 \qquad \text{para (x,y)} \in \text{L5} \qquad (5.50)$$

$$\frac{\partial u}{\partial n}(x,y) = \frac{x+y}{\sqrt{2}} \qquad \text{para (x,y)} \in \text{L1 e para (x,y)} \in \text{L3} \qquad (5.51)$$

$$u(x,y) = 0.1\,x + 4 \qquad \text{para (x,y)} \in \text{L4} \qquad (5.52)$$

Onde $\frac{\partial u}{\partial n}(x,y)$ denota a derivada na direção normal ao contorno da Figura 5.9 nos pontos x e y.

Neste problema há a combinação de todas as irregularidades que podem estar contidas no problema, aplicadas a um domínio não convexo.

A solução analítica do problema é dada por u(x, y) = x y + 4. Para resolução deste problema utilizou-se a malha da Figura 5.10 (utilizando funções de interpolação lineares e quadráticas.

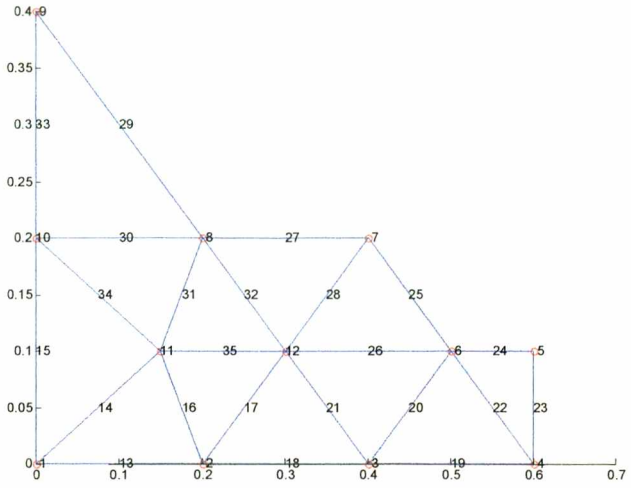


Figura 5.10 Malha para o Problema 6 (elementos quadráticos)

Utilizando funções de interpolação lineares foi obtido $\|erro\|_{\infty}^{MEFL}$ = 1.0714 E-3 e utilizando funções quadráticas obteve-se o $\|erro\|_{\infty}^{MEFQ}$ = 2.6645 E-15.

5.7 Conclusão

Com base nos resultados dos problemas analisados, foi verificado que o código computacional implementado está de acordo com o objetivo proposto e, portanto, está adequado à resolução de problemas práticos, os quais não possuem ou é muito difícil de encontrar soluções analiticamente.

Capítulo 6 – Descrição do Programa

6.1 Introdução

Neste capítulo é descrito o código computacional desenvolvido. O software escolhido para a implementação foi o MatLAB da MathWorks Laboratory. A escolha desta ferramenta foi baseada em experiências passadas, onde foi constatado que a utilização deste software diminui esforço de depuração de erro em relação a outras linguagens de programação.

Para um melhor entendimento do código computacional, o programa foi dividido em três módulos: Pré-Processamento, Processamento, Pós-Processamento. A Figura 6.1 ilustra as etapas realizadas em cada um dos módulos.

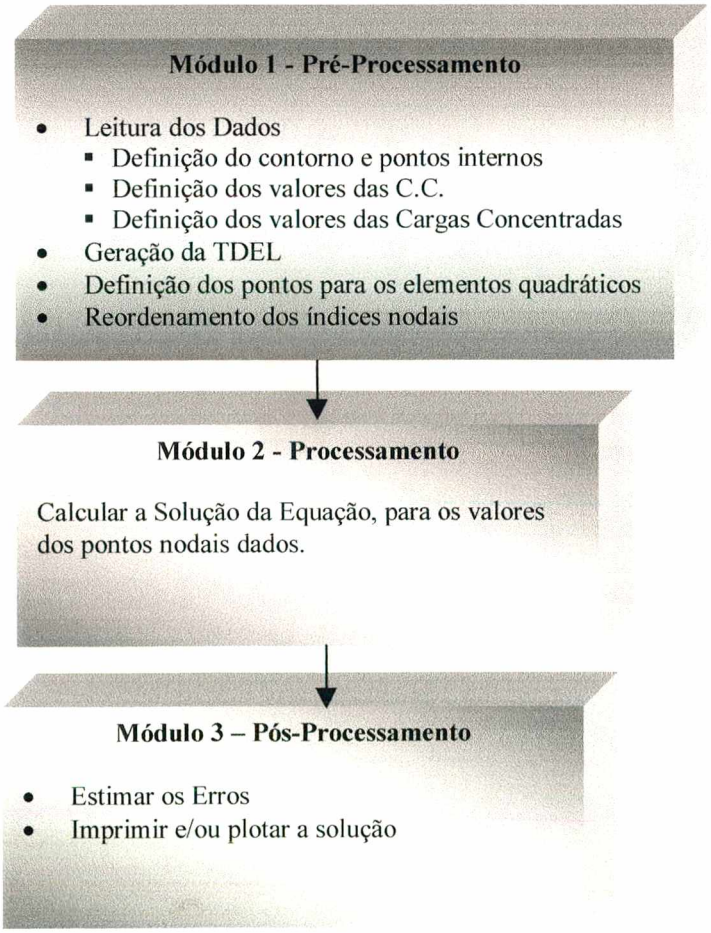


Figura 6.1 Módulos do Programa.

6.2 Módulo 1 – Pré-Processamento

Conforme citado no Capítulo 2, o código computacional implementado resolve inúmeros problemas cuja equação governante é a Equação de Poisson. No pré-processamento são determinados os seguintes dados do problema:

- Definição da malha;
- Definição das cargas concentradas;
- Definição das condições de contorno.

O fluxograma abaixo ilustra os procedimentos realizados neste módulo.

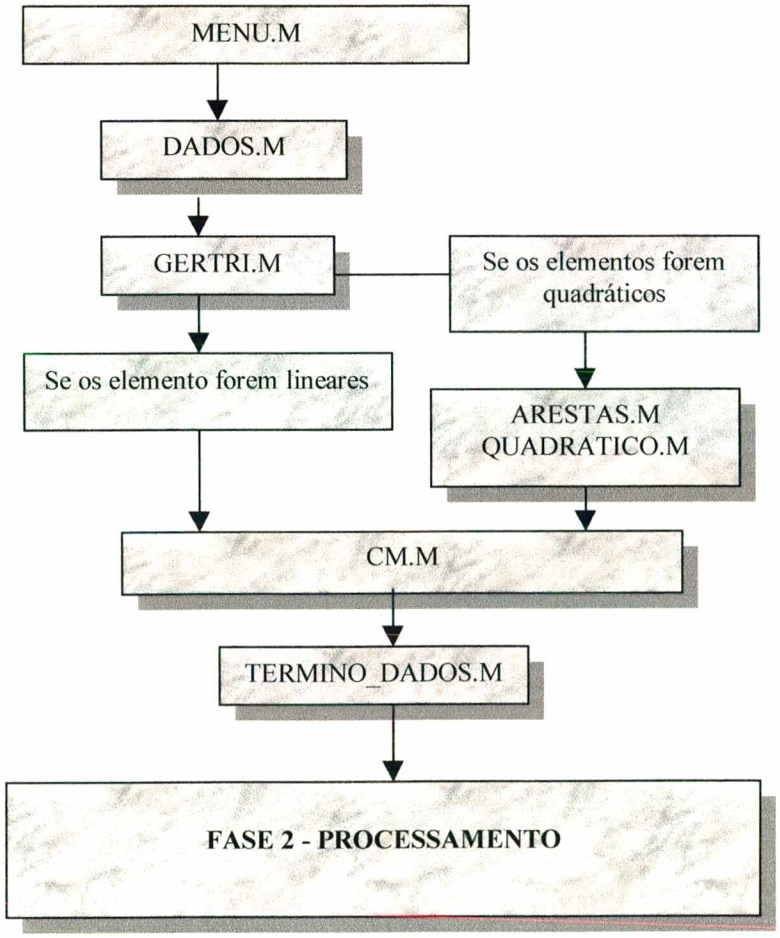


Figura 6.2 Fluxograma do Módulo 1 – Pré-Processamento.

Na Figura 6.2 são mencionados sete arquivos fontes (MENU.M, DADOS.M, GERTRI.M, ARESTAS.M, QUADRATICO.M, CM.M e TERMINO_DADOS.M), abaixo serão descritas as respectivas funções de cada um.

No arquivo "menu.m" (anexo 2) é onde se define os parâmetros do problema, como por exemplo, o tipo de elemento utilizado, a ordem da integração, etc. A etapa seguinte, para resolução de qualquer problema utilizando o código computacional implementado, é a definição da malha, ou seja, a definição do contorno e dos pontos internos ao contorno. São previstas três maneiras de entrada destes pontos. A primeira é informar os pontos em dois vetores: x e y, sendo que os primeiros "n_pontos" de cada vetor, referem-se aos "n_pontos" do contorno, os quais devem estar ordenados no sentido anti-horário, os demais são os pontos internos. Uma segunda forma alternativa é entrar com os pontos das extremidades do contorno em dois vetores x e y, definir um "passo" e executar um código computacional adicional implementado que é o "gerpto.m" (anexo 2.2), e então os pontos internos são gerados automaticamente a partir dos dados iniciais. A terceira alternativa é entrar com os pontos das extremidades do contorno. Em seguida é vizualizado na tela o contorno e com o clicar do mouse define-se os pontos internos. Após a entrada dos dados por uma das alternativas citadas, é executado o arquivo "gertri.m" conforme Figura 6.2. A principal função do arquivo "gertri.m" (anexo 3) é gerar a Triangulação de Delaunay (TDEL). Para geração da TDEL em domínio convexos está sendo utilizada a TDEL do *software* MATLAB, a qual tem a limitação de poder ser empregada somente para domínio convexos. Para domínios não-convexos, utiliza-se a TDEL implementada por Silva (1999). Neste módulo, ainda são determinados:

- O tamanho de cada elemento (parâmetro h), ou seja, é calculado o raio da circunferência circunscrita em cada um dos elementos, armazenando o raio do maior elemento (h_e), esta informação é útil na análise do erro;
- Uma lista dos pontos adjacentes de cada ponto da malha: esta informação é necessária para a realização do reordenamento dos índices nodais. A definição de pontos adjacentes encontra-se no capítulo 4.

Após gerar a TDEL, tem-se os dados suficientes para definir os elementos quadráticos, se for o caso, ou, se os elementos forem lineares, segue-se para o passo

seguinte (reordenamento). Caso os elementos sejam quadráticos, executa-se os arquivos "arestas.m" (anexo 4) e "quadrático.m" (anexo5). O primeiro cria uma lista de arestas da malha e o segundo calcula o ponto médio de cada aresta, acrescentando cada um destes pontos aos vetores x e y do arquivo "dados.m". Então passa-se à etapa seguinte, o reordenamento. Conforme o capítulo 4, o reordenamento tem a finalidade de reduzir a banda da matriz.

O algoritmo aplicado no reordenamento é o Cuthill-Mckee, que é baseado na Teoria de Grafos. O algoritmo e as definições básicas para o entendimento do mesmo foram apresentadas no capítulo 4.

Após realizar o reordenamento, tem-se a numeração de todos os nós. Neste momento deve-se informar os valores das condições de contorno e cargas concentradas no arquivo "termino_dados.m". Os dados citados são informados em vetores, contendo o número do nó e o respectivo valor da condição de contorno ou carga concentrada. Então, passa-se para a Fase 2 (Processamento), mas antes disso descreve-se a seguir os arquivos "gerpto.m", "arestas.m" e "quadratico.m".

A geração automática dos pontos (anexo 2.2) é muito importante na resolução de problemas. Isso porque os métodos numéricos são utilizados em malhas com um grande número de pontos (10.000 ou 50.000 pontos), o que dificulta a entrada de dados das duas outras formas previstas (coordenadas x e y, ou clicar do mouse). A seguir é descrita uma forma de gerar os pontos automaticamente.

Para um melhor entendimento do algoritmo, considere a Figura 6.3 (Figura não-convexa).

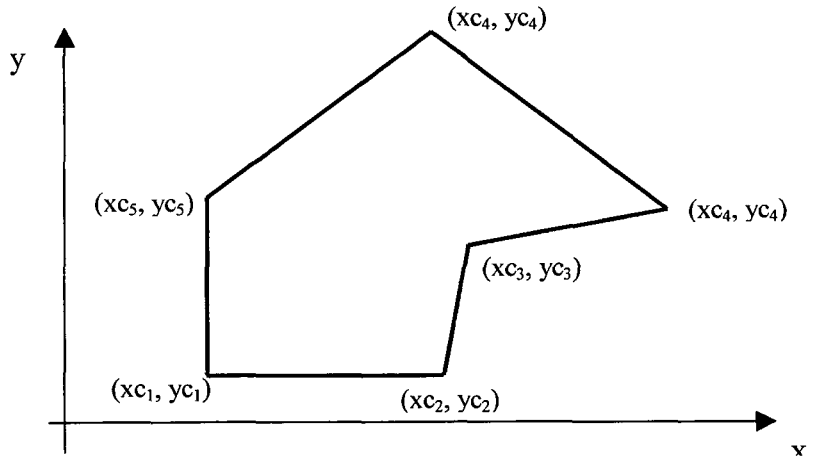


Figura 6.3 Primeiro exemplo para a geração automática dos pontos em um domínio não-convexo.

Para a geração automática dos pontos, é necessário informar os pontos das extremidades do contorno, no exemplo: {(xc_1, yc_1), (xc_2, yc_2), (xc_3,yc_3), (xc_4, yc_4), (xc_5, yc_5), (xc_6=xc_1, yc_6=yc_1)}. Outro valor necessário são as variáveis x0, x1 que são os menores valores em x, ou seja, é o segmento de reta ou o ponto onde começa a geração. Devem ser informados ainda os valores de y0, y1, que são respectivamente o menor e o maior valor em y. Para a Figura 6.3, x0 = xc_1, x1 = xc_2; y0 = yc_1; y1 = yc_4. Para o contorno da Figura 6.4, tem-se x0 = xc_1; x1 = xc_1; y0 = yc_1; y1 = yc_3 = yc_4.

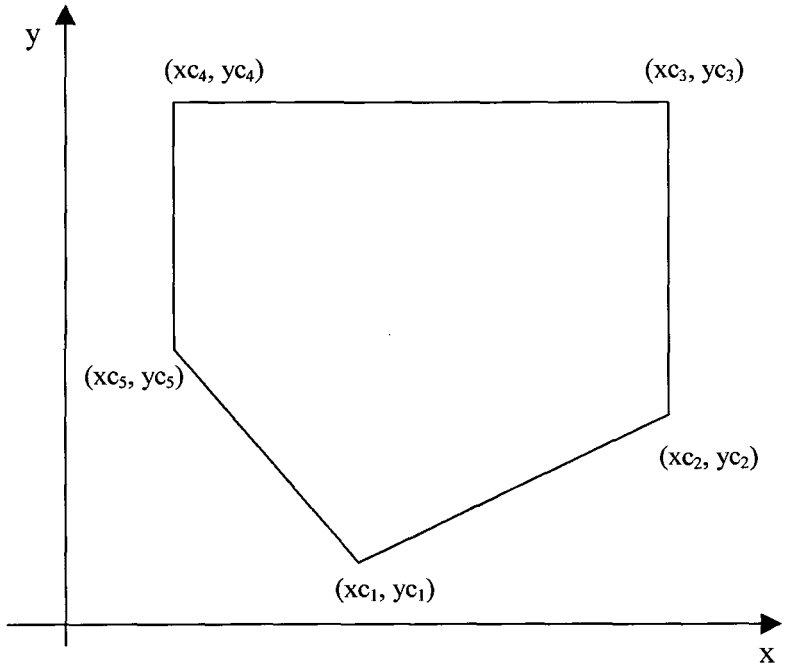


Figura 6.4 Segundo exemplo para geração automática dos pontos.

Para ambos os casos são informados os valores para hx e hy, que são respectivamente os valores dos "passos" ou "avanços" nas coordenadas x e y. A idéia central é fixar o ponto hy e dividir a distância do segmento de reta (x0, x1) por hx, obtendo os respectivos pontos em x, com os respectivos valores em y (que é o ponto hy = y0). Em seguida, incrementa-se "hy" e determina-se os novos valores de x0 e x1 e repete-se esta operação para todo o contorno.

Para encontrar os pontos x0 e x1, após cada "passo" do problema é empregado o Algoritmo Corta Contorno (anexo 2.2.1). Para melhor entender o algoritmo, considere a Figura 6.5 e suponha y0 = hy = 0.3

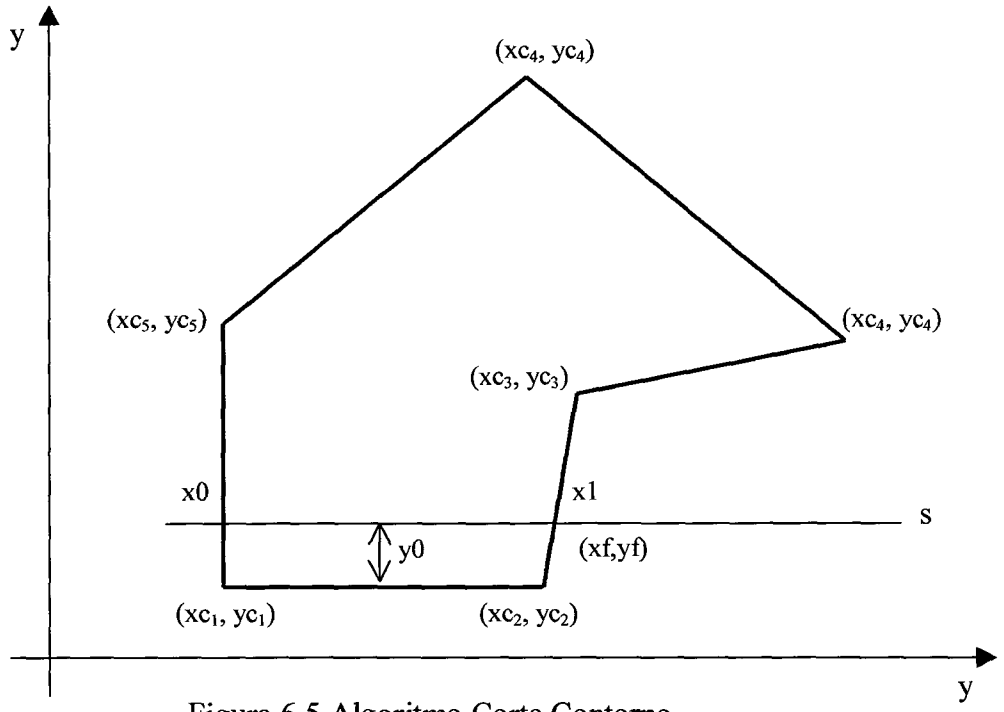


Figura 6.5 Algoritmo Corta Contorno.

A idéia geral para encontrar os pontos procurados (x0 e x1) é muito simples, pois dados os pontos das extremidades do contorno, pode-se facilmente identificar a equação de cada um dos segmentos de reta que formam o contorno. Em seguida, dado um ponto qualquer (no exemplo y0 = 0.3), traça-se uma reta paralela ao eixo x (reta s) com base no ponto y0, então é determinado os pontos x0 e x1. O algoritmo para encontrar os valores de x0 e x1, foi feito com base na definições abaixo.

Definição de Reta no Plano

Uma linha reta no plano é definida por três coeficientes A, B, C, tais que um ponto genérico p de coordenadas cartesianas X, Y está nessa linha se e somente se AX + BY + C = 0. Traduzindo essa equação para coordenadas homogêneas, conlui-se que um ponto finito p=[w, x, y] está nessa linha se e somente se A(x/w) + B(y/w) + C = 0, isto é, Ax + By + Cw = 0, ou, Cw + Ax + By = 0 (Resende, 1994).

Os dois lados da reta (lado positivo e negativo)

Os pontos [w, x, y] que não estão sobre uma reta r = [C, A, B] podem ser divididos em dois conjuntos, os *lados* da reta, conforme o sinal da expressão Cw + Ax + By. Este teste define o lado positivo e o lado negativo da reta r (Resende, 1994).

Cortes de Segmentos

Dado um segmento p_0p_1 e uma reta m, pode-se determinar onde este segmento corta a reta m. Em primeiro lugar, para se saber se o segmento de fato cruza a reta, basta verificar se seus extremos estão do mesmo lado da reta. Especificamente, se p_o = [w0, x0, y0], p1 = [w1, x1, y1] e m = [C, A, B], então basta calcular os números reais dados por:

$$\alpha_0 = Cw_0 + Ax_0 + Cy_0 \qquad (6.1)$$

$$\alpha_1 = Cw_1 + Ax_1 + Cy_1 \qquad (6.2)$$

O segmento p_0p_1 cruza (ou toca) a reta m se e somente se α_0 e α_1 tem sinais opostos, ou um deles é nulo; isto é, se $\alpha_0\ \alpha_1 <= 0$. Se esta condição é satisfeita, o ponto de intersecção pode ser determinado, através das propriedades de intersecção de retas.

O programa fonte apresentado no anexo 2.2.1 ("corta_contorno.m") tem a limitação de poder ser aplicado somente em contornos que possuem uma forma tal que, quando se traça qualquer reta paralela ao eixo x, esta reta corta o contorno em no máximo dois pontos, ou seja, ele não pode possuir a forma, por exemplo, da Figura 6.6.

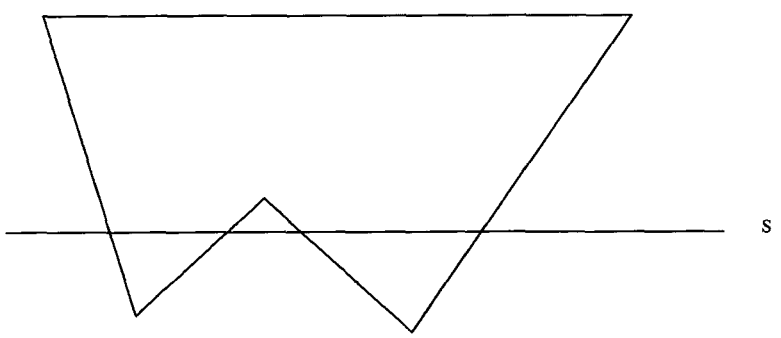


Figura 6.6 Forma de contorno Inválida para o Algoritmo Corta Contorno.

Como a reta s em análise é paralela ao eixo x, seu coeficiente angular (m) é igual a zero. Por esse motivo é que no algoritmo definimos m = 0, no caso m = 1.0e-20 (precisão da máquina) para evitar erros de underflow e overflow durante os testes. Um problema previsto no algoritmo é a situação em que a reta s corta o contorno exatamente em uma de suas extremidades, pois neste caso o produto dos testes será igual a zero. No entanto, um dos testes é diferente de zero, o que permite determinar os pontos (xf, yf).

6.3 Módulo 2 - Processamento

A idéia geral do MEF é que os cálculos essenciais sejam realizados para um elemento mestre, e depois a partir das informações dos elementos (coordenadas x e y) e o auxílio de transformações, calcula-se K_{ij}^{e} e F_i^{e}, armazenando os valores no sistema global. O fluxograma da Figura 6.7, mostra os passos executados neste módulo.

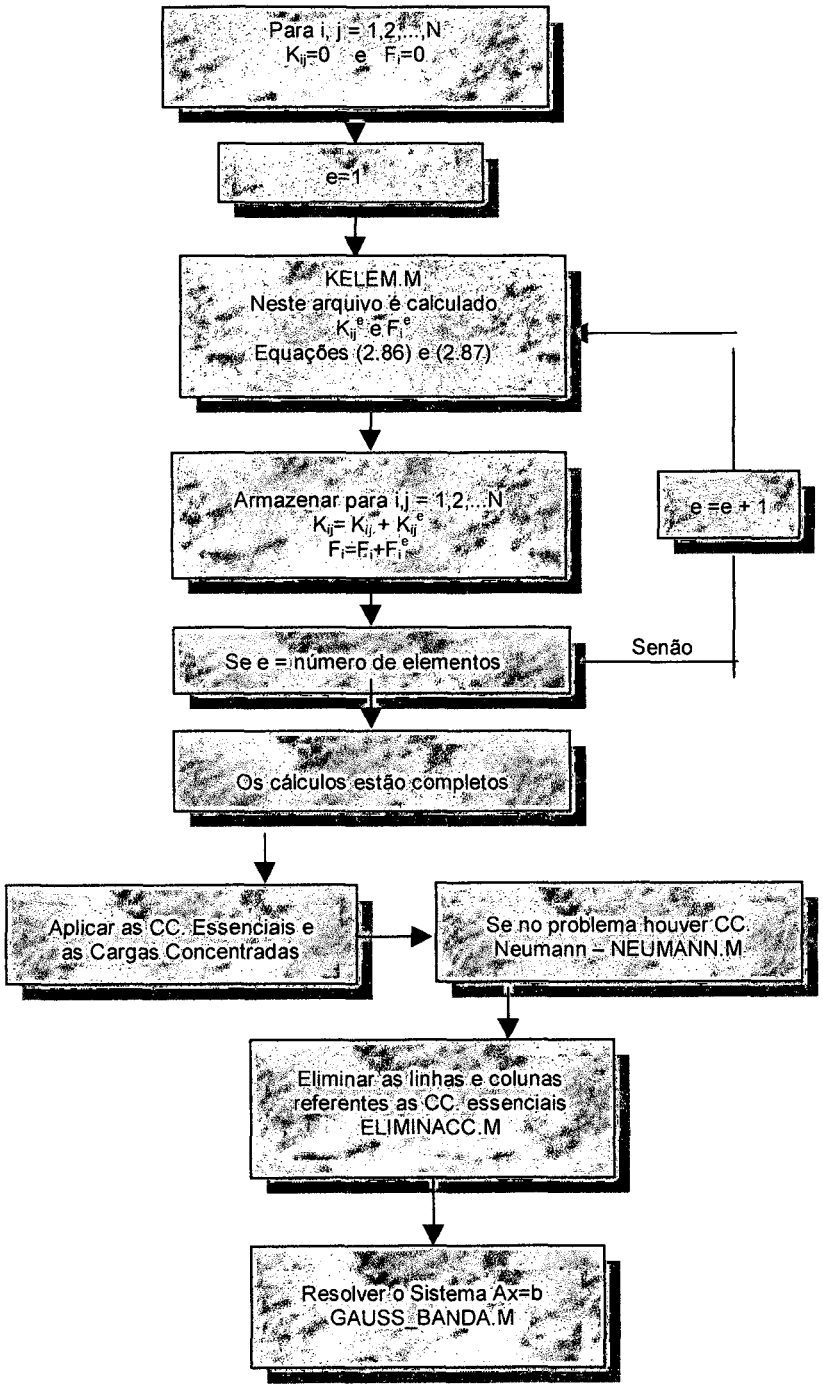


Figura 6.7 Fluxograma do Módulo de Processamento.

No problema em análise, a matriz de rigidez é dada pela equação (2.80). Após utilizar as transformações elementares, tem-se a equação (2.86), ou seja, neste momento todos os termos a serem integrados estão em coordenadas $\xi(0,1)$ e $\eta(0,1)$. Como não se consegue resolver esta integral de forma analítica para elementos de grau maior que o

linear, recorre-se então para a integração numérica. Tradicionalmente no MEF, utiliza-se a quadratura gaussiana para tal problema, um procedimento que é realizado no arquivo "kelem.m" (anexo 8). Para realizar esta integração, informa-se os pontos de integração para ξ e η com os respectivos pesos, que são repectivamente os pontos e pesos definidos na regra de quadratura. Na tabela 4.1 são dados os valores para as ordens de integrações quadrática, cúbica e quíntica. A partir dos dados de cada elemento, resolve-se a integral elementar obtendo a matriz K_{ij}^{e} e analogamente F_i^e. Em seguida armazena-se os valores nos respectivos nós no sistema global, um exemplo do armazenamento é visto no capítulo 4.

A partir das coordenadas globais dos elementos, tem-se os valores de x e y para cada elemento. A partir da transformação dada pela equação (2.72) são obtidos os valores de $x(\xi, \eta)$ e $y(\xi, \eta)$, os quais são utilizados na determinação de $k_x(x,y)$, $k_y(x,y)$ e f(x,y). Ressalta-se que a discretização da equação governante foi feita com base na equação (2.1). Portanto, se a equação a ser resolvida estiver na forma da equação (6.3), o valor de f(x,y) deve ser armazenado com o sinal oposto.

$$\frac{\partial}{\partial x}\left[k_x(x,y)\frac{\partial u}{\partial x}\right]+\frac{\partial}{\partial y}\left[k_y(x,y)\frac{\partial u}{\partial y}\right]=f(x,y) \tag{6.3}$$

Após realizar os passos acima descritos, tem-se o sistema global que representa a solução no domínio como um todo. Neste momento são impostas as condições de contorno e as cargas concentradas conforme descrito no capítulo 2.

No arquivo "termino_dados.m" os nós com as condições de contorno essencias e os respectivos valores são informados em dois vetores (c_essencial, valor_c_essencial), analogamente são informados os nós com cargas concentradas. Conforme visto na discretização (capítulo 2), as cargas concentradas devem ser simplesmente adicionadas ao vetor F, ou seja, em F(i) onde i representa o nó com carga concentrada.

Condições de Contorno

No capítulo 2, foi exemplificada uma maneira de impor as condições de contorno. Optou-se por eliminar as linhas onde são conhecidos os valores das CC. essenciais, pois

assim diminui-se o tamanho do sistema de equações algébricas a ser resolvido. No entanto, há outras maneiras de impor as condições de contorno essenciais, como por exemplo, colocando zeros na linha e coluna onde deve ser inserida a condição (matriz **K**) e o valor 1 no termo da diagonal, e em F coloca-se o valor da CC. No arquivo "programaprincipal.m" pode-se ver como foi implementada a condição de contorno essencial no código computacional desenvolvido. Ressalta-se que os pontos do contorno estão armazenados em um vetor, os quais são determinados por uma rotina do próprio Matlab (convhull), e os números dos nós que estão sujeitos às condições de contorno essenciais, estão armazenados no vetor "c_essencial" e seus respectivos valores no vetor "valor_essencial".

Após a imposição das condições de contorno essenciais, deve-se impor as condições de contorno naturais. Este procedimento é necessário somente quando os valores das CC. naturais forem diferentes de zero. Quando isso ocorrer, é realizada a integração de linha apresentada no capítulo 2, conforme equação (2.88). Na aplicação das condições de contorno de Neumann (naturais), considere a Figura 6.8.

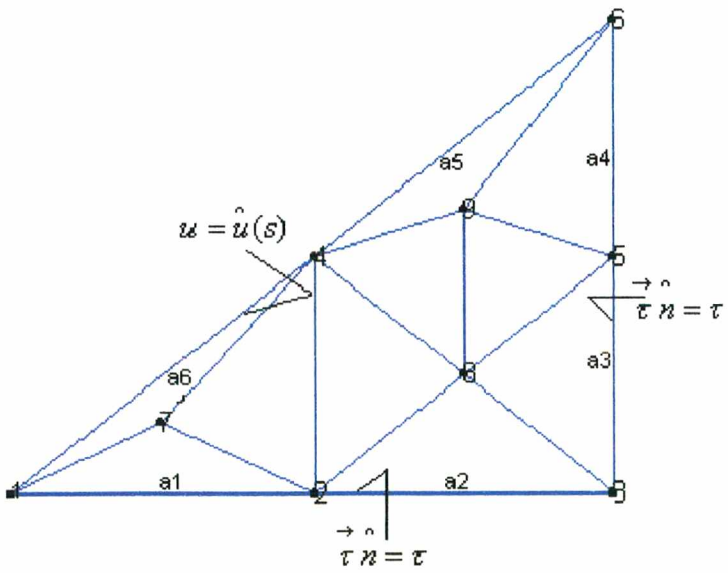


Figura 6.8 Condições de Contorno de Neumann.

Quando há condições de contorno naturais, deve-se informar no arquivo "termino_dados.m" as arestas com as referidas CC.. No exemplo da Figura 6.8 armazena-se na matriz "c_natural" as arestas [1 2; 2 3; 3 5; 5 6] e os respectivos valores

na matriz valor_natural [τ_1 τ_2; τ_2 τ_3; τ_3 τ_5; τ_5 τ_6]. O programa fonte da imposição das condições de contorno naturais é o arquivo "neumann.m" (anexo 9).

Após calcular os valores das condições de contorno naturais (fluxos), estes valores são adicionados a F. O próximo passo é a eliminação das linhas e colunas da matriz K, cuja implementação dos valores das condições de contorno essenciais é feita pelo arquivo "eliminacc.m". Os nós submetidos a condições de contorno essenciais estão armazenados em ordem crescente no vetor "c_essencial". Na matriz KG e no vetor FG estão os valores do sistema global. Como saída tem-se a matriz KGNN e FGN, que representam o sistema de equações algébricas e ser resolvido.

O passo seguinte é a resolução do Sistema de Equações Algébricas, obtendo-se assim a solução do problema. Nota-se no problema que está sendo resolvido que a matriz global é um sistema com as seguintes características: Simétrico; Positivo Definido e Banda, conforme descrito no capítulo 4. Uma das formas de resolver o sistema (Ax = b) é usar as rotinas do próprio MATLab para a resolução. No entanto, optou-se em implementar a Eliminação Gaussiana para este sistema, armazendo-se somente "meia banda" conforme descrito no capítulo 4. Um primeiro procedimento para aplicar esta forma de resolução é identificar o tamanho da banda e em seguida, armazenar somente "meia-banda" do sistema global (KG).

O algoritmo para encontrar o tamanho da banda é baseado nas informações de cada Triângulo de Delaunay, ou seja, quando é gerada a triangulação, é armazenada uma matriz onde cada linha refere-se aos números de cada elemento. É importante notar que os elementos, contendo algum nó no contorno e possuindo condições de contorno essenciais devem ser desconsiderado, pois estes nós foram eliminados do sistema. Assumindo que todos os elementos estão armazenados em uma matriz "tri", pode-se determinar a solução do problema pelo arquivo "gauss_banda.m".

6.4 Módulo 3 – Pós-Processamento

Esta é a etapa em que deve ser feita a análise a posteriori do erro. Esta análise, embora descrita no capítulo 4, não foi implementada no código computacional utilizado

no presente trabalho. No entanto, é realizada uma estimativa a priori do erro (baseadas nas soluções exatas dos problemas).

6.5 Conclusão

Neste capítulo foi descrito os procedimentos do desenvolvimento do código computacional implementado e foram apresentadas as definições necessárias para a construção do programa. Em anexos são apresentados os códigos fonte.

Capítulo 7 - Resultados

7.1 Introdução

Neste capítulo são apresentados alguns resultados com base em três artigos. O objetivo é comparar o código computacional implementado com outras metodologias de resolução.

O primeiro trabalho reproduzido é o de Zlámal (1969) que utilizou o MEF para a resolução da Equação de Poisson utilizando funções de interpolação cúbicas. O problema por ele resolvido está sujeito a condições de contorno de Neumann e Dirichlet. A solução analítica e o termo fonte do problema são polinômios de terceiro grau, logo, utilizando funções de interpolação cúbicas foram obtidos excelentes resultados.

Logo em seguida, o código é utilizado para reproduzir o trabalho de Oliveira and Silva (1999) que resolveram a Equação de Poisson em um domínio quadrado por um método semi-analítico, representando a solução por Séries de Fourier, com a finalidade de remover o fenômeno de Gibs os coeficientes são pós-processados por polinômios de Gegenbauer. Este trabalho é uma extensão do trabalho de Gottlieb *et al.* (1992) para funções bidimensionais ($f \in L_2([-1, 1] \times [-1, 1])$), na resolução foi obtida convergência espectral, ou seja, o erro tende a zero exponencialmente.

Vozovoi *et al.* (1998) apresentaram um algoritmo para resolver de maneira direta a equação de Poisson em regiões retangulares. O método é baseado na aproximação "pseudo-spectral" de Fourier e técnica de subtração polinomial, o procedimento de resolução é essencialmente analítico e possui uma alta ordem de precisão. Este método é aplicável à resolução de problemas onde a função fonte é irregular, ou então quando fontes randômicas estão localizadas em determinadas regiões do domínio. São apresentados gráficos e Tabelas comparando as soluções.

Utiliza-se a notação $\|erro\|_{\infty}^{MEFL}$ para representar os erros obtidos na norma infinito utilizando elementos lineares e, $\|erro\|_{\infty}^{MEFQ}$ para representar os erros utilizando funções

quadráticas.

7.2 Problema 1

No primeiro trabalho, Zlámal (1969) resolveu a seguinte equação:

$$-\frac{\partial^2 u}{\partial x^2}-\frac{\partial^2 u}{\partial y^2}=2x^2(3-2x)+6y(1-2x)(-2+y) \tag{7.1}$$

Da equação (7.1) pode-se facilmente deduzir os seguintes dados:

$$k_x(x,y)=1 \tag{7.2}$$

$$k_y(x,y)=1 \tag{7.3}$$

$$f(x,y)=2x^2(3-2x)+6y(1-2x)(-2+y) \tag{7.4}$$

A equação (7.1) está sujeita as seguintes condições de contorno:

$$u(x,0)=0 \qquad \frac{\partial u}{\partial y}(x,1)=0 \qquad \text{para} \quad 0\le x\le 1 \tag{7.5}$$

$$\frac{\partial u}{\partial x}(0,y)=0 \qquad \frac{\partial u}{\partial x}(1,y)=0 \qquad \text{para} \quad 0\le y\le 1 \tag{7.6}$$

Zlámal resolveu este problema utilizando duas malhas computacionais, com 32 e 128 triângulos pelo Método dos Elementos Finitos utilizando funções de interpolação cúbicas. Foram reproduzidas as mesmas malhas computacionais e o problema foi resolvido utilizando funções de interpolação lineares e quadráticas. Zlámal apresenta a sua solução em alguns pontos estratégicos, os quais também foram analisados neste trabalho. Na Tabela 7.1 são apresentados os resultados.

Tabela 7.1 Resultados obtidos nos pontos analisados por Zlámal.

		(1/2, 1/2)	(1/4, 3/4)	(1/2, 3/4)	(1/4,3/4)
Solução Exata	u(x,y)	0.375000	0.117187	0.468750	0.146484
Zlámal (1969)	32 triângulos	0.375120	0.117248	0.468829	0.146518
	128 triângulos	0.375008	0.117190	0.468754	0.146483
$\|erro\|_{\infty}^{MEFQ}$	32 triângulos	0.374747	0.116749	0.468599	0.146176
	128 triângulos	0.374984	0.117160	0.468740	0.146465
$\|erro\|_{\infty}^{MEFL}$	32 triângulos	0.376151	0.122553	0.465677	0.150378
	128 triângulos	0.375325	0.118520	0.467975	0.147514

Conforme pode-se observar na Tabela 7.1, os dados obtidos por Zlámal e os dados obtidos no presente trabalho possuem uma concordância. E, mesmo utilizando funções de interpolação lineares e quadráticas chega-se a ótimos resultados. Na Tabela 7.2 são apresentados os máximos erros obtidos neste trabalho (funções lineares e quadráticas) e os erros obtidos por Zlámal (funções cúbicas).

Tabela 7.2 Resultados obtidos para o Problema 1.

Número de Elementos	h	$\|erro\|_{\infty}^{MEFL}$	$\|erro\|_{\infty}^{MEFQ}$	$\|e\|_{\infty}$ - Zlámal Elem. Cúbicos
32	0.1768	2.7864 E-2	2.1337 E-3	5.2 E-4
128	0.0884	1.1027 E-4	2.3844 E-4	2.5 E-5

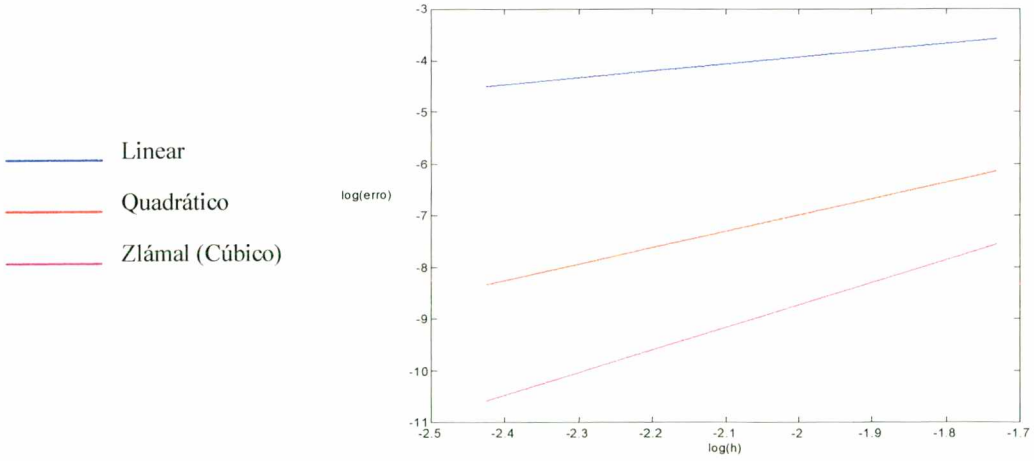


Figura 7.1 Evolução do erro para o Problema 1.

7.3 Problema 2

Oliveira e Silva (1999) resolveram a equação (7.7)

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 6xy(x^2 + y^2 - 2) \tag{7.7}$$

Ou seja, foi resolvida a equação de Poisson com os seguintes dados:

$$k_x(x, y) = 1 \tag{7.8}$$

$$k_y(x, y) = 1 \tag{7.8}$$

$$f(x, y) = 6xy(x^2 + y^2 - 2) \tag{7.10}$$

Tabela 7.3 Comparação dos resultados obtidos

N	$\|e\|_\infty^G$	$\|e\|_\infty^F$	$\|e\|_\infty^{FPS}$	$\|erro\|_\infty^{MEFL}$	$\|erro\|_\infty^{MEFQ}$
64	3.8710 E-1	7.3868 E-4	1.1474 E-2	1.6615 E-2	1.1179 E-3
144	1.8861 E-6	3.5659 E-4	3.2005 E-3	6.6379 E-3	2.4991 E-4
256	4.0786 E-8	2.0747 E-4	1.2942 E-3	5.6430 E-3	1.5481 E-4

Onde, N refere-se ao número de pontos da malha, (G) é o erro obtido por Oliveira and Silva com pós-processamento do coeficientes, (F) é o erro obtido por Oliveira and Silva sem pós-processar os coeficientes, (FPS) são os resultados obtidos de IMSL Fast Poisson Solver (diferenças finitas de quarta ordem). Conforme referenciado na introdução deste capítulo, o método implementado por Oliveira and Silva possui convergência espectral e é um método semi-analítico de resolução de equações diferenciais parciais. No entanto, pode-se notar que o MEF é superior ao MDF utilizando elementos quadráticos.

7.4 Problema 3

O problema resolvido por Vozovoi *et al.* (1998) é descrito pela equação:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 4\mu(\mu r^2 - 1)e^{-\mu r^2} \tag{7.11}$$

Ou seja, é resolvida a equação de Poisson com os seguinte coeficientes:

$$k_x(x, y) = 1 \tag{7.12}$$

$$k_y(x, y) = 1 \tag{7.13}$$

$$f(x, y) = 4\mu(\mu r^2 - 1)e^{-\mu r^2} \tag{7.14}$$

Onde $\mu = 200$ e $r^2 = x^2 + y^2$, o problema é resolvido no domínio $\Omega = [-0.5, 0.5]^2$ com condições de contorno de Dirichlet. A solução deste problema é conhecida e é dada pela equação (7.15).

$$u(x, y) = e^{-\mu r^2} \tag{7.15}$$

Na Figura 7.1 pode-se ver o gráfico do termo fonte deste problema.

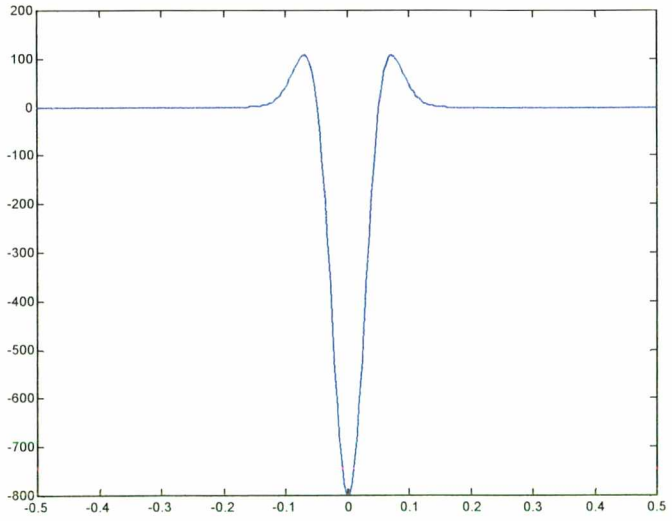


Figura 7.2 Termo fonte do problema 3.

Conforme pode ser observado, esta é uma função complicada de ser representada. Este problema é resolvido utilizando duas malhas computacionais com a mesma quantidade de pontos utilizados por Vozovoi *et al.* Os resultados são apresentados na Tabela 7.4.

Tabela 7.4 Comparação dos Resutados problema 3

N	h	$\|e\|_\infty$ - Método GF Vozovoi *et al.*	$\|erro\|_\infty^{MEFL}$	$\|erro\|_\infty^{MEFQ}$
256	0.0472	2.1 E-2	5.9815 E-2	1.9299 E-2
1024	0.0236	1.7 E-7	2.1493 E-2	6.9241 E-3

onde N é o número de pontos da malha.

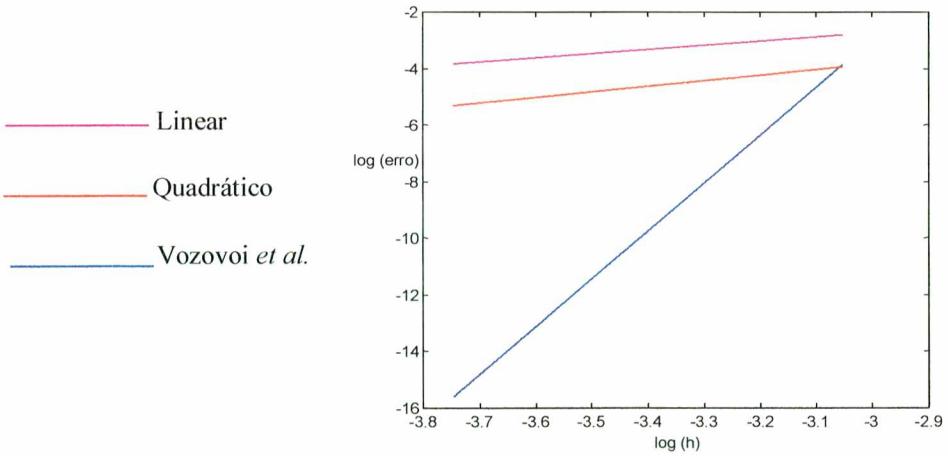


Figura 7.3 Evolução do erro para o Problema 3.

Pela Figura 7.3 pode-se observar uma alta taxa de convergência obtida por Vozovoi *et al.*

7.5 Conclusão

Comparando o programa implementado com outros trabalhos científicos foram obtidos resultados muito próximos. A taxa de convergência dos trabalhos de Oliveira and Silva (1999) e Vozovoi *et al.* (1998) são bem melhores que as obtidas neste

trabalho. Porém, as metodologias de solução apresentadas naqueles artigos são essencialmente analíticas e desta forma não podem ser utilizadas em domínios retangulares.

Capítulo 8 - Considerações Finais

Para a realização deste trabalho, foram estudados os pontos chave da análise numérica aplicada à resolução das equações diferenciais parciais (EDPs). Este foi o primeiro trabalho do Curso de Pós-Graduação em Ciência da Computação (CPGCC) utilizando o Método dos Elementos Finitos (MEF) como metodologia de resolução dessa classe de equações. Por esse motivo foi realizado um levantamento bibliográfico de todos os aspectos envolvidos na resolução, os quais foram apresentados ao longo do trabalho.

Neste trabalho implementou-se um código computacional utilizando o MEF, capaz de resolver problemas envolvendo a equação de Poisson em domínio não-convexos e sujeitos a condições de contorno de Neumann e Dirichlet, podendo ainda ter diferentes propriedades físicas no domínio e quaisquer pontos com cargas concentradas e/ou distribuídas.

Foi apresentada a fundamentação teórica da geração de malhas, dando ênfase na Triangulação de Delaunay, todos os conceitos fundamentais para a construção de um algoritmo utilizando esta técnica são expostos.

Encontrou-se na literatura diversos trabalhos sobre reordenamento dos índices nodais, os quais foram referenciados no Capítulo 1, sendo que foi implementado somente uma técnica de reordenamento (o algoritmo CM), esta é uma área importante dentro do MEF, mas não é o objetivo principal deste trabalho. Segundo Liu and Shermann (1976), na aplicação do algoritmo CM, nunca é gerado um envelopamento (largura de banda) maior do que o original, por esse motivo é que foi apresentado os resultados somente para um problema teste.

Alguns procedimentos numéricos e os conceitos fundamentais para a realização de estimativas de erros foram apresentados, na validação do problema explorou-se alguns desses resultados.

As definições necessárias para a construção do programa implementado foram descritas, comparando os resultados deste trabalho com outros trabalhos científicos, observou-se ótimos resultados. A taxa de convergência dos trabalhos de Oliveira e Silva

(1999) e Vozovoi *et al.* (1998) são melhores que as obtidas neste trabalho. Porém, as metodologias de solução apresentadas naqueles artigos são essencialmente analíticas e possuem a limitação de poder serem utilizadas somente em domínios retangulares. Então, com base nos resultados, foi verificado que o código computacional implementado está de acordo com o objetivo proposto e, portanto, está adequado à resolução de problemas práticos, os quais não possuem solução analítica.

Investigou-se ainda, outros aspectos intermediários, como a integração numérica e a resolução de sistemas lineares.

8.1 Perspectivas Futuras

A partir do presente trabalho, muitas outras aplicações utilizando o MEF poderão ser desenvolvidas, como por exemplo:

1) estudar e implementar novas técnicas de ordenações dos pontos nodais: este foi um aspecto considerado neste trabalho, porém esse é um estudo que necessita de um maior aprofundamento; um excelente trabalho nesta área é o de Paulino *et al.* (1994), além das técnicas de reordenamento clássicas que são as bases para qualquer estudo deste tipo.

2) Estudar outras metodologias de resolução do sistema de equações algébricas: neste trabalho foi implementado um método de resolução direto explorando as características da banda da matriz. É necessário a implementação de métodos de resolução iterativos, como os métodos da família do Gradiente Conjugado, traçando um paralelo sobre as duas técnicas.

3) Implementar as estimativas de erros *a posteriori* apresentadas no Capítulo 4 e utilizar esta informação como critério de refinamento da malha (Métodos Adaptativos).

Referências Bibliográficas

ADAMIK, V. and MATEJOVIC, P., 1989, "A diffusion equation with hourglass control in na axisymmetric geometry", Computer Methods in Applied Mechanics and Engineering, vol. 76, nº. 2, pp. 135-156.

ARMSTRONG, B. A., 1984, "A hybrid Algorithm for reducing matrix bandwidth", International Journal for Numerical Methods in Engineering, vol. 20, pp. 1929-1940.

BABUSKA, I. and RHEINBOLDT, W.C., 1978a, "Error estimates for adaptative finite element computations", SIAM J. Numer. Anal., vol. 15, pp. 736-754.

BABUSKA, I. and RHEINBOLDT, W.C., 1978b, "A posteriori erro estimates for the finite element method", International Journal for Numerical Methods in Engineering, vol. 12, pp. 1597-1607.

BANK, R. E., WEISER, A., 1985, "Some a posteriori error estimators for elliptic partial differential equations" , Mathematics of Computations, vol. 44, pp. 283-301.

BRAESS, Dietrich, 1997, "Finite Elements: Theory, Fast Solvers, na Applications in Solid Mechanics", Cambridge University Press.

CAMPBELL C. and REDFERN, D., 1997, "The Matlab 5 Handbook", Springer-Verlag, New Jersey.

CAREY, G. F. and ODEN, J. T., 1981, "Finite elements – An Introduction", Prentice-Hall, New Jersey, vol. I.

CAREY, G. F. and ODEN, J. T., 1983, "Finite elements – Mathematical Aspects", Prentice-Hall, New Jersey, vol. IV.

CAREY, G. F. and ODEN, J. T., 1984, "Finite elements – Computational Aspects", Prentice-Hall, New Jersey, vol. III.

CIARLET, P. G., 1978, "The Finite Element Method for Elliptic Problems", North-Holland, Amsterdan.

DAICHAO, S., KENNET, B. A. and SVEN, K., 1992, "Finite element analysis for convective heat diffusion with phase change", Computer Methods in Applied Mechanics and Engineering, vol. 104, pp. 19-30.

Demo program for Delaunay Triangulation and Voronoi Diagram [on line]. Disponível na internet. http://itp-www.colorado.edu/~tlen5510/delaunay/delaunay.html. 20 fev 1999.

DRACOPOULOS, M. C. and CRISFIELD, M. ª, 1995, "Coarsel/Fine Mesh Preconditioners for the Iterative Solution of Finite Element Problems", International Journal for Numerical Methods in Engeneering, vol. 38, pp. 3297-3313

ERIKSSON, K. and JOHNSON, C., 1988, "An adaptive finite element method for linear elliptic problems", Mathematics of Computations, vol. 50, nº. 182, pp. 361-383.

FLETCHER, C. A. J., 1997, "Computational Techniques for Fluid Dynamics", Springer-Verlag, New York, 3ª. ed.

FOMIN, S. V., GELFAND, I. M., 1963, "Calculus of Variations", Moscow State University, Prentice-Hall, New Jersey

GOLUB, G. H. and VAN LOAN, C. F., 1996, "Matrix Computations". The Johns Hopkins University Press, 3ª. edition

HAN, R. P. S. and SCOTT, D. G., 1994, "Basis of na improved hybrid node renumbering algorithm for matrix bandwidth reduction", Computers Methods in Applied Mechanics and Engineering, vol. 118, pp. 309-318.

HAQUE, K. A. and STROUBOULIS, T., 1992, "Recent experinces with error estimation and adaptivity, Part II: Error estimation for h-adaptive approximations on grids of triangles and quadrilaterals", Computer Methods in Applied Mechanics and Engineering, vol. 100, nº. 3, pp. 359-430.

HARARI, I. and HUGHES, T. J. R., 1994. "Stabilized finite element methods for steady advection-diffusion with production", Computer Methods in Applied Mechanics and Engineering, vol. 155, nº. 1-2, pp. 167-191.

HUGHES, T. J. R. and STEWART, J. R., 1998, "A tutorial in elementary finite element error analysis: A systematic presentation of a priori and a posteriori erros estimates", Computer Methods in Applied Mechanics and Engineering, vol. 158, nº. 1-2, pp. 1-22.

JOHNSON, C., 1990, "Adaptive finite element methods for diffusion and convective problems", Computer Methods in Applied Mechanics and Engineering, vol. 82, nº. 1-3 pp. 301-322.

JOHNSON, C., 1994, "Numerical solution of partial differential equations by finite element method", Denmark & Iceland, New York, 5ª. ed.

KOO, B. U. and LEE, B. C., 1992, "An efficient profile reduction algorithm based on the frontal ordering scheme and the graph-theory", Computers & Structures, vol. 44, pp. 1339-1347.

KREITH, Frank, 1988, "Princípios da Transmissão de Calor", Edgard Blücher Ltda, São Paulo, 5ª. ed.

LIU, W. H. and SHERMAN, A. H., 1976, "Comparative analysis of the Cuthill-McKEE and the Reverse Cuthill-McKEE ordering algorithms for sparse matrices", SIAM J. Numer. Anal., vol. 13, nº. 2, pp. 198-213.

MALISKA, C. R., 1995, "Transferência de calor e mecânica dos fluidos computacional". Livros Técnicos e Científicos Editora, Rio de Janeiro.

MARIANI, V. C., 1997, "Resolução de Sistemas Lineares Gerados na Discretização das Equações de Navier-Stokes em Malhas de Voronoi", Dissertação de Mestrado, Universidade Federal de Santa Catarina, Florianópolis.

MATLAB - versão do estudante: guia do Usuário. São Paulo: Makron Books do Brasil, 1997.

MEDEIROS, L. H. A., 1994, "Elementos infinitos mapeados e elementos virtuais no cálculo de campos eletromagnéticos em problemas com fronteiras abertas pelo método de elementos finitos", Dissertação de Mestrado, Universidade Federal de Santa Catarina, Florianópolis.

O'ROURKE, Joseph, 1993, "Computational Geometry in C", Cambridge University Press, New York.

ODEN, J. T., 1972, "Finite Elements of Nonlinear Continuo", McGraw-Hill, New York.

OLIVEIRA, J. C. and SILVA, A. V., 1999, "Numerical investigation of the Fourier-Gegenbauer method applied to boundary value problems", Congresso Íbero Latino Americano de Métodos Computacionais para Engenharia – CILAMCE (*No prelo*).

OLIVEIRA, Jáuber C. "Elementos Finitos II", setembro a dezembro de 1999. Notas de Aula.

PAULINO, G. H., *et al.*, 1994, "Node and element resequencing using the laplacian of a finite-element graph.1. general concepts and algorithm", International Journal for Numerical Methods in Engineering, vol. 37, pp. 1511-1530.

PAULINO, G. H., *et al.*, 1994, "Node and element resequencing using the laplacian of a finite-element graph.2. implementation and numerical results", International Journal for Numerical Methods in Engineering, vol. 37, pp. 1531-1555.

PREPARATA, F. P. and SHAMOS, M., 1988, "Computational Geometry na Introduction", Spring Verlag. New York.

RABELO, J. J. E., 1992, "Análise de placas laminadas espessas por modelos de elementos finitos sólidos ortotrópicos", Dissertação de Mestrado, Universidade Federal de Santa Catarina, Florianópolis.

RABUSKE, M. A., 1992, "Introdução à teoria dos grafos", editora da UFSC, Florianópolis.

RAIZER, Adroaldo, 1987, "Contribuição a elaboração de um sistema tridimensional de cálculo de campos elétricos e magnéticos, utilizando a técnica de elementos finitos", Dissertação de Mestrado, Universidade Federal de Santa Catarina, Florianópolis.

RESENDE, Pedro J. de e STOLFI, Jorge, 1994, "Fundamentos da Geometria Computacional", IX Escola de Computação, Recife-PE.

SERAFIM, E. S., 1998, "Implementação de uma biblioteca informática para diversos tipos de elementos finitos em 2D e 3D", Dissertação de Mestrado, Universidade Federal de Santa Catarina, Florianópolis.

SILVA, Andrea V., 1999, "Geração de Malhas Computacionais por Triangulação de Delaunay em domínios não-convexos" , Projeto em Ciência da Computação II, Universidade Federal de Santa Catarina, Florianópolis.

SILVA, Luiz F., 1999, "Geração de diagramas de Voronoi em contornos arbitrários convexos a partir da Triangulação de Delaunay", Projeto em Ciência da Computação II, Universidade Federal de Santa Catarina, Florianópolis.

SMOLINSKI, P., 1991, "Stability of variable explicit time integration for unsteady diffusion problems", Computer Methods in Applied Mechanics and Engineering, vol. 93, nº. 2, pp. 247-252.

Source program of Voronoi and Delaunay [on line]. Disponível na internet. http://www.info.waseda.ac.jp/muraoka/members/kono/voronoi-e.html. 20 fev 1999.

STRANG, G., 1973, "Na analysis of the finite element method", Prentice-Hall, London.

SUGIHARA, K. and IRI M., 1992, "Construction of the Diagram for "One Million" Generators in Single-Precision Arithmetic", Proceedings of the IEEE, vol. 80, nº. 9

TELLÓ, M., 1991, "Dimensionamento de sistemas de aterramento em baixas freqüências usando o método de elementos finitos em três dimensões", Dissertação de Mestrado, Universidade Federal de Santa Catarina, Florianópolis.

THOMPSON, J. F. *et al.*, 1999, "Handbook of grid generation", CRC Press, New York.

VOZOVOI, L., *et al.*, 1998, "A fast Poisson solver of arbitrary order accuracy in rectangular regions", SIAM Journal on Scientific Computing, vol. 19, pp. 933-952.

ZIENKIEWICZ, O. C., 1992, "Computational Mechanics Today", International Journal for Numerical Methods in Engineering, vol. 34, pp. 9-33.

ZIENKIEWICZ, O. C., 1994, "The Finite Element Method", 4ª. ed. McGraw-Hill, New York, vol. 1, 4ª. ed.

ZLÁMAL, M., 1969, "On some finite element procedures for solving second order boundary value problems", Numer. Mathematics, vol. 14, pp. 42-48.

APÊNDICE A

Algumas definições, teoremas e lemas necessários ao entendimento das análises dos erros apresentadas no capítulo 4 são dadas neste apêndice. Uma pesquisa mais completa pode ser realizada em (Carey and Oden, 1983; Zienkiewicz, 1994; Strang, 1973)

Definição 1 – Elemento Finito no R^n

Um elemento finito no R^n é definido por (G, D, P), onde:

i) G é um conjunto fechado e não-vazio do R^n com um contorno $\partial\Omega$ Lipschitziano;

ii) D é um conjunto finito de funcionais lineares l_i, $1 \leqslant i \leqslant N_G$, definidos sobre $C^\infty(G)$, o qual denomina os graus de liberdade do elemento;

iii) P é um espaço de funções definidas sobre G, $P \subset C^\infty(G)$, tal que para quaisquer números reais α_i, $1 \leqslant i \leqslant N_G$, existe um único $\Psi \in P$ tal que $l_i(\Psi) = \alpha_i$, $1 \leqslant i \leqslant N_G$.

A condição (iii) implica na existência de N_G funções $\Psi_j \in P$ tais que $l_i(\Psi_j) = \delta_{ij}$, $1 \leqslant i, j \leqslant N_G$. Ou seja, a restrição de D a P forma uma base no espaço dual P' de P (Oden, 1972). Quando (iii) é válida, diz-se que D é P-unisolúvel. Quando (G, D, P) é um membro da partição Q_h de um dado domínio, tem-se que $G = \overline{\Omega}_e$ e então é adicionado o rótulo e, escrevendo $(\overline{\Omega}_e, D_e, P_e)$. Freqüentemente $\overline{\Omega}_e$ é denominado de um elemento finito ao invés de $(\overline{\Omega}_e, D_e, P_e)$.

Uma conseqüência de que $l_i(\Psi_j) = \delta_{ij}$, $1 \leqslant i, j \leqslant N_G$, é o fato de que:

$$v(x) = \sum_{i=1}^{N_G} l_i(v)\Psi_i(x) \ , \ \forall\, v \in P \tag{A.1}$$

Logo, pode-se empregar as funções de base $\{\Psi_i\}_{i=1}^{N_G}$ de P e os funcionais lineares

$\{l_i\}_{i=1}^{N_G}$ de D para construir P-interpolantes das funções v suficientemente suaves. Portanto, se $\prod: C^{\infty}(\Omega) \rightarrow P$ é um operador de projeção com a propriedade da seguinte equação:

$$l_i(\prod v) = l_i(v), 1 \leq i \leq N_G \tag{A.2}$$

Então, o P-interpolante de v é:

$$\prod v = \sum_{i=1}^{N_G} l_i(v)\Psi_i \tag{A.3}$$

Onde Ψ_i são as funções de base, $l_i(v)$ são funcionais lineares limitados e $v \in C^{\infty}(\Omega)$.

Definição 2 – Elemento Finito "P-unisolúvel"

Segundo Oden and Carey (1983), uma discussão completa sobre este assunto pode ser encontrada em Ciarlet (1978). A idéia geral é a seguinte: Os termos de D devem ser construídos de forma que quando todos os valores são especificados, um único polinômio P seja determinado. Em particular, se P contém polinômios de grau $\leq$ k, então os graus de liberdade dos elementos devem ser escolhidos de forma que especificando todos os valores dos funcionais l_i de D em $\chi \in$ P, seja determinada uma única função contendo um polinômio de grau k.

Por exemplo, seja $\Omega \subset R^2$ e considerando o grau de liberdade:

$$D = \{l_i / l_i(\chi) = \chi(b_i),\ i = 1, 2, 3,\ \chi \in P\} \tag{A.4}$$

Onde $\{b_i\}_{i=1}^{3}$ são os pontos nodais conforme Figura A.1 com coordenadas $\{b_{ij}\}_{j=1}^{2}$. Se os pontos não estão sobre a mesma linha, como no caso da Figura A.1a, então, D determina um único polinômio de grau 1. Se os três pontos são colineares, conforme Figura A.1b, então, não é determinado um único polinômio de grau 1. Assim, se $P = P_1(\Omega)$ é o espaço dos polinômios de grau $\leq$ 1 em $\overline{\Omega}$, D é P-unisolúvel quando os pontos b_i não são colineares.

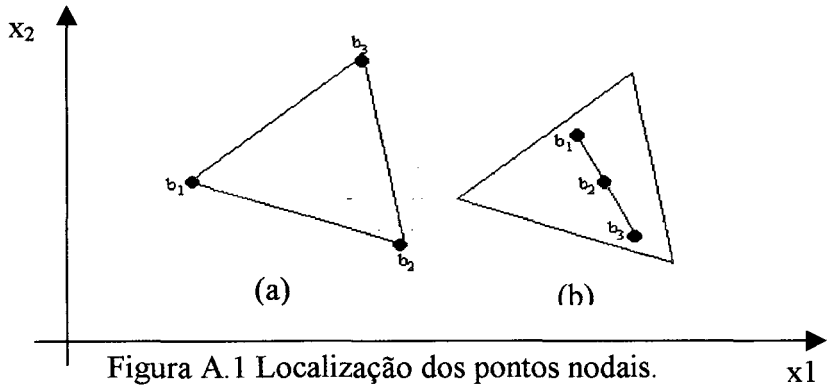


Figura A.1 Localização dos pontos nodais.

Fonte: Carey and Becker (1983)

Para a funções de interpolação quadráticas, tem-se que:

$$D_2 = \{l_i / l_i(\chi) = \chi(b_i);\ i = 1 \le i \le 6\ \ \chi \in P\} \tag{A.5}$$

Então D_2 é P-unisolúvel sempre que $P = P_2(\Omega)$.

Teoria de Interpolação em Espaços de Sobolev

A teoria de interpolação tem papel central na obtenção de estimativas de erros dos elementos finitos. Em seguida, é apresentado os fundamentos da teoria de interpolação de funções em espaços de Sobolev $W^{m,p}(\Omega), m \ge 0, 1 \le p \le \infty$, via elementos finitos.

Um modo direto de realizar tal interpolação é o seguinte: dado $u \in W^{m,p}(\Omega)$ e um espaço de elementos finitos $S^h(\Omega) \subset W^{m,p}(\Omega)$, pode-se definir o seguinte operador de interpolação:

$$\prod_h : W^{m,p}(\Omega) \to S^h(\Omega) \text{ tal que } \prod_h u := \sum_{i=1}^{M} l_i(u)\varphi_i \tag{A.6}$$

Onde $\{\varphi_i\}_{i=1}^{M}$ são as funções de base globais para $S^h(\Omega)$, e $\{l_i\}_{i=1}^{M}$ são os graus de liberdade na representação global. O objetivo é estudar a qualidade da aproximação $\prod_h u$ quando comparada a u para malhas cada vez mais refinada. Para isso são necessários alguns resultados, os quais são apresentados a seguir.

Teorema 1 – Mapeamentos Afins

Este teorema identifica o fato de que os mapeamentos dependem da geometria da malha, o teorema tem o seguinte enunciado: "Sejam Ω e $\hat{\Omega}$ dois domínios tais que $\forall x \in \overline{\Omega}, \mathrm{x} = \mathrm{T}\,\hat{\mathrm{x}} + \mathrm{c},\ \hat{\mathrm{x}} \in \overline{\hat{\Omega}}$, onde T é uma matriz não singular e c é um vetor de translação, então é definido":

$$\mathrm{h} = \mathrm{dia}(\Omega) \qquad \text{e} \qquad \hat{h} = \mathrm{dia}(\hat{\Omega}) \tag{A.7}$$

$$\rho := \sup\{dia\,(\mathrm{S}) \text{ tal que S é uma esfera contida em } \Omega\} \tag{A.8}$$

$$\hat{\rho} := \sup\{dia\,(\hat{\mathrm{S}}) \text{ tal que } \hat{\mathrm{S}} \text{ é uma esfera contida em } \hat{\Omega}\} \tag{A.9}$$

Então:

$$\|T\| \le \frac{\mathrm{h}}{\hat{\rho}} \qquad \text{e} \qquad \|T^{-1}\| \le \frac{\hat{\mathrm{h}}}{\hat{\rho}} \tag{A.10}$$

Ou seja, dadas duas malhas quaisquer, tem-se que ρ e $\hat{\rho}$ são respectivamente os maiores raios das circunferências circunscritas nos triângulos de cada malha.

Lema 1

Seja $\mathrm{W}^{k+1,p}(\Omega)$ um espaço de Sobolev incluso continuamente em um outro espaço de Sobolev $\mathrm{W}^{m,q}(\Omega), W^{\kappa+1,p}(\Omega) \to W^{m,q}(\Omega)$. Seja $\prod \in \alpha(W^{k+1,p}(\Omega), W^{m,q}(\Omega))$ um operador linear contínuo mapeando $W^{k+1,p}(\Omega)$ em $W^{m,q}(\Omega)$ tal que $\prod w = w$, $\forall w \in P_k(\Omega)$. Então, existe uma constante C(Ω) tal que $\forall v \in W^{k+1,p}(\Omega)$:

$$\left|v - \prod v\right|_{m,q,\Omega} \le C(\Omega)\left\|I - \prod\right\| \alpha(W^{k+1,p}(\Omega), W^{m,q}(\Omega))|v|_{k+1,p,\Omega} \tag{4.19}$$

Teorema 2 – Propriedades Globais de Interpolação

Seja $\{(\overline{\Omega}_e, D_e, P_e) : \overline{\Omega}_e \in Q_h, 1 \le e \le E\}$ uma família regular de elementos finitos afim a um elemento mestre $(\hat{\Omega}, \hat{D}, \hat{P})$; se p = q e as condições estabelecidas para a interpolação local (Teorema 1 – capítulo 4) são válidas com $\prod e = \prod h \mid e$, onde $\prod h$ é o operador de interpolação global $\prod h : W^{k+1,p}(\Omega) \to S^h(\Omega) \subset W^{m,p}(\Omega)$, então existe uma constante C > 0 tal que $\forall v \in W^{k+1,p}(\Omega)$:

$$\left| v - \prod {}_h v \right|_{m,p,\Omega} \le C h^{k+1-m} |v|_{k+1;p,\Omega} \tag{A.12}$$

Onde:

$$h := \max_{1 \le e \le E} h_e \tag{A.13}$$

É importante notar que para aplicar o Teorema 3, assumi-se que todas as suas hipóteses são satisfeitas e, portanto, é necessário contruir uma malha uniforme.

Teorema 3 – Teorema 2 particularizado

Assumindo as seguintes hipóteses:

(i) As hipóteses do teorema 3 são válidas;

(ii) $r \geqslant 0$;

(iii) $P_k(\hat{\Omega}) \subset \hat{P} \subset H^r(\hat{\Omega}) = W^{r,2}(\hat{\Omega}), S^h(\Omega) \subset C^0(\Omega)$;

(iv) $W^{k+1,2}(\hat{\Omega}) = H^{k+1}(\hat{\Omega}) \to \overline{C}^s(\hat{\Omega})$;

(v) p = q = 2;

(vi) A família $\{(\overline{\Omega}_e, D_e, P_e) : \overline{\Omega}_e \in Q_h, 1 \le e \le E\}$ é regular.

Então, existe uma constante C > 0 tal que $\forall$ v $\in H^{k+1}(\Omega)$, tem-se:

$$\left\| v - \prod {}_h v \right\|_{m,\Omega} \le C h^{k+1-m} |v|_{k+1,\Omega} \ , \ 0 \le m \le \min(1, r) \tag{A.14}$$

$$\left[\sum_{e=1}^{E} \left\| v - \prod {}_h v \right\|_{m,\Omega_e}^2 \right]^2 \le C h^{k+1-m} |v|_{k+1,\Omega} \ , \ 2 \le m \le \min(k+1, r) \tag{A.15}$$

Até aqui foi apresentado os resultados fundamentais da teoria de interpolação, os quais são utilizados nas estimativas *a priori* do erro. Em seguida são apresentados outros resultados relevantes para as estimativas.

Estimativas de Erro Padrão

Para a realização de estimativas de erros, alguns resultados da teoria de interpolação devem ser satisfeitos, destacando-se:

i) Seja $\Omega \subset R^n$ com contorno ($\partial\Omega$) Lipschitiziano e seja $\{Q_h\}_{0<h\leq 1}$ uma família de partições de Ω dependendo do parâmentro h da malha, $0 < h \leqslant 1$.

ii) Para todo h, sendo que $\{(\overline{\Omega}_e, D_e, P_e), \overline{\Omega}_e \in Q_h, 1 \leq e \leq E\}$ denota uma família de elementos finitos, os quais levam a um conjunto de funções de interpolação locais $\{\phi_i\}_{i=1}^{M}$, M= M(h) dos quais vem a base para um espaço linear de dimensão finita $H^h(\Omega)$.

iii) $\prod h$ denota o operador global de interpolação correspondendo para $\{\phi_i\}_{i=1}^{M}$ e seja $\prod h$ tal que para todo h, tem-se:

$$\begin{aligned} &\prod h : H^{k+1}(\Omega) \to H^h(\Omega) \subset H^m(\Omega); \quad k+1 \succ m \geq 0 \\ &\left(\prod{}_h v\right)\Big|_{\Omega_e} = \prod{}_e (v|_{\Omega_e}) \ , \ \forall \ v \in H^{k+1}(\Omega) \\ &\prod{}_h p = p \ , \ \forall \ p \in P_k(\Omega) \end{aligned} \tag{A.16}$$

Onde $H^{k+1}(\Omega) = W^{k+1,2}(\Omega)$, $H^m(\Omega) = W^{m,2}(\Omega)$, e usualmente, $P_k(\Omega)$ é o espaço de polinômios de grau $\leqslant$ h em Ω.

iv) Por (ii), há correspondência de família para família $\{Q_h\}_{0<h\leq 1}$ do subespaço $\{H^h(\Omega)\}_{0<h\leq 1}$. Esta família tem a seguinte propriedade: Para todo $v \in H^r(\Omega)$ e todo h, existe uma constante $C > 0$ e um elemento $\tilde{v}_h \in H^h(\Omega)$ tal que:

$$\|v - \tilde{v}_h\|_{m,\Omega} \leq Ch^{\mu}|v|_{r,\Omega} \tag{A.17}$$

Onde

$$\mu = \min(k+1-m, r-m) \tag{A.18}$$

Teorema 4 – Estimando Erros

Se as condições seguintes são válidas:

i) A forma bilinear b: $H_0^m(\Omega) x H_0^m(\Omega) \rightarrow R$ definida via equação (4.21), satisfazendo as equações (4.22) a (4.24) e a forma bilinear $F : H_0^m(\Omega) \rightarrow R$ é definida via equação (4.26).

ii) A família de subespaços $\{H_0^m(\Omega)\} 0 < h \leq 1$ satisfaz as propriedades de interpolação citadas anteriormente

iii) As condições estabelecidas nas equações (4.22) a (4.24) são também satisfeitas em $H_0^m(\Omega)$, com $\alpha=\alpha_h>0$ para todo h. Então, existem soluções únicas $u^* \in H_0^m(\Omega)$ e $u_h^* \in H_0^m(\Omega)$, correspondentes respectivamente aos problemas das equações (4.26) e (4.28). Além disso, se $u^* \in H^r(\Omega) \cap H_0^m(\Omega)$, $r \geq m$, então, quando $h \rightarrow 0$, o erro $e := u^* - u_h^*$, satisfaz a seguinte desigualdade.

$$\|e\|_{m,\Omega} \leq C\left(1 + \frac{M}{\alpha_h}\right) h^{\mu} |u^*|_{r,\Omega} \tag{A.19}$$

Onde μ:=min{k+1-m, r –m}, onde r-m determina a regularidade de u* e k+1-m o grau da função de interpolação (k+1 ≥ m).

Teorema 5 – Regularidade

Seja $u^* \in H^r(\Omega)$, $r \geq 2$, a solução do seguinte PVC elíptico:

$$Au = \mathrm{f} \qquad \text{(A.20)}$$
$$\gamma_j \mathrm{u} = \mathrm{g}_j \text{ em } \partial\Omega \ , \ 0 \le \mathrm{j} \le \mathrm{m}-1$$

Onde $\partial\Omega$ é "suave" (por exemplo, $\partial\Omega \in C^m$) e A é um operador fortemente elíptico definido por:

$$Au := \sum_{|\alpha|,|\beta| \le m} (-1)^{|\beta|} D^{\beta}(a_{\alpha,\beta} D^{\alpha} v) \qquad \text{(A.21)}$$

Então, $\exists$ c > 0 tal que:

$$\|u^*\|_{r,\Omega} \le C\left\{ \|f\|_{r-2m,\Omega} + \sum_{j=0}^{m-1} \|g_j\|_{r-j-\frac{1}{2},\Omega} \right\} \qquad \text{(A.22)}$$

A equação (A.22) expressa o resultado da Teoria de EDP's elípticas.

ANEXOS

```
%#########################################################################
% Última atualização: 17/11/99.
% Programa Principal da Resolução da Equação de Poisson pelo MEF
%#########################################################################

global n_pts n_tri n_cvh n_viz x y

%#########################################################################
% O arquivo MENU.M, é onde são definidas as informações do problema
%#########################################################################

menu

%#########################################################################
% A partir das informações iniciais, tem-se os valores de x e y, neste
% momento pode-se gerar a TDEL - que é realizada pelo arquivo GERTRI.M.
% Neste arquivo, também são definidos os pontos que estao no contorno e
% a lista de pontos adjacentes a cada nó. Estas duas informações serão
% necessárias para o reordenamento. % Ainda neste arquivo é calculados
%o tamanho do elemento (parâmetro he)
%#########################################################################

gertri

%#########################################################################
% Se os elementos forem quadratico, é necessário definir os pontos dos
% elementos quadráticos, ou seja, é necessario executar os    arquivos
% ARESTAS.M  e QUADRATIC0.M.  Neste caso o contorno que tinha sido
% preliminarmente  calculado em GERTRI.M, mudou e os pontos  da malha
% também, então tem-se que recalcular "cvh" e plotar os novos números.
%#########################################################################
if tipo_elemento == 2
   arestas
   quadratico

   % Convhull, encontra os pontos do novo casco convexo.
   cvh=convhull(x,y);
   cvh=cvh';
   n_cvh=size(cvh,1);

   % Plotar os números
   n_pts=size(x,2);
   i=1:n_pts;
   it=num2str(i');
   text(x,y,it);
   hold on
end

%#########################################################################
% Neste momento, se desejar fazer o reordenamento do indices nodais,
% tem-se que  executar o arquivo CM.M, e depois tem-se que executar o
% GERTRI.M novamente para para ter uma nova triangulação. Nesta nova
% execução do GERTRI.M, também já é recalculado os pontos do contorno
% "cvh"
%#########################################################################
if reordenamento == 1
   cm
   gertri
end
```

```
%#########################################################################
% Após as estapas acima, é necessário calcular a distância entre os pontos
% do contorno, pois estas informações são necessária na imposição das CC.
% de Neumann. Não se está mais utilizando esta maneira, está sendo
% utilizada a integração numérica em todos os
% casos. Esta informação é calculada, no arquivo DISTANCIA.M
%#########################################################################

distancia

%#########################################################################
% Neste momento, estah faltando ainda, informar as Condições de
% Contorno, cargas concentradas, etc... Neste momento tem-se os nós
% definitivos, onde é possível identificar a numeração final dos mesmos
% e informar as CC. etc....
% Então neste momento é dado uma "pausa" no programa para informar as
% condições de contorno, cargas concentradas e propriedades físicas
% diferentes em um arquivo separado.
%#########################################################################

'Neste momento é necessário verificar no arquivo termino_dados.m, se as
condições de contorno, 'cargas concentradas, estao corretas'
'Pressione Qualquer tecla para continuar'
pause

if problema ==1
   termino_dados
end

if problema == 2
   termino_dados2
end

KG=zeros(n_pts,n_pts);
FG=zeros(n_pts,1);

for i=1:n_tri
   tri_elem=tri(i,:);

   % A Função que define os K(elementar) e F(elementar)
   %#######################################################################
   % Se os elementos forem lineares, teremos 3 nós
   % Se os elementos forem quadráticos, serão 6 nós
   % No arquivo KELEM.M é realizada a integração numérica, neste
   % arquivo também deve-se informar as funções kx(x,y), ky(x,y) e
   % f(x,y), as quais são avaliadas nos pontos de Gauss.
   %#######################################################################

   kelemquad2

%#########################################################################
% O procedimento abaixo, é realizado para Montar a Matriz K(global),
% colocando todas as contribuicoes de cada triangulo, que são
% calculadas em (KELEM).
%#########################################################################

   for it=1:n_nos  %(varrendo as n_nos colunas de cada triangulo it)
      for ic=1:n_nos
         KG(tri_elem(it),tri_elem(ic))=KG(tri_elem(it),tri_elem(ic))+
         KE(it,ic);
```

```
      end
      FG(tri_elem(it))=FG(tri_elem(it))+FE(it);
   end
end


%#########################################################################
% Cargas concentradas, caso tenha cargas concentradas, o valor deve ser
% informado no arquivo termino_dados.M
%#########################################################################

for i=1:n_concent
   FG(c_concent(i))=FG(c_concent(i))+valor_concent(i);
end


%#########################################################################%
Impondo as condicoes de contorno - Essenciais
%#########################################################################

SG=zeros(n_pts,1);

%Tratando o caso especial do 1. nó

for i=1
   for j=1:n_c_essencial
      if c_essencial(j)==cvh(i)
         SG=SG+valor_essencial(j)*KG(:,cvh(1));
      end
   end
end

% Tratando os demais nós
for i=2:n_cvh-1
   for j=1:n_c_essencial
      if c_essencial(j)==cvh(i)
         SG=SG+valor_essencial(j)*KG(:,cvh(i));
      end
   end
end

% Subtraindo os valores de K que passam para F....
FG=FG-SG;



%#########################################################################
% Impondo as condicoes de contorno - Naturais
%#########################################################################
% O arquivo neumann soh deve ser executado nos casos onde se tem
% condicoes de contorno naturais e diferentes de zero.... Pois nos
% casos em que isso não acontece não é necessário.

if condicoes_neumann==1
   neumann
   % Se o fluxo está entrando então SG1 deve ser adicionado, no
   % entanto, se fosse ao contrário bastaria subtrair.
   FG=FG+SG1;
end
```

```
%#########################################################################
% Eliminando as colunas onde se tem condicoes de contorno essenciais
% O procedimento de eliminacao das linhas é importante para diminuir
% o tamanho do sistema a ser resolvido.
% No entanto, uma outra forma de impor as condições de contorno, seria
% zerar todos os elemento da linha de K(i,j) que se conhece o valor e
% colocar 1 no elemento conhecido - diagonal 1
%#########################################################################


[KGNN, FGN]=eliminacc(KG,FG, c_essencial, n_pts, n_c_essencial);


%#########################################################################
% A ultima etapa do Programa é resolver o sistema de equações lineares
%#########################################################################

gauss_banda

%u=KGNN\FGN;

% Em "un" - ficará armazenados os valores conhecidos (Condições de
% Contorno)
un=zeros(n_pts,1);
for i=1:n_c_essencial
   un(c_essencial(i))=un(c_essencial(i))+valor_essencial(i);
end


%#########################################################################
% Nesta última etapa, é necessário juntar os valores conhecidos das
% condições de contorno, com os valores calculados na resolução do
% sistema de equações algébricas.
%#########################################################################

teste_essencial=c_essencial;
n_teste_essencial=n_c_essencial;

for i=1:n_pts
   uuuuu=1;
   for j=1:n_teste_essencial
      teste=teste_essencial(j);
      if teste==i
         uuuuu=0;
      end
   end
   if uuuuu==1
      n_u=size(u,1);
      un(i)=u(1);
      u=u(2:n_u);
   end
end

% Estimando o erro

if problema == 1
   % Calculando a Solução exata
   for i=1:n_pts
      uexato(i)=x(i)*y(i)*log(x(i)*y(i));
   end
```

```
    uexato1=uexato';
    erro=abs(un-uexato1);
    erro=max(erro)
end


if problema == 2
    % Calculando a Solução exata
    for i=1:n_pts
        uexato(i)=25*x(i)+50;
    end

    uexato1=uexato';
    erro=abs(un-uexato1);

    erro=max(erro)
end
```

```
%#########################################################################
%                          MENU DE ENTRADA DE DADOS
% Neste arquivo/menu é onde são definidas as informações inicias do
% problema
%#########################################################################

% Problema a ser resolvido

disp('Resolver qual problema?')
problema = input('Qual o problema a ser resolvido =');

% Definir a forma de entrada dos pontos do contorno e dos pontos internos
disp('Qual a forma de entrada dos dados?')
disp('      1) Via arquivo de dados')
disp('      2) Via interface gráfica ')
disp('      3) Geração automática dos Pontos ')
entrada_dados = input('Escolha uma das Formas =');

if entrada_dados ~=1 & entrada_dados~=2 & entrada_dados ~=3
   'Entrada de Dados Inválida - Pressione Ctrl + C e entre novamente com os
dados'
   pause
end

if entrada_dados == 1 & problema ==1
   dados
end

if entrada_dados == 1 & problema ==2
   dados1
end

if entrada_dados == 2
   xc = input('Entre com os pontos do Contorno em "x" =')
   yc = input('Entre com os pontos do Contorno em "y" =')

   total_xc=size(xc,2);
   total_yc=size(yc,2);

   if total_xc ~= total_yc
      'ERRO - O número de pontos nos vetores "xc" e "yc" devem ser iguais -
      Pressione Ctrl+C e execute o programa novamente'
      pause
   end

   pontos_internos = input('Qual o número de pontos internos do contorno
   =');
   'Pressione <ENTER> para plotar o contorno, em seguida defina os pontos
   internos com o clicar do Mouse'
   pause
   plot(xc, yc, '-')
   [x, y] = ginput(pontos_internos) ;

   x=[xc x];
   y=[yc y];
end
if entrada_dados == 3
   xc = input('Entre com os pontos das Extremidades do Contorno em "x" =')
   yc = input('Entre com os pontos das Extremidades do Contorno em "y" =')
```

```
   total_xc=size(xc,2);
   total_yc=size(yc,2);

   xc(total_xc+1)=xc(1);
   yc(total_yc+1)=yc(1);


   if total_xc ~= total_yc
      'ERRO - O número de pontos nos vetores "xc" e "yc" devem ser iguais -
      Pressione Ctrl+C e execute o programa novamente'
      pause
   end

   x0 = input('Entre com o menor valor em "x" =')
   x1 = input('Entre com o menor valor em "x" =')
   y0 = input('Entre com o menor valor em "y" =')
   y1 = input('Entre com o maoir valor em "y" =')
   hx = input('Entre com o "passo" em "x" =')
   hy = input('Entre com o "passo" em "y" =')

   gerpto;
end


% Escolha do Tipo de Elemento
disp('Qual tipo de elemento voce deseja utilizar?')
disp('      1) Triângulos Lineares')
disp('      2) Triângulos Quadrático')
tipo_elemento = input('Escolha uma das Formas =');

if tipo_elemento ~=1 & tipo_elemento ~=2
   'Entrada de Dados Inválida - Pressione Ctrl + C e entre novamente com os
   dados'
   pause
end

% fazer reordenamento?

disp('Você deseja reordenar os indices nodais?')
disp('      1) Sim')
disp('      2) Não')
reordenamento = input('Escolha uma das Formas =');


if reordenamento ~=1 & reordenamento ~=2
   'Entrada de Dados Inválida - Pressione Ctrl + C e entre novamente com os
   dados'
   pause
end


% A ordem da Integração numérica

disp('Você deseja qual ordem de integração?')
disp('      1) Quadrática')
disp('      2) Cúbica')
disp('      3) Quíntica')

integracao = input('Escolha uma das Formas =');
```

```
if integracao ~=1 & reordenamento ~=2 & reordenamento ~=2
   'Entrada de Dados Inválida - Pressione Ctrl + C e entre novamente com os
   dados'
   pause
end

% Se há condicoes de Neumann diferentes de zero no problema

disp('No problema há condições de Neumann diferentes de zero?')
disp('      1) Sim')
disp('      2) Não')

condicoes_neumann=input('Escolha uma das Formas =');
```

```
%#########################################################################
% Neste arquivo é onde são informados os pontos do contorno e pontos
% internos, se a opção de entrada de dados for via arquivo... No
% arquivo menu.m, é citado dados.m e também dados1.m. Os dois arquivos
% possuem a mesma estrutura, o que varia é o contorno.
%#########################################################################

%*************************************************************************
% Problema 1
%
%*************************************************************************

%#########################################################
%Malha 1 - 32 elementos
%#########################################################

if malha==11
   x=[1.0000 1.0000 1.0000 1.2500 1.5000 1.7500 2.0000 2.0000 2.0000 2.0000
   2.0000 1.7500 1.5000 1.2500 1.0000 1.0000 1.2500 1.5000 1.7500 1.2500
   1.5000 1.7500 1.2500 1.5000 1.7500];
   y=[1.5000 1.2500 1.0000 1.0000 1.0000 1.0000 1.0000 1.2500 1.5000 1.7500
   2.0000 2.0000 2.0000 2.0000 2.0000 1.7500 1.2500 1.2500 1.2500 1.5000
   1.5000 1.5000 1.7500 1.7500 1.7500];
end

%#########################################################
% Malha 2 - 64 elementos
%#########################################################

if malha==12
   x=[1.0000 1.0000 1.0000 1.0000 1.0000 1.1250 1.2500 1.3750 1.5000 1.6250
   1.7500 1.8750 2.0000 2.0000 2.0000 2.0000 2.0000 2.0000 2.0000 2.0000
   2.0000 1.8750 1.7500 1.6250 1.5000 1.3750 1.2500 1.1250 1.0000 1.0000
   1.0000 1.0000 1.1250 1.2500 1.3750 1.5000 1.6250 1.7500 1.8750 1.1250
   1.2500 1.3750 1.5000 1.6250 1.7500 1.8750 1.1250 1.2500 1.3750 1.5000
   1.6250 1.7500 1.8750 1.1250 1.2500 1.3750 1.5000 1.6250 1.7500 1.8750
   1.1250 1.2500 1.3750 1.5000 1.6250 1.7500 1.8750 1.1250 1.2500 1.3750
   1.5000 1.6250 1.7500 1.8750 1.1250 1.2500 1.3750 1.5000 1.6250 1.7500
   1.8750];
   y=[1.5000 1.3750 1.2500 1.1250 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
   1.0000 1.0000 1.0000 1.1250 1.2500 1.3750 1.5000 1.6250 1.7500 1.8750
   2.0000 2.0000 2.0000 2.0000 2.0000 2.0000 2.0000 2.0000 2.0000 1.8750
   1.7500 1.6250 1.1250 1.1250 1.1250 1.1250 1.1250 1.1250 1.1250 1.2500
   1.2500 1.2500 1.2500 1.2500 1.2500 1.2500 1.3750 1.3750 1.3750 1.3750
   1.3750 1.3750 1.3750 1.5000 1.5000 1.5000 1.5000 1.5000 1.5000 1.5000
   1.6250 1.6250 1.6250 1.6250 1.6250 1.6250 1.6250 1.7500 1.7500 1.7500
   1.7500 1.7500 1.7500 1.7500 1.8750 1.8750 1.8750 1.8750 1.8750 1.8750
   1.8750];
end
```

```
%#########################################################################
% Neste script, basta informar o contorno, o parâmetro hx e hy, para
% que automaticamente sejam gerados os pontos no interior do domínio,
% os quais serão triangulados pelo programa "gerpto"
%#########################################################################


% Fixa-se o y e faz-se a varredura em x
x(1)=x0;
y(1)=y0;
ny=floor((y1-y0)/hy);
hy=(y1-y0)/ny;
nx=floor((x1-x0)/hx);

if nx==0
   h=0
else
   h=(x1-x0)/nx;
end

if h==0
   i=1

else
   for i=2:nx+1
      x(i)=x(i-1)+h;
      y(i)=y0;
   end
end

it=i+1;
while (y(it-1)+hy)<y1
   y(it)=y(it-1)+hy-hy*0.000001;y0=y(it);
   [x0,x1]=corta_contorno(xc,yc,y0);
   x(it)=x0;
   nx=floor((x1-x0)/hx);
   h=(x1-x0)/nx;
   for i=it+1:it+nx
      x(i)=x(i-1)+h;
      y(i)=y(it);
   end
   it=i+1;
end
```

```
%#########################################################################
% Algoritmo Corta Contorno
%#########################################################################

function [xi,xf]=corta_contorno(xc,yc,y0)

n_con=size(xc,2);v=[];
m=1.0e-20;
x0=xc(1);
for ic=1:n_con-1
   teste1=m*xc(ic)-yc(ic)+(-m*x0+y0);
   teste2=m*xc(ic+1)-yc(ic+1)+(-m*x0+y0);
   prod=teste1*teste2;
% Prod<0 - significa que a reta com inclinação m ancorada em (x0,y0)
% corta o contorno (xc(ic+1),yc(ic+1))-(xc(ic),yc(ic)) em (xf,yf);
   if prod<0
      a= yc(ic+1)-yc(ic);
      b=-xc(ic+1)+xc(ic);
      c=-xc(ic)*(yc(ic+1)-yc(ic))+yc(ic)*(xc(ic+1)-xc(ic));
      xf=( b*m*x0-b*y0-c)  /(a+b*m);
      yf=(-m*c-m*a*x0+a*y0)/(a+b*m);
      v=[v xf];
   else
       if prod==0 & (teste1~=0 | teste2~=0)
          a= yc(ic+1)-yc(ic);
          b=-xc(ic+1)+xc(ic);
          c=-xc(ic)*(yc(ic+1)-yc(ic))+yc(ic)*(xc(ic+1)-xc(ic));
          xf=( b*m*x0-b*y0-c)  /(a+b*m);
          yf=(-m*c-m*a*x0+a*y0)/(a+b*m);
          v=[v xf];
       end
   end
end
xi=min(v);xf=max(v);
```

```
%#########################################################################
% Gerador da triangulação de Delaunay para domínios convexos. Neste
% caso é utilizado a geração do Matlab com pequenas alterações
% Este programa também está adaptado para calcular o tamanho do
% elemento, o qual está sendo armazenado na variável "he" .
%#########################################################################

n_pts=size(x,2);
colormap(gray);
% Gerar a triangulação e obter os ponto em sentido anti-horário
tri=delaunay(x,y);
aux=tri(:,1);
tri(:,1)=tri(:,3);
tri(:,3)=aux;
tri
n_tri=size(tri,1);
for i=1:n_tri
      k1=tri(i,1);
      k2=tri(i,2);
      k3=tri(i,3);
      xt=[x(k1),x(k2),x(k3),x(k1)]';
      yt=[y(k1),y(k2),y(k3),y(k1)]';
      plot(xt,yt,'-')
      hold on
      plot(x,y,'k.')
      axis([0.5, 2.5, 0.5, 2.5]);
end

% Plotar os números
if tipo_elemento==1
   n_pts=size(x,2);
   i=1:n_pts;
   it=num2str(i');
   text(x,y,it);
   hold on
end


% Convhull, encontra os pontos do contorno.
cvh=convhull(x,y);
cvh=cvh';
n_cvh=size(cvh,1);


% Determina uma lista de pontos adjacentes
vizo=vizinho(tri,cvh);

%Cálculo do tamanho de he
% Para esse cálculo foi utilizado o fato que: a distância de qualquer
% um dos três pontos do triângulo até o centro do círculo circunscrito
% são iguais, depois foi montado o sistema para determinar o xc (centro
% em x) e yc (centro em y) d1^2=d2^2;  d1^2=d3^2 - Assim consegue-se
% determinar "xc" e "yc".
% Depois basta calcular a distância  desse ponto (xc, yc) até qualquer
% ponto do triângulo, e então, obter o raio da circunferência
% circunscrita em cada triângulo, e tomando o maior valor como "he"
```

```
for i=1:n_tri
   xc=(x(tri(i,1))^2*(y(tri(i,2))-
   y(tri(i,3)))+x(tri(i,2))^2*(y(tri(i,3))-
   y(tri(i,1)))+(x(tri(i,3))^2+(y(tri(i,1))...
   -y(tri(i,3)))*(y(tri(i,2))-y(tri(i,3))))*(y(tri(i,1))-
   y(tri(i,2))))/(2*x(tri(i,1))*(y(tri(i,2))-y(tri(i,3)))...
   +2*x(tri(i,2))*(y(tri(i,3))-y(tri(i,1)))+2*x(tri(i,3))*(y(tri(i,1))-
   y(tri(i,2))));


   yc=-(x(tri(i,1))^2*(x(tri(i,2))-x(tri(i,3)))-x(tri(i,1))*(x(tri(i,2))^2-
   x(tri(i,3))^2+(y(tri(i,2))...
   +y(tri(i,3)))*(y(tri(i,2))-
   y(tri(i,3))))+x(tri(i,2))^2*x(tri(i,3))+x(tri(i,2))*((y(tri(i,1)))...
   +y(tri(i,3)))*(y(tri(i,1))-y(tri(i,3)))-
   x(tri(i,3))^2)+x(tri(i,3))*(y(tri(i,1))+...
   y(tri(i,2)))*(y(tri(i,2))-y(tri(i,1))))/(2*x(tri(i,1))*(y(tri(i,2))-
   y(tri(i,3)))+...
   2*x(tri(i,2))*(y(tri(i,3))-y(tri(i,1)))+2*x(tri(i,3))*(y(tri(i,1))-
   y(tri(i,2))));

   h(i)=sqrt((x(tri(i,1))-xc)^2+(y(tri(i,1))-yc)^2);
end

he=max(h);
```

```
%#####################################################################
% Neste arquivo determina-se os pontos adjacentes a cada da malha
%#####################################################################

function vizo=vizinho(tri,cvh)
global n_pts n_tri n_cvh n_viz n_con x y xc yc
viz=zeros(n_pts,10);

for itriang=1:n_pts
   [itr jtr]=find(tri==itriang);
   [nitr mitr]=size(itr);
   vizaux=[];
   for i=1:nitr
      vizaux=[vizaux tri(itr(i),:)];
   end
   vizaux=sort(vizaux);

% =====>  mata repetidos de vizaux
   iaux=vizaux(1);
   k=0;
   if iaux~=itriang
      k=k+1;
      viz(itriang,k)=iaux;
   end
   for i=2:3*nitr
      if vizaux(i)~=iaux & vizaux(i)~=itriang
         k=k+1;
         viz(itriang,k)=vizaux(i);
         iaux=vizaux(i);
      end
   end
end
%


vizo=zeros(n_pts,10);
% Ordenacao dos vizinhos no sentido de caminhamento anti-horario
% Cada ponto nodal terá seus vizinhos ordenados em relacao a
% (xref,yref)

for i=1:n_pts
    [m n]=find(viz(i,:)~=0);
    [m n_viz(i)]=size(n);
    ang=[];
      for j=1:n_viz(i)
         jviz=viz(i,j);
         dx=x(jviz)-x(i);
         dy=y(jviz)-y(i);
         ang(j)=atan2(dy,dx);
      end
       [lixo,I]=sort(ang);
      for j=1:n_viz(i)
          vizo(i,j)=viz(i,I(j));
      end
      % Caso o ponto i pertença ao k-ésimo elemento do casco convexo
      % (cvhl(k)), então deve-se verificar se o seu primeiro vizinho
      % (vizo(i,1)) corresponde ao K-ésimo+1 elemento do casco convexo
```

```
        vizaux=[];
        for k=1:n_cvh-1
            if i==cvh(k)
               if vizo(i,1)~=cvh(k+1)
                  for ia=2:n_viz(i)
                     if vizo(i,ia)==cvh(k+1)
                        for iaux=ia:n_viz(i)
                              vizaux=[vizaux vizo(i,iaux)];
                        end
                        for iaux=1:ia-1
                            vizaux=[vizaux vizo(i,iaux)];
                        end
                     end
                  end
                  for ia=1:n_viz(i)
                     vizo(i,ia)=vizaux(ia);
                  end
               end
            end
         end
end
```

```
%#########################################################################
% Esta rotina percorre cada linha de vizo e verifica os elementos
% da linha que sao menores que i.... e seta esse elementos menores
% que i para zeros, assim elimina o problema de criacao de arestas
% repetidas.
%#########################################################################

jjj=size(vizo,2);

vizofinal=zeros(n_pts,jjj);
for i=1:n_pts
   vizo1=vizo(i,:);
   for j=1:jjj
      teste=vizo(i,j);
      if i > teste & teste ~= 0
         vizofinal(i,j)=0;
      else
         vizofinal(i,j)=teste;
      end
   end
end


% Achar as arestas dos triangulos

Aresta=[];
for i=1:n_pts
   vizo_aux=vizofinal(i,:);
for j=1:jjj
      if vizo_aux(j) ~= 0
         Aresta=[Aresta; i vizo_aux(j)];
      end
   end
end

n_arestas=size(Aresta,1);
```

```
% ######################################################################
% Neste arquivo, são determinadas as arestas dos elementos quadráticos,
% a partir das informações dos triângulos lineares.
% ######################################################################

tri1=[tri zeros(n_tri, 3)];
ponto=n_pts;

% Pelas arestas que não estão repetidas, pode-se calcular os valores
% de x e y, que são armazenados na sequência....
for i=1:n_arestas;
   aresta_aux=Aresta(i,:);
   x=[x (x(aresta_aux(1))+x(aresta_aux(2)))/2];
   y=[y (y(aresta_aux(1))+y(aresta_aux(2)))/2];
end


% Este procedimento eh para numerar os nos, para isso tem-se
% que encontrar a linha que está a aresta que está sendo procurada,
% pois os valores de x, y destes pontos das arestas, estão armazenados
% na posiçao "n_pts" + n. linha que está a aresta
for i=1:n_tri
   aux=tri(i,:);

   for j=1:3
      j1=aux(j);
      if j==3
         j2=aux(1);
      else
         j2=aux(j+1);
      end
      [ii,jj]=find(Aresta==j1);
      [iiii,jjjj]=find(Aresta(ii,:)==j2);
      tri1(i,j+3)=ii(iiii)+ponto;
   end
end
tri=tri1;
```

```
%#########################################################################
% Reordenamento dos índices nodais
%#########################################################################


n=size(vizo,1);
% Graus dos vertices: Esta função nnz - retorna o numero de elementos
% nao-nulos
grau=zeros(1,n);
for i=1:n
   grau(i)=nnz(vizo(i,:));
end
maxcol=[];

% Ordenando as adjacencias segundo o seu grau:

for i=1:n
   adjai=suprimzeros(vizo(i,:));
   ncol=size(adjai,2);
   maxcol=[maxcol; ncol];
   gad=grau(adjai);
   [gord,iord]=sort(gad);
   adjai=adjai(iord);
   vizo(i,1:ncol)=adjai;
end
I=1:n;

% Executando o reordenamento, segundo o algoritmo CM

k=0;  i=1;  N=n;
v=zeros(1,n);
while k < N
   if i<=k
      vi=v(i);
      for j=1:maxcol(vi)
         w=vizo(vi,j);
         if ~isempty(find(I==w))
            k=k+1;
            v(k)=w;
            [ii,jj]=find(I==w);
            I=suprimel(I,jj);
            grau=suprimel(grau,jj);
            n=n-1;
         end
      end
      i=i+1;
   else
      k=k+1;
      mg=min(grau);
      [ii,jj]=find(grau==mg);
      v(k)=I(1,jj(1));
      w=v(k);
      I=suprimel(I,jj(1));
      grau=suprimel(grau,jj(1));
      n=n-1;
   end
end
```

```
%Colocando os dados x e y nas novas posicoes, ou seja, reordenados de
% acordo com o algoritmo CM.

for i=1:n_pts
   vaux=v(i);
   xaux(i)=x(vaux);
   yaux(i)=y(vaux);
end

x=xaux;
y=yaux;
```

```
%#########################################################################
% Esta funcao suprime os zeros do vetor dado v e
%    coloca o resultado no vetor vsz.
%#########################################################################


function vsz=suprimzeros(v)

n=length(v);

vsz=[];
for j=1:n
   if v(j)~=0
      vsz=[vsz v(j)];
   end
end
```

```
%#########################################################################
% Esta funcao suprime o elemento vj do vetor v e
%    cria um vetor reduzido vr = v - {vj}
%#########################################################################


function vr=suprimel(v,j)


n=max(size(v));
ve=v(1,1:j-1);
vd=v(1,j+1:n);
vr=[ve vd];
```

```
%#########################################################################
% Neste arquivo, é onde devem ser informados os valores das condicoes
% de contorno e cargas concentradas.
%#########################################################################


%#########################################################
%Dominio 1
%#########################################################

c_essencial=[1; 2; 6; 7; 8; 9; 10; 14; 15; 16];
valor_essencial=[50; 50; 100; 100; 100; 100; 100; 50; 50; 50];
n_c_essencial=size(c_essencial,1);

c_natural=[2 3; 3 4; 4 5; 5 6; 10 11; 11 12; 12 13; 13 14];
valor_natural=[0 0; 0 0; 0 0; 0 0; 0 0; 0 0; 0 0; 0 0];
n_c_natural=size(c_natural,1);

c_concent=[];
valor_concent=[];
n_concent=size(c_concent,1);
```

```
%#########################################################################
% Cálculo do KE e FE....
% Utilizando a integracao numerica (Gauss) Quadrática, Cúbica e Quintica.
%#########################################################################

% Para integração quadrática
if integracao == 1
   qsi_gauss=[0.5 0.5 0];
   eta_gauss=[0 0.5 0.5];
   peso=[1/3 1/3 1/3];
   n_pontos_gauss=3;
end

%Para integração Cúbica
if integracao == 2
   qsi_gauss=[1/3 0.2 0.6 0.2];
   eta_gauss=[1/3 0.2 0.2 0.6];
   peso=[-27/48 25/48 25/48 25/48];
   n_pontos_gauss=4;
end

%Para integração Quintica
if integracao == 3
   qsi_gauss=[1/3 0.4701420641 0.0597158717 0.4701420641 0.1012865073
   0.7974269853 0.1012865073];
   eta_gauss=[1/3 0.4701420641 0.4701420641 0.0597158717 0.1012865073
   0.1012865073 0.7974269853];
   peso=[0.225 0.1323941527 0.1323941527 0.1323941527 0.1259391805
   0.1259391805 0.1259391805];
   n_pontos_gauss=7;
end

if tipo_elemento==1
   KE=zeros(3,3);  FE=zeros(3,1);
   n_nos=3;
end

if tipo_elemento==2
   KE=zeros(6,6);  FE=zeros(6,1);
   n_nos=6;
end

for ig=1:n_pontos_gauss
   qsi=qsi_gauss(ig);
   eta=eta_gauss(ig);

   if tipo_elemento==1
      [psi,dpsdqsi,dpsdeta]=fformalin(qsi,eta);
   end

   if tipo_elemento==2
      [psi,dpsdqsi,dpsdeta]=fforma(qsi,eta);
   end

   xx=x(tri_elem);
   yy=y(tri_elem);
   aux1=yy*dpsdeta';
   aux2=yy*dpsdqsi';
   aux3=xx*dpsdeta';
   aux4=xx*dpsdqsi';
```

```
    Jacobi=(aux4*aux1)-(aux3*aux2);

    dpsidx=(dpsdqsi*aux1-dpsdeta*aux2)/Jacobi;
    dpsidy=(-dpsdqsi*aux3+dpsdeta*aux4)/Jacobi;

    %############################################################
    %Definindo Kx(x,y),  Ky(x,y) e f(x,y)
    %############################################################

    if problema==1
       kx=1;
       ky=1;
       xg=(xx*psi');
       yg=(yy*psi');
       f=-(xg/yg)-(yg/xg);
    end

    if problema==2
       kx=1;
       ky=1;

       f=0;
    end

    % Integração
    for i=1:n_nos
       for j=i:n_nos
       KE(i,j)=KE(i,j) + peso(ig)*(kx*dpsidx(i)*dpsidx(j)+
       ky*dpsidy(i)*dpsidy(j))*Jacobi;
       KE(j,i)=KE(i,j);
       end
    end

    for i=1:n_nos
       FE(i)=FE(i)+peso(ig)*psi(i)*Jacobi*f;
    end

end % for ig
```

```
% Funções de forma para elementos lineares

function [psi,dpsdq,dpsdet]=fformalin(qsi,eta)

psi(1)=(1-qsi-eta);
psi(2)=qsi;
psi(3)=eta;

dpsdq(1)=-1;
dpsdq(2)=1;
dpsdq(3)=0;

dpsdet(1)=-1;
dpsdet(2)=0;
dpsdet(3)=1;
```

```
% Funções de forma para elementos quadráticos

function [psi,dpsdq,dpsdet]=fforma(qsi,eta)

psi(1)=2*(1-qsi-eta)*(1-qsi-eta-0.5);
psi(2)=2*qsi*(qsi-0.5);
psi(3)=2*eta*(eta-0.5);
psi(4)=4*(1-qsi-eta)*qsi;
psi(5)=4*qsi*eta;
psi(6)=4*eta*(1-qsi-eta);

dpsdq(1)=4*eta+4*qsi-3;
dpsdq(2)=4*qsi-1;
dpsdq(3)=0;
dpsdq(4)=-4*(eta+2*qsi-1);
dpsdq(5)=4*eta;
dpsdq(6)=-4*eta;

dpsdet(1)=4*eta+4*qsi-3;
dpsdet(2)=0;
dpsdet(3)=4*eta-1;
dpsdet(4)=-4*qsi;
dpsdet(5)=4*qsi;
dpsdet(6)=-4*(2*eta+qsi-1);
```

```
%#########################################################################
% Preparando os dados para aplicar as condicoes de contorno de Neumann
%#########################################################################

%#########################################################################
%É necessário informar as aresta do contorno e o os respectivos valores
%#########################################################################

arestaaux=c_natural;
valor=valor_natural;
n_arestaaux=size(arestaaux,1);

% Criando mais duas colunas na matriz aresta, para aramazenar as
% informações do elemento
lixo=zeros(1,n_arestaaux)';
arestaaux=[arestaaux lixo];

% Os triângulos auxiliares são utilizados no cálculo do Jacobiano,
% etc... e nn=indica qual a coluna que estará armazenado a informacao
% do tipo do elemento (1, 2 ou 3), ou seja, em qual lado está dS.

if tipo_elemento==1
   tri_aux=zeros(n_arestaaux,3);
   nn=3;
end

if tipo_elemento==2
   tri_aux=zeros(n_arestaaux,6);
   nn=4;
end

% O procedimento abaixo é para encontrar em qual dos lados do triangulo
% está dS. Esta informacao é armazenada na mesma matriz do tri_aux.

for i=1:n_tri
   teste=tri(i,:);
   for j=1:n_arestaaux
      if (arestaaux(j,1)==teste(1) & arestaaux(j,2)==teste(2)) |
         (arestaaux(j,1)==teste(2) & arestaaux(j,2)==teste(1))
         if tipo_elemento==1
            arestaaux(j,3)=1;
         else
            arestaaux(j,4)=1;
         end
         tri_aux(j,:)=teste;
      end

      if (arestaaux(j,1)==teste(2) & arestaaux(j,2)==teste(3)) |
         (arestaaux(j,1)==teste(2) & arestaaux(j,2)==teste(3))
         if tipo_elemento==1
            arestaaux(j,3)=3;
         else
            arestaaux(j,4)=3;
         end
         tri_aux(j,:)=teste;
      end

      if (arestaaux(j,1)==teste(1) & arestaaux(j,2)==teste(3)) |
         (arestaaux(j,1)==teste(3) & arestaaux(j,2)==teste(1))
         if tipo_elemento==1
```

```
            arestaaux(j,3)=2;
         else
            arestaaux(j,4)=2;
         end
         tri_aux(j,:)=teste;
      end
   end
end

%#########################################################################
% Verificar a contribuição para cada nó... armazenando as informações
% em SG()- correspondente. Para depois subtrair de FG
%#########################################################################

SG1=zeros(n_pts,1);

for i=1:n_arestaaux
   aresta=arestaaux(i,:);
   if aresta(nn)==1
      tri_neumann=tri_aux(i,:);
      valor1=valor(i,:);
      % Arquivo que faz a integração numérica neumann1.m (para esse
      caso)
      neumann1
      if tipo_elemento==1
         SG1(aresta(1))=SG1(aresta(1))+contribui1;
         SG1(aresta(2))=SG1(aresta(2))+contribui2;
      end

      if tipo_elemento==2
         SG1(aresta(1))=SG1(aresta(1))+contribui1;
         SG1(aresta(2))=SG1(aresta(2))+contribui2;
         SG1(aresta(3))=SG1(aresta(3))+contribui3;
      end
   end

   if aresta(nn)==2
      tri_neumann=tri_aux(i,:);
      valor1=valor(i,:);
      % Arquivo que faz a integração numérica neumann1.m (para esse caso)
      neumann2
      if tipo_elemento==1
         SG1(aresta(1))=SG1(aresta(1))+contribui1;
         SG1(aresta(2))=SG1(aresta(2))+contribui2;
      end

      if tipo_elemento==2
         SG1(aresta(1))=SG1(aresta(1))+contribui1;
         SG1(aresta(2))=SG1(aresta(2))+contribui2;
         SG1(aresta(3))=SG1(aresta(3))+contribui3;
      end
   end
```

```
    if aresta(nn)==3
        tri_neumann=tri_aux(i,:);
        valor1=valor(i,:);
        % Arquivo que faz a integração numérica neumann1.m (para esse caso)
        neumann3
        if tipo_elemento==1
            SG1(aresta(1))=SG1(aresta(1))+contribui1;
            SG1(aresta(2))=SG1(aresta(2))+contribui2;
        end

        if tipo_elemento==2
            SG1(aresta(1))=SG1(aresta(1))+contribui1;
            SG1(aresta(2))=SG1(aresta(2))+contribui2;
            SG1(aresta(3))=SG1(aresta(3))+contribui3;
        end
    end
end
```

```
%#################################################################
% Integracao - para L1
%#################################################################

   qsi_gauss=[0.7745966692 0 -0.7745966692];
   peso=[5/9 8/9 5/9];
   n_pontos_gauss=3;
   %transformar..... qsi=(qsi_gauss+1)/2

   % Elementos Lineares
   if tipo_elemento==1
      for i=1:3
         qsi=qsi_gauss(i);
         a=1-((qsi+1)/2);
         b=((qsi+1)/2);
         c=0;
         aa=-1;
         bb=1;
         cc=0;

         jacobi1(i)=((x(tri_neumann(1))*aa+x(tri_neumann(2))*bb)^2+(y(tri_n
         eumann(1))*aa+y(tri_neumann(2))*bb)^2)^0.5;
         interpola(i)=valor1(1)*a+valor1(2)*b;
         psi1(i)=a;
         psi2(i)=b;
      end

      contribui1=(interpola(1)*psi1(1)*jacobi1(1)*peso(1)+interpola(2)*psi1
      (2)*jacobi1(2)*peso(2)+interpola(3)*psi1(3)*jacobi1(3)*peso(3));

      contribui2=(interpola(1)*psi2(1)*jacobi1(1)*peso(1)+interpola(2)*psi2
      (2)*jacobi1(2)*peso(2)+interpola(3)*psi2(3)*jacobi1(3)*peso(3));
   end
```

```
% Elementos Quadráticos
if tipo_elemento==2
   for i=1:3
      qsi=qsi_gauss(i);

      a=2*(1-((qsi+1)/2))*(1-((qsi+1)/2)-0.5);
      b=2*((qsi+1)/2)*(((qsi+1)/2)-0.5);
      c=0;
      d=4*(1-((qsi+1)/2))*((qsi+1)/2);
      e=0;
      f=0;

      aa=4*((qsi+1)/2)-3;
      bb=4*((qsi+1)/2)-1;
      cc=0;
      dd=-4*(2*((qsi+1)/2)-1);
      ee=0;
      ff=0;

      jacobi1(i)=((x(tri_neumann(1))*aa+x(tri_neumann(2))*bb+x(tri_neuma
      nn(4))*dd)^2+(y(tri_neumann(1))*aa+y(tri_neumann(2))*bb+y(tri_neum
      ann(4))*dd)^2)^0.5;
      interpola(i)=valor1(1)*a+valor1(2)*b+valor1(3)*d;

      psi1(i)=a;
      psi2(i)=b;
      psi3(i)=d;
   end



   contribui1=(interpola(1)*psi1(1)*jacobi1(1)*peso(1)+interpola(2)*psi1
   (2)*jacobi1(2)*peso(2)+interpola(3)*psi1(3)*jacobi1(3)*peso(3));

   contribui2=(interpola(1)*psi2(1)*jacobi1(1)*peso(1)+interpola(2)*psi2
   (2)*jacobi1(2)*peso(2)+interpola(3)*psi2(3)*jacobi1(3)*peso(3));

   contribui3=(interpola(1)*psi3(1)*jacobi1(1)*peso(1)+interpola(2)*psi3
   (2)*jacobi1(2)*peso(2)+interpola(3)*psi3(3)*jacobi1(3)*peso(3));
end
```

```
%#############################################################
% Integracao - para L2
%
%#############################################################

eta_gauss=[0.7745966692 0 -0.7745966692];
peso=[5/9 8/9 5/9];
n_pontos_gauss=3;


% Elementos Lineares
if tipo_elemento==2
      for i=1:3
         eta=eta_gauss(i);

         a=1-((eta+1)/2);
         b=0;
         c=((eta+1)/2);

         aa=-1;
         bb=0;
         cc=1;


         jacobi1(i)=((x(tri_neumann(1))*aa+x(tri_neumann(3))*cc)^2+(y(tri_n
         eumann(1))*aa+y(tri_neumann(3))*cc)^2)^0.5;
         interpola(i)=valor1(1)*a+valor1(3)*c;

         psi1(i)=a;
         psi2(i)=c;
      end


    contribui1=interpola(1)*psi1(1)*jacobi1(1)*peso(1)+interpola(2)*psi1(2
    )*jacobi1(2)*peso(2)+interpola(3)*psi1(3)*jacobi1(3)*peso(3);

    contribui2=interpola(1)*psi2(1)*jacobi1(1)*peso(1)+interpola(2)*psi2(2
    )*jacobi1(2)*peso(2)+interpola(3)*psi2(3)*jacobi1(3)*peso(3);
end
```

```
% Elementos Quadráticos
if tipo_elemento==2
   for i=1:3
      eta=eta_gauss(i);

      a=2*(1-((eta+1)/2))*(1-((eta+1)/2)-0.5);
      b=0;
      c=2*((eta+1)/2)*(((eta+1)/2)-0.5);
      d=0;
      e=0;
      f=4*((eta+1)/2)*(1-((eta+1)/2));

      aa=4*((eta+1)/2)-3;
      bb=0;
      cc=4*((eta+1)/2)-1;
      dd=0;
      ee=0
      ff=-4*(2*((eta+1)/2)-1);


      jacobi1(i)=((x(tri_neumann(1))*aa+x(tri_neumann(3))*cc+x(tri_neuma
      nn(6))*ff)^2+(y(tri_neumann(1))*aa+y(tri_neumann(3))*cc+y(tri_neum
      ann(6))*ff)^2)^0.5;
      interpola(i)=valor1(1)*a+valor1(2)*c+valor1(3)*f;

      psi1(i)=a;
      psi2(i)=c;
      psi3(i)=f;
   end

   contribui1=interpola(1)*psi1(1)*jacobi1(1)*peso(1)+interpola(2)*psi1(
   2)*jacobi1(2)*peso(2)+interpola(3)*psi1(3)*jacobi1(3)*peso(3);

   contribui2=interpola(1)*psi2(1)*jacobi1(1)*peso(1)+interpola(2)*psi2(
   2)*jacobi1(2)*peso(2)+interpola(3)*psi2(3)*jacobi1(3)*peso(3);

   contribui3=interpola(1)*psi3(1)*jacobi1(1)*peso(1)+interpola(2)*psi3(
   2)*jacobi1(2)*peso(2)+interpola(3)*psi3(3)*jacobi1(3)*peso(3);
end
```

```
%#################################################################
% Integracao - para L3
%
%#################################################################

   qsi_gauss=[0.7745966692 0 -0.7745966692];
   peso=[5/9 8/9 5/9];
   n_pontos_gauss=3;


   % Elementos Lineares
   if tipo_elemento==1
      for i=1:3
         qsi=qsi_gauss(i);

         a=0;
         b=((qsi+1)/2);
         c=1-((qsi+1)/2);

         aa=0;
         bb=1;
         cc=-1;


         jacobi1(i)=((x(tri_neumann(2))*bb+x(tri_neumann(3))*cc)^2+(y(tri_n
         eumann(2))*bb+y(tri_neumann(3))*cc)^2)^0.5;
         interpola(i)=valor1(1)*b+valor1(2)*c;

         psi1(i)=b;
         psi2(i)=c;
      end


      contribui1=interpola(1)*psi1(1)*jacobi1(1)*peso(1)+interpola(2)*psi1(
      2)*jacobi1(2)*peso(2)+interpola(3)*psi1(3)*jacobi1(3)*peso(3);

      contribui2=interpola(1)*psi2(1)*jacobi1(1)*peso(1)+interpola(2)*psi2(
      2)*jacobi1(2)*peso(2)+interpola(3)*psi2(3)*jacobi1(3)*peso(3);
   end
```

```
% Elementos Quadráticos
if tipo_elemento==2
   for i=1:3
      eta=eta_gauss(i);

      a=2*(1-((eta+1)/2))*(1-((eta+1)/2)-0.5);
      b=0;
      c=2*((eta+1)/2)*(((eta+1)/2)-0.5);
      d=0;
      e=0;
      f=4*((eta+1)/2)*(1-((eta+1)/2));

      aa=4*((eta+1)/2)-3;
      bb=0;
      cc=4*((eta+1)/2)-1;
      dd=0;
      ee=0
      ff=-4*(2*((eta+1)/2)-1);


      jacobi1(i)=((x(tri_neumann(1))*aa+x(tri_neumann(3))*cc+x(tri_neuma
      nn(6))*ff)^2+(y(tri_neumann(1))*aa+y(tri_neumann(3))*cc+y(tri_neum
      ann(6))*ff)^2)^0.5;
      interpola(i)=valor1(1)*a+valor1(2)*c+valor1(3)*f;

      psi1(i)=a;
      psi2(i)=c;
      psi3(i)=f;
   end

   contribui1=interpola(1)*psi1(1)*jacobi1(1)*peso(1)+interpola(2)*psi1(
   2)*jacobi1(2)*peso(2)+interpola(3)*psi1(3)*jacobi1(3)*peso(3);

   contribui2=interpola(1)*psi2(1)*jacobi1(1)*peso(1)+interpola(2)*psi2(
   2)*jacobi1(2)*peso(2)+interpola(3)*psi2(3)*jacobi1(3)*peso(3);

   contribui3=interpola(1)*psi3(1)*jacobi1(1)*peso(1)+interpola(2)*psi3(
   2)*jacobi1(2)*peso(2)+interpola(3)*psi3(3)*jacobi1(3)*peso(3);
end
```

```
%#########################################################################%
Eliminando as linhas e colunas conhecidas
% Condições de contorno essenciais
%#########################################################################
function [KGNN, FGN]=eliminacc(KG,FG, c_essencial, n_pts, n_c_essencial);

i=1;
KGN=[];
FGN=[];
II=1;
IT=c_essencial(i)-1;

% Eliminando as linhas que correspondem as condições de contorno
% conhecidas
while II <= c_essencial(n_c_essencial)-1
   KG1=KG(II:IT,:);
   KGN=[KGN;KG1];
   FG1=FG(II:IT);
   FGN=[FGN;FG1];

   i=i+1;
   II=IT+2;
   if II < c_essencial(n_c_essencial)
      IT=c_essencial(i)-1 ;
   end
end
KG1=KG(c_essencial(n_c_essencial)+1:n_pts,:);
KGN=[KGN; KG1];

FG1=FG(c_essencial(n_c_essencial)+1:n_pts);
FGN=[FGN;FG1];
i=1;

KGNN=[];

II=1;
IT=c_essencial(i)-1;

% Eliminando as colunas que correspondem as condições de contorno
% conhecidas
while II <= c_essencial(n_c_essencial)-1
   KG1=KGN(:,II:IT);
   KGNN=[KGNN KG1];

   i=i+1;
   II=IT+2;

   if II < c_essencial(n_c_essencial)
      IT=c_essencial(i)-1 ;
   end
end
KG1=KGN(:,c_essencial(n_c_essencial)+1:n_pts);
KGNN=[KGNN KG1];
```

```
[nn nn]=size(KGNN);

%Eliminacao Gaussiana

% Deve-se eliminar os triângulos com condições de contorno prescritas.


% Neste passo eu percorro todos os triângulos e coloco zero nos locais
% com CC prescritas

 tri_banda=tri;
 for i=1:n_tri
   aux=tri_banda(i,:);
   for j=1:n_c_essencial
      teste=c_essencial(j)
      [ii,jj]=find(tri_banda==teste);
      teste1=isempty(ii);
      if teste1 ~= 1
         tri_banda(ii,jj)=0;
      end
   end
end


% Para verificar o tamanho da banda, é só verificar a diferenca entre o
% primeiro e o último valor dos triangulos, desconsiderando os que
% tiverem o valor = 0

 for i=1:n_tri_banda
   triteste=sort(tri_banda(i.:));
   if teste1 ~= 0
      banda(i)=triteste(3)-triteste(1);
   end
end

banda=max(banda);


% Agora armazena-se somente meia banda

  A=zeros(nn, banda);
  nnn=0;
  for i=1:nn
     for j=1:banda
        if j+nnn > nn
           A(i,j)=A(i,j);
        else
           A(i,j)=A(i,j)+KGNN(i,j+nnn);

        end
     end
     nnn=nnn+1;
  end

  KGNN=[A];

tic;
```

```
% Zerando abaixo da diagonal principal
for k=1:nn-1
    for i=2:banda
       ii=i+(k-1);
       if ii <= nn
          mki=KGNN(k,i)/KGNN(k,1);
          jj=0;
          for j=i:banda
             jj=jj+1;
             KGNN(ii,jj)=KGNN(ii,jj)-mki*KGNN(k,j);
          end
          FGN(ii)=FGN(ii)-mki*FGN(k);
       end
    end
end

% Retrosubstituicao

KGNN=[KGNN FGN];

u(nn)=KGNN(nn,banda+1)/KGNN(nn,1);
for i=nn-1:-1:1
   u(i)=KGNN(i,banda+1);
   for j=banda:-1:2
      jj=i+(j-1);
      if jj <= nn
         u(i)=u(i)-(KGNN(i,j)*u(jj));
      end
   end
   u(i)=u(i)/KGNN(i,1);
end

tempo1=toc;

u=u';
```