ESTE EXEMPLAR CORRESPONDE A REDAÇÃO FINAL DA TESE DEFENDIDA POR Geraldo Creci Filho E APROVADA PELA COMISSÃO JULGADORA EM 27 / 02 / 04.

ORIENTADOR

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA

Geração de Malha Tridimensional para o Método dos Elementos de Contorno

Autor: **Geraldo Creci Filho**

Orientador: **Prof. Dr. Paulo Sollero**

Co-orientador: **Prof. Dr. Eder Lima de Albuquerque**

06-04

BIBLIOTECA CENTRAL
DESENVOLVIMENTO
COLEÇÃO
UNICAMP

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
DEPARTAMENTO DE MECÂNICA COMPUTACIONAL

Geração de Malha Tridimensional para o Método dos Elementos de Contorno

Autor: **Geraldo Creci Filho**

Orientador: **Prof. Dr. Paulo Sollero**

Co-orientador: **Prof. Dr. Eder Lima de Albuquerque**

Curso: Engenharia Mecânica

Área de concentração: Mecânica dos Sólidos e Projeto Mecânico

Dissertação de mestrado apresentada à comissão de Pós-Graduação da Faculdade de Engenharia Mecânica, como requisito para a obtenção do título de Mestre em Engenharia Mecânica.

Campinas, 2004

S.P. - Brasil

UNIDADE BC
Nº CHAMADA
T/UNICAMP
C86g
EX
TOMBO BC/ 66040
PROC. 16-86-05
C D
PREÇO 11,00
DATA 26-10-05
Nº CPD
BIBID - 366649

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

C86g

Creci Filho, Geraldo
Geração de malha tridimensional para o método dos elementos de contorno / Geraldo Creci Filho. --Campinas, SP: [s.n.], 2004.

Orientadores: Paulo Sollero e Eder Lima de Albuquerque.
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Mecânica.

1. Triangulação. 2. Geração numérica de malhas (Análise numérica). 3. Métodos de elementos de contorno. 4. Geometria computacional. 5. Computação Gráfica. I. Sollero, Paulo. II. Albuquerque, Eder Lima de. III. Universidade Estadual de Campinas. Faculdade de Engenharia Mecânica. IV. Título.

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
DEPARTAMENTO DE MECÂNICA COMPUTACIONAL

DISSERTAÇÃO DE MESTRADO ACADÊMICO

Geração de Malha Tridimensional para o Método dos Elementos de Contorno

Autor: **Geraldo Creci Filho**
Orientador: **Prof. Dr. Paulo Sollero**
Co-orientador: **Prof. Dr. Éder Lima de Albuquerque**

Prof. Dr. Paulo Sollero, Presidente
Universidade Estadual de Campinas

Prof. Dr. Jonas de Carvalho
Universidade de São Paulo

Prof. Dr. Janito Vaqueiro Ferreira
Universidade Estadual de Campinas

Campinas, 27 de fevereiro de 2004.

BIBLIOTECA CENTRAL
DESENVOLVIMENTO
COLEÇÃO
UNICAMP

Dedicatória

Dedico este trabalho a Deus, aos meus pais e ao meu irmão.

Agradecimentos

Agradeço a CAPES pelo apoio financeiro e aos professores e pesquisadores do Departamento de Mecânica Computacional da Faculdade de Engenharia Mecânica da Unicamp pelo ótimo acolhimento e apoio técnico e científico.

Agradeço a todos os meus novos colegas e amigos que fiz durante a realização deste trabalho. Sou grato em especial pelos momentos de alegria e pela convivência que muito me orgulha e que jamais esquecerei.

Ao meu orientador, Paulo Sollero, e ao meu co-orientador, Eder Lima de Albuquerque, agradeço pela oportunidade de desenvolver esse trabalho e pelo constante estímulo.

Gostaria, também, de mencionar minha sincera admiração e gratidão ao meu orientador de iniciação científica, Gerald Weber, que fundamentou minha base científica e muito contribuiu para o meu crescimento profissional.

Por fim e, talvez mais importante, agradeço a Pedro Yoshito Noritomi, principal desenvolvedor do *solver* em Elementos de Contorno, E-Con3D, pela incansável ajuda e generosidade em compartilhar seus fascinantes conhecimentos de engenharia.

"É melhor calar-se e deixar que as pessoas pensem que você é um idiota do que falar e acabar com a dúvida."

Abraham Lincoln

Resumo

CRECI, Geraldo Filho, *Geração de Malha Tridimensional para o Método dos Elementos de Contorno*, Campinas: Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2004. 143 p. Dissertação (Mestrado).

Este trabalho apresenta o desenvolvimento de um programa de geração de malhas não estruturadas que serão usadas em análises numéricas pelo método dos elementos de contorno. O método dos elementos de contorno é um dos métodos numéricos de destaque usados atualmente em simulações computacionais. Ele possui algumas características particulares que o tornam mais atrativo em determinados tipos de aplicações, quando comparado com outros métodos numéricos. Dentre essas características, está a vantagem de se discretizar somente o contorno das geometrias para o cálculo da solução aproximada do problema. Isso significa dizer que em casos tridimensionais somente elementos de superfície precisam ser gerados na malha. O gerador de malhas implementado neste trabalho foi desenvolvido para gerar malhas com elementos triangulares-lineares em geometrias tridimensionais compostas por faces planas. A idéia geral consiste em deslocar cada face que compõe a geometria tridimensional para o espaço bidimensional usando-se transformações geométricas e, em seguida, aplicar o algoritmo de triangulação de Delaunay para geração dos elementos. Depois de gerados os elementos, novas transformações geométricas são aplicadas a fim de enviar a face de volta para sua posição original no espaço tridimensional. A continuidade da malha é assegurada promovendo-se a geração dos nós nas arestas antes da geração dos nós no interior das faces. Vários exemplos de malhas em geometrias tridimensionais são apresentados para ilustrar a capacidade do programa desenvolvido e algumas análises numéricas foram feitas para demonstrar a qualidade das malhas em problemas com condições de contorno específicas.

Palavras-Chave: Triangulação de Delaunay, Geração de Malha, Método dos Elementos de Contorno, Geometria Computacional, Computação Gráfica, Transformações Geométricas.

Abstract

CRECI, Geraldo Filho, *Three-Dimensional Mesh Generation for Boundary Element Method*, Campinas: Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2004. 143 p. Dissertação (Mestrado).

This work presents the development of an unstructured mesh generator to be used in numerical analysis by the boundary element method. The boundary element method is one of the prominence methods recently used in computational simulations. It has some particular characteristics that make it more favorable in certain types of applications, when compared to other numerical methods. Among those characteristics, one is especially important from the mesh generation point of view. It is the fact that, in boundary element method, it is only necessary the discretization of the geometry boundaries. In other words, in three-dimensional cases only surface elements should be generated. The mesh generator implemented in this work has been developed to generate meshes with triangular-linear elements over three-dimensional geometries composed by plane faces. The general idea consists of moving each face belonging to the three-dimensional geometry to bi-dimensional space using geometrical transformations and the Delaunay triangulation algorithm for element generation. After element generation, new geometrical transformations are applied in order to send the meshed face back to its original position in three-dimensional space. The continuity of the final mesh is assured by generating the nodes of the edges of the geometry prior to the generation of the nodes in the interior of the face. Several examples of meshes in three-dimensional geometries are presented to illustrate the capabilities of the developed program and some numerical analyses have been performed to show the quality of the meshes in problems with specific boundary conditions.

Key-Words: Delaunay Triangulation, Mesh Generation, Boundary Element Method, Computational Geometry, Graphics Computing, Geometrical Transformations.

Sumário

Lista de Figuras iv

Lista de Tabelas vii

Nomenclatura viii

1 Introdução 1

1.1 Introdução 1

1.2 Motivação e Objetivos 3

1.3 Contribuições deste Trabalho 5

1.4 Descrição do Trabalho 6

2 Revisão Bibliográfica 9

2.1 Considerações Iniciais Sobre Malhas 9

2.1.1 Propriedades de Natureza Geométrica 9

2.1.2 Propriedades de Natureza Física 10

2.2 Técnicas de Geração de Malha 11

2.2.1 Geração de Malha por Transporte-Mapeamento 11

2.2.2 Geração de Malha por Solução de um Sistema de EDP 13

2.2.3 Geração de Malha por Deformação de um *Grid* de Pontos 15

2.2.4 Geração de Malha por Avanço de Fronte 16

2.2.5 Geração de Malha por Voronoï-Delaunay 19

2.3 Revisão Bibliográfica 24

3 Método dos Elementos de Contorno para Elasticidade **29**

3.1 Equações Básicas da Elasticidade 29

3.2 Formulação do Método dos Elementos de Contorno 31

3.2.1 As Soluções Fundamentais de Kelvin 32

3.2.2 Teorema da Reciprocidade de Betti 33

3.2.3 Equações Integrais de Contorno para os Deslocamentos 35

3.2.4 Discretização das Equações Integrais de Contorno 37

4 Implementação Computacional **41**

4.1 Programação Orientada a Objetos 41

4.1.1 Obtendo um Modelo Orientado a Objetos 42

4.2 Características do Gerador 43

4.3 Estrutura de Dados Usada pelo Programa 44

4.4 Fluxograma do Programa 49

4.5 Diagrama de Classes 51

4.5.1 Estrutura das Classes 51

4.6 Transformações Geométricas 57

4.7 Geração do *Grid* de Pontos 59

4.8 Seleção dos Pontos que Serão os Nós da Malha 61

4.9 Triangulação de Delaunay 64

4.9.1 O Algoritmo de Geração dos Elementos 65

4.9.2 Determinação da Intersecção de Dois Segmentos 67

4.10 Critério de Qualidade das Malhas 68

4.10.1 Cálculo do Ângulo Interno Mínimo de um Triângulo 70

5 Resultados e Discussões **71**

5.1 Interface de Visualização 71

5.1.1 Programa de Visualização das Malhas em MatLAB® 74

5.2 Geração de Malhas em Geometrias Tridimensionais 75

5.2.1 Tabelas com Parâmetros de Qualidade da Malhas Geradas 84

5.3 Análises Numéricas 88
5.3.1 Primeiro Estudo de Caso 88
5.3.2 Segundo Estudo de Caso 90
5.3.3 Terceiro Estudo de Caso 93

6 Conclusões e Sugestões para Próximos Trabalhos **97**
6.1 Conclusões 97
6.2 Publicações 99
6.3 Sugestões para Trabalhos Futuros 100

Referências Bibliográficas **103**

Anexo A: OpenGL® e MS Visual C++ **109**

Anexo B: Elementos Finitos do Ansys® - (*shell63*, *solid45* e *solid92*) **117**

Lista de Figuras

2.1 Exemplo de malha conforme. 10

2.2 Exemplo de malha não-conforme. 10

2.3 Transporte-mapeamento usando um quadrilátero unitário. 12

2.4 Os dois sistemas de variáveis envolvidos no método. 14

2.5 Aplicação de um *grid* regular de pontos para geração da malha. 16

2.6 Eliminação dos quadriláteros não necessários a geração da malha. 16

2.7 Malha sem a aplicação da função de deformação $\mathcal{F}_\delta$. 16

2.8 Malha final com a aplicação da função de deformação $\mathcal{F}_\delta$. 16

2.9 Fluxograma simplificado de execução do algoritmo de avanço de fronte. . . . 17

2.10 Situação em que $\alpha < \frac{\pi}{2}$. 18

2.11 Situação em que $\frac{\pi}{2} \leqslant \alpha \leqslant \frac{2\pi}{3}$. 18

2.12 Situação em que $\frac{2\pi}{3} < \alpha$. 18

2.13 Fronte inicial formado pelo contorno da geometria. 19

2.14 Novos segmentos que constituem o fronte após primeira iteração. 19

2.15 Novos segmentos que constituem o fronte após segunda iteração. 19

2.16 Geometria com a malha final gerada após todas as iterações. 19

2.17 Diagrama de Voronoï para um conjunto de 3 pontos no plano. 20

2.18 Diagrama de Voronoï para um conjunto de 8 pontos no plano. 21

2.19 Situação de quatro pontos em uma circunferência de Delaunay. 22

2.20 Triangulação de Delaunay do diagrama de Voronoï da figura (2.18). 22

2.21 Propriedade dos circuncírculos vazios da triangulação de Delaunay $\mathcal{D}(P)$. . . 23

2.22 Triângulo formado por p_1, p_2 e p_3 circunscrito a uma circunferência. 24

2.23 Propriedade de maximização dos ângulos internos mínimos de $\mathcal{D}(P)$. 24

3.1 Modelagem de um problema típico de valor de contorno. 31
3.2 Contorno auxiliar usado para o processo de limite ($X' \rightarrow x'$). 36
3.3 Exemplo de malhas 2D: (a) elementos de contorno e (b) elementos finitos. . . 38
3.4 Elemento de contorno triangular-linear contínuo de três nós. 38

4.1 Exemplo de geometria tridimensional aceita pelo programa. 44
4.2 Formatos do arquivo de entrada (a) e do arquivo de saída (b). 45
4.3 Processo de obtenção do arquivo de entrada de dados para o gerador. 45
4.4 Orientação do vetor normal com relação a ordem dos segmentos S_1 e S_2. . . 47
4.5 Orientação do vetor normal da face calculado a partir de $\mathbf{v_1}$ e $\mathbf{v_2}$. 47
4.6 Fluxograma do processo para realização de uma análise numérica. 48
4.7 Fluxograma de execução do gerador de malhas implementado ECon-MG. . . 50
4.8 Diagrama de classe que relaciona a classe *Node* com a classe *Element*. 52
4.9 Diagrama apresentando a modelagem das classes *Segment*, *Facet* e *Edge*. . . 54
4.10 Classe *GeoTrafo* modelada usando a notação UML. 55
4.11 Classe *Mesh* modelada usando a notação UML. 56
4.12 Processo para deslocar a face F_i no espaço $\Re^3$ para o plano xy. 57
4.13 Geração do *grid* de pontos para a triangulação de Delaunay. 59
4.14 Ilustração do algoritmo para determinação de ponto interno a uma face. . . . 61
4.15 Eliminação dos pontos muito próximos a algum ponto do contorno. 63
4.16 Conjunto de pontos final que serão os nós da malha. 63
4.17 Propriedade do circuncírculo vazio da triangulação de Delaunay. 64
4.18 Triângulos gerados fora do domínio da face. 66
4.19 Malha final da face obtida pelo conjunto de pontos da figura (4.16). 67
4.20 Ilustração da intersecção de dois segmentos. 67
4.21 Triângulo qualquer da malha, onde α é o ângulo mínimo. 69
4.22 Triângulo qualquer da malha, onde *ang_min* é o ângulo mínimo. 70

5.1 Malha de domínio bidimensional com 421 nós e 649 elementos. 72
5.2 Malha de domínio bidimensional com 356 nós e 602 elementos. 72

5.3 Malha de uma geometria multifacetada com 162 nós e 247 elementos. 73
5.4 Malha 3D para elementos de contorno com 655 nós e 1314 elementos. 76
5.5 Malha 3D para elementos de contorno com 787 nós e 1582 elementos. 76
5.6 Malha 3D para elementos de contorno com 1508 nós e 3020 elementos. . . . 77
5.7 Malha 3D para elementos de contorno com 1776 nós e 3624 elementos. . . . 77
5.8 Malha 3D para elementos de contorno com 766 nós e 1532 elementos. 78
5.9 Malha 3D para elementos de contorno com 2721 nós e 5442 elementos. . . . 79
5.10 Malha 3D para elementos de contorno com 2039 nós e 4074 elementos. . . . 79
5.11 Malha 3D para elementos de contorno com 1722 nós e 3444 elementos. . . . 80
5.12 Malha 3D para elementos de contorno com 1107 nós e 2210 elementos. . . . 80
5.13 Malha 3D para elementos de contorno com 1536 nós e 3072 elementos. . . . 81
5.14 Malha 3D para elementos de contorno com 962 nós e 1920 elementos. 81
5.15 Malha 3D para elementos de contorno com 1148 nós e 2292 elementos. . . . 82
5.16 Malha 3D para elementos de contorno com 957 nós e 1910 elementos. 82
5.17 Malha 3D para elementos de contorno com 1737 nós e 3474 elementos. . . . 83
5.18 Malha 3D para elementos de contorno com 1058 nós e 2116 elementos. . . . 83
5.19 Análise usando o *solver* E-Con3D com malha gerada pelo ECon-MG. 89
5.20 Análise usando o *solver* E-Con3D com malha gerada pelo Ansys®. 89
5.21 Variação da solução ao longo do raio do tubo. 90
5.22 Análise usando o *solver* E-Con3D com malha gerada pelo Ansys®. 91
5.23 Análise usando o *solver* E-Con3D com malha gerada pelo ECon-MG. 91
5.24 Análise usando o *solver* Ansys® com malha gerada pelo próprio Ansys®. . . 92
5.25 Malha gerada pelo ECon-MG com 778 nós e 1564 elementos. 93
5.26 Malha do Ansys® com 624 nós e 1846 elementos, usando *solid45*. 94
5.27 Malha do Ansys® com 11968 nós e 22525 elementos, usando *solid92*. 94

Lista de Tabelas

5.1 Dados para avaliação da qualidade da malha da geometria (5.4). 84
5.2 Dados para avaliação da qualidade da malha da geometria (5.5). 84
5.3 Dados para avaliação da qualidade da malha da geometria (5.6). 85
5.4 Dados para avaliação da qualidade da malha da geometria (5.7). 85
5.5 Dados para avaliação da qualidade da malha da geometria (5.8). 85
5.6 Dados para avaliação da qualidade da malha da geometria (5.9). 85
5.7 Dados para avaliação da qualidade da malha da geometria (5.10). 86
5.8 Dados para avaliação da qualidade da malha da geometria (5.11). 86
5.9 Dados para avaliação da qualidade da malha da geometria (5.12). 86
5.10 Dados para avaliação da qualidade da malha da geometria (5.13). 86
5.11 Dados para avaliação da qualidade da malha da geometria (5.14). 87
5.12 Dados para avaliação da qualidade da malha da geometria (5.15). 87
5.13 Dados para avaliação da qualidade da malha da geometria (5.16). 87
5.14 Dados para avaliação da qualidade da malha da geometria (5.17). 87
5.15 Dados para avaliação da qualidade da malha da geometria (5.18). 88

Nomenclatura

Letras Latinas

a_i^k - pontos intermediários localizados na aresta de um polígono

$\hat{a}_i^k$ - pontos intermediários localizados na aresta do elemento de referência

ang_min - ângulo mínimo de um triângulo

b_i - componentes da força de corpo

B_{ij} - bissetor perpendicular ao segmento formado por p_i e p_j

$comp_seg$ - comprimento do segmento

c_v - centro do círculo que circunscreve um triângulo de Delaunay

$\mathcal{C}$ - conjunto de pontos, segmentos e triângulos de uma malha triangular

$\mathcal{CC}$ - conjunto de condições de contorno

$d_{j,j+1}$ - distância entre os pontos a_i^j e a_i^{j+1}

d_{p_i,p_j} - distância entre os pontos p_i e p_j

$\mathcal{D}\{P\}$ - triangulação de Delaunay de um conjunto de pontos P

e_i - elemento triangular no espaço Euclidiano $\Re^2$

E - módulo de elasticidade

$\mathcal{E}$ - conjunto de elementos da malha

f - função qualquer contínua no domínio Ω

f_i - função parametrizada do lado de um polígono em Ω

$\mathcal{F}$ - fronte constituído por uma lista de segmentos

$\mathcal{F}_\delta$ - função de deformação da malha

$\mathcal{F}_t$ - função de transporte-mapeamento

G - matriz que contém as integrais envolvendo os termos U_{ij}

H - matriz que contém as integrais envolvendo os termos T_{ij}

$H(p_i, p_j)$ - semi-plano com contorno B_{ij} contendo p_i

J - Jacobiano da transformação do espaço real para o espaço isoparamétrico

M - número de nós por elemento

$MeshPar$ - parâmetro que determina o grau de refinamento da malha

n_j - componentes do vetor normal unitário

N - número total de elementos da malha

N^m - funções de forma para um elemento de m nós

p_i - ponto no espaço Euclidiano $\Re^2$

P - conjunto de pontos no espaço Euclidiano $\Re^2$

R_i - componentes cartesianas do raio $(R_i = X_i - X'_i)$

$\mathcal{S}$ - sistema de equações diferenciais parciais

t_i - componentes da força de superfície

$\overline{t}_i$ - componentes da força de superfície conhecidas

$T_{ij}(X', X)$ - solução fundamental de Kelvin para forças de superfície

u_i - componentes de deslocamento

$\overline{u}_i$ - componentes de deslocamento conhecidas

$U_{ij}(X', X)$ - solução fundamental de Kelvin para deslocamentos

v_i - vértices de um polígono

$\hat{v}_i$ - vértices do elemento de referência

$V(p_i)$ - região de Voronoï do ponto p_i

$\mathcal{V}(P)$ - diagrama de Voronoï de um conjunto de pontos P

x - coordenada x no espaço Euclidiano

x_i - componentes de coordenadas cartesianas

X_i - componentes cartesianas do ponto campo

X'_i - componentes cartesianas do ponto fonte

y - coordenada y no espaço Euclidiano

z - coordenada z no espaço Euclidiano

Letras Gregas

α - ângulo formado entre dois segmentos

β - ângulo formado entre dois segmentos

γ - ângulo formado entre dois segmentos

μ - módulo de cisalhamento

ν - coeficiente de Poisson

ξ - coordenada ξ no espaço isoparamétrico

η - coordenada η no espaço isoparamétrico

ζ - coordenada ζ no espaço isoparamétrico

κ - representa o número do nó da malha

δ_{ij} - função delta de Kronecker

ε_{ij} - tensor de deformação simétrico

σ_{ij} - tensor de tensão simétrico

ω^*_{ij} - tensor de rotação anti-simétrico

$\Delta(X' - X)$ - função delta de Dirac

∇ - operador de Laplace

Ω - domínio

$\widehat{\Omega}$ - domínio de referência

Γ - contorno

$\widehat{\Gamma}$ - contorno de referência

Nomes e Siglas

Ansys® - pacote comercial de análise numérica em elementos finitos

CAD - projeto auxiliado por computador

DC - contexto de dispositivo

DMC - Departamento de Mecânica Computacional

DLL - biblioteca com vinculação dinâmica

ECon-MG - programa de geração de malhas implementado neste trabalho

BIBLIOTECA CENTRAL
DESENVOLVIMENTO
COLEÇÃO
UNICAMP

E-Con3D - programa de análise numérica em elementos de contorno desenvolvido no DMC

Excel® - programa para geração de planilhas e gráficos

EDP - sistema de equações diferenciais parciais

GDI - dispositivo de interface gráfica

IGES - formato de arquivo de geometria

MatLAB® - ambiente de computação numérica baseado em matrizes

Meflab^{++} - programa de análise numérica em elementos finitos implementado no DMC

MFC - classes de base da Microsoft®

Microsoft® - empresa desenvolvedora de programas computacionais

Microsoft Visual C^{++}® - compilador C^{++} da Microsoft®

Nastran® - pacote comercial de análise numérica em elementos finitos

Newmat10 - biblioteca em C^{++} para cálculo matricial

ON - caracteriza os triângulos que estão dentro da faixa de aspecto aceitável

OFF - caracteriza os triângulos que estão fora da faixa de aspecto aceitável

OpenGL® - biblioteca gráfica em C^{++}

Rational Rose® - modelador de classes da Rational®

RC - contexto de renderização

RGB - sistema de cores baseado no vermelho, verde e azul

SolidEdge® - programa CAD para elaboração de desenhos

STL - arquivos obtidos pelo processo de estereolitografia

UML - notação de modelagem unificada

Unicamp - Universidade Estadual de Campinas

Windows® - sistema operacional desenvolvido pela Microsoft®

Capítulo 1

Introdução

1.1 Introdução

Atualmente a simulação computacional de fenômenos físicos de problemas de engenharia tem sido uma das principais ferramentas que auxiliam os engenheiros no desenvolvimento de projetos. Vários fenômenos físicos, nas mais diversas áreas do conhecimento, podem ser representados por equações diferenciais parciais. Em casos práticos, dificilmente essas equações possuem soluções analíticas. Por isso, métodos numéricos, como o método dos elementos de contorno, são empregados para se obter uma solução aproximada das equações que governam o problema. Problemas envolvendo análises térmicas, análises de tensões, escoamento de fluidos, eletromagnetismo, só para citar alguns, podem ser modelados sob as mais variadas condições de contorno com o objetivo de simular computacionalmente as condições reais de serviço de um determinado componente mecânico. Essa prática tem sido cada vez mais utilizada atualmente, pois proporciona uma significativa economia de tempo e recursos financeiros durante o desenvolvimento de projetos. Vários ensaios práticos e a construção de protótipos, que outrora eram indispensáveis, podem ser substituídos por uma adequada simulação computacional feita por um programa de análise numérica.

A essência do método dos elementos de contorno consiste em transformar a equação ou conjunto de equações diferenciais que são aplicáveis a todo o domínio do problema em equações integrais no contorno. O processo de transformação das equações diferenciais em equações integrais pode ser obtido por diversas formas. Geralmente ele é realizado pelo uso de

soluções fundamentais do problema a ser analisado, em conjunto com um teorema de reciprocidade. Algumas vantagens são obtidas realizando-se esse processo. Dentre elas destaca-se a redução da complexidade do problema em uma dimensão, pois os termos do domínio não serão utilizados para o cálculo computacional e nenhuma aproximação matemática precisa ser introduzida.

Depois de obtidas as equações integrais de contorno é possível calcular o valor da solução do problema em específicos pontos de interesse, chamados de nós. A partir desse conjunto de valores calculados é possível derivar o valor da solução nas demais posições do contorno do problema, usando funções de forma apropriadas para interpolação. Se for necessário, ainda, pode-se posteriormente obter a solução em algum ponto no domínio do problema a partir dos resultados já calculados no contorno.

Portanto, como primeiro passo para esse cálculo numérico é necessário a construção de uma malha, que definirá a posição dos nós e a formação dos elementos na geometria estudada. Os elementos nada mais são que uma conectividade de nós que definem uma região no espaço. Os valores físicos encontrados em cada nó serão interpolados pelas funções de forma inerentes a cada tipo de elemento a fim de se determinar os valores da solução dentro da região definida pelo elemento. Como em elementos de contorno o problema está definido no contorno, somente é preciso gerar nós e elementos no contorno da geometria. Isso quer dizer que em casos tridimensionais somente é necessário gerar nós e elementos na superfície do problema. Isso é extremamente benéfico do ponto de vista de geração de malha, pois possibilita a implementação de um gerador de malhas de forma bem mais simples, robusta e eficiente.

O processo de geração de malhas é uma fase extremamente importante em uma análise numérica, uma vez que, gerar uma malha em uma geometria complexa não é uma tarefa trivial e pode ser bastante custoso do ponto de vista de processamento computacional. Além disso, é crucial que a malha seja gerada de uma forma que melhor se adapte as condições

físicas do problema, pois a qualidade da solução computacional calculada está fortemente relacionada com a qualidade da malha gerada. Neste sentido, é desejável que a malha seja gerada de forma adaptativa, isto é, mais refinada nos locais onde existam altos gradientes de solução e menos refinada nos outros locais. É desejável, também, que essa variação de refinamento ocorra da forma mais suave possível.

Neste trabalho é apresentado um gerador de malhas que foi desenvolvido para modelar problemas tridimensionais que serão analisados pelo método dos elementos de contorno. Na seção seguinte são apresentadas as características do gerador desenvolvido, bem como, os motivos que levaram ao seu desenvolvimento e o contexto em que ele será usado.

1.2 Motivação e Objetivos

O gerador de malhas implementado nesse trabalho foi concebido para ser um novo módulo a ser incorporado nas ferramentas computacionais que vêm sendo desenvolvidas pelo grupo de pesquisa em elementos de contorno, supervisionado pelo professor Dr. Paulo Sollero, da Universidade Estadual de Campinas (Unicamp). Esse grupo de pesquisa faz parte do Departamento de Mecânica Computacional (DMC) da Faculdade de Engenharia Mecânica da Unicamp. Um dos princípios mais fortes que norteiam as pesquisas desenvolvidas por esse grupo está no fato de se investir no desenvolvimento de programas para análises numéricas pelo método dos elementos de contorno com o código fonte aberto. Objetiva-se com isso, a médio e longo prazo, dispor-se de uma plataforma de análise numérica em elementos de contorno implementada em linguagem C^{++} usando o paradigma da orientação a objetos.

A linguagem de programação C^{++} foi escolhida para desenvolvimento dos programas por ser uma linguagem extremamente poderosa e eficiente para aplicações científicas. Além disso, é possível usar ferramentas para modelagem das classes e objetos para facilitar na construção da estrutura do programa e realizar tarefas de manutenção.

Sabe-se, todavia, que existem inúmeros pacotes de análises numéricas disponíveis atualmente no mercado, usando-se os mais variados métodos numéricos. Esses pacotes como o Ansys® e o Nastran®, que usam elementos finitos como formulação de base, já vêm sendo desenvolvidos por um longo período e contam com uma sólida infra-estrutura que possibilita os avanços contínuos e a boa qualidade dos resultados alcançados. Entretanto, esses são pacotes comerciais fechados, ou seja, não possuem o código fonte disponível para alterações ou novas implementações. Se for necessário analisar um problema cuja formulação não tenha sido implementada em um desses pacotes, o engenheiro não terá ferramentas para obter informações suficientes para solucionar o problema e desenvolver o projeto de forma adequada.

Desenvolvendo um programa de análise numérica com o código fonte aberto e escrito de forma bem estruturada, é possível direcionar as pesquisas para a implementação de novas formulações que possibilitem a análise de problemas que não podem ser analisados pelos pacotes acima mencionados. Neste estágio, os verdadeiros e melhores resultados, do ponto de vista de avanços científicos poderão ser obtidos. Entre os trabalhos desenvolvidos pelo grupo de pesquisa supervisionado pelo professor Dr. Paulo Sollero vale ressaltar o trabalho de Albuquerque [1] na análise de problemas dinâmicos em materiais anisotrópicos; o trabalho de Paiva [45] na análise da solução fundamental para placas finas anisotrópicas; e, principalmente, o trabalho de Veras [60] e Noritomi [42] no desenvolvimento de uma interface gráfica para análise de tensões em problemas tridimensionais.

O programa de análise tridimensional de tensões desenvolvido por Veras e Noritomi, chamado de E-Con3D, utiliza elementos de contorno do tipo contínuo, triangular-linear de três nós e isoparamétrico. A solução fundamental implementada é a isotrópica linear homogênea. Essas particularidades definiram as características do gerador de malhas implementado neste trabalho que vai ser usado em conjunto com o E-Con3D. O gerador de malhas desenvolvido tem como objetivo gerar malhas em geometrias tridimensionais simples, compostas por faces planas. Essa aproximação permitiu o desenvolvimento de um programa de geração de malhas tridimensionais suficientemente robusto e eficiente. Mesmo considerando o

fato de que somente geometrias compostas por faces planas podem ser analisadas, o programa implementado permite uma grande liberdade de uso de geometrias com um considerável grau de complexidade, o que para o momento tem se mostrado suficiente. A idéia geral consiste em deslocar cada face que compõe a geometria tridimensional para o espaço bidimensional usando-se transformações geométricas. Depois que a face é deslocada para o espaço bidimensional, aplica-se o algoritmo de triangulação de Delaunay, gerando-se os elementos triangulares na face. Portanto, o problema é reduzido e transportado para o espaço bidimensional. Em seguida, novas transformações geométricas são aplicadas para enviar a face de volta para sua posição no espaço tridimensional. Uma atenção especial é dada para assegurar a continuidade final da malha. Como a geração da malha é feita face a face, poderia existir descontinuidade de nós e elementos nas arestas da geometria. Apesar de em elementos de contorno ser possível o uso de elementos descontínuos, um cuidado especial foi tomado para manter a malha final contínua, o que significa que, as malhas geradas por este gerador, também podem ser usadas em análises tridimensionais por elementos finitos, usando-se a formulação de casca.

1.3 Contribuições deste Trabalho

A fase de pré-processamento é considerada como a fase mais custosa em uma análise numérica, [30]. Isto porque na fase de pré-processamento a malha é gerada e o problema é modelado com suas respectivas condições de contorno. Gerar uma boa malha para análise numérica é uma tarefa bastante difícil, especialmente em problemas envolvendo geometrias tridimensionais complexas e/ou condições de contorno complicadas.

Os algoritmos para geração de malhas tridimensionais são geralmente muito complexos e difíceis de serem implementados para casos arbitrários. Os custos computacionais são altos, [21]. Isto faz com que, ainda hoje em dia, grandes esforços e investimentos sejam realizados com a finalidade de que novas técnicas e novos algoritmos sejam desenvolvidos para melhorar o processo de geração de malha em geometrias tridimensionais.

O programa de geração de malhas proposto neste trabalho possui vários aspectos positivos tais como: simplicidade, eficiência e robustez. O código fica extremamente simplificado uma vez que o problema é transformado do espaço tridimensional para o espaço bidimensional pelo uso de matrizes de transformação geométrica. Portanto, os algoritmos para geração dos elementos são bidimensionais, ou seja, mais confiáveis e simples de serem implementados. Com relação a eficiência e robustez, como cada face da geometria tridimensional é movida para o espaço bidimensional e a malha é gerada sobre uma face por vez, a possibilidade de geração de elementos ruins é minimizada. Como a malha final é construída face a face, o problema é subdividido em vários problemas menores e o algoritmo principal assume o paradigma de dividir-para-conquistar, que assegura alta eficiência e velocidade de processamento.

Todo o código foi escrito usando a linguagem C^{++} orientada a objetos e as classes foram modeladas usando-se a notação UML. Isso faz com que o programa possa ser mais facilmente modificado, os custos de manutenção sejam reduzidos e novas implementações possam ser realizadas conforme a necessidade e desejo do programador. Por tudo isso que foi exposto, pode-se perceber que esse trabalho pode ser útil para aquelas pessoas que possuem um código de geração de malhas bidimensional já implementado e pretendem expandir suas funcionalidades para tratar geometrias tridimensionais simples compostas por faces planas.

1.4 Descrição do Trabalho

Procurou-se escrever esta obra da forma mais clara e simples possível, a fim de possibilitar, realmente, a reprodução e o aprimoramento deste trabalho por qualquer pessoa que possua os conhecimentos mínimos necessários na área. A apresentação da teoria matemática necessária para o desenvolvimento computacional foi apresentada de forma simplificada, detalhando somente os pontos mais importantes. Com isso objetiva-se que o leitor tenha uma leitura agradável e rica em informações nos conceitos importantes que envolvem o desenvolvimento de um gerador de malhas com as características já destacadas. A seguir é apresentado um breve resumo do conteúdo dos capítulos seguintes.

No capítulo 2 é apresentada uma breve explanação sobre as principais técnicas de geração de malha existentes. São apresentados métodos de geração de malha por técnicas de mapeamento, por técnicas que envolvem a solução de um sistema de equações diferenciais parciais (EDP), técnicas baseadas na deformação de um *grid* de pontos e técnicas de geração de malha elemento a elemento a partir das informações do contorno. Além disso, é feita uma breve revisão bibliográfica usando os principais artigos publicados recentemente, que contribuiram para o desenvolvimento do gerador de malhas implementado.

No capítulo 3, a formulação tridimensional do método dos elementos de contorno para elasticidade é descrita. Essa formulação é apresentada para ilustrar o que está implementado no programa E-Con3D e servirá de base teórica para as análises numéricas realizadas com as malhas geradas pelo presente trabalho.

O capítulo 4 trata da implementação computacional realizada. Nele são apresentados o fluxograma de funcionamento do gerador de malhas, o diagrama de classes que define a estrutura do programa, os algoritmos implementados nos principais métodos de cada classe e informações a respeito da estrutura de dados utilizada.

O capítulo 5 apresenta os resultados obtidos. Várias malhas em geometrias tridimensionais arbitrárias são apresentadas para ilustrar a capacidade e versatilidade do gerador de malhas desenvolvido. Além disso, algumas análises numéricas foram realizadas, usando-se o *solver* em elementos de contorno, E-Con3D. Essas análises servem para mostrar a aplicabilidade das malhas geradas e demonstrar a qualidade dos resultados encontrados. O programa Ansys® também é usado para realizar comparações com os resultados obtidos e contribuir para uma melhor análise das soluções calculadas.

No capítulo 6 são apresentadas as conclusões e sugestões para trabalhos futuros.

Capítulo 2

Revisão Bibliográfica

Antes de apresentar uma revisão bibliográfica com os principais trabalhos que ajudaram no desenvolvimento desta obra, são apresentadas algumas considerações iniciais a respeito de malhas para análises numéricas. Em seguida são descritas as principais técnicas de geração de malhas existentes com o objetivo de posicionar esta obra em um contexto mais amplo.

2.1 Considerações Iniciais Sobre Malhas

Uma malha composta por elementos triangulares de um domínio Ω é definida por um conjunto $\mathcal{C}$ constituído por um número finito de pontos, segmentos e triângulos. Os elementos $\mathcal{E}$ de tal malha devem satisfazer algumas propriedades que serão apresentadas nas seções seguintes.

2.1.1 Propriedades de Natureza Geométrica

1. Os elementos devem ser gerados de forma a constituir uma malha conforme, ou seja, os elementos devem preencher todo o domínio, Ω, e a intersecção de dois triângulos em $\mathcal{C}$ deve ser de tal modo que ocorra nos vértices dos triângulos, ou seja reduzida a uma aresta, vide figuras (2.1) e (2.2);

2. É desejável que a variação em tamanho entre dois elementos adjacentes seja feita de forma progressiva e gradual;

3. A densidade de elementos em determinadas regiões do domínio deve ser mais intensa, especialmente nos locais com altos gradientes de solução;

4. Para malhas com elementos triangulares, como é o caso, a existência de ângulos internos muito pequenos deve ser evitada. Considera-se um ângulo em torno de 13° para controle do ângulo mínimo, [50];

5. É desejável, também, que os elementos sejam gerados e orientados por algumas direções preferenciais de acordo com o problema físico em questão.

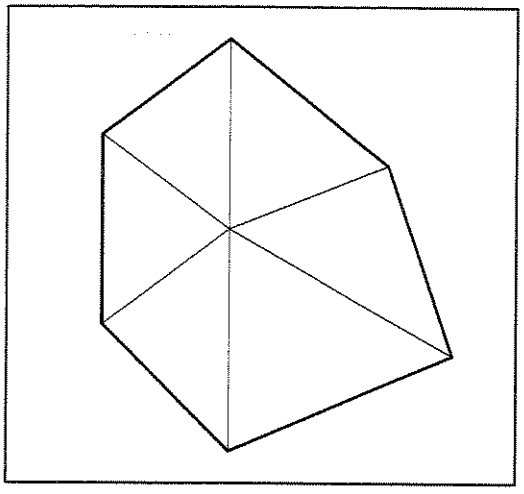


Figura 2.1: Exemplo de malha conforme.

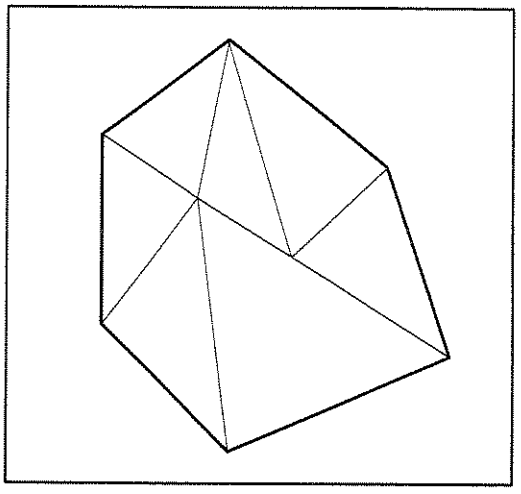


Figura 2.2: Exemplo de malha não-conforme.

2.1.2 Propriedades de Natureza Física

As propriedades de natureza física estão fortemente relacionadas com os aspectos físicos do problema em consideração. Isto talvez seja um dos pontos que originam maior dificuldade durante o processo de desenvolvimento de geradores de malhas. Dependendo do tipo de problema é desejável produzir elementos pequenos, elementos orientados por algumas direções preferenciais, elementos com formas específicas, de modo que o comportamento físico existente no problema seja melhor modelado.

2.2 Técnicas de Geração de Malha

Existem muitas técnicas desenvolvidas destinadas à geração de malhas. Todas elas têm suas vantagens e desvantagens umas em relação às outras. Dentre essas técnicas destacam-se as técnicas de mapeamento, as técnicas que envolvem a solução de um sistema de equações diferenciais parciais (EDP), as técnicas baseadas na deformação de um *grid* de pontos e as técnicas de geração de malha elemento a elemento a partir das informações do contorno. As principais idéias de cada técnica serão apresentadas nas seções seguintes a fim de fornecer uma idéia geral das metodologias existentes.

2.2.1 Geração de Malha por Transporte-Mapeamento

Este método baseia-se no mapeamento de uma malha de referência correspondente a uma geometria elementar, por exemplo, um triângulo ou um quadrilátero em um domínio real. Este tipo de método também é conhecido como um método algébrico. A malha de referência, obtida trivialmente, é mapeada para o domínio real com o auxílio de uma função de transporte. A robustez e a generalidade de aplicação deste método está relacionada com a escolha da função de transporte-mapeamento.

Princípio Geral do Método

Primeiramente é considerado um domínio $\widehat{\Omega}$ simples na forma de um triângulo ou quadrilátero, por exemplo. Em seguida, é definida uma função de transporte que respeite o contorno poligonal do domínio real Ω, conforme a figura (2.3). Os pontos do contorno do polígono são transportados para o quadrilátero unitário de tal modo que as distâncias entre eles sejam preservadas e a distribuição mantida. Uma abordagem mais aprofundada pode ser encontrada em [11, 29]. No caso do quadrilátero unitário, se $|v_i, v_{i+1}|$ é um dos lados do domínio real Ω e a_i^k são n pontos intermediários localizados neste lado, então, os $\hat{a}_i^k$ pontos intermediários localizados no lado correspondente do quadrilátero unitário são criados por

$$\hat{a}_i^1 = \hat{v}_i \qquad \hat{a}_i^k = \frac{\sum_{j=1}^{k-1} d_{j,j+1}}{\sum_{j=1}^{n-1} d_{j,j+1}} \qquad \hat{a}_i^n = \hat{v}_{i+1} \tag{2.1}$$

sendo que $d_{j,j+1}$ é a distância entre os pontos a_i^j e a_i^{j+1}. O índice i define a aresta que está sendo analisada e o índice k representa o número do ponto na aresta que vai determinar o intervalo em relação à uma extremidade.

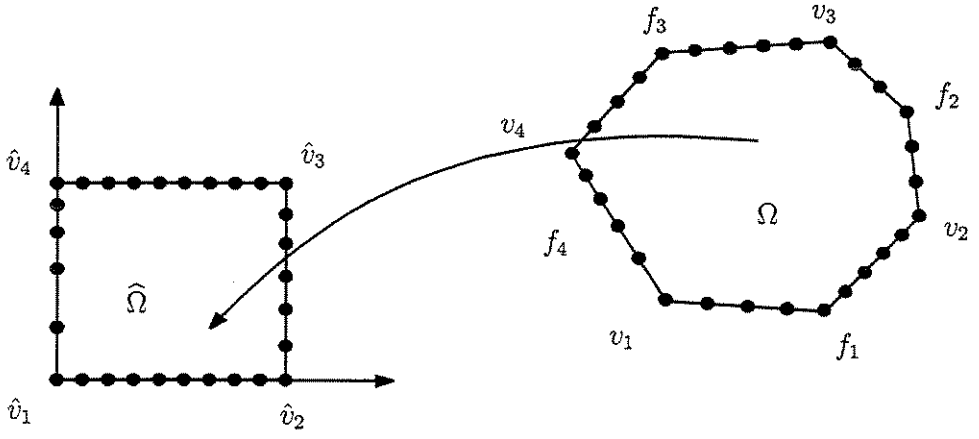


Figura 2.3: Transporte-mapeamento usando um quadrilátero unitário.

Depois que os pontos são gerados no domínio $\widehat{\Omega}$ do quadrilátero unitário, a malha é obtida de forma trivial unindo-se os pontos dos lados opostos. No caso de um triângulo no domínio $\widehat{\Omega}$, a construção da malha pode ser feita a partir de seu centróide, por exemplo. O transporte da malha de referência gerada no domínio de referência $\widehat{\Omega}$ para o domínio real, Ω, é feito usando-se uma função de mapeamento $\mathcal{F}_t$.

$$\boxed{\text{Malha de Referência}} + \boxed{\mathcal{F}_t} \rightarrow \boxed{\text{Malha Real}}$$

No caso de um quadrilátero unitário, a função de mapeamento $\mathcal{F}_t$, para transportar um ponto $\hat{p}$ no domínio $\widehat{\Omega}$ com coordenadas ξ e η para o domínio real Ω é dada por

$$\mathcal{F}_t(\xi,\eta) = (1-\eta)f_1(\xi) + \xi f_2(\eta) + \eta f_3(\xi) + (1-\xi)f_4(\eta)$$
$$- ((1-\xi)(1-\eta)v_1 + \xi(1-\eta)v_2 + \xi\eta v_3 + (1-\xi)\eta v_4) \quad (2.2)$$

sendo que os f_i's são parametrizações dos lados do domínio Ω e os v_i's são os vértices correspondentes. Os lados são aproximados pelo pontos a_i^k. Para o caso do triângulo, a seguinte função de transformação é usada:

$$\mathcal{F}_t(\xi,\eta) = \frac{1-\xi-\eta}{1-\xi}f_1(\xi) + \frac{\xi}{1-\eta}f_2(\eta) + \frac{\eta}{\xi+\eta}f_3(1-\xi-\eta)$$
$$- (\frac{\eta}{\xi+\eta}(1-\xi-\eta)v_1 + \frac{1-\xi-\eta}{1-\xi}\xi v_2 + \frac{\xi}{1-\eta}\eta v_3). \quad (2.3)$$

2.2.2 Geração de Malha por Solução de um Sistema de EDP

A técnica de geração de malha pela solução de um sistema de equações diferenciais parciais (EDP) possui uma abordagem parecida com a técnica de transporte-mapeamento apresentada na seção anterior, com relação aos tipos de domínios envolvidos. Por outro lado, a função usada para gerar a malha não é definida *a priori* como na técnica de transporte-mapeamento. Essa função é obtida pela solução de um sistema de equações diferenciais parciais. Nos trabalhos de Thompson [54, 55] é possível encontrar um estudo completo deste tipo de método e suas diferentes aplicações.

Princípio Geral do Método

Assim como no método de transporte-mapeamento, define-se um domínio de referência $\widehat{\Omega}$ do tipo quadrilateral unitário. Neste domínio de referência é gerada uma malha estruturada cuja conectividade dos nós é do tipo (i, j), como mostrado na figura (2.4). Uma malha estruturada é aquela na qual cada vértice, exceto nas bordas da malha, tem uma vizinhança

isomórfica. Em duas dimensões uma malha estruturada corresponde, na maior parte dos casos, a uma grade quadrada deformada por alguma transformação de coordenadas. Uma malha não estruturada é aquela na qual os seus vértices podem apresentar vizinhos locais arbitrariamente variados. Em geral, malhas não estruturadas correspondem a triangulações com vizinhança local não isomórfica. Define-se, então, as variáveis de espaço x, y e z, conectadas ao domínio físico Ω e as variáveis ξ, η e ζ, relacionadas ao domínio de referência $\widehat{\Omega}$.

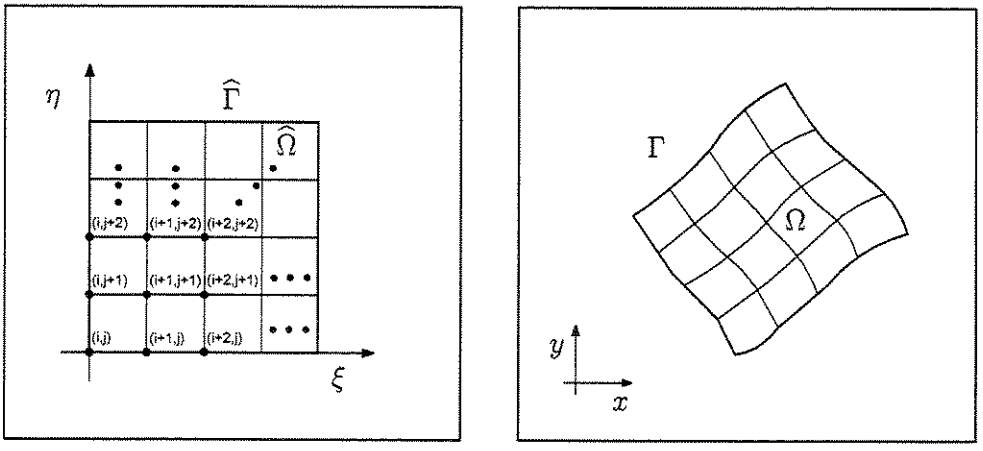


Figura 2.4: Os dois sistemas de variáveis envolvidos no método.

Neste método de geração de malha é necessário considerar um sistema de equações diferenciais parciais $\mathcal{S}$ e um conjunto de condições de contorno $\mathcal{CC}$. Para cada par formado por um sistema de equações diferenciais parciais $\mathcal{S}$ e um conjunto de condições de contorno $\mathcal{CC}$ deve existir pelo menos uma solução computacional possível. A dificuldade está em definir o sistema $\mathcal{S}$, chamado de sistema gerador. A escolha deve ser feita de forma a assegurar a geração de uma malha aceitável e satisfatória para uma análise numérica. Neste método é possível usar uma abordagem adaptativa de geração de malha em função dos valores nodais calculados. Como a função de transformação mapeia o domínio real Ω no domínio de referência $\widehat{\Omega}$ e, vice-versa, pode-se associar uma possível variação dos valores das variáveis físicas do domínio real com uma variação na forma da malha do domínio de referência. Isso traz grandes benefícios do ponto de vista de convergência de resultados e adaptação da malha ao problema físico estudado.

2.2.3 Geração de Malha por Deformação de um *Grid* de Pontos

Este método constrói uma malha em um domínio Ω essencialmente a partir da geração de um *grid* de pontos e dos dados do contorno. Vários tipos de *grid* podem ser gerados de modo a preencher todo o domínio do problema. O ideal é gerar um tipo de *grid* que melhor se adapte às características físicas do problema em consideração [7]. Depois de estabelecida a conectividade dos nós da malha, aplica-se uma função de deformação $\mathcal{F}_\delta$ para deslocar os pontos do *grid* a fim de proporcionar uma melhor adaptação dos elementos da malha ao tipo de geometria e problema físico em questão. Uma abordagem mais aprofundada sobre esta técnica pode ser encontrada em [10].

Princípio Geral do Método

O princípio do método é apresentado nesta seção usando-se um *grid* regular de pontos. Para o domínio Ω, apresentado na figura (2.5), é traçado um *grid* regular de pontos de forma a preencher toda uma área retangular que contém Ω. O espaçamento do *grid*, geralmente, é definido em função do menor segmento do contorno ou, ainda, é deixado como parâmetro de entrada pelo usuário para definição do grau de refinamento da malha. O processo se realiza nas seguintes etapas: primeiramente são removidos os quadriláteros que não interceptam o contorno Γ e não pertencem ao domínio Ω, figura (2.6). Depois, os pontos situados muito próximos ao contorno são removidos para se evitar a formação de elementos com uma forma ruim. Os pontos exteriores a geometria também são removidos. Aplica-se, então, um método, que pode ser o método de triangulação restrita de Delaunay, para estabelecer a conectividade entre os pontos do contorno da geometria e os pontos no interior que estão livres, figura (2.7). É possível ainda, aplicar-se uma função de deformação $\mathcal{F}_\delta$ para deslocar as coordenadas dos pontos da malha com a finalidade de melhor adaptar os elementos às características físicas da geometria e do problema estudado, (2.8).

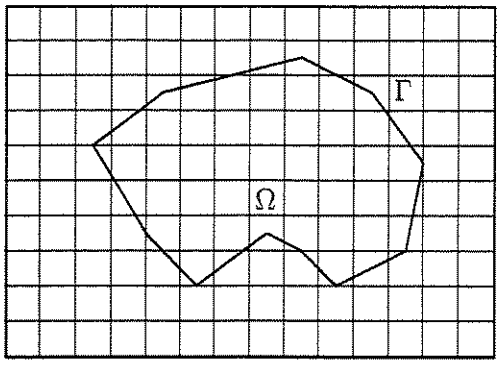


Figura 2.5: Aplicação de um *grid* regular de pontos para geração da malha.

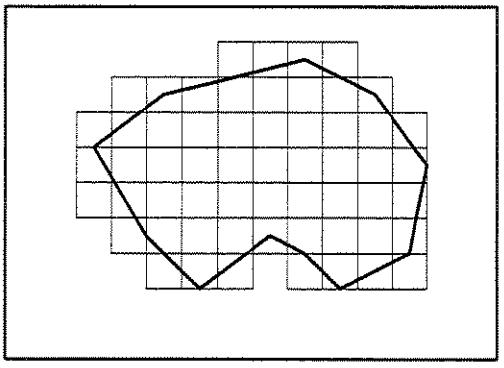


Figura 2.6: Eliminação dos quadriláteros não necessários a geração da malha.

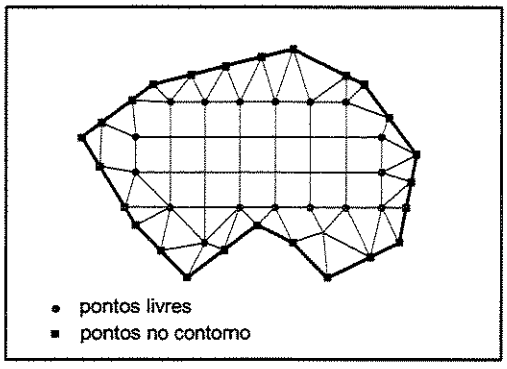


Figura 2.7: Malha sem a aplicação da função de deformação $\mathcal{F}_\delta$.

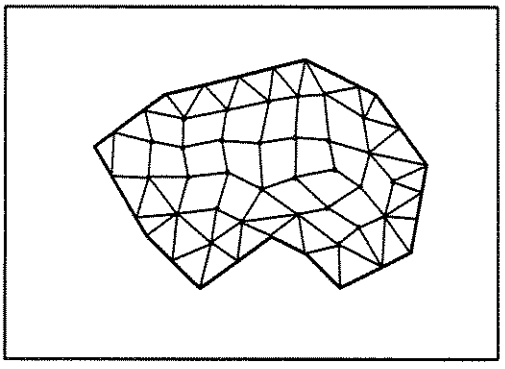


Figura 2.8: Malha final com a aplicação da função de deformação $\mathcal{F}_\delta$.

2.2.4 Geração de Malha por Avanço de Fronte

A geração de malha por avanço de fronte é bem adequada para o caso de geometrias arbitrárias e foi estudado por George em [25, 27] e Mavriplis em [37]. Este tipo de método constrói a malha de domínio a partir das informações do contorno. O processo é iterativo: um fronte é inicializado por um conjunto de segmentos que definem o contorno da geometria e, então, é feita uma análise para se determinar a zona de partida, a partir da qual um ou vários elementos serão criados. O fronte então é atualizado e o processo de geração de elementos continua indefinidamente até que o fronte fique vazio. O fluxograma do algoritmo é apresentado pela figura (2.9).

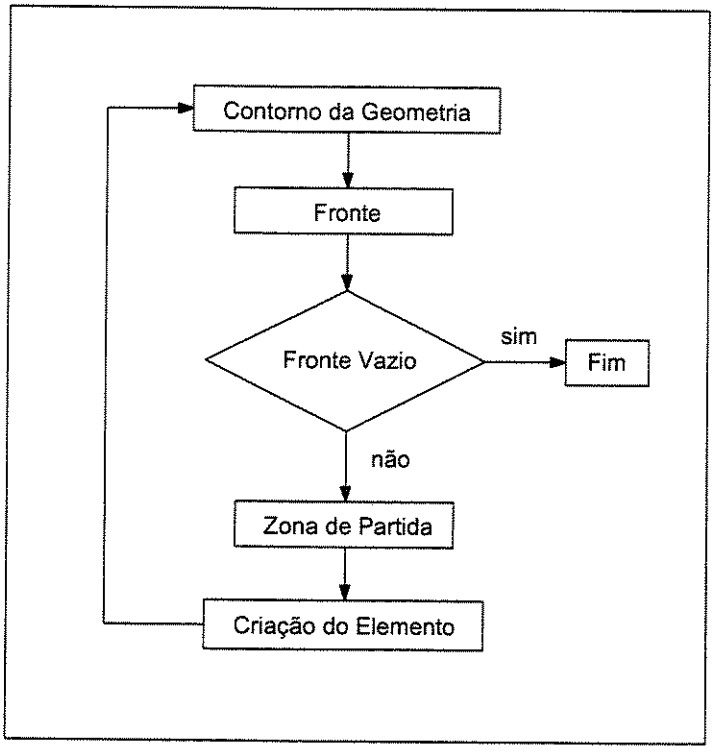


Figura 2.9: Fluxograma simplificado de execução do algoritmo de avanço de fronte.

Princípio Geral do Método

Este método de geração de malha proporciona uma solução automática para o problema de geração de malha usando elementos triangulares em geometrias arbitrárias. Para um fronte $\mathcal{F}$ constituído pelos segmentos do contorno de uma determinada geometria e, sendo α o ângulo formado por dois segmentos consecutivos pertencentes ao fronte, três situações de especial interesse podem ser identificadas:

- $\alpha < \frac{\pi}{2}$: os dois pontos livres, p_2 e p_4, dos segmentos que formam o ângulo α são unidos para criar um triângulo, figura (2.10);
- $\frac{\pi}{2} \leqslant \alpha \leqslant \frac{2\pi}{3}$: a partir dos dois segmentos que formam o ângulo α, um ponto interno p e dois triângulos e_1 e e_2 são gerados, figura (2.11);
- $\frac{2\pi}{3} < \alpha$: um ponto interno p e dois segmentos são gerados para formar um triângulo, figura (2.12).

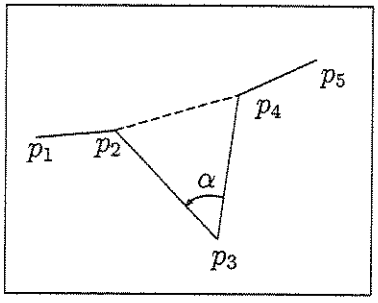


Figura 2.10: Situação em que $\alpha < \frac{\pi}{2}$.

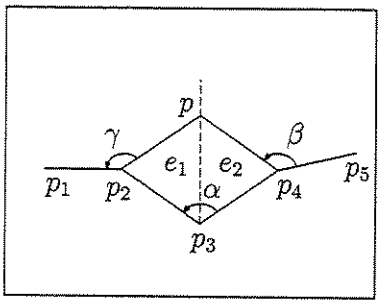


Figura 2.11: Situação em que $\frac{\pi}{2} \leqslant \alpha \leqslant \frac{2\pi}{3}$.

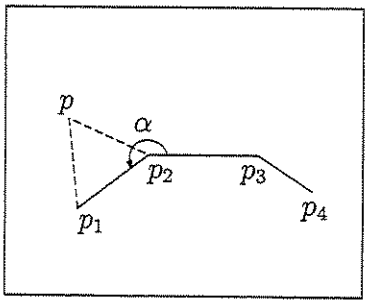


Figura 2.12: Situação em que $\frac{2\pi}{3} < \alpha$.

A posição dos pontos internos criados é definida de tal forma que possibilite a geração dos triângulos com a forma mais próxima possível a de um triângulo equilátero. No caso em que $\frac{\pi}{2} \leqslant \alpha \leqslant \frac{2\pi}{3}$, o novo ponto p é gerado no segmento bissetor de α. A localização deste ponto é dada por

$$d_{pp_3} = \frac{1}{6}(2d_{p_2p_3} + 2d_{p_3p_4} + d_{p_1p_2} + d_{p_4p_5}). \tag{2.4}$$

Isso é feito admitindo-se que os ângulos $\frac{\pi}{5} \leq \beta \leq \left(\pi - \frac{\pi}{5}\right)$ e $\frac{\pi}{5} \leq \gamma \leq \left(\pi - \frac{\pi}{5}\right)$, figura (2.11). O ângulo de $\frac{\pi}{5}$ radianos é definido empiricamente, [26]. Caso β e γ não estejam nessa faixa de valores, é adotado o procedimento para geração de elemento no caso em que $\alpha < \frac{\pi}{2}$. Na situação ilustrada pela figura (2.12), o menor segmento do fronte é usado para gerar os triângulos da forma mais regular possível.

Neste processo, para cada ponto criado é necessário verificar se ele se encontra dentro do domínio e se ele não se encontra no interior de algum elemento gerado. Esta verificação é crucial para este tipo de método e baseia-se em algoritmos bem estruturados que possibilitam o conhecimento rápido dos entes geométricos em uma determinada zona de interesse.

Uma ilustração passo a passo do método de avanço de fronte pode ser vista pelas figuras (2.13), (2.14), (2.15) e (2.16).

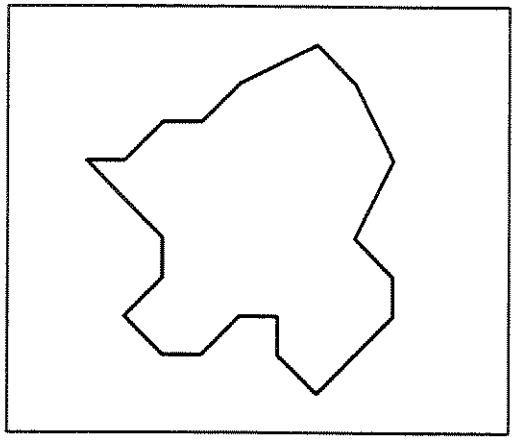


Figura 2.13: Fronte inicial formado pelo contorno da geometria.

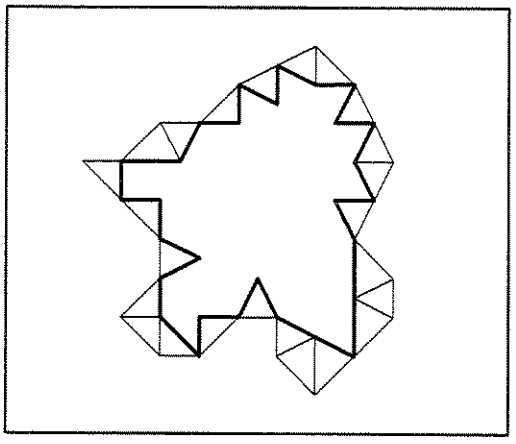


Figura 2.14: Novos segmentos que constituem o fronte após primeira iteração.

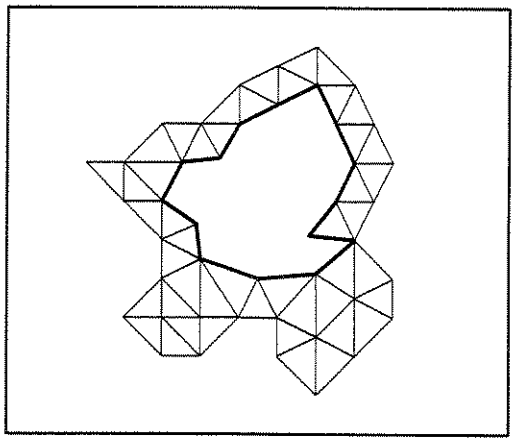


Figura 2.15: Novos segmentos que constituem o fronte após segunda iteração.

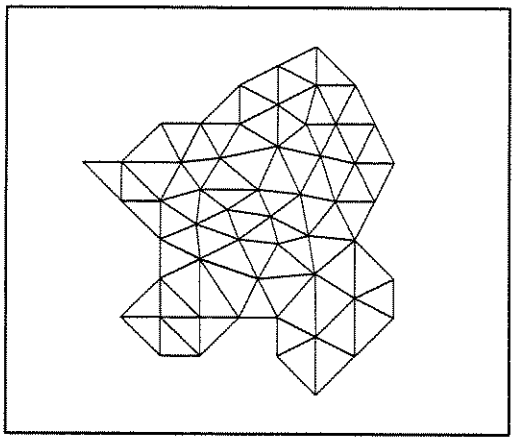


Figura 2.16: Geometria com a malha final gerada após todas as iterações.

2.2.5 Geração de Malha por Voronoï-Delaunay

A geração de malha usando o diagrama de Voronoï e a triangulação de Delaunay é uma forma bastante apropriada para se gerar malhas com elementos triangulares-lineares, que serão destinadas a análises numéricas. Isto deve-se ao fato do método da triangulação de Delaunay conferir a propriedade de maximizar os menores ângulos internos de cada triângulo da malha [44, 47], como poderá ser visto nas seções seguintes.

Diagrama de Voronoï

Um diagrama de Voronoï contém informações referentes a proximidade entre os pontos de um dado conjunto de pontos P. O conceito foi inicialmente desenvolvido por Dirichlet em 1850 e publicado por Voronoï em 1908, [61].

Definições e Propriedades Básicas

Dado um conjunto de pontos $P = \{p_1, p_2, ..., p_n\}$ no plano Euclidiano $\Re^2$, pode-se particionar esse plano em regiões que delimitam todos os pontos desse plano que estão situados mais próximos a um determinado ponto p_i do conjunto de pontos P. Cada ponto p_i forma uma dessas regiões pela expressão

$$V(p_i) = \{p \in \Re^2 : |p_i - p| \leq |p_j - p|, \forall j \neq i\} \tag{2.5}$$

sendo que $V(p_i)$ é uma região de Voronoï para o ponto p_i. Definindo-se todas as regiões $V(p_i)$ do conjunto de pontos P, obtém-se o diagrama de Voronoï $\mathcal{V}(P)$.

O diagrama de Voronoï $\mathcal{V}(P)$ pode ser traçado considerando-se os bissetores B_{ij} perpendiculares aos segmentos formados por p_i e p_j. Qualquer ponto p situado em B_{ij} é equidistante a p_i e p_j, como pode ser visto na figura (2.17).

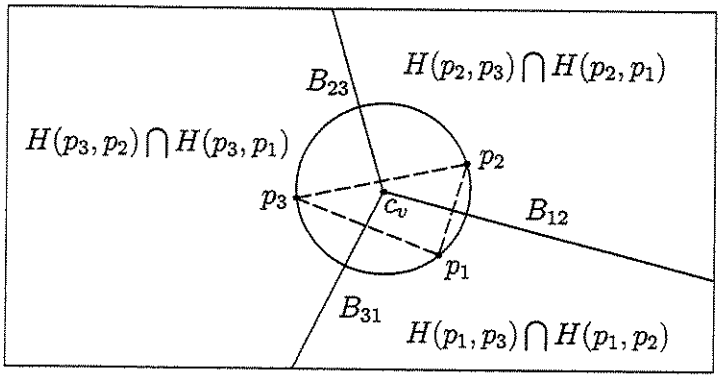


Figura 2.17: Diagrama de Voronoï para um conjunto de 3 pontos no plano.

Por exemplo, considerando-se os pontos p_1 e p_2 tem-se que B_{12} é o bissetor perpendicular ao segmento p_1p_2. Todos os pontos situados no bissetor B_{12} são equidistantes aos pontos p_1 e p_2. Da mesma forma para os segmentos p_2p_3 e p_3p_1 tem-se os bissetores B_{23} e B_{31}.

O ponto c_v originado pela interseção dos bissetores B_{12}, B_{23} e B_{31} é o centro do único círculo que passa sobre o triângulo formado pelos pontos p_1, p_2 e p_3, pois, pela construção, estes pontos estão a mesma distância de c_v.

Sendo $H(p_i, p_j)$ o semi-plano com contorno B_{ij} e contendo p_i, então, $H(p_i, p_j)$ pode ser visto como o conjunto dos pontos p do plano $\Re^2$ tais que $|p - p_i| \leq |p - p_j|$. Isso permite escrever a seguinte expressão

$$V(p_i) = \bigcap_{i \neq j} H(p_i, p_j) \tag{2.6}$$

ou seja, é feita a intersecção de todos os semi-planos $H(p_i, p_j)$, enquanto $i \neq j$. A figura (2.18) mostra um diagrama de Voronoï para um conjunto de oito pontos no plano. As linhas sólidas que delimitam as regiões de Voronoï são traçadas a partir dos bissetores perpendiculares a cada segmento formado por dois pontos do conjunto de oito pontos que são vizinhos.

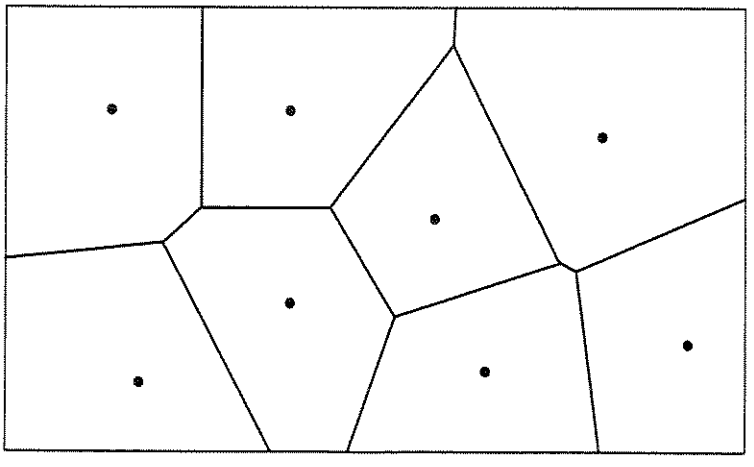


Figura 2.18: Diagrama de Voronoï para um conjunto de 8 pontos no plano.

Triangulação de Delaunay

Em 1934, Delaunay estabeleceu a dualidade entre o diagrama de Voronoï e um grafo planar constituído por segmentos lineares [16], construindo uma malha de elementos triangulares. O caso de mais de três pontos sobre uma circunferência de Delaunay deve ser evitado pois implica na formação de triângulos sobrepostos [44], como mostrado na figura (2.19).

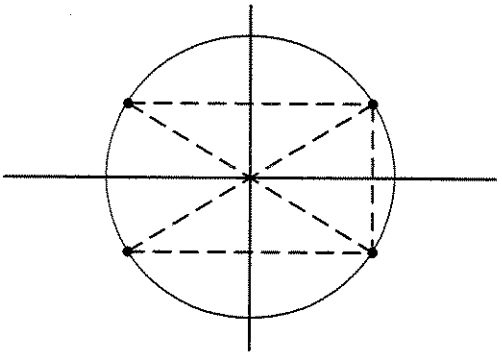


Figura 2.19: Situação de quatro pontos em uma circunferência de Delaunay.

A triangulação de Delaunay $\mathcal{D}\{P\}$ do diagrama de Voronoï apresentado pela figura (2.18) pode ser vista pela figura (2.20). Os triângulos são formados unindo-se cada ponto p_i aos outros pontos do conjunto de pontos P que estão situados nas regiões de Voronoï vizinhas a região de Voronoï $V(p_i)$.

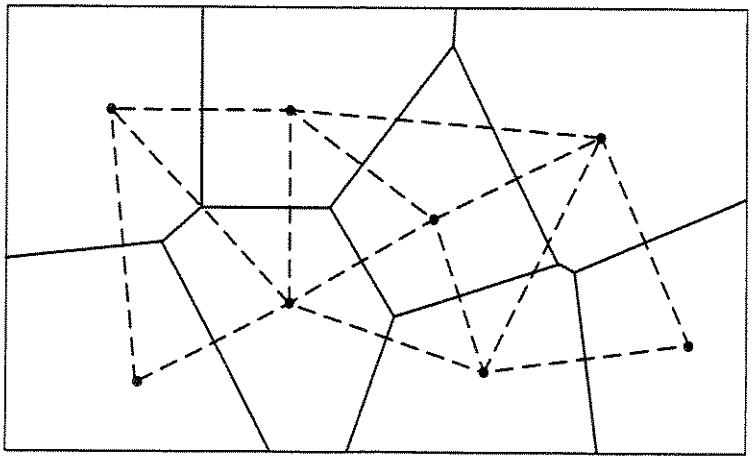


Figura 2.20: Triangulação de Delaunay do diagrama de Voronoï da figura (2.18).

Propriedades da Triangulação de Delaunay

A seguir são apresentadas algumas propriedades da triangulação de Delaunay sem as respectivas provas matemáticas formais. Caso o leitor deseje uma referência para consultar os teoremas e as provas matemáticas, vide [23, 47].

- $\mathcal{D}\{P\}$ é um grafo dual de $\mathcal{V}\{P\}$;
- $\mathcal{D}\{P\}$ é conforme se não existe quatro pontos cocirculares, figura (2.19);
- Cada triângulo $\mathcal{D}\{P\}$ corresponde em quantidade a um vértice $\mathcal{V}\{P\}$;
- Cada lado de $\mathcal{D}\{P\}$ corresponde em quantidade a um lado de $\mathcal{V}\{P\}$;
- Cada nó de $\mathcal{D}\{P\}$ corresponde a uma região de $\mathcal{V}\{P\}$.

Sendo c_v um vértice de $\mathcal{V}(P)$ na junção de $V(p_1)$, $V(p_2)$ e $V(p_3)$, então, c_v é o centro do círculo que circunscreve o triângulo formado por p_1, p_2 e p_3. No interior desse círculo não existe nenhum outro ponto do conjunto de pontos P, como garante a expressão (2.5) e como pode ser visto na figura (2.21).

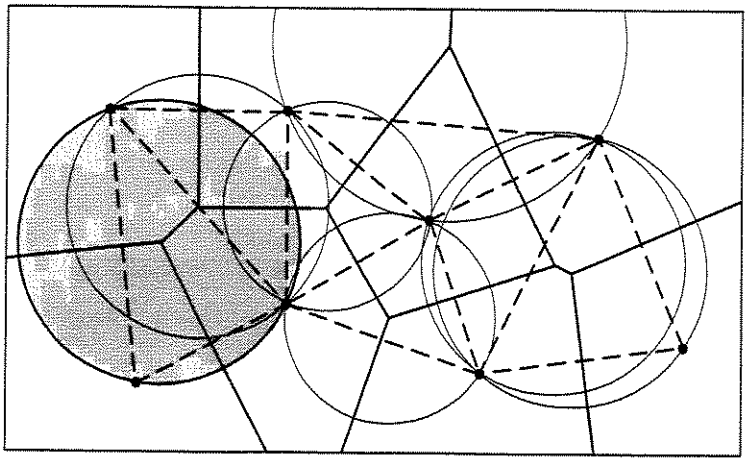


Figura 2.21: Propriedade dos circuncírculos vazios da triangulação de Delaunay $\mathcal{D}(P)$.

Dessa forma, os triângulos que são formados a partir da triangulação de Delaunay possuem os ângulos internos mínimos maximizados. Essa propriedade pode ser verificada

admitindo-se um triângulo formado pelos pontos p_1, p_2, e p_3 de um conjunto de pontos P, circunscrito a uma circunferência como mostrado na figura (2.22). Sendo p_1p_2 o menor segmento do triângulo, tem-se que o $\angle(p_1p_3p_2)$ é o menor ângulo interno do triângulo $p_1p_2p_3$. Admitindo-se um outro triângulo qualquer, formado pelos pontos p_1, p_2 e p_4, tal que $p_4 \in P$, tem-se que se $\angle(p_1p_4p_2) > \angle(p_1p_3p_2)$, então, p_4 é interior a circunferência que circunscreve o $\Delta p_1p_2p_3$, o que contradiz o fato do triângulo formado pelos pontos p_1, p_2, e p_3 ser um triângulo de Delaunay, figura (2.23).

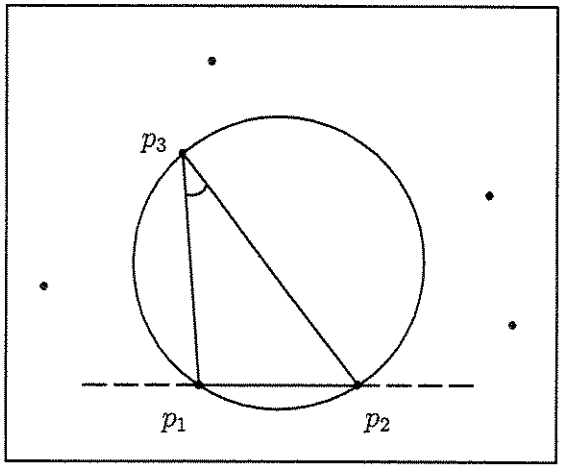


Figura 2.22: Triângulo formado por p_1, p_2 e p_3 circunscrito a uma circunferência.

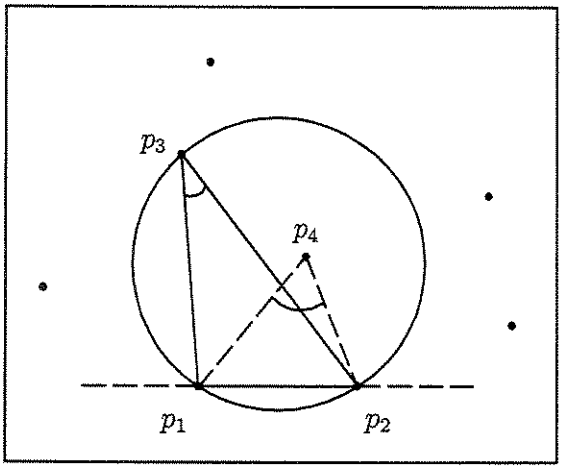


Figura 2.23: Propriedade de maximização dos ângulos internos mínimos de $\mathcal{D}(P)$.

2.3 Revisão Bibliográfica

A área de pesquisa em geração de malha é uma área extremamente vasta e muito rica para um pesquisador. Existem inúmeras publicações na área e, ainda hoje em dia, muitos esforços e muitos recursos têm sido dedicados para a melhoria dos geradores de malha existentes. Nesta seção é apresentado uma breve revisão bibliográfica, destacando-se as principais publicações que ajudaram no desenvolvimento deste trabalho.

No trabalho de Lee e Dyczij-Edlinger [34] são apresentadas técnicas de geração de malha usando o método da triangulação de Delaunay. A abordagem é feita visando a implementação

de geradores automáticos tanto para o caso bidimensional, como para o caso tridimensional. São apresentados alguns algoritmos para construção da triangulação de Delaunay como o algoritmo de Bowyer-Watson, o algoritmo de Green-Sibon e o algoritmo de triangulação de Delaunay por avanço de fronte.

No trabalho de Ruppert [50] é apresentado um algoritmo para triangulação de Delaunay destinado a gerar malhas em geometrias planas compostas por segmentos lineares. Este trabalho trata o problema de geração de malha no caso bidimensional e mostra importantes informações sobre como refinar as malhas usando-se como critério o ângulo interno mínimo dos triângulos. Um ótimo controle da forma e do tamanho dos triângulos é alcançado com esta técnica. Demonstrações e provas matemáticas são apresentadas para comprovar a eficácia dos resultados obtidos.

O trabalho de Shewchuk [52] é baseado no estudo de Ruppert. Neste artigo é apresentado um modelo intuitivo para análise do algoritmo de refinamento proposto em [50] e, pequenas melhorias em alguns aspectos, são mostradas. Em especial, é dado destaque ao controle dos ângulos internos dos elementos no caso de geometrias que já contém ângulos internos muito pequenos.

Outro trabalho importante para este estudo foi desenvolvido por Nicolas e Nicolas [41]. Este trabalho apresenta um gerador de malha tridimensional automático cujo método de execução é baseado principalmente na combinação de dois algoritmos diferentes: um para a geração de triângulos e outro para a geração de quadriláteros. A geração da malha é feita pela discretização de superfícies bidimensionais. Posteriormente é feita uma translação dessa superfície discretizada ao longo de uma linha ou, uma rotação, em relação a um eixo, para se obter malhas tridimensionais.

Nos trabalhos de Tsuboi *et. all.* [58, 59] o processo de geração de malhas é direcionado especificamente para o método dos elementos de contorno. Na publicação [58] é apresentada a

implementação de um *kernel* para se gerar malhas com elementos triangulares em linguagem *Lisp*. Um destaque especial é dado para a estrutura de dados utilizada pelo programa. Na publicação [59], é apresentado uma algoritmo de triangulação adaptativa com um critério de refinamento baseado em uma função Laplaciana de estimativa de erro.

Em [18], Du apresenta um algoritmo para triangulação de Delaunay para domínios planos arbitrários. Os triângulos de Delaunay são construídos a partir dos ângulos internos e não pela propriedade do circuncírculo vazio. O algoritmo baseia-se nas informações do contorno e move-se dinamicamente para o interior da geometria.

No trabalho de Mavriplis [37] é descrito um algoritmo para geração de malhas não-estruturadas em um domínio bidimensional arbitrário. Os nós da malha são gerados automaticamente pelo algoritmo de modo que uma suave variação no tamanho dos elementos é assegurada. A triangulação resultante constitui a triangulação de Delaunay desses pontos. O algoritmo proposto combina a eficiência e elegância matemática proporcionada pelo método de Delaunay com a robustez de geração de pontos associada às técnicas de avanço de fronte.

Secchi e Simoni apresentam em [51] um procedimento para discretização de domínios bidimensionais que destina-se especialmente à problemas que exigem uma continua atualização da malha. É usado o método de Delaunay para geração dos elementos.

Com relação a geração de *grid*, Qiang e Gunzburger em [48], apresentam um procedimento para geração de pontos baseado na construção de diagramas de Voronoï. O *grid* de pontos é gerado a fim de se obter soluções de equações diferenciais parciais usando-se o método dos elementos finitos. Em se tratando de optimização de *grid*, Castillo e Otto em [8] apresentam uma metodologia usando uma formulação variacional para geração de *grid* de pontos com ênfase no funcional de comprimento, que é o funcional mais importante na geração de *grid* por métodos variacionais.

Em [46], Phongthanapanich e Dechaumphai descrevem um procedimento de triangulação de Delaunay, usando programação orientada a objetos. Este procedimento é constituído por criação de nós, geração da malha, suavização da malha e remalhamento adaptativo. Bastian e Li em [2] implementaram um gerador de malha automático em C^{++}, também usando o paradigma da orientação a objetos, para geração de elementos quadrilaterais.

No trabalho de Lee *et. all.* [35] é proposto um algoritmo para geração de malha bidimensional com elementos quadrilaterais. O algoritmo é baseado na triangulação de Delaunay restrita e no algoritmo de Q-Morph. Em [20], El-Hamalawi propõe outro algoritmo para geração bi-dimensional de malhas que combina o método de triangulação de Delaunay com a técnica de avanço de fronte.

Considerando a geração de malha tridimensional para o método dos elementos finitos, George em [28] apresenta um algoritmo eficiente para geração de malha de forma adaptativa. Os pontos de campo são definidos usando a abordagem de avanço de fronte e a conectividade é estabelecida pelo método da triangulação tridimensional de Delaunay generalizada. Vários aspectos da triangulação tridimensional de Delaunay são apresentados no trabalho de Escobar e Montenegro [21].

Tianyu em sua tese de doutorado [56] desenvolveu a formulação para o problema de triangulação de superfície genérico minimizando os números de triângulos e vértices sob a restrição de distorção $\mathcal{L}_{-\infty}$. Esse problema se mostrou do tipo *NP-hard* de complexidade computacional. Malhas de alta qualidade foram obtidas tanto para superfícies explícitas $z = f(x, y)$, como para superfícies implícitas $f(x, y, z) = 0$.

Capítulo 3

Método dos Elementos de Contorno para Elasticidade

Antes de apresentar a formulação matemática do método dos elementos de contorno, é feita uma breve explicação sobre as notações usadas nas seções seguintes. O componente de coordenadas cartesianas de um ponto ou vetor no espaço tridimensional é denotado por x_i. O subscrito i varia de 1 a 3, uma vez que são considerados problemas tridimensionais. As notações dos tensores cartesianos e a convenção de soma de Einstein são adotadas. Uma leitura mais detalhada sobre os postulados básicos da elasticidade pode ser encontrada em [5, 9]. A derivada parcial com respeito as coordenadas cartesianas é feita em notação indicial, por exemplo, $u_{i,j}$ equivale a $\frac{\partial u_i}{\partial x_j}$. A função delta de Kronecker é denotada por δ_{ij}. Os materiais em consideração são admitidos como isotrópicos e homogêneos. Isso quer dizer que as propriedades elásticas do material são as mesmas em todas as direções e posições. Além disso, é postulado o caso de pequenos deslocamentos e pequenas deformações. Adotando esse postulado, o produto das derivadas dos deslocamentos são tão pequenos que podem ser desprezados quando comparados com os próprios deslocamentos.

3.1 Equações Básicas da Elasticidade

O estado de um corpo elástico tridimensional com um volume de domínio Ω e um contorno Γ sujeito a condições de contorno e forças de corpo, é caracterizado por quinze variáveis:

- as três componentes de deslocamento u_i;
- as seis componentes de deformação ε_{ij}, pois $(\varepsilon_{ij} = \varepsilon_{ji})$;
- as seis componentes de tensão σ_{ij}, pois $(\sigma_{ij} = \sigma_{ji})$.

As quinze variáveis podem ser representadas usando-se as equações abaixo:

As seis equações de deslocamento-deformação

$$\varepsilon_{ij} = \frac{1}{2}(u_{i,j} + u_{j,i}) \tag{3.1}$$

As três equações de equilíbrio

$$\sigma_{ij,j} + b_i = 0 \tag{3.2}$$

As seis equações constitutivas

$$\sigma_{ij} = \frac{2\nu\mu}{1-2\nu}\delta_{ij}\varepsilon_{mm} + 2\mu\varepsilon_{ij} \tag{3.3}$$

sendo que b_i são as componentes da força de corpo, $\mu = E/2(1+\nu)$ é o módulo de cisalhamento, E é o módulo de elasticidade e ν é o coeficiente de Poisson.

Um problema de elasticidade pode ser solucionado a partir das quinze equações (3.1), (3.2) e (3.3) para quinze variáveis desconhecidas, se forem aplicadas condições de contorno adequadas. As condições de contorno consistem, geralmente, na imposição de deslocamentos e na aplicação de forças de superfície. Em um problema típico de valor de contorno, assume-se que o contorno Γ é dividido em duas partes Γ_1 e Γ_2, ver figura (3.1). Admite-se que os deslocamentos u_i são conhecidos em Γ_1 e as forças de superfície t_i são conhecidas em Γ_2. As forças de superfície podem ser calculadas pela fórmula de Cauchy apresentada abaixo

$$t_i = \sigma_{ij}n_j \tag{3.4}$$

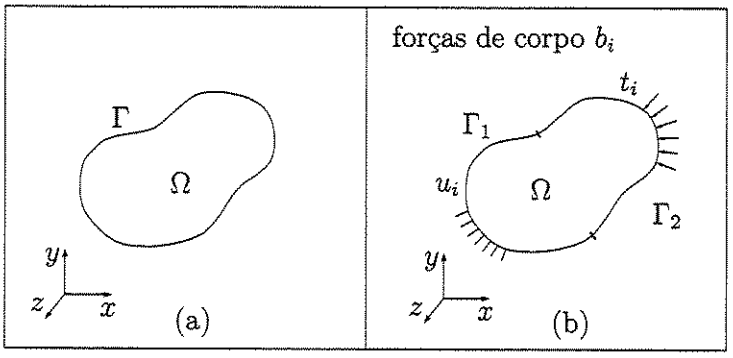


Figura 3.1: Modelagem de um problema típico de valor de contorno.

onde n_j são as componentes dos vetores normais unitários que apontam para fora da geometria no ponto considerado. Deste modo, as condições de contorno podem ser escritas como

$$\begin{array}{lll} u_i = \overline{u}_i & \text{em} & \Gamma_1 \\ t_i = \overline{t}_i & \text{em} & \Gamma_2 \end{array} \tag{3.5}$$

sendo que $\overline{u}_i$ e $\overline{t}_i$ são valores conhecidos de deslocamentos e forças de superfície, respectivamente. As equações de (3.1) a (3.5) constituem um problema típico de valor de contorno. Na prática algumas das quinze variáveis são eliminadas usando as equações (3.1), (3.2) e (3.3), antes que o processo para a obtenção da solução seja realizado. Por exemplo, substituindo a equação (3.3) na (3.2) e usando a equação (3.1), leva a

$$\mu u_{i,jj} + \frac{\mu}{1-2\nu} u_{j,ji} + b_i = 0 \tag{3.6}$$

onde somente as três componentes de deslocamento u_i são desconhecidas. A equação (3.6) é a equação de Navier e a partir dela é derivada a forma da equação integral da elasticidade.

3.2 Formulação do Método dos Elementos de Contorno

A formulação do método dos elementos de contorno é baseada em equações integrais singulares de contorno. A formulação foi obtida com sucesso para a elasticidade por Muskhelishvili [40] e Kupradze [33], baseando-se nos trabalhos de Betti [4] e Somigliana [53]. A

formulação analítica envolve a transformação da equação diferencial que é aplicável a todo o domínio do problema em uma equação integral no contorno. A equação integral de contorno pode ser derivada por várias maneiras, embora, geralmente, ela é obtida pelo uso de soluções fundamentais do problema em conjunto com um teorema de reciprocidade. O problema de elasticidade apresentado na forma da equação diferencial parcial (3.6), com a condições de contorno dadas pela equação (3.5), pode ser transformado em equações integrais no contorno usando-se as soluções fundamentais de Kelvin e o teorema da reciprocidade de Betti. Fazendo uma discretização isoparamétrica da equação integral de contorno para os deslocamentos, obtêm-se as equações características do método dos elementos de contorno, como pode ser visto em [38, 57].

3.2.1 As Soluções Fundamentais de Kelvin

Considera-se um domínio elástico infinito Ω com as forças de corpo dadas por

$$b_j^i = \Delta(X' - X)\delta_{ij} \tag{3.7}$$

sendo que X' representa o ponto fonte e X representa o ponto campo, que pode ser qualquer ponto exceto X'. A função $\Delta(X' - X)$ é a função Delta de Dirac que possui a seguinte propriedade

$$\int_\Omega f(X)\Delta(X' - X)d\Omega = f(X') \tag{3.8}$$

para qualquer função f contínua no domínio Ω. A equação de Navier (3.6), para deslocamentos desconhecidos u_j^i pode ser escrita por

$$\mu u_{j,kk}^i + \frac{\mu}{1-2\nu} u_{k,kj}^i + \Delta(X' - X)\delta_{ij} = 0. \tag{3.9}$$

Usando $U_{ij}(X', X)$ para representar u_j^i, a j-ésima componente da i-ésima solução da equação (3.9), as expressões de $U_{ij}(X', X)$ são dadas por

$$U_{ij}(X',X) = \frac{1}{16\pi(1-\nu)\mu R}\{(3-4\nu)\delta_{ij} + R_{,i}R_{,j}\} \tag{3.10}$$

sendo que $R = \sqrt{R_i R_i}$, $R_i = X_i - X'_i$ e $R_{,i} = \frac{\partial R}{\partial X_i} = \frac{R_i}{R}$.

As forças de superfície correspondentes em X, na direção do vetor normal $n = (n_1, n_2, n_3)$, podem ser obtidas pelas equações (3.10), (3.1), (3.3) e (3.4) na forma da equação seguinte

$$T_{ij}(X',X) = \frac{-1}{8\pi(1-\nu)R^2}\left\{\frac{\partial R}{\partial n}[(1-2\nu)\delta_{ij} + 3R_{,i}R_{,j}] - (1-2\nu)(n_j R_{,i} - n_i R_{,j})\right\} \tag{3.11}$$

onde $\frac{\partial R}{\partial n} = R_{,i}n_i$. As soluções (3.10) e (3.11) são as soluções fundamentais para problemas em elasticidade. O significado físico dessas soluções fundamentais é que, $U_{ij}(X',X)$ e $T_{ij}(X',X)$, representam os deslocamentos e as forças de superfície na direção j em um ponto campo X, devido a aplicação de uma carga unitária concentrada na direção i em um ponto fonte X'.

3.2.2 Teorema da Reciprocidade de Betti

Admitindo-se uma função vetorial contínua u_i^* que representa o deslocamento de um estado elasto-estático definido sobre um domínio Ω como sendo uma função pseudo-residual da equação de equilíbrio (3.2), tem-se

$$\int_\Omega \sigma_{ij,j}u_i^* d\Omega + \int_\Omega b_i u_i^* d\Omega = 0 \tag{3.12}$$

considerando somente o primeiro termo da equação, pela regra de derivação do produto de duas funções, tem-se

$$(\sigma_{ij}u_i^*)_{,j} = \sigma_{ij,j}u_i^* + \sigma_{ij}u_{i,j}^* \tag{3.13}$$

$$\int_\Omega \sigma_{ij}u_{i,j}^* d\Omega = \int_\Omega (\sigma_{ij}u_i^*)_{,j} d\Omega - \int_\Omega \sigma_{ij,j}u_i^* d\Omega \tag{3.14}$$

usando o teorema da divergência, $\int_\Omega \frac{\partial f_i}{\partial x_i} d\Omega = \int_\Gamma f_i n_i d\Gamma$, tem-se que

$$\int_\Omega (\sigma_{ij} u_i^*)_{,j} d\Omega = \int_\Gamma (\sigma_{ij} u_i^*) n_j d\Gamma = \int_\Gamma t_i u_i^* d\Gamma \tag{3.15}$$

substituindo em (3.14)

$$\int_\Omega \sigma_{ij} u_{i,j}^* d\Omega = \int_\Gamma t_i u_i^* d\Gamma - \int_\Omega \sigma_{ij,j} u_i^* d\Omega \tag{3.16}$$

pode-se escrever $u_{i,j}^*$ como a soma de um tensor simétrico e um anti-simétrico, da forma

$$u_{i,j}^* = \frac{1}{2}(u_{i,j}^* + u_{j,i}^*) + \frac{1}{2}(u_{i,j}^* - u_{j,i}^*) = \varepsilon_{ij}^* + w_{ij}^* \tag{3.17}$$

sendo que ε_{ij}^* e ω_{ij}^* representam, respectivamente, os tensores de deformação (simétrico) e de rotação (anti-simétrico) do estado " $*$ ". Multiplicando por σ_{ij} tem-se

$$\sigma_{ij} u_{i,j}^* = \sigma_{ij} \varepsilon_{ij}^* + \sigma_{ij} w_{ij}^* \ . \tag{3.18}$$

Como o produto de um tensor simétrico σ_{ij}^* por um tensor anti-simétrico ω_{ij}^* é nulo

$$\sigma_{ij} u_{i,j}^* = \sigma_{ij} \varepsilon_{ij}^* \tag{3.19}$$

substituindo em (3.16) tem-se

$$\int_\Omega \sigma_{ij} \varepsilon_{ij}^* d\Omega = \int_\Gamma t_i u_i^* d\Gamma - \int_\Omega \sigma_{ij,j} u_i^* d\Omega \tag{3.20}$$

o que conduz a

$$\int_\Omega \sigma_{ij}^* \varepsilon_{ij} d\Omega = \int_\Gamma t_i^* u_i d\Gamma - \int_\Omega \sigma_{ij,j}^* u_i d\Omega \ . \tag{3.21}$$

A partir das equações constitutivas apresentadas em (3.3) pode-se escrever

$$\sigma_{ij}\varepsilon^*_{ij} = (\frac{2\nu\mu}{1-2\nu}\delta_{ij}\varepsilon_{mm} + 2\mu\varepsilon_{ij})\varepsilon^*_{ij} = \frac{2\nu\mu}{1-2\nu}\varepsilon^*_{ll}\varepsilon_{mm} + 2\mu\varepsilon_{ij}\varepsilon^*_{ij} = \sigma^*_{ij}\varepsilon_{ij} \tag{3.22}$$

igualando-se as equações (3.20) e (3.21), como é permitido por (3.22), tem-se

$$\int_\Gamma t_i u^*_i d\Gamma - \int_\Omega \sigma_{ij,j} u^*_i d\Omega = \int_\Gamma t^*_i u_i d\Gamma - \int_\Omega \sigma^*_{ij,j} u_i d\Omega \ . \tag{3.23}$$

A equação (3.23) pode ser reorganizada como

$$\int_\Omega (\sigma_{ij,j} u^*_i - \sigma^*_{ij,j} u_i) d\Omega = \int_\Gamma (t_i u^*_i - t^*_i u_i) d\Gamma \tag{3.24}$$

que é análogo à fórmula de Green generalizada para o operador Laplace ∇ apresentada abaixo

$$\int_\Omega (f^* \nabla f - f \nabla f^*) d\Omega = \int_\Gamma (\frac{\partial f}{\partial n} f^* - \frac{\partial f^*}{\partial n} f) d\Gamma \ . \tag{3.25}$$

Levando em consideração a equação (3.2), a equação (3.23) pode ser escrita como

$$\int_\Omega b^*_i u_i d\Omega + \int_\Gamma t^*_i u_i d\Gamma = \int_\Gamma t_i u^*_i d\Gamma + \int_\Omega b_i u^*_i d\Omega \tag{3.26}$$

que é conhecida como o teorema da reciprocidade de Betti. A equação (3.26) indica que o trabalho total realizado pelas forças do estado real b e t, agindo sobre os deslocamentos produzidos pelo estado fundamental u^*_i, é igual ao trabalho total realizado pelas forças do estado fundamental, t^* e b^*, agindo sobre os deslocamentos produzidos pelo estado real, u_i.

3.2.3 Equações Integrais de Contorno para os Deslocamentos

As equações integrais de contorno para os deslocamentos podem ser derivadas a partir do teorema da reciprocidade de Betti e das soluções fundamentais de Kelvin. Se os deslocamentos, forças de superfície e forças de corpo são dados por u_j, t_j e b_j, para o estado real de

um problema de elasticidade com domínio Ω e contorno Γ, o estado fundamental é definido pelas soluções de Kelvin, U_{ij}, T_{ij} e $b_{ij} = \Delta(X' - X)\delta_{ij}$. O ponto fonte X' é admitido ser um ponto interno a Ω e a partir do teorema da reciprocidade de Betti e da propriedade da função Delta de Dirac, apresentada pela equação (3.8), a identidade de Somigliana para os deslocamentos pode ser escrita como

$$u_i(X') + \int_\Gamma T_{ij}(X', x)u_j(x)d\Gamma(x) = \int_\Gamma U_{ij}(X', x)t_j(x)d\Gamma(x) + \int_\Omega U_{ij}(X', X)b_j(X)d\Omega(X) \; . \tag{3.27}$$

As equações integrais de contorno relacionadas aos deslocamentos e forças de superfície agindo no contorno podem ser obtidas a partir da equação (3.27) considerando o processo de limite de um ponto interno indo para o contorno ($X' \to x'$). Um contorno auxiliar $\Gamma - \Gamma_\varepsilon + \Gamma_\varepsilon^*$ é adotado como mostrado pela figura (3.2), sendo que Γ_ε^* representa a superfície de uma esfera de raio ε.

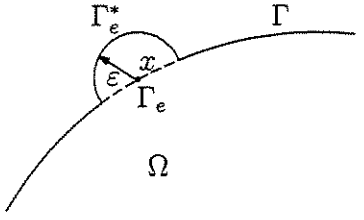


Figura 3.2: Contorno auxiliar usado para o processo de limite ($X' \to x'$).

Conforme $\varepsilon \to 0$, Γ_ε^* tende a Γ_ε e, portanto, o contorno auxiliar tende a Γ. O processo de limite de $X' \to x'$ para a equação (3.27) pode ser substituído fazendo $\varepsilon \to 0$

$$u_i(x') + \lim_{\varepsilon \to 0} \int_{\Gamma - \Gamma_\varepsilon + \Gamma_\varepsilon^*} T_{ij}(x', x)u_j(x)d\Gamma(x) = \lim_{\varepsilon \to 0} \int_{\Gamma - \Gamma_\varepsilon + \Gamma_\varepsilon^*} U_{ij}(x', x)t_j(x)d\Gamma(x) \tag{3.28}$$

sendo que a existência das forças de corpo é desprezada. A primeira integral da equação (3.28) contém um integrando fortemente singular de ordem $O(1/R^2)$ que é avaliada no sentido do valor principal de Cauchy. A segunda integral da equação (3.28) contém um integrando

singular de ordem $O(1/R)$, que é integrado como uma integral imprópria. Uma discussão detalha do processo de limite pode ser encontrada em [32]. Finalmente, as equações integrais de contorno para os deslocamentos podem ser escritas como

$$c_{ij}(x')u_j(x') + \int_\Gamma T_{ij}(x', x)u_j(x)d\Gamma(x) = \int_\Gamma U_{ij}(x', x)t_j(x)d\Gamma(x) \tag{3.29}$$

sendo que $c_{ij}(x')$ é uma função da variação do contorno da geometria no ponto x' situado no contorno. É válido ressaltar que até esse momento nenhuma aproximação matemática foi feita para a transformação da equação diferencial dada pela (3.6) que governa o problema, para a forma integral apresentada pela equação (3.29). A obtenção de uma solução analítica para os deslocamentos e forças de superfície a partir da equação (3.29) pode ser muito difícil de ser obtida e na maioria das vezes é impossível, especialmente em casos tridimensionais com geometrias complexas. Portanto, é necessário fazer uma adequada discretização do contorno para o cálculo da solução aproximada do problema em específicos pontos de interesse. A seção seguinte trata do processo de discretização com maiores detalhes.

3.2.4 Discretização das Equações Integrais de Contorno

Inicialmente são apresentadas, a título de ilustração, duas malhas bidimensionais para análises numéricas. A figura (3.3a) mostra uma malha de elementos de contorno bidimensional composta por elementos lineares. A figura (3.3b) mostra uma malha de elementos finitos bidimensional composta por elementos triangulares. É importante perceber a diferença entre as duas discretizações realizadas. A geometria do problema é a mesma, o que muda é o método de análise numérica empregado. Fica nítido, então, que em elementos de contorno, um número bem menor de elementos precisa ser gerado. Isso traz grandes benefícios como a redução do custo computacional durante a fase de pré-processamento.

Devido a equação (3.29) possuir somente termos dependentes do contorno o processo de geração de malha pode ser realizado de forma bem mais robusta, rápida e eficiente, uma vez que, em casos tridimensionais somente elementos de superfície precisam ser gerados. A malha

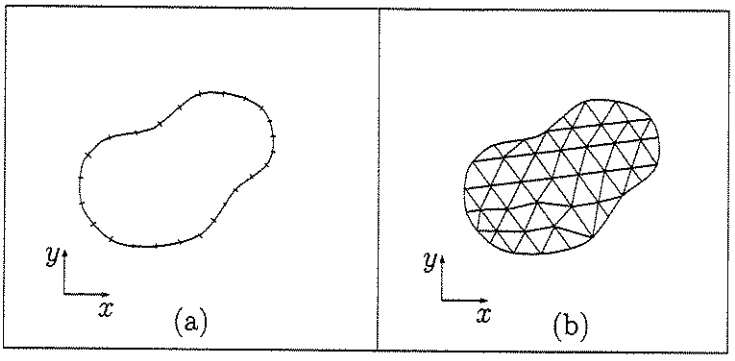


Figura 3.3: Exemplo de malhas 2D: (a) elementos de contorno e (b) elementos finitos.

tridimensional de elementos de contorno é como se fosse uma casca constituída por vários elementos que modelam um sólido geométrico. A implementação computacional fica extremamente simplificada e no caso específico deste trabalho, a discretização é feita usando-se elementos contínuos triangulares-lineares de três nós e com funções de forma isoparamétricas, como mostrado na figura (3.4).

Neste trabalho, o método dos elementos de contorno é apresentado para problemas da elasticidade, mas este método também é bastante útil em problemas de bioengenharia [43]; em problemas que necessitam de contínuo remalhamento, como propagação de trincas; em problemas de avanço de superfície, no estudo do espalhamento de poluição; em problemas cujos domínios são ilimitados; dentre outros.

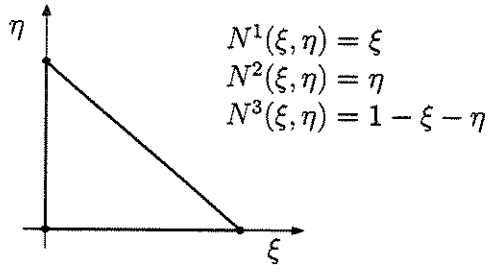


Figura 3.4: Elemento de contorno triangular-linear contínuo de três nós.

Usando as funções de forma N^1, N^2 e N^3, que estão implementadas no solver E-Con3D, o método dos elementos de contorno pode ser escrito na forma discreta como

$$c_{ij}(x^\kappa)u_j(x^\kappa) + \sum_{n=1}^{N}\sum_{m=1}^{M} u_j^{nm} \int_{-1}^{1}\int_{-1}^{1} T_{ij}[x^\kappa, x(\xi,\eta)]N^m(\xi,\eta)J^n(\xi,\eta)d\xi d\eta$$

$$= \sum_{n=1}^{N}\sum_{m=1}^{M} t_j^{nm} \int_{-1}^{1}\int_{-1}^{1} U_{ij}[x^\kappa, x(\xi,\eta)]N^m(\xi,\eta)J^n(\xi,\eta)d\xi d\eta \qquad (3.30)$$

sendo que M é o número de nós por elemento, N é o número total de elementos da malha, κ representa o número do nó da malha que está sendo avaliado como ponto fonte e J^n é o Jacobiano da transformação. O Jacobiano da transformação pode ser calculado por $J^n = |x_\xi \times x_\eta|$, sendo que os subscritos ξ e η denotam as derivadas com relação a ξ e η respectivamente.

A equação (3.30), apresentada na forma discreta, pode ser reescrita na forma matricial de um sistema de equações lineares como

$$[\mathbf{H}]\{\mathbf{u}\} = [\mathbf{G}]\{\mathbf{t}\} \qquad (3.31)$$

sendo que a matriz $[\mathbf{H}]$ contém as integrais envolvendo os termos T_{ij} e a matriz $[\mathbf{G}]$ contém os termos envolvendo U_{ij}. Os vetores $\{\mathbf{u}\}$ e $\{\mathbf{t}\}$ consistem, respectivamente, de todas as componentes dos deslocamentos e de todas as componentes das forças de superfície em cada nó da malha. Aplicando-se as condições de contorno o sistema pode ser rearranjado na forma

$$[\mathbf{A}]\{\mathbf{x}\} = \{\mathbf{f}\} \qquad (3.32)$$

sendo que $\{\mathbf{x}\}$ é o vetor das componentes das variáveis desconhecidas de u_i e t_i. A matriz quadrada $[\mathbf{A}]$ é obtida pelo processo de permutação das colunas das matrizes $[\mathbf{H}]$ e $[\mathbf{G}]$ a fim de se isolar as variáveis desconhecidas no vetor $\{\mathbf{x}\}$. Todos os valores nodais no contorno desconhecidos de u_i e t_i podem, então, ser calculados solucionando-se a equação (3.32). O vetor $\{\mathbf{f}\}$ é obtido usando as condições de contorno do problema. Caso necessário, os deslocamentos e as tensões internas ao domínio Ω podem ser calculados depois de ser obtida a solução no contorno Γ, usando-se a identidade de Somigliana, dada pela equação (3.27).

Capítulo 4

Implementação Computacional

Neste capítulo será apresentada a implementação computacional, que foi feita baseada nas técnicas apresentadas no capítulo 2, para o desenvolvimento do gerador de malhas proposto. O gerador de malhas proposto usa conceitos da técnica de transporte-mapeamento, da técnica de geração de um *grid* de pontos e, principalmente, do método de triangulação de Delaunay. O código do programa foi escrito em linguagem C^{++} usando o paradigma da orientação a objetos. O programa foi escrito usando o compilador Microsoft Visual C^{++}® e o modelador visual de classes Rational Rose® [49]. Foram usadas bibliotecas auxiliares como a biblioteca Newmat10, para cálculo matricial, e a biblioteca OpenGL®, para desenvolvimento de uma interface gráfica para exibição gráfica das geometrias e malhas.

4.1 Programação Orientada a Objetos

A linguagem C^{++} oferece pleno suporte a programação orientada a objetos, incluindo-se os conceitos de encapsulamento, herança e polimorfismo, [36]. A linguagem C^{++} suporta múltiplas entidades e bibliotecas usando a abstração de dados. Abstração de dados significa que é possível combinar diferentes tipos de estruturas de dados e operações em um novo tipo abstrato de dados, [19]. Um tipo abstrato de dados comporta-se como os tipos de dados que são inerentes a linguagem de programação, só que são criados pelo programador. A estratégia de implementação usando programação orientada a objeto consiste em identificar os aspectos do problema em questão, como os conceitos físicos envolvidos e as etapas que permitem a

obtenção da solução e, a partir daí, representar esses aspectos computacionalmente, usando a abstração de dados, na forma da criação de classes e objetos. Os objetos contêm todo o conhecimento da classe, na forma de atributos e métodos. Eles podem acessar os atributos e métodos da classe através de um controle de acesso. Esse controle de acesso é feito usando-se a palavra-chave *public*, que significa que todos os objetos da classe têm acesso aos atributos e métodos, a palavra-chave *private*, que significa que somente os próprios membros da classe têm acesso e, a palavra-chave *protected*, que significa que somente os membros da própria classe e de classes derivadas têm acesso.

A utilização desses conceitos de programação permite a manipulação dos dados do programa de forma mais fácil, eficiente e segura. Com essas vantagens o programador pode se concentrar mais na conceituação do problema, criando um código que melhor adapte o computador à resolução do problema. Outro ponto positivo é que fica bem mais fácil manter a complexidade da solução sob controle. Assim, a montagem do sistema final ou mudanças de foco durante o desenvolvimento do programa podem ser conduzidas de forma mais eficiente e bem menos custosa.

4.1.1 Obtendo um Modelo Orientado a Objetos

Um modelo orientado a objeto pode ser conseguido por um processo de desenvolvimento que possui três etapas:

- Análise orientada a objetos: é a fase onde um especialista descreve o problema em termos das necessidades e tarefas a serem resolvidas pelo programador;
- Projeto orientado a objetos: é a fase de definição das classes e de seus atributos e métodos que irão fazer a vinculação e a comunicação entre os objetos;
- Programação orientada a objetos: é a fase de implementação, na qual ocorre a codificação dos métodos, os testes e a integração do programa como um todo.

Uma vez definidas as classes, a programação orientada a objeto compõe o problema em um conjunto de objetos que se relacionam entre si. Esse relacionamento ocorre pelos métodos das classes da qual esses objetos foram instanciados. Assim sendo, os objetos são criados e comunicam-se entre si, recebendo e distribuindo tarefas, e após cada um cumprir o seu propósito, o problema estará resolvido.

Para auxiliar no desenvolvimento do programa foi usado o modelador visual Rational Rose®, que é um programa de engenharia de *software* que fornece uma representação gráfica e possibilita um entendimento mais claro do programa em desenvolvimento. No modelador de classes Rational Rose® são criadas e definidas todas as classes do programa, com seus respectivos atributos e métodos. As relações entre as classes, como relações de herança, posse e uso, também podem ser facilmente estabelecidas com o auxílio dessa ferramenta. Essas informações formam a estrutura básica do programa. Em seguida, com um recurso existente no programa Rational Rose®, a estrutura desenvolvida é codificada automaticamente na linguagem C^{++}, restando ao programador a tarefa de implementar os métodos.

4.2 Características do Gerador

O programa implementado neste trabalho tem capacidade de gerar malhas com elementos de superfície triangulares-lineares em geometrias tridimensionais arbitrárias, compostas por faces planas. Um exemplo de geometria tridimensional aceita pelo programa está apresentado na figura (4.1).

O gerador de malhas implementado não pode ser classificado única e exclusivamente por uma das técnicas de geração de malha apresentadas no capítulo 2. Na verdade, ele usa conceitos da técnica de transporte-mapeamento, da técnica de geração de um *grid* de pontos e, principalmente, do método de triangulação de Delaunay. O princípio de funcionamento consiste em deslocar cada face da geometria do espaço tridimensional, usando transformações geométricas, para o plano xy. Então, no plano xy gera-se um *grid* de pontos sobre a face e

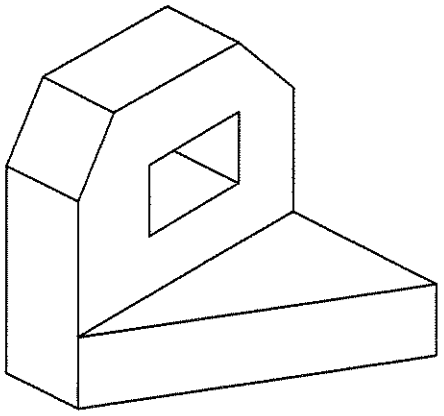


Figura 4.1: Exemplo de geometria tridimensional aceita pelo programa.

aplica-se o método da triangulação de Delaunay. Algumas verificações são necessárias para o caso das faces possuírem uma geometria muito irregular e com rasgos internos, a fim de garantir que a malha seja gerada de forma correta. Essas verificações visam evitar a geração de elementos fora do domínio da face e a formação de triângulos sobrepostos. Elas estão apresentadas em detalhes na seção 4.9. Depois de gerada a malha na face, novamente, um conjunto de transformações geométricas são aplicadas para mover a face de volta para sua posição original no espaço. A continuidade final da malha é assegurada promovendo-se a discretização das arestas da geometria, antes da geração do *grid* de pontos na face. O funcionamento do programa poderá ser melhor entendido analisando-se o seu fluxograma de execução que é apresentado na seção 4.4.

4.3 Estrutura de Dados Usada pelo Programa

Os formatos dos arquivos de entrada e saída usados pelo gerador de malha são apresentados pela figura (4.2). O arquivo de entrada contém as informações da geometria e o valor do parâmetro $MeshPar$ fornecido pelo usuário. Este arquivo é obtido usando-se um tradutor IGES, ver processo na figura (4.3). Primeiramente a geometria é desenhada em um programa CAD, neste trabalho é usado o SolidEdge® para elaboração dos desenhos e, em seguida, o desenho é salvo no formato IGES.

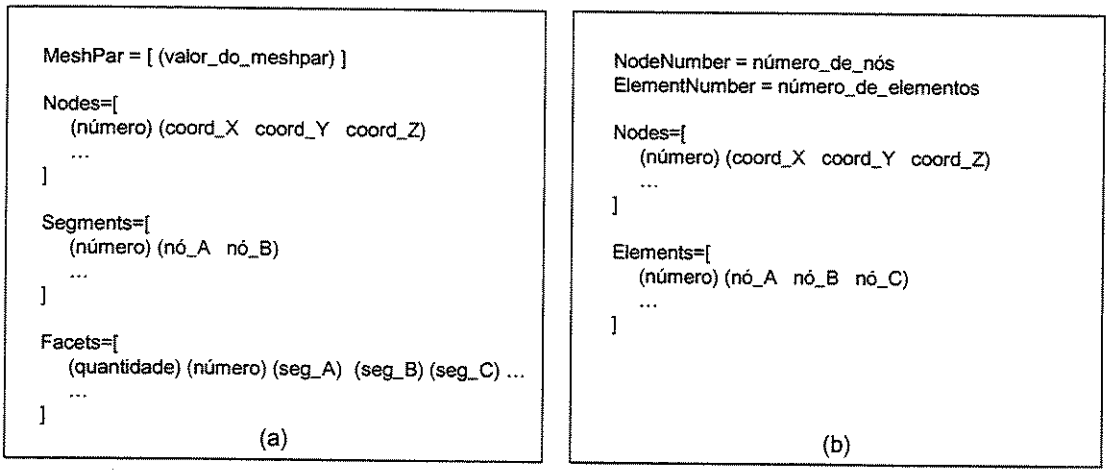


Figura 4.2: Formatos do arquivo de entrada (a) e do arquivo de saída (b).

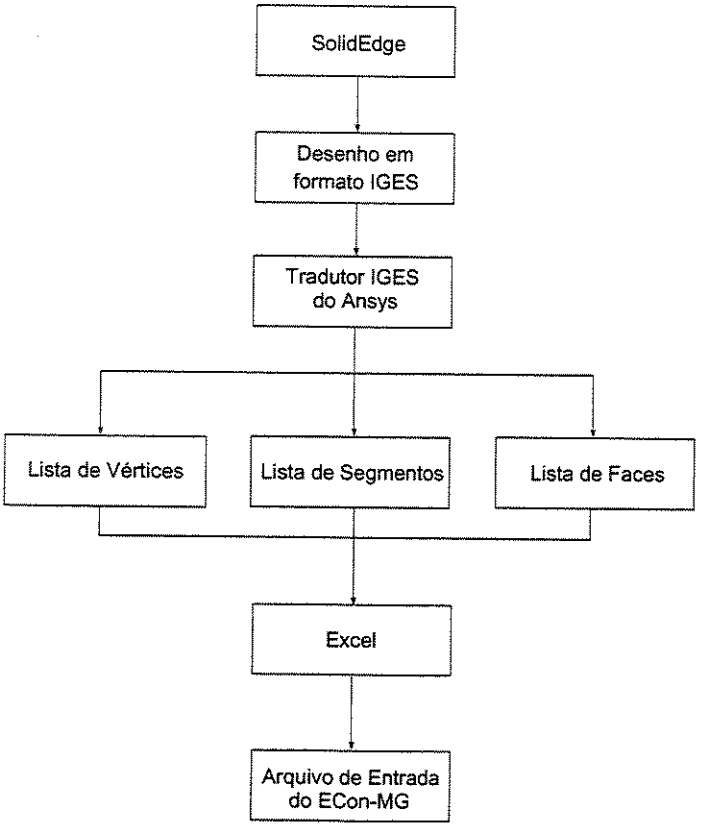


Figura 4.3: Processo de obtenção do arquivo de entrada de dados para o gerador.

O desenho é, então, aberto no Ansys® e as informações do desenho são exportadas de modo a gerar três lista: uma lista de vértices, uma lista de segmentos e uma lista de faces. É usado, então, o programa Excel® para gerar o arquivo de entrada de dados no formato apresentado pela figura (4.2a). Percebe-se aqui, a necessidade de se implementar um tradutor IGES para tornar o gerador de malhas independente de outros programas, com exceção dos programas de CAD, naturalmente. O valor do $MeshPar$ vai ser usado por uma função do programa para determinar o número de divisões a serem feitas nos segmentos do contorno da geometria. Ele é dado em unidade de comprimento. Por exemplo, se um segmento da geometria possui comprimento de 1 unidade e o valor do $MeshPar = 0,3$ o segmento será dividido em três partes iguais com $1/3$ de unidades de comprimento cada. Portanto, o número de divisões a ser feita no segmento é um valor inteiro N_div calculado por

$$N_div = int\left(\frac{comp_seg}{MeshPar}\right) \tag{4.1}$$

onde o $comp_seg$ é o comprimento do segmento analisado. Como o $MeshPar$ é um parâmetro de discretização, reduzindo-se seu valor implica em um aumento no número de elementos da malha. Dessa forma, cada segmento da geometria possui uma lista de nós que contém todos os nós desse segmento discretizado.

O arquivo de entrada contém uma listagem dos vértices da geometria. Cada vértice é representado por um número e suas respectivas coordenadas x, y e z, associado a palavra *Nodes*. Os segmentos são representados pelo seu respectivo número e pelos números dos vértices que o definem. As faces são descritas pela quantidade de segmentos que é constituída, por seu respectivo número e pelos números dos seus segmentos. É importante ressaltar que a ordem dos dois primeiros segmentos de cada face definem a orientação do vetor normal da face como mostrado pela figura (4.4).

No caso (4.4a), o segmento S_1 foi fornecido antes do segmento S_2, o que resultou em um vetor normal com a orientação (0 0 -1) $\otimes$. No caso (4.4b), o segmento S_1 foi fornecido depois do segmento S_2, o que resultou em um vetor normal com a orientação (0 0 1) $\odot$.

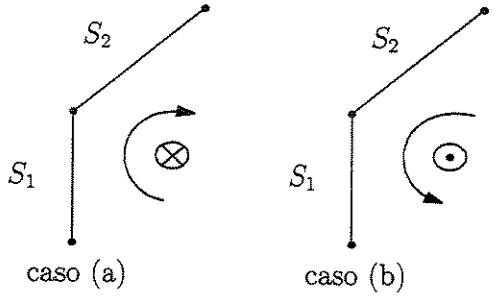


Figura 4.4: Orientação do vetor normal com relação a ordem dos segmentos S_1 e S_2.

É preciso, também, levar em consideração a situação apresentada na figura (4.5). Tanto para o caso (4.5a) como para o caso (4.5b), o segmento S_1 que origina o vetor $\mathbf{v_1}$ foi fornecido antes do segmento S_2 que origina o vetor $\mathbf{v_2}$.

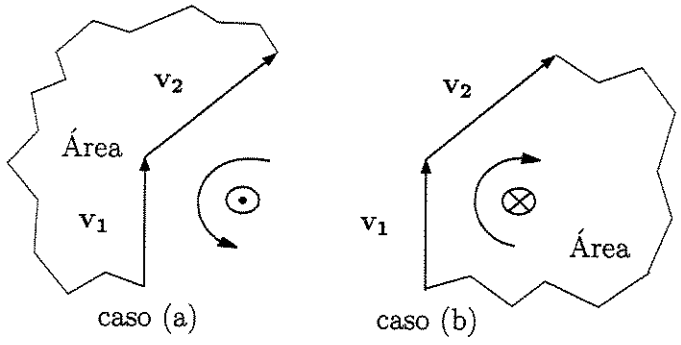


Figura 4.5: Orientação do vetor normal da face calculado a partir de $\mathbf{v_1}$ e $\mathbf{v_2}$.

No caso (4.5a) o vetor normal deve ter orientação (0 0 1) $\odot$ e no caso (4.5b) deve ter orientação (0 0 -1) $\otimes$. Isso pode ser verificado pelas posições das áreas das faces com relação ao vetor $\mathbf{v_1}$. Entretanto, quando o produto vetorial $\mathbf{v_1} \times \mathbf{v_2}$ é calculado para as duas situações o resultado obtido é (0 0 -1). Esse problema pode ser solucionado gerando-se um ponto auxiliar deslocado para direita ou para a esquerda do ponto médio do segmento S_1, que origina $\mathbf{v_1}$, por um incremento de $\frac{|\mathbf{v_1}|}{1000}$. Então verifica-se se esse ponto está ou não no interior da face pelo algoritmo apresentado na seção 4.8. Dessa forma é possível corrigir a orientação do vetor normal da face para o caso (4.5a).

O método que calcula o vetor normal de uma face deslocada para o plano xy está presente na classe $Mesh$ e seu nome é $give_ValueOfFacetNormalVector2D(List < Segment > theFacetSegmentList,\ double\&\ FaceNormal_K)$. Este método recebe como argumento uma lista de segmentos que define a face e passa por referência o valor do índice k do vetor normal, pela variável do tipo *double*, $FaceNormal_K$.

Em elementos de contorno é necessário que todos os vetores normais dos elementos da malha estejam apontando para fora da geometria. Desse modo, os vetores normais de todos os elementos de uma face da geometria serão alinhados segundo o vetor normal da face, que deverá estar apontando para fora da geometria. O arquivo de saída gerado pelo programa contém a quantidade de nós da malha, a quantidade de elementos, as informações de cada nó da malha e suas respectivas conectividades que definem os elementos, ver figura (4.2b). O arquivo de saída é usado pelo *solver* E-Con3D para a realização de análises numéricas.

O fluxograma do processo para realização de uma análise numérica é apresentado na figura (4.6). O arquivo com as informações da geometria do problema é usado pelo gerador de malhas implementado nesse trabalho, ECon-MG. O gerador cria um arquivo com os dados da malha da geometria analisada. Em seguida, esse arquivo é usado pelo *solver* E-Con3D para realização da análise numérica. Depois que a análise é feita pelo *solver*, os resultados obtidos são carregados no MatLAB® para fins de visualização.

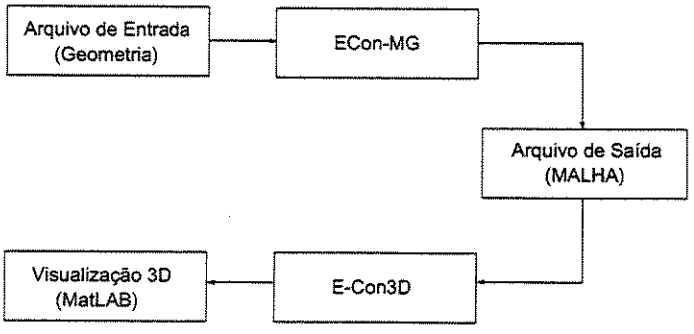


Figura 4.6: Fluxograma do processo para realização de uma análise numérica.

4.4 Fluxograma do Programa

O fluxograma simplificado de execução do programa de geração de malhas implementado é apresentado pela figura (4.7). O processo de geração da malha se inicia com a leitura do arquivo de entrada. Assim que esse arquivo de entrada é lido, o programa já tem armazenado na forma de listas todas as informações necessárias para a geração da malha. São então criadas três listas: uma lista de nós que contém objetos do tipo *Node* com as informações dos vértices da geometria, uma lista de segmentos que contém objetos do tipo *Segment* e uma lista de faces que contém objetos do tipo *Facet*. O valor do $MeshPar$ é armazenado em uma variável do tipo *double*.

O processo se inicia a partir de um *loop* na lista de faces. Para cada face da lista de faces é aplicada uma matriz de transformação geométrica que irá deslocá-la para o plano xy. Esse processo é detalhado na seção 4.6. Depois que a face está no plano xy é feito um *loop* em todos os segmentos que definem a face. Cada face possui como atributo uma lista de segmentos que contém todos os segmentos da face. Inicialmente verifica-se se o segmento já foi previamente discretizado ou não. Essa verificação é feita pois cada segmento pertence a duas faces. Se o segmento já foi discretizado, dá-se continuidade ao *loop* analisando-se o segmento seguinte da lista. Caso não tenha sido discretizado, a discretização é então realizada usando como parâmetro o valor do $MeshPar$, explicado anteriormente. Portanto, os nós nas arestas da geometria são definidos *a priori*, garantindo a continuidade final da malha nas arestas.

Depois que o *loop* na lista de segmentos da face é encerrado, é chamado o método que gera um *grid* de pontos alternados que cobre a região retangular mínima que contém a face. Esse procedimento será apresentado de forma mais detalha na seção 4.7. Em seguida é chamado um método que faz uma seleção dos pontos desse grid de pontos gerado, definindo-se, assim, os futuros nós que estarão presentes na malha, ver seção 4.8. Com o conjunto de pontos definido, aplica-se o método da triangulação de Delaunay verificando se ocorre a geração de um, ou alguns elementos, de forma não desejada. É necessário verificar se todos os triângulos gerados estão dentro da área da face e se não existe caso de formação de

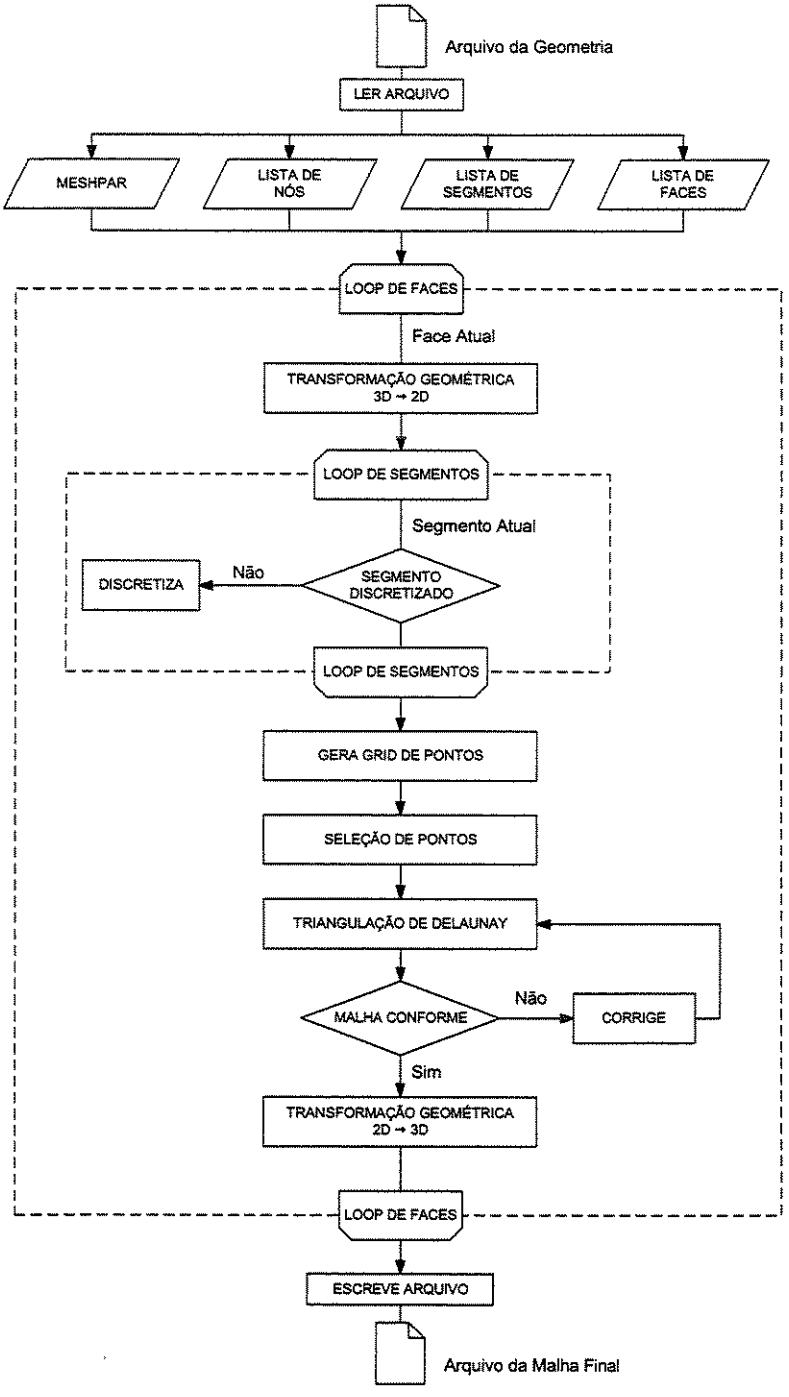


Figura 4.7: Fluxograma de execução do gerador de malhas implementado ECon-MG.

triângulos sobrepostos quando existem mais de três pontos em um círculo de Delaunay. Essas verificações serão apresentadas de forma mais detalhada na seção 4.9. Assim que a geração da malha for concluída de forma adequada na face, a inversa da matriz de transformação, que foi usada para deslocar a face do espaço tridimensional para o plano xy, é aplicada para enviar a face de volta para sua posição original no espaço. Então, outra face da lista de faces é selecionada e o processo continua até que todas as faces tenham sido analisadas. Ao final do *loop* de faces o programa terá armazenado uma lista de nós contendo as informações dos nós da malha e uma lista de elementos contendo a conectividade nodal. Finalmente, um arquivo com as informações da malha final é escrito e servirá de entrada para o *solver* E-Con3D para a realização das análises numéricas.

4.5 Diagrama de Classes

Nesta seção serão apresentadas as classes principais do programa. As classes foram modeladas visualmente usando-se a notação UML no programa Rational Rose®. A notação UML foi escolhida por se tratar de uma notação de modelagem de sistemas e programas orientados a objetos que foi originada a partir de um processo de unificação de várias notações. Essa notação tende a se tornar a escolha padrão para especificar, documentar e construir *softwares* de sistemas no futuro. Maiores informações sobre a notação UML podem ser encontradas em [49]. Um enfoque especial será dado para as classes *Node*, *Segment*, *Facet*, *Element*, *Triag*, *Mesh* e *GeoTrafo*, que são as principais classes do programa.

4.5.1 Estrutura das Classes

Na figura (4.8) são apresentadas as classes *Node*, *Element*, *Triag* e *Centroid* modeladas no programa Rose® usando UML. É possível ver, também, a classe *Dictionary*, com seus atributos e métodos escondidos. A classe *Dictionary* não foi desenvolvida nesse trabalho. Essa classe foi importada do *Meflab*$^{++}$, um *solver* numérico em elementos finitos, desenvolvido principalmente pelo prof. Dr. Janito Vaqueiro Ferreira e pelo prof. Dr. Renato Pavanello,

ambos do Departamento de Mecânica Computacional da Unicamp. A classe *Dictionary* é muito importante para o programa desenvolvido pois possibilita a leitura do arquivo de entrada de dados de maneira fácil e eficiente. Percebe-se pelo diagrama apresentado pela figura (4.8) que tanto a classe *Element* como a classe *Node* dependem da classe *Dictionary*. Aliás, as classes *Segment* e *Facet* também dependem, só que não foram exibidas nesse diagrama.

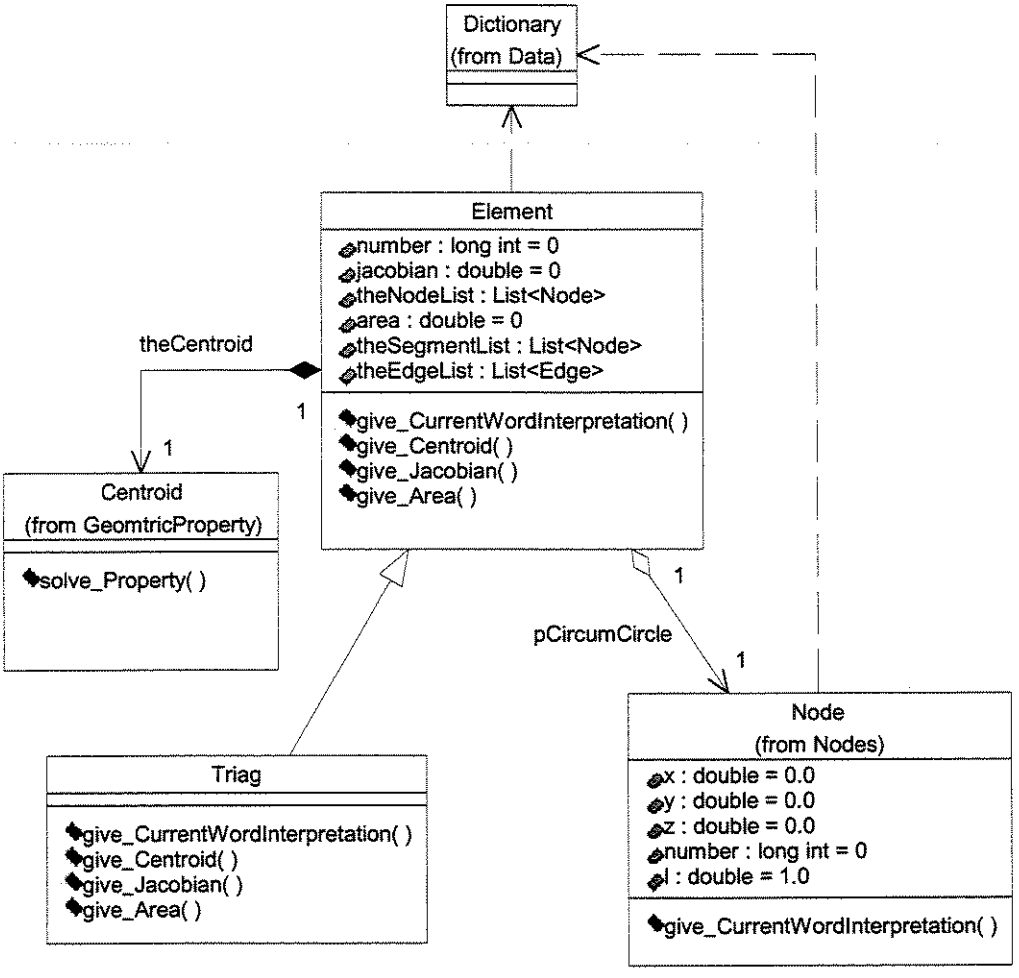


Figura 4.8: Diagrama de classe que relaciona a classe *Node* com a classe *Element*.

A classe *Element* possui uma série de atributos e métodos como pode ser visto. A assinatura dos métodos, isto é, o tipo do método e os argumentos que são passados a ele, não foi mostrada no diagrama para simplificar visualmente a modelagem desenvolvida. O método

give_CurrentWordInterpretation() é um método estático e, por isso, pode ser chamado sem a necessidade de se instanciar um objeto. Basta usar o nome da classe e o operador "::" seguido pelo nome do método. A classe *Triag* é a única classe derivada da classe *Element*, pois as malhas geradas serão constituídas somente por elementos triangulares. O método *give_CurrentWordInterpretation()* é um especialista que tem a capacidade de interpretar as informações do arquivo de entrada de dados relacionadas a uma palavra previamente adicionada ao dicionário. Percebe-se então que as classe *Node*, *Segment* e *Facet* também possuem o método *give_CurrentWordInterpretation()* para possibilitar a leitura da estrutura de dados apresentada pela figura (4.2). Os demais métodos da classe *Element*, *give_Centroid()*, *give_Jacobian()* e *give_Area()* são métodos virtuais, que permitem a atribuição de ponteiros do tipo *Element* a objetos do tipo *Triag*. Com isso, o código para cálculo do centróide, do jacobiano e da área de um elemento triangular são implementados nos métodos da classe *Triag*.

Pode-se perceber, ainda, duas associações existentes na figura (4.8): uma associação entre a classe *Element* e a classe *Node* e uma associação entre a classe *Element* e a classe *Centroid*. Na primeira, um objeto do tipo *Node* é agregado por referência a classe *Element*, o que implica que todo objeto do tipo *Element* vai possuir um atributo do tipo *Node*, chamado de *pCircumCircle*, agregado por referência. Esse atributo *pCircumCircle* vai possuir as coordenadas do centro do círculo que circunscreve um elemento *Triag*. Na segunda, um objeto do tipo *Centroid*, *theCentroid*, é agregado por valor a classe *Element*.

Na figura (4.9) são apresentadas as classes *Segment*, *Facet* e *Edge*. São apresentadas, também, as classes *List* e *Container* que são classes *Template*, isto é, são parametrizadas. Essas classes não foram desenvolvidas neste trabalho e, também, foram importadas do *Meflab*$^{++}$. Algumas pequenas modificações foram feitas na classe *List* mas, essencialmente, ela é a mesma que se encontra implementada no *Meflab*$^{++}$. Percebe-se que a classe *List* depende das classes *Node*, *Segment*, *Facet*, *Edge* e *Element*, pois é necessário criar-se listas de objetos do tipo *Node*, do tipo *Segment* e assim por diante.

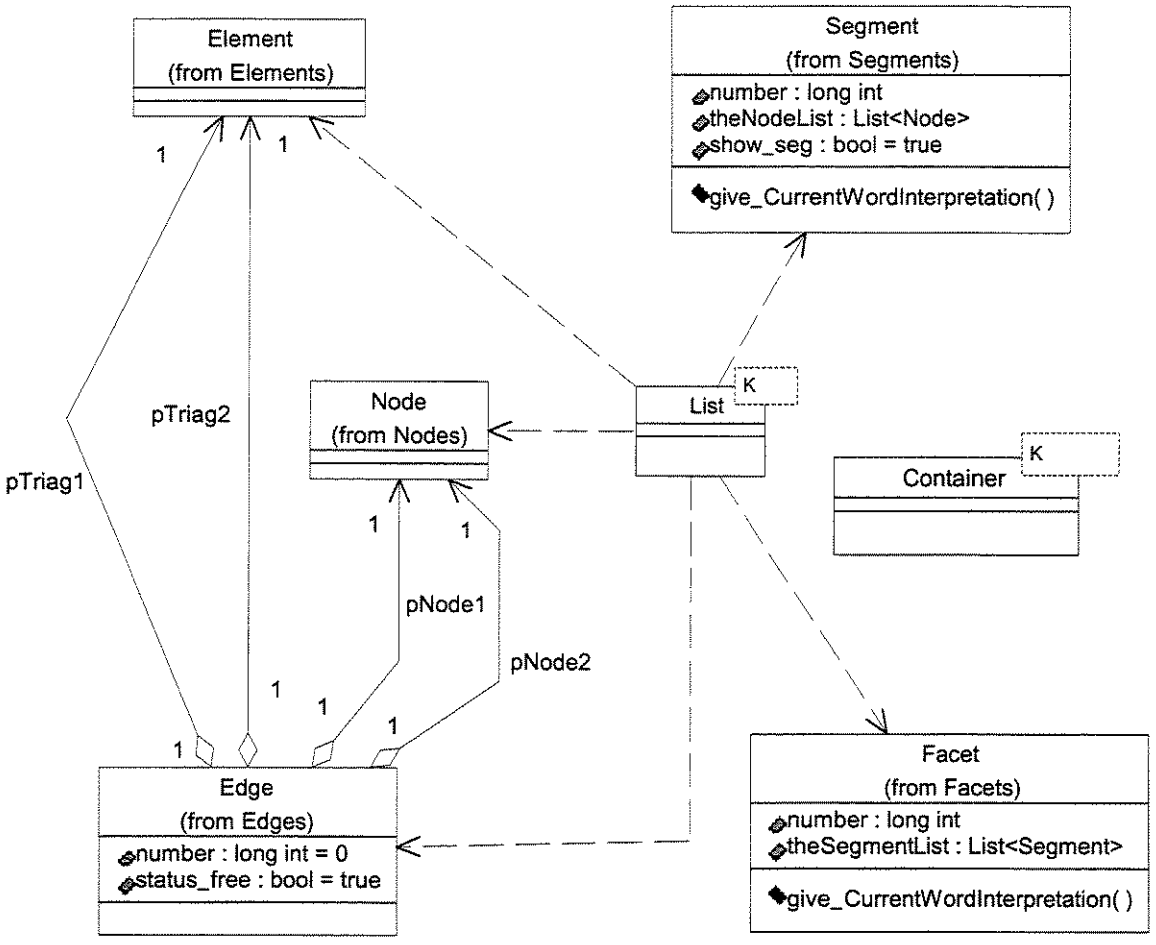


Figura 4.9: Diagrama apresentando a modelagem das classes *Segment*, *Facet* e *Edge*.

A classe *Edge* foi abstraída visando o aprimoramento do algoritmo de geração de elementos. Com a classe *Edge* é possível conhecer os entes geométricos em uma determinada região de interesse de forma mais rápida e eficiente. Por exemplo, da forma como a classe *Edge* se relaciona com a classe *Element* e com a classe *Node*, é possível perceber que cada objeto do tipo *Edge* possui dois objetos do tipo *Element* (*pTriag1* e *pTriag2*) e dois objetos do tipo *Node* (*pNode1* e *pNode2*). Dessa forma, para um determinado objeto *Edge*, ficam imediatamente conhecidos quais são os dois triângulos da malha que fazem parte dele, e quais são os dois nós que o definem. Isso é extremamente benéfico do ponto de vista de economia de tempo de processamento, pois não é mais necessário fazer uma busca por todo um

conjunto de entes geométricos, mas somente em uma determinada região de interesse. Isso possibilita uma futura implementação de um novo algoritmo para geração de malha mais eficiente e veloz, aproveitando a estrutura que foi desenvolvida até o momento. Devido a estrutura ter sido desenvolvida usando o paradigma da orientação a objetos, basta adicionar um novo método na classe *Mesh*, que é apresentada pela figura (4.11), com o código desse novo algoritmo para geração de elementos.

A seguir serão apresentadas as classes *GeoTrafo* e *Mesh* que são as classes mais importantes do programa e que contém os métodos para a geração das malhas nas geometrias. Os atributos e métodos da classe *GeoTrafo* podem ser visualizados pela figura (4.10) e os atributos e métodos da classe *Mesh* podem ser visualizados pela figura (4.11). Novamente, para facilitar a visualização, a assinatura dos métodos não foi exibida. O código implementado nos principais métodos dessas classes poderá ser entendido nas seções seguintes onde os pontos mais importantes da estrutura do programa serão abordados em detalhe.

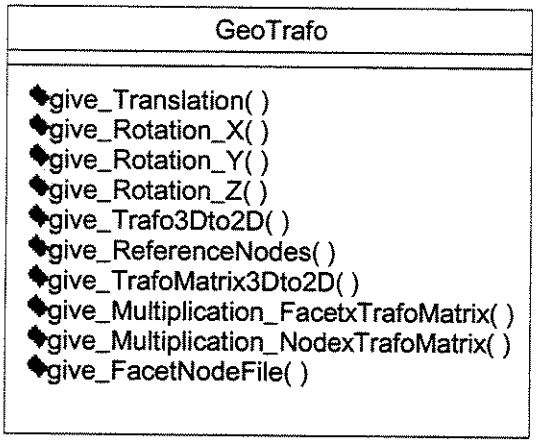


Figura 4.10: Classe *GeoTrafo* modelada usando a notação UML.

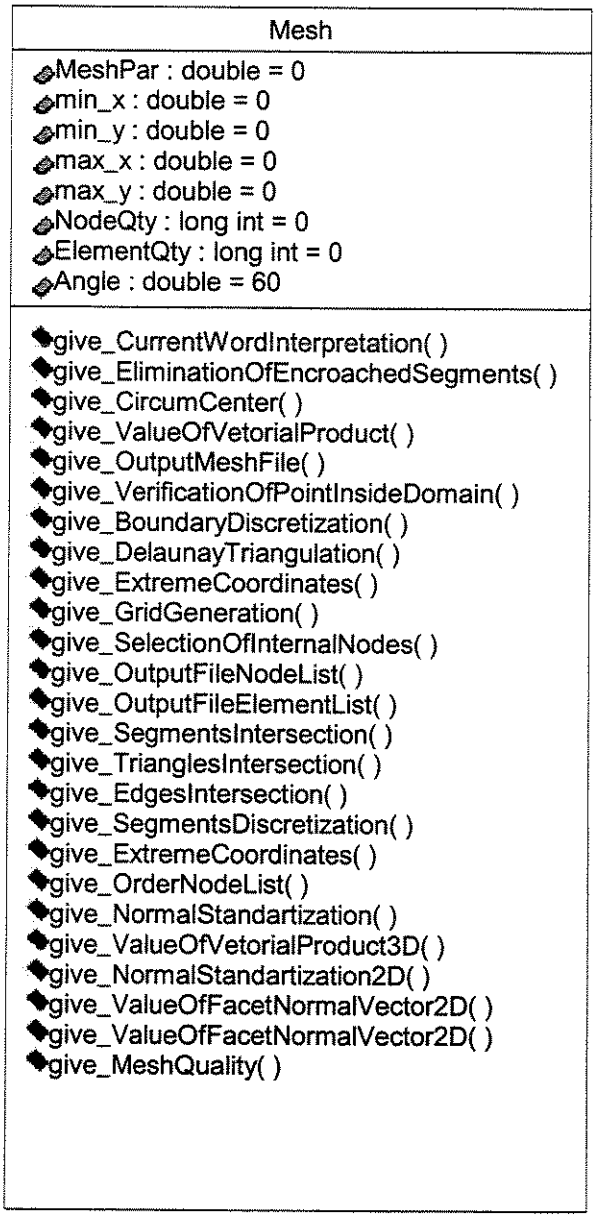


Figura 4.11: Classe *Mesh* modelada usando a notação UML.

4.6 Transformações Geométricas

O processo de deslocar cada face do espaço $\Re^3$ para o plano xy é conseguido pelo uso de matrizes de transformação [24]. Na figura (4.12), esse processo é apresentado passo a passo.

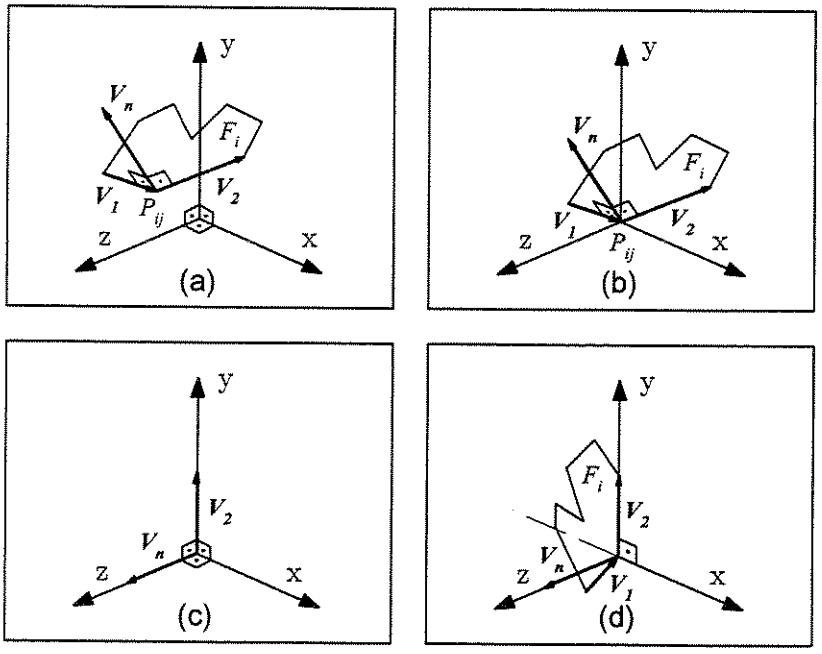


Figura 4.12: Processo para deslocar a face F_i no espaço $\Re^3$ para o plano xy.

Na figura (4.12a) é apresentada uma face plana qualquer no espaço tridimensional, formada por segmentos de reta. Esta face pode ser um polígono não-convexo e possuir rasgos em seu interior. Inicialmente é detectado o ponto da face que contém a coordenada y mínima, neste caso P_{ij}. A partir desse ponto aplica-se uma matriz de translação a toda a face, a fim de deslocá-la para a origem do sistema de coordenadas, conforme a figura (4.12b). A matriz de translação [**A**] é dada por

$$A_{ij} = \delta_{ij} + \delta_{i4}(-P_{ki}) - \delta_{i4}(-P_{ki})\delta_{j4} \quad \text{para} \quad i,j,k = 1,4 \tag{4.2}$$

que na forma expandida é apresentada como

$$[\mathbf{A}] = \begin{bmatrix} 1 & 0 & 0 & -P_{ix} \\ 0 & 1 & 0 & -P_{iy} \\ 0 & 0 & 1 & -P_{iz} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

sendo que P_{ix}, P_{iy} e P_{iz} são as coordenadas x, y e z do ponto P_i. O vetor $\mathbf{V_n}$ é o vetor normal a face que pode ser obtido pelo produto vetorial de $\mathbf{V_1}$ e $\mathbf{V_2}$, como dito anteriormente. Depois de deslocar a face para a origem do sistema de referência, basta aplicar uma matriz de transformação geométrica $[\mathbf{B}]$ a fim de alinhar $\mathbf{V_n}$ com o eixo z e $\mathbf{V_2}$ com o eixo y. Isto é mostrado pela figura (4.12c). A matriz de transformação $[\mathbf{B}]$ que desloca a face F_i do espaço $\Re^3$ para o plano xy é dada por

$$B_{ij} = R_{ij} - (\delta_{4j}R_{ij} + \delta_{i4}R_{ij}) + \delta_{i4}\delta_{j4}R_{ij} + \delta_{i4}\delta_{j4} \quad \text{para} \quad i,j = 1,4 \tag{4.4}$$

que na forma expandida é apresentada como

$$[\mathbf{B}] = \begin{bmatrix} R_{1x} & R_{1y} & R_{1z} & 0 \\ R_{2x} & R_{2y} & R_{2z} & 0 \\ R_{3x} & R_{3y} & R_{3z} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.5}$$

sendo que as componentes dos vetores $\mathbf{R_1}$, $\mathbf{R_2}$ e $\mathbf{R_3}$ podem ser calculados por

$$\mathbf{R_3} = \frac{\mathbf{V_n}}{|\mathbf{V_n}|}, \qquad \mathbf{R_1} = \frac{\mathbf{V_2} \times \mathbf{R_3}}{|\mathbf{V_2} \times \mathbf{R_3}|} \quad \text{e} \quad \mathbf{R_2} = \mathbf{R_3} \times \mathbf{R_1}.$$

Portanto, a matriz de transformação $[\mathbf{T}]$ a ser implementada é calculada pelo produto da matriz $[\mathbf{B}]$ com a matriz $[\mathbf{A}]$

$$T_{ij} = B_{ik}A_{kj} \tag{4.6}$$

aplicando sobre um conjunto de pontos P, a face F_i é deslocada para o plano xy como mostrado pela figura (4.12d) pela transformação

$$P_{ij}^{T} = T_{ik} P_{kj} \tag{4.7}$$

O método que calcula a matriz de transformação [**T**], pelo uso da biblioteca Newmat10, é o método $give_TrafoMatrix3Dto2D(List < Segment > \&\ \ theSegmentList,\ \ Matrix\& \ TrafoMatrix)$. A inversa da matriz de transformação, $[\mathbf{T}]^{-1}$, desloca a face do plano xy para sua posição original no espaço $\Re^3$.

4.7 Geração do *Grid* de Pontos

Depois de aplicada a matriz de transformação geométrica [**T**], a face já se encontra no plano xy. O método de geração do *grid* de pontos escolhido consiste em gerar um conjunto de pontos P no plano xy de forma alternada. Primeiramente determinam-se as coordenadas x e y máximas e mínimas de modo a formar o menor retângulo possível que contenha toda a área da face F_i, figura (4.13).

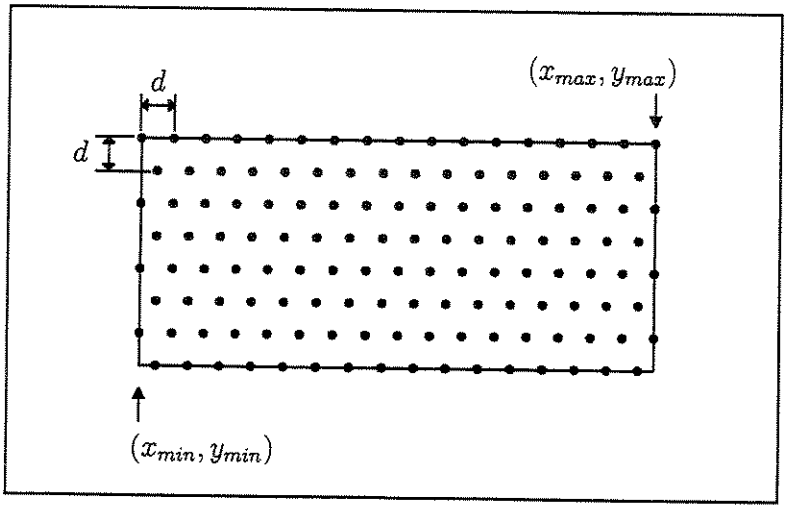


Figura 4.13: Geração do *grid* de pontos para a triangulação de Delaunay.

Em seguida calcula-se o parâmetro d para realizar a geração dos pontos de forma incremental. O parâmetro d é determinado com o auxílio do valor do $MeshPar$. Primeiramente calcula-se o número de divisões N_div, que é um inteiro, pela equação

$$N_div = int\left(\frac{|(x_{max} - x_{min})|}{MeshPar}\right) \tag{4.8}$$

e depois o valor d por

$$d = \frac{|(x_{max} - x_{min})|}{N_div} \tag{4.9}$$

em seguida, a partir do ponto (x_{min}, y_{min}) cria-se o *grid* de pontos usando como incremento de coordenadas o parâmetro d, nas direções x e y. Ao final do processo, os novos pontos criados preenchem toda a área retangular previamente estabelecida.

Esse é um método bastante simples de geração de pontos e que proporciona um alto controle na forma dos triângulos da malha. Os triângulos gerados a partir de um *grid* de pontos como esse, tendem a possuir uma forma muito próxima à de um triângulo equilátero. O método $give_GridGeneration(List < Node > \&\quad theGridList)$ é o método responsável pela geração desse *grid*. O argumento *theGridList* é uma lista de nós que é passado por referência e possui todos pontos que preenchem a área retangular minima definida por (x_{max}, y_{max}) e (x_{min}, y_{min}). Pode-se perceber, ainda, que esse método de geração de pontos não gera pontos de forma adaptativa, isto é, com uma maior concentração de pontos nas regiões onde existem os maiores gradientes de solução. Entretanto, outros tipos de método para a geração de *grid* de pontos podem ser facilmente implementados de acordo com a necessidade do problema analisado e aproveitando toda a estrutura computacional já desenvolvida. Para isto basta adicionar um novo método que gere os pontos com as características desejadas ou modificar o método já existente.

4.8 Seleção dos Pontos que Serão os Nós da Malha

Depois de gerado o *grid* de pontos é necessário fazer uma seleção para se determinar os pontos que serão os futuros nós da malha. Primeiramente, é preciso eliminar os nós que não estão no interior da face. O algoritmo para verificar se um ponto está no interior ou não da face é ilustrado pela figura (4.14).

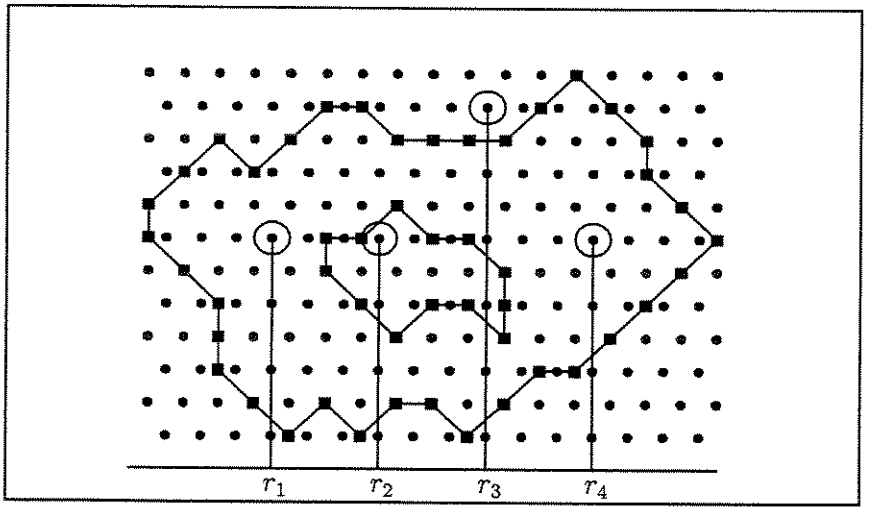


Figura 4.14: Ilustração do algoritmo para determinação de ponto interno a uma face.

Para cada ponto do conjunto de pontos é "traçada uma reta" partindo do ponto analisado para o exterior da geometria e verifica-se quantas vezes essa reta intersecciona o contorno da face. Se o número de intersecções for par, significa que o ponto está fora da face, se for um número ímpar de intersecções, o ponto está no interior da face. Demonstrações e provas matemáticas podem ser encontradas em [31]. Os métodos encarregados de filtrar os pontos internos a face são os métodos *give_SelectionOfInternalNodes*(*List* < *Node* > & *theGridList*, *List* < *Node* > & *theBoundaryNodeList*, *List* < *Segment* > *theFacetSegmentList*, *List* < *Node* > & *theInternalFacetNodeList*) e o método que efetivamente faz a verificação *give_VerificationOfPointInsideDomain*(*double xc*, *double yc*, *List* < *Segment* > *theFacetSegmentList*).

Por exemplo, analisando os quatro pontos circulados representados na figura (4.14) e traçando-se as respectivas retas r_1, r_2, r_3 e r_4, tem-se:

- *para a reta* r_1: o ponto circulado que dá origem a reta r_1 está dentro da geometria, pois r_1 intersecciona o contorno somente uma vez;
- *para a reta* r_2: a reta r_2 intersecciona o contorno duas vezes, portanto, o ponto analisado está fora da geometria;
- *para a reta* r_3: o ponto circulado que dá origem a reta r_3 também está fora da geometria, pois r_3 intersecciona o contorno quatro vezes;
- *para a reta* r_4: a reta r_4 intersecciona o contorno somente uma vez, portanto, o ponto circulado está dentro da geometria.

No algoritmo implementado computacionalmente, são usadas as coordenadas do ponto a ser analisado e a lista de segmentos que define o contorno da face. Primeiramente é feito um incremento temporário na coordenada x do ponto analisado por um valor definido empiricamente de 0,001. Isto é feito para se evitar problemas com vértices do contorno que estão alinhados verticalmente com este ponto. Em seguida é feito um *loop* nos segmentos que define a face, determinando-se todos os segmentos cujas coordenadas x de suas extremidades englobam o valor da coordenada x do ponto analisado. Depois disso, todos os segmentos do contorno que estão abaixo do ponto analisado são separados e contados. Se o número de segmentos for par implica que o ponto está fora do domínio da face, caso contrário ele está dentro, como dito anteriormente.

Depois que os pontos externos a face são eliminados, obtém-se um conjunto de pontos como mostrado na figura (4.15). Todos os pontos desse conjunto estão no interior da face e são representados por "•". O pontos representados por "▪"são os pontos originados pela discretização já realizada anteriormente no contorno. A fim de se evitar a formação de triângulos ruins, ou seja, com ângulos internos muito pequenos, eliminam-se os pontos que estão muito próximos a algum ponto do contorno. Os pontos que serão eliminados foram

circulados na figura (4.15). Esses pontos foram selecionados usando um critério de distância entre pontos. Atualmente o programa usa como distância mínima permitida entre os pontos do *grid*, o valor de $0,8 * MeshPar$, definido empiricamente. O conjunto de pontos final, ao qual será aplicado o método da triangulação de Delaunay é apresentado na figura (4.16).

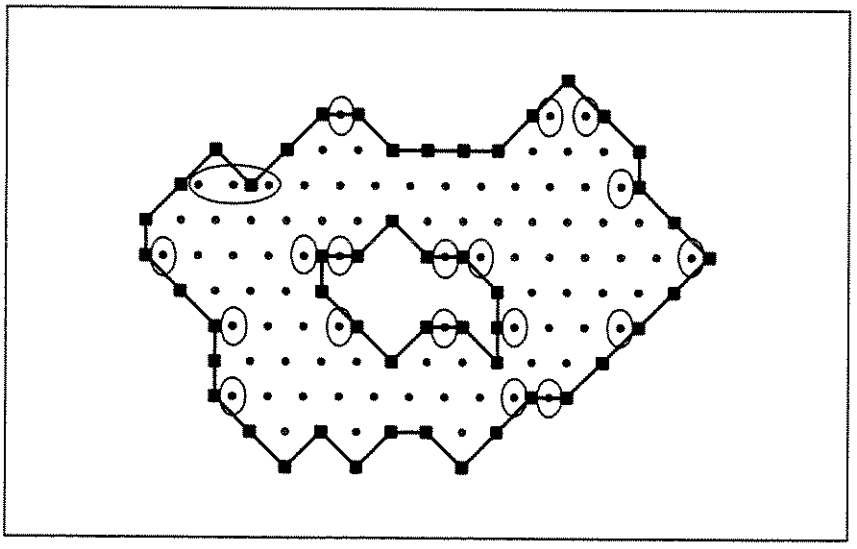


Figura 4.15: Eliminação dos pontos muito próximos a algum ponto do contorno.

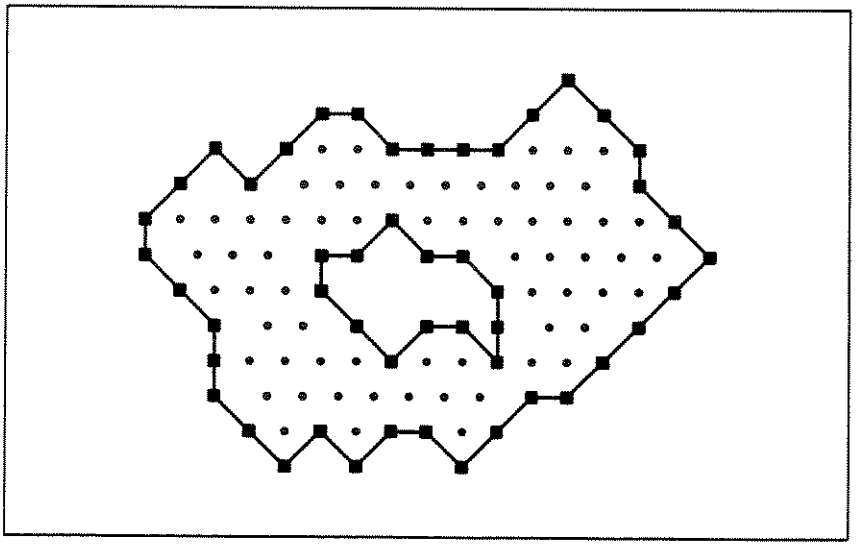


Figura 4.16: Conjunto de pontos final que serão os nós da malha.

4.9 Triangulação de Delaunay

O método da triangulação de Delaunay foi apresentado em detalhes na seção 2.2.5. O programa de geração de malhas implementado nesse trabalho, chamado de ECon-MG, usa esse método para estabelecer a conectividade entre os nós da malha, baseando-se, principalmente, na propriedade dos circuncírculos vazios. O princípio consiste em realizar uma varredura no conjunto de pontos P, selecionando três pontos p_1, p_2 e p_3 em cada iteração e, verificar se existe algum outro ponto do conjunto de pontos P interior a circunferência que circunscreve o triângulo formado por p_1, p_2 e p_3, figura (4.17). O método que realiza a triangulação de Delaunay no ECon-MG é o *give_DelaunayTriangulation(double normal_k, List < Node > theNodeList, List < Element > & theElementList, List < Segment > theFacetSegmentList)*. Este método recebe uma variável do tipo *double normal_k* que indica a orientação da face para que os vetores normais dos elementos a serem gerados apontem para fora da geometria. Além disso, ele recebe uma lista de nós, a partir da qual os elementos serão gerados e, uma lista de segmentos, que determina o contorno da face. O resultado obtido é uma lista de elementos que é passada por referência pelo método.

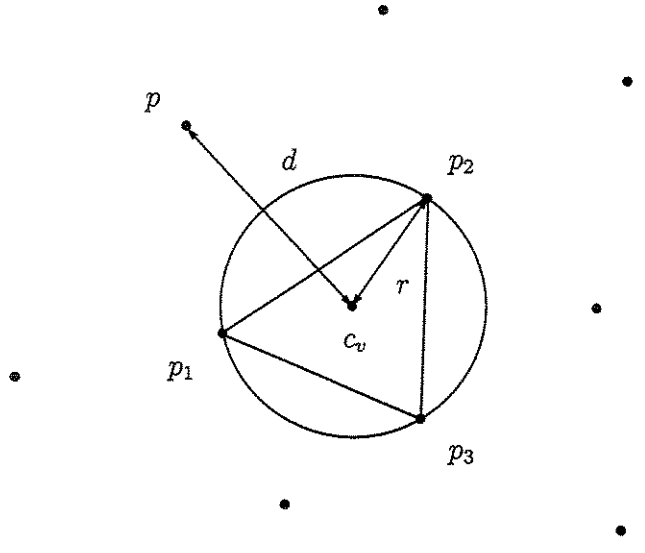


Figura 4.17: Propriedade do circuncírculo vazio da triangulação de Delaunay.

O cálculo das coordenadas do centro do círculo que circunscreve o $\Delta p_1 p_2 p_3$ é feito por

$$c_{vx} = \frac{(p_{1x}^2 + p_{1y}^2)(p_{2y} - p_{3y}) + (p_{2x}^2 + p_{2y}^2)(p_{3y} - p_{1y}) + (p_{3x}^2 + p_{3y}^2)(p_{1y} - p_{2y})}{2\mathrm{D}(\mathbf{C})} \tag{4.10}$$

$$c_{vy} = \frac{(p_{1x}^2 + p_{1y}^2)(p_{3y} - p_{2y}) + (p_{2x}^2 + p_{2y}^2)(p_{1y} - p_{3y}) + (p_{3x}^2 + p_{3y}^2)(p_{2y} - p_{1y})}{2\mathrm{D}(\mathbf{C})} \tag{4.11}$$

sendo que $\mathrm{D}(\mathbf{C})$ é o determinante da matriz $[\mathbf{C}] = \begin{bmatrix} (p_{2x} - p_{1x}) & (p_{2y} - p_{1y}) \\ (p_{3x} - p_{1x}) & (p_{3y} - p_{1y}) \end{bmatrix}$, vide [44].

As coordenadas x e y do centro do círculo c_v são denotadas por c_{vx} e c_{vy}, respectivamente. Com isso determina-se o módulo do raio r e calcula-se o valor da distância d. Se $d > r$, o ponto p está fora do círculo que circunscreve o $\Delta p_1 p_2 p_3$, caso contrário, o ponto p está dentro.

4.9.1 O Algoritmo de Geração dos Elementos

```
for (i = 1; i < N - 1; i++) {
    for (j= i + 1; j < N; j++) {
        for (k = j + 1; k <= N; k++) {
            ⟼ calcula cvx e cvy do triângulo formado por pi, pj e pk
            ⟼ calcula o raio r de Δpipjpk
            for (l = 1; l <= N; l++) {
                ⟼ calcula a distância d do ponto pl a cv
                if(distância d < r)
                    ⟼ descarta Δpipjpk
            }
            ⟼ insere Δpipjpk na lista de elementos
        }
    }
}
```

O algoritmo de geração de elementos implementado no ECon-MG, foi apresentado acima de forma simplificada. Esse é um algoritmo de complexidade $O(n)^4$ e está apresentado em [44]. Objetivou-se principalmente, neste trabalho, desenvolver um estrutura para geração de malhas suficientemente robusta e eficiente. Posteriormente pode e deve ser implementado um outro algoritmo para realizar a varredura dos pontos e geração dos elementos de forma mais rápida. Isto é dito porque é sabido que o algoritmo de triangulação de Delaunay de complexidade $O(n)^4$ é demasiadamente lento e apresenta o problema de *overflow* em malhas com uma grande quantidade de nós. No código real foi necessário implementar alguns métodos para garantir a boa formação da malha. Nas faces com geometrias muito irregulares e com rasgos no interior é preciso verificar se acontece a geração de um, ou alguns, elementos fora da face, como ilustrado pela figura (4.18).

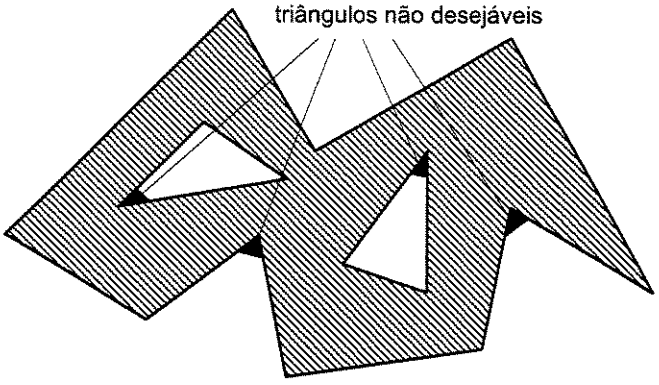


Figura 4.18: Triângulos gerados fora do domínio da face.

Essa verificação é feita calculando-se o centróide de cada triângulo e determinando se o centróide se encontra no interior da face, usando o método apresentado na seção 4.8. Além disso, é preciso evitar a formação de triângulos sobrepostos, no caso de existir mais de três pontos em um círculo de Delaunay, quando $d = r$. No programa foi definido d igual a r quando $(r - d) \geq -1 \times 10^{-9}$ e $(r - d) < 1 \times 10^{-9}$. Dessa forma, a formação de triângulos sobrepostos é evitada, quando existem mais de três pontos sobre uma circunferência de Delaunay, inserindo-se todos os triângulos circunscritos a essa circunferência em uma lista de

elementos auxiliar. Então, é feito um *loop* nessa lista e o triângulo que intersecciona algum outro triângulo dessa lista é removido, até que, ao final do processo, essa lista de elementos esteja totalmente filtrada e possa ser transferida para a lista de elementos principal. A determinação da intersecção entre dois triângulos é feita segmento a segmento de cada triângulo, ver seção 4.9.2. A malha final obtida para a figura (4.16) é mostrada na figura (4.19).

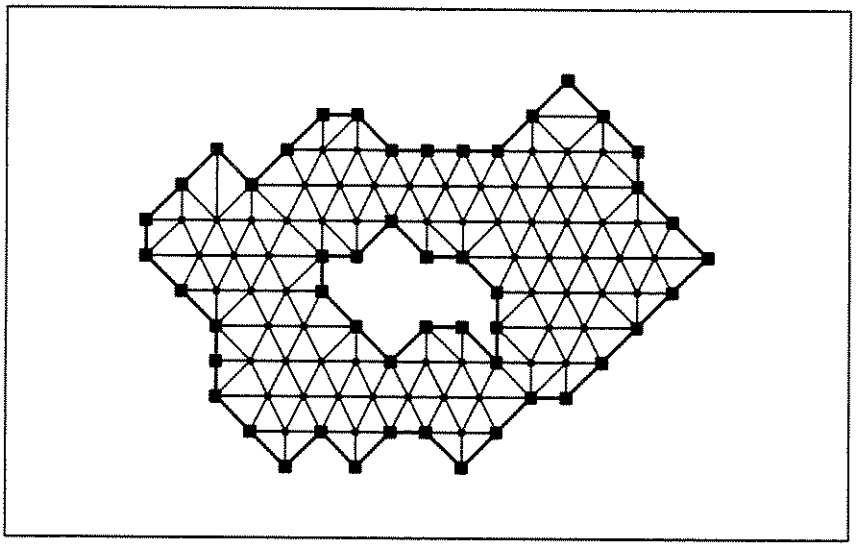


Figura 4.19: Malha final da face obtida pelo conjunto de pontos da figura (4.16).

4.9.2 Determinação da Intersecção de Dois Segmentos

Sejam dois segmentos p_1p_2 e p_3p_4 no plano xy, como mostrado pela figura (4.20). Os segmentos se interseccionam se p_3 e p_4 estão em lados opostos em relação a p_1p_2 e, ao mesmo

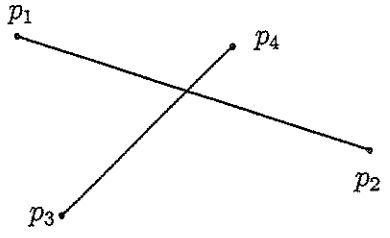


Figura 4.20: Ilustração da intersecção de dois segmentos.

tempo, p_1 e p_2 estão em lados opostos em relação a p_3p_4. A primeira condição ocorre se $(p_1p_2 \times p_1p_3)$ e $(p_1p_2 \times p_1p_4)$ têm sinais opostos. Analogamente, os produtos vetoriais $(p_3p_4 \times p_3p_1)$ e $(p_3p_4 \times p_3p_2)$ devem ter sinais opostos. Portanto os segmentos se interseccionam se, e somente se, $(p_1p_2 \times p_1p_3)(p_1p_2 \times p_1p_4) < 0$ e $(p_3p_4 \times p_3p_1)(p_3p_4 \times p_3p_2) < 0$. O método que determina se dois segmentos se interseccionam é o método *give_SegmentsIntersection(Segment* pSegment1, Segment* pSegment2)*. Este método retorna uma variável do tipo *bool*, *true* ou *false*, indicando se os segmentos se interseccionam ou não.

4.10 Critério de Qualidade das Malhas

A qualidade de uma malha é medida pela forma dos seus triângulos que, em aplicações numéricas, é desejável que se aproxime da forma de um triângulo equilátero. Medidas típicas analisam o maior ou o menor dos ângulos, a razão entre a menor e a maior de suas arestas, e, a razão entre os raios dos círculos inscrito e circunscrito. Os valores encontrados são analisados tendo por parâmetro as relações correspondentes nos triângulos equiláteros. A qualidade da malha é então dada pelo elemento de pior medida ou, ainda, pode ser feita uma média ponderada levando em consideração todos os elementos da malha.

A verificação da qualidade das malhas geradas pelo ECon-MG foi feita baseando-se no trabalho de Ruppert [50], que é uma das principais medidas para se verificar a qualidade de malhas formadas por elementos triangulares-lineares. Essa medida é calculada pela razão do aspecto do triângulo, que é avaliada pela razão do tamanho de sua maior aresta pelo tamanho da menor de suas alturas. A especificação de um ângulo mínimo α para os triângulos de uma malha fornece um limite de $(\pi - 2\alpha)$ para seu ângulo máximo e garante uma razão de aspecto entre $\left|\frac{1}{\text{sen}(\alpha)}\right|$ e $\left|\frac{2}{\text{sen}(\alpha)}\right|$. Isso pode ser verificado considerando-se a figura (4.21) e α como sendo o menor ângulo do triângulo formado pelos pontos a, b e c. Pode-se, então, escrever as seguintes relações, uma vez que, o lado ab é no mínimo tão longo quanto cb

$$|b-a| \geq \frac{|c-a|}{2} \; e \; |b-a| \leq |c-a|.$$

Se d é a projeção de b na aresta ac, então, $|b-d| = |b-a|\text{sen}(\alpha)$, o que conduz a

$$\left|\frac{1}{\text{sen}(\alpha)}\right| \leq \frac{|c-a|}{|b-d|} \leq \left|\frac{2}{\text{sen}(\alpha)}\right| \tag{4.12}$$

e, portanto, um limite para α, fornece um limite para a razão de aspecto e vice-versa.

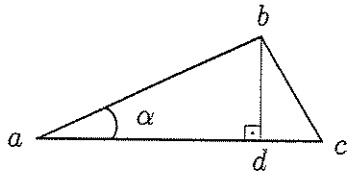


Figura 4.21: Triângulo qualquer da malha, onde α é o ângulo mínimo.

Como objetiva-se que a forma dos triângulos da malha fique mais próxima possível à forma de um triângulo equilátero, α é admitido ser 60° para determinação da razão do aspecto do triângulo. Calcula-se, então, o ângulo mínimo de cada triângulo da malha e verifica-se se esse ângulo está dentro ou fora da faixa de valores aceitável, usando a expressão (4.12) para $\alpha = 60°$. Definiu-se a característica ON para os triângulos que estão dentro da faixa de valores e OFF para os triângulos que estão fora. Além disso, os triângulos são classificados de acordo com os seus ângulos internos mínimos nos seguintes intervalos:

- A0-A5 ($0° < ang_min \leq 5°$);
- A5-A15 ($5° < ang_min \leq 15°$);
- A15-A25 ($15° < ang_min \leq 25°$);
- A25-A35 ($25° < ang_min \leq 35°$);
- A35-A60 ($35° < ang_min \leq 60°$);

sendo que *ang_min* é o menor ângulo interno do triângulo.

4.10.1 Cálculo do Ângulo Interno Mínimo de um Triângulo

O método que calcula os parâmetros de qualidade das malhas das geometrias estudadas é o *give_MeshQuality(double& minimum_angle, long int last_value, List < Element > theElementList, List < Element > & theElementList_A0_A5, List < Element > & theElementList_A5_A15, List < Element > & theElementList_A15_A25, List < Element > & theElementList_A25_A35, List < Element > & theElementList_A35_A60, List < Element > & theElementList_ON, List < Element > & theElementList_OFF)*.

Para se determinar o ângulo interno mínimo de um triângulo genérico, como mostrado na figura (4.22), é feito o seguinte procedimento:

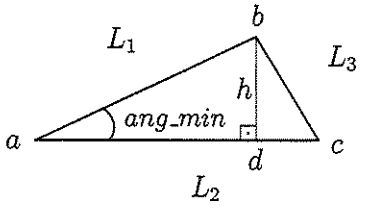


Figura 4.22: Triângulo qualquer da malha, onde *ang_min* é o ângulo mínimo.

- determina-se o tamanho de cada lado do triângulo;
- os lados são classificados com relação ao tamanho:
  - L_1 : maior tamanho;
  - L_2 : tamanho intermediário;
  - L_3 : menor tamanho;
- pela lei dos cossenos tem-se

$$ang_min = acos\left(\frac{L_1^2 + L_3^2 - L_2^2}{2L_1L_3}\right) \qquad \text{e} \qquad h = L_3 sen(ang_min).$$

Assim o ângulo mínimo de qualquer triângulo é facilmente determinado, bem como, sua altura h que é fundamental para a determinação da razão do aspecto do triângulo.

Capítulo 5

Resultados e Discussões

5.1 Interface de Visualização

Inicialmente é apresentada a interface gráfica do programa de geração de malhas desenvolvido, chamado de ECon-MG, que utiliza funções da biblioteca OpenGL® para exibição gráfica [62]. Essa interface ainda se encontra em uma fase inicial de desenvolvimento e é usada, somente, para exibir malhas de domínio em geometrias bidimensionais [12, 13]. Para exibição das malhas tridimensionais de superfície, que serão usadas pelo método dos elementos de contorno, foi usado o programa MatLAB®.

As malhas tridimensionais não foram exibidas usando a interface desenvolvida porque objetivou-se construir um projeto único e integrado. Se for apenas uma questão de visualização, a interface já está pronta para exibir malhas 3D, basta usar as funções da biblioteca OpenGL® apropriadas. Entretanto, procurou-se desenvolver um ambiente integrado usando o *MFC AppWizard (exe)*, onde estariam presentes tanto os algoritmos de geração de malha como os de visualização. Como não foi conseguido compilar a biblioteca NewMat10, que possui vinculação estática (*.lib*) no projeto *MFC AppWizard (exe)*, a interface de visualização ficou separada do módulo de geração de malha e as malhas 3D foram plotadas no MatLAB® para simplificar e agilizar a visualização dos resultados. Uma descrição detalhada sobre como criar uma aplicação usando o compilador MS-Visual C++® para exibição gráfica usando a biblioteca OpenGL® pode ser encontrada no Anexo A.

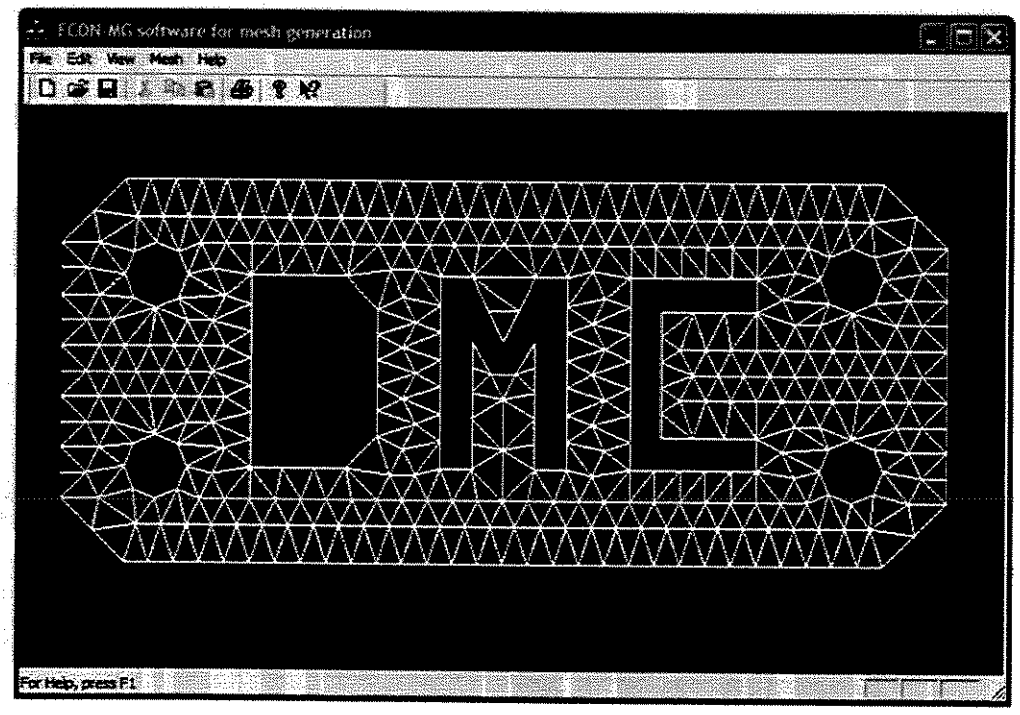


Figura 5.1: Malha de domínio bidimensional com 421 nós e 649 elementos.

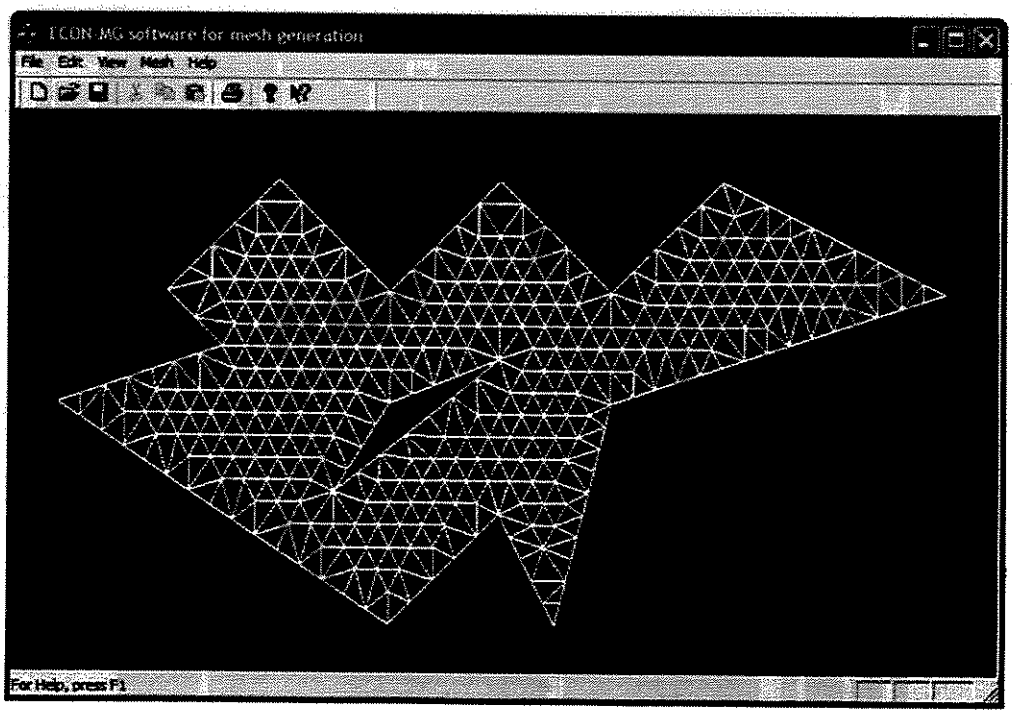


Figura 5.2: Malha de domínio bidimensional com 356 nós e 602 elementos.

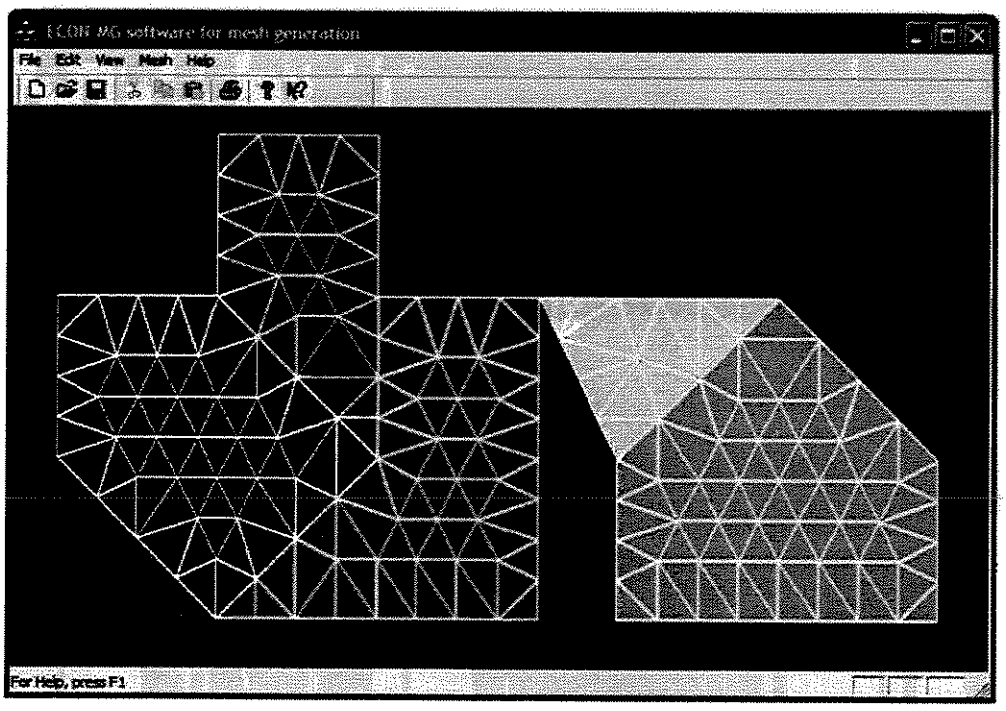


Figura 5.3: Malha de uma geometria multifacetada com 162 nós e 247 elementos.

Na figura (5.1) é apresentada a janela do programa ECon-MG que funciona em ambiente Windows®. A área de saída gráfica está incorporada na janela principal do programa, o que proporciona robustez e praticidade. A malha apresentada na figura (5.2) é uma malha de domínio em uma geometria bidimensional arbitrária. Pode-se notar, tanto na figura (5.1) como na figura (5.2), uma boa qualidade dos triângulos formados.

A figura (5.3) representa uma geometria bidimensional composta por 5 faces. Cada região delimitada por uma cor representa uma face. Nessa geometria pode-se verificar a continuidade da malha nas arestas. Depois de estabelecida a continuidade da malha nas arestas, a composição de uma malha em uma geometria tridimensional fica definida pela aplicação das matrizes de transformação geométrica apresentadas na seção 4.6.

BIBLIOTECA CENTRAL
DESENVOLVIMENTO
COLEÇÃO
UNICAMP

5.1.1 Programa de Visualização das Malhas em MatLAB®

Nesta seção é apresentado um programa em MatLAB® para visualização tridimensional das malhas geradas. O programa necessita de duas matrizes previamente carregadas no *Workspace* do MatLAB®: *Nodes* e *Elements*. A matriz *Nodes* contém as informações dos nós da malha, sendo que, as coordenadas x, y, e z de cada nó da malha, são dadas por suas colunas 2, 3 e 4, respectivamente. A matriz *Element* possui a conectividade nodal. Nas colunas 2, 3 e 4 da matriz *Element* estão os números do primeiro, segundo e terceiro nós que formam um triângulo da malha.

Primeiramente é feito um *loop* de 1 até o número de elementos da malha. São criados três vetores x, y e z, que contêm as coordenadas x, y e z de cada nó do elemento que está sendo construído. Em seguida define-se a cor para cada nó da malha. No programa apresentado a cor é definida no vetor c. Quando $c = [0.9\ 0.9\ 0.9]$ significa que todos os nós do elemento possuem a cor cinza. O comando patch é usado para desenhar e colorir o triângulo a partir das coordenadas e cores dos nós que o definem. Pode-se associar a cada nó uma cor diferente que representa um valor da solução numérica. O preenchimento do triângulo será feito pela interpolação linear das cores do nós.

```
figure

    for t = 1 : length(Elements)
        x = [Nodes(Elements(t,2),2);Nodes(Elements(t,3),2);Nodes(Elements(t,4),2)];
        y = [Nodes(Elements(t,2),3);Nodes(Elements(t,3),3);Nodes(Elements(t,4),3)];
        z = [Nodes(Elements(t,2),4);Nodes(Elements(t,3),4);Nodes(Elements(t,4),4)];

        %c = [Nodes(Elements(t,2),5);Nodes(Elements(t,3),5);Nodes(Elements(t,4),5)];
        c=[0.9 0.9 0.9];
        patch(x,y,z,c)
     end;

xlabel('x(m)')
ylabel('z(m)')
ylabel('y(m)')

axis equal
grid off
axis off
```

5.2 Geração de Malhas em Geometrias Tridimensionais

Nesta seção são apresentados os resultados obtidos pelo ECon-MG para geometrias tridimensionais. São apresentadas as malhas geradas em diversas geometrias tridimensionais arbitrárias compostas por faces planas, figuras (5.4), (5.5), (5.6), (5.7), (5.8), (5.9), (5.10), (5.11), (5.12), (5.13), (5.14), (5.15), (5.16), (5.17) e (5.18). Essas geometrias foram desenhadas no programa de CAD SolidEdge® e exportadas para o formato do arquivo de entrada do ECon-MG, como apresentado na seção 4.3. A partir daí o programa ECon-MG se encarrega de gerar as malhas de acordo com o valor do $MeshPar$ fornecido pelo usuário.

As malhas foram geradas em um computador Pentium 4 com 1 GB de memória RAM e *clock* de 2.6 GHz. O tempo gasto de processamento para gerar as malhas apresentadas variou aproximadamente de 3 a 5 minutos. Por exemplo, a malha mostrada na figura (5.6) precisou de 04 min e 11 seg para ser gerada. Já a malha da engrenagem apresentada pela figura (5.8) levou 03 min e 06 seg para ser gerada. Na geração de malhas muito refinadas, isto é, com uma grande quantidade de nós, pode acontecer problemas de *overflow*. Problemas de *overflow* ocorrem devido ao fato do algoritmo de varredura implementado ser de complexidade $O(n)^4$, conforme explicado na seção 4.9. De um modo geral, o programa não apresenta dificuldades em gerar malhas de até 10000 elementos. Este número de elementos, no entanto, pode variar de acordo com a forma da geometria.

Logo depois que as malhas são mostradas, uma série de tabelas, correspondentes a cada uma das malhas geradas são apresentadas para avaliação da qualidade das malhas. O critério de Ruppert foi escolhido como principal meio de avaliação das malhas geradas. Os resultados estão dispostos em tabelas levando em consideração os ângulos internos mínimos dos triângulos das malhas e suas razões de aspecto. Vide tabelas 5.1, 5.2, 5.3, 5.4, 5.5, 5.6, 5.7, 5.8, 5.9, 5.10, 5.11, 5.12, 5.13, 5.14 e 5.15.

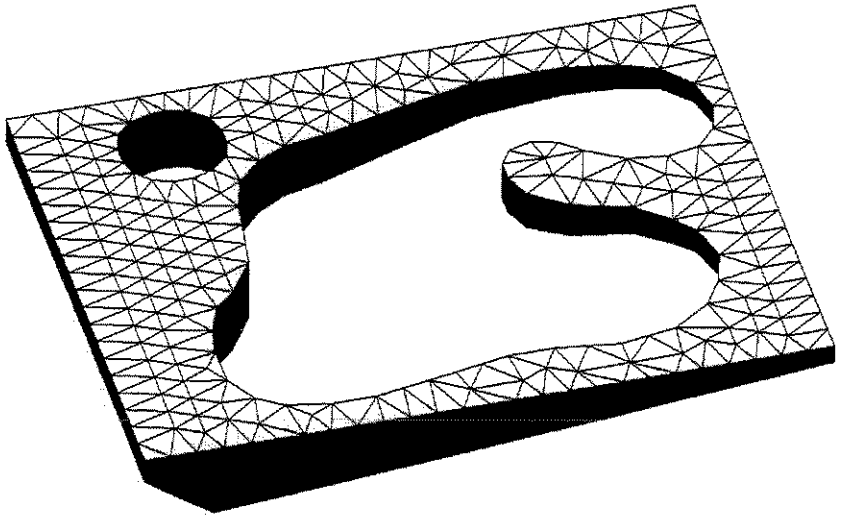


Figura 5.4: Malha 3D para elementos de contorno com 655 nós e 1314 elementos.

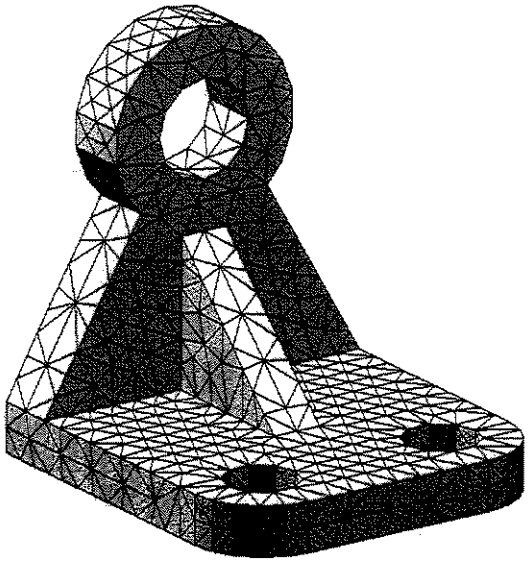


Figura 5.5: Malha 3D para elementos de contorno com 787 nós e 1582 elementos.

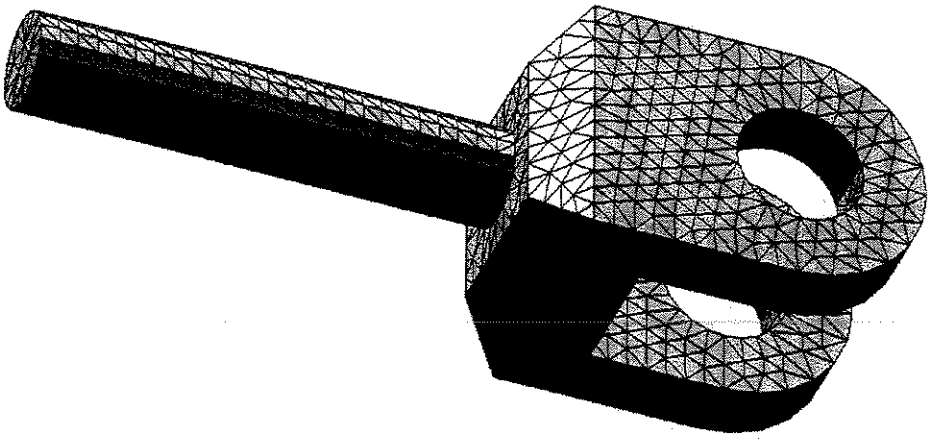


Figura 5.6: Malha 3D para elementos de contorno com 1508 nós e 3020 elementos.

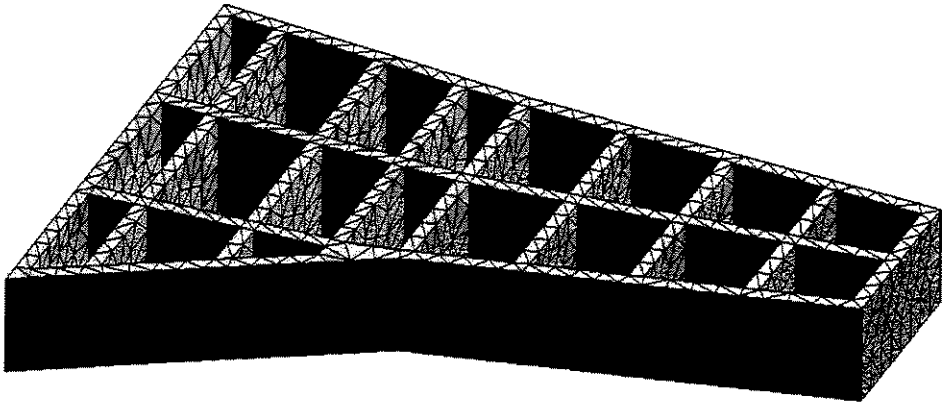


Figura 5.7: Malha 3D para elementos de contorno com 1776 nós e 3624 elementos.

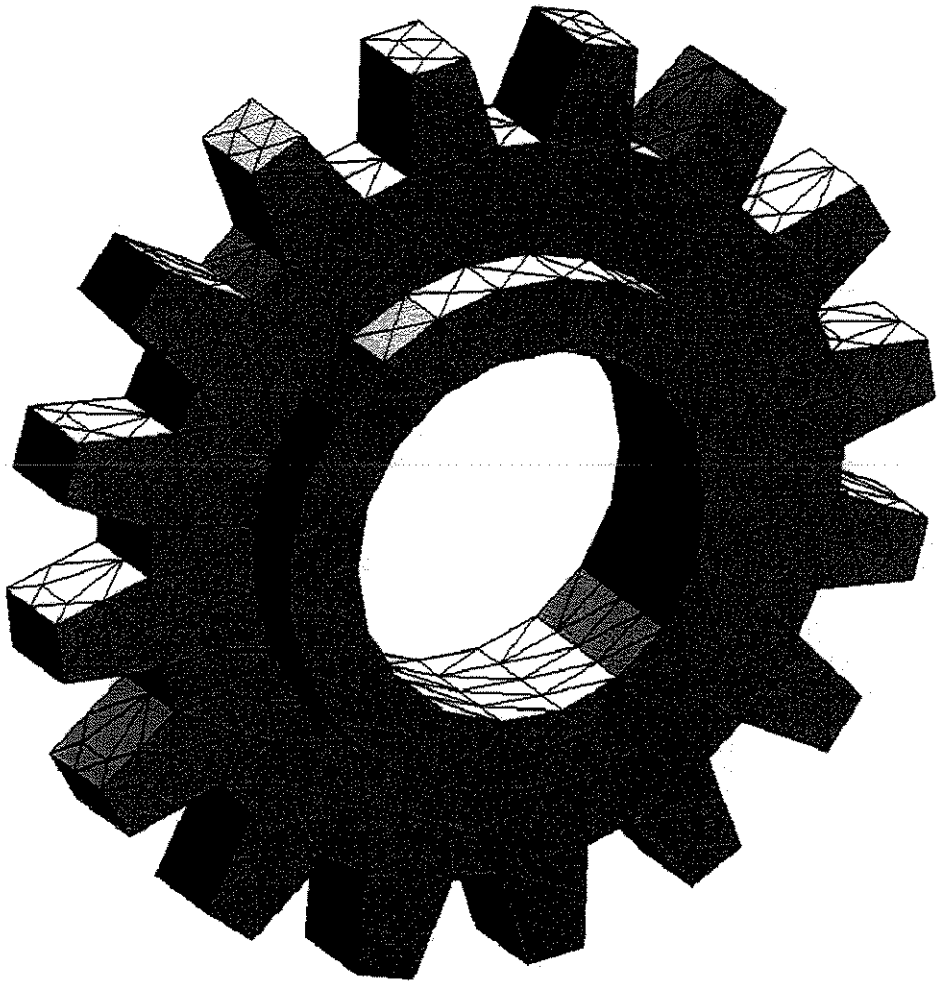


Figura 5.8: Malha 3D para elementos de contorno com 766 nós e 1532 elementos.

Pode-se perceber em todas as geometrias uma boa formação de elementos. As geometrias são arbitrárias e servem para demonstrar a capacidade do gerador em lidar com diferentes tipos de objetos. Não se teve a preocupação de modelar componentes mecânicos reais de engenharia, mas sim ilustrar diversas malhas em várias geometrias. Mesmo fazendo a modelagem das geometrias por uma composição de faces planas, percebe-se que um grande número problemas 3D com um razoável grau de complexidade podem ser modelados. Se necessário um maior refinamento da malha, dentro de um processo de regeneração por exemplo, deve-se reiniciar a geração da malha ajustando-se o parâmetro $MeshPar$ para um menor valor.

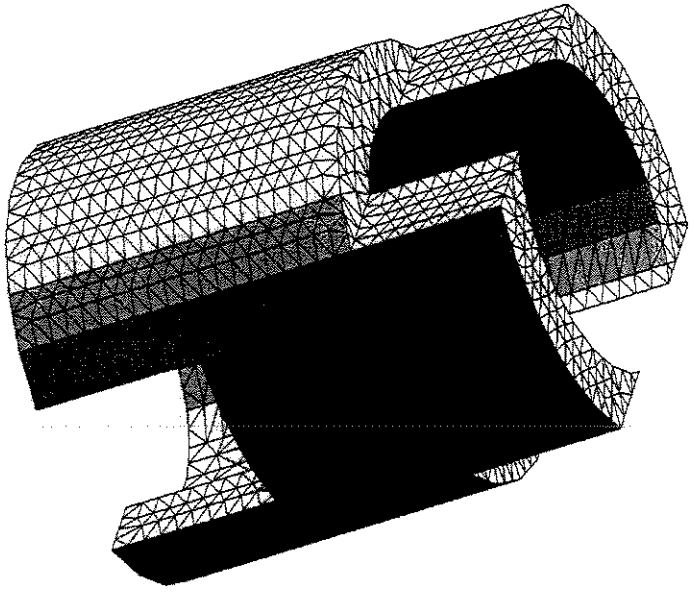


Figura 5.9: Malha 3D para elementos de contorno com 2721 nós e 5442 elementos.

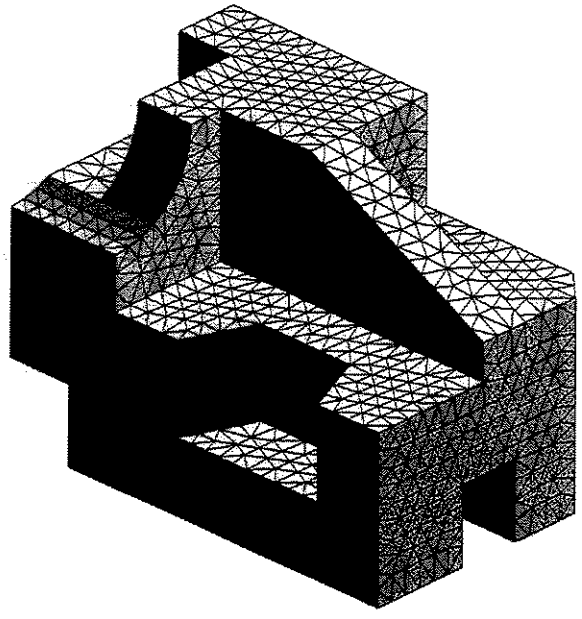


Figura 5.10: Malha 3D para elementos de contorno com 2039 nós e 4074 elementos.

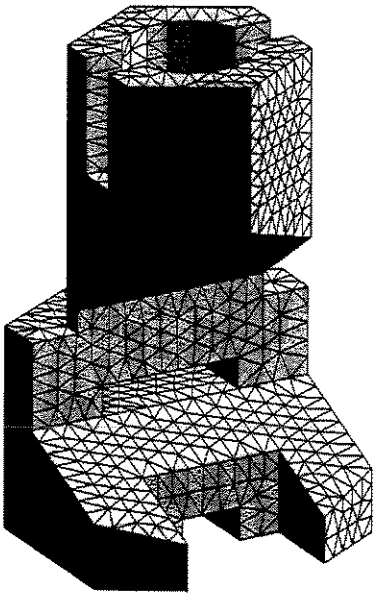


Figura 5.11: Malha 3D para elementos de contorno com 1722 nós e 3444 elementos.

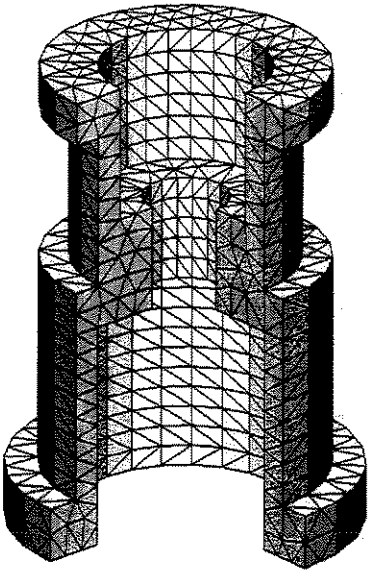


Figura 5.12: Malha 3D para elementos de contorno com 1107 nós e 2210 elementos.

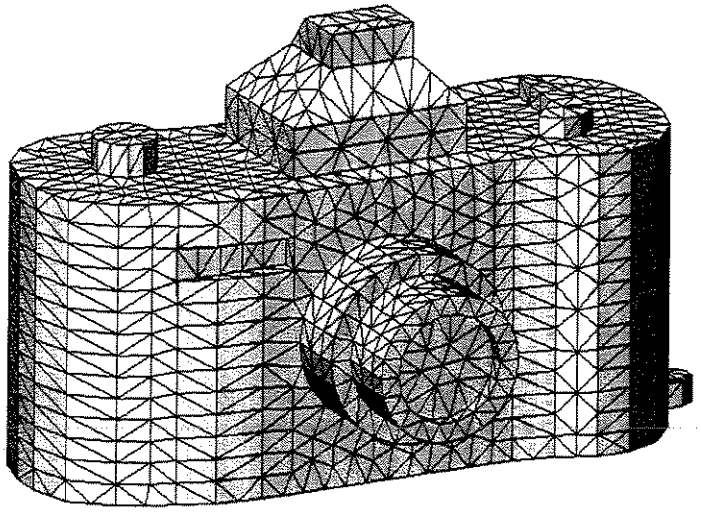


Figura 5.13: Malha 3D para elementos de contorno com 1536 nós e 3072 elementos.

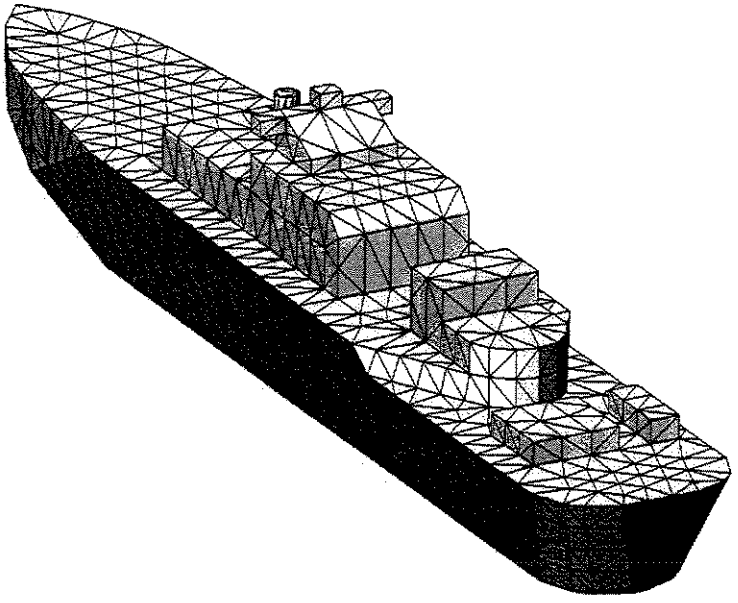


Figura 5.14: Malha 3D para elementos de contorno com 962 nós e 1920 elementos.

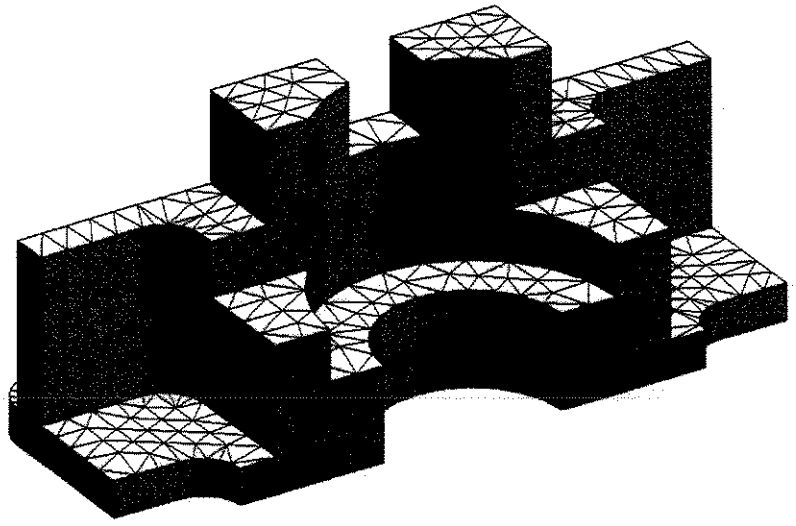


Figura 5.15: Malha 3D para elementos de contorno com 1148 nós e 2292 elementos.

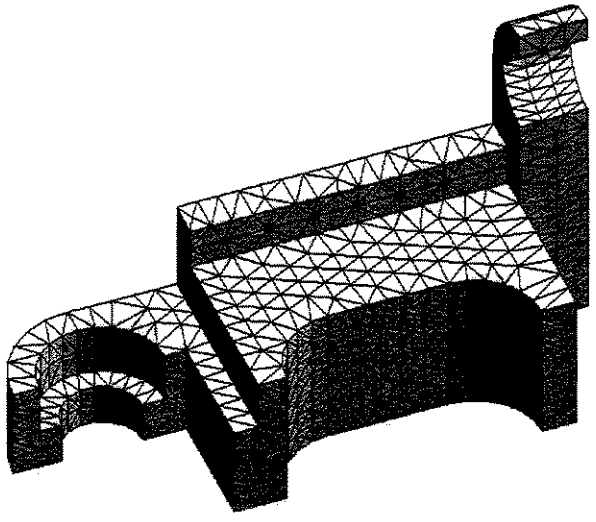


Figura 5.16: Malha 3D para elementos de contorno com 957 nós e 1910 elementos.

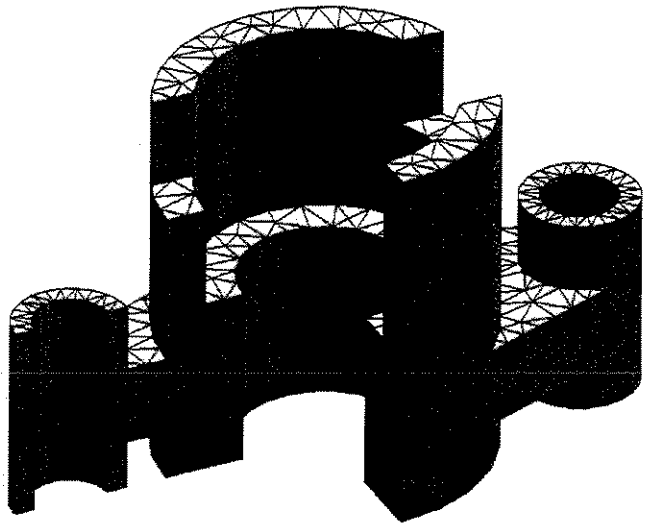


Figura 5.17: Malha 3D para elementos de contorno com 1737 nós e 3474 elementos.

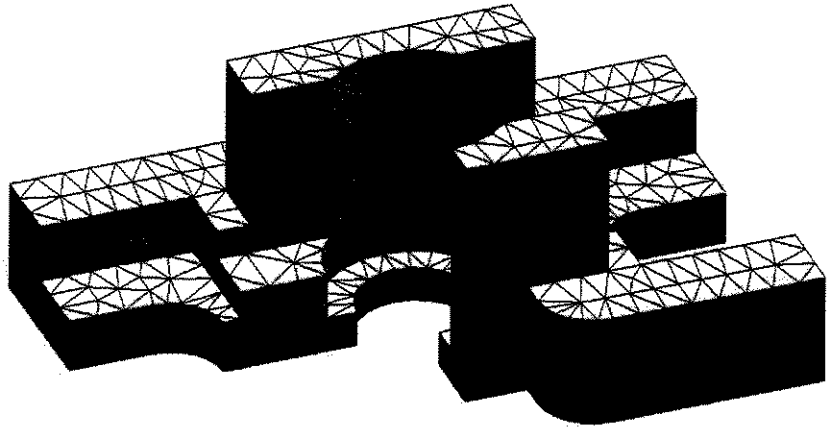


Figura 5.18: Malha 3D para elementos de contorno com 1058 nós e 2116 elementos.

5.2.1 Tabelas com Parâmetros de Qualidade da Malhas Geradas

Nesta seção são apresentadas as tabelas para avaliação da qualidade das malhas geradas. Pode-se notar que todas as malhas apresentam bons indicadores de qualidade e os ângulos mínimos encontrados, na grande maioria dos casos, possuem valores altos. Na literatura encontra-se o valor de 13° para o ângulo mínimo de uma boa malha para análise numérica. No caso específico da figura (5.13), onde o ângulo mínimo encontrado na malha foi de 1,09°, como pode ser visto na tabela 5.10, observa-se que esse ângulo é devido a própria forma da geometria. Portanto, não foi um triângulo gerado de forma ruim pelo gerador. Esse ângulo é intrínseco à geometria e fica no entalhe da parte frontal da camera, reservado para o adesivo com o logotipo do fabricante da camera. Nas demais geometrias que apresentaram ângulos mínimos abaixo de 13°, como as geometrias (5.9), (5.14) e (5.18), basta diminuir o valor do $MeshPar$ para se conseguir uma malha mais refinada e com triângulos com ângulos internos mínimos maiores.

	A0-A5	A5-A15	A15-A25	A25-A35	A35-A60	Soma
Quantidade	0	0	6	264	1044	1314
(%)	0	0	0,46	20,09	79,45	100

	ON	OFF	Soma	Ângulo Mínimo
Quantidade	1200	114	1314	22,73°
(%)	91,32	8,68	100	

Tabela 5.1: Dados para avaliação da qualidade da malha da geometria (5.4).

	A0-A5	A5-A15	A15-A25	A25-A35	A35-A60	Soma
Quantidade	0	0	5	135	1442	1582
(%)	0	0	0,32	8,53	91,15	100

	ON	OFF	Soma	Ângulo Mínimo
Quantidade	1472	110	1582	23,93°
(%)	93,05	6,95	100	

Tabela 5.2: Dados para avaliação da qualidade da malha da geometria (5.5).

	A0-A5	A5-A15	A15-A25	A25-A35	A35-A60	Soma
Quantidade	0	0	2	483	2535	3020
(%)	0	0	0,07	15,99	83,94	100

	ON	OFF	Soma	Ângulo Mínimo
Quantidade	2885	135	3020	22,11°
(%)	95,53	4,47	100	

Tabela 5.3: Dados para avaliação da qualidade da malha da geometria (5.6).

	A0-A5	A5-A15	A15-A25	A25-A35	A35-A60	Soma
Quantidade	0	0	4	1090	2530	3624
(%)	0	0	0,11	30,08	69,81	100

	ON	OFF	Soma	Ângulo Mínimo
Quantidade	3446	178	3624	23,85°
(%)	95,09	4,91	100	

Tabela 5.4: Dados para avaliação da qualidade da malha da geometria (5.7).

	A0-A5	A5-A15	A15-A25	A25-A35	A35-A60	Soma
Quantidade	0	0	4	310	1218	1532
(%)	0	0	0,26	20,23	79,50	100

	ON	OFF	Soma	Ângulo Mínimo
Quantidade	1344	188	1532	23,03°
(%)	87,73	12,27	100	

Tabela 5.5: Dados para avaliação da qualidade da malha da geometria (5.8).

	A0-A5	A5-A15	A15-A25	A25-A35	A35-A60	Soma
Quantidade	0	7	326	1779	3330	5442
(%)	0	0,13	5,99	32,69	61,19	100

	ON	OFF	Soma	Ângulo Mínimo
Quantidade	4747	695	5442	9,18°
(%)	87,23	12,77	100	

Tabela 5.6: Dados para avaliação da qualidade da malha da geometria (5.9).

	A0-A5	A5-A15	A15-A25	A25-A35	A35-A60	Soma
Quantidade	0	0	1	401	3672	4074
(%)	0	0	0,02	9,84	90,13	100

	ON	OFF	Soma	Ângulo Mínimo
Quantidade	3959	115	4074	23,86°
(%)	97,18	2,82	100	

Tabela 5.7: Dados para avaliação da qualidade da malha da geometria (5.10).

	A0-A5	A5-A15	A15-A25	A25-A35	A35-A60	Soma
Quantidade	0	0	28	507	2909	3444
(%)	0	0	0,81	14,72	84,47	100

	ON	OFF	Soma	Ângulo Mínimo
Quantidade	3235	209	3444	22,66°
(%)	93,93	6,07	100	

Tabela 5.8: Dados para avaliação da qualidade da malha da geometria (5.11).

	A0-A5	A5-A15	A15-A25	A25-A35	A35-A60	Soma
Quantidade	0	0	16	550	1644	2210
(%)	0	0	0,72	24,89	74,39	100

	ON	OFF	Soma	Ângulo Mínimo
Quantidade	2059	151	2210	17,60°
(%)	93,17	6,83	100	

Tabela 5.9: Dados para avaliação da qualidade da malha da geometria (5.12).

	A0-A5	A5-A15	A15-A25	A25-A35	A35-A60	Soma
Quantidade	4	1	97	612	2358	3072
(%)	0,13	0,03	3,16	19,92	76,76	100

	ON	OFF	Soma	Ângulo Mínimo
Quantidade	2565	507	3072	1,09°
(%)	83,50	16,50	100	

Tabela 5.10: Dados para avaliação da qualidade da malha da geometria (5.13).

	A0-A5	A5-A15	A15-A25	A25-A35	A35-A60	Soma
Quantidade	0	11	50	265	1594	1920
(%)	0	0,57	2,60	13,80	83,02	100

	ON	OFF	Soma	Ângulo Mínimo
Quantidade	1732	188	1920	7,83°
(%)	90,21	9,79	100	

Tabela 5.11: Dados para avaliação da qualidade da malha da geometria (5.14).

	A0-A5	A5-A15	A15-A25	A25-A35	A35-A60	Soma
Quantidade	0	1	91	555	1645	2292
(%)	0	0,04	3,97	24,21	71,77	100

	ON	OFF	Soma	Ângulo Mínimo
Quantidade	1976	316	2292	14,14°
(%)	86,21	13,79	100	

Tabela 5.12: Dados para avaliação da qualidade da malha da geometria (5.15).

	A0-A5	A5-A15	A15-A25	A25-A35	A35-A60	Soma
Quantidade	0	0	31	372	1507	1910
(%)	0	0	1,62	19,48	78,90	100

	ON	OFF	Soma	Ângulo Mínimo
Quantidade	1720	190	1910	16,75°
(%)	90,05	9,95	100	

Tabela 5.13: Dados para avaliação da qualidade da malha da geometria (5.16).

	A0-A5	A5-A15	A15-A25	A25-A35	A35-A60	Soma
Quantidade	0	6	734	1064	1670	3474
(%)	0	0,17	21,13	30,63	48,07	100

	ON	OFF	Soma	Ângulo Mínimo
Quantidade	2435	1039	3474	13,07°
(%)	70,09	29,91	100	

Tabela 5.14: Dados para avaliação da qualidade da malha da geometria (5.17).

	A0-A5	A5-A15	A15-A25	A25-A35	A35-A60	Soma
Quantidade	0	4	90	507	1515	2116
(%)	0	0,19	4,25	23,96	71,60	100

	ON	OFF	Soma	Ângulo Mínimo
Quantidade	1689	427	2116	5,41°
(%)	79,82	20,18	100	

Tabela 5.15: Dados para avaliação da qualidade da malha da geometria (5.18).

5.3 Análises Numéricas

Foram realizadas algumas análises numéricas, usando o *solver* em elementos de contorno E-Con3D e o *solver* em elementos finitos Ansys® a fim de se demonstrar aplicações práticas das malhas geradas. Para análise dos resultados numéricos buscou-se inicialmente a comparação dos resultados obtidos com uma solução analítica conhecida para um problema de 1/4 de anel simulando a parede de um tubo com pressão interna. Nos demais estudos de casos, os resultados obtidos foram comparados com soluções calculadas pelo Ansys®, que é um programa comercial de grande aceitação e validado por diversos *benchmarks*. Para maiores informações consultar *Ansys® Inc. Theory Reference.*

5.3.1 Primeiro Estudo de Caso

A primeira análise realizada foi um problema com uma geometria do tipo 1/4 de anel simulando a parede de um tubo com pressão interna, figura (5.19). Esse problema encontra-se resolvido em [32] e possui solução analítica. Primeiramente, gerou-se a malha no ECon-MG com 200 nós e 396 elementos. As condições de contorno aplicadas foram as seguintes:

- restrição de deslocamento na direção x dos nós da face paralela ao plano yz;
- restrição de deslocamento na direção y dos nós da face paralela ao plano xz;
- restrição de deslocamento na direção z dos nós das faces paralelas ao plano xy;
- aplicação de pressão interna de 50 unidades nos elementos da parede interna do tubo.

O coeficiente de Poisson usado foi de 0,3 e o módulo de elasticidade $E = 2600$ unidades. O mapa de deslocamentos obtido pelo *solver* E-Con3D pode ser visto na figura (5.19). Analisou-se o mesmo problema, também no E-Con3D, para uma malha gerada no Ansys® com 212 nós e 420 elementos, usando-se o elemento *shell63*, ver figura (5.20).

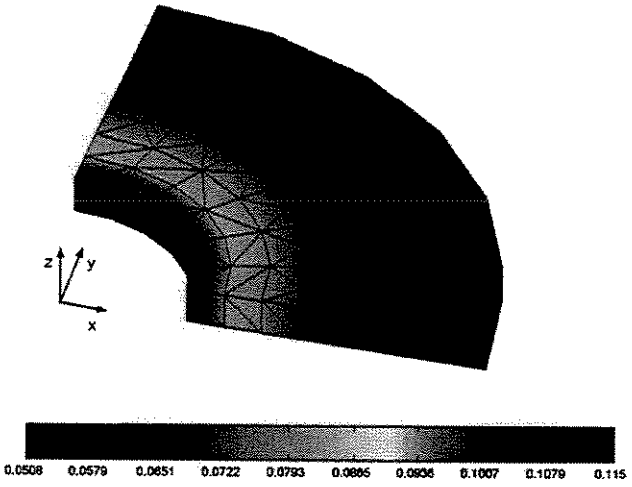


Figura 5.19: Análise usando o *solver* E-Con3D com malha gerada pelo ECon-MG.

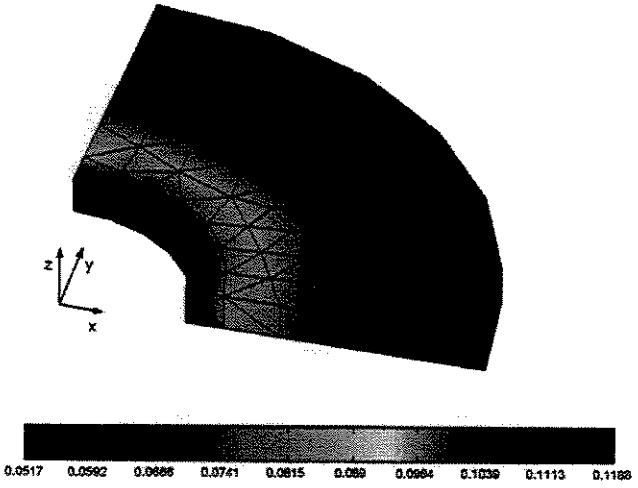


Figura 5.20: Análise usando o *solver* E-Con3D com malha gerada pelo Ansys®.

Para facilitar a comparação dos resultados traçou-se um gráfico com a variação da solução ao longo do raio do tubo, figura (5.21). O raio interno do tubo mede 4 unidades e o raio externo 12 unidades. Nota-se que ambas as soluções ficaram muito próximas da solução analítica. Os resultados obtidos com duas malhas geradas pelo ECon-MG, uma mais grosseira com 280 elementos e outra mais refinada com 396 elementos, estão apresentadas no gráfico para evidenciar a convergência dos resultados.

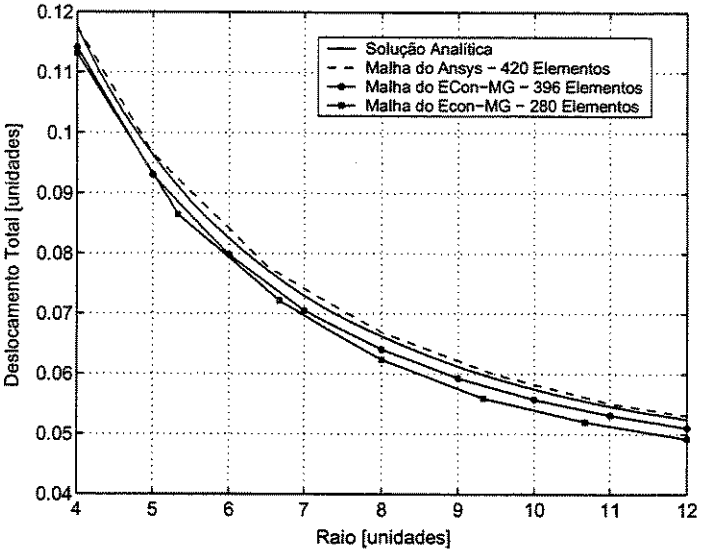


Figura 5.21: Variação da solução ao longo do raio do tubo.

5.3.2 Segundo Estudo de Caso

Um segundo estudo de caso foi realizado. Inicialmente foi gerada uma malha no Ansys® com 934 nós e 1832 elementos, usando o elemento *shell63*, para a geometria apresentada pela figura (5.22). Foi usado um coeficiente de Poisson igual a 0,21 e um módulo de elasticidade $E = 2900$ unidades. As condições de contorno aplicadas foram as seguintes:

- restrição de deslocamento nas direções x e z dos nós dos dois furos;
- restrição de deslocamento na direção y para os nós da duas faces laterais externas que contém os furos;
- aplicação de pressão de 500 unidades nos elementos da face na extremidade da haste.

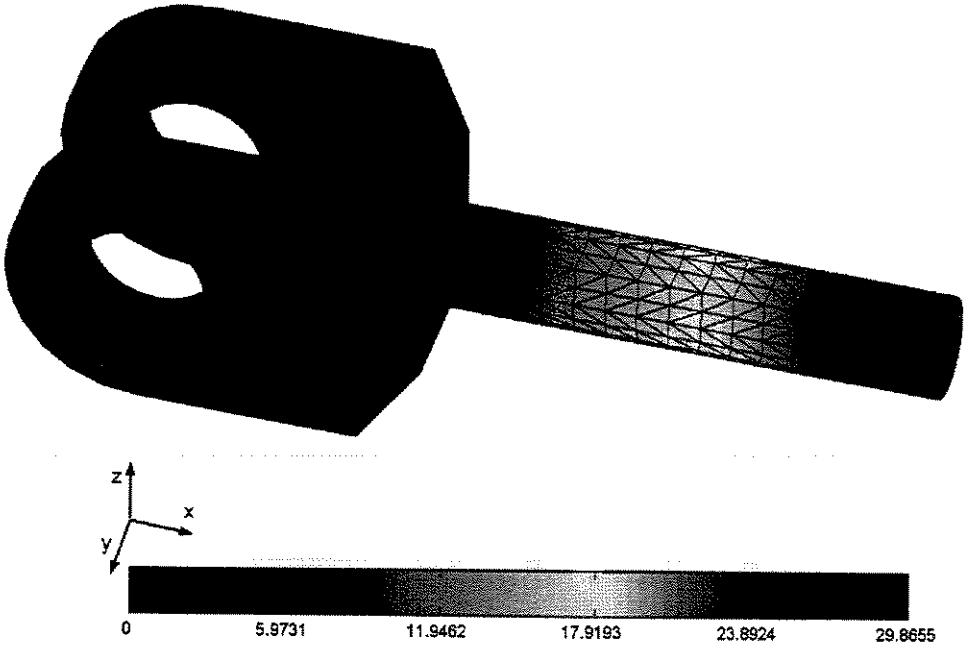


Figura 5.22: Análise usando o *solver* E-Con3D com malha gerada pelo Ansys®.

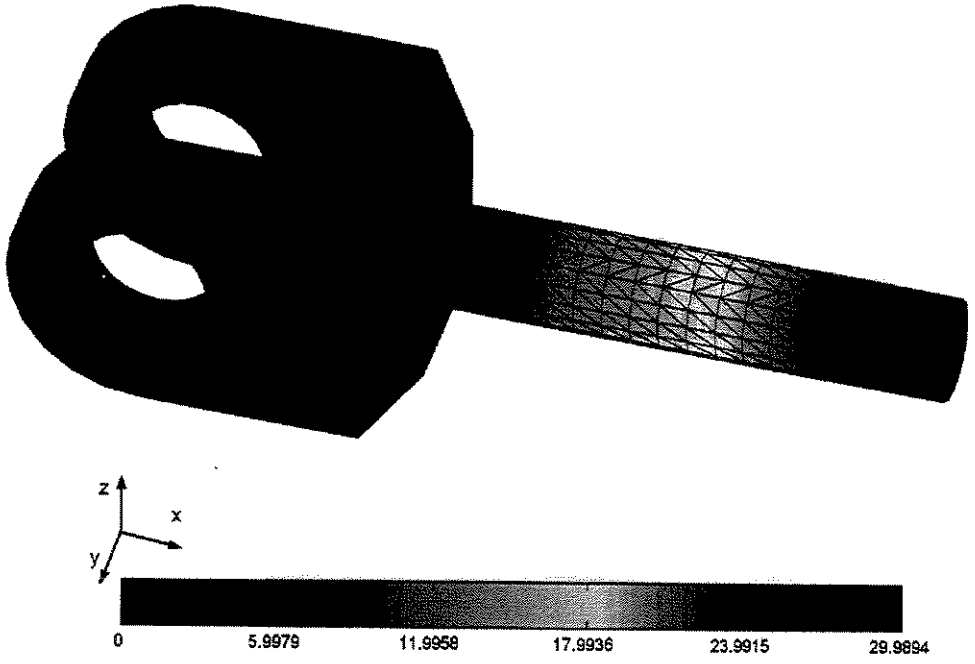


Figura 5.23: Análise usando o *solver* E-Con3D com malha gerada pelo ECon-MG.

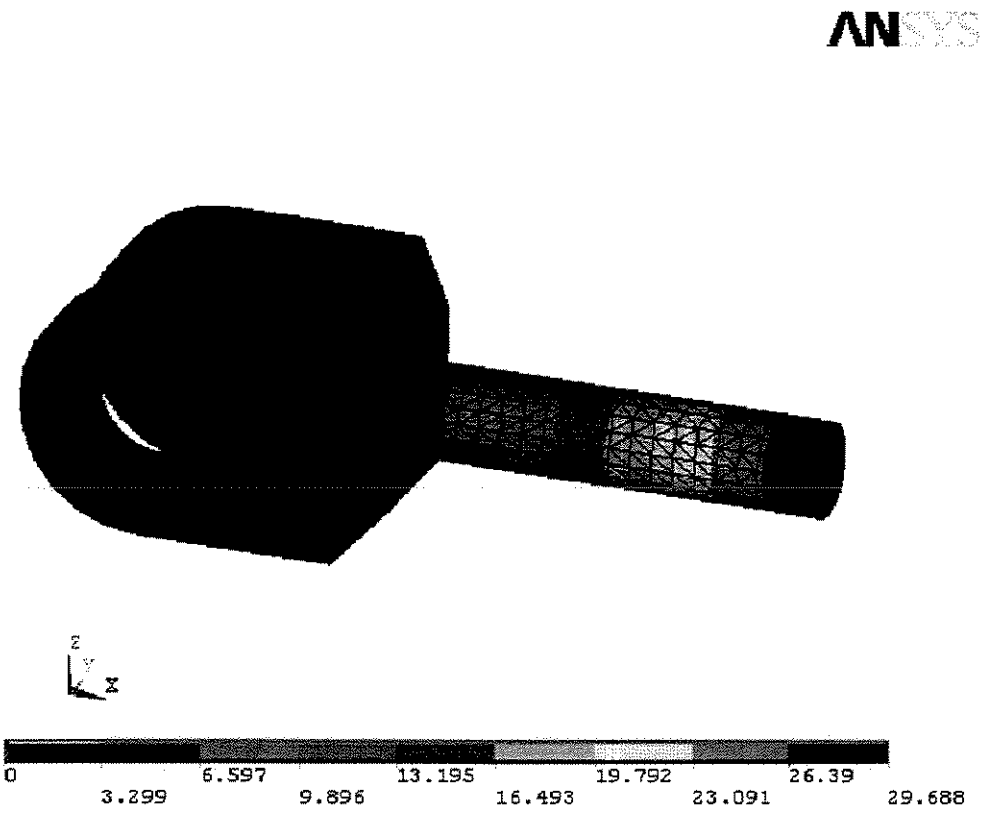


Figura 5.24: Análise usando o *solver* Ansys® com malha gerada pelo próprio Ansys®.

A solução calculada com a malha do Ansys®, pelo *solver* ECon-3D, para os deslocamentos é mostrada pela figura (5.22). A malha, como dito anteriormente, possui 934 nós e 1832 elementos. Foi feita outra análise no *solver* ECon-3D para o mesmo problema, usando uma malha gerada pelo ECon-MG com 936 nós e 1876 elementos, vide figura (5.23). Para finalizar, foi realizada uma outra análise usando elementos finitos, com elementos do tipo *solid45* no próprio Ansys®, para uma malha com 1618 nós e 6013 elementos. O resultado obtido é apresentado pela figura (5.24). Constata-se que as três análises apresentaram resultados muito parecidos. Isto leva a concluir que tanto a malha gerada pelo ECon-MG como o *solver* em elementos de contorno, E-Con3D, estão apresentando bons resultados. Foi usado o elemento *solid45* porque ele possui interpolação linear. Desse modo a comparação dos resultados fica justa pois, como já foi dito, o ECon-3D usa elementos triangulares-lineares.

5.3.3 Terceiro Estudo de Caso

Um terceiro estudo de caso foi feito. A geometria do problema é apresentada pela figura (5.25). Foi usado um coeficiente de Poisson igual 0,25 e $E = 2100$ unidades. As condições de contorno aplicadas foram as seguintes:

- restrição de deslocamento na direção z dos nós da face inferior paralela ao plano xy;
- restrição de deslocamento nas direções x e y para os nós interiores ao furos de fixação;
- pressão de 250 unidades nos elementos da metade inferior do furo do mancal.

Na figura (5.25) é apresentado o mapa dos deslocamentos calculado pelo E-Con3D a partir de uma malha de 778 nós e 1564 elementos gerada pelo ECon-MG. Na figura (5.26) é apresenta a solução calculada pelo Ansys®, com uma malha gerada pelo próprio Ansys® com 624 nós e 1846 elementos do tipo *solid45*.

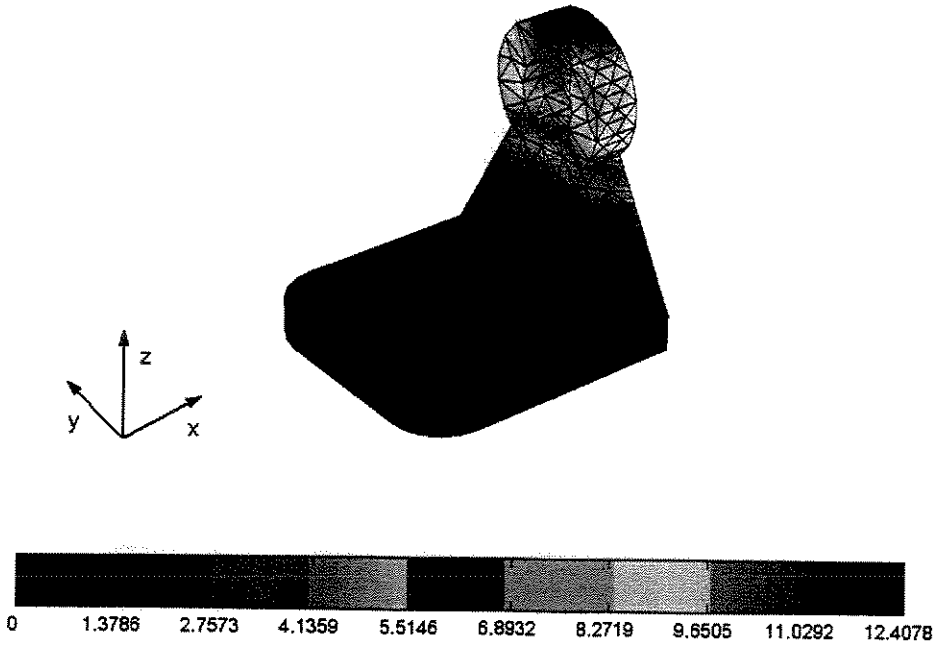


Figura 5.25: Malha gerada pelo ECon-MG com 778 nós e 1564 elementos.

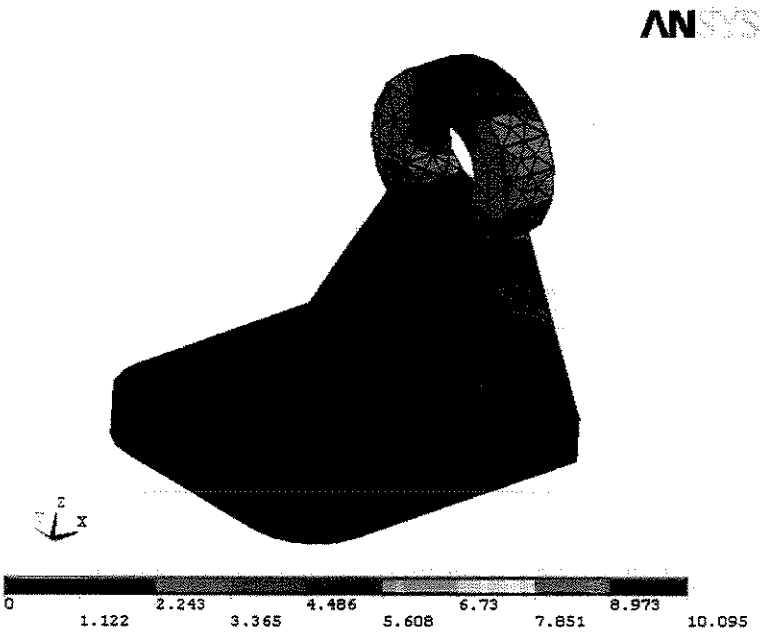


Figura 5.26: Malha do Ansys® com 624 nós e 1846 elementos, usando *solid45*.

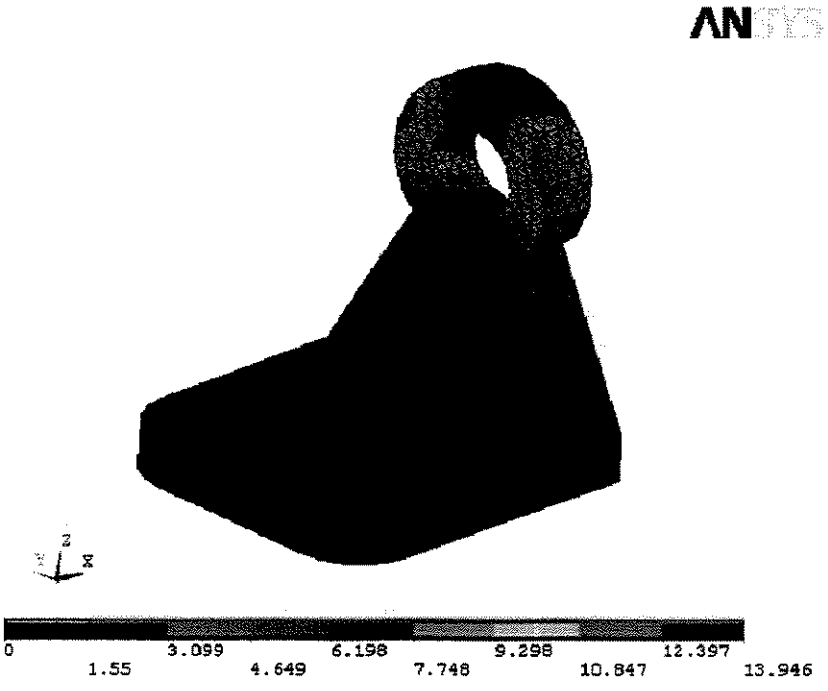


Figura 5.27: Malha do Ansys® com 11968 nós e 22525 elementos, usando *solid92*.

Comparando os resultados obtidos, verifica-se que o deslocamento máximo obtido pelo *solver* E-Con3D usando uma malha gerada pelo ECon-MG foi de 12,408 unidades, e o deslocamento máximo obtido pelo Ansys® foi de 10,095 unidades, conforme figura (5.26). Por se tratar de uma diferença significativa de valores, foi feita uma nova análise no Ansys® usando-se uma malha com 11968 nós e 22525 elementos do tipo *solid92*. O elemento *solid92* possui funções de forma com interpolação quadrática, o que faz com que a solução encontrada tenha maior credibilidade quando comparada com a análise feita com o elemento *solid45*. Uma descrição mais detalhada dos elementos implementados no Ansys®, como os elementos *shell63*, *solid45* e *solid92*, pode ser encontrada no Anexo B. Além da diferença entre os elementos, uma malha muito mais refinada foi usada. Os resultados obtidos são apresentados na figura (5.27). Verifica-se que o deslocamento máximo encontrado foi de 13,946 unidades, o que leva a concluir que, a solução apresentada pelo solver E-Con3D, usando a malha gerada pelo ECon-MG, representou melhor o comportamento físico real do problema quando comparado com a análise feita pelo Ansys® com o elemento *solid45*. Isto nos faz acreditar que nossas ferramentas estão apresentando bons resultados quando comparadas com outras ferramentas já consagradas, como o Ansys® e, principalmente, que estamos no caminho certo.

Capítulo 6

Conclusões e Sugestões para Próximos Trabalhos

6.1 Conclusões

O gerador de malhas desenvolvido neste trabalho mostrou-se eficiente para a geração de malhas em geometrias tridimensionais arbitrárias compostas por faces planas. Devido a programação ter sido feita usando o paradigma da orientação a objetos, foi possível construir a estrutura do programa de forma bastante clara e eficiente. Além disso, essa estrutura é extremamente flexível e permite a expansão e o aprimoramento do código de maneira mais fácil e rápida.

Em todas as geometrias analisadas pelo gerador ECon-MG pôde-se gerar a malha de forma adequada. Em algumas situações foi necessário fazer um reajuste do valor do $MeshPar$ até se conseguir uma malha com qualidade satisfatória para se realizar uma análise numérica. Apesar do gerador somente aceitar geometrias compostas por faces planas, pode-se constatar, pela análise dos resultados obtidos, que uma grande variedade de problemas tridimensionais pode ser modelada com uma complexidade de forma considerável.

O procedimento empregado na geração de elementos assegura a formação de triângulos com bons parâmetros de qualidade, como pôde ser visto no Capítulo 5. O *grid* de pontos, gerado de forma alternada, já contribui para a maximização dos ângulos internos dos triângulos. Além disso, vários métodos de verificação foram implementados para se impedir

a geração de elementos sobrepostos ou fora do contorno da geometria. Na grande maioria das geometrias constatou-se que os ângulos mínimos encontrados nas malhas possuiam um valor bastante elevado, geralmente em torno de 20°. Em algumas geometrias ocorreu a geração de ângulos mínimos com valores muito baixos. Isso, entretanto, deve-se ao fato da própria geometria possuir ângulos muito pequenos e não uma falha do gerador. Nas malhas de algumas outras geometrias verificou-se a presença de ângulos em torno de 10°, que é um valor bastante razoável, mas esse valor de ângulo mínimo pode ser facilmente aumentado reduzindo o valor do $MeshPar$, caso necessário.

Por possuir o código fonte aberto e ter sido desenvolvido usando uma codificação orientada a objetos, os métodos e classes já desenvolvidos neste trabalho, podem ser reaproveitados ou alterados da forma que melhor for conveniente. Isso quer dizer que um novo método para a geração do *grid* de pontos pode ser desenvolvido para substituir aquele que está implementado atualmente e, pode-se, ainda, modificar o processo de escolha de três pontos do *grid* de pontos para se verificar a propriedade do circuncírculo vazio de Delaunay.

O algoritmo proposto neste trabalho, baseado no uso de matrizes de transformação geométrica, possui vários aspectos positivos com relação a simplicidade, eficiência e robustez. O código fica bastante simplificado, uma vez que, o problema é transformado do espaço tridimensional para o espaço bidimensional. Os algoritmos e métodos para geração dos elementos são bidimensionais, ou seja, são mais simples, confiáveis e fáceis de serem implementados. Com relação a eficiência e robustez, como cada face da geometria tridimensional é deslocada para o espaço bidimensional e a malha é gerada sobre uma face por vez, a possibilidade de ocorrer uma formação de elementos ruins é minimizada. Como a malha final é construída face a face, o problema é subdividido em vários problemas menores e o algoritmo principal assume o paradigma de dividir-para-conquistar, o que assegura alta eficiência e velocidade de processamento computacional.

Pelas análises numéricas realizadas verifica-se que a malha influencia de maneira extremamente significativa no resultado numérico da solução do problema físico em estudo. Portanto, é desejável que a malha seja gerada de forma adaptativa, ou seja, mais refinada nos locais de maior concentração de solução e menos refinada nos demais outros locais. As malhas geradas pelo ECon-MG, apesar de não serem geradas de forma adaptativa, mostraram bons resultados quando usadas em conjunto com o *solver* E-Con3D e comparadas com as malhas geradas pelo Ansys® analisadas por elementos finitos.

Por fim, é importante mencionar que investir no desenvolvimento de códigos para realizar a fase de pré-processamento de uma análise numérica é uma tarefa bastante árdua e custosa. Especialmente conhecendo as capacidades de modelagem e exibição de resultados dos programas de análise numérica disponíveis atualmente no mercado. Entretanto, o importante a ser ressaltado é o fato desse trabalho contribuir para o desenvolvimento de uma estrutura computacional com o código fonte aberto que destina-se a realizar análises numéricas usando o método dos elementos de contorno. Com isso, objetiva-se, no futuro, dispor-se de uma plataforma computacional de análise numérica que permita a implementação de novas formulações e o direcionamento das pesquisas para o estudo de problemas que não podem ser analisados pelos pacotes numéricos disponíveis no mercado. Neste estágio é que poderão ser obtidos os melhores resultados sob a óptica de avanços científicos.

6.2 Publicações

As publicações originadas a partir desse trabalho de mestrado foram as seguintes: um artigo publicado no *XXIV Cilamce* [12] e um resumo publicado no *7th US National Congress on Computational Mechanics, 4th Symposium on Trends in Unstructured Mesh Generation* [13]. Estão submetidos dois outros artigos: um para o *XXV Cilamce* [14] e outro para a revista internacional indexada *Advances in Engineering Software* [15].

6.3 Sugestões para Trabalhos Futuros

Da forma que o programa foi desenvolvido, usando o paradigma da orientação a objetos e, dada a importância de um gerador de malhas para o desenvolvimento de análises numéricas, especialmente, no contexto do grupo de pesquisa ao qual ele se insere, uma grande variedade de opções de trabalhos futuros aparecem. Pode-se transformar o programa de geração de malhas ECon-MG, aproveitando-se toda a estrutura desenvolvida, em um gerador extremamente eficiente com relação ao tempo de processamento e a qualidade dos elementos gerados. Para isso, aconselho a eliminação da geração do *grid* de pontos e a substituição do algoritmo de triangulação de Delaunay de complexidade $O(n)^4$. Recomendo a implementação do método de triangulação de Delaunay por avanço de fronte, a fim de maximizar os ângulos internos mínimos dos triângulos da malha e aumentar de forma extremamente significativa a eficiência do programa [20, 37]. Desse modo não há necessidade de se gerar um *grid* de pontos e a geração dos nós e elementos ocorreria iterativamente. Isso modificaria a complexidade do algoritmo de geração dos elementos para $O(n)^2$ no pior dos casos, o que já muito bem aceitável quando comparado, até mesmo, com geradores comerciais.

Com relação as dificuldades que não foram superadas, mas sim contornadas, vale destacar a não implementação de um interpretador IGES para leitura dos arquivos de geometria desenhados em um programa CAD e, sua respectiva tradução para um arquivo com a formatação aceita pelo ECon-MG. Por isso, ainda, usa-se o interpretador IGES do Ansys® para geração dos arquivos de entrada para o ECon-MG, como foi explicado na seção 4.3. Fica aqui então a sugestão para implementação futura. Outro ponto importante está relacionado ao fato do módulo de visualização estar desvinculado das classes de geração de malha. Isso deve-se ao fato de não se ter conseguido compilar a biblioteca NewMat10 para cálculo matricial, que possuiu vinculação estática (*.lib*), em um projeto *MFC AppWizard (exe)* onde está o módulo de visualização gráfica pela biblioteca OpenGL®. Deve-se investir ainda neste sentido de forma a criar um projeto único e integrado, pois isso traz grandes benefícios do ponto de vista de robustez e praticidade.

Uma questão interessante que surgiu durante o desenvolvimento deste trabalho e que não foi investigada mais aprofundadamente foi a questão de ordenação dos números dos nós nas malhas. Neste trabalho não se teve preocupação em implementar algoritmos para definir a ordenação dos números dos nós presentes na malha. Sabe-se que em elementos finitos a ordenação dos números dos nós é muito importante, pois proporciona uma matriz global melhor condicionada e com os valores predominantemente na diagonal principal. Isso é muito benéfico para elementos finitos pois proporciona resultados numéricos da solução do problema de forma mais precisa. Entretanto, ficou a dúvida se para análises feitas por elementos de contorno uma melhor ordenação dos números dos nós da malha traria benefícios do ponto de vista de precisão de resultados. Isto é dito porque em elementos de contorno a matriz global é totalmente cheia e não possui a diagonal principal predominante. Seria, portanto, bastante interessante como trabalho futuro implementar algum método de ordenação dos números dos nós das malhas no gerador desenvolvido para verificar se ocorre uma melhora nos resultados obtidos pelas análises numéricas feitas por elementos de contorno.

Por possuir a estrutura do programa na forma de classes e objetos modelados visualmente no Rational Rose® abre-se a perspectiva de novas implementações e mudanças de foco do gerador. Um dos objetivos futuros pode ser o tratamento de arquivos STL. Esses arquivos nada mais são que malhas compostas por elementos triangulares geradas pelo processo de esterolitografia [3]. Essas malhas geralmente modelam ossos e outros componentes do corpo humano. Entretanto, essas malhas não são apropriadas para análises numéricas pois os triângulos da malha são gerados sem nenhum controle de qualidade e via de regra possuem parâmetros de qualidade muito ruins. Usando-se arquivos STL abre-se a perspectiva de se analisar numericamente não somente problemas de engenharia, mas ossos e outras partes do corpo humano [6]. É uma questão de implementar um algoritmo tridimensional para melhorar a qualidade dos triângulos contidos nas malhas dos arquivos STL. Esta é uma área de pesquisa bastante recente em geração de malha e que tem recebido muita atenção devido sua importância.

Referências Bibliográficas

[1] Albuquerque, E. L., Sollero, P. and Aliabadi, M. H. (2002). *The boundary element method applied to time dependent problem in anisotropic material.* Int. Journal of Solids and Structures, 39: 104-1422.

[2] Bastian, M. and Li, B. Q. (2003). *An efficient automatic mesh generator for quadrilateral elements implemented using C++.* Finite Elements in Analysis and Design, 39: 905-930.

[3] Béchet, E., Cuilliere, J.-C. and Trochu, F. (2002). *Generation of a finite element mesh from stereolithography (STL) files.* Computer-Aided Design, 34: 1-17.

[4] Betti, E. (1872). *Teoria dell´ elasticità. Il Nuovo Cimento.* (Ser.2), tt. 7 & 8.

[5] Brebbia, C. A. and Dominguez, J. *Boundary Elements: An Introductory Course.* Boston: Computational Mechanics Publications - USA, 1989, 322p.

[6] Britton, J. R., Walsh, L. A. and Prendergast, P. J. (2003). *Mechanical simulation of muscle loading on the proximal femur: analysis of cemented femoral component migration with and without muscle loading.* Clinical Biomechanics, 18: 637-646.

[7] Castillo, J. E. and Otto, J. S. (1997). *A generalized length strategy for direct optimization in planar grid generation.* Mathematics and Computers in Simulation, 144: 441-456.

[8] Castillo, J. E. and Otto, J. S. (1999). *A pratical guide to direct optimization for planar grid-generation.* Computers and Mathematics with Applications, 37: 123-156.

[9] Chen, W. and Saleeb, A. F. *Constitutive Equations for Engineering Materials.* New York: Elsevier Science - USA, 1994, 1096p.

[10] Cheng, J. H., Finnigan, P. M., Hathaway, A. F., Kela, A. and Schoeder, W. J. (1988). *Quadtree/octree meshing with adaptive analysis.* Numerical grid generation in computaional fluid mechanics'88, Miami, USA.

[11] Cook, W. A. (1974). *Body oriented coordinates for generating 3-dimensional meshes.* International Journal for Numerical Methods in Engineering, 8: 27-43.

[12] Creci, G., Sollero, P., Albuquerque, E. L. (2003). *Surface mesh generation and visualization for boundary element method.* Cilamce XXIV, ISBN Proceedings CD: 85-288-0040-7.

[13] Creci, G., Albuquerque, E. L., Sollero, P., Ferreira, J. V. and Pavanello, R. (2003). *Boundary element mesh generation for modeling crack problems.* 7th US National Congress on Computational Mechanics, 4th Symposium on Trends in Unstructured Mesh Generation - Part I.

[14] Creci, G., Sollero, P., Albuquerque, E. L. and Noritomi, P. Y. (2004). *Three-dimensional mesh generation for the boundary element method.* Submitted to Proceedings of XXV Cilamce, Refice - Brazil.

[15] Creci, G., Albuquerque, E. L., Sollero, P. and Noritomi, P. Y. (2004). *A simplified mesh generation scheme for three-dimensional boundary element method.* Preprint submitted to Advances in Engineering and Software - Elsevier Science.

[16] Delaunay, B. (1934). *Sur la sphère vide.* Bull. Acad. Sci. USSR VII: Class. Sci. Mat. Nat., 7: 793-800.

[17] Desai, C. S. and Abel, J. F. *Introduction to Finite Element Method.* New York: Van Nostrand Reinhold - USA, 1972, 477p.

[18] Du, C. (1996). *An algorithm for automatic Delaunay triangulation of arbitrary planar domains.* Advances in Engineering Software, 27: 21-26.

[19] Eckel, B. *C++ Inside & Out.* Berkley: Osborne McGraw-Hill - USA, 1993, 754p.

[20] El-Hamalawi, A. (2004). *A 2D combined advancing front-Delaunay mesh generation scheme.* Finite Elements in Analysis and Design, 40: 967-989.

[21] Escobar, J. M. and Montenegro, R. (1996). *Several aspects of three-dimensional Delaunay triangulation.* Advances in Engineering Software, 27:27-39.

[22] Fenner, R. T. *Egineering Elasticity - Application of Numerical and Analytical Techniques.* New York: Ellis Horwood Limited, John Wiley & Sons - USA, 1986, 446p.

[23] Figueiredo, L. H. and Carvalho, P. C. P. *Introdução a Geometria Computacional.* Rio de Janeiro: IMPA - Brasil, 1991, 111p.

[24] Foley, J. D., van Dam, A., Feiner, S. K. and Hughes, J. F. *Computer Graphics: Principles and Practice in C.* New York: Addison-Wesley Publishing Company - USA, 1996, 1175p.

[25] George, P. L., Hecht, F. and Saltel, F. (1990). *Automatic 3D mesh generation with prescribed meshed boudaries.* IEEE Transactions on Magnetics, 26: 771-774.

[26] George, P. L. *Automatic Mesh Generation: Application to Finite Element Methods.* New York: John Wiley & Sons - USA, 1991, 348p.

[27] George, P. L. and Seveno, E. (1994). *The advancing-front mesh generation method revisited.* International Journal for Numerical Methods in Engineering, 37: 3605-3619.

[28] George, P. L., Borouchaki, H. and Laug, P. (2002). *An efficient algorithm for 3D adaptive meshing.* Advances in Engineering Software, 33: 377-387.

[29] Gordon, W. J. and Hall, C. A. (1973). *Construction of curvilinear coordinate systems and applications to mesh generation.* International Journal for Numerical Methods in Engineering, 7: 461-477.

[30] Gyimóthy, S., Vágvölgyi, A. and Sebestyén, I. (2000). *Boundary refinement in n-dimensional Delaunay meshing.* IEEE Transactions on Magnetics, 36: 1569-1573.

[31] Hirsch, M. W. *Differential Topology.* New York: Springer - USA, 1976, 221p.

[32] Kane, J. H. *Boundary Element Analysis in Engineering Continuum Mechanics.* New Jersey: Prentice Hall, Clarkson University - USA, 1994, 676p.

[33] Kupradze, V. D. *Potential Methods in the Theory of Elasticity.* Jerusalem: Israel Program for Scientific Translations - Israel, 1965.

[34] Lee, J.-F. and Dyczij-Edlinger, R. (1997). *Automatic mesh generation using a modified Delaunay tesselation.* IEEE Antennas and Propagation Magazine, 39: 34-45.

[35] Lee, K.-Y., Kim, I.-I., Cho, D.-Y. and Kim, T.-w. (2003). *An algorithm for automatic 2D quadrilateral mesh generation with line constraints.* Computer-Aided Design, 35: 1055-1068.

[36] Liberty, J. *Teach Yourself C++ in 24 Hours.* New York: Sams Publishing, 1999, 912p.

[37] Mavriplis, D. J. (1995). *An advancing front Delaunay triangulation algorithm designed for robustness.* Journal of Computational Physics, 117: 90-101.

[38] Mi, Y. *Three-Dimensional Analysis of Crack Growth.* Southampton: Computational Mechanics Publications - UK, 1996, 184p.

[39] Moaveni, S. *Finite Element Analysis - Theory and Applications with Ansys.* New Jersey: Prentice Hall, Upper Saddle River - USA, 1999, 527p.

[40] Muskhelishvili, N. I. *Singular Integral Equations.* Groningen: Noordhoff - Holland, 1953.

[41] Nicolas, A. and Nicolas, L. (1990). *3D Automatic mesh generation for boundary element method.* IEEE Transactions on Magnetics, 26: 767-770.

[42] Noritomi, P. Y., Lourenço, F. A. G. and Sollero, P. (2000). *Boundary element programs applied to stress analysis of bones.* Advances in Comput. Eng. & Sciences, (S. N. Atlury e F. W. Burst, Eds), Proc. Int. Conf., Los Angeles, 2000, 1:39-44.

[43] Noritomi, Pedro Yoshito. *Desenvolvimento de um programa base em elementos de contorno para aplicações a analises mecanicas tridimensionais em bioengenharia.* Campinas: Universidade Estadual de Campinas, SP - Brasil, 2000, 145p. Tese (Mestrado).

[44] O'Rourke, J. *Computational Geometry in C.* New York: Cambridge University Press - USA, 1993, 346p.

[45] Paiva, W. P., Sollero, P. and Albuquerque, E. L. (2002). *Analysis of the fundamental solution for anisotropic thin plates.* 15th ASCE Engineering Mechanics Conference.

[46] Phongthanapanich, S. and Dechaumphai, P. (2004). *Adaptive triangulation with object-oriented programming for crack propagation analysis.* Finite Elements in Analysis and Design;In Press, Corrected Proof Available Online.

[47] Preparata, F. P. and Shamos, M. I. *Computational Geometry: An Introduction.* New York: Springer-Verlag - USA,1985, 348p.

[48] Qiang, D. and Gunzburger, M. (2002). *Grid generation and optimization based on centroidal Voronoi tessellations.* Applied Mathematics and Computation, 133: 591-607.

[49] Quatrani, T. *Visual Modeling with Rational Rose 2002 and UML.* Boston: Addison Wesley - USA, 2002, 288p.

[50] Ruppert, J. (1995). *A Delaunay refinement algorithm for quality 2-dimensional mesh generation.* Journal of Algoritmhs, 18: 548-585.

[51] Secchi, S. and Simoni, L. (2003). *An improved procedure for 2D unstructured Delaunay mesh generation.* Advances in Engineering Software, 34: 217-234.

[52] Shewchuk, J. R. (2002). *Delaunay refinement algorithms for triangular mesh generation.* Computational Geometry-Theory and Applications, 22: 21-74.

[53] Somigliana, C. (1885). *Sopra l'equilibrio di un corpo elastico isotropo.* Il Nuovo Cimento. (Ser.3), tt. 17-20.

[54] Thompson, J. F. (1982). *Numerical grids generation.* Applied Mathematics and Computation, 10-11.

[55] Thompson, J. F., Warsi, Z. U. A. and Mastin, C. W. *Numerical Grid Generation, Foundations and Applications.* New York: North-Holland - USA, 1985, 483p.

[56] Tianyu, Lu. *Triangular mesh generation for surfaces of 3D objects.* Hawaii: University of Hawaii - USA, 2000, 173p. Thesis (Doctorate).

[57] Trevelyan, J. *Boundary Elements for Engineers - Theory and Applications.* Southampton: Computational Mechanics Inc. - UK, 1994, 211p.

[58] Tsuboi, H., Shimotsukasa, T. and Kato, K. (1990). *Triangular mesh generation using knowledge base for three-dimensional boundary element method.* IEEE Transactions on Magnetics, 26: 799-802.

[59] Tsuboi, H., Asahara, T., Kobayashi, F. and Misaki, T. (1998). *Adaptive triangular mesh generation for boundary element method in three-dimensional electrostatic problems.* IEEE Transactions on Magnetics, 34: 3379-3382.

[60] Veras, A. D. *Desenvolvimento de uma interface gráfica para aplicativos de análise estrutural.* Relatório de Iniciação Científica - FAPESP, 2001.

[61] Voronöi, M. G. (1908). *Nouvelles applications des paramètres continus à la théorie des formes quadratiques.* J. Reine Angew. Math., 134: 198-287.

[62] Woo, M., Neider, J., Davis, T. and Shreiner, D. *OpenGL Programming Guide: the Official Guide to Learning OpenGL.* New York: Addison-Wesley Publishing Company - USA, 1999, 730p.

[63] Zienkiewicz, O. C. *The Finite Element Method in Engineering and Science.* London: McGraw-Hill - UK, 1971, 521p.

Anexo A

OpenGL® e MS Visual C++

O objetivo deste anexo é mostrar como criar uma aplicação gráfica usando o compilador Microsoft Visual C++® e a biblioteca gráfica OpenGL®. Primeiramente será demonstrado como configurar os requisitos mínimos necessários para um aplicativo Windows® exibir gráficos OpenGL®. Como um dispositivo de interface gráfica (GDI) necessita de um contexto de dispositivo (DC) para desenhar imagens, o OpenGL® necessita de um contexto de renderização (RC). Um contexto de dispositivo (DC) nada mais é que uma estrutura que define um conjunto de objetos gráficos e seus respectivos atributos, que são passados para um dispositivo de interface gráfica (GDI). O OpenGL® usa o conceito de contexto de renderização (RC). Uma vez que o contexto de renderização está ativo em um *thread*, todas as chamadas em OpenGL® neste *thread* usarão esse mesmo contexto de renderização. Enquanto que múltiplos contextos de renderização podem ser usados para desenhar em uma janela única, somente um contexto de renderização pode estar ativo em um único *thread*.

Inicialmente será criado e tornado ativo um contexto de renderização do OpenGL®. Existem três passos para criar e tornar ativo um contexto de renderização:

- Configurar o formato do *pixel* da janela;
- Criar o contexto de renderização;
- Ativar o contexto de renderização.

Siga os seguintes passos para criar o projeto:

1. Crie um novo projeto do tipo *MFC AppWizard (exe)*. Selecione o diretório onde você deseja que seu projeto seja criado e digite um nome para o projeto, por exemplo "GLInterface". Aperte o botão *OK*. Em seguida será apresentada uma lista de passos e parâmetros que você deve escolher. Os parâmetros não listados são opcionais;

2. Escolha *Single Document Interface*;

3. Suporte a banco de dados: *None*;

4. *Compond Document Support: None*;

5. *Docking Toolbar: ON* (opcional);
   Initial Status Bar: ON (opcional);
   Printing and Preview: ON (A impressão de imagens OpenGL® é realizada criando um RC usando um modo de impressão DC);
   Context Sentive Help: ON (opcional);
   3D Controls: ON (opcional);

6. Use o estilo padrão de projeto MFC;
   Gere comentários nos arquivos de código fonte;
   Use a biblioteca MFC como uma DLL compartilhada;

7. Mantenha tudo como *default*;
   Aperte *Finish.*

Confira tudo na caixa de diálogo *New Project Information* e aperte *OK*. O novo projeto será criado dentro do diretório GLInterface. Primeiro é necessário incluir os arquivos OpenGL® neste projeto. Selecione *Project Settings* no menu e escolha a pasta *Link*. Escolha a categoria *General* e entre com o seguinte na caixa de edição *Object/Library modules*: "opengl32.lib glu32.lib glaux.lib". Aperte *OK*. Abra o aquivo "stdafx.h" e insira as seguintes linhas:

```
#define VC_EXTRALEAN                    // Exclude rarely-used stuff from  Windows headers

#include <afxwin.h>                     // MFC core and standard components
#include <afxext.h>                     // MFC extensions
#include <gl\gl.h>
#include <gl\glu.h>

#include <afxdisp.h>                    // MFC Automation classes
#include <afxdtctl.h>                   // MFC support for Internet Explorer 4 Common Controls
#ifndef _AFX_NO_AFXCMN_SUPPORT

#include <afxcmn.h>                     // MFC support for Windows Common Controls
#endif                                  // _AFX_NO_AFXCMN_SUPPORT
```

OpenGL® precisa que a janela possua estilo WS_CLIPCHILDREN and WS_CLIPSIBLINGS configurados. Edite o método PreCreateWindow da classe CGLInterfaceView de modo a ficar como o seguinte:

```
BOOL CGLInterfaceView::PreCreateWindow(CREATESTRUCT& cs)
{
 // TODO: Modify the Window class or styles here by modifying
 //  the CREATESTRUCT cs

 cs.style |= (WS_CLIPCHILDREN | WS_CLIPSIBLINGS);

 return CView::PreCreateWindow(cs);
}
```

A primeira coisa a ser feita para criar um RC é definir o formato do *pixel* da janela. O formato do *pixel* descreve como os gráficos exibidos pela janela são representados na memória do computador. Alguns parâmetros controlados pelo formato do *pixel* são: profundidade de cor, método de buferização e o tipo de interfaces gráficas suportadas. Primeiro crie um novo método *protected* na classe CGLInterfaceView chamado BOOL SetWindowPixelFormat(HDC hDC) e edite-o da forma como apresentado abaixo. Adicione a variável membro *protected* do tipo *int* de nome m_GLPixelIndex a classe CGLInterfaceView.

```
BOOL CGLInterfaceView::SetWindowPixelFormat(HDC hDC)
{

 PIXELFORMATDESCRIPTOR pixelDesc;
 pixelDesc.nSize = sizeof(PIXELFORMATDESCRIPTOR);
 pixelDesc.nVersion = 1;

 pixelDesc.dwFlags = PFD_DRAW_TO_WINDOW |
                     PFD_DRAW_TO_BITMAP |
                     PFD_SUPPORT_OPENGL |
                     PFD_SUPPORT_GDI    |
                     PFD_STEREO_DONTCARE;

 pixelDesc.iPixelType = PFD_TYPE_RGBA;
 pixelDesc.cColorBits = 32;
 pixelDesc.cRedBits = 8;
 pixelDesc.cRedShift = 16;
 pixelDesc.cGreenBits = 8;
 pixelDesc.cGreenShift = 8;
 pixelDesc.cBlueBits = 8;
 pixelDesc.cBlueShift = 0;
 pixelDesc.cAlphaBits = 0;
 pixelDesc.cAlphaShift = 0;
 pixelDesc.cAccumBits = 64;
 pixelDesc.cAccumRedBits = 16;
 pixelDesc.cAccumGreenBits = 16;
 pixelDesc.cAccumBlueBits = 16;
 pixelDesc.cAccumAlphaBits = 0;
 pixelDesc.cDepthBits = 32;
 pixelDesc.cStencilBits = 8;
 pixelDesc.cAuxBuffers = 0;
 pixelDesc.iLayerType = PFD_MAIN_PLANE;
 pixelDesc.bReserved = 0;
 pixelDesc.dwLayerMask = 0;
 pixelDesc.dwVisibleMask = 0;
 pixelDesc.dwDamageMask = 0;

 m_GLPixelIndex = ChoosePixelFormat( hDC, &pixelDesc);
 if (m_GLPixelIndex==0) // Let's choose a default index.
 {
  m_GLPixelIndex = 1;
  if (DescribePixelFormat(hDC, m_GLPixelIndex,
    sizeof(PIXELFORMATDESCRIPTOR), &pixelDesc)==0)
 {
    return FALSE;
 }
 }
 if (SetPixelFormat( hDC, m_GLPixelIndex, &pixelDesc)==FALSE)
 {
  return FALSE;
 }
 return TRUE;
}
```

Usando o *ClassWizard*, adicione OnCreate em resposta a mensagem WM_CREATE e edite-o como mostrado a seguir. Compile o programa e corrija quaisquer erros de sintaxe.

```
int CGLInterfaceView::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
 if (CView::OnCreate(lpCreateStruct) == -1)
  return -1;

 HWND hWnd = GetSafeHwnd();
 HDC hDC = ::GetDC(hWnd);

 if (SetWindowPixelFormat(hDC)==FALSE)
  return 0;

 return 0;
}
```

Agora que o formato do *pixel* já está configurado, basta criar um contexto de renderização e torná-lo ativo. Comece adicionando um novo método *protected* a classe CGLInterfaceView chamado BOOL CreateViewGLContext(HDC hDC) e codifique-o como mostrado.

```
BOOL CGLInterfaceView::CreateViewGLContext(HDC hDC)
{

 m_hGLContext = wglCreateContext(hDC);
  if (m_hGLContext == NULL)
 {
   return FALSE;
 }
  if (wglMakeCurrent(hDC, m_hGLContext)==FALSE)
 {
   return FALSE;
 }
  return TRUE;

}
```

Adicione a variável de membro *protected* do tipo HGLRC e de nome m_hGLContext à classe CGLInterfaceView. Edite o método OnCreate para chamar a nova função.

```
int CGLInterfaceView::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
 if (CView::OnCreate(lpCreateStruct) == -1)
  return -1;

 HWND hWnd = GetSafeHwnd();
 HDC hDC = ::GetDC(hWnd);

 if (SetWindowPixelFormat(hDC)==FALSE)
  return 0;

 if (CreateViewGLContext(hDC)==FALSE)
  return 0;

 return 0;
}
```

Adicione OnDestroy em resposta a mensagem WM_DESTROY e edite-o como mostrado.

```
void CGLInterfaceView::OnDestroy()
{

 if(wglGetCurrentContext()!=NULL)
 {
  // make the rendering context not current
    wglMakeCurrent(NULL, NULL) ;
 }
  if (m_hGLContext!=NULL)
 {
    wglDeleteContext(m_hGLContext);
    m_hGLContext = NULL;
 }
  // Now the associated DC can be released.
  CView::OnDestroy();

}
```

Edite o construtor da classe CGLInterfaceView como mostrado a seguir.

```
CGLInterfaceView::CGLInterfaceView()
{
 m_hGLContext = NULL;
 m_GLPixelIndex = 0;
}
```

Mais uma vez compile o programa e corrija quaisquer erros de sintaxe. Quando você executa o programa ele se parece com uma *shell* genérica de um programa MFC, mas só que agora ele está habilitado para desenhos em OpenGL®. O método CreateViewGLContext cria e torna ativo um contexto de renderização (RC). A função wglCreateContext retorna um manipulador para o RC. O formato do *pixel* para o dispositivo associado ao DC, que é passado para dentro dessa função deve ser configurado antes que o método CreateViewGLContext seja chamado. A função wglMakeCurrent torna o RC ativo. O DC passado para dentro dessa função deve ter o mesmo dispositivo e formato de *pixel*. Se um outro contexto de renderização estiver ativo quando for chamado wglMakeCurrent, a função simplesmente exibe o antigo RC e o substitui pelo novo. Pode-se chamar wglMakeCurrent(NULL, NULL) para fazer com que nenhum RC fique ativo.

Use o *ClassWizard* para adicionar o método OnSize à classe CGLInterfaceView em resposta a mensagem WM_SIZE. Edit OnSize da seguinte forma.

```
void CGLInterfaceView::OnSize(UINT nType, int cx, int cy)
{

 CView::OnSize(nType, cx, cy);

 GLsizei width, height;
 GLdouble aspect;

 width = cx;
 height = cy;

 if (cy==0)
  aspect = (GLdouble)width;
 else
  aspect = (GLdouble)width/(GLdouble)height;

 glViewport(0, 0, width, height);
 glMatrixMode(GL_PROJECTION);
 glLoadIdentity();
 gluOrtho2D(0.0, 500.0*aspect, 0.0, 500.0);
 //gluPerspective(45, aspect, 1, 10.0);
 glMatrixMode(GL_MODELVIEW);
 glLoadIdentity();
 //glDrawBuffer(GL_BACK);
 //glEnable(GL_LIGHTING);
 //glEnable(GL_DEPTH_TEST);
}
```

Agora use o *ClassWizard* para adicionar o método OnPaint à classe CGLInterfaceView em resposta a mensagem WM_PAINT.

```
void CGLInterfaceView::OnPaint()
{
  CPaintDC dc(this); // device context for painting
                     // (added by ClassWizard)

  glLoadIdentity();
  glClear(GL_COLOR_BUFFER_BIT);

  glBegin(GL_POLYGON);

  glColor4f(1.0f, 0.0f, 0.0f, 1.0f);
  glVertex2f(100.0f, 50.0f);
  glColor4f(0.0f, 1.0f, 0.0f, 1.0f);
  glVertex2f(450.0f, 400.0f);
  glColor4f(0.0f, 0.0f, 1.0f, 1.0f);
  glVertex2f(450.0f, 50.0f);

  glEnd();

  glFlush();
}
```

Compile e execute o programa. Você deve ver uma janela com um fundo preto e um triângulo multicolorido. Conforme a janela é redimensionada o triângulo também é redimensionado. Todos os comandos OpenGL® usam o prefixo "gl". No método OnPaint pode-se notar o comando glColor4f, que define a cor no padrão RGB e, o comando glVertex2f que especifica as coordenadas dos vértices no plano 2D. Portanto, o resultado gráfico dos comandos que estão dentro do escopo do modo GL_POLYGON é um triângulo que possui um vértice vermelho, outro verde e outro azul. A cor no interior desse triângulo é definida pela interpolação linear das cores dos vértices. Esse programa, apesar de simples, já permite o desenho de geometrias tridimensionais basta usar as apropriadas funções do OpenGL® e projeções perspectivas. Este código, ainda, pode servir de base para um uso mais avançado do OpenGL® como a criação de animações e efeitos especiais. Para maiores informações e uma abordagem mais aprofundada consultar [24, 62].

Anexo B

Elementos Finitos do Ansys® - (*shell63*, *solid45* e *solid92*)

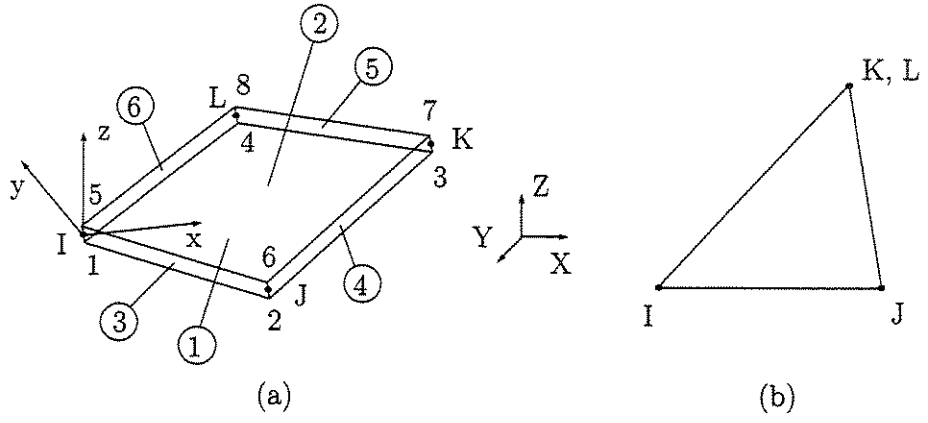
Introdução

O método dos elementos finitos é um procedimento numérico que pode ser aplicado para obter soluções em uma grande variedade de problemas de engenharia [22]. Várias abordagens do método como a *formulação direta*, a *formulação da energia potencial mínima* e a *formulação por resíduos ponderados*, foram desenvolvidas para tratar problemas estacionários, transientes, lineares e não-lineares em: análise de tensões, transferência de calor, escoamento de fluidos, eletromagnetismo e outras áreas. Referências clássicas do método incluem [17, 63].

Todas as abordagens do método dos elementos finitos envolvem a divisão do sistema físico estudado em pequenas sub-regiões ou elementos. Cada elemento é essencialmente uma unidade simples cujo comportamento pode ser prontamente analisado. As complexidades do sistema global são representadas usando uma grande quantidade de elementos. No caso de uma estrutura pino-junta por exemplo, a escolha do elemento é óbvia; cada membro da estrutura é tratado como um elemento sujeito à forças de tração e compressão. A partir das análise simples de rigidez dos elementos individuais, é possível estabelecer o comportamento da estrutura completa em termos das respostas de deslocamento originadas pelas cargas externas aplicadas. Essas cargas são aplicadas nas juntas da estrutura e é conveniente considerar os deslocamentos nesses mesmos pontos, que são referidos como nós. A seguir são apresentados os elementos do Ansys® usados nas análises numéricas mostradas no capítulo 5 com suas respectivas características e capacidades de análise.

Elemento *shell63*

O elemento *shell63* tem capacidade de flexão e características de membrana. Tanto carregamento na direção normal como no plano são permitidos. Cada nó do elemento possui seis graus de liberdade: translações nas direções nodais x, y e z; e rotações ao redor dos eixos nodais x, y e z. Na formulação do elemento também estão incluídas as características de rigidez a tensão e grandes deformações. Uma opção de matrix de rigidez especial está disponível para o uso em análises envolvendo problemas com grandes deformações (rotações finitas). Para maiores informações consultar *Ansys® Inc. Theory Reference* e [39].



Elemento *shell63*: (a) opção quadrilateral; e (b) opção triangular.

Dados de Entrada Principais

O elemento é definido por quatro nós, quatro espessuras, uma rigidez elástica e pelas propriedades do material.

Elemento:

Nome → *shell63*

Nós → I, J, K, L

Graus de Liberdade:

Deslocamento → UX, UY, UZ

Rotações → ROTX, ROTY, ROTZ

Constantes Reais:

Espessura dos nós I, J, K, L → TK(I), TK(J), TK(K), TK(L)

Rigidez Elástica → EFS

Ângulo de Rotação do Eixo x → THETA

Propriedades do Material:

Módulo de Elasticidade → EX, EY, EZ

Coeficiente de Poisson → PRXY, PRYZ, PRXZ, ou NUXY, NUYZ, NUXZ

Densidade → DENS

Forças de Superfície:

Pressão → face1, face2, face3, face4, face5, face6

Forças de Corpo:

Temperatura → T1, T2, T3, T4, T5, T6, T7, T8

KEYOPT(1):

0 → rigidez de membrana e flexão

1 → rigidez de membrana somente

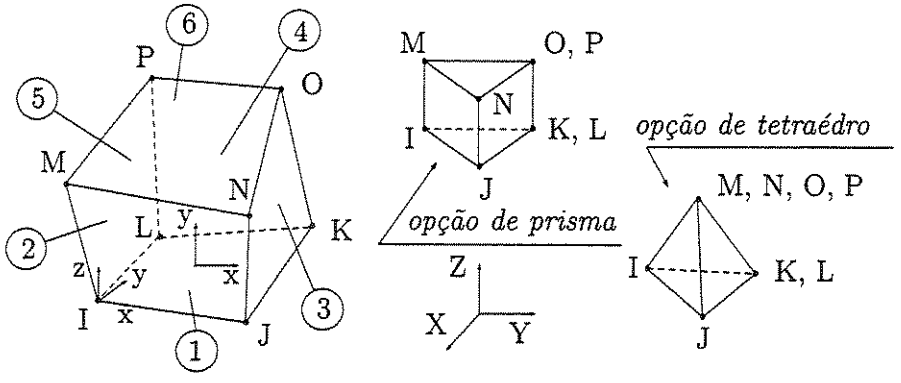
2 → rigidez de flexão somente

Dados de Saída Principais

As principais informações de saída, depois de realizada a análise numérica são: valor de tensão nodal; solução nodal de temperatura; intensidades de tensão e deslocamento; tensão equivalente; deformação elástica média e deformação térmica média.

Elemento *solid45*

O elemento *solid45* é usado para modelagem sólida de estruturas tridimensionais. O Elemento é definido por oito nós, cada um com três graus de liberdade: translações nas direções nodais x, y e z. O elemento possui características de plasticidade, fluência, *swelling*, rigidez a tensão, grandes flexões e grandes deformações. Uma opção de integração reduzida com controle de tempo de processamento é disponível.



Elemento *solid45*: com opção de forma prismática e tetraédrica.

Dados de Entrada Principais

O elemento é definido por oito nós e pelas propriedades do material.

Elemento:

Nome → *solid45*

Nós → I, J, K, L, M, N, O, P

Graus de Liberdade:

Deslocamento → UX, UY, UZ

Propriedades do Material:

Módulo de Elasticidade → EX, EY, EZ

Coeficiente de Poisson → PRXY, PRYZ, PRXZ, ou NUXY, NUYZ, NUXZ

Densidade → DENS

Forças de Superfície:

Pressão → face1, face2, face3, face4, face5, face6

Forças de Corpo:

Temperatura → T(I), T(J), T(K), T(L), T(M), T(N), T(O), T(P)

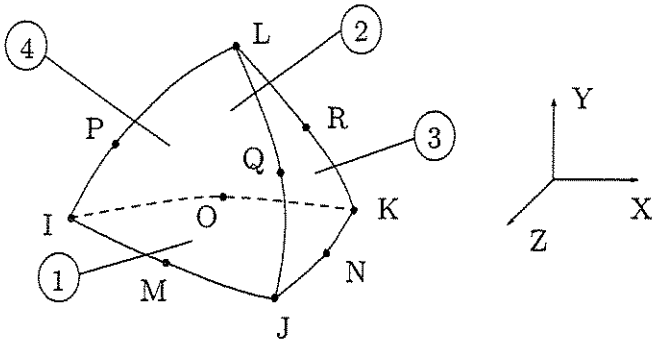
Fluências → FL(I), FL(J), FL(K), FL(L), FL(M), FL(N), FL(O), FL(P)

Dados de Saída Principais

As principais informações de saída, depois de realizada a análise numérica são: valor de deslocamento nodal; solução nodal de temperatura; intensidades de tensão e deslocamento; tensão equivalente.

Elemento *solid92*

O elemento *solid92* é usado para modelagem sólida de estruturas tridimensionais. Por possuir um comportamento de deslocamento quadrático, este elemento é bastante adequado para modelar geometrias irregulares. O Elemento é definido por dez nós, cada um com três graus de liberdade: translações nas direções nodais x, y e z. O elemento possui características de plasticidade, fluência, *swelling*, rigidez a tensão, grandes flexões e grandes deformações.



Elemento *solid92*: tetraédro de 10 nós com interpolação quadrática.

Dados de Entrada Principais

O elemento é definido por dez nós e pelas propriedades do material.

Elemento:

Nome → *solid92*

Nós → I, J, K, L, M, N, O, P, Q, R

Graus de Liberdade:

Deslocamento → UX, UY, UZ

Propriedades do Material:

Módulo de Elasticidade → EX, EY, EZ

Coeficiente de Poisson → PRXY, PRYZ, PRXZ, ou NUXY, NUYZ, NUXZ

Densidade → DENS

Forças de Superfície:

Pressão → face1, face2, face3, face4

Forças de Corpo:

Temperatura → T(I), T(J), T(K), T(L), T(M), T(N), T(O), T(P), T(Q), T(R)

Fluências → FL(I), FL(J), FL(K), FL(L), FL(M), FL(N), FL(O), FL(P), FL(Q), FL(R)

Dados de Saída Principais

As principais informações de saída, depois de realizada a análise numérica são: valor de deslocamento nodal; solução nodal de temperatura; intensidades de tensão e deslocamento; tensão equivalente.