
PUC
RIO

**JOAQUIM BENTO CAVALCANTE NETO**

# *GERAÇÃO DE MALHA E ESTIMATIVA DE ERRO PARA MODELOS TRIDIMENSIONAIS DE ELEMENTOS FINITOS COM TRINCAS*

**TESE DE DOUTORADO**

Departamento de Engenharia Civil
PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

Rio de Janeiro, Setembro de 1998

PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO

Rua Marquês de São Vicente, 225 - Gávea
CEP 22453-900 Rio de Janeiro RJ Brasil
http://www.puc-rio.br

Joaquim Bento Cavalcante Neto

# Geração de Malha e Estimativa de Erro para Modelos Tridimensionais de Elementos Finitos com Trincas

Tese apresentada ao Departamento de Engenharia Civil da PUC-Rio como parte dos requisitos para obtenção do título de Doutor em Engenharia Civil: Estruturas.

Orientador: Luiz Fernando C. R. Martha

Departamento de Engenharia Civil
TeCGraf - Grupo de Tecnologia em Computação Gráfica

Pontifícia Universidade Católica do Rio de Janeiro

Rio de Janeiro, 28 de Setembro de 1998

92957

Aos meus pais, Joaquim Bento e Francisca Maria,
e aos meus irmãos, Patrícia, Alexandre e André.

# Agradecimentos

A minha família, pelo apoio, incentivo, confiança e carinho em todos os momentos da minha vida.

A Luiz Fernando Martha, pela amizade formada durante a convivência, fundamental no decorrer deste período todo, e também pela orientação, apoio, incentivo e confiança recebidos durante o desenvolvimento do trabalho.

Aos amigos Alexandre Miranda Mont'Alverne e Paulo Rodacki Gomes, pelo apoio nessa difícil tarefa que é fazer uma tese de doutorado.

A todas as pessoas que contribuíram de maneira direta e indireta para a realização deste trabalho. Em especial, a Ivan F.M. Menezes pela colaboração nos aspectos relacionados à implementação das técnicas de estimativa de erro de discretização; a Marcelo Tílio M. de Carvalho pelas longas discussões sobre o algoritmo de geração da malha e sobre o trabalho como um todo; a Luiz Cristóvão G. Coelho, pelas discussões sobre geração de malha de volume e superfície; a Evandro Parente, pelo trabalho em conjunto no programa de análise de elementos finitos; a João Luiz Campos pela discussão de vários aspectos do trabalho; a Waldemar Celes Filho, sempre aberto e disposto a colaborar em todos os aspectos do trabalho; Camilo da Fonseca Freire, pelas correções em algumas figuras.

A todos os amigos, professores e funcionários do Departamento de Engenharia Civil, em especial aos amigos Eduardo Setton, Marcos Aurélio M. Noronha, Carlos Eduardo Kubrusly, João Batista Sousa Marques, Andréia A. Diniz, Áurea Holanda, Raquel Prates, William Wagner Matos Lira, Carlos Vitor de Alencar Carvalho, Antonio Carlos de Oliveira Miranda.

A todos os amigos do TeCGraf, pelo apoio recebido e pela amizade em todos os momentos, que faz do grupo um ambiente ideal para o desenvolvimento de qualquer trabalho.

A todos os amigos do Grupo de Fratura da Universidade de Cornell/EUA, em especial a Paul A. Wawrzynek e a Anthony R. Ingraffea.

Ao CNPq e ao convênio PUC–PETROBRÁS, pelo auxílio financeiro.

# Resumo

Este trabalho propõe uma técnica de geração de malhas volumétricas de elementos tetraédricos, para domínios tridimensionais arbitrários. A principal motivação no desenvolvimento deste algoritmo foi a necessidade da consideração de alguns requisitos geométricos que não podem ser tratados por outros algoritmos encontrados na literatura, especialmente em problemas de fraturamento. Além disso, este trabalho apresenta uma implementação genérica, baseada em programação orientada a objetos, de eficientes estimadores de erro de discretização encontrados na literatura, para modelos de elementos finitos tridimensionais. As técnicas apresentadas são integradas em um sistema gráfico computacional, permitindo modelagem, análise e visualização de modelos sólidos de elementos finitos. Este sistema serve como base para validar essas técnicas e como protótipo para um futuro sistema para simulações genéricas adaptativas tridimensionais usando o método dos elementos finitos.

# Abstract

This work proposes a volume mesh generation technique for tetrahedral elements for three-dimensional arbitrary domains. The main motivation in the development of this algorithm was the need of some specific requirements that could not be treated by other algorithms found in the technical literature, specially in fracture problems. In addition, this work presents a general implementation, based on object oriented programming, of some efficient error estimation techniques found in the literature, for three-dimensional finite element models. The presented techniques are integrated in a graphical computational system, allowing modeling, analysis and visualization of solid finite element models. This system is used as a test-bed to validate these techniques and as a prototype for a future system for generic adaptive three-dimensional simulations using the finite element method.

# Sumário

# Lista de figuras

# Lista de pseudo-códigos

# 1

# Introdução

Dentre os métodos numéricos usados para simular as classes de problemas de engenharia em que a solução analítica é desconhecida ou difícil de se obter, o método dos elementos finitos (*MEF*) é provavelmente o mais usado. Uma razão para isso é a confiabilidade da solução numérica obtida pelo método. Outra razão é a relativa simplicidade de implementação dos procedimentos numéricos relacionados ao método.

Em uma visão geral, a idéia principal do método é subdividir o domínio do problema em pequenas regiões (elementos), onde o comportamento do campo de interesse possa ser aproximado por funções simples, tais como polinômios ou funções harmônicas. Essas funções são expressas pelos valores do campo nos vértices (nós) destes elementos; estes valores, as incógnitas do problema discreto, são determinados pela minimização de um funcional associado às equações diferenciais que governam o problema. O modelo discreto resultante é denominado *malha de elementos finitos*. Em geral, o domínio de interesse é intrinsicamente tridimensional para a maioria dos problemas, mas várias vezes a simulação é feita em duas dimensões, e algumas considerações adicionais são necessárias. Uma das razões para esta simplificação é que, em alguns casos, é realmente difícil criar um modelo discreto adequado para descrever a geometria e a topologia do domínio tridimensional e suas condições de contorno. Por exemplo, procedimentos para gerar malhas tridimensionais volumétricas estão ainda sob investigação e pesquisa. Em outros casos, evita-se a criação de um modelo tridimensional pois isto acarretaria em uma malha muito grande, tornando a análise cara do ponto de vista computacional. Existem situações, entretanto, em que uma aproximação bidimensional não é correta, e todas essas dificuldades devem ser superadas para realmente tratar um modelo tridimensional.

Outro importante aspecto do *MEF* é a definição do grau de refinamento da malha. Pelo fato das funções de aproximação dos elementos serem geralmente funções simples, é necessário aumentar o grau de refinamento em regiões de gradientes elevados do campo de solução. O problema com isso é que o campo da solução é desconhecido. Práticas normais para resolver esse problema envolvem aumentar o número de pontos

de discretização no domínio computacional e resolver o sistema de equações resultantes para examinar a mudança relativa na solução numérica. Em geral, este procedimento consome tempo, depende da experiência do analista e pode guiar a interpretações erradas se a solução não tiver entrado em uma faixa assintótica. Nesse contexto, processos adaptativos tem se tornado muito importantes para aumentar a confiabilidade do procedimento de análise, já que não dependem da experiência, ou inexperiência, do analista. Um processo adaptativo é um procedimento para avaliar o erro de discretização do modelo para uma dada solução e para melhorar esta discretização até se atingir uma tolerância de erro fornecida para a solução numérica.

## 1.1 *Motivação e trabalhos relacionados*

Pode-se deduzir que, para se realizar simulações reais de elementos finitos, geração de malha e métodos adaptativos são de grande importância. Algumas estratégias têm sido propostas para considerar esses aspectos de uma maneira eficiente. Em um trabalho anterior deste autor (Cavalcante Neto, 1994), foi desenvolvida uma estratégia auto-adaptativa bidimensional capaz de realizar a simulação, envolvendo geração de malha e métodos adaptativos, de uma maneira automática. Outros trabalhos têm sido feitos para considerar o mesmo problema usando diferentes enfoques, como por exemplo o trabalho desenvolvido pelo grupo de Mark Shephard do Rensselaer Polytechnic Institute, EUA (Baehmann e Shephard, 1989). Para simulações tridimensionais, geração de malha volumétrica e estimativas de erro numérico são necessárias. Entretanto, essas são tarefas relativamente complexas. Pesquisa tem sido feita em ambas as áreas para desenvolver métodos e algoritmos capazes de realizar essas tarefas.

Na área de geração de malha volumétrica, existem vários algoritmos na literatura técnica para gerar malhas não-estruturadas tridimensionais, especialmente para elementos tetraédricos, e vários enfoques têm sido levados em consideração. Malhas não-estruturadas não têm nenhum tipo de regularidade ou periodicidade, diferente de malhas estruturadas onde existe alguma estrutura local ou global. Apesar do uso de malhas estruturadas em algumas áreas, como aerodinâmica, malhas não-estruturadas são geralmente preferidas porque elas podem ser aplicadas para representar domínios tridimensionais de forma arbitrária e são mais adequadas para métodos adaptativos. Dentre as diferentes técnicas não-estruturadas usadas, existem algoritmos baseados em triangulação de Delaunay (Watson, 1981; Field, 1986; Baker, 1987; Joe, 1990), em decomposição espacial recursiva (Yerry e Shephard, 1984; Perucchio *et al.*, 1989, Shephard

e Georges, 1991), e em métodos de avanço da fronteira (Peraire *et al.*, 1988; Lohner e Parikh, 1988; Moller e Hansbo, 1995) [1].

Essas técnicas têm seus requisitos específicos em relação a tamanho e forma do elemento finito ou em relação a aspectos de transição da malha. Recentemente, mais atenção tem sido dada à qualidade da malha, para assegurar que todos os elementos tenham uma "boa" forma de acordo com uma medida escolhida. É óbvio que estes requisitos, junto com a confiabilidade do algoritmo, isto é, a capacidade do algoritmo de gerar uma malha válida para domínios arbitrários, são muito mais difíceis de serem satisfeitos em três dimensões.

Em relação a procedimentos adaptativos de elementos finitos, desde os anos 70, um grande número de trabalhos sobre técnicas adaptativas tem sido publicados na literatura. Em 1978, Babuška e Rheinboldt apresentaram um trabalho sobre estimativa de erro avaliando os resíduos da solução aproximada e usando-os para obter respostas locais mais apuradas (Babuška e Rheinboldt, 1978). Eles desenvolveram a base matemática de técnicas adaptativas.

Zienkiewicz *et al.* apresentaram, em 1982, um enfoque hierárquico para métodos adaptativos (Zienkiewicz *et al.*, 1982). Em 1987, Zienkiewicz e Zhu introduziram um estimador de erro baseado na obtenção de valores melhorados dos gradientes (tensões) usando alguns processos de recuperação de tensão disponíveis (Zienkiewicz e Zhu, 1987). Fácil de ser implementada em qualquer código de elementos finitos, esta técnica, baseada em média e na chamada projeção $L_2$, tem sido usada para recuperar os gradientes, e razoáveis estimadores de erro têm sido obtidos. Em 1992, esta técnica foi melhorada e corrigida pelos mesmos autores, levando à criação da técnica chamada *Superconvergent Patch Recovery* (*SPR*) (Zienkiewicz e Zhu, 1992a; Zienkiewicz e Zhu, 1992b; Zienkiewicz e Zhu, 1994). Este método é um procedimento de suavização de gradientes sobre grupos (*patches*) locais de elementos. Ele se baseia em um procedimento de mínimos quadrados, em sua forma discreta, de um campo de gradiente polinomial de ordem mais elevada com relação ao gradiente considerado, em pontos de amostragem superconvergentes. Estes pontos de amostragem são obtidos do cálculo de elementos finitos. Tentativas de melhorar posteriormente o processo de recuperação podem ser encontradas, por exemplo, em alguns outros trabalhos (Wiberg e Abdulwahab, 1993; Wiberg *et al.*, 1994; Blacker e Belytschko, 1994; Tabbara *et al.*, 1994). Essencialmente, essas técnicas melhoradas

[1] Na verdade, existem muitos outros trabalhos relacionados a cada técnica além dos mencionados, mas a maioria deles seguem conceitos similares.

incorporam equilíbrio e condições de contorno no processo de recuperação. Um estudo mostrou, através de numerosos exemplos, seu excelente desempenho e superioridade sobre enfoques baseados em resíduos (Babuška *et al.*, 1994a; Babuška *et al.*, 1994b).

Recentemente, Boroomand e Zienkiewicz apresentaram um novo método superconvergente satisfazendo a condição de equilíbrio em uma forma fraca, que não requer nenhum conhecimento de pontos superconvergentes (Boroomand e Zienkiewicz, 1997). Essa técnica de recuperação foi chamada *Recovery by Equilibrium in Patches* (REP ).

A maioria dos trabalhos mencionados são relacionados aos aspectos teóricos do problema e idealmente eles podem ser empregados nos casos bidimensional e tridimensional. Na prática, entretanto, eles têm sido aplicados somente em duas dimensões por causa de algumas limitações, como geração de malha volumétrica ou definição do *patch* tridimensional para a aplicação das técnicas de *SPR* ou *REP*, por exemplo.

## 1.2 *Objetivos do trabalho*

O objetivo deste trabalho é propor uma técnica de geração de malhas volumétricas e apresentar uma implementação genérica, baseada em Programação Orientada a Objetos (*POO*), de eficientes técnicas de estimativa de erro encontradas na literatura, para domínios tridimensionais arbitrários. Estas técnicas são integradas em um sistema gráfico computacional, permitindo modelagem, análise e visualização de modelos sólidos de elementos finitos. Este sistema serve como base para validar essas técnicas e como protótipo para um futuro sistema para simulações genéricas adaptativas tridimensionais usando o método dos elementos finitos.

### 1.2.1 *Geração de malha volumétrica*

A técnica de geração de malhas volumétricas proposta é capaz de gerar malhas tridimensionais de elementos tetraédricos para domínios arbitrários. O algoritmo também funciona para domínios com restrições geométricas na forma de uma ou múltiplas trincas, usadas no escopo de simulações de fraturamento, como as realizadas pelo Grupo de Fratura da Universidade de Cornell nos EUA (*Cornell Fracture Group - CFG*), ao qual este trabalho está relacionado. Trincas são falhas que podem potencialmente degradar a vida de uma estrutura, e simulações de fraturamento têm se tornado cada vez mais importantes nos últimos anos. O *CFG* tem realizado simulações de fraturamento usando o método dos elementos de contorno (*MEC*). O grupo tem sido muito bem sucedido

nestas simulações, mas em alguns casos, como em análise elasto-plástica, é preferível empregar o *MEF*. Portanto, geração de malha volumétrica para domínios com trincas é fundamental neste contexto.

O algoritmo combina uma técnica de avanço da fronteira com uma decomposição espacial recursiva, neste caso uma *octree* (Samet, 1984), para lidar com a criação de nós internos e com a transição da malha. A maior motivação no desenvolvimento deste algoritmo foi a necessidade de alguns requisitos específicos que não são diretamente ou eficientemente tratados por outros algoritmos encontrados na literatura.

Primeiro, o algoritmo deve produzir elementos de boa forma, evitando a geração de elementos com má qualidade, sempre que possível. O algoritmo possui procedimentos que cuidam para que os melhores elementos possíveis sejam gerados a cada passo. Além disso, o algoritmo propõe um procedimento *a posteriori* que procura melhorar localmente a qualidade dos elementos gerados.

Um segundo requisito é que o algoritmo deve gerar uma malha volumétrica que se conforme com uma dada malha triangular no contorno do modelo. Isto é importante porque permite uma nova geração da malha ocorrer localmente em uma região perto de uma trinca se propagando. Nesta região, um grupo pequeno de elementos pode ser eliminado, criando um vazio na malha por onde a trinca se propaga, e então o algoritmo pode ser usado para preencher este vazio com novos elementos que se conformam com os elementos que não foram removidos. Um procedimento similar pode ser aplicado em problemas adaptativos, preenchendo com novos elementos mais refinados somente regiões com alto erro de discretização. O algoritmo, entretanto, não se aplica somente a regiões pequenas perto de trincas ou com alto erro de discretização. Ele pode ser aplicado para gerar malhas volumétricas em modelos tridimensionais de qualquer tamanho.

O terceiro requisito desejado é que o algoritmo seja capaz de gerar uma boa transição entre regiões de diferentes tamanhos de elementos. Em análises de fraturamento, por exemplo, não é incomum que elementos perto da trinca tenham um tamanho característico duas ordens de magnitude menor do que o de outros elementos no domínio. O algoritmo foi desenvolvido para ser capaz de gerar malhas com boa transição entre diferentes tamanhos de elementos.

Finalmente, o algoritmo deve ser capaz de lidar com modelos que possuam trincas, o que confere um grau de complexidade maior na geração de malhas volumétricas para domínios arbitrários. Trincas são idealizadas normalmente como não tendo vo-

lume, isto é, as superfícies representando os dois lados de uma trinca são distintas, mas geometricamente coincidentes. Isto significa que nós em lados opostos das faces da trinca têm coordenadas idênticas. O algoritmo deve ser capaz de discriminar entre os nós e escolher aquele que se localiza na face correta da trinca.

### 1.2.2 *Estimativa de erro em três dimensões*

A metodologia proposta da estimativa de erro de discretização do *MEF* consiste em uma implementação geral de duas eficientes técnicas encontradas na literatura, a técnica chamada *Superconvergent Patch Recovery* (*SPR*), usando sistemas de mínimos quadrados ponderados, e a técnica chamada *Recovery by Equilibrium in Patches* (*REP*), recentemente proposta. Esta implementação genérica é baseada em *POO*.

Com o objetivo de se obter um sistema computacional confiável e eficiente para simulações de elementos finitos, considerando estimativas de erro de discretização, um tratamento genérico para lidar com diferentes elementos finitos (por consequência, diferentes formas e diferentes graus de funções de interpolação) tem de ser levado em consideração. Nos últimos cinco anos, o programa *FEMOOP* (Martha *et al.*, 1996) tem sido desenvolvido no Departamento de Engenharia Civil da Universidade Católica do Rio de Janeiro (PUC-Rio) para realizar análises de elementos finitos. Inicialmente, este programa foi desenvolvido em linguagem *C*, mas considerando uma disciplina de *POO* para lidar com todas as diferentes formas e diferentes graus de funções de interpolação. Este código foi usado em trabalhos anteriores deste autor para implementar um estimador de erro baseado em uma técnica de recuperação chamada *ZZ* (Zienkiewicz e Zhu, 1987), e para construir uma estratégia auto-adaptativa para problemas bidimensionais (Cavalcante Neto, 1994). Com o objetivo de se melhorar seu desempenho e funcionalidade, o programa foi migrado para uma programação formal orientada a objetos, utilizando a linguagem *C++* [2]. Isto permite uma manutenção mais fácil do código e uma expansão localizada, com uma menor chance de introdução de erros. As novas técnicas de recuperação mostradas neste trabalho foram implementadas no *FEMOOP*.

O maior desafio na implementação destas técnicas para modelos tridimensionais é a formação de *patches* em três dimensões, já que a formulação original básica mostrada por Zienkiewicz e Zhu foi exemplificada para modelos bidimensionais. O conceito de *POO* facilita a definição e implementação da estimativa de erro para modelos

[2] Para maiores detalhes sobre a linguagem *C++*, veja livro de Stroustrup (1997).

tridimensionais, já que a maioria dos métodos já implementados pelo autor para casos bidimensionais (Cavalcante Neto *et al.*, 1998; Paulino *et al.*, 1998) podem ser utilizados ou facilmente estendidos para três dimensões.

## 1.3 *Metodologia e principais contribuições*

O algoritmo proposto para geração de malhas volumétricas foi introduzido em um programa chamado *FRANC3D* (Martha, 1989; Wawrzynek, 1991), usado como plataforma para realizar gerações de malha neste trabalho. *FRANC3D* é um programa desenvolvido pelo *CFG*, originalmente para gerar modelos para o *MEC*, e possui algumas funcionalidades para permitir a modelagem da geometria do domínio, incluindo a definição de superfícies de trincas, especificação de atributos, e geração de malhas de contorno para domínios arbitrários. Aproveitando a estrutura do programa já disponível, as malhas de contorno são usadas como entrada para o algoritmo de geração da malha volumétrica e o *FRANC3D* foi estendido para gerar modelos para o *MEF*.

Tradicionalmente, simulações numéricas envolvem três fases. Em uma etapa de pré-processamento, o modelo numérico é gerado, o que representa os dados necessários para a análise numérica. Esta etapa é realizada pelo programa *FRANC3D* neste trabalho. Após esta etapa, a análise é realizada para o modelo numérico gerado. Neste trabalho isto é realizado pelo programa *FEMOOP*, que é também capaz de calcular os erros de discretização para o modelo, se desejado. A análise gera resultados que podem ser analizados e interpretados por um pós-processador. Neste trabalho isto é feito pelo programa *POS3D* (Celes Filho *et al.*, 1991), desenvolvido pelo Grupo de Tecnologia em Computação Gráfica da PUC-Rio (*TeCGraf*), que é outro grupo ao qual este trabalho está também relacionado. A figura 1.1 mostra o programa *FRANC3D* com um modelo hipotético que contém uma trinca circular em seu interior.

Técnicas de geração de malhas volumétricas e de estimativa de erro compatíveis são cruciais para o desenvolvimento de uma estratégia adaptativa tridimensional para elementos finitos. Acredita-se que as técnicas propostas neste trabalho possam ser usadas no desenvolvimento de uma estratégia adaptativa tridimensional confiável e eficiente, incluindo a possibilidade de lidar com trincas. Nesse contexo, as principais contribuições deste trabalho podem ser resumidas como:

- O desenvolvimento de uma técnica de geração de malhas tridimensionais para elementos tetraédricos, garantindo confiabilidade e qualidade, para domínios arbitrários, incluindo trincas.

- Uma implementação geral, baseada em *POO*, de eficientes técnicas de estimativas de erro, chamadas *Superconvergent Patch Recovery* (*SPR*) e *Recovery by Equilibrium in Patches* (*REP*), para modelos tridimensionais.
- A integração destas técnicas em um sistema gráfico computacional, permitindo modelagem, análise e visualização, e criando um sistema base para validação das técnicas e para servir como protótipo para uma futuro sistema para simulações genéricas adaptativas tridimensionais usando o *MEF*.

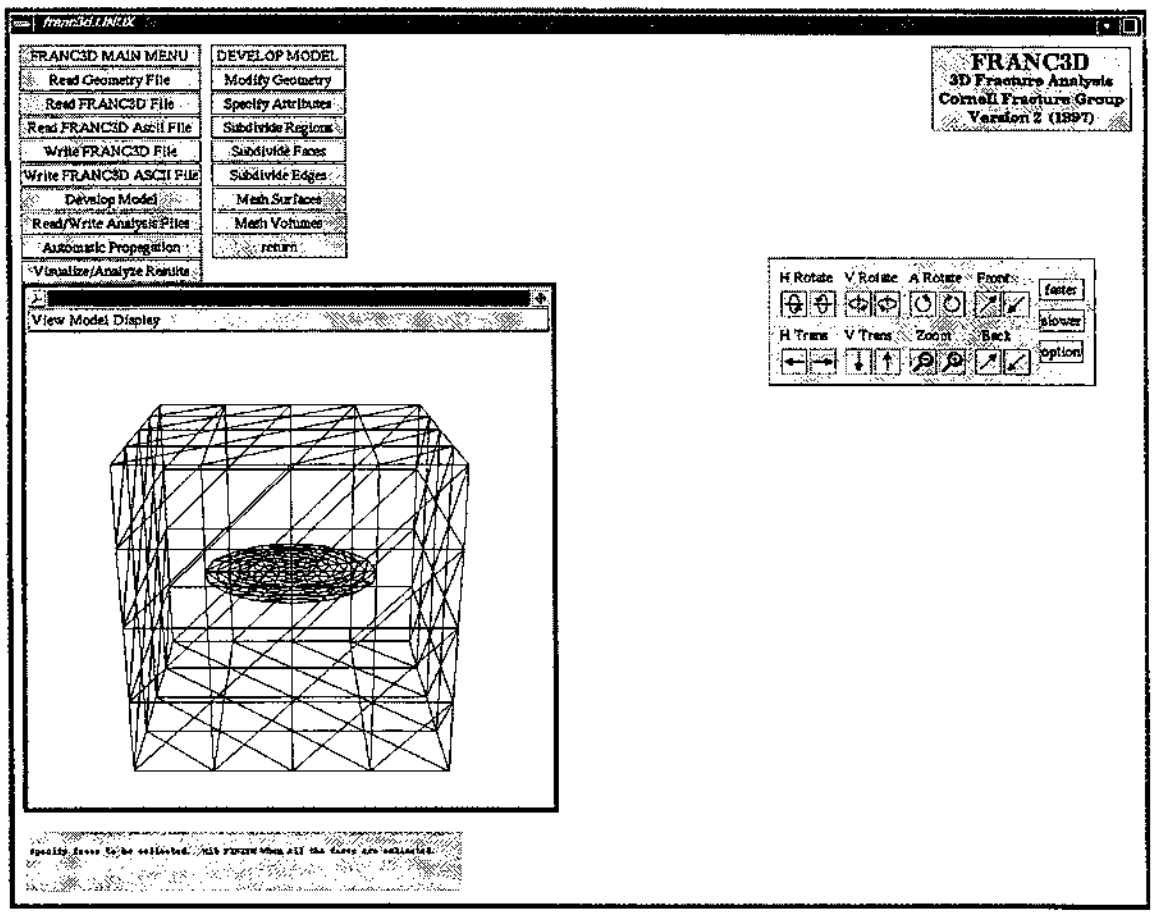



Figura 1.1: Interface gráfica do programa *FRANC3D*.

## 1.4 *Organização do trabalho*

O restante deste trabalho está organizado como descrito a seguir. O capítulo 2 apresenta a implementação geral, em três dimensões, da técnica de *Superconvergent Patch Recovery* (*SPR*), usando sistemas de mínimos quadrados ponderados, e da técnica *Recovery by Equilibrium in Patches* (*REP*).

O capítulo 3 apresenta o algoritmo proposto para geração de malhas volumétricas de tetraedros para domínios arbitrários, incluindo trincas. Além disso, uma análise de qualidade das malhas geradas é realizada e medidas empíricas da performance do algoritmo são descritas.

O capítulo 4 mostra alguns exemplos de aplicação da técnica de geração de malhas volumétricas juntamente com as técnicas de estimativa de erro em três dimensões. São mostrados nestes exemplos refinamentos sucessivos baseado nas estimativas de erro de discretização advindas de análises em passos anteriores. Além disso, a análise de um problema com uma trinca é realizada.

Finalmente no capítulo 5 algumas considerações finais são feitas com o objetivo de avaliar a eficiência e aplicabilidade das técnicas propostas. São apresentadas ainda sugestões para trabalhos futuros.

# 2

# Estimativa de erro em três dimensões

Este capítulo apresenta uma implementação geral de técnicas de estimativa de erro de discretização encontradas na literatura técnica, chamadas *Superconvergent Patch Recovery (SPR)* (Zienkiewicz e Zhu, 1992a; Zienkiewicz e Zhu, 1992b; Zienkiewicz e Zhu, 1994) e a recentemente proposta técnica chamada *Recovery by Equilibrium in Patches (REP)* (Boroomand e Zienkiewicz, 1997). Uma implementação revisitada da técnica de *SPR*, usando sistemas de mínimos quadrados ponderados durante o processo de recuperação, é mostrada. A formulação proposta, utilizando programação orientada a objetos, é genérica para problemas bidimensionais ou tridimensionais. Considerações sobre a aplicação dessa formulação para estimativa de erro para problemas tridimensionais, que é o objetivo deste trabalho, são feitas.

Análise de erros de discretização tem se tornado cada vez mais importante para aumentar a confiabilidade de modernos métodos numéricos aplicados a problemas de engenharia. Sempre que um método numérico é usado para solucionar equações diferenciais relacionadas a um funcional definido para um problema discreto, erros são introduzidos pelo processo de discretização que reduz o modelo matemático contínuo para um modelo numérico com um número finito de graus de liberdade. Os erros de discretização são definidos como a diferença entre a solução exata analítica e sua aproximação numérica. Por definição, o erro local

$$e = \phi - \hat{\phi}, \tag{2.1}$$

é uma medida da diferença entre a solução exata ($\phi$) e uma solução aproximada ($\hat{\phi}$). Neste contexo, $\phi$ é análogo a uma resposta (por exemplo, deslocamentos) em um procedimento típico de solução numérica. Métodos adaptativos são esquemas numéricos que se ajustam com o objetivo de melhorar a solução do problema para uma especificada acurácia [1]. Os dois componentes básicos em um método adaptativo são a estimativa de erro e a estratégia adaptativa.

[1] Se os esquemas são feitos sem a intervenção do usuário, eles são chamados auto-adaptativos.

Em geral, existem dois tipos de estimativas de erro de discretização: *a priori* e *a posteriori*. Embora estimativas *a priori* sejam apuradas para o pior caso em uma classe particular de soluções de um problema, elas geralmente não fornecem informações acerca do erro exato para um dado modelo. Estimativas *a posteriori* usam informações obtidas durante o processo de solução, em combinação com algumas hipóteses assumidas *a priori* sobre a solução. Estimativas *a posteriori*, que podem fornecer medidas quantitativamente apuradas do erro de discretização, são adotadas neste trabalho.

No contexto de estratégias adaptativas, métodos de extensão têm sido preferidos sobre outros enfoques (por exemplo, métodos duais ou complementares). Estes métodos incluem extensões $h$, $p$ e $r$. A implementação computacional se refere a versões $h$, $p$ e $r$, respectivamente. Na extensão $h$, a malha é refinada quando o indicador de erro excede uma tolerância pré-estabelecida. A extensão $p$ é geralmente empregada em uma malha fixa. Neste caso, se o erro em um elemento excede a tolerância pré-estabelecida, a ordem local da função de aproximação é aumentada para reduzir o erro. A extensão $r$ é empregada em um número fixo de nós e tenta mover dinamicamente os nós da malha para áreas de erros elevados. Qualquer uma dessas extensões podem ser também combinadas em uma estratégia especial, por exemplo, *h-p*, *r-h*, entre outras (Las Casas, 1990; Cyrino, 1989; Ribeiro, 1991).

## 2.1 *Estimativa de erro e refinamento adaptativo*

Como assinalado por vários autores (por exemplo, Zienkiewicz e Taylor, 1989), a especificação do erro na maneira dada na equação (2.1) não é geralmente conveniente e ocasionalmente pode levar a caminhos errados. Baseado nisso, normas matemáticas são introduzidas para medir o erro de discretização. O erro de discretização na solução de problemas de elasticidade estrutural por elementos finitos é freqüentemente quantificado com base na norma de energia para o erro nos deslocamentos, $||e||$, que pode ser expressa, em termos das tensões, como

$$\|e\| = \left( \int_{\Omega} (\sigma - \hat{\sigma})^t \, D^{-1} \, (\sigma - \hat{\sigma}) \; d\Omega \right)^{1/2}, \tag{2.2}$$

onde $\sigma$ é o campo de tensões exato, e $\hat{\sigma}$ é o campo de tensões dado por elementos finitos, $D$ é a matriz constitutiva, e $\Omega$ é o domínio de definição do problema.

A idéia básica dos estimadores de erro é substituir o campo $\sigma$, que é geralmente desconhecido, pelo campo $\overline{\sigma}$, obtido por meio de procedimentos de recuperação (por

exemplo, *ZZ, SPR* ou *REP*). Então, a expressão para cálculo da norma de energia para o erro nos deslocamentos $||\overline{e}||$ passa a ser

$$||\overline{e}|| = \left( \int_{\Omega} (\overline{\sigma} - \hat{\sigma})^t \, D^{-1} \, (\overline{\sigma} - \hat{\sigma}) \; d\Omega \right)^{1/2}. \tag{2.3}$$

Levando em consideração a discretização por elementos finitos e considerando um elemento específico $i$, a equação (2.3) pode ser reescrita como

$$||\overline{e}||_i = \left( \int_{\Omega_i} (\overline{\sigma} - \hat{\sigma})^t \, D^{-1} \, (\overline{\sigma} - \hat{\sigma}) \; |J| d\Omega_i \right)^{1/2}, \tag{2.4}$$

onde elementos isoparamétricos padrões foram assumidos; $\overline{\sigma}$ indica o campo de tensões recuperado, $|J|$ é o determinante da matriz Jacobiana, e $\Omega_i$ é o domínio do elemento.

A norma de energia para o erro pode ser avaliada sobre todo o domínio ou parte dele. A contribuição de todos os elementos na malha é dada por

$$||e||^2 = \sum_{i=1}^{m} ||e||_i^2, \tag{2.5}$$

onde $m$ representa o número total de elementos, $i$ refere-se ao elemento de domínio $\Omega_i$, e $\cup_{i=1}^{m} \Omega_i = \Omega$.

O erro relativo da norma de energia ($\eta$) para o domínio inteiro ou parte dele pode ser obtida como

$$\eta = \frac{||e||}{||u||}, \tag{2.6}$$

onde $||u||$ é a raiz quadrada do dobro da energia de deformação:

$$||u|| = \left( \int_{\Omega} \sigma^t D^{-1} \sigma \; d\Omega \right)^{1/2}. \tag{2.7}$$

O erro relativo da norma de energia ($\eta$) é usado em estratégias adaptativas, para avaliar o erro de discretização do problema em questão e para guiar para um refinamento e melhoria da malha e conseqüentemente dos resultados. Em uma estratégia adaptativa de refinamento baseada em uma extensão $h$, o estimador de erro irá definir como o modelo discreto vai ser refinado ou desrefinado. Um critério simples para se atingir o erro da solução dentro de um nível aceitável para todo do domínio pode ser estabelecido como

$$\overline{\eta} \leq \eta^*, \tag{2.8}$$

onde $\eta^*$ é o erro máximo permitido, e $\overline{\eta}$ é dado por

$$\overline{\eta} = \frac{\|\overline{e}\|}{\sqrt{\|\hat{u}\|^2 + \|\overline{e}\|^2}}, \tag{2.9}$$

onde $\|\hat{u}\|$ é a norma de energia obtida da solução por elementos finitos.

Um critério razoável utilizado para uma "malha ótima" é forçar que a norma de energia do erro ($\|e\|_i$) seja igualmente distribuída entre os elementos. Foi mostrado que este procedimento leva a malhas com altas taxas de convergência (Zienkiewicz e Zhu, 1987). Então, para um dado elemento $i$,

$$\|\overline{e}\|_i \leq \eta^* \left[ (\|\hat{u}\|^2 + \|\overline{e}\|^2)/m \right]^{1/2} = \overline{e}_m, \tag{2.10}$$

onde $m$ é o número total de elementos.

Definindo a razão de erro

$$\xi_i = \|\overline{e}\|_i / \overline{e}_m, \tag{2.11}$$

é óbvio que o refinamento é necessário se

$$\xi_i > 1. \tag{2.12}$$

Um procedimento mais eficiente consiste em designar completamente uma nova malha (re-geração) que satisfaça o requisito

$$\xi_i \leq 1, \tag{2.13}$$

no limite do refinamento da malha. Assumindo uma certa taxa de convergência (Zienkiewicz and Taylor, 1989), o valor da razão de erro $\xi_i$ pode ser usado para decidir o novo tamanho do elemento. Então

$$h = h_i / \xi_i^{1/p}, \tag{2.14}$$

onde $h_i$ é o tamanho inicial do elemento, $h$ é o tamanho final e $p$ é a ordem polinomial da função de aproximação. Este procedimento não considera efeitos de singularidade, como os gerados por cantos agudos ou trincas. Tratamentos para comportamentos singulares, com variados graus de sofisticação, podem ser encontrados em Zienkiewicz e Taylor (1989), Babuška e Szabó (1991), Babuška *et al.* (1994c), e Coorevits *et al.* (1994).

Um aspecto importante é que é interessante para uma estratégia adaptativa ser capaz de calcular novos tamanhos de elementos para vários tipos de elementos finitos. Neste trabalho, através do uso de programação orientada a objetos, é permitido tratar diferentes tipos de elementos, isto é, o método não é restrito a um elemento em específico.

## 2.2 *Técnicas de recuperação por patches*

Esta seção descreve procedimentos de recuperação usados para se obter o campo $\overline{\sigma}$, já que o campo de tensões exato $\sigma$ é geralmente desconhecido. Inicialmente, é descrita a técnica *SPR* baseada em sistemas de mínimos quadrados ponderados. Após isso, a técnica *REP* é descrita e as conexões entre os dois procedimentos de recuperação são apontadas.

### 2.2.1 *Superconvergent patch recovery (SPR)*

Um campo genérico (por exemplo, tensão) pode ser aproximado pela *expansão polinomial*

$$\overline{\sigma} = \mathcal{P}a, \tag{2.15}$$

onde $\mathcal{P}$ contém os termos polinomais apropriados, e $a$ é um conjunto de parâmetros desconhecidos. Note que esta expansão é usada para cada componente do tensor de tensões. Por exemplo, para problemas bidimensionais utilizando o elemento finito isoparamétrico quadrático de 8 nós (veja figura 2.1) [2], a aproximação abaixo é recomendada (Zienkiewicz and Zhu, 1992a):

$$\mathcal{P} = [1,\ x,\ y,\ x^2,\ xy,\ y^2], \tag{2.16}$$

$$a = [a_0,\ a_1,\ a_2,\ a_3,\ a_4,\ a_5\ ]^t. \tag{2.17}$$

Os coeficientes desconhecidos $a$ podem ser obtidos por um ajuste através do método dos mínimos quadrados ponderados da expansão polinomial (2.15) com relação aos valores de $\sigma$ obtidos da solução de elementos finitos nos pontos de amostragem, isto é, $\hat{\sigma}$. Pequenos grupos (*patches*) de elementos são usados para realizar ajustes locais de mínimos quadrados, e um peso ($w_i$) é considerado aqui para enfatizar a influência dos pontos de amostragem que estão mais próximos do nó que forma o *patch* (veja figura 2.1). Então,

$$w_i = 1/\rho_i^p, \tag{2.18}$$

onde $\rho_i$ é a distância Euclidiana entre o ponto de amostragem $i$ e o nó que forma o patch, e $p$ é um inteiro. O caso $p = 0$ corresponde ao *SPR* original (Zienkiewicz e

[2] Apesar de este capítulo se relacionar com estimativa de erro em três dimensões, a formulação vai ser desenvolvida para este *patch* bidimensional, sem perda de generalidade.

Zhu, 1992a), isto é, com peso uniforme. O uso de pesos maiores que zero pode ser efetivo para resolver problemas com gradientes bem elevados. Além disso, alivia o mau condicionamento de alguns problemas associados com a técnica *SPR* original.

Para ilustrar a recuperação superconvergente discreta, considere um *patch* de elementos contendo $m$ pontos de amostragem, como ilustrado na figura 2.1. Para um ponto de amostragem genérico $i$ neste *patch*, sejam $(x_i,\ y_i)$ as suas coordenadas Cartesianas em relação aos eixos globais. Então, o problema de mínimos quadrados ponderados é reduzido à minimização do seguinte funcional

$$\mathcal{G} = \sum_{i=1}^{m} w_i^2 \left[\hat{\sigma}(x_i, y_i) - \overline{\sigma}(x_i, y_i)\right]^2 . \tag{2.19}$$

Substituindo a equação (2.15) na equação (2.19), obtém-se

$$\mathcal{G} = \sum_{i=1}^{m} w_i^2 \left[\hat{\sigma}(x_i, y_i) - \mathcal{P}(x_i, y_i)a\right]^2 . \tag{2.20}$$

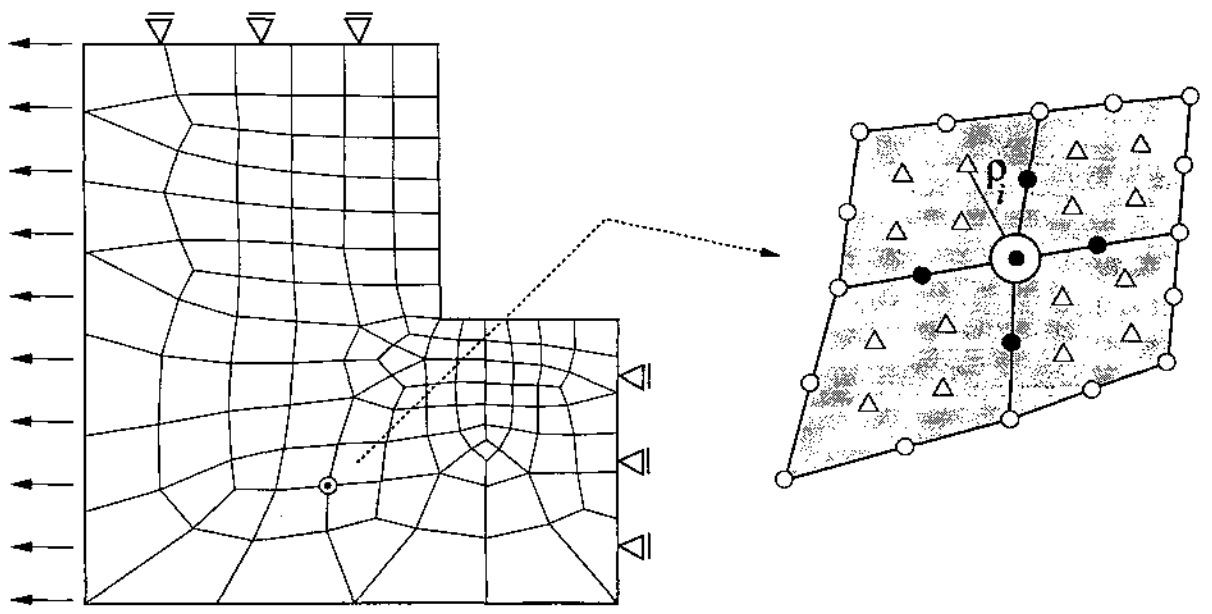



Figura 2.1: Notação da recuperação por *patch*: $\odot$ nó que forma o *patch*; $\triangle$ pontos de amostragem; $\bullet$ valores nodais determinados pelo procedimento de recuperação; $\circ$ pontos nodais; $\rho_i$ distância entre o ponto de amostragem $i$ e o nó que forma o *patch*; $\Omega$ denota o domínio do problema e $\Omega_P$ denota o domínio do *patch*.

O problema de minimização é resolvido fazendo-se $\partial \mathcal{G}/\partial a = 0$, o que leva ao seguinte conjunto de equações lineares:

$$Aa = b, \tag{2.21}$$

onde

$$A = \sum_{i=1}^{m} w_i^2 \mathcal{P}^t(x_i, y_i)\mathcal{P}(x_i, y_i), \tag{2.22}$$

$$b = \sum_{i=1}^{m} w_i^2 \mathcal{P}^t(x_i, y_i)\hat{\sigma}(x_i, y_i). \tag{2.23}$$

Finalmente, o sistema de equações dado por (2.21) pode ser resolvido para o vetor de incógnitas $a$ e conseqüentemente pode-se calcular as tensões usando-se a expansão polinomial apresentada em (2.15).

### 2.2.2 *Recovery by equilibrium in patches (REP)*

No *MEF* baseado em deslocamentos, sendo a equação de equilíbrio solucionada, o equilíbrio discreto de todos os elementos é assegurado. Então, cada *patch* isolado $\Omega_P$ (veja figura 2.1) estará também em equilíbrio. Usando este argumento, Boroomand and Zienkiewicz (1997) mostraram que

$$\int_{\Omega_P} B^t \sigma d\Omega = \int_{\Omega_P} B^t \hat{\sigma} d\Omega, \tag{2.24}$$

onde $B$ é a matriz de compatibilidade de deslocamentos (relaciona deformações internas com deslocamentos nodais), e como descrito anteriormente, $\hat{\sigma}$ é a solução de elementos finitos para as tensões. Para o caso elástico,

$$\hat{\sigma} = DB\hat{u}, \tag{2.25}$$

onde $\hat{u}$ representa os valores dos deslocamentos nodais e $D$ é a matriz elástica. Substituindo a equação (2.25) no lado direito da equação (2.24), tem-se que

$$\int_{\Omega_P} B^t \sigma d\Omega = \int_{\Omega_P} B^t DB\hat{u} d\Omega. \tag{2.26}$$

Considere agora uma representação suavizada do campo de tensões no *patch* $\Omega_P$, como dado pela equação (2.15). Usando esta representação no lado esquerdo da equação (2.26), obtém-se

$$\left(\int_{\Omega_P} B^t \mathcal{P} d\Omega\right) a = \left(\int_{\Omega_P} B^t DB d\Omega\right) \hat{u}. \tag{2.27}$$

Este conjunto de equações pode ser reescrito na forma matricial como

$$Ha = F_p, \tag{2.28}$$

onde

$$H = \int_{\Omega_p} B^t \mathcal{P} d\Omega, \tag{2.29}$$

e

$$F_p = \left( \int_{\Omega_P} B^t D B d\Omega \right) \hat{u}. \tag{2.30}$$

O cálculo dos coeficientes polinomiais pode ser feito por meio de um esquema de mínimos quadrados. Seja o erro $\delta$ definido por:

$$\delta = Ha - F_p. \tag{2.31}$$

Então, o funcional a ser minimizado é

$$\mathcal{F} = \delta^t \delta = (Ha - F_p)^t (Ha - F_p). \tag{2.32}$$

O problema de minimização é resolvido fazendo-se $\partial \mathcal{F} / \partial a = 0$, o que leva a

$$a = \left[ H^t H \right]^{-1} H^t F_p, \tag{2.33}$$

onde $H$ e $F_p$ são dados pelas equações (2.29) e (2.30), respectivamente.

Um importante fator afetando a acurácia do método é o número de elementos e a configuração do *patch*. Por questão de simplicidade e para ser possível uma comparação entre os dois métodos, o mesmo procedimento adotado para a construção dos *patches* no método *SPR* é também adotado para o método *REP*.

### 2.2.3 *Detalhes adicionais*

Alguns detalhes adicionais relativos às técnicas de recuperação por *patches* descritas devem ser considerados. Inicialmente, é importante mencionar que somente nós de canto de cada elemento finito formam *patches*. Isto acarreta que, em elementos finitos com graus de interpolação superior à linear (por exemplo, elementos quadráticos), os valores relativos aos nós de meio de lado são recuperados quando da formação de um *patch* para um nó de canto de um elemento ao qual esses nós pertecem. Este aspecto faz com que alguns nós possam ser recuperados por mais de um *patch*. Recomenda-se então que os resultados finais obtidos para um nó deste tipo sejam obtidos através de uma média dos resultados recuperados de todos os *patches* que influenciam o nó (Zienkiewicz and Zhu, 1987; Zienkiewicz e Zhu, 1992a). A figura 2.2 mostra exemplos de *patches* bidimensionais típicos e os nós que são recuperados nestes casos.

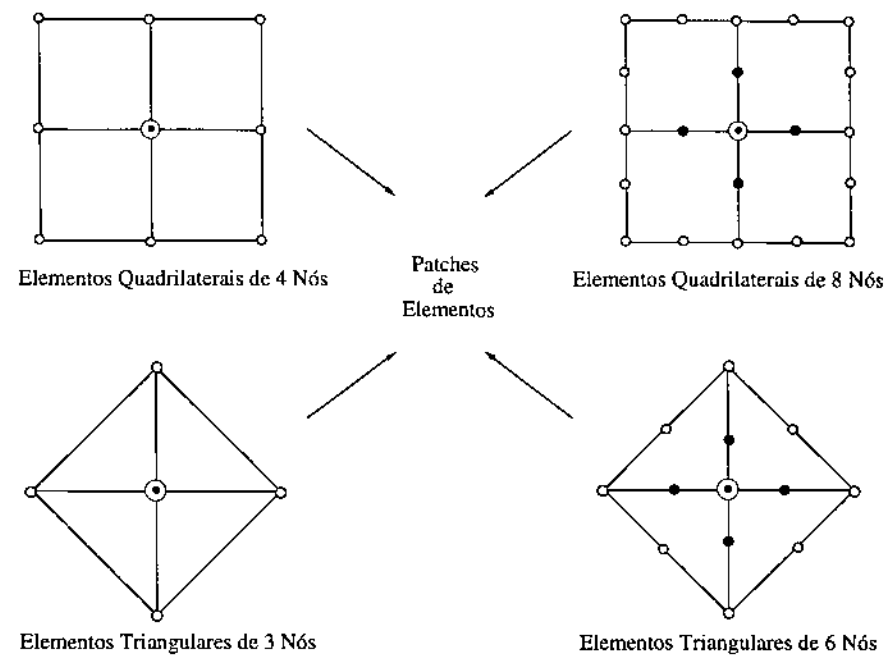



Figura 2.2: Exemplos de *patches* típicos: ⊙ nó que forma o *patch*; • valores nodais determinados pelo procedimento de recuperação; ○ pontos nodais.

Outra importante observação diz respeito à recuperação de valores relacionados a nós perto do contorno e na fronteira entre diferentes materiais. Para nós do contorno, possíveis *patches* a serem formados por estes nós podem ter somente um ou dois elementos. A figura 2.3 mostra um caso bidimensional linear em que esta situação se apresenta. Para estes casos, o número de elementos envolvidos no *patch* não é suficiente para determinar os parâmetros $a$ da expansão polinomial mostrada na expressão (2.15). Além disso, em nós localizados na interface de dois materiais distintos, as discontinuidades obtidas na recuperação considerando-se a formação de um possível *patch* que conteria elementos com diferentes materiais, não recomendam o uso de *patches* relacionados a esses nós. A solução para estes problemas é considerar que estes nós (de contorno e em interface de materiais diferentes) não formam *patches* e conseqüentemente os seus valores serão recuperados por *patches* internos nos quais esses nós façam parte (Zienkiewicz e Zhu, 1994). O nó $b$ da figura 2.3 não forma *patch* porque se localiza no contorno do modelo e só tem dois elementos adjacentes. O valor para esse nó é recuperado pelo *patch* interior relacionado à este nó. Já o nó $a$ não forma *patch* por se localizar na fronteira de dois materiais diferentes. Os valores para esse nó são recuperados pelos dois *patches* interiores mostrados na figura.

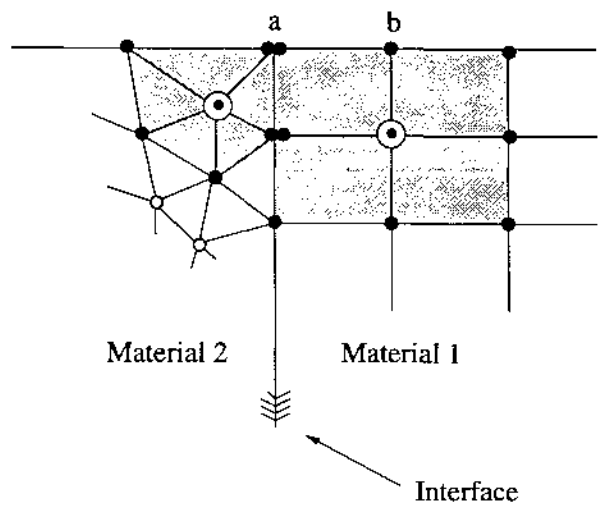



Figura 2.3: Exemplos de *patches* interiores para recuperação de valores no contorno e na fronteira de diferentes materiais: ⊙ nós que formam os *patches*; • valores nodais determinados pelo procedimento de recuperação; ○ pontos nodais.

Todos estes aspectos mencionados implicam que, para uma implementação eficiente das técnicas de recuperação propostas, estruturas de dados mais sofisticadas são necessárias. Por exemplo, cada nó do modelo deve conter nos seus dados locais uma lista de elementos adjacentes, que são usados para a formação do *patch* relacionado à esse nó e quando da recuperação do valor para esse nó e para outros possíveis nós deste *patch*. Com relação à identificação de que nós do *patch*, além obviamente do nó que formou esse *patch*, serão recuperados, a filosofia de programação orientada a objetos usada neste trabalho é importante nesta tarefa. Nesta implementação, existe uma classe responsável por indicar, dado um nó que formou um *patch* e o determinado tipo de elemento finito, que outros nós de meio de lado e localizados no contorno e em interface de diferentes materiais também serão recuperados. Isto é determinado baseado na lista de elementos adjacentes do nó que formou o *patch*.

## 2.3 *Aspectos da implementação numérica*

Com o objetivo de se obter um sistema computacional confiável e eficiente para simulações de elementos finitos, considerando estimativas de erro de discretização, um tratamento genérico para lidar com diferentes elementos finitos (por conseqüência, com diferentes formas e diferentes graus de funcões de interpolação) tem de ser levado em consideração. Nos últimos cinco anos, o programa *FEMOOP* (Martha *et al.*, 1996)

tem sido desenvolvido no Departamento de Engenharia Civil da Pontíficia Universidade Católica do Rio de Janeiro (PUC-Rio) para realizar análises por elementos finitos. Inicialmente, este programa foi desenvolvido na linguagem *C*, mas considerando uma disciplina de programação orientada a objetos para lidar com todas as diferentes formas e diferentes graus de funcões de interpolação. Este código foi usado em trabalhos anteriores deste autor para implementar um estimador de erro baseado em uma técnica de recuperação chamada *ZZ* (Zienkiewicz and Zhu, 1987), e para construir uma estratégia auto-adaptativa para problemas bidimensionais (Cavalcante Neto, 1994). Com o objetivo de se melhorar sua performance e funcionalidade, o programa foi migrado para uma programação formal orientada a objetos, utilizando a linguaguem *C++*. Isto permite uma manutenção mais fácil do código e uma expansão localizada, com uma menor chance de introdução de erros. As novas técnicas de recuperação mostradas neste trabalho foram implementadas no *FEMOOP*.

A tarefa mais importante no desenvolvimento de um programa orientado a objetos é a definição da estrutura das classes que o compõem. A possibilidade de re-uso do código e sua extensão dependem fortemente da qualidade desta organização. Com respeito à implementação do código de elementos finitos e dos algoritmos de estimativa de erro, várias classes foram definidas, como as *Fem*, *Node*, *Element*, *Material*, *AnModel*, *Gauss*, *Shape*, *LoadElem*, e *Error*. A comunicação entres essas classes é realizada através de instâncias (objetos) das classes. Esses objetos possuem, nos seus dados locais, ponteiros abstratos para os objetos das outras classes. A figura 2.4 mostra a organização de classes do *FEMOOP*.

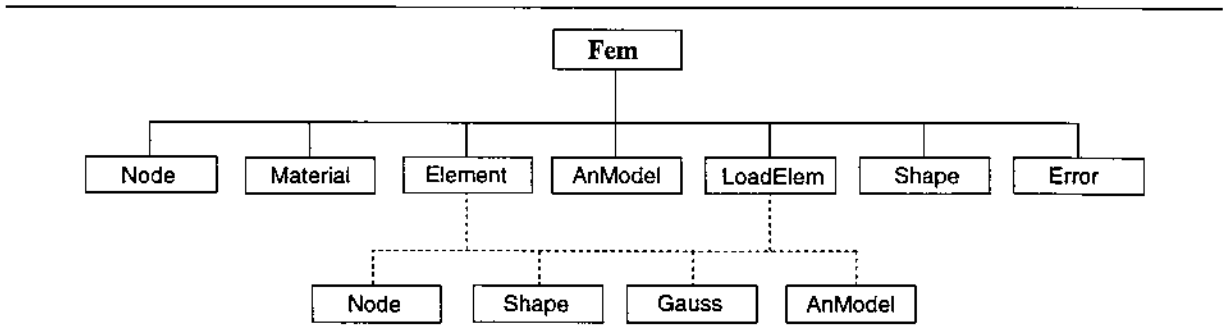



Figura 2.4: Organização de classes do *FEMOOP*.

A classe *Fem* representa a discretização numérica do modelo em elementos finitos. Ela contém referências para objetos de várias classes na organização: para uma lista de objetos da classe *Node* (os nós da malha), para uma lista de objetos da classe

*Element* (elementos da malha), para uma lista de objetos da classe *Material* (os grupos de materiais distintos usados), e uma referência para um objeto da classe *AnModel*. Esta última classe é responsável pelos aspectos específicos do tipo de análise sendo realizado (por exemplo, estado plano de tensão, estado plano de deformação, flexão de placas, cascas ou sólidos). A classe *Gauss* contém informações sobre o processo de integração numérica. Um objeto desta classe é associado com um ponto de integração de um elemento. A classe *Shape* contém os aspectos de geometria e das funcões de interpolação de um elemento. Esta classe é também responsável pelas técnicas de recuperação, já que os termos da expansão polinomial mostrados na equação (2.15) dependem do tipo de elemento finito, e são especificados nesta classe. A classe *LoadElem* consiste de elementos fictícios que transferem condições de contorno naturais dos elementos para os nós. Finalmente, a classe *Error* é responsável pelo cálculo da distribuição de erro relativa $||e||$, como mostrado nas equações (2.2) até (2.14).

A seguir, são mostrados alguns aspectos da implementação do *FEMOOP*, ilustrando as interconexões entre os diferentes módulos de todo o sistema. Pseudo-códigos específicos para o *SPR* e *REP* são fornecidos, que mostram como transformar a formulação que governa o problema para a sua forma matricial usando uma filosofia de programação orientada a objetos. A notação adotada para os pseudo-códigos é:

- As letras maísculas iniciais de cada pseudo-código indicam a classe que é responsável pela implementação do método correspondente, por exemplo, **FEM_StressRecoverySPR** significa que este método pertence à classe *Fem*;
- Cada pseudo-código tem uma linha de definição no início que define explicitamente os parâmetros de entrada e saída, que são indicados como **in** ou **out**, respectivamente;
- Palavras chaves dos algoritmos como **if**, **begin**, **end**, **foreach**, entre outras, são indicadas em negrito;
- Vetores e matrizes são denotados por $\{\cdot\}$ e $[\cdot]$, respectivamente, enquanto componentes dos vetores ou matrizes são denotados por meio de sub-escritos (por exemplo, $\{\cdot\}_{(i)}$, $[\cdot]_{(i,j)}$);
- Comentários são indicados em itálico (iniciados pelo símbolo //).

Somente os métodos mais importantes, necessários à compreensão da técnica de *SPR* são apresentados. Eles são apresentados nos pseudo-códigos 2.1 a 2.4, chamados:

- **FEM_StressRecoverySPR** - Método que devolve as tensões recuperadas para todos os nós da malha;
- **NODE_ComputeStressSPR** - Método que recebe um nó formador de *patch* e de-

volve as tensões recuperadas para este nó e outros possíveis nós de meio de lado e de contorno ou interface de materiais diferentes existentes neste *patch*;

- **ELEMENT_ComputeAbMatricesSPR** - Método que monta a contribuição da matriz $A$ e $b$ da expressão (2.22) e (2.23) para um elemento de um *patch* de um nó;
- **ELEMENT_RecoverStressSPR** - Método que recupera os valores para um dado nó através da expansão polinomial (2.15), dado os valores dos parâmetros $a$ calculados.

Vale a pena ressaltar que, para implementar a técnica *REP*, são necessárias modificações relativamente pequenas nos pseudo-códigos relativos à técnica *SPR* são necessárias. Estas modificações são, na sua maioria, relativas à implementação das equações (2.24) a (2.33), isto é, o cálculo das matrizes $H$ e $F$. Portanto, somente o pseudo-código responsável pelo cálculo das matrizes $A$ e $b$ (veja equações (2.15) a (2.23)) precisa ser modificado para se implementar a técnica *REP*. Então, de acordo com a notação adotada, o pseudo-código é renomeado para **ELEMENT_ComputeHFMatricesREP**, e é apresentado no pseudo-código 2.5.

```
FEM_StressRecoverySPR( out: [rec_stress], out: {counter} )
begin
// Get total number of nodes in the mesh
num_mesh_node ⟵ NODE_GetNumMeshNodes( );
// Get number of stress components
num_stress_comp ⟵ ANMODEL_GetNumStressComp( );
// Initialize recovered stress data structure
foreach nodal point (i), i = 1,...,num_mesh_node
    begin
    {counter}(i) ⟵ 0
        foreach stress component (j), j = 1,...,num_stress_comp
        begin
        [rec_stress](i,j) ⟵ 0.0
        end
    end
// Loop over the nodes to recover nodal stresses by means of the SPR procedure
foreach nodal point (i), i = 1,...,num_mesh_node
    begin
    NODE_ComputeStressSPR( i, [rec_stress], {counter} )
    end
// Update nodal recovered stresses
foreach nodal point (i), i = 1,...,num_mesh_node
    begin
    foreach stress component (j), j = 1,...,num_stress_comp
        begin
        [rec_stress](i,j) ⟵ [rec_stress](i,j) / {counter}(i);
        end
    end
end
```

Pseudo-código 2.1: *FEM_StressRecoverySPR*.

```
NODE_ComputeStressSPR( in: node, out: [rec_stress], out: {counter} )
begin
// Check to see if current node can be considered as a PATCH NODE:
// Get list of adjacent elements
num_adj_elem ⟵ ELEMENT_GetAdjElem( node, {adj_elem} );
if( num_adj_elem <= 2 ) return;
// Check to see if adjacent elements have different materials
material = ELEMENT_GetMaterial( {adj_elem}(1) );
foreach adjacent element (i), i = 2,...,num_adj_elem
    begin
    current_material = ELEMENT_GetMaterial( {adj_elem}(i) );
    if( material ≠ current_material ) return;
    end
// Check to see if current node is a corner node
if( SHAPE_CornerNode( node ) = false ) return;
// Get number of nodes per element
num_node ⟵ SHAPE_GetNumElemNodes( {adj_elem}(1) );
// Get number of polynomial terms
num_poly_term ⟵ SHAPE_GetNumPolyTerms( {adj_elem}(1) );
// Get number of stress components
num_stress_comp ⟵ ANMODEL_GetNumStressComp( );
// Get memory for vector of polynomial terms
{poly_term} ⟵ AllocVector( num_poly_term );
// Get memory for vector of recovered nodes
{rec_node} ⟵ AllocVector( num_node * num_adj_elem );
// Get memory for matrices [A], [a], and [b]
[A] ⟵ AllocMatrix( num_poly_term, num_poly_term );
[a] ⟵ AllocMatrix( num_poly_term, num_stress_comp );
[b] ⟵ AllocMatrix( num_poly_term, num_stress_comp );
// Loop over the elements of the current PATCH
foreach adjacent element (i), i = 1,...,num_adj_elem
    begin
    // Compute element contributions for matrices [A] and [b]
    ELEMENT_ComputeAbMatricesSPR( node, {adj_elem}(i), [A], [b] );
    end
// Solve linear system of equations with multiple right hand sides: [A] [a] = [b]
[a] ⟵ [A]^-1 [b]
// Loop over the elements of the current PATCH for defining the nodes to be recovered
foreach adjacent element (i), i = 1,...,num_adj_elem
    begin
    // Select nodes to be recovered
    ELEMENT_GetRecoveredNodes( {adj_elem}(i), num_rec_node, {rec_node} );
    end
// Recover stress values for each selected node
foreach recovered node (i), i = 1,...,num_rec_node
    begin
    // Get Cartesian coordinates of current recovered node
    Cart_coord ⟵ NODE_GetCartCoord( {rec_node}(i) );
    // Get polynomial terms for current recovered node
    SHAPE_GetPolyTerms( Cart_coord, {poly_term});
    // Recover stress values for current node
    ELEMENT_RecoverStressSPR( {rec_node}(i), num_poly_term, {poly_term}, [a],
                              [rec_stress], {counter});
    end
end
```

Pseudo-código 2.2: *NODE_ComputeStressSPR.*

```
ELEMENT_ComputeAbMatricesSPR( in: node, in: elem, out: [A], out: [b] )
begin
// Get number of nodes per element
num_node ⟵ SHAPE_GetNumElemNodes( elem );
// Get number of polynomial terms
num_poly_term ⟵ SHAPE_GetNumPolyTerms( elem );
// Get number of stress components
num_stress_comp ⟵ ANMODEL_GetNumStressComp( );
// Get memory for vector of stresses
{stress} ⟵ AllocVector( num_stress_comp );
// Get memory for vector of polynomial terms
{poly_term} ⟵ AllocVector( num_poly_term );
// Get memory for vector of shape functions
{shape} ⟵ AllocVector( num_node );
// Get memory for vector of nodal coordinates
{nodal_coord} ⟵ AllocVector( num_node );
// Get Cartesian coordinates of element nodes
SHAPE_GetCartCoord( elem, {nodal_coord} );
// Get Cartesian coordinates of the patch node
patch_coord ⟵ NODE_GetCartCoord( node );
// Get number of integration points of current element
num_int_point ⟵ GAUSS_GetNumIntPoints( elem );
// Loop over the integration points
foreach integration point (i), i = 1,...,num_int_point
    begin
    // Get local coordinates of current integration point
    local_int_point_coord ⟵ GAUSS_GetLocalCoord( i );
    // Get shape functions for current integration point
    SHAPE_GetShapeFunction( local_int_point_coord, {shape} );
    // Get stress values at integration point (from previous finite element analysis)
    {stress} ⟵ {FiniteElementStressData}
    // Compute Cartesian coordinates of the current integration point
    foreach element node (j), j = 1,...,num_node
        begin
        Cart_int_point_coord ⟵ Cart_int_point_coord + shape(j) * nodal_coord(j)
        end
    // Compute Euclidean distance between integration point and patch node
    distance ⟵ ComputeDistance( patch_coord, Cart_int_point_coord );
    // Compute weighting parameter: "p" is given by the user in the range from 0 to 4
    weight ⟵ 1/distance^p;
    // Get polynomial terms for current integration point
    SHAPE_GetPolyTerms( Cart_int_point_coord, {poly_term} );
    // Compute contribution for matrices [A] and [b]
    foreach polynomial term (j), j = 1,...,num_poly_term
        begin
        foreach polynomial term (k), k = 1,...,num_poly_term
            begin
            [A](j,k) ⟵ [A](j,k) + weight^2 * poly_term(j) * {poly_term}(k);
            end
        foreach stress component (k), k = 1,...,num_stress_comp
            begin
            [b](j,k) ⟵ [b](j,k) + weight^2 * poly_term(j) * {stress}(k);
            end
        end
    end
end
```

Pseudo-código 2.3: *ELEMENT_ComputeAbMatricesSPR.*

```
ELEMENT_RecoverStressSPR( in: node, in: num_poly_term, in: {poly_term}, in: [a],
                          out: [rec_stress], out: {counter} )
begin
// Get number of stress components
num_stress_comp ⟵ ANMODEL_GetNumStressComp( );
// Update number of times that current node has been recovered
{counter}(node) ⟵ {counter}(node) + 1;
// Loop over the stresses components
foreach stress component (i), i = 1,...,num_stress_comp
    begin
    // Loop over the polynomial terms
    foreach polynomial term (j), j = 1,...,num_poly_term
        begin
        [rec_stress](node,i) ⟵ [rec_stress](node,i) + {poly_term}(j) * [a](j,i);
        end
    end
end
```

Pseudo-código 2.4: *ELEMENT_RecoverStressSPR.*

```
ELEMENT_ComputeHFMatricesREP( in: node, in: elem, out: [H], out: [F]
begin
// Get number of nodes per element
num_node ⟵ SHAPE_GetNumElemNodes( elem );
// Get number of degrees of freedom per node
num_dof_node ⟵ ANMODEL_GetNumDofNode( node );
// Compute number of degrees of freedom of current element
num_dof_elem ⟵ num_node * num_dof_node;
// Get number of polynomial terms
num_poly_term ⟵ SHAPE_GetNumPolyTerms( elem );
// Get dimension of matrix [B] (strain-displacement matrix)
dim_B_mat ⟵ ANMODEL_GetDimBMatrix( );
// Get total number of components to be recovered
num_rec_comp ⟵ num_poly_term * dim_B_mat;
// Get memory for element stiffness matrix
[elem_stiff] ⟵ AllocMatrix( num_dof_elem, num_dof_elem );
// Get memory for element displacement vector
{elem_disp} ⟵ AllocVector( num_dof_elem );
// Get memory for vector of shape functions
{shape} ⟵ AllocVector( num_node );
// Get memory for vector of nodal coordinates
{nodal_coord} ⟵ AllocVector( num_node );
// Get Cartesian coordinates of element nodes
SHAPE_GetCartCoord( elem, {nodal_coord} );
// Get memory for matrix [B] (strain-displacement matrix)
[B] ⟵ AllocMatrix( dim_B_mat, num_dof_elem );
// Get memory for matrix [P] (matrix with the polynomial terms)
[P] ⟵ AllocMatrix( dim_B_mat, num_rec_comp );
// Get number of integration points of current element
num_int_point ⟵ GAUSS_GetNumIntPoints( elem );
// Loop over the integration points
foreach integration point (i), i = 1,...,num_int_point
    begin
    // Get local coordinates of current integration point
    local_int_point_coord ⟵ GAUSS_GetLocalCoord( i );
    // Get shape functions for current integration point
    SHAPE_GetShapeFunction( local_int_point_coord, {shape} );
    // Compute Cartesian coordinates of the current integration point
    foreach element node (j), j = 1,...,num_node
        begin
        Cart_int_point_coord ⟵ Cart_int_point_coord + shape(j) * nodal_coord(j)
        end
    // Get [B] matrix at current integration point
    ELEMENT_GetBMatrix( local_int_point_coord, [B] );
    // Get [P] matrix at current integration point
    ELEMENT_GetPMatrix( Cart_int_point_coord, [P] );
    // Compute contribution of current element to matrix [H]
    [H] ⟵ [H] + [B]^T * [P];
    end
// Compute current element stiffness matrix
[elem_stiff] ⟵ ELEMENT_ComputeElemStiffMat( elem );
// Get element displacements (from previous finite element analysis)
{elem_disp} ⟵ {FiniteElementDisplacementData}
// Compute contribution of current element to matrix [F]
[F] ⟵ [F] + [elem_stiff] * {elem_disp};
end
```

Pseudo-código 2.5: *ELEMENT_ComputeHFMatricesREP.*

## 2.4 *Extensão tridimensional*

O maior desafio para a implementação das técnicas de recuperação *SPR* e *REP* descritas, e conseqüentemente da estimativa de erro de discretização para problemas tridimensionais, é a formação de *patches* em três dimensões. A formulação apresentada pode ser idealmente aplicada em duas e três dimensões, mas na prática tem sido aplicada na maioria das vezes somente em problemas bidimensionais, devido exatamente a essa dificuldade de formação de um *patch* tridimensional e também a outras restricões, como a disponibilidade de algoritmos para geração de malhas volumétricas, por exemplo.

Conforme descrito na seção 2.2, os processos de recuperação assumem que os valores a serem recuperados (por exemplo, a tensão $\overline{\sigma}$) podem ser aproximados pela *expansão polinomial* descrita na equação (2.15) e são válidas em um *patch* de elementos envolvendo um determinado nó considerado a cada vez. Este *patch* representa a união de elementos contendo esse nó, em uma maneira similar à usada no *patch test* (Zienkiewicz e Taylor, 1989). Para problemas tridimensionais conseqüentemente, como no caso bidimensional, os termos polinomiais apropriados de $\mathcal{P}$ da expansão polinomial (2.15) são exatamente iguais ao número de termos polinomiais usados nas funções de forma que definem deslocamentos no interior do elemento a partir dos deslocamentos nodais. Por exemplo, para um elemento finito tetraédrico isoparamétrico quadrático de 10 nós, $\mathcal{P}$ é dado por:

$$\mathcal{P} = [1,\ x,\ y,\ z,\ x^2,\ xy,\ y^2,\ yz,\ z^2,\ zx]. \tag{2.34}$$

Entretanto, como mencionado anteriormente, somente nós de canto de cada elemento finito formam *patches*. Isto acarreta que, em elementos com graus de função de interpolação superior à linear, os valores relativos aos nós que não são de canto são recuperados quando da formação de um *patch* para um nó de canto de um elemento ao qual esses nós pertencem.

Portanto, para cada *patch* formado, além do nó que forma o *patch*, podem ser recuperados os valores das tensões para um número de outros nós dentro deste *patch*. Na figura 2.5 são mostrados exemplos de *patches* tridimensionais para elementos tetraédricos lineares e quadráticos. Os mesmos conceitos se aplicam a outros tipos de elementos finitos tridimensionais, como os elementos hexaédricos (*bricks*), por exemplo.

O conceito de programação orientada a objetos, no qual o *FEMOOP* se baseia, facilita a definição e implementação da estimativa de erro para problemas tridimensionais, já que a maioria dos métodos mostrados na seção anterior, implementados

inicialmente para duas dimensões, podem ser utilizados também para três dimensões, ou podem ser facilmente estendidos.

Na verdade, para os *patches* tridimensionais mostrados na figura 2.5, o que muda em relação ao caso bidimensional é a implementação da definição de, para um dado determinado tipo de elemento e um determinado nó que formou um *patch*, saber que outros nós deste elemento serão também recuperados. Portanto, o método chamado **ELEMENT_GetRecoveredNodes**, mostrado no pseudo-código 2.2 do método **NODE_ComputeStressSPR**, avalia em um determinado elemento do *patch*, para um determinado nó deste elemento que formou esse *patch*, que outros nós terão seus valores recuperados.

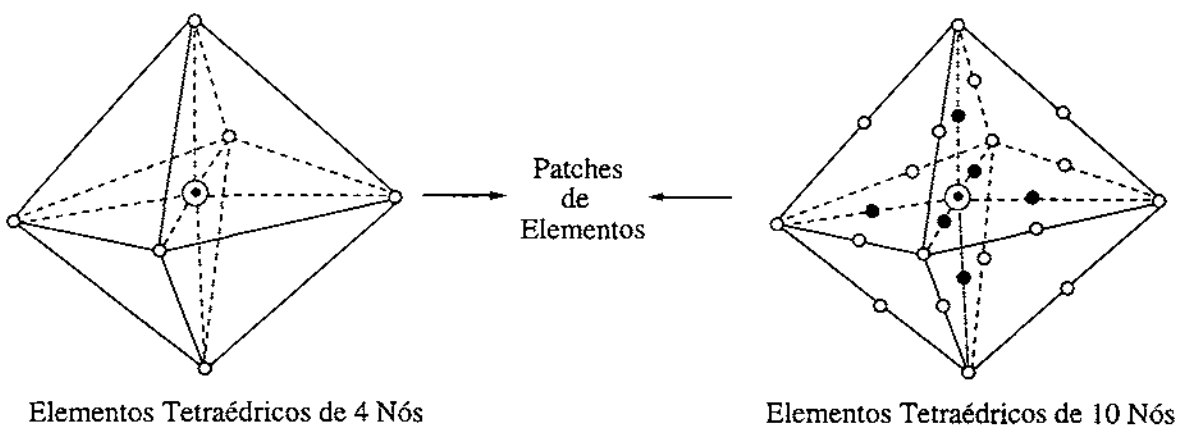
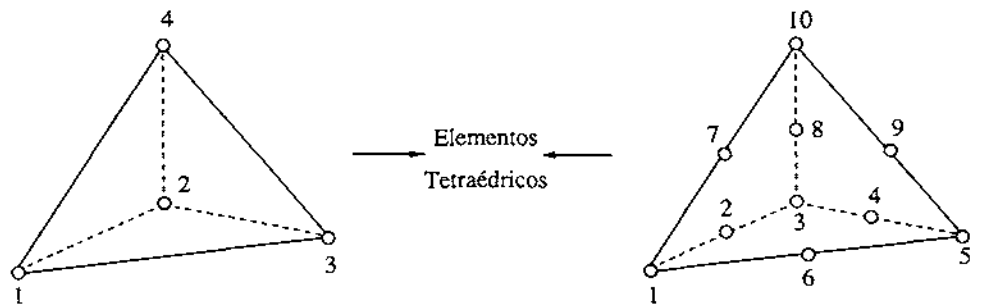



Figura 2.5: Notação da recuperação por *patch*: ⊙ nó que forma o *patch*, • valores nodais determinados pelo procedimento de recuperação e ○ pontos nodais.

Observando-se a figura 2.6, verifica-se que, para elementos tetraédricos lineares, apenas o nó que forma o *patch* tem seu valor recuperado, enquanto que para elementos tetraédricos quadráticos, para um dado nó de canto, três outros nós de meio de aresta são também recuperados. Na implementação, é necessário apenas que exista um método para cada tipo de elemento que forneça, para um dado nó de canto, os nós internos a serem recuperados, se eles existirem. A classe *Shape*, que trata os diferentes tipos de elementos finitos, é responsável pela implementação desse método. Com isso, os métodos principais mostrados na seção anterior podem ser utilizados sem nenhuma modificação e tem-se uma implementação computacional eficiente e rápida das técnicas de *SPR* e *REP*, e conseqüentemente da estimativa de erro para problemas tridimensionais. Na atual implementação, os seguintes elementos finitos sólidos já foram implementados:

elemento tetraédrico linear de 4 nós (TET4), elemento tetraédrico quadrático de 10 nós (TET10), elemento hexaédrico linear de 8 nós (BRICK8) e elemento hexaédrico quadrático de 20 nós (BRICK20).




Elemento Tetraédrico de 4 Nós

| Nó Formador de Patch | Nós Recuperados |
|---|---|
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |

Elemento Tetraédrico de 10 Nós

| Nó Formador de Patch | Nós Recuperados |
|---|---|
| 1 | 1 2 6 7 |
| 3 | 3 2 4 8 |
| 5 | 5 4 6 9 |
| 10 | 10 7 8 9 |

Figura 2.6: Nós a serem recuperados para elementos tetraédricos.

# 3

# Geração de malha volumétrica

Este capítulo descreve um algoritmo para geração de malhas volumétricas não estruturadas de tetraedros para domínios tridimensionais de forma arbitrária. O algoritmo também funciona para domínios com uma ou múltiplas trincas. Estas trincas podem ser internas ou localizadas na superfície do domínio.

O algoritmo foi desenvolvido para atender quatro requisitos específicos. Primeiro, o algoritmo deve produzir elementos de boa forma, evitando a geração de elementos com má qualidade, sempre que possível. Apesar de o algoritmo não garantir limites na qualidade dos elementos, cuidado é tomado a cada passo para gerar os melhores elementos possíveis. Além disso, o algoritmo contém um procedimento *a posteriori* que procura melhorar localmente a qualidade dos elementos gerados. Isto é baseado em um estudo aprofundado sobre parâmetros que qualificam a forma de um elemento finito tetraédrico. Observações empíricas, descritas na seção 3.2, mostram que o algoritmo é muito bem sucedido no aspecto de qualidade geométrica dos elementos gerados.

O segundo requisito é que o algoritmo deve gerar uma malha volumétrica que se conforme com uma dada malha triangular existente no contorno do domínio. Isto é importante porque permite uma nova geração de malha ocorrer localmente em uma região perto de uma trinca se propagando. Isto é, um grupo relativamente pequeno de elementos perto da trinca pode ser eliminado, criando um vazio na malha. A trinca se propaga, e então este algoritmo pode ser usado para gerar novos elementos que preencham este vazio, e que se conformam com os elementos que não foram removidos. A região onde uma nova malha foi gerada é pequena e localizada, levando a uma geração de malha rápida e, para problemas não-lineares, minimizando a quantidade de informações de estado que precisam ser mapeadas entre as malhas nova e velha. Além disso, em um procedimento similar, isto pode ser aplicado em problemas adaptativos, onde apenas os elementos de uma região do domínio com um alto erro de discretização indicado por uma análise em um passo anterior são eliminados, criando similarmente um vazio, e este vazio é então preenchido com novos elementos que respeitem a distribuição de erro desejada.

O algoritmo entretanto, não é restrito a pequenas regiões próximas a trincas ou com alto erro de discretização. Os exemplos mostrados nas seções 3.2 e 3.3 demonstram o seu eficiente uso para geração de malhas volumétricas em regiões de tamanho considerável.

Muitos outros algoritmos encontrados na literatura descrevem a geração de malha no contorno do domínio junto com a geração da malha volumétrica. Como citado acima, um dado de entrada necessário para este algoritmo é uma malha de contorno fornecida. Não se considera isso uma limitação significante porque existem um grande número de técnicas de geração de malhas em superfície que podem ser usadas para gerar a malha de contorno requisitada (Lau e Lo, 1996; Lewis *et al.*, 1996b; Gomes Coelho, 1998). Neste trabalho, as malhas de contorno foram geradas pelo programa *FRANC3D* (Martha 1989; Wawrzynek, 1991), desenvolvido pelo *CFG* (Grupo de Fratura da Universidade de Cornell/EUA), grupo ao qual este trabalho é relacionado. Entretanto, o algoritmo pode ser usado para qualquer malha de contorno fornecida. Além disso, este requisito permite a combinação deste algoritmo com outros tipos de algoritmos, como mapeamentos por exemplo, em modelos que contenham regiões que compartilham um mesmo contorno, onde a malha de superfície deve ser compatível.

O terceiro requisito do algoritmo é que ele possa gerar uma boa transição entre regiões com diferentes tamanhos de elementos. Em uma análise de fraturamento, por exemplo, não é incomum que elementos perto da trinca tenham um tamanho característico duas ordens de magnitude menor do que o de outros elementos no domínio. Alguns outros algoritmos encontrados na literatura funcionam melhor quando todos os elementos gerados têm um tamanho característico semelhante. Isto é claramente inaceitável no caso de trincas, e o algoritmo proposto foi desenvolvido para ser capaz de gerar malhas com boa transição entre diferentes tamanhos de elementos. Como no segundo requisito, este aspecto é também muito importante em análises adaptativas, onde regiões com altos gradientes devem ser muito refinadas, enquanto que em outras regiões os elementos podem ter tamanhos maiores. Um algoritmo capaz de gerar malhas com boa transição é fundamental neste contexto.

O quarto requisito surge porque trincas são normalmente idealizadas como não tendo volume, isto é, as superfícies representando os dois lados de uma trinca são distintas, mas geometricamente coincidentes. Isto significa que nós em lados opostos de faces da trinca têm coordenadas idênticas. O algoritmo deve ser capaz de discriminar entre os nós e escolher aquele que se localiza na face correta da trinca.

Este capítulo apresenta uma descrição detalhada dos passos do algoritmo para geração de malhas volumétricas. Além disso, uma análise da qualidade das malhas geradas é feita, baseada em medidas estatísticas para uma série de exemplos. Finalmente, medidas empíricas do desempenho assintótica do algoritmo são descritas.

## 3.1 *Descrição do algoritmo*

O algoritmo proposto incorpora aspectos de técnicas para geração de malhas bem conhecidas na literatura, e define alguns passos originais adicionais. O algoritmo se baseia em uma técnica de avanço da fronteira (*advancing front technique*), mas usa também uma técnica de decomposição espacial recursiva (*recursive spatial decomposition*), no caso uma árvore octária (*octree*), para desenvolver diretrizes locais usadas para definir o tamanho dos elementos a serem gerados. A técnica de avanço da fronteira usada neste algoritmo é baseada em um procedimento padrão encontrado na literatura (Peraire *et al.*, 1988; Lohner e Parikh, 1988; Moller e Hansbo, 1995), com duas fases adicionais, propostas para garantir a geração de uma malha volumétrica válida para virtualmente qualquer domínio. Para melhorar a qualidade das malhas geradas (no que diz respeito à forma dos elementos), um procedimento de melhoria local *a posteriori* é usado.

Os dados de entrada do algoritmo são descritos por uma lista de nós definidos por suas coordenadas, e uma lista de faces definidas por sua conectividade nodal. Este tipo de estrutura de dados tem alguns aspectos a serem considerados: pode representar geometrias de qualquer forma, incluindo furos, cavidades e trincas de uma maneira simples, e pode ser facilmente incorporado em qualquer sistema de elementos finitos.

O algoritmo é organizado nos seguintes passos, que serão descritos nesta seção:

**I. Geração da octree**

- **a. Geração baseada na malha do contorno**
- **b. Refinamento para forçar um tamanho de célula máximo**
- **c. Refinamento para forçar disparidade de tamanho mínima entre células adjacentes**

**II. Procedimento de avanço da fronteira**

- **a. Geração de elementos baseada em geometria**
- **b. Geração de elementos baseada em topologia**
- **c. Geração de elementos baseada em procedimento de "volta-passo" com eliminação de faces**

**III. Melhoria local da malha**

- **a. Suavização Laplaciana com testes de validação**
- **b. Avaliação de qualidade e procedimento de "volta-passo" local com eliminação de elementos**

### 3.1.1 *Geração da octree*

Como mencionado anteriormente, a funcionalidade principal da *octree* é desenvolver diretrizes locais usadas para definir o tamanho dos elementos tetraédricos a serem gerados durante o procedimento de avanço da fronteira. A distribuição do tamanho dos elementos tetraédricos através do domínio é deduzida pelo tamanho dos elementos triangulares na malha de contorno fornecida como dado de entrada.

A geração da *octree* envolve três passos. Em um primeiro passo, a *octree* é inicializada baseada nos dados de entrada. Em outros dois passos, a *octree* é posteriormente refinada. Um exemplo hipotético bidimensional, mostrado na figura 3.1, é usado aqui para ilustrar o processo de geração da *octree*, que neste caso é mostrada como uma árvore quaternária (*quadtree*). Este modelo possui uma aresta de trinca no seu lado direito. Na ponta da trinca, o contorno é contraído como se tivesse elementos de ponta de trinca especialmente colocados. O modelo de contorno apresenta um grau de refinamento gradativamente maior do lado esquerdo para o lado direito. Pode-se facilmente deduzir através do exemplo bidimensional os procedimentos análogos que se desenvolvem em três dimensões, que são difíceis de serem mostrados em uma figura.

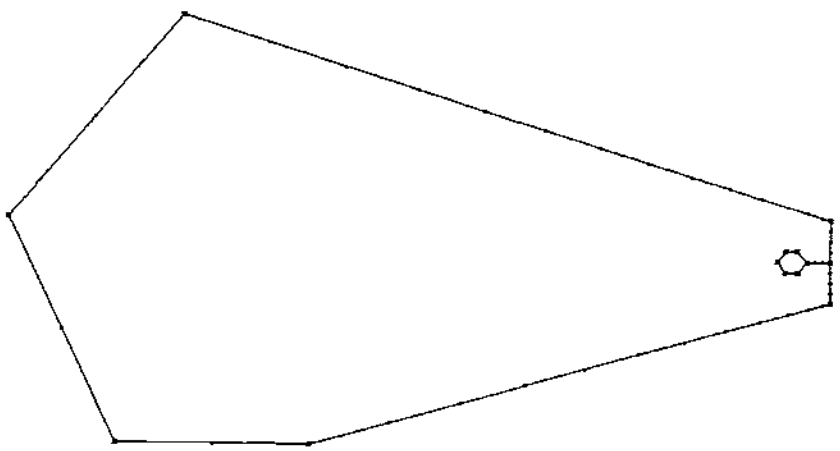


Figura 3.1: Modelo hipotético bidimensional e seu refinamento do contorno.

#### 3.1.1.1 *Geração da octree baseada na malha do contorno*

Inicialmente, um cubo envolvente é criado baseado no máximo intervalo de qualquer uma das três coordenadas cartesianas dos nós dos dados de entrada. Este cubo é a célula raiz

da *octree*, conforme ilustado na figura 3.2. No primeiro passo de refinamento da *octree*, cada face da malha de contorno fornecida é usada para determinar a profundidade local da subdivisão. A célula da *octree* contendo o centróide de cada face triangular da malha de contorno é determinada. Se a área da face da célula [1] é maior do que a área da face do contorno, então esta célula é subdividida em oito células menores. Este processo é repetido recursivamente e acaba quando a área da face da célula é menor do que um fator da área da face do contorno. Nesta implementação, um fator de 0.4 foi usado. O uso deste fator é recomendado em alguns trabalhos baseados em *octree* encontrados na literatura (Shephard e Georges, 1991), para evitar o excessivo refinamento quando da construção da *octree*. Este processo é repetido para todas as faces do contorno do modelo fornecidas. O resultado é ilustrado na figura 3.2 para o exemplo bidimensional.

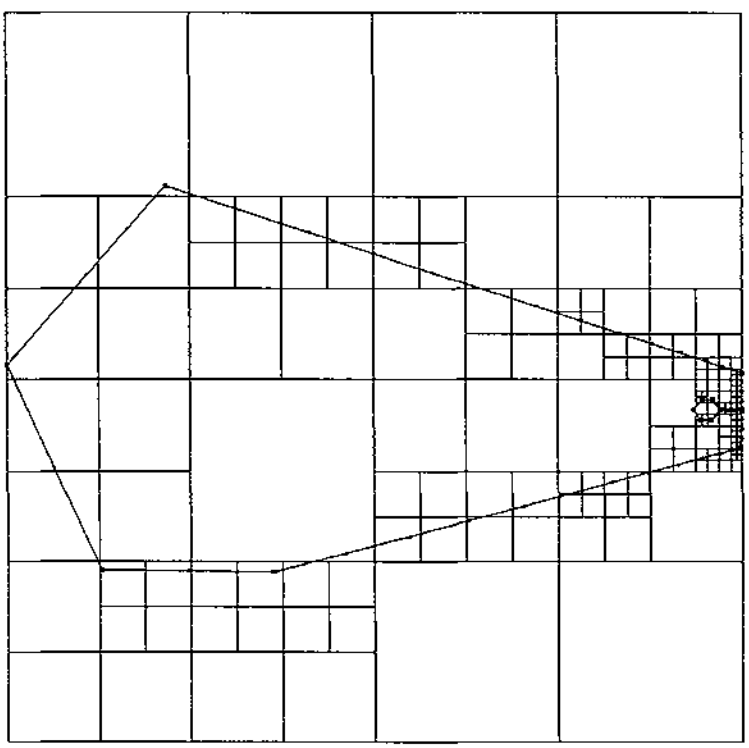


Figura 3.2: Quadtree inicial do modelo bidimensional.

### 3.1.1.2 *Refinamento da octree para forçar um tamanho de célula máximo*

O passo anterior pode deixar grandes células da *octree* no interior do domínio. Neste passo, a *octree* é refinada para garantir que nenhuma célula no interior do domínio é

[1] Cada célula da *octree* é um cubo com seis faces iguais, então qualquer face pode ser usada.

maior do que a maior célula no contorno. Isto evitará a geração de elementos excessivamentes grandes no interior do domínio. A figura 3.3 mostra a quadtree resultante após esta operação para o exemplo bidimensional.

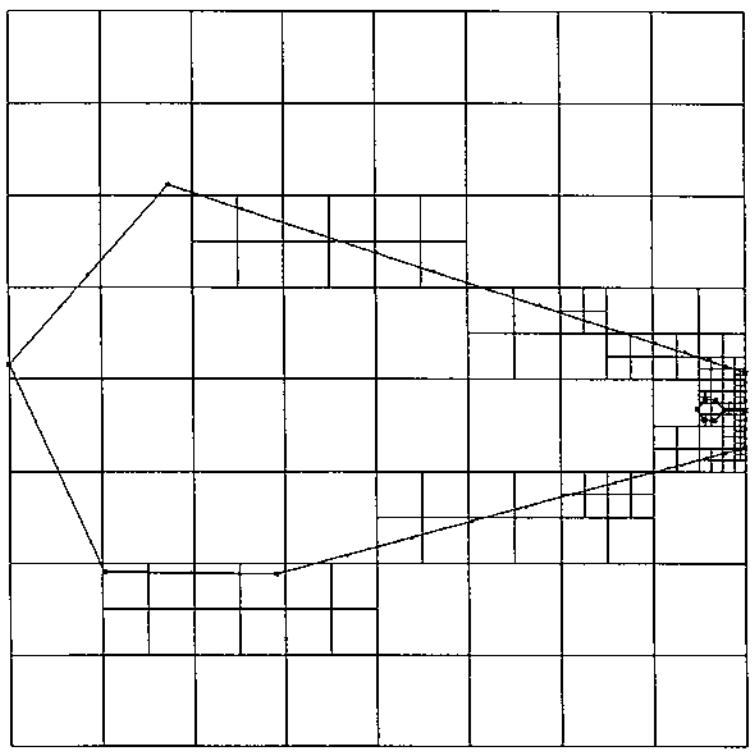


Figura 3.3: Quadtree após forçar o maior tamanho de célula no contorno.

### 3.1.1.3 *Refinamento da octree para forçar disparidade de tamanho mínima*

A *octree* é posteriormente processada para forçar um único nível de diferença entre células vizinhas. Isto força uma transição natural entre regiões com diferentes graus de refinamento. Esta operação é realizada percorrendo a *octree* e examinando o nível de refinamento entre células adjacentes. Se a diferença é mais do que um nível, as células apropriadas são refinadas até que o critério seja satisfeito. A figura 3.4 mostra a quadtree gerada para o exemplo bidimensional após este procedimento. Pode-se notar que as células próximas à ponta da trinca foram afetadas por esta fase, se comparada com a quadtree da fase anterior.

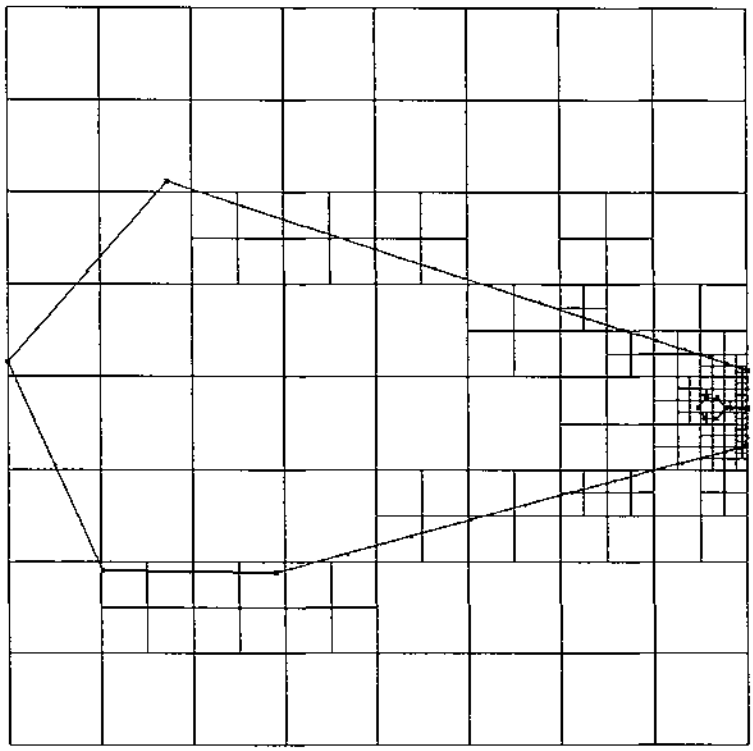


Figura 3.4: Quadtree após forçar disparidade de tamanho mínima.

### 3.1.2 *Procedimento de avanço da fronteira*

O procedimento de avanço da fronteira, descrito nessa seção, começa com a superfície que limita o domínio a ser preenchido com uma malha volumétrica. Elementos volumétricos são "extraídos" ou "cortados" do domínio um a cada vez. Quando cada elemento é extraído, a superfície limitante é atualizada e o processo é repetido. O procedimento termina quando uma malha foi gerada para o domínio inteiro, ou quando uma ou mais cavidades internas, onde malhas não foram geradas, permanecem, de onde elementos válidos não podem ser extraídos.

Neste algoritmo, o processo de avanço da fronteira é dividido em três fases para garantir a geração de malhas volumétricas válidas. Em uma primeira fase, uma geração de elementos baseada em geometria é tentada para gerar elementos de formas ótimas. Após esta fase ideal ser exaurida, e elementos ótimos não poderem ser mais criados, uma geração de elementos baseada em topologia é realizada, tentando-se criar elementos válidos, mas não necessariamente de boa forma, na região restante. Na última fase, um procedimento de "volta-passo" (*back-tracking*) é usado para eliminar algumas faces dos elementos que estão impedindo o algoritmo de completar a malha.

#### 3.1.2.1 *Geração de elementos baseada em geometria*

Idealmente, a malha volumétrica inteira seria gerada na fase baseada em geometria. Isto depende da geometria e da topologia do contorno do modelo fornecido e é fortemente relacionado com a qualidade da malha de contorno fornecida, no que diz respeito à forma das faces triangulares.

##### 3.1.2.1.1 *Listas de contração de contorno*

O processo começa com a criação de uma fronteira de avanço inicial, que é formada pela malha de contorno fornecida como dado de entrada. A malha de contorno corrente é armazenada em duas listas duplamente encadeadas separadas. A primeira lista é uma lista de faces ativas, com as faces da malha de contorno que não foram ainda usadas em uma tentativa de gerar tetraedros válidos. A outra é uma lista de faces rejeitadas, isto é, com as faces que falharam na geração de elementos para a fase corrente. Inicialmente, todas as faces da malha de contorno fornecida são inseridas na primeira lista.

A lista de faces ativas é ordenada pela área das faces. Isto garante que a primeira face selecionada será sempre a face com a menor área. Isto tem sido recomendado por outros autores (Peraire *et al.*, 1988) para prevenir que elementos grandes penetrem em regiões com faces de áreas pequenas.

Também é conveniente para alguns passos do algoritmo ter-se uma estrutura de dados adicional que contém a lista de faces do contorno adjacentes para cada nó da fronteira de avanço corrente. Esta estrutura de dados é inicializada para todos os nós da malha de contorno fornecida. A estrutura de dados é atualizada quando o procedimento de contração do contorno progride.

##### 3.1.2.1.2 *Geração de elementos ótimos*

Na fase de geração de elementos baseada em geometria, a malha de contorno corrente avança tentando formar tetraedros baseados principalmente em considerações geométricas. A cada passo, uma face triangular do contorno, referenciada como *face base* é escolhida da lista de faces ativas. Esta face possui a menor área na fronteira corrente e sua normal aponta para o interior da região a ser gerada a malha.

Os critérios para gerar um tetraedro nesta fase estão explicados por meio das figuras 3.5 e 3.6. A figura 3.5 ilustra o caso bidimensional e a figura 3.6 mostra sua equivalência tridimensional. O procedimento é dividido nos seguintes passos:

- A localização ótima $N1$ para o vértice do tetraedro a ser formado é determinada com a ajuda da *octree*. A célula da *octree* contendo o centróide da face base é determinada. O ponto ótimo $N1$ se localiza na linha perpendicular à face base passando pelo seu centróide, $M$. A distância do ponto ótimo para o centróide da face base é igual ao tamanho da célula da *octree*.

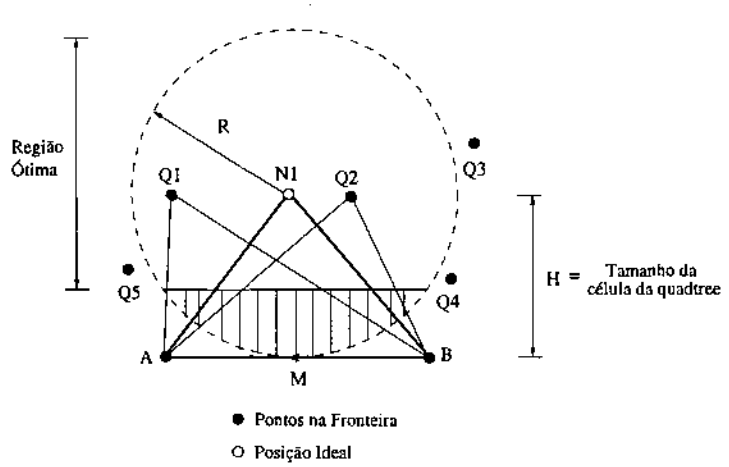



Figura 3.5: Região ótima de vértice candidato de um triângulo em duas dimensões.

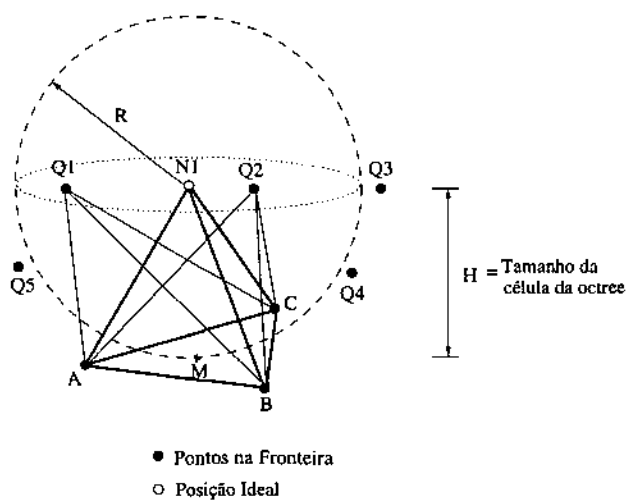



Figura 3.6: A determinação de um tetraedro em três dimensões.

- O ponto ótimo define uma região ótima onde o vértice do tetraedro a ser formado está localizado. Esta região é um setor de esfera cujo centro é o ponto ótimo e cujo raio é proporcional ao tamanho da célula da *octree*. Na implementação corrente, um

fator igual a 1 foi adotado para essa proporção. Esta esfera define um limite superior para a distância entre o vértice alvo do tetraedro e o centróide da face base. Um limite inferior é definido para garantir que o tetraedro a ser gerado terá volume maior que o menor volume aceitável. Nesta implementação este limite inferior é definido pelo tetraedro com altura igual a 1/10 da distância entre $N1$ e $M$. A região ótima é usada por duas razões. Primeiro, para assegurar qualidade na forma de todos os elementos a serem gerados e, segundo, para assegurar que novos nós internos somente sejam criados quando é estritamente necessário e sempre em boas posições.

- Se nenhum nó existente está dentro da região ótima, um novo nó é inserido na localização ótima $N1$ e um elemento é gerado usando este nó. Se somente um nó existe nesta região, este nó é usado para formar o elemento. Se mais de um nó são encontrados na região ótima, eles são classificados de acordo com o ângulo sólido que eles formarão com a face base. Uma lista auxiliar é usada para selecionar eficientemente esses nós. O nó que formará o maior ângulo sólido (veja figura 3.7) é usado para gerar o elemento.

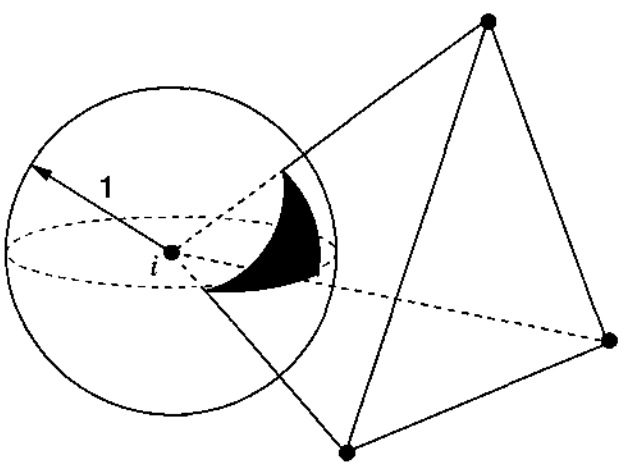



Figura 3.7: Definição de um ângulo sólido para um vértice $i$ de um tetraedro.

- Testes geométricos adicionais são realizados para assegurar que as faces do novo elemento não interceptam nenhuma das faces existentes na fronteira de avanço. Neste caso, o elemento é rejeitado. Estes testes são tarefas computacionalmente muito caras do algoritmo e alguns filtros são usados para melhorar a sua eficiência. Por exemplo, para cada face existente na fronteira de avanço cuja interseção deve ser testada contra as faces do novo elemento, os retângulos envolventes de ambas as

faces são computados. Se esses retângulos não se interceptam, os procedimentos de verificação de interseção entres essas duas faces não são necessários.

- Como mencionado anteriormente, para problemas de fraturamento poderão existir dois ou mais nós com as mesmas coordenadas. A figura 3.8 ilustra este caso para um exemplo bidimensional no lado esquerdo. Esta figura também mostra uma situação tridimensional no seu lado direito. O algoritmo seleciona o nó apropriado usando um teste simples, que é baseado nas listas de faces de contorno adjacentes dos nós. Quando dois nós candidatos com as mesmas coordenadas localizados na superfície da trinca são selecionados para formar um elemento, o nó que se situa no mesmo lado da face base com respeito à superfície da trinca é escolhido. As normais das faces da trinca adjacentes aos nós selecionados são usadas para realizar este teste, como ilustrado na figura 3.8. O teste assume que todas as superfícies de trincas são suaves (sem mudança de direção abrupta e sem bifurcação, o que não é realmente uma limitação em aplicações práticas de problemas de fraturamento).

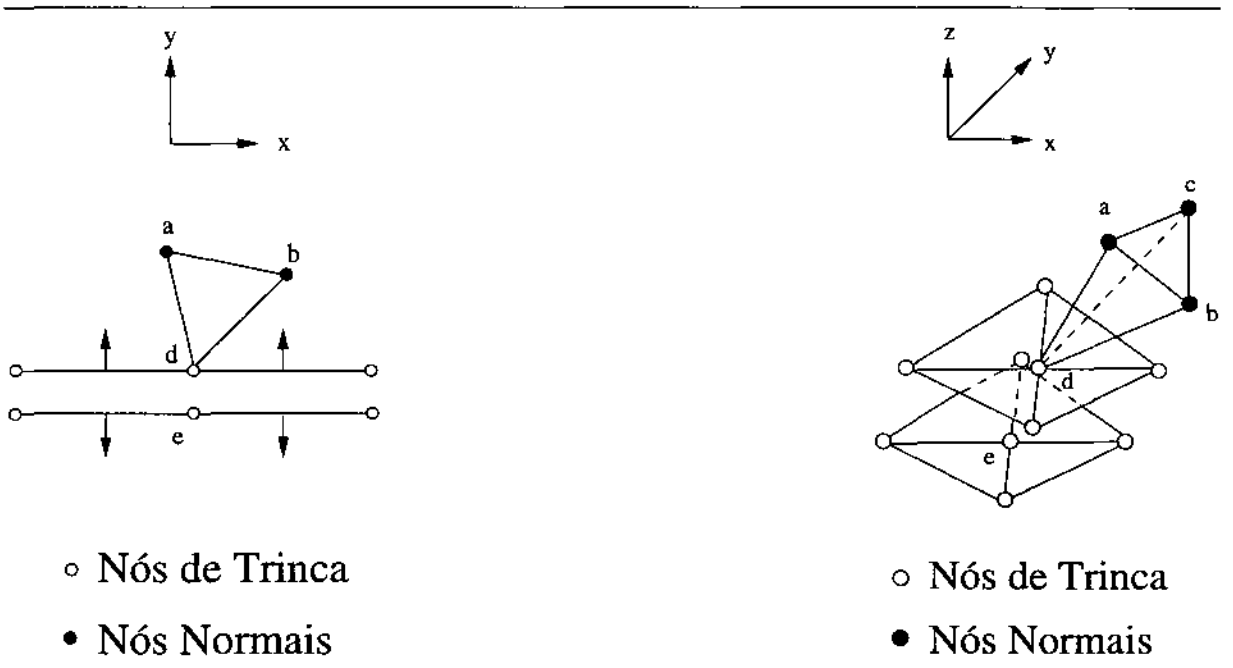



Figura 3.8: Seleção de um nó candidato em uma trinca.

- Uma vez que um tetraedro válido é formado para a face base corrente, a lista de faces ativas é atualizada. Isto é feito através dos seguintes passos. Primeiro, a face base é removida da lista de faces ativas. Então, para as outras faces do elemento: cada face é eliminada se coincide com uma face já existente na lista, ou é inserida na lista como

uma nova face. Esta inserção é feita mantendo-se a ordenação da lista de acordo com a área das faces.

- Devido aos limites geométricos impostos pela fase corrente do avanço da fronteira, existem situações em que o algoritmo falha em formar um tetraedro válido para a face base corrente do contorno. Nestes casos, a face base corrente é removida da lista de faces ativas da contração do contorno e é armazenada na lista separada de faces rejeitadas. Pode acontecer que esta face é removida posteriormente desta última lista se ela for usada na criação de um tetraedro válido para uma face base adjacente.
- Quando não existem mais faces na lista de faces ativas da contração do contorno, o algoritmo tenta gerar elementos usando as faces que foram rejeitadas previamente. Pode acontecer de faces que falharam previamente na tentativa de formar um elemento possam ser capazes de funcionar agora porque a fronteira mudou com a adição de novas faces de novos elementos gerados. Um procedimento similar é mencionado em outros trabalhos da literatura (Moller e Hansbo, 1995). A fase de geração de elementos baseada em geometria acaba quando ou não existem mais faces em ambas as listas da contração do contorno (em cujo caso uma malha ótima foi gerada) ou quando uma face rejeitada falha uma segunda vez.

#### 3.1.2.2 *Geração de elementos baseada em topologia*

O objetivo desta fase do algoritmo é forçar a geração de tetraedros válidos se possível, mesmo que os novos elementos não satisfaçam os limites usados na fase anterior para a forma dos elementos.

A fase de geração de elementos baseada em topologia começa quando uma face de contração do contorno falha duas vezes na tentativa de gerar um elemento ótimo. A lista de faces de contorno rejeitadas da fase anterior é transformada na lista de faces ativas de contorno e, analogamente à fase baseada em geometria, uma lista de faces rejeitadas é criada para faces que eventualmente falharem na geração de tetraedros válidos.

Na fase de geração de elementos baseada em topologia, qualquer nó perto da face base corrente é selecionado e armazenado em uma lista local de nós candidatos. O nó que tem a melhor métrica de ângulo sólido com respeito à face base é escolhido para a geração do novo tetraedro. Se as faces deste tetraedro não interceptarem qualquer outra face da fronteira de avanço corrente, o elemento é criado e o contorno é contraído de acordo. Como na fase baseada em geometria, a fase baseada em topologia acaba

quando a lista de faces ativas do contorno está vazia, ou quando uma face da fronteira de avanço corrente é rejeitada duas vezes devido a problemas de interseção.

### 3.1.2.3 *Geração de elementos baseada em procedimento de "volta-passo"*

Em modelos tridimensionais complexos, existem situações em que uma malha válida não pode ser gerada, mesmo com os procedimentos realizados na fase de geração de elementos baseada em topologia. Considere o poliedro mostrado no lado esquerdo da figura 3.9. A única maneira possível de se gerar uma malha de tetraedros para este poliedro é inserindo um novo nó no seu centro e conectando este nó às faces triangulares do contorno. Existem casos, entretanto, em que não é possível inserir um nó para formar tetraedros válidos, como o poliedro no lado direito da figura 3.9. Esse poliedro não possui um "núcleo" no qual qualquer ponto é visível através de uma linha reta traçada desde todos os seus vértices (Perucchio e Saxena, 1990). No presente algoritmo, a solução para este problema é localmente modificar a fronteira de avanço, eliminando tetraedros adjacentes já formados até que um poliedro "quase" convexo não "malhado" é formado. É interessante observar que este problema não possui equivalência em duas dimensões. Pode ser demonstrado (O'Rourke, 1987) que qualquer polígono que não se auto-intercepta pode ser triangulado sem a necessidade de inserção de vértices adicionais.

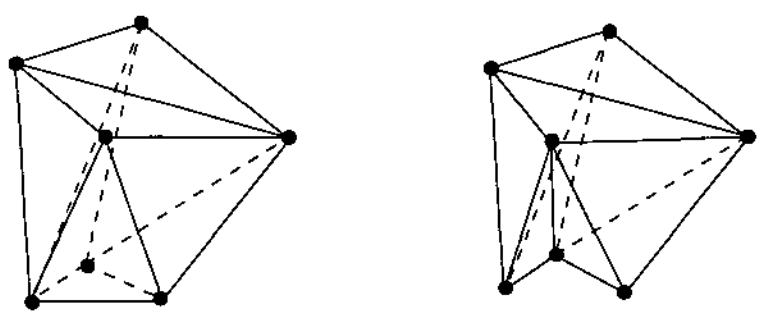


Figura 3.9: Poliedros não "malháveis".

O procedimento usado para transformar um poliedro de forma ruim em um com um núcleo de visibilidade é descrito como se segue:

- Primeiro, um poliedro de forma ruim associado à face base corrente é identificado. É comum, neste estágio do procedimento de contração de contorno, que a fronteira de avanço possa estar dividida em vários poliedros separados. Uma lista de nós

auxiliares é criada para conter os nós desse poliedro. Primeiro, os três nós da face base que não pode formar um tetraedro válido são inseridos nessa lista auxiliar. Então, recursivamente, para cada face do contorno adjacente ao nó corrente os outros nós da face são inseridos na lista sem duplicação. Com base nesta lista de nós, as faces que pertecem ao poliedro de forma ruim são facilmente identificadas: elas serão as faces do contorno ativo que possuem todos os seus vértices nessa lista.

- Após o contorno do poliedro de forma ruim ser determinado, um teste de visibilidade é realizado. Este teste consiste em se calcular as coordenadas do centróide do poliedro e calcular o número de interseções que ocorreriam se linhas fossem traçadas do centróide a cada vértice do poliedro.
- Se pelo menos uma interseção existe, então o poliedro deve ser modificado. O processo de se converter o poliedro em um "quase" convexo consiste em remover os elementos já formados que possuem a face com o maior número de interseções. A fronteira é atualizada para refletir a remoção deste elemento. Após isto, o teste de visibilidade é realizado novamente. O processo é repetido até que o centróide seja visível por todos os nós do poliedro.

A figura 3.10 mostra o processo de se transformar um poliedro em um convexo após uma face "interceptada" ser removida. Neste exemplo, a face formada pelos nós *a*, *c* e *d* precisa ser eliminada. Isto resulta na remoção desta face e da face *abd*, e na inclusão das faces *abc* e *bcd* na lista de contorno do poliedro. Esta figura também mostra os elementos tetraédricos gerados após a inserção de um nó no centróide do poliedro transformado.

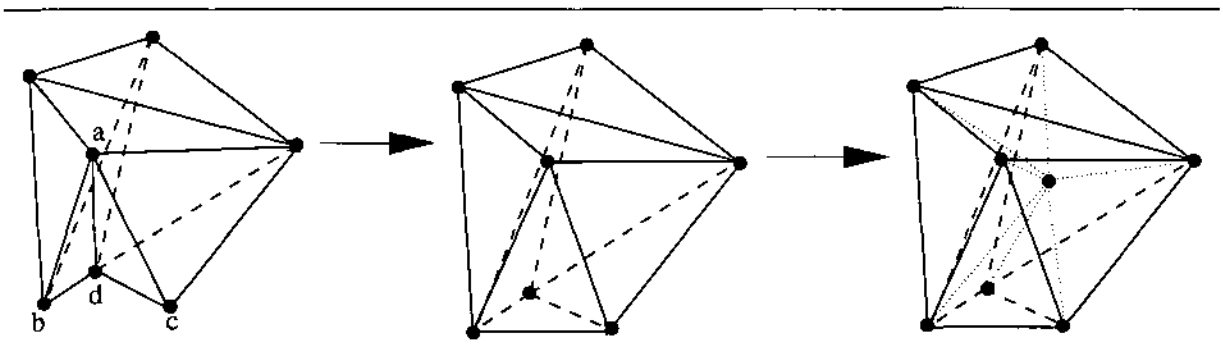



Figura 3.10: Transformação de um poliedro em um outro convexo.

Pode ser argumentado que o processo de se achar um poliedro "quase" convexo pode falhar se faces forem removidas até que a malha de contorno original, que não pode

ser modificada, seja atingida. Quando isto ocorre o algoritmo não é capaz de gerar uma malha para o modelo. Entretanto, foi observado através de vários exemplos que são mostrados na sequência deste capítulo que, para malhas de contorno razoavelmente refinadas fornecidas, poliedros de forma ruim quase sempre ocorrem no meio do modelo e é muito improvável que o contorno seja atingido antes que um poliedro onde uma malha possa ser gerada, seja formado.

### 3.1.3 *Melhoria local da malha*

Nas últimas duas fases do procedimento de avanço da fronteira descrito anteriormente, elementos tetraédricos de forma não-ótima podem ser gerados. Esta seção descreve dois procedimentos locais *a posteriori* que ocorrem para melhorar a qualidade da malha. O primeiro é uma técnica de suavização convencional que consite na relocação de nós baseada na média das coordenadas nodais, com testes de validação. A segunda é um procedimento de "volta-passo", similar à última fase do procedimento de avanço da fronteira, que remove faces de elementos de forma ruim para criar uma região onde elementos com melhor forma possam ser gerados.

Os procedimentos de melhoria local implicam que medidas de qualidade da forma dos elementos são necessárias. Neste trabalho, uma métrica de qualidade de forma foi adotada baseada em um estudo aprofundado de qualidade de forma de elementos finitos tetraédricos realizado (ver apêndice). Esta métrica é a razão normalizada entre a raiz quadrada média (*root mean square*) dos comprimentos das arestas do tetraedro ($S_{rms} = \sqrt{\frac{1}{6}\sum_{i=0}^{5} S_i^2}$, onde $S_i$ é o comprimento de uma aresta) e o volume $V$ do tetraedro:

$$\gamma = \frac{S_{rms}^3}{V}. \tag{3.1}$$

Esta métrica apresenta uma boa capacidade de caracterizar a qualidade de um tetraedro e é computacionalmente eficiente. O intervalo de valores válidos varia de um até infinito ($[1, \infty)$), e o valor ótimo para o tetraedro regular é aproximadamente 8.5.

#### 3.1.3.1 *Suavização Laplaciana com testes de validação*

Uma técnica de suavização é usada para melhorar a qualidade da malha através do reposicionamento de nós dentro de um *patch* (grupo de elementos adjacentes). Uma

formulação geral desta técnica é dada através da equação (3.2), que é a forma genérica de uma função Laplaciana ponderada (Foley *et al.*, 1990):

$$\mathbf{X}_O^{n+1} = \mathbf{X}_O^n + \phi \frac{\sum_{i=1}^{m} \omega_{iO}(\mathbf{X}_i^n - \mathbf{X}_O^n)}{\sum_{i=1}^{m} \omega_{iO}}. \tag{3.2}$$

Na equação 3.2, $m$ é o numéro de nós conectados ao nó $O$, $X_O^{n+1}$ é a posição do nó $O$ na iteração de suavização $n+1$, $\omega_{iO}$ é a função ponderada entre os nós $i$ e $O$, e $\phi$ é o parâmetro de relaxação que é normalmente escolhido no intervalo $(0, 1]$. Em teoria, esta técnica de suavização genérica realiza a relaxação de nós interiores de uma malha e resulta em uma malha de melhor qualidade. Em três dimensões, entretanto, esta técnica ocasionalmente resultará em elementos inválidos com volumes negativos. Isto pode acontecer também em duas dimensões, gerando elementos inválidos com área negativa, mas é mais raro. Reduzir o valor do coeficiente de relaxação $\phi$ é uma maneira sugerida para diminuir as chances de ocorrer este problema (Lewis *et al.*, 1996c). A técnica de suavização genérica tem provado ser útil para melhorar a qualidade da malha, mas deve se tomar cuidado para selecionar o valor do parâmetro que controla o procedimento de suavização. Existem casos, entretanto, em que esta técnica sozinha simplesmente não é capaz de melhorar a qualidade da malha inteira. Neste trabalho, um valor de $\phi = 0.5$ foi adotado, e a técnica de suavização é repetida 5 vezes. A cada passo de suavização, após calcular a nova posição de cada nó interno, um teste de consistência é realizado. Este teste verifica se algum elemento adjacente terá um volume negativo considerando a nova posição do nó, caso em que a relocação do nó não é realizada para este passo corrente.

### 3.1.3.2 *Avaliação da qualidade e procedimento de "volta-passo" local*

A suavização Laplaciana não é suficiente para garantir a qualidade da malha. Neste trabalho, um procedimento de "volta-passo" (*back-tracking*) é adotado para melhorar a qualidade da malha volumétrica gerada. Esta técnica pode também ser usada para qualquer malha dada que não seja relacionada com o algoritmo descrito. Isto pode ser muito útil para melhorar a qualidade de malhas obtidas por outros métodos.

O procedimento de "volta-passo" consiste em remover um elemento que é classificado como de forma ruim e um grupo de elementos na sua vizinhança. A classificação de tetraedros ruins é baseada em uma métrica especificada, que neste trabalho é a métrica $\gamma$ descrita anteriormente, embora qualquer outra métrica representativa

da qualidade dos elementos possa ser usada. Para cada elemento da malha gerada, a métrica de qualidade $\gamma$ é avaliada. Se o valor desta métrica está fora de um intervalo pré-definido, o elemento é classificado como um elemento de forma ruim. Os limites inferior e superior deste intervalo são definidos empiricamente. Neste trabalho, o limite inferior é 5.0 e o limite superior é o valor da métrica "ótima" 8.5 multiplicado por um fator de 30.

O objetivo do procedimento de "volta-passo" é eliminar faces de elementos envolvendo um elemento ruim para criar um poliedro local que pode ser "remalhado" com elementos de melhor forma. Um poliedro inicial local a ser "malhado" é criado por um algoritmo que é definido através dos passos detalhados a seguir. Isto é ilustrado por meio do exemplo bidimensional mostrado na figura 3.11:

- Passo 1: criação de uma lista inicial de nós adjacentes. Para cada elemento que é adjacente a um elemento de forma ruim (o elemento em preto da figura 3.11), seus nós são armazenados em uma lista. Um elemento adjacente é um elemento que compartilha três nós (uma face) com o elemento ruim.
- Passo 2: criação de uma lista ampliada de nós adjacentes. Para cada elemento que tem três nós na lista inicial de nós adjacentes, é adicionado seu quarto nó na lista se este nó pertence a pelo menos dois destes elementos.
- Passo 3: criação de uma lista de elementos a serem removidos. Estes elementos são definidos como tendo todos os seu quatro nós na lista de nós adjacentes.
- Passo 4: criação do poliedro local a ser "remalhado". Para cada elemento a ser removido, cada uma das suas faces é inserida no contorno do poliedro se ela já não está lá ainda, caso contrário a face é eliminada.

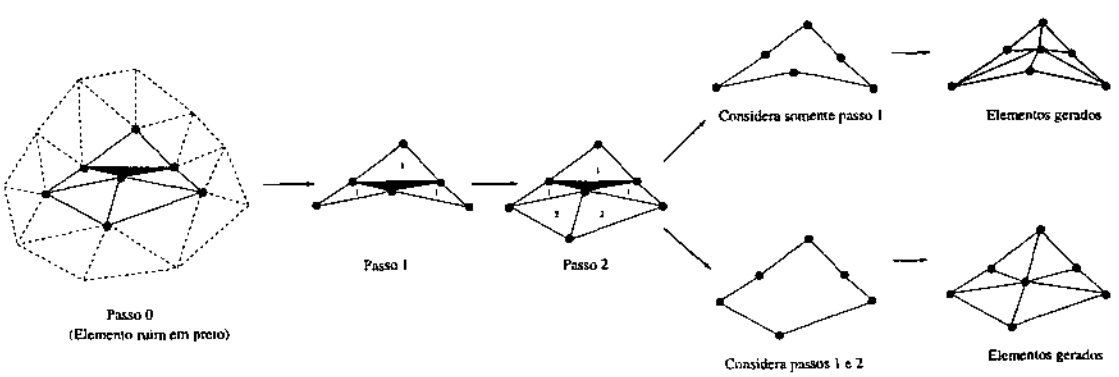



Figura 3.11: Procedimento de "volta-passo" bidimensional.

A figura 3.11 mostra, para o caso bidimensional, procedimentos alternativos com ou sem o passo 2 do algoritmo. Deduz-se que o poliedro ampliado resultante do passo 2 tem mais chance de ser convexo, embora isto não seja garantido. No caso do poliedro resultante não poder ser "malhado", os procedimentos descritos em seções anteriores para transformar este poliedro em um "quase" convexo são também usados aqui.

A figura 3.12 mostra, no seu lado esquerdo, um poliedro tridimensional obtido pelo uso do procedimento de "volta-passo" descrito. O tetraedro formado pelos nós *a*, *b*, *c* e *d* foi o elemento de forma ruim considerado para ser removido (pode ser notado que este elemento tem o volume quase zero). A figura também mostra o novo nó interno *i* criado e os novos elementos gerados.

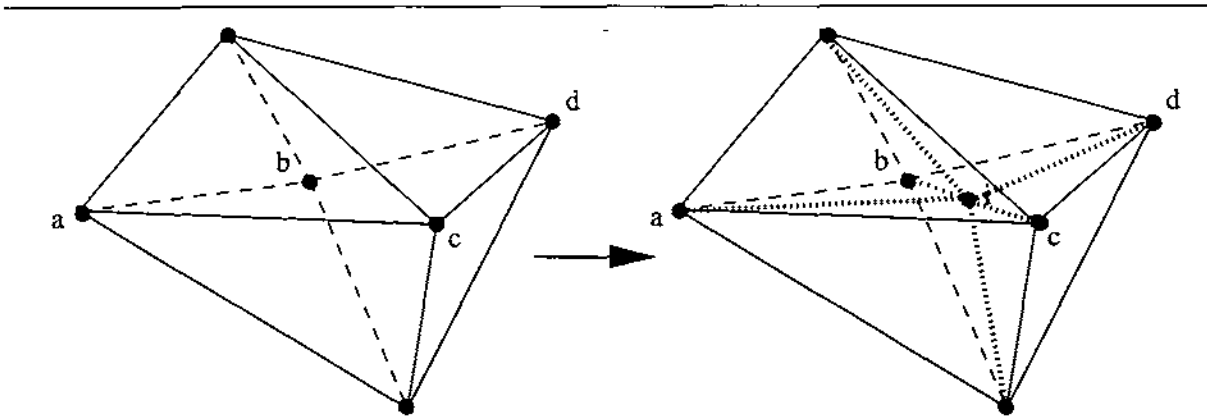



Figura 3.12: Procedimento de "volta-passo" tridimensional.

O procedimento de "volta-passo" seguido pela re-geração dos elementos, analogamente à técnica de suavização, é aplicado uma série de vezes (5 vezes). Neste trabalho, a técnica de suavização foi usada em conjunto com o procedimento de "volta-passo". Foi observado que resultados melhores são obtidos se "volta-passo"/re-geração é aplicado após cada passo de suavização.

## 3.2 *Qualidade das malhas geradas pelo algoritmo*

Para se avaliar a qualidade das malhas volumétricas geradas pelo algoritmo descrito, uma análise pode ser feita baseada na métrica $\gamma$ adotada. Como explicado anteriormente, esta métrica tem um intervalo de valores válidos que variam de 1.0 até infinito ($[1, \infty)$), e o valor ótimo para o tetraedro regular é aproximadamente 8.5. É desejado se

ter uma malha com $5.0 \leq \gamma \leq (8.5 * fator)$ para a maioria dos elementos, o que representa um intervalo de elementos aceitáveis. Este fator foi empiricamente definido como 30, e corresponde ao fator usado para definir o limite superior quando da fase de melhoria local da malha. Conseqüentemente, é desejado ter-se uma malha com $5.0 \leq \gamma \leq 255$ para a maioria dos elementos gerados. Para se avaliar a qualidade da malha, entretanto, desejou-se ser mais restritivo e um fator de 16 foi usado ao invés de 30, o que representa que somente elementos com $5.0 \leq \gamma \leq 136$ são considerados "bons" elementos.

Na seqüência são mostrados histogramas de qualidade da malha para três exemplos. Para cada exemplo, é mostrado um histograma correspondendo a qualidade dos elementos gerados antes da aplicação dos procedimentos de melhoria da qualidade da malha no lado esquerdo da figura, e após a aplicação desses procedimentos no lado direito. As malhas para esses exemplos, que demonstram a confiabilidade do algoritmo descrito para os casos até agora avaliados, também são mostradas, bem como uma tabela com alguns paramêtros estatísticos, como máximo, médio e mínimo valores para $\gamma$ e o desvio padrão $\sigma$. Embora o limite inferior estabelecido para o algoritmo é 5.0, normalmente elementos com métrica iguais ou maiores que o valor ótimo 8.5 são gerados, e por isto os histogramas começam deste valor. Pode ser visto que a maioria dos elementos gerados são localizados no intervalo $[8.5, 34]$, o que representa elementos de muito boa forma, mesmo antes da aplicação dos procedimentos de melhoria da qualidade da malha. Entretanto, os histogramas relacionados às malhas geradas antes da aplicação dos procedimentos de melhoria de qualidade mostram que existe uma percentagem razoável de elementos com valores da métrica maiores que 136, o que representa um número indesejável de elementos de forma ruim. Após a aplicação dos procedimentos de melhoria de qualidade da malha, estas percentagens são reduzidas a um nível aceitável.

É importante mencionar que os exemplos possuem pequenas trincas no interior, conforme será mostrado, e conseqüentemente os aspectos de transição previamente discutidos são importantes e podem ser observados. Outra importante observação, que não pode ser deduzida dos histogramas, é que este algoritmo normalmente gera os elementos de melhor forma no contorno ou perto dele. Isto acontece porque, como o algoritmo começa do contorno e nesta fase a maioria dos elementos são gerados na fase de geração de elementos baseada em geometria, estes elementos respeitam a faixa de valores ótimos e conseqüentemente são elementos de boa forma. Isto é uma importante propriedade, considerando-se que geralmente os elementos perto do contorno, de trincas ou de regiões de altos gradientes de resposta, são os elementos que devem ter as melhores formas.

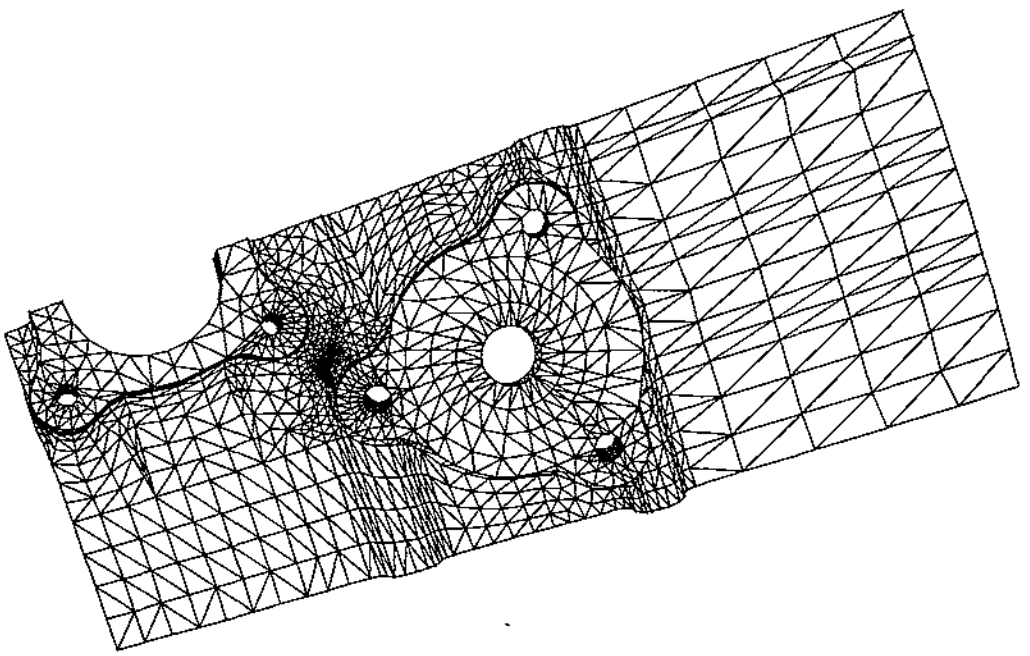


Figura 3.13: Malha de contorno para o exemplo de um difusor.

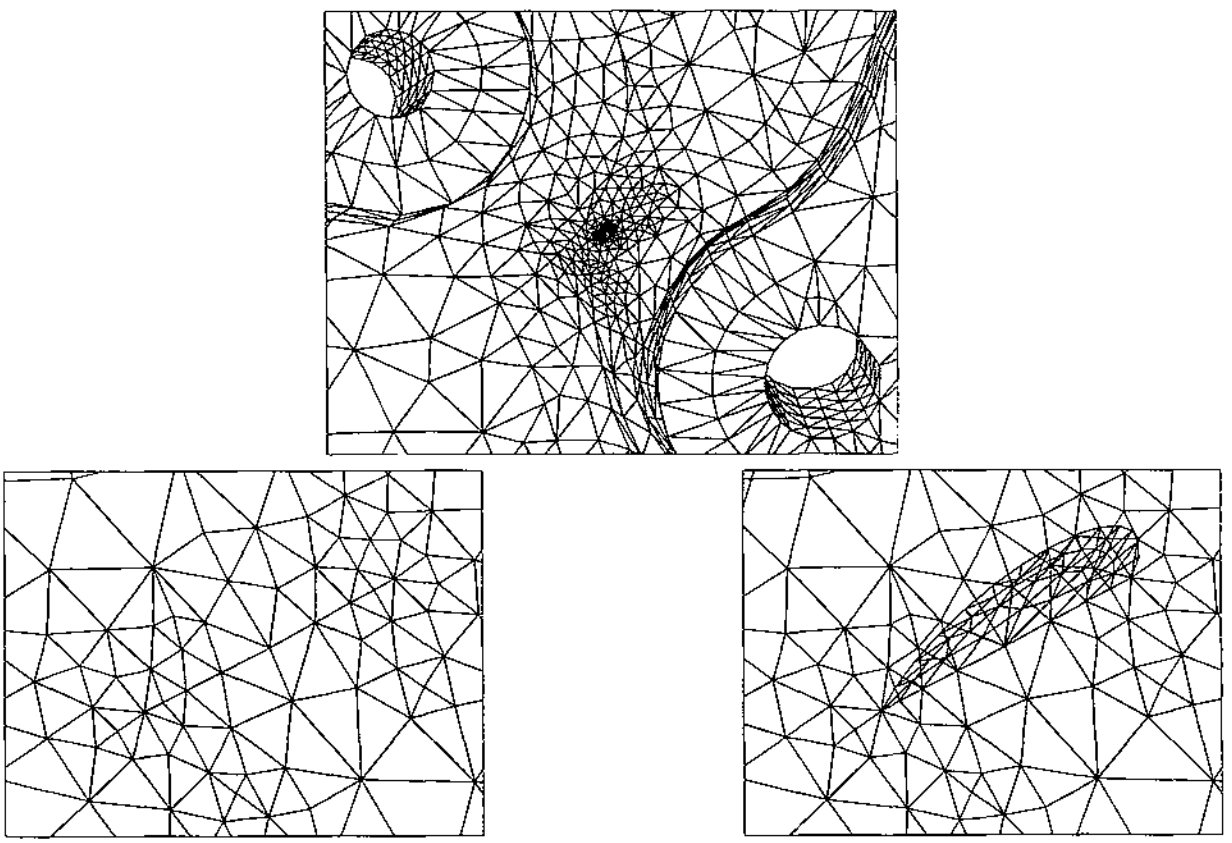


Figura 3.14: Detalhe da malha de contorno mostrando uma trinca para o exemplo de um difusor: a) acima: detalhe da região da trinca; b) esquerda: malha da região da trinca; c) direita: malha da região da trinca com a malha da trinca.

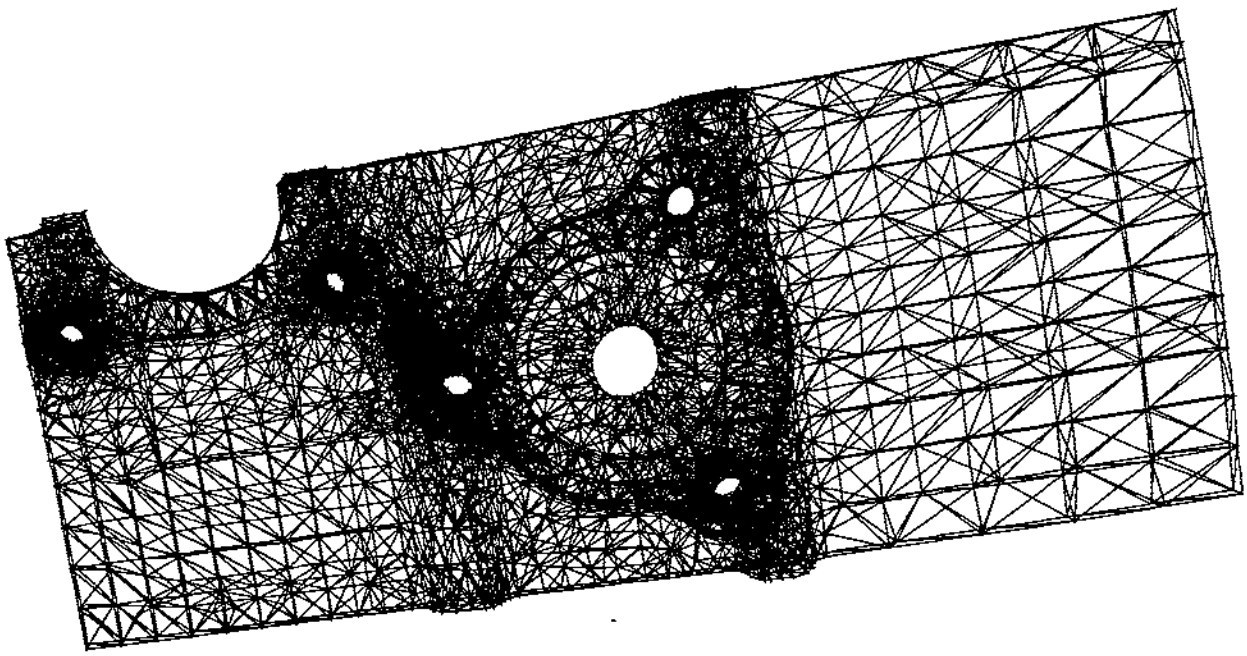


Figura 3.15: Malha de volume para o exemplo de um difusor.

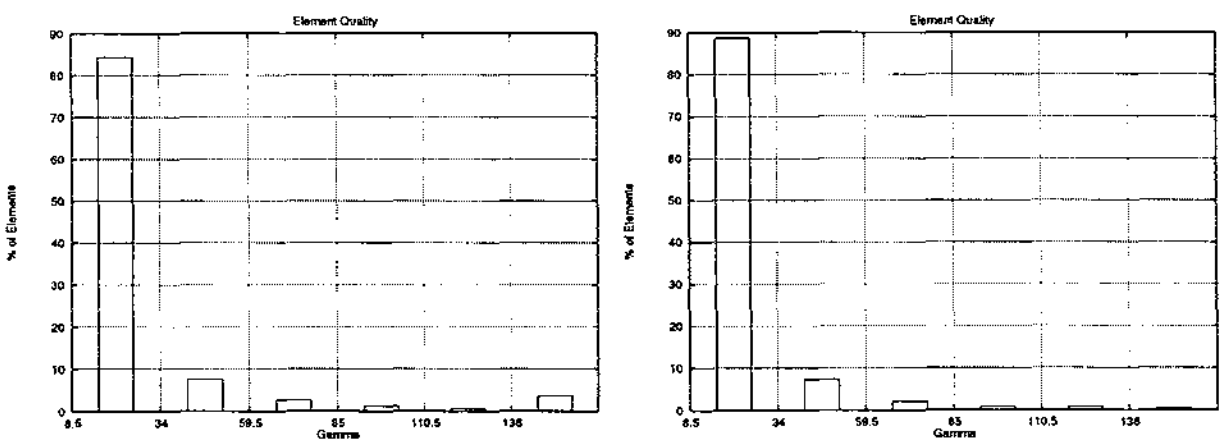



Figura 3.16: Histogramas para o exemplo de um difusor: a) esquerda: histograma de qualidade dos elementos antes do procedimento de melhoria da malha; b) direita: histograma de qualidade dos elementos depois do procedimento de melhoria da malha.

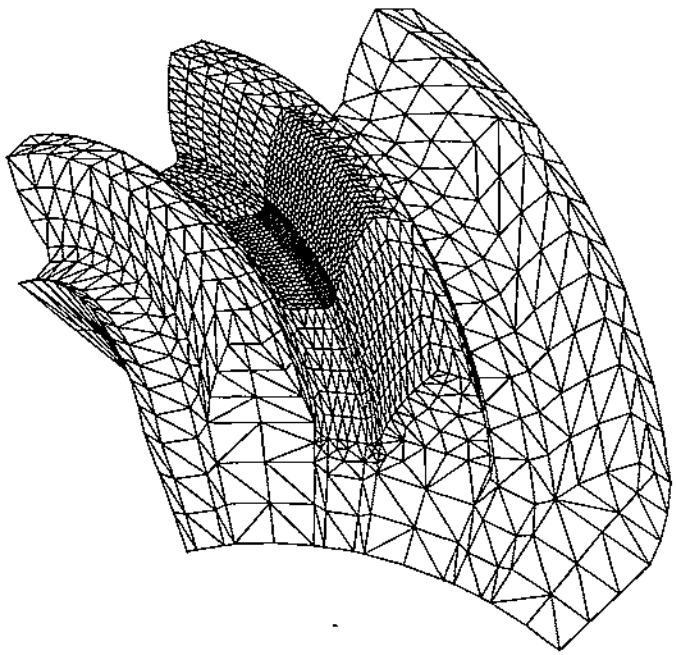


Figura 3.17: Malha de contorno para o exemplo de uma engrenagem.

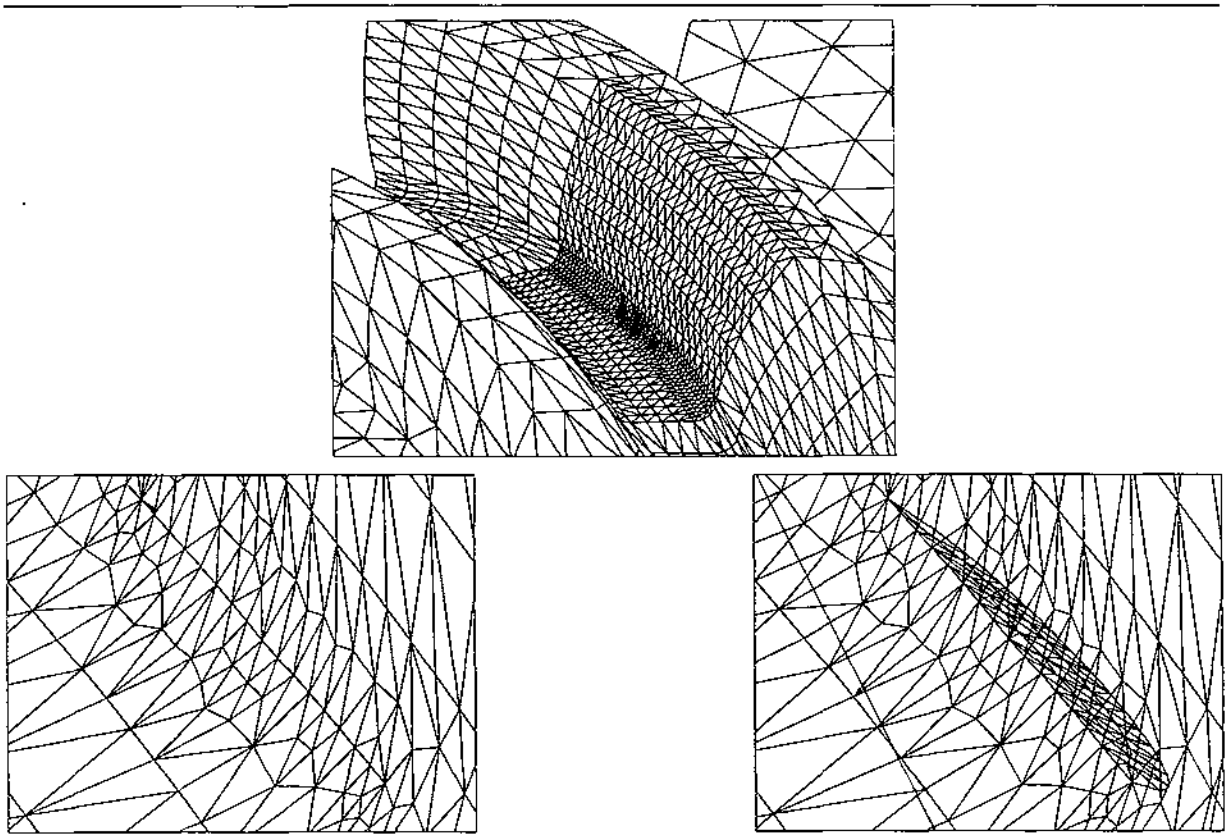


Figura 3.18: Detalhe da malha de contorno mostrando uma trinca para o exemplo de uma engrenagem: a) acima: detalhe da região da trinca; b) esquerda: malha da região da trinca; c) direita: malha da região da trinca com a malha da trinca.

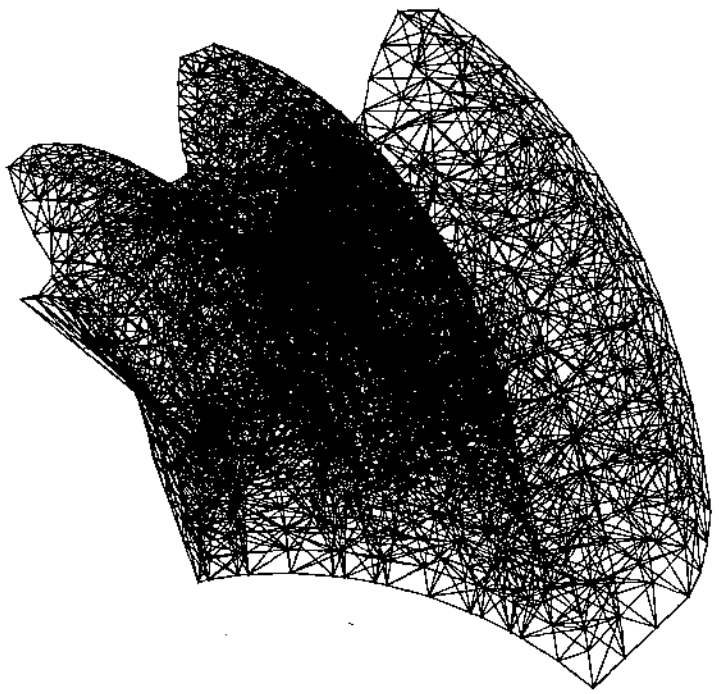


Figura 3.19: Malha de volume para o exemplo de uma engrenagem.

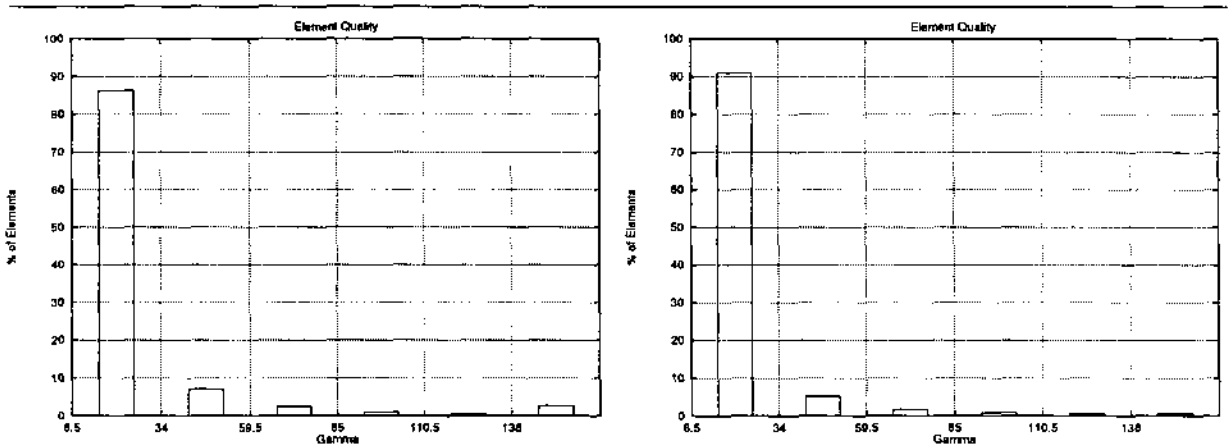



Figura 3.20: Histogramas para o exemplo de uma engrenagem: a) esquerda: histograma de qualidade dos elementos antes do procedimento de melhoria da malha; b) direita: histograma de qualidade dos elementos depois do procedimento de melhoria da malha.

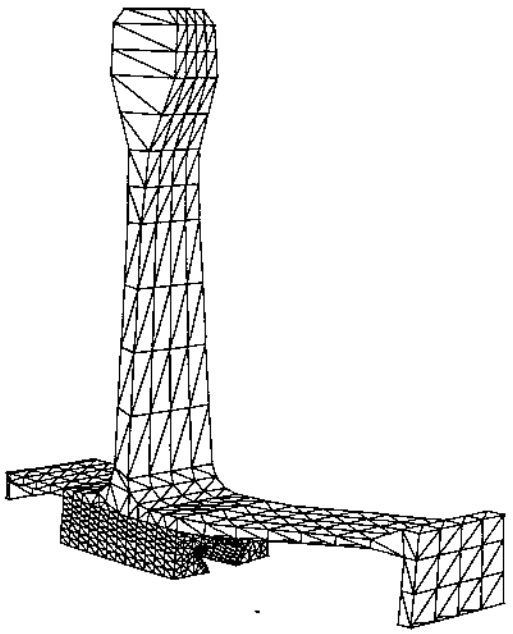


Figura 3.21: Malha de contorno para o exemplo de uma pá de turbina.

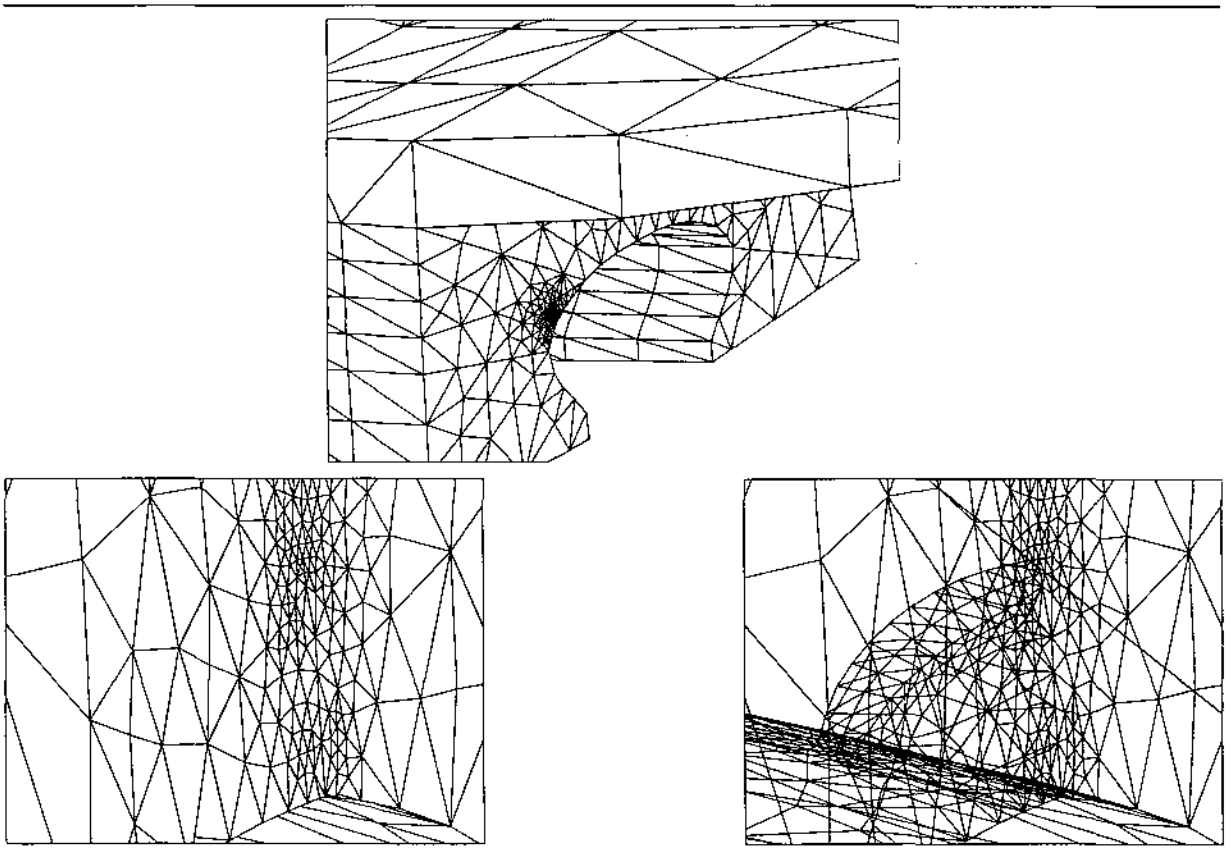


Figura 3.22: Detalhe da malha de contorno mostrando uma trinca para o exemplo de uma pá de turbina: a) acima: detalhe da região da trinca; b) esquerda: malha da região da trinca; c) direita: malha da região da trinca com a malha da trinca.

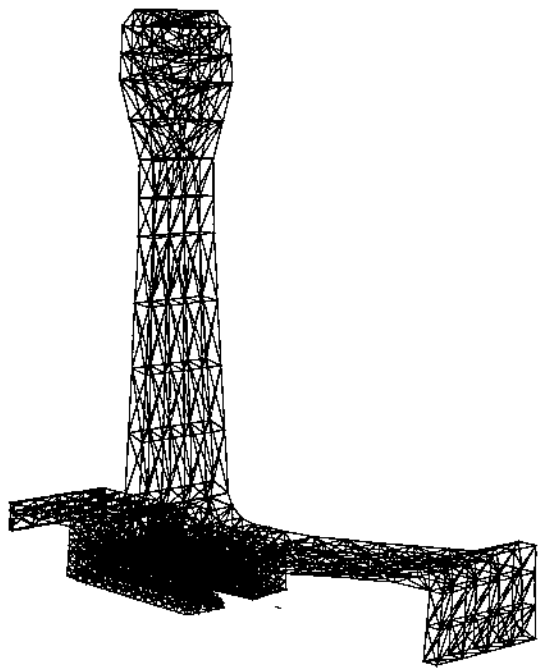


Figura 3.23: Malha de volume para o exemplo de uma pá de turbina.

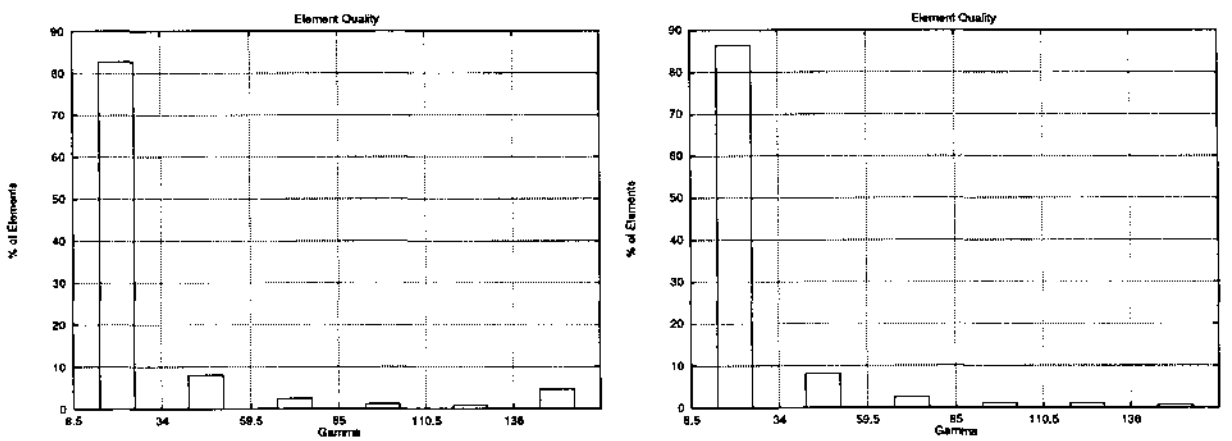



Figura 3.24: Histogramas para o exemplo de uma pá de turbina: a) esquerda: histograma de qualidade dos elementos antes do procedimento de melhoria da malha; b) direita: histograma de qualidade dos elementos depois do procedimento de melhoria da malha.

| Exemplo | Histograma | # Elementos | $\gamma_{avg}$ | $\gamma_{min}$ | $\gamma_{max}$ | $\sigma$ |
|---|---|---|---|---|---|---|
| difusor | Antes | 16463 | 26.01 | 6.48 | 505.45 | 38.34 |
| | Depois | 17073 | 21.26 | 8.52 | 253.63 | 20.79 |
| engrenagem | Antes | 17386 | 25.04 | 8.43 | 505.09 | 37.72 |
| | Depois | 16990 | 20.37 | 8.51 | 293.82 | 21.01 |
| pá de turbina | Antes | 9628 | 28.35 | 6.72 | 508.69 | 44.07 |
| | Depois | 10046 | 23.24 | 8.53 | 318.15 | 25.01 |

Figura 3.25: Valores estatísticos relacionados à qualidade dos exemplos.

## 3.3 *Desempenho na geração das malhas pelo algoritmo*

A complexidade computacional da técnica de avanço da fronteira tem sido estudada na literatura. Enquanto o comportamento $O(NlogN)$, onde $N$ é o número de elementos gerados na malha final, tem sido mencionado de ter sido alcançado em determinados instantes do processo (Lohner 1988; Bonet and Peraire, 1991), existem alguns trabalhos que atestam que um desempenho $O(N^plogN)$, com $p$ ligeiramente acima de um é uma expectativa mais realista (Moller e Hansbo, 1995; Jin e Tanner, 1993). Todos estes estudos consideram o uso de estruturas de dados especiais, como *octrees*, para melhorar as operações de busca necessárias. Além disso, o desempenho é fortemente dependente da maneira como essas estruturas de dados se desenvolvem. Conseqüentemente, esta performance pode variar dependendo do problema (Moller e Hanbo, 1995).

No algoritmo descrito neste trabalho, a fase de geração de elementos baseada em geometria corresponde à técnica de avanço de fronteira padrão, com algumas pequenas modificações. Entretanto, algumas fases adicionais, como a fase de geração de elementos baseada em topologia, fase de "volta-passo" (*back-tracking*) e fase de melhoria de qualidade, foram incluídas no algoritmo, para obter-se uma maior confiabilidade e para garantir-se a qualidade das malhas geradas para um número grande de problemas. Baseado nisto, o desempenho do algoritmo se torna difícil de ser avaliado. Entretanto, experimentos numéricos podem ser usados para examinar a performance do algoritmo em aplicações práticas. Isto é uma maneira geral de investigar a complexidade computacional de um algoritmo quando este algoritmo se torna sofisticado e isto tem sido usado na literatura para alguns algoritmos baseados em diferentes enfoques para gerar malhas volumétricas (Moller e Hansbo, 1995; Lewis *et al.*, 1996c).

Para investigar a performance do algoritmo descrito neste trabalho, quatro discretizações de diferentes ordens de magnitude para dois exemplos foram consideradas. O primeiro exemplo é um toro e as malhas são mostradas a seguir. O mesmo problema foi analizado por Moller e Hansbo (1995), e um desempenho $O(N^plogN)$ foi obtido. No presente trabalho, o mesmo desempenho foi experimentalmente observado e os tempos de geração da malha mostram uma parte linear do começo da curva logarítimica. O segundo exemplo é um poço, definido por um cubo unitário com um cilindro inclinado em seu interior, onde suas malhas também são mostradas. Um desempenho similar também foi observado para este exemplo. Tempos totais de computação foram avaliados para a geração das malhas volumétricas para estes problemas, usando uma SUN SPARC Station 20. Os tempos indicam uma boa performance para aplicações práticas.

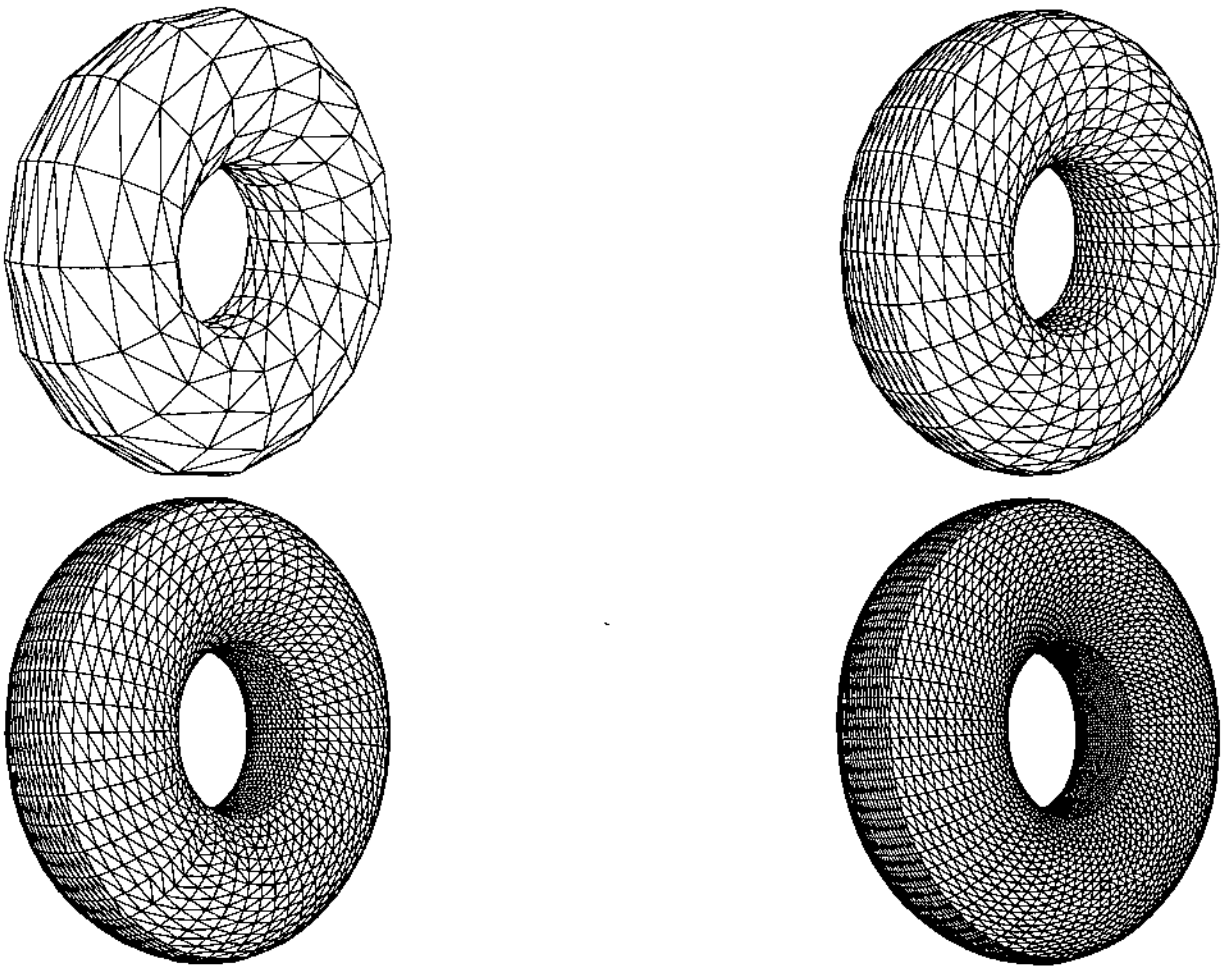


Figura 3.26: Quatro discretizações do toro.

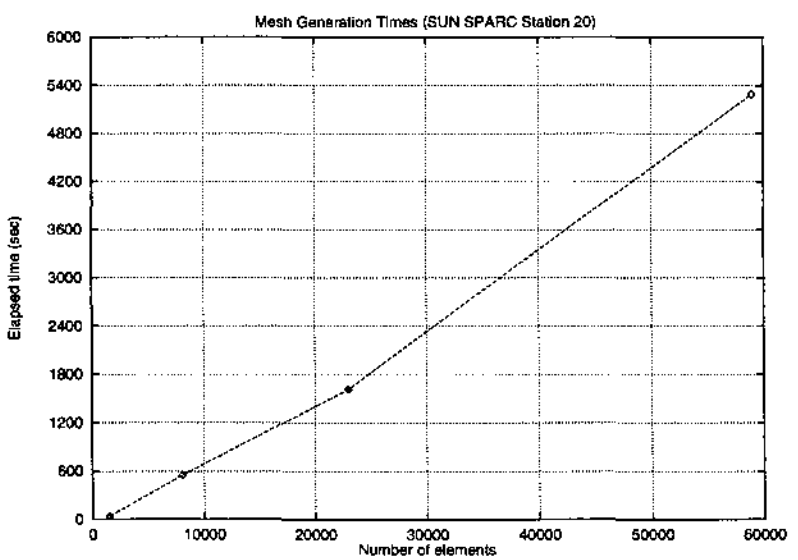



Figura 3.27: Tempos de geração para o toro.

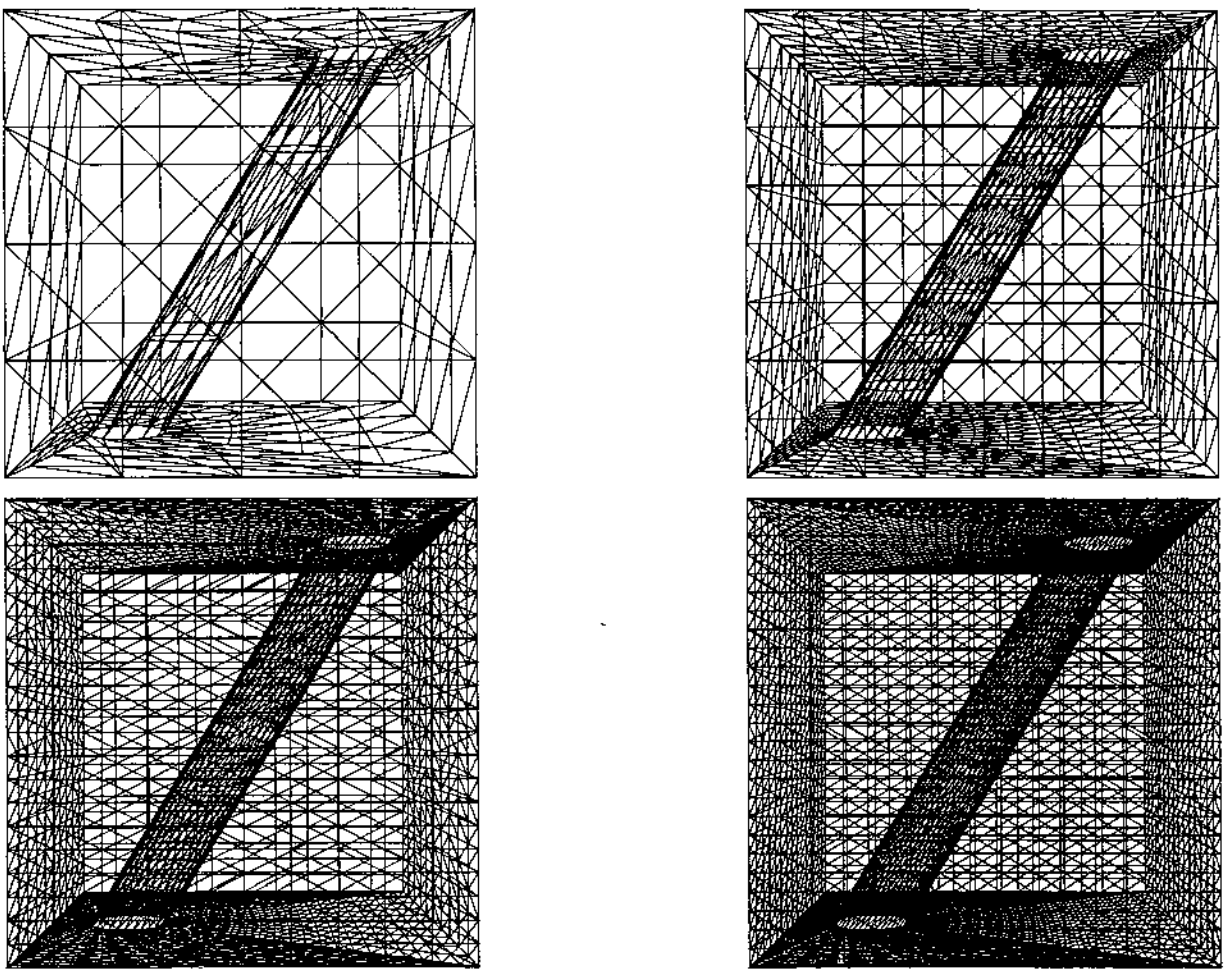


Figura 3.28: Quatro discretizações do poço.

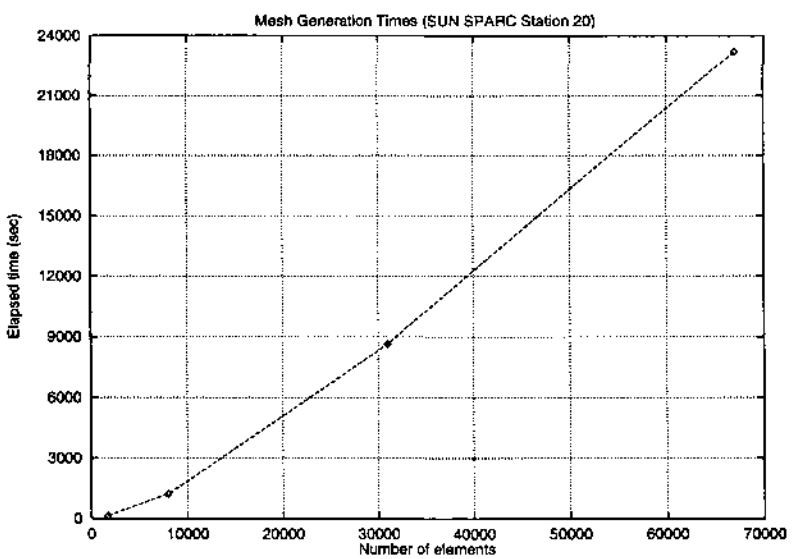



Figura 3.29: Tempos de geração do poço.

# 4

# Exemplos

Este capítulo apresenta exemplos resultantes da aplicação da técnica de geração de malha volumétrica proposta neste trabalho para análise de modelos reais tridimensionais pelo método dos elementos finitos. Além disso, para alguns dos exemplos, essa técnica é combinada com as técnicas de estimativa de erro de discretização em três dimensões também descritas neste trabalho, e refinamentos adaptativos são realizados. A principal intenção destes exemplos é ilustrar o uso e a efetividade dessas técnicas como ferramentas práticas para análise de modelos tridimensionais pelo métodos dos elementos finitos, incluindo trincas. Essas técnicas são integradas em um sistema gráfico computacional, permitindo modelagem, análise e visualização, e criando um sistema base para validar estas técnicas e para servir como protótipo para um futuro sistema para simulações genéricas auto-adaptativas tridimensionais usando o método dos elementos finitos.

Todas as malhas de contorno e volumétricas mostradas foram geradas no programa *FRANC3D*, onde o algoritmo de geração de malha volumétrica proposto neste trabalho foi incluído. Isto permitiu a modelagem de exemplos considerando trincas, como será mostrado. Além disso, os exemplos que envolveram refinamentos adaptativos foram refinados pelo autor no *FRANC3D*, isto é, sucessivas malhas de superfície foram geradas semi-automaticamente baseando-se nas análises de erro de malhas de passos anteriores. No futuro deseja-se incluir nesse sistema tridimensional procedimentos para realizar esse refinamento adaptativo tridimensional de forma automática. Para isso, a eficiência das técnicas descritas nesse trabalho é fundamental.

Três exemplos são apresentados. O primeiro exemplo é uma viga curta engastada em balanço. Refinamentos adaptativos são realizados para esse problema, baseados na técnica de recuperação *SPR*. O segundo exemplo é uma placa com furo circular carregada com pressão axial. Refinamentos adaptativos também são realizados para este exemplo e comparações dos valores de tensão recuperados para as diferentes técnicas de recuperação descritas são mostradas. O terceiro exemplo é um fecho de uma porta de um avião. Esta é uma peça real que apresenta problemas de trincagem por fadiga. O modelo sem trinca é analisado e a região onde uma provável trinca se desenvolveria

é mostrada. Uma trinca arbitrária é inserida e uma nova análise é realizada para o modelo modificado.

Todos os exemplos foram analisados usando-se elemento finito tetraédrico isoparamétrico quadrático de 10 nós (TET10). Para os exemplos onde refinamentos adaptativos foram realizados o critério de parada é dado por

$$\overline{\eta} \leq \eta^*, \tag{4.1}$$

onde $\eta^*$ é o erro relativo máximo permitido, e $\overline{\eta}$ é o erro relativo calculado e é dado por

$$\overline{\eta} = \frac{\|\overline{e}\|}{\sqrt{\|\hat{u}\|^2 + \|\overline{e}\|^2}}, \tag{4.2}$$

onde $\|\overline{e}\|$ é norma de energia para o erro nos deslocamentos e $\|u\|$ é a raiz quadrada do dobro da energia de deformação, conforme descrito no capítulo 2. Nos exemplos $\eta^*$ foi usado com 5%, o que é usual na literatura para problemas considerando-se elementos quadráticos.

## 4.1 *Exemplo 1 - Viga curta em balanço*

O primeiro exemplo é uma viga curta em balanço, fixa em uma extremidade (face *ABCO*) e submetida à uma pressão distribuída vertical na sua extremidade livre (face *DHGF*). A figura 4.1 mostra suas dimensões, condições de contorno e atributos. Devido à simetria, metade da viga foi analizada fazendo-se com que na face *OHGA* os deslocamentos na direção $x$, $u$, sejam nulos.

As malhas para três passos de refinamento são mostradas na figura 4.3. Essas figuras mostram também as deformadas para cada malha. A figura 4.2 mostra uma tabela com os principais parâmetros obtidos durante os refinamentos. Observa-se que após um primeiro refinamento o erro já caiu para abaixo do parâmetro desejado, mas um passo seguinte foi realizado só para efeito de verificação e o erro final foi de 2.3%, ou seja, bastante satisfatório. Esse problema também foi analizado em um trabalho bastante recente por Lee e Lo (1997), também foram necessários três passos para atingir o erro desejado.

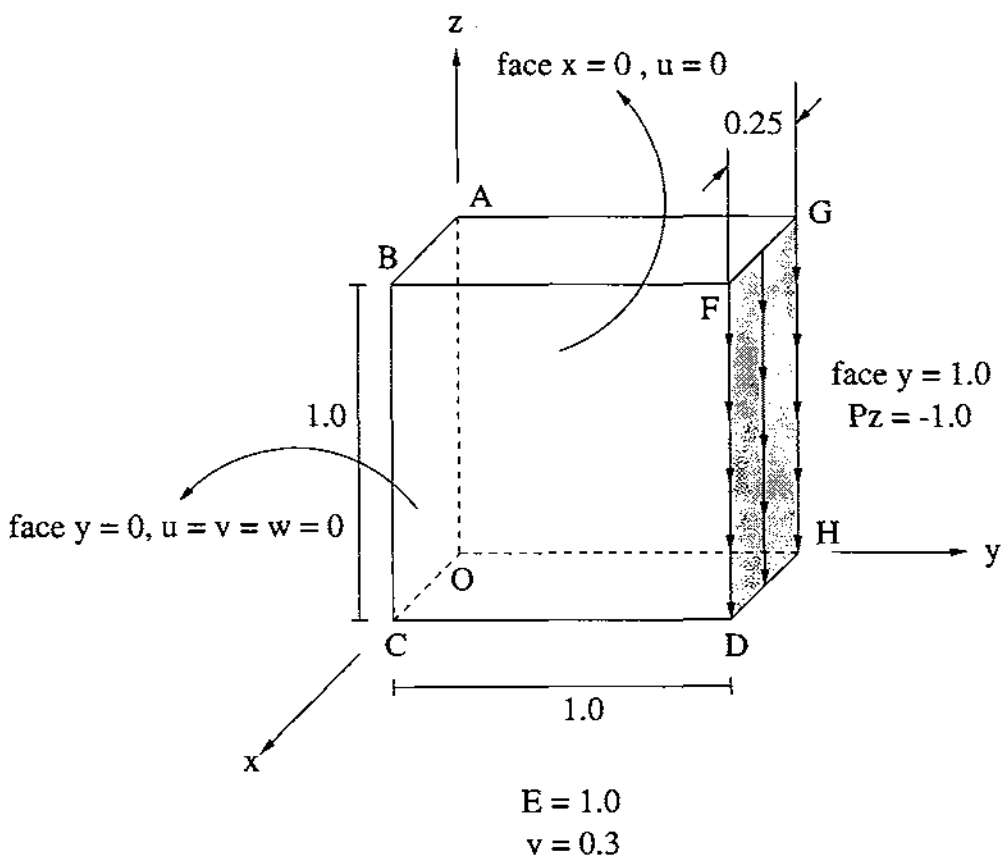



Figura 4.1: Viga curta em balanço.

| Malha # | NN | NE | NF | $\overline{\eta}$ (%) |
|---|---|---|---|---|
| 1 | 386 | 177 | 807 | 7.9 |
| 2 | 1122 | 609 | 2625 | 4.2 |
| 3 | 2495 | 1301 | 5721 | 2.3 |

Figura 4.2: Paramêtros obtidos para o exemplo 1.

Notação: NN = número de nós, NE = número de elementos, NF = número de graus liberdade.

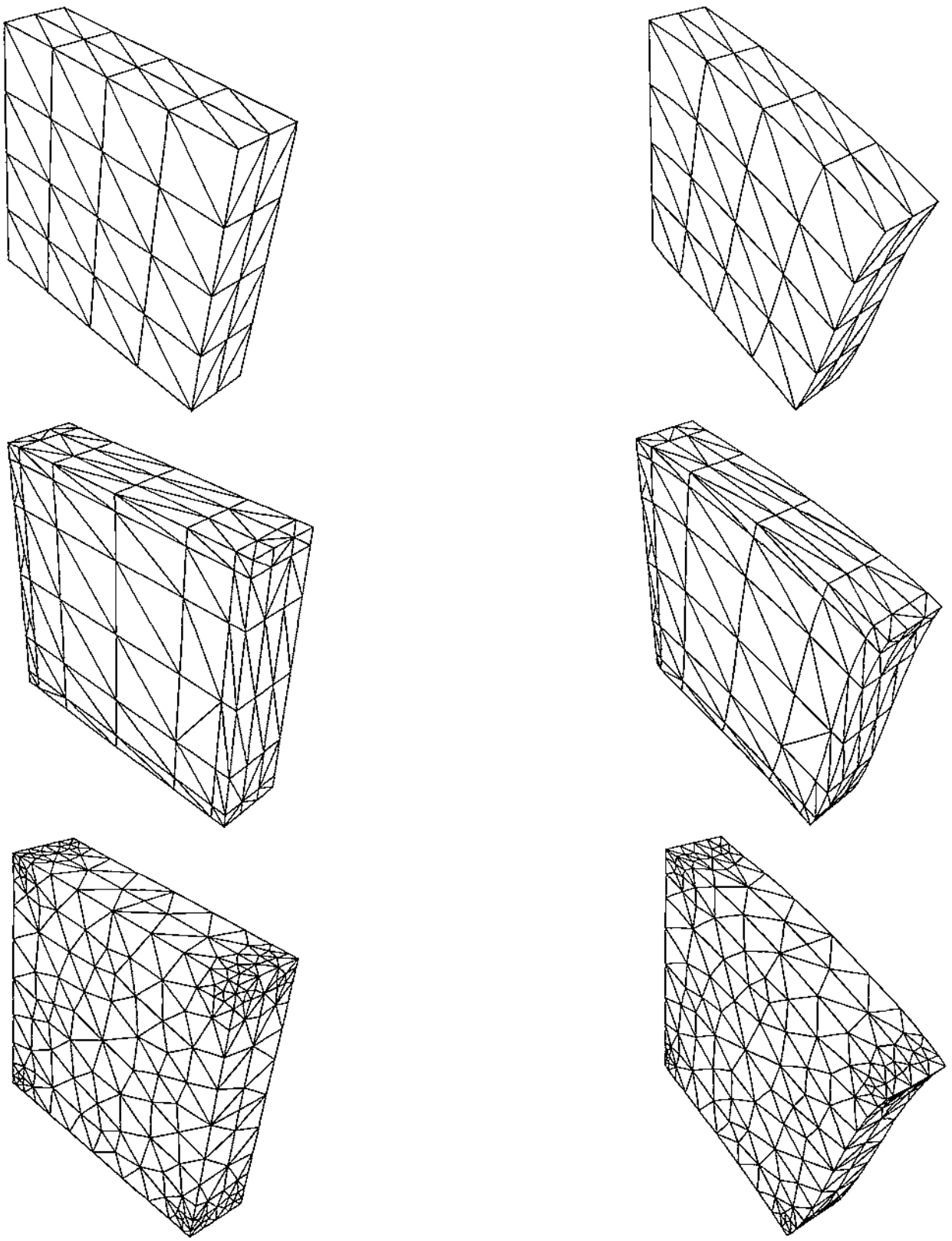


Figura 4.3: Refinamentos para viga curta em balanço.

## 4.2 *Exemplo 2 - Placa com furo circular*

O segundo exemplo é uma placa fina com um furo circular, submetida à uma pressão axial. As dimensões, condições de contorno e atributos para o problema são mostrados na figura 4.4. Devido à simetria, apenas um quarto da placa foi analisado.

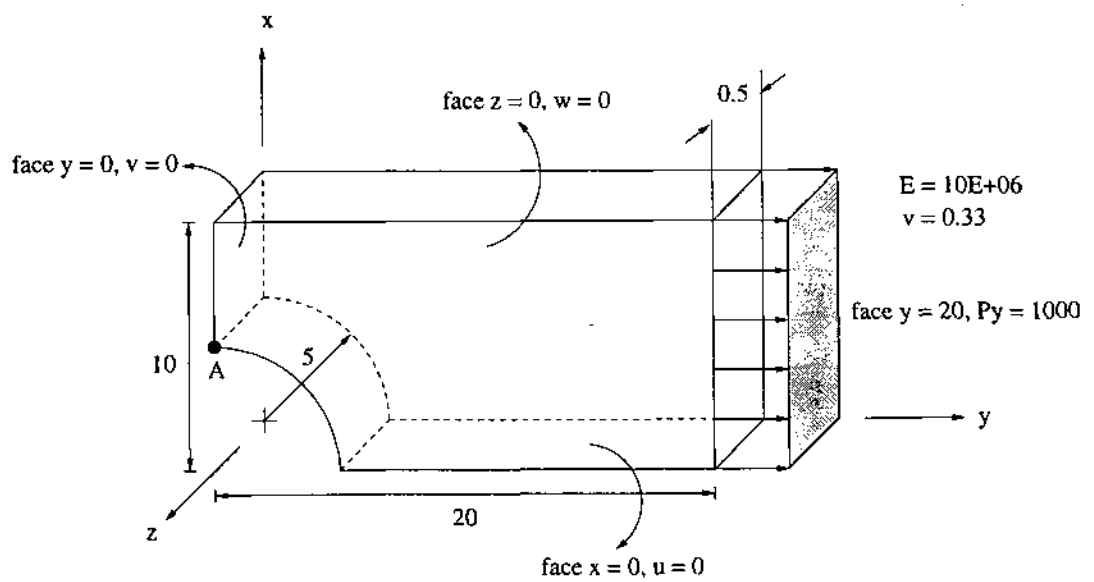



Figura 4.4: Placa com furo circular.

Três passos de refinamento também foram realizados para esse exemplo. A figura 4.5 mostra as três malhas usadas neste processo. A primeira malha, bem desrefinada com somente 12 elementos, foi feita baseada em uma mesma malha inicial usada em um trabalho recente que também analisou este mesmo problema (Schuetze *et al.*, 1995), para efeito de comparação. Analogamente a esse trabalho, foi atingido a convergência final do erro desejado. No presente trabalho, esse problema foi analisado usando-se as duas técnicas de suavização descritas, *SPR* e *REP*. Além disso, foram comparados os resultados usando-se um método alternativo de recuperação, baseado em um procedimento de projeção local (Hinton e Campbell, 1974) e pela média das tensões nos nós, que vai ser referenciado como *HC*. A figura 4.6 mostra uma tabela com os paramêtros importantes da análise. A figura 4.7 mostra os valores da tensão máxima na direção y para o ponto $A$ mostrado na figura 4.4, comparados com o valor teórico calculado para esse ponto.

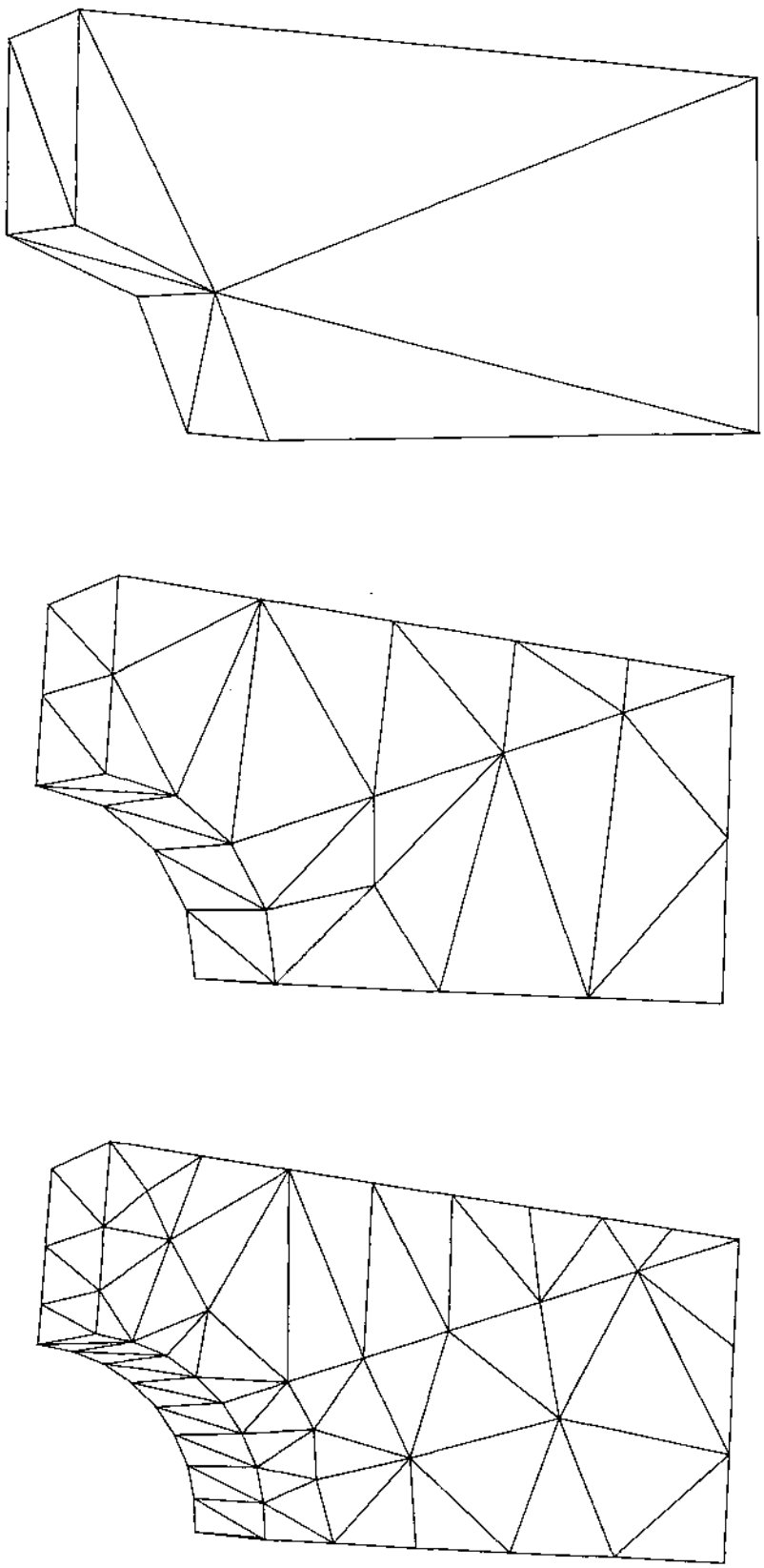


Figura 4.5: Refinamentos para placa com furo circular.

| Malha # | NN | NE | NF | $\overline{\eta}$ (%) [HC] | $\overline{\eta}$ (%) [SPR] | $\overline{\eta}$ (%) [REP] |
|---|---|---|---|---|---|---|
| 1 | 45 | 12 | 54 | 10.5 | 17.9 | 15.8 |
| 2 | 313 | 160 | 684 | 6.5 | 8.1 | 7.9 |
| 3 | 604 | 312 | 1326 | 4.2 | 3.8 | 3.5 |

Figura 4.6: Paramêtros obtidos para o exemplo 2.

| Enfoque | Tensão $\sigma_{yy}$ em A | Erro em relação à teoria (%) |
|---|---|---|
| HC | 4424 | 2.41 |
| SPR | 4275 | 1.04 |
| REP | 4280 | 0.92 |
| TEORIA | 4320 | — |

Figura 4.7: Resultados no ponto A para a última malha do exemplo 2.

Os resultados mostrados levam a algumas conclusões a respeito das técnicas de recuperação descritas. Pode-se notar da tabela na figura 4.6 que para as duas primeiras malhas os resultados decorrentes da suavização normal por elementos finitos (média nodal) foram melhores do que as técnicas de *SPR* e *REP*. Para a última malha, entretanto, essas técnicas superaram a suavização. Por outro lado, a tabela da figura 4.7, que é relativa à malha do último passo, demonstra que os valores da tensão para o ponto $A$ são melhores aproximados em relação à teoria, usando as técnicas de recuperação descritas neste trabalho, do que pela técnica de suavização. Pode-se então discutir alguns pontos importantes em relação a esses resultados:

- As técnicas de *SPR* e *REP* são obviamente extremamentes dependentes dos *patches* a serem formados, que por sua vez dependem das malhas de elementos finitos. Isto explica em parte, o não muito bom desempenho na primeira malha, por exemplo, onde nota-se que os *patches* a serem formados são muito pobres e conseqüentemente os resultados são abaixo da suavização simples. Na última malha, onde os *patches* são de melhor qualidade, os resultados são superiores à suavização.
- Tem-se discutido muito na literatura, desde a proposição destas técnicas de recuperação de valores baseadas em *patches*, o problema da efetividade da recuperação de valores no contorno dos modelos, onde geralmente se tem os gradientes mais elevados. Isto inclusive foi discutido neste trabalho no capítulo 2. Esse problema pode causar o mau condicionamento da matriz $A$ mostrada no capítulo 2, podendo esta matriz em alguns casos ser singular. O fato de evitar que nós do contorno e na fonteira de materiais formem *patches*, recuperando seus valores por *patches* formados por nós interiores, evita em muitos casos esses problemas de mau condicionamento da matriz e, em duas dimensões, trabalhos anteriores deste autor (Cavalcante Neto *et al.*, 1998; Paulino *et al.*, 1998) mostraram que é possível obter resultados bem apurados e convergências de erro bem eficientes. Em três dimensões esse problema pode ser mais complexo, como observado nestes exemplo e citado em referências da literatura.

Apesar então de ter-se obtido bons resultados usando as técnicas de *SPR* e *REP*, algumas melhorias podem ser sugeridas:

- É preferível que o procedimento de mínimos quadrados envolvidos na formulação das técnicas de recuperação seja realizado em um sistema local de coordenadas do *patch* para evitar dificuldades numéricas (Zienkiewicz e Zhu, 1994).
- Além disso, termos adicionais de equilíbrio devem ser introduzidos tanto na formulação da técnica de *SPR* quanto na de *REP*, somando aos funcionais dados pela

parcelas normais das formulações mostradas nas equações (2.20) e (2.32), respectivamente, um termo de equilíbrio adicional (Zienkiewicz e Zhu, 1997; Lee e Lo, 1997).

- Deve-se avaliar mais profundamente o uso da ponderação descrito no capítulo 2 (pelo peso $\omega$ da expressão (2.18)). Avaliações preliminares em outros trabalhos (Mosalam e Paulino, 1997) concluiram a eficiência da ponderação para outros casos que não tridimensionais. Investigações para esse uso em problemas tridimensionais sugerem uma possível melhoria na solução.

## 4.3 *Exemplo 3 - Fecho de porta de avião*

O terceiro exemplo é um fecho de porta de avião, cujas dimensões, condições de contorno e atributos são mostrados na figura 4.8. Esse problema foi analisado por Martha (1989), usando uma formulação de elementos de contorno. Naquele trabalho foi feita uma análise com propagação da trinca através do uso de fatores de intensidade de tensão. No presente trabalho, entretanto, não se tem a intenção de fazer uma análise de propagação tridimensional de trinca usando elementos finitos. Quer-se mostrar a capacidade do sistema de analisar problemas tridimensionais práticos e relativamente complexos, envolvendo os aspectos de modelagem, geração de malha volumétrica descrito neste trabalho e análise por elementos finitos. Para tanto, primeiramente é feita uma análise no modelo sem trinca, onde a região de concentração de tensão é mostrada e a seguir uma trinca passante é inserida nesta região e uma nova análise é efetuada. A figura 4.9 mostra a imagem sombreada do sólido para esse problema.

A figura 4.10 mostra a malha de elementos finitos utilizada no modelo sem trinca, usada em uma primeira análise de elementos finitos. Na verdade, a malha de superfície usada na análise inicial é a mesma malha, com exceção da malha na trinca, usada na análise considerando-se essa trinca feita a seguir.

Resultados da tensão normal na direção vertical da análise inicial deste modelo são mostradas nas figuras 4.11 e 4.12, onde são pintados contornos de tensões sobre a configuração deformada do modelo. A imagem inferior da figura 4.11 mostra um corte por um plano vertical.

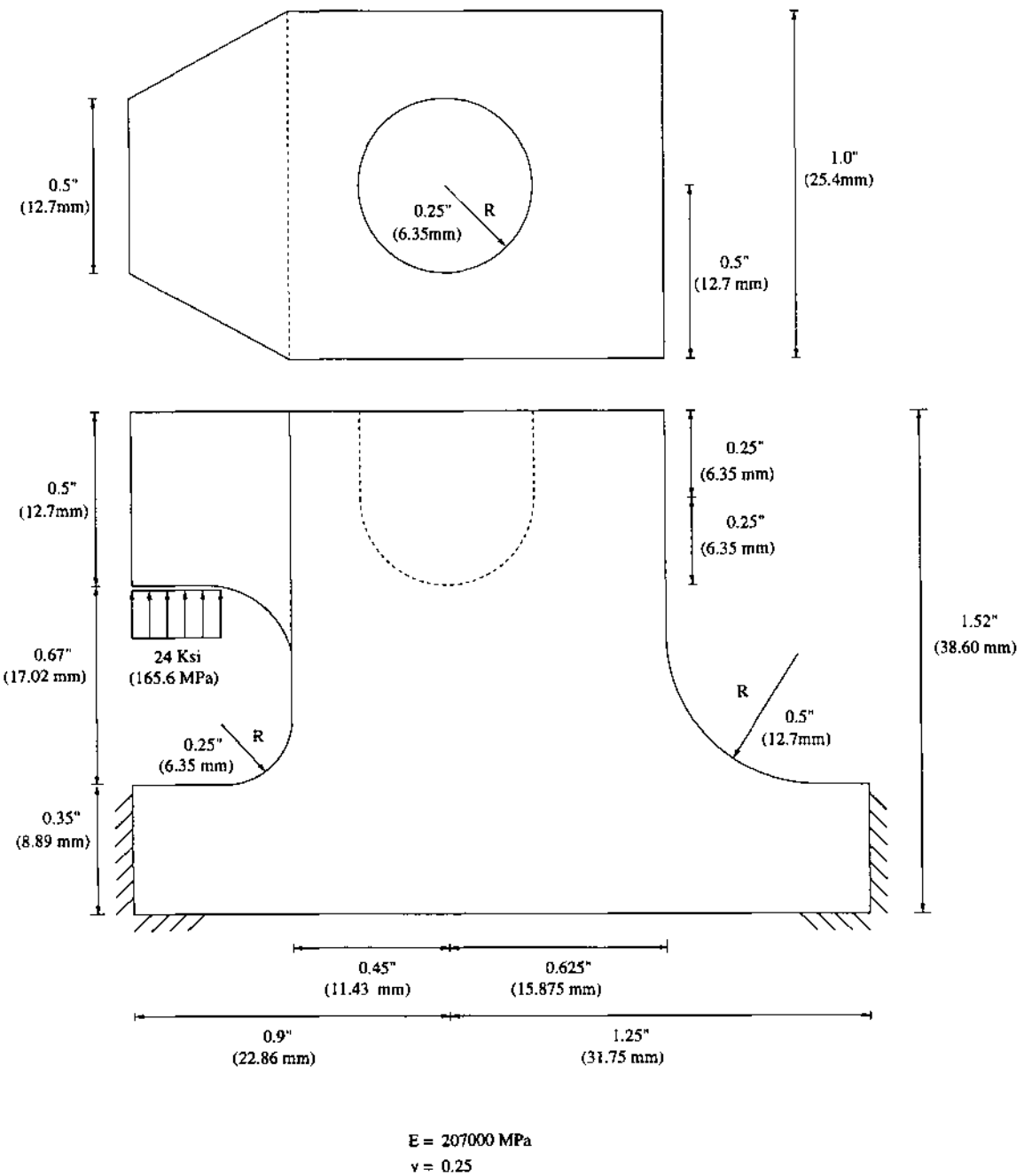



Figura 4.8: Fecho de porta de avião.

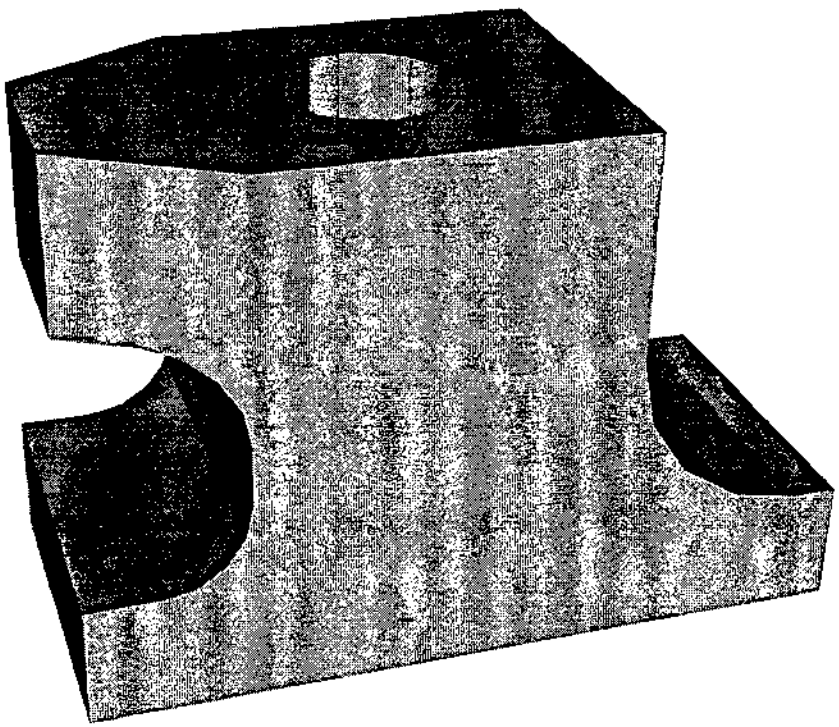


Figura 4.9: Visão sombreada do fecho de porta de avião.

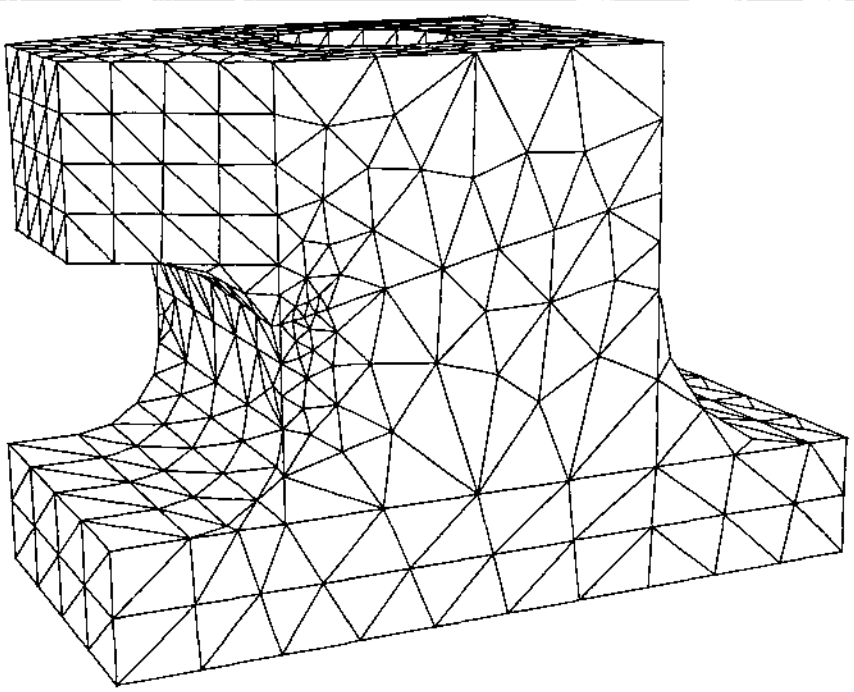


Figura 4.10: Modelo e malha inicial.

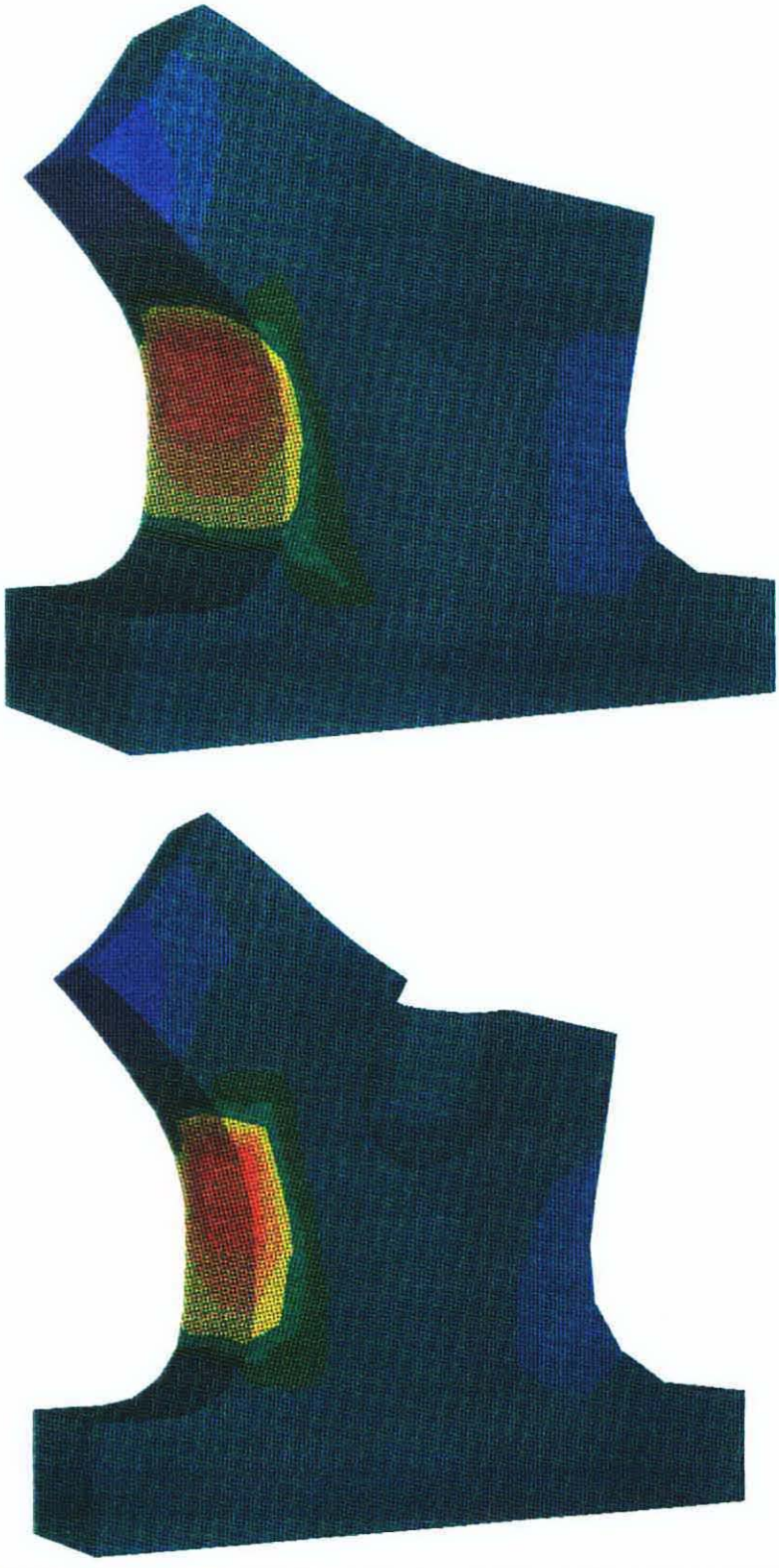


Figura 4.11: Contorno de tensões normais na direção vertical para o modelo.

Figura 4.12: Detalhe do contorno de tensões normais na direção vertical para o modelo.

Pode notar claramente pelas figuras 4.11 e 4.12 uma região de concentração de tensões. Baseado nisso, uma trinca passante foi arbitrariamente inserida nesta região. A figura 4.13 mostra o modelo com a trinca e a malha usada para este passo. As figuras seguintes mostram a configuração do contorno das mesmas tensões para a peça já com a trinca.

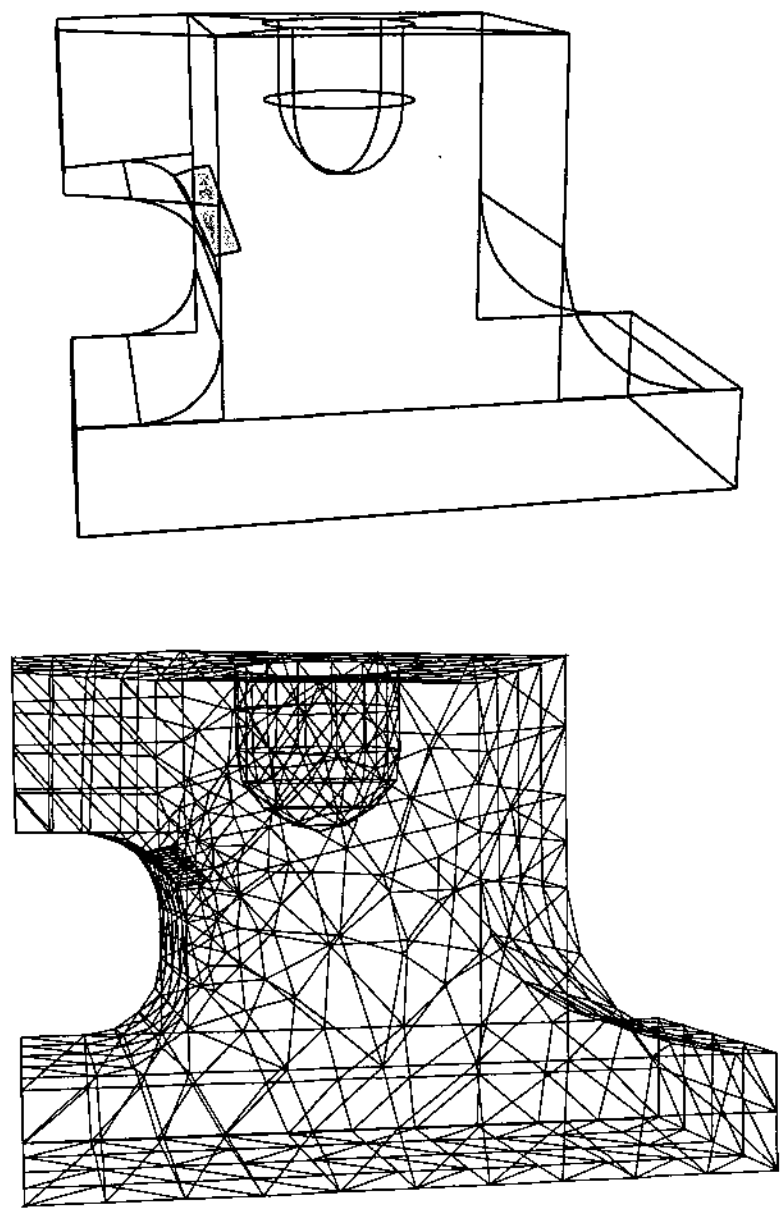


Figura 4.13: Modelo com trinca e malha.

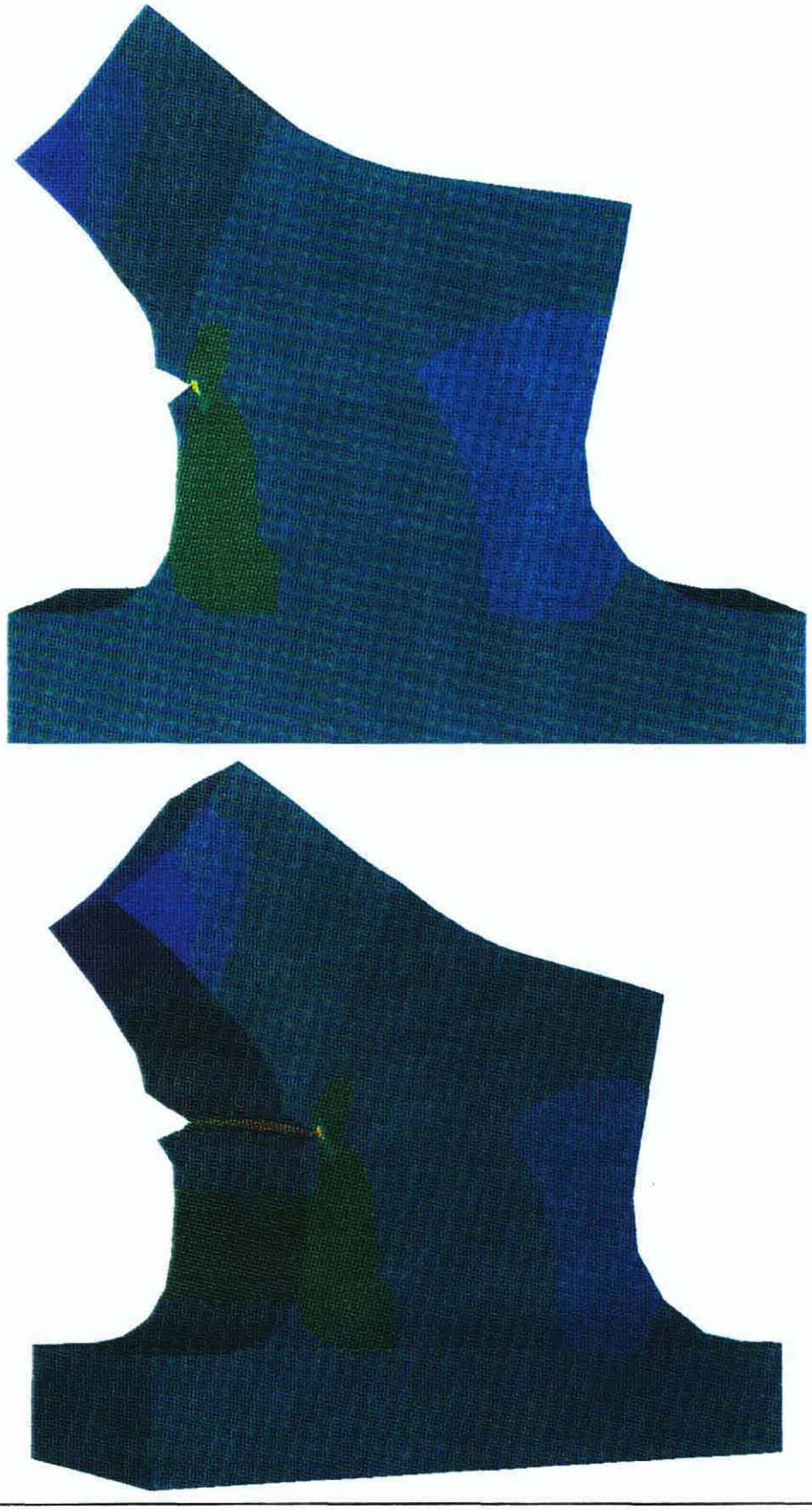


Figura 4.14: Contorno de tensões normais na direção vertical para o modelo.

Figura 4.15: Detalhe do contorno de tensões tensões normais na direção vertical para o modelo.

# 5

# Conclusão

Este trabalho propõe uma técnica de geração de malhas volumétricas de elementos tetraédricos, para domínios tridimensionais arbitrários. A principal motivação no desenvolvimento deste algoritmo foi a necessidade da consideração de alguns requisitos geométricos que não podem ser tratados por outros algoritmos encontrados na literatura, especialmente em problemas de fraturamento. Além disso, este trabalho apresenta uma implementação genérica, baseada em programação orientada a objetos, de eficientes estimadores de erro de discretização encontrados na literatura, para modelos de elementos finitos tridimensionais. As técnicas apresentadas são integradas em um sistema gráfico computacional, permitindo modelagem, análise e visualização de modelos sólidos de elementos finitos. Este sistema serve como base para validar essas técnicas e como protótipo para um futuro sistema para simulações genéricas adaptativas tridimensionais usando o método dos elementos finitos.

## 5.1 *Principais contribuições*

Procedimentos eficazes para a geração de malhas volumétricas e estimativa de erro de discretização compatíveis são fundamentais no desenvolvimento de uma estratégia adaptativa tridimensional para o método dos elementos finitos. Neste contexto, uma importante contribuição deste trabalho é o algoritmo para geração de malhas. A importância da contribuição pode ser medida pelos requisitos atendidos por esta técnica:

- O algoritmo é capaz de gerar malhas volumétricas válidas para domínios arbitrários, incluindo trincas, produzindo elementos de boa qualidade e evitando a geração de elementos com forma ruim, sempre que possível. Além disso, os elementos com melhor qualidade são gerados próximos ao contorno do modelo, onde ocorrem os maiores gradientes do campo fundamental.
- O algoritmo gera malhas volumétricas que se conformam com uma malha de contorno fornecida, o que é importante em alguns problemas, como propagação de trincas em problemas de fraturamento, por exemplo.

- O algoritmo é capaz de gerar uma boa transição entre regiões de diferentes tamanhos de elementos no modelo. Esse aspecto é importante pois evita a geração de malhas onde todos ou a maioria dos elementos tem o mesmo tamanho, o que não é desejável em muitos casos.
- O algoritmo é capaz de tratar modelos com trincas, onde as superfícies representando os dois lados de uma trinca são distintas, mas geometricamente coincidentes. A consideração de trincas confere um grau maior de complexidade na geração de uma malha volumétrica.

A qualidade geométrica dos tetraedros gerados é garantida basicamente por três fatores. Primeiro, o algoritmo envolve, na sua fase inicial, um procedimento de contração de contorno que força a geração de elementos de boa qualidade no que diz respeito à sua forma. Segundo, em uma fase posterior do algoritmo, um procedimento de "volta-passo" é utilizado para gerar tetraedros com boa qualidade em regiões onde não foi possível completar a geração da malha. Finalmente, uma estratégia *a posteriori* de "volta-passo", semelhante à anterior, força a geração de tetraedros com boa qualidade geométrica em regiões onde elementos de forma ruim foram gerados.

A capacidade de gerar uma malha que atenda aos requisitos de boa transição entre regiões com diferentes tamanhos de elementos é conseguida com o uso de uma estrutura auxiliar de árvore octária (*octree*). A *octree* é gerada com base na discretização do contorno fornecida e, portanto, fica mais refinada nas regiões onde a malha de contorno é mais refinada. A *octree* é subseqüentemente refinada para forçar a boa transição de tamanhos de elementos. Isto é alcançado no procedimento de geração da malha por contração de contorno onde a *octree* é usada como um "pano-de-fundo" para a definição dos tamanhos dos elementos gerados.

Outra importante contribuição deste trabalho é uma implementação genérica, baseada em programação orientada a objetos, de eficientes técnicas de estimativa de erro de discretização encontradas na literatura. A programação orientada a objetos, implementada inicialmente para problemas bidimensionais, permitiu uma extensão para casos tridimensionais de uma maneira relativamente simples e eficiente. As diferenças entre as implementações bi e tridimensionais estão apenas nas montagens dos *patches* (grupos de elementos adjacentes) e nos métodos que identificam que nós de um *patch*, para um dado tipo de elemento finito, devem ser recuperados. Todos os outros métodos de montagem de matrizes e recuperação de gradientes, que foram implementados de uma forma genérica para o caso bidimensional, são re-usados sem modificação no caso

tridimensional. Esse tipo de implementação criou também uma plataforma que permite fácil expansão da estratégia para outros tipos de estimadores de erro e para consideração de outros tipos de elementos finitos tridimensionais.

A implementação dos estimadores de erro, juntamente com técnica de geração de malha volumétrica, permitiu a realização de simulações adaptativas tridimensionais e bons resultados foram obtidos. Entretanto, as técnicas de estimativa de erro usadas, *SPR* e *REP*, podem e devem ser melhoradas incluindo-se termos de equilíbrio adicionais nos funcionais da sua formulação. Outra melhoria necessária é a consideração da ponderação nos funcionais, conforme descrito no capítulo 2 e salientado no capítulo 4, que pode ser usada para tratar problemas em regiões de altos gradientes.

Finalmente, estas técnicas foram integradas em um protótipo de sistema gráfico computacional, possuindo características de modelagem, análise e visualização. Esse protótipo demonstrou que as técnicas propostas são viáveis e poderão, no futuro, resultar em um sistema de modelagem numérica tridimensional por elementos finitos de grande utilidade.

## 5.2 *Sugestões para trabalhos futuros*

Nas simulações adaptativas mostradas no capítulo 4, as malhas de contorno e volumétricas construídas para os refinamentos sucessivos foram geradas por este autor usando o programa *FRANC3D*. Isso foi feito modelando-se as sucessivas malhas de contorno baseando-se nas análises de erro de discretização de passos anteriores, e então aplicando-se o algoritmo para se obter a malha volumétrica correspondente. Uma sugestão para trabalho futuro é o desenvolvimento de procedimentos para realizar esse refinamento adaptativo tridimensional de forma automática, minimizando-se a intervenção do usuário. Para isso, as técnicas descritas neste trabalho são fundamentais.

Um possível enfoque seria a extensão do trabalho realizado pelo autor em duas dimensões (Cavalcante Neto, 1994). Neste enfoque, a discretização do domínio é precedida pela discretização das curvas do contorno. Ambas as discretizações são baseadas na estimativa de erros do passo anterior de análise. O refinamento das curvas de contorno é baseado em uma estrutura de dados de árvore binária e o refinamento do domínio é baseado em uma estrutura de dados de árvore quaternária (*quadtree*). A vantagem deste enfoque é uma discretização mais regular nas regiões próximas ao contorno do modelo.

Em três dimensões, um enfoque semelhante seria adotado. Primeiro as arestas do sólido seriam refinadas. Isto é baseado na estimativa de erros do passo anterior e com o auxílio de uma estrutura de dados de árvore binária. Em seguida as superfícies do sólido seriam refinadas com auxílio de uma estrutura de dados de subdivisões espaciais recursivas no espaço paramétrico das superfícies. Finalmente, o domínio do sólido seria discretizado com base no algoritmo descrito neste trabalho com a modificação que o refinamento da *octree* auxiliar também seria dependente da estimativa de erros do passo anterior.

Verificou-se neste trabalho que, apesar das técnicas de estimativa de erro de discretização apresentarem um bom desempenho em geral, algumas melhorias podem ser efetuadas. Uma sugestão é a introdução de termos adicionais de equilíbrio nos funcionais relativos às técnicas *SPR* e *REP* mostrados no capítulo 2. Além disso, sugere-se que os procedimentos de mínimos quadrados que envolvem essas técnicas sejam realizados em coordenadas locais dos *patches*, para evitar problemas numéricos. Outra sugestão é o estudo mais aprofundado da ponderação apresentada também no capítulo 2 para casos tridimensionais.

As simulações tridimensionais realizadas mostraram que, para problemas tridimensionais de tamanho razoável, pode ser bastante caro computacionalmente realizar as análises pelo método dos elementos finitos. Pode-se facilmente atingir um número grande de graus de liberdade para esses problemas em três dimensões. Sugere-se então o uso de técnicas de solução iterativas, onde não seja necessário montar matrizes globais do sistema, e de procedimentos de processamento paralelo para a análise dos modelos. A geração das malhas volumétricas pode igualmente levar muito tempo de execução para se obter uma malha, dependendo-se do modelo. A paralelização do algoritmo proposto também seria muito importante neste contexto.

Outra sugestão é a expansão do sistema proposto para abordar outros tipos de análise, além da análise de problemas de elasticidade linear elástica. O paradigma de programação orientada a objetos pode ser explorado para isso (Martha *et al.*, 1996). Além disso, a implementação de outros estimadores de erro de discretização e de outros tipos de elementos finitos seria também importante.

Finalmente, sugere-se a expansão do sistema para análises automáticas de propagação de trincas em problemas de fraturamento tridimensional pelo método dos elementos finitos.

# Apêndice

## Estudo de qualidade da forma de tetraedros

Pesquisa em busca da caracterização de uma "boa" malha e de métricas de qualidade tem aumentado muito nos últimos anos. Alguns pesquisadores tem tentado caracterizar elementos ruins baseado nestas métricas. A figura A.1 mostra, por exemplo, algumas configurações comuns de elementos tetraédricos de qualidade pobre encontrados na literatura.

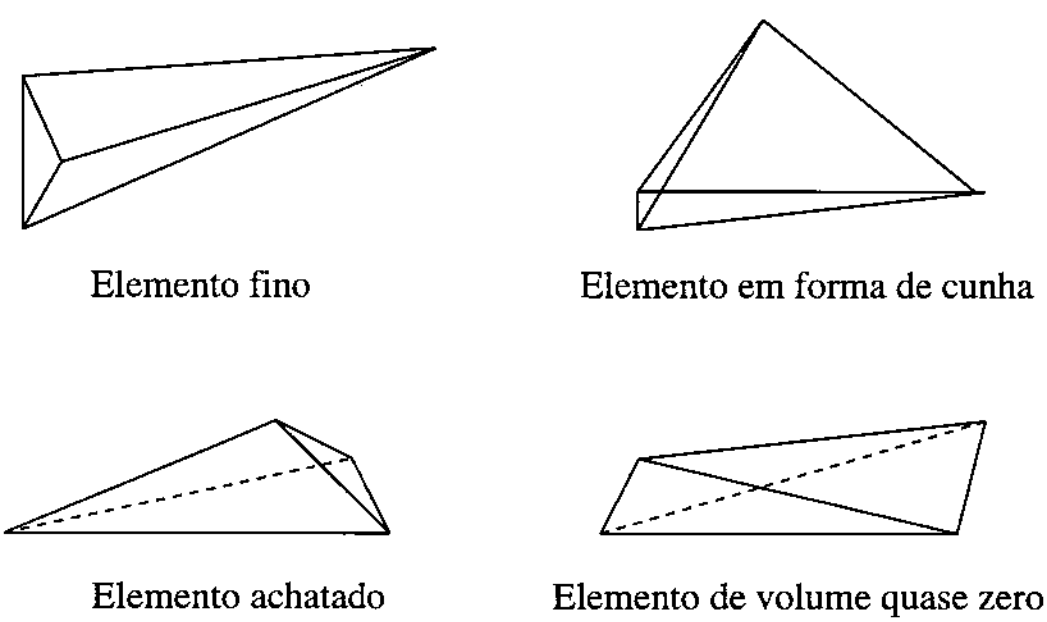



Figura A.1: Configurações comuns de elementos tetraédricos de qualidade pobre.

Uma série de métricas de qualidade encontradas na literatura tem motivação no método dos elementos finitos. Para a estabilidade numérica do método, é necessário que os elementos tenham algum limite característico relacionado com suas formas. Alguns estudos mostram que o condicionamento numérico das matrizes em uma análise bidimensional pelo métodos dos elementos finitos é relacionado com o ângulo mínimo na triangulação, assim como é o erro em uma aproximação por elementos finitos (Bramble e Zlámal, 1970; Ciarlet, 1970). Foi mostrado, entretanto, que a condição de ângulo mínimo pode ser muito restritiva e pode ser substituída por uma condição que limita

o máximo ângulo permitido (Babuška and Aziz, 1976). Estas referências fornecem justificação teórica provando-se a convergência do método dos elementos finitos quando o tamanho do triângulo diminui, ao longo de que o ângulo máximo afasta-se de 180 graus. Estas referências também mostram exemplos em que a convergência falha quando os ângulos crescem arbitrariamente muito pequenos. Alguns outros estudos relacionam outros aspectos de uma triangulação, como a máxima altura e o máximo comprimento do lado do triângulo, à acurácia numérica do método dos elementos finitos.

Baseado nestes aspectos descritos, várias métricas são propostas na literatura para caracterizar a forma de um tetraedro. A figura A.2 mostra uma tabela com métricas coletadas da literatura (Lewis *et al.*, 1996c; Parthasarathy *et al.*, 1993) [1].

| Métrica de Qualidade | Valor Ótimo | Intervalo |
|---|---|---|
| $\beta = \frac{R_c}{R_i}$ | $\beta^* = 3.0$ | $[1, \infty)$ |
| $\lambda = \frac{S_{max}}{R_i}$ | $\lambda^* = 4.898979$ | $[1, \infty)$ |
| $\omega = \frac{R_c}{S_{max}}$ | $\omega^* = 0.612372$ | $(0, \infty)$ |
| $\tau = \frac{S_{max}}{S_{min}}$ | $\tau^* = 1.0$ | $[1, \infty)$ |
| $\kappa = \frac{V^4}{(\sum_{i=0}^{3} A_i^2)^3}$ | $\kappa^* = 4.5724e-4$ | $(0, 1]$ |
| $\alpha = \frac{S_{avg}^3}{V}$ | $\alpha^* = 8.485281$ | $[1, \infty)$ |
| $\gamma = \frac{S_{rms}^3}{V}$ | $\gamma^* = 8.485281$ | $[1, \infty)$ |
| $\delta$ | $\delta^* = 1.2309594$ | $[1, 1.25522]$ |
| $\theta_{min}$ | $\theta_{min}^* = 0.551285$ | $(0, 1]$ |
| $\eta = \frac{12(3V)^{2/3}}{\sum_{i=0}^{5} S_i^2}$ | $\eta^* = 1.0$ | $(0, 1]$ |

Figura A.2: Lista de métricas de qualidade encontradas na literatura.

[1] Nomenclatura: $R_c$ = raio da esfera circunscrita, $R_i$ = raio da esfera inscrita, $S_i$ = comprimento de uma aresta i, $S_{max}$ = max($S_i$) com (i=0..5), $S_{min}$ = min($S_i$) com (i=0..5), A = área de uma face triangular, V = volume do tetraedro, $S_{avg}$ = média($S_i$) com (i=0...5), $S_{rms} = \sqrt{\frac{1}{6}\sum_{i=0}^{5} S_i^2}$.

Em triangulações bidimensionais, o ângulo interior mínimo de um triângulo é uma métrica usualmente usada e conseqüentemente o ângulo sólido mínimo $\theta_{min}$ de um tetraedro é sua extensão para casos tridimensionais. A condição de ângulo sólido mínimo, entretanto, pode ser muito restritiva como explicado anteriormente, mas pode ser reformulada para um requisito de que o círculo inscrito não seja muito pequeno e essa condição é facilmente extendida para três dimensões, gerando a métrica definida pelo razão $\beta$ do raio da esfera circunscrita ao tetraedro pelo raio da esfera inscrita (Cavendish *et al.*, 1985; Joe, 1991). Uma outra métrica, $\delta$, relacionada com ângulos é a que considera o máximo ângulo diédrico (Weatherill e Hassan, 1994). Considerando-se métricas relacionadas com comprimento, a razão $\lambda$ do máximo comprimento de uma aresta do tetraedro pelo raio da esfera inscrita e a razão do máximo comprimento de uma aresta do tetraedro pelo mínimo comprimento, são métricas propostas na tentativa de caracterizar tetraedros finos e achatados (Baker, 1989). A razão média (*mean ratio*) $\eta$, baseada em uma transformação afim do tetraedro regular foi também usada para analizar a qualidade das malhas geradas por um processo de bisseção (Joe, 1995). O volume do tetraedro e as áreas das quatro facetas que o compõem são também usados para especificar uma outra métrica, $\kappa$, que define uma razão de aspecto normalizada (Cougny *et al.*, 1990). Além disso, a razão $\alpha$ envolvendo o volume do tetraedro e a média do comprimento de suas seis arestas é outra métrica proposta (Dannelogue e Tanguy, 1991), a qual foi modificada substituindo-se a média do comprimento das seis arestas do tetraedro pela raiz quadrada média (*root mean square*) do comprimento dessas seis arestas (Parthasarathy *et al.*, 1993), resultando na razão $\gamma$. Os valores ótimos das métricas da forma de um tetraedro mostradas na figura A.2 correspondem ao tetraedro regular no qual o comprimento de suas arestas é igual à unidade.

Considerando-se as métricas propostas, surgem algumas questões relacionadas à escolha da métrica que deve ser usada ou à capacidade das métricas de caracterizar a qualidade de um tetraedro. Usando os valores ótimos para todas as métricas, elas podem ser normalizadas para melhores comparações. Além disso, uma medida global da qualidade de uma malha pode ser definida como a média dos valores de uma dada métrica para todos os elementos da malha.

Um trabalho recente (Liu e Joe, 1994) provou que, para qualquer par $u$, $v$ das métricas $\eta$, $\sigma_{min}$, e $\rho$ existem constantes positivas $c_0$, $c_1$, $e_0$, e $e_1$ tais que $c_0 \cdot u^{e_0} \leq v \leq c_1 \cdot u^{e_1}$, onde $u$ ou $v$ denotam uma das três métricas comparadas. Isto significa que estas três métricas são "equivalentes". Adotando-se uma versão escalada de $\sigma_{min}$ onde

$\sigma = 3\sqrt{6}\sigma_{min}/2$, tal que $\sigma = 1$ para o tetraedro regular, mostrou-se que, embora estas métricas sejam "equivalentes", elas não se aproximam do valor de um tetraedro de pior forma (no caso 0) e o valor ótimo para o tetraedro equilátero (neste caso 1) na mesma taxa. Estes autores mostraram que, para tetraedros de boa forma, quando as métricas são próximas de 1, uma ordem $\eta > \rho > \sigma$ é observada [2]. Essa ordem mencionada indica que $\rho$ tende a distribuir os valores de sua métrica mais uniformemente no interior do intervalo $[0, 1]$ e isto indica uma preferência pelo uso da métrica $\beta$ para avaliar a qualidade de uma malha.

Outro trabalho (Parthasarathy *et al.*, 1993) comparou as métricas $\beta$, $\lambda$, $\omega$, $\tau$, $\kappa$, $\alpha$, e $\gamma$, usando uma série de testes de sensitividade que avaliam a fidelidade destas métricas de qualidade descritas. Estes testes são baseados na natureza das distorções que ocorrem durante estágios da geração da malha ou de qualquer processo de melhoria dessa malha (suavização, por exemplo) de um gerador de malha tetraédrica. Foram definidos quatro testes para avaliar estas métricas:

Teste A: Este teste começa com o tetraedro equilátero. A figura A.3 mostra a configuração do elemento para este teste. O ápice do tetraedro é movido em incrementos pequenos ao longo da linha vertical que liga este ápice ao centróide da faceta oposta. Para localizações do ápice próximas da base do tetraedro, este teste simula a forma de um tetraedro achatado da figura A.1, e para alturas grandes do ápice, a forma de um elemento fino da mesma figura.

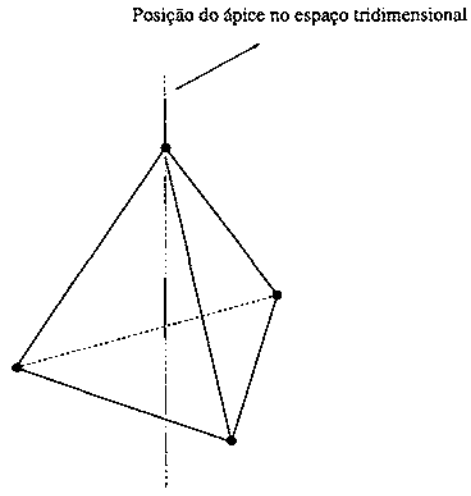



Figura A.3: Configuração do elemento para o teste A.

[2] Para evitar o uso de funções trigométricas, $\sigma_{min} = \sin(\theta_{min}/2)$ foi na verdade usado ao invés da métrica $\theta_{min}$. Também para facilitar as comparações, $\rho = 3/\beta$ foi usado ao invés de $\beta$.

Test B: Este teste começa com o tetraedro equilátero como mostrado na figura A.4. Ao ápice do tetraedro é permitido traçar, em incrementos pequenos, um quarto de arco definido pela altura do tetraedro como raio. Na posição final, o ápice se localiza no plano definido pela faceta do tetraedro oposta, simulando um elemento de volume quase zero da figura A.1.

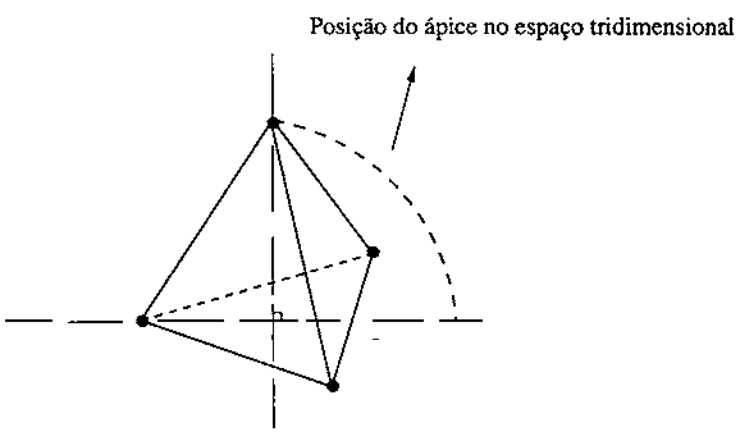



Figura A.4: Configuração do elemento para o teste B.

Test C: Este teste muda o comprimento de uma das arestas mais curtas de um elemento em forma de cunha da figura A.1, enquanto preserva a sua configuração original. A figura A.5 mostra a configuração do elemento para este teste.

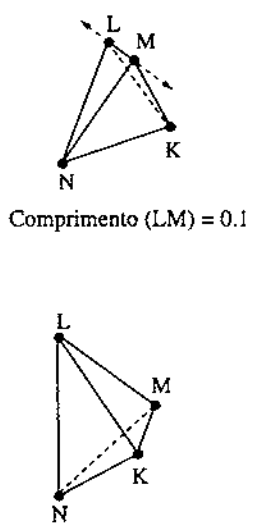
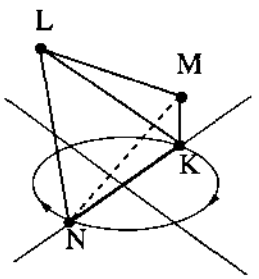
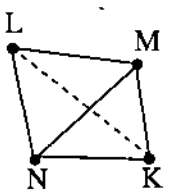



Figura A.5: Configuração do elemento para o teste C.

Test D: Este teste faz uma das arestas mais curtas de um elemento em forma de cunha da figura A.1 ficar paralela à outra aresta, permitindo que dois nós da aresta tracem simultaneamente um quarto de arco em um plano. A figura A.6 mostra a configuração do elemento para este teste.




Configuração Inicial




Configuração Final

[Nós N,K traçam um quarto de círculo,
enquanto nós L,M permanecem fixos]

Figura A.6: Configuração do elemento para o teste D.

Das observações baseadas nestes testes, este trabalho concluiu que as métricas $\alpha$, $\beta$, $\lambda$, e $\gamma$ são capazes de caracterizar distorções para todos os testes. A métrica $\tau$ é boa apenas para avaliar distorções onde pelo menos um dos comprimentos das arestas desenvolve uma mudança significativa, como observado nos testes A e C. A métrica $\kappa$ é também sensível à todos os testes exceto que, diferente das outras métricas, um valor menor indica uma distorção maior e vice-versa. Uma análise do esforço computacional foi então realizada baseada nas tarefas envolvidas no cálculo de cada uma dessas métricas, como o cálculo do comprimento das seis arestas de um tetraedro e o volume do elemento para a métrica $\alpha$, por exemplo. A conclusão recomenda o uso da métrica $\gamma$ como uma boa/computacionalmente-barata medida de qualidade, embora seja salientado que esse esforço computacional depende da implementação e da estrutura de dados usada na geração da malha e das técnicas de melhoria de qualidade usadas na criação dessa malha.

# Referências bibliográficas


(Ainsworth *et al.*, 1989)

Ainsworth, M.; Zhu, J.Z.; Craig, A.W. e Zienkiewcz, O.C. - "Analysis of the Zienkiewicz-Zhu *a-posteriori* Error Estimator in the Finite Element Method," *Int. J. Num. Meth. Engng.*, vol.28, pp. 2161-2174, 1989.

(Babuška e Aziz, 1976)

Babuška, I. e Aziz, A.K. - "On the Angle Condition in the Finite Element Method," *SIAM J. Num. Anal.*, vol. 13, pp. 214-226, 1976.

(Babuška e Rheinboldt, 1976)

Babuška, I. e Rheinboldt, W.C. - "Analysis of Optimal Finite Element Meshes in $R^1$," *Mathematics of Computations*, vol. 33, no. 146, pp. 435-463, 1976.

(Babuška e Rheinboldt, 1978)

Babuška, I. e Rheinboldt, W.C. - "*A-posteriori* Error Estimates for Finite Element Problems," *Int. J. Num. Meth. Engng.*, vol. 12, pp. 1597-1615, 1978.

(Babuška e Rheinboldt, 1979)

Babuška, I. e Rheinboldt, W.C. - "Adaptive Approaches and Reliability Estimation in Finite Element Analysis," *Comp. Meth. in Applied Mec. and Engn.*, vol. 18, pp. 519-540, 1979.

(Babuška e Szabó, 1991)

Babuška, I. e Szabó, B. - *Finite Element Analysis*, Jonh Wiley & Sons, 1991.

(Babuška *et al.*, 1994a)

Babuška, I.; Strouboulis, T.; Upadhyay, C. S. e Gangaraj, S.K. - "A Model Study of the Quality of *a posteriori* Estimators for Linear Elliptic Problems Error Estimation in the Interior of Patchwise Uniform Grids of Triangles," *Comp. Meth. in Applied Mec. and Engn.*, vol. 114, pp. 307-378, 1994.

(Babuška *et al.*, 1994b)

Babuška, I.; Strouboulis, T.; Upadhyay, C. S.; Gangaraj, S.K. e Copps, K. - "Validation of *a posteriori* Error Estimators by a Numerical Approach," *Int. J. Num. Meth. Engng.*, vol. 37, pp. 1073-1123, 1994.

(Babuška *et al.*, 1994c)

Babuška, I.; Von Petersdorff, T. e Andersson B. - "Numerical Treatment of Vertex Singularities and Intensity Factors for Mixed Boundary Value Problems for the Laplace Equation in $\mathbf{R}^3$," *SIAM J. Appl. Math.*, vol. 31, pp. 1265-1288, 1994.

(Baker, 1987)

Baker, T.J. - "Three Dimensional Mesh Generation by Triangulation of Arbitrary Point Sets," *AIAA-87-1124-CP*, 1987.

(Baker, 1989)

Baker, T.J. - "Element Quality in Tetrahedral Meshes," *Proc. 7th Int. Conf. on Finite Element Meth. in Flow Problems*, Huntsville, AL, 1989.

(Baehmann *et al.*, 1987)

Baehmann, P.L.; Wittchen, S.L. e Shephard, M.S. - "Robust Geometrically Based, Automatic Two-Dimensional Mesh Generation," *Int. J. Num. Meth. Engng.*, vol. 24, pp. 1043-1078, 1987.

(Baehmann e Shephard, 1989)

Baehmann, P.L. e Shephard, M.S. - "Adaptive Multiple-Level $h$-Refinement in Automated Finite Element Analysis," *Engng. with Comput.*, vol. 5, pp. 235-247, 1989.

(Baehmann, 1989)

Baehmann, P.L. - "Automated Finite Element Modeling e Simulation," Ph. D. Thesis, Rensselaer Polytechnic Institute, Troy - New York, 1989.

(Barbalho, 1994)

Barbalho, V.M.S. - "Triangulação Adaptativa de Sólidos CSG para Geração de Malhas de Elementos Finitos," Dissertação de Mestrado, COPPE, UFRJ, 1994.

(Blacker e Belytschko, 1994)

Blacker, T. e Belytschko, T. - "Superconvergent Patch Recovery with Equilibrium and Conjoint Interpolant Enhancements", *Int. J. Num. Meth. Engng.*, vol. 37, pp. 517-536, 1994.

(Borouchaki e Lo, 1995)

Borouchaki, H. e Lo, S.H. - "Fast Delaunay Triangulation in Three Dimensions," *Comp. Meth. in Appl. Mec. and Engng.*, vol. 128, pp. 153-167, 1995.

(Boroomand e Zienkiewicz, 1997)

Boroomand, B. e Zienkiewicz, O.C. - "Recovery by Equilibrium in Patches (REP)," *Int. J. Num. Meth. Engng.*, vol. 40, pp. 137-164, 1997.

(Bramble e Zlámal, 1970)

Bramble, J.H. e Zlámal, M. - "Triangular Elements in the Finite Element Method," *Math. Comp.*, vol. 24, pp. 809-810, 1970.

(Carvalho, 1995)

Carvalho, M.T.M. - "Uma Estratégia para o Desenvolvimento de Aplicações Configuráveis em Mêcanica Computacional," - Tese de Doutorado, Departamento de Engenharia Civil, PUC-Rio, 1995.

(Cavalcante Neto *et al.*, 1993a)

Cavalcante Neto, J.B.; Carvalho, M.T.M. e Martha, L.F. - "Combinação das Técnicas de Quadtree e Delaunay para Geração Automática de Malhas de Elementos Finitos," anais do VI SIBGRAPI, Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens, Recife - PE, pp. 285-291, 1993.

(Cavalcante Neto *et al.*, 1993b)

Cavalcante Neto, J.B.; Carvalho, M.T.M. e Martha, L.F. - "Geração Auto-Adaptativa de Malhas de Elementos Finitos Utilizeo uma Técnica de Enumeração Espacial Recursiva," anais do XIV CI-LAMCE, Congresso Ibero-Latino Americano sobre Métodos Computacionais para Engenharia, São Paulo - SP, pp. 1285-1294, 1993.

(Cavalcante Neto, 1994)

Cavalcante Neto, J.B. - "Simulação Auto-Adaptativa Baseada em Enumeração Espacial Recursiva de Modelos Bidimensionais de Elementos Finitos," Dissertação de Mestrado, Departamento de Engenharia Civil, 1994.

(Cavalcante Neto *et al.*, 1998)

Cavalcante Neto, J.B.; Martha, L.F.; Menezes, I.F.M. e Paulino, G.H. - "A Methodology for Self-Adaptive Finite Element Analysis using an Object Oriented Approach," IV-WCCM: Fourth World Congress on Computational Mechanics, Buenos Aires, Argentina, 1998.

(Cavendish *et al.*, 1985)

Cavendish, J.C.; Field, D.A. e Frey, W.H. - "An Approach to Automatic Three-dimensional Finite Element Mesh Generation," *Int. J. Num. Meth. Engng.*, vol 21, pp. 329-347, 1995.

(Celes Filho *et al.*, 1991)

Celes Filho, W.; Martha, L.F. e Gattass, M. - "An Efficient Data Structure for Solid Finite Element Analysis Post-Processor (*in Portuguese*)," XI Brazilian Congress of Mechanic Engineering, pp. 105-108, 1991.

(Chew, 1989)

Chew, L.P. - "Constrained Delaunay Triangulation," *Algorithmica*, vol. 4, pp. 97-108, 1989.

(Ciarlet, 1970)

Ciarlet, P.G. - "Orders of Convergence in Finite Element Method," *The Mathematics of Finite Elements and Applications*, J.R. Whiteman ed., Academic Press, New York - NY, pp. 59-82, 1970.

(Cook, 1989)

Cook, R.D.; Malkus, D.S. e Plesha, M.E. - *Concepts and Applications of Finite Element Analysis*, John Wiley, New York - NY, 1989.

(Coorevits *et al.*, 1994)

Coorevits, P.; Ladeveze, P. e Pelle, J.P. - "Mesh Optimization for Problems with Steep Gradients," *Eng. Comput. (Swansea Wales)*, vol. 11, pp. 129-144, 1994.

(Cougny *et al.*, 1990)

Cougny, H.L.; Shephard, M.S. e Georges, M.K. - "Explicit Node Smoothing Within Octree," Report 10-1990, SCOREC, Rensselaer Polytechnic Institute, Troy - New York, 1990.

(Cyrino, 1989)

Cyrino, J.C.R. - "Convergência Acelerada pela Relocalização de Nós e Refinamento de Malhas de Elementos Finitos," Tese de Doutorado, COPPE/UFRJ, 1989.

(Dannelongue e Tanguy, 1991)

Dannelongue, H.H. e Tanguy, P.A. - "Three-dimensional Adaptive Finite Element Computations and Applications to non-Newtonian Flows," *Int. J. Num. Meth. Engng.*, vol. 13, pp. 145-165, 1991.

(Field, 1986)

Field, D.A. - "Implementing Watson's Algoritm in Three Dimensions", Proc. 2nd ACM Symposium on Computational Geometry, pp. 246-259, 1986.

(Foley *et al.*, 1990)

Foley, J.D.; van Dam, A.; Feiner, S.K. e Hughes, J.F. - *Computer Graphics (Principles and Practice)*, 2nd. edition, Wesley Publishing Company, Reading - MA, 1990.

(Gomes Coelho, 1998)

Gomes Coelho, L.C. - "Modelagem de Cascas com Interseções Paramétricas", Tese de Doutorado, Departamento de Informática, PUC-Rio, 1998.

(Haber *et al.*, 1981)

Haber, R., Shephard, M.S., Abel, J.F., Gallagher, R.H., e Greenberg, D.P. - "A General Two-Dimensional, Graphical Finite Element Pre-processor Utilizing Discrete Transfinite Mappings," *Int. J. Num. Meth. Engng.*, vol 17, pp. 1015-1044, 1981.

(Hinton e Campbell, 1974)

Hinton, E. e Campbell, J.S. - "Local and Global Smoothing of Discontinuous Finite Element Functions Using a Least Squares Method," *Int. J. Num. Meth. Engng.*, vol. 8, pp. 461-480, 1974.

(Joe, 1986)

Joe, B. - "Delaunay Triangular Meshes in Convex Polygons," *SIAM J. Sci. Stat. Comput.*, vol. 7, pp. 514-539, 1986.

(Joe, 1989)

Joe, B. - "Three-dimensional Triangulations from Local Transformations," *SIAM J. Sci. Stat. Comput.*, vol. 10, pp. 718-741, 1989.

(Joe, 1990)

Joe, B. - "Construction of Three-Dimensional Delaunay Triangulations Using Local Transformations," *Comp. Aided Geom. Design*, vol. 8, pp. 123-142, 1990.

(Joe, 1991)

Joe, B. - "Delaunay Versus Max-Min Solid Angle Triangulations for Three-Dimensional Mesh Generation," *Int. J. Num. Meth. Engng.*, vol. 31, pp. 987-997, 1991.

(Las Casas, 1990)

Las Casas, E.B. - "Um Processo Adaptativo Misto Global para o Método dos Elementos Finitos," *RBE - Revista Brasileira de Engenharia - Caderno de Eng. Estrutural*, vol. 6, no. 2, pp. 41-51, 1990.

(Lau e Lo, 1996)

Lau, T.S. e Lo, S.H. - "Finite Element Mesh Generation over Analytical Curved Surfaces," *Computers & Structures*, vol. 59, pp. 301-309, 1996.

(Lee e Lo, 1992)

Lee, C.K. e Lo, S.H. - "An Automatic Adaptative Refinement Finite Element Procedure for 2D Elastostatic Analysis," *Int. J. Num. Meth. Engng.*, vol.35, pp. 1967-1989, 1992.

(Lee e Lo, 1997)

Lee, C.K. e Lo, S.H. - "Automatic Adaptative Refinement Finite Element Procedure for 3D Stress Analysis," *Finite Elem. Anal. Des.*, vol.25, pp. 135-166, 1997.

(Lewis *et al.*, 1996a)

Lewis, R.W.; Zheng, Y. e Gethin, D.T. - "Three-dimensional Unstructured Mesh Generation: Part 1. Fundamental Aspects of Triangulation and Point Creation," *Comput. Meth. Appl. Mech.*, vol. 134, pp. 249-268, 1996.

(Lewis *et al.*, 1996b)

Lewis, R.W.; Zheng, Y. e Gethin, D.T. - "Three-dimensional Unstructured Mesh Generation: Part 2. Surface Meshes," *Comput. Meth. Appl. Mech.*, vol. 134, pp. 269-284, 1996.

(Lewis *et al.*, 1996c)

Lewis, R.W.; Zheng, Y. e Gethin, D.T. - "Three-dimensional Unstructured Mesh Generation: Part 3. Volume Meshes," *Comput. Meth. Appl. Mech.*, vol. 134, pp. 285-310, 1996.

(Lo, 1989)

Lo, S.H. - "Delaunay Triangulation of Non-Convex Planar Domains," *Int. J. Num. Meth. Engng.*, vol. 28, pp. 2695-2707, 1989.

(Lohner e Parikh, 1988)

Lohner, R. e Parikh, P. - "Generation of Three-Dimensional Unstructured Grids by the Advancing-Front Method," *Int. J. Num. Meth. Fluids*, vol. 8, pp. 1135-1149, 1988.

(Liu e Joe, 1994)

Liu, A. e Joe, B. - "Relationship Between Tetrahedron Shape Measures," *BIT*, vol. 34, pp. 268-287, 1994.

(Martha, 1989)

Martha, L.F - "Topological and Geometrical Modeling Approach to Numerical Discretization and Arbitrary Fracture Simulation in Three Dimensions," Ph.D. Thesis, School of Civil Engineering, Cornell University, 1989.

(Martha *et al.*, 1996)

Martha, L.F; Menezes, I.F.M.; Lages, E.N.; Parente Jr., E. e Pitangueira, R.L.S - "An *OOP* Class Organization for Materially Nonlinear Finite Element Analysis," anais do XVII CILAMCE, Congresso Ibero-Latino Americano sobre Métodos Computacionais para Engenharia, Veneza, Italia, pp. 229-232, 1996.

(Moller e Hansbo, 1995)

Moller, P. e Hansbo, P. - "On Advancing Front Mesh Generation in Three Dimensions," *Int. J. Num. Meth. Engng.*, vol 38, pp. 3551-3569, 1995.

(Mosalam e Paulino, 1997)

Mosalam, K.M. e Paulino, G.H. - "Evolutionary Characteristic Length Method for Smeared Cracking Finite Element Methods", *Finite Elem. Anal. Des.*, vol. 27, pp. 99-108, 1997.

(Parthasarathy *et al.*, 1993)

Parthasarathy, V.N.; Graichen, C.M. e Hathaway, A.F. - "A Comparison of Tetrahedron Quality Measures," *Finite Elem. Anal. Des.*, vol. 15, pp. 255-261, 1993.

(Paulino *et al.*, 1998)

Paulino, G.H.; Menezes, I.F.M.; Cavalcante Neto, J.B. e Martha, L.F. - "A Methodology for Adaptive Finite Element Analysis: Towards an Integrated Computational Environment," Submitted to Computational Mechanics, 1998.

(Peraire *et al.*, 1987)

Peraire J.; Vahdati, M.; Morgan K. e Zienkiewicz, O.C. - "Adaptive Remeshing for Compressive Flow Computations," *Journal of Computational Physics*, vol. 72, pp. 449-466, 1987.

(Peraire *et al.*, 1988)

Peraire, J.; Peiro, J.; Formaggia, L.; Morgan, K. e Zienkiewicz, O.C. - "Finite Euler Computation in Three-Dimensions," *Int. J. Num. Meth. Engng.*, vol 26, pp. 2135-2159, 1988.

(Peraire *et al.*, 1992)

Peraire, J.; Peiro, J.; e Morgan, K. - "Adaptive Remeshing for Three-Dimensional Compressive Flow Computations," *Journal of Computational Physics*, vol. 103, pp. 269-285, 1992.

(Perucchio *et al.*, 1989)

Perucchio, R.; Saxena, M. e Kela, A. - "Automatic Mesh Generation from Solid Models Based on Recursive Spatial Decompositions," *Int. J. Num. Meth. Engng.*, vol. 28, pp. 2469-2501, 1989.

(Potyondy, 1993)

Potyondy, D.O. - "A Software Framework for Simulating Curvilinear Crack Growth in Pressurized Thin Shells," Ph.D. Thesis, School of Civil Engineering, Cornell University, 1993.

(Potyondy *et al.*, 1995a)

Potyondy, D.O.; Wawrzynek, P.A. e Ingraffea, A.R. - "Discrete Crack Growth Analysis Methodology for Through Cracks in Pressurized Fuselage Structure," *Int. J. Num. Meth. Engng.*, vol. 38, pp. 1611-1633, 1995.

(Potyondy *et al.*, 1995b)

Potyondy, D.O.; Wawrzynek, P.A. e Ingraffea, A.R. - "An Algorithm to Generate Quadrilateral or Triangular Element Surface Meshes in Arbitrary Domains with Applications to Crack Propagation," *Int. J. Num. Meth. Engng.*, vol. 38, pp. 2577-2701, 1995.

(Ribeiro, 1991)

Ribeiro, L.F.B. - "Estratégia H-P de Refinamento para o Método de Elementos Finitos, " Tese de Doutorado, UFRJ, 1991.

(O'Rourke, 1994)

O'Rourke, J. - *Computational Geometry in C*, Cambridge University Press, 1994.

(O'Rourke, 1987)

O'Rourke, J. - *Art Gallery Theorems and Algorithms*, Oxford University Press, New York - NY, 1987.

(Samet, 1984)

Samet, H. - "The Quadtree and Related Hierarchical Data Structures," *ACM Computer Surveys*, vol. 16, no. 2, 1984.

(Samet, 1990)

Samet, H. - *Applications of Spatial Data Structures - Computer Graphics, Image Processing and GIS*, Addison-Wesley, 1st. edition, Reading - MA, 1990.

(Shaw e Pitchen, 1978)

Shaw, R.D. e Pitchen, R.G. - "Modifications to the Suhara-Fukuda Method of Network Generation," *Int. J. Num. Meth. Engng.*, vol. 12, pp. 93-99, 1978.

(Schuetze et. al., 1993)

Schuetze, K.T.; Shiakolas, P.S.; Muthukrishnan, S.N.; Nambiar, R.V. e Lawrence, K.L - "A Study of Adaptively Remeshed Finite Element Problems Using Higher Order Tetrahedra," *Computers & Structures*, vol. 54, no. 2, pp. 279-288, 1995.

(Schroeder e Shephard, 1990)

Schoereder, W.J. e Shephard, M.S. - "A combined Octree/Delaunay Method for Fully Automatic 3-D Mesh Generation," *Int. J. Num. Meth. Engng.*, vol 29, pp. 37-55, 1990.

(Shephard, 1986)

Shephard, M.S. - "Adaptive Finite Elements e CAD, Accuracy Estimates and Adaptive Refinements in Finite Element Computations," I. Babuška et al. Edited, pp. 205-225, John Wiley & Sons, 1986.

(Shephard, 1988)

Shephard, M.S. - "Approaches to the Automatic Generation and Control of Finite Element Meshes," Appl. Mech., Rev. 41, pp. 169-185, 1988.

(Shephard e Georges, 1991)

Shephard, M.S. e Georges, M. K. - "Automatic Tri-Dimensional Mesh Generation by the Finite Octree Technique," *Int. J. Num. Meth. Engng.*, vol. 32, no. 4, pp. 709-749, 1991.

(Stroustrup, 1997)

Stroustrup, B. - *The C++ Programming Language*, 3rd. edition, Addison Wesley, 1991.

(Tabbara *et al.*, 1994)

Tabbara, M; Blacker, T. e Belytschko, T. - "Finite Element Derivative Recovery by Moving Least Square Interpolants", *Comp. Meth. Appl. Mech. Engng.*, vol. 117, pp. 211-223, 1994.

(Vianna, 1992)

Vianna, A. C. - "Modelagem Geométrica completa para modelos bidimensionais de elementos finitos ", Dissertação de Mestrado, Departamento de Engenharia Civil, PUC-Rio, 1992.

(Von Herzen e Barr, 1987)

Von Herzen B. e Barr A. H. - "Accurate Triangulations of Deformed, Intersecting Surfaces", *Computer Graphics*, vol. 21, pp. 103-110, 1987.

(Wawrzynek, 1991)

Wawrzynek, P. A. - "Discrete Modeling of Crack Propagation: Theoretical Aspects and Implementation Issues in Two and Three Dimensions, Ph.D. Thesis, School of Civil Engineering, Cornell University, 1991.

(Watson, 1981)

Watson, D.F. - "Computing the $n$-dimensional Delaunay tesselation with application to Voronoi polytopes", *The Computer Journal*, vol. 24, no. 2, pp. 167-172, 1981.

(Weatherill e Hassan, 1994)

Weatherill, N.P. e Hassan, O. - "Efficient Three-dimensional Delaunay Triangulation with Automatic Point Creation and Imposed Boundary Constraints," *Int. J. Num. Meth. Engng.*, vol 37, pp. 2005-2039, 1994.

(Wiberg e Abdulwahab, 1993)

Watson, N-E. e Abdulwahab, F. - "Patch Recovery Based on Superconvergent Derivatives and Equilibrium", *Int. J. Num. Meth. Engng.*, vol. 36, pp. 2703-2724, 1993.

(Wiberg *et al.*, 1994)

Watson, N-E.; Abdulwahab, F. e Ziukas, S. - "Enhnaced Superconvergent Patch Recovery Incorporating Equilibrium and Boundary Conditions", *Int. J. Num. Meth. Engng.*, vol. 37, pp. 3417-3440, 1994.

(Yerry e Shephard, 1984)

Yerry, M.A. e Shephard, M.S. - "Automatic Three-Dimensional Mesh Generation by Modified-Octree Technique," *Int. J. Num. Meth. Engng.*, vol. 20, pp. 1965-1990, 1984.

(Yerry e Shephard, 1986)

Yerry, M.A. e Shephard, M.S. - "Toward Automatic Finite Element Modeling," Finite Elements Anal. Des., vol. 2, pp. 143-160, 1986.

(Zienkiewicz *et al.*, 1982)

Zienkiewicz, O.C.; Kelly, D.W.; Gago, J. e Babuška, I. - "Hierarchical Finite Element Approaches, Error Estimador and Adaptive Refinement," in J. Whiteman (ed.), *Mathematics of Finite Elements and Applications (IV)*, Academic Press, New York - NY, 1982.

(Zienkiewicz e Zhu, 1987)

Zienkiewicz, O.C. e Zhu, J.Z. - "A Simple Error Estimator and Adaptive Procedure for Practical Engineering Analysis," *Int. J. Num. Meth. Engng.*, vol. 24, pp. 337-357, 1987.

(Zienkiewicz e Taylor, 1989)

Zienkiewicz, O.C. e Taylor, R.L. - *The Finite Element Method - Volume 1, Basic Formulation and Linear Problems*, McGraw-Hill, New York - NY, 1989.

(Zienkiewicz *et al.*, 1989)

Zienkiewicz, O.C.; Zhu, J.Z. e Gong, N.G. - "Effective and Pratical h-p-Version Adaptive Procedures for the Finite Element Method," *Int. J. Num. Meth. Engng.*, vol. 28, pp. 879-891, 1989.

(Zienkiewicz e Zhu, 1992a)

Zienkiewicz, O.C. e Zhu, J.Z. - "The Superconvergent Patch Recovery and A Posteriori Error Estimates. Part 1: The Recovery Technique," *Int. J. Num. Meth. Engng.*, vol. 33, pp. 1331-1364, 1992.

(Zienkiewicz e Zhu, 1992b)

Zienkiewicz, O.C. e Zhu, J.Z. - "The Superconvergent Patch Recovery and A Posteriori Error Estimates. Part 2: Error Estimates and Adaptivity," *Int. J. Num. Meth. Engng.*, vol. 33, pp. 1365-1382, 1992.

(Zienkiewicz e Zhu, 1994)

Zienkiewicz, O.C. e Zhu, J.Z. - "The SPR Recovery and Boundaries," *Int. J. Num. Meth. Engng.*, vol. 37, pp. 3195-3196, 1994.

(Zienkiewicz e Zhu, 1997)

Zienkiewicz, O.C. e Zhu, J.Z. - "A posteriori Error Estimation and Three-Dimensional Automatic Mesh Generation,", *Finite Elem. Anal. Des.*, vol. 25, pp. 167-184, 1997.

"Geração de Malha e Estimativa de Erro para Modelos Tridimensionais de Elementos Finitos com Trincas". Tese de Doutorado apresentada por JOAQUIM BENTO CAVALCANTE NETO em 28 de setembro de 1998 ao Departamento de Engenharia Civil da PUC-Rio e aprovada pela Comissão Julgadora, formada pelos seguintes professores:

Prof. Luiz Fernando C.R. Martha (Orientador)
Departamento de Engenharia Civil / PUC-Rio

Prof. Luiz Eloy Vaz
Departamento de Engenharia Civil / PUC-Rio

Prof. Raul Rosas e Silva
Departamento de Engenharia Civil / PUC-Rio

Eng. Túlio N. Bittencourt
EPUSP

Prof. Fernando Luiz B. Ribeiro
COPPE/UFRJ

Prof. Marcelo Gattass
Departamento de Informática / PUC-Rio

Visto e permitida a impressão,

Rio de Janeiro, 29/10/98

Prof. Paulo Batista Gonçalves
Coordenador dos Programas de Pós-Graduação
do Centro Técnico Científico