WILLIAM WAGNER MATOS LIRA

MODELAGEM GEOMÉTRICA PARA ELEMENTOS FINITOS USANDO MULTI-REGIÕES E SUPERFÍCIES PARAMÉTRICAS

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
Pontifícia Universidade Católica do Rio de Janeiro

Rio de Janeiro, Maio de 2002

Agradecimentos

Aos meus pais Ferreira e Ana, às minhas irmãs Déia e Karllinha, à minha avó Amparo, à minha namorada Isabelle e a todos os meus familiares, pelo apoio, paciência e incentivo demonstrados ao longo do desenvolvimento deste trabalho.

Ao professor Luiz Fernando Martha e ao amigo Lula, orientadores deste trabalho, pelo incentivo, ensinamento, orientação e, principalmente, pela amizade e dedicação demonstradas ao longo do desenvolvimento deste trabalho.

A todos os amigos do Tecgraf e da PUC pela amizade consolidada durante os anos de convívio no Rio de Janeiro, e pelo apoio e colaboração no desenvolvimento deste trabalho.

Aos amigos de Maceió e aos companheiros de chopp no Rio pelas conversas, discussões e, principalmente, pela nossa amizade.

A todos os professores do curso de Engenharia Civil da Universidade Federal de Alagoas (UFAL) pela amizade e ensinamentos passados durante o curso de graduação; ao PET de Engenharia Civil da UFAL pelos bons momentos vividos na graduação.

A Ana Roxo e a todos os funcionários e professores do Departamento de Engenharia Civil da PUC.

Ao Tecgraf pelo apoio financeiro e tecnológico durante os cursos de pós-graduação.

A CAPES e ao CNPq pelo apoio financeiro durante os cursos de graduação e pós-graduação.

Resumo

Este trabalho apresenta um ambiente computacional para modelagem geométrica aplicada à análise por elementos finitos usando multi-regiões e superfícies paramétricas representadas por NURBS. O principal objetivo é gerar modelos 3D para serem usados em simulações numéricas baseadas no Método de Elementos Finitos (MEF). Nessa proposta, a metodologia adotada consiste na combinação de alguns aspectos da modelagem geométrica tais como a detecção automática de regiões e a interseção de superfícies com geração de malhas de elementos finitos.

No contexto da programação orientada a objetos, uma nova organização de classes para o modelador geométrico usado neste trabalho, denominado MG, é apresentada. Essa organização de classes permite a implementação do ambiente proposto, mantendo a interface com o usuário tão simples e eficiente quanto à versão original.

A organização de classes também provê suporte para a geração de modelos usados em análise por elementos finitos. Enquanto as malhas de elementos finitos requeridas para simulações numéricas são geradas por algoritmos específicos implementados no modelador MG, os atributos são gerenciados por um sistema, denominado ESAM (*Extensible System Attributes Management*), que também é incorporado ao MG. Esse sistema permite que os atributos do modelador MG sejam configurados para o uso em diversos tipos de problemas de engenharia.

A estrutura de dados usada neste ambiente é representada por um enfoque híbrido baseado na combinação de uma representação CGC (*Complete Geometric Complex*) do modelo e na estrutura de dados estendida do modelador MG. Além disso, a determinação da interseção de superfícies é realizada usando um algoritmo implementado no MG, enquanto que a representação CGC é responsável pelo reconhecimento de multi-regiões. O algoritmo de interseção de superfícies é modificado para tratar casos especiais não considerados na sua versão original.

Abstract

This work presents a computational environment for geometric modeling applied to finite-element analysis using multi-regions and parametric surfaces represented as NURBS. The main goal is to generate 3D models to be used in numerical simulations based on the Finite-Element Method (FEM). For this purpose, the adopted methodology consists of combining some aspects of geometric modeling, such as automatic region detection and surface intersection, with finite-element mesh generation.

In the context of Object-Oriented Programming, a new class organization for the geometric modeler used in this work, called MG (Mesh Generation), is presented. This class organization allows the implementation of the proposed environment, keeping the user interface as simple and efficient as in the original version of the MG modeler.

The proposed class organization also provides support for the generation of models used in finite-element analysis. While the finite-element meshes required for the numerical simulations are generated by specific algorithms implemented in MG, the attributes are managed by a system called ESAM (Extensible System Attributes Management), which also is incorporated into MG. This system allows the customization of simulation attributes in the MG modeler for use in different types of Engineering problems.

The data structure used in this environment is represented by a hybrid approach based on the combination of a CGC (Complete Geometric Complex) representation and the MG's data structure, which has been extended for this purpose. Moreover, the computation of surface intersections is accomplished by using an algorithm implemented in the MG, while the CGC representation is responsible for multi-region recognition. The surface-intersection algorithm has been modified in order to handle special cases that have not been treated in the original version.

Índice

Lista de Figuras

Lista de Símbolos

MEF	Método dos Elementos Finitos	1
MG	Modelador Geométrico denominado *Mesh Generator*	3
CGC	*Complete Geometric Complex*	5
POO	Programação Orientada a Objetos	5
ESAM	*Extensible System Attributes Management*	7
NURBS	*Non Uniform Rational B-Splines*	9
CAGD	*Computer-Aided Geometric Design*	19
CSG	*Constructive Solid Geometry*	45
B-REP	*Boundary-Representation*	45
RED	Estrutura de Dados *Radial-Edge*	47
DCEL	Estrutura de Dados *Doubly Connected Edge List*	69

1. Introdução

Análise de elementos finitos [6,82] e modelagem geométrica de sólidos [28] são itens importantes no processo de simulação de problemas de engenharia (veja, por exemplo, a Figura 1.1), principalmente quando a solução analítica é desconhecida ou de difícil obtenção. Na maioria desses problemas, faz-se necessário utilizar modeladores que possam reproduzir as diversas formas geométricas utilizadas, bem como gerar a malha de elementos finitos correspondente.

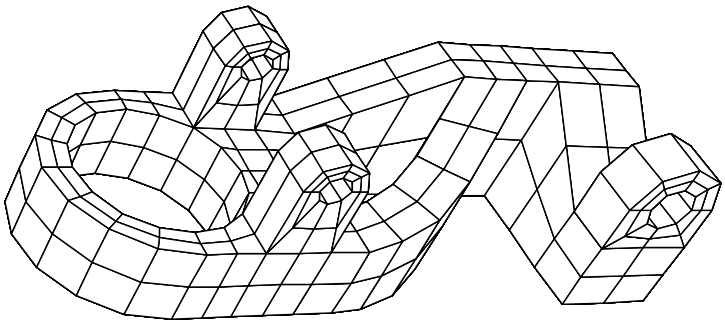


Figura 1.1 – Modelagem geométrica e malha de elementos finitos.

De uma maneira geral, o método dos elementos finitos (MEF) é baseado em um modelo numérico obtido a partir da subdivisão ou "discretização" do domínio do problema, juntamente com informações adicionais associadas a essa discretização, necessárias para a completa definição do problema físico. Tais informações consistem de um conjunto de parâmetros, chamados *atributos* da simulação [9,34]. A discretização, denominada *malha de elementos finitos*, consiste em um conjunto de nós ou vértices (pontos com coordenadas) e um conjunto de células, denominadas elementos finitos, com uma *topologia* pré-definida (triângulos, quadriláteros, ou tetraedros, por exemplo). Os elementos são definidos por uma lista de conectividades de seus vértices (seqüência de vértices que pertencem a cada elemento). Um modelo de elementos finitos é a associação de uma malha de elementos finitos com um conjunto de atributos da simulação.

Um aspecto importante em uma simulação 3D usando o MEF é a geração de malhas. Esta é uma área de pesquisa que está ativa desde a criação do método. Em geral, o processo de geração de malhas demanda tempo e é bastante cansativo, além de exigir uma certa experiência do profissional responsável por esta tarefa. Neste contexto, algoritmos automáticos para geração de malhas têm se tornado bastante úteis para aumentar a confiabilidade dos procedimentos de análise numérica pelo MEF [13].

No MEF, a malha de elementos finitos normalmente é definida sobre a descrição geométrica do domínio do problema que está sendo estudado. Neste sentido, a criação do modelo geométrico torna-se um importante aspecto dentro do contexto de simulações tridimensionais usando o MEF. Existem várias fases envolvidas nessa tarefa, que vão desde as estratégias de interface com o usuário até os esquemas de representação dos dados. Além disso, a geometria e a forma dos objetos reais de engenharia são intrinsecamente complexas, usualmente compostas por vários materiais e regiões. Então, na criação do modelo geométrico é necessário usar programas especiais, chamados modeladores, que podem reproduzir, em formato digital, as formas geométricas dos objetos da simulação [38].

1.1 - Motivações e Trabalhos Relacionados

Baseado nas informações citadas acima, uma das preocupações da linha de pesquisa de Computação Gráfica Aplicada do Departamento de Engenharia Civil da PUC-Rio, na qual este trabalho está inserido, é a busca da automação real no processo de simulação de problemas tridimensionais pelo MEF. Neste contexto, as questões referentes à modelagem geométrica e à geração de malhas de elementos finitos assumem uma importância fundamental e vêm sendo estudadas durante os últimos anos.

Qualquer metodologia de modelagem deve tratar dois importantes aspectos na simulação 3D de problemas de engenharia usando o MEF:

- modelagem geométrica com detecção automática de múltiplas regiões fechadas e interseção de superfícies, e

- suporte para geração automática de malhas de elementos finitos.

Essa metodologia pode ser implementada usando-se bibliotecas para modelagem geométrica encontradas na literatura, tais quais ACIS [1], Parasolids [51] e Pro/ENGINEER API Toolkit [57]. Essas bibliotecas provêm representações geométricas e topológicas, bem como funções de interface entre o programa e a aplicação, que são necessárias nesse tipo de modelagem. Entretanto, os aspectos relacionados à modelagem e à geração de malhas são tratados separadamente nessas bibliotecas, não existindo uma comunicação entre eles. Essa comunicação é um aspecto-chave no uso dessas bibliotecas. Por outro lado, programas comerciais como o PATRAN [37] e o ANSYS [50] consideram esse aspecto-chave, tratando o problema de interseção entre superfícies e fazendo a reconstrução das malhas associadas simultaneamente. No entanto, apenas casos de interseções onde as superfícies possuem formas bem definidas (superfícies implícitas) são considerados (por exemplo, superfícies cilíndricas, esféricas, etc.). Não existe uma metodologia para o cálculo de interseções de superfícies com geometrias quaisquer.

Para atender o aspecto acima citado, esse tipo de metodologia foi implementado em um modelador de elementos finitos existente, chamado MG [83]. O MG é um pré-processador que foi originalmente desenvolvido para geração de superfícies (cascas) em modelos de elementos finitos [19], sendo, na seqüência, estendido para também considerar malhas sólidas [14,44]. O MG incorpora duas capacidades importantes em modelagem 3D pelo MEF. A primeira está relacionada com a interface com o usuário e procedimentos gráficos iterativos para gerar malhas em superfícies [17], e a segunda relaciona-se com a interseção de superfícies paramétricas, tal qual mostrada na Figura 1.2.

O algoritmo para interseção de superfícies [19,20] usado no MG é baseado em um esquema para interseção de superfícies paramétricas onde as malhas de superfícies existentes são usadas como suporte para a definição das curvas de interseção. Uma estrutura de dados auxiliar, definida no espaço paramétrico de cada superfície, permite a construção das curvas de *trimming* (curvas resultantes da interseção entre superfícies e que possuem representação nos espaços paramétricos destas e no espaço Cartesiano) sem a necessidade de se realizar buscas globais.

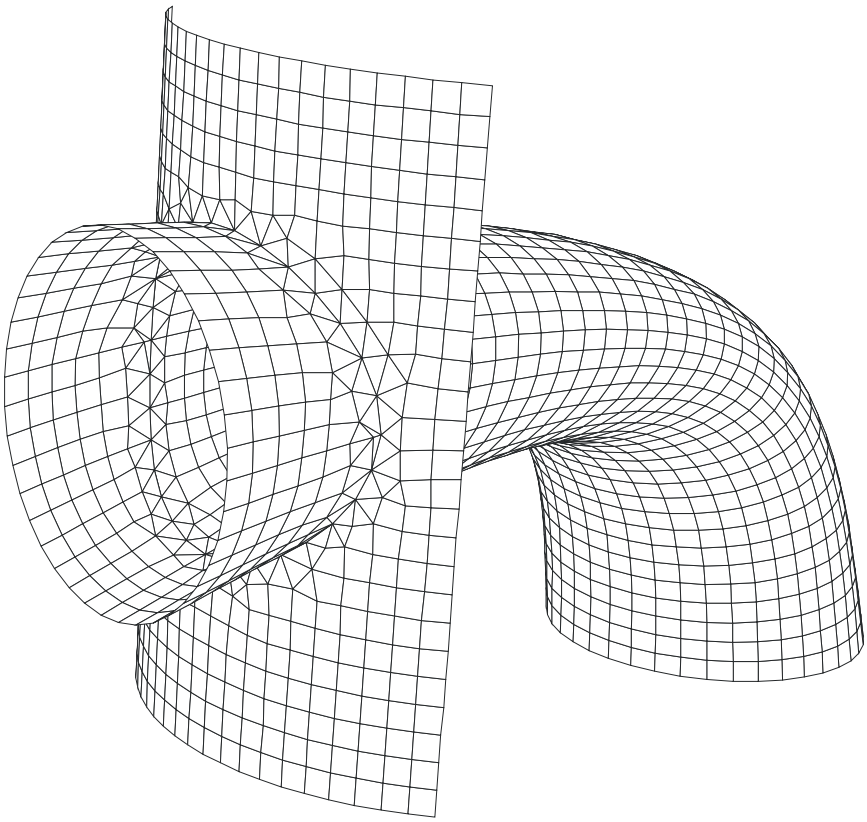


Figura 1.2 – Interseção entre malhas de superfícies.

A versão original do modelador MG é útil na representação de modelos geométricos, e tem uma interface relativamente simples e eficiente. No entanto, sua estrutura de dados não foi baseada em qualquer conceito formal de modelagem geométrica. Em muitas situações, a consistência geométrica de um modelo é realizada com a intervenção do usuário. Por exemplo, não existe a capacidade de detectar automaticamente quando um volume no espaço é definido por um conjunto de retalhos de superfícies (ou seja, superfícies limitadas pelas suas curvas de bordo). O usuário deve indicar a construção explicitamente, o que pode não ser uma tarefa trivial na modelagem de um problema real de engenharia (veja, por exemplo, a Figura 1.3). Essa capacidade é particularmente importante na geração de malhas formadas por elementos finitos sólidos, onde algumas vezes é desejável ter várias regiões com mapeamentos distintos, cada uma definida com suas respectivas malhas de contorno geradas por diferentes tipos de algoritmos.

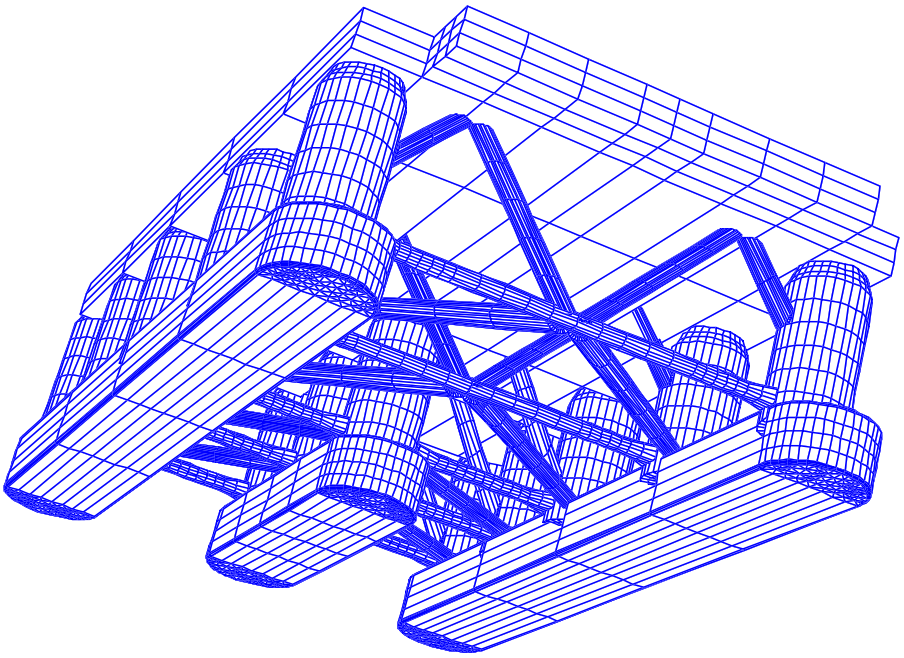


Figura 1.3 – Problema real de engenharia – detecção de regiões.

O formalismo necessário para atender as capacidades de modelagem descritas acima, que permite o reconhecimento automático de regiões sólidas criadas, é um dos objetivos principais deste trabalho. Existem diversos trabalhos publicados na literatura que apresentam esquemas que tratam tais capacidades [16,22,31,32,60,61,74,75]. Neste trabalho adota-se uma metodologia que é baseada em uma representação topológica completa de uma subdivisão espacial, chamada *Complete Geometric Complex* (CGC) [16]. Dentre outros motivos, esta metodologia foi escolhida pois apresenta uma implementação computacional eficiente e robusta, já tendo sido testada e usada em outros sistemas de grande porte (por exemplo, o sistema MultiMesh [41]).

A representação CGC é implementada como uma biblioteca de classes, no contexto da programação orientada a objetos (POO), provendo um conjunto de operadores de alto nível que manipulam uma subdivisão espacial. Esses operadores recebem como parâmetros de entrada as informações geométricas dos retalhos de superfícies que são, então, inseridas na subdivisão espacial. Estas informações geométricas são automaticamente transformadas em informações topológicas requeridas pelos operadores de baixo nível. A implementação original da CGC [16] trata apenas de retalhos de superfícies planas.

O modelador MG oferece ainda suporte à geração de malhas de elementos finitos tanto em superfícies quando em sólidos. Nas simulações por elementos finitos modeladas com o MG, procedimentos para geração de malhas não-estruturadas de elementos finitos são particularmente importantes. Diversos trabalhos vêm sendo desenvolvidos nessa área utilizando várias técnicas. Existem algoritmos baseados em triangulação de Delaunay [72], em decomposição espacial recursiva [71] e em métodos de avanço de fronteira [36,45,52].

Uma das possibilidades para a geração de malhas em superfícies implementada no MG utiliza um algoritmo, apresentado por Miranda [44], que combina técnicas de avanço de fronteira com decomposição espacial recursiva. Esse algoritmo é utilizado para geração de malhas triangulares em superfícies com geometria e topologia arbitrárias baseando-se na descrição paramétrica dessas superfícies. A malha de elementos finitos é gerada no espaço paramétrico, usando-se técnicas bidimensionais, e é feito o seu mapeamento para o espaço 3D. Nesse mapeamento, uma estrutura quadtree armazena parâmetros que permitem a correção de distorções métricas entre o espaço Riemaniano e o paramétrico. O algoritmo faz ainda um refinamento baseado na curvatura da superfície, gerando malhas mais discretizadas em regiões com curvaturas mais acentuadas.

Na geração de malhas tridimensionais não-estruturadas de elementos finitos, o modelador MG utiliza um algoritmo, apresentado por Cavalcante Neto [13], que também combina técnicas de avanço de fronteira com decomposição espacial recursiva. Esse algoritmo implementa uma técnica para geração de malhas volumétricas de elementos tetraédricos, para domínios tridimensionais arbitrários.

Algoritmos para geração de malhas estruturadas [43,44] também estão implementados neste modelador, tornando-o capaz de gerar malhas compatíveis com os mais diversos tipos de elementos finitos conhecidos na literatura [40,82].

É importante notar, ainda, que a metodologia adotada na discretização da geometria (domínio) é comum para os diversos tipos de problemas de engenharia. Os atributos, no entanto, são específicos para o tipo de simulação desejada. Por exemplo, em uma análise de tensões de um problema estrutural, alguns atributos se referem às cargas

distribuídas, às condições de apoio e aos deslocamentos prescritos. Por outro lado, em uma análise térmica os atributos que particularizam o problema são, dentre outros, o fluxo de calor e o gradiente térmico.

Neste sentido, é desejável que um modelador seja capaz de especificar atributos para os mais diversos tipos de problemas de engenharia, ou seja, que os atributos possam ser configuráveis com relação ao tipo de análise desejada. Na literatura, existem alguns trabalhos [64,76] que tratam da questão de configuração dos atributos de uma simulação. Em particular, pode-se citar o sistema ESAM (*Extensible System Attributes Management*) [9,34], que é responsável pelo gerenciamento e configuração dos atributos de uma simulação. Neste trabalho, o sistema ESAM foi incorporado ao MG, possibilitando a utilização desse modelador nos mais diversos problemas de engenharia usando elementos finitos.

1.2 – Objetivos e Principais Contribuições

Baseado nas informações acima, este trabalho tem como objetivo principal propor um ambiente para a realização de modelagens geométricas para elementos finitos usando superfícies paramétricas como primitivas básicas, provendo um algoritmo bastante robusto para reconstrução das interseções entre elas, e identificando automaticamente as regiões fechadas formadas ao longo do processo de modelagem. O ambiente também oferece suporte à geração de malhas de elementos finitos e à configuração de atributos para diversos tipos de simulações em engenharia.

O ponto central desse trabalho é a implementação no MG das propostas do ambiente de modelagem mencionadas acima. Para isso, é descrita uma organização de classes, no contexto da programação orientada a objetos (POO), de uma nova versão do modelador MG que, mantendo a simplicidade e eficiência das características de interface com o usuário, implementa o ambiente de modelagem que está sendo proposto nesse trabalho. Para realizar essa tarefa, um enfoque híbrido para a representação de dados é adotado. Neste enfoque, a representação CGC do modelo é combinada com uma extensão da estrutura de dados original do MG.

A representação CGC não é mantida durante todo o processo de modelagem. Essa representação do modelo CGC só é criada quando requisitada pelo usuário do MG. Modelagem geométrica tridimensional é uma tarefa complexa que pode envolver uma série de etapas intermediárias, e dependendo da complexidade do modelo, pode não ser útil manter a consistência entre geometria e topologia após a realização de cada passo do processo de modelagem. Por estes motivos, surge a importância da utilização de uma estrutura temporária (ou auxiliar) para realizar tal consistência.

A interseção de superfícies é realizada usando o algoritmo proposto por Coelho [19,20] e implementado no MG, enquanto que a representação CGC é responsável exatamente pela detecção de multi-regiões. Neste trabalho, o algoritmo para interseção de superfícies foi atualizado para considerar casos especiais não tratados na sua versão original.

A implementação original da CGC trata apenas de retalhos de superfícies planas. Para atender ao contexto de modelagem deste trabalho, que é bem mais geral, retalhos de superfícies curvas são importantes e também devem ser considerados. Neste sentido, realizou-se uma extensão da representação CGC capacitando-a a suportar superfícies curvas, tornando-a mais útil na manipulação de modelos manufaturados, como cascos de navios, peças mecânicas, etc..

Com o objetivo de aumentar o poder de expressão do MG, incorporou-se uma biblioteca de funções [30] que implementa representações geométricas do tipo NURBS (*Non Uniform Rational B-Splines*) [53,54]. Essa incorporação viabilizou a construção de um conjunto mais amplo de superfícies (incluindo quádricas e cônicas), além de padronizar as funcionalidades necessárias a descrição do espaço paramétrico das superfícies.

Para completar as implementações do ambiente de modelagem proposto, foi incorporado ao MG o gerenciador de atributos ESAM [9,34], permitindo não só a descrição da geometria e geração da malha de elementos finitos, mas também a definição de atributos específicos para os mais diversos tipos de simulações.

As principais contribuições desse trabalho podem ser resumidas assim:

- Desenvolvimento de um esquema de dados híbrido capaz de oferecer suporte à modelagem geométrica de superfícies e de sólidos, provendo capacidade para realizar o reconhecimento automático de regiões e determinar interseções de superfícies.
- Tratamento de casos patológicos do algoritmo de interseção de superfícies apresentado por Coelho [19,20], permitindo a sua utilização, de forma confiável e robusta, em uma grande quantidade de problemas.
- Incorporação ao modelador MG de uma biblioteca de funções com representações de curvas e superfícies do tipo NURBS, possibilitando a implementação de uma grande variedade de superfícies, bem como o uso genérico do espaço paramétrico associado a essas superfícies. O uso do espaço paramétrico tem alguns aspectos relevantes, dentre os quais pode-se destacar o suporte à geração de malhas, usando-se algoritmos bidimensionais, e a sua utilização no algoritmo de interseção de superfícies do modelador MG.
- Inclusão no modelador MG do sistema configurável para gerenciamento de atributos ESAM, permitindo o uso do MG na simulação de problemas de engenharia nas mais diversas áreas (estrutural, geotécnica, naval, etc.).
- Extensão da biblioteca CGC, que gerencia a modelagem de subdivisões espaciais em 3D, incorporando curvas e superfícies paramétricas do tipo NURBS.
- A integração dos itens descritos acima com algoritmos para geração de malhas, estruturadas e não-estruturadas, de elementos finitos (de superfícies e sólidos), em uma nova versão do modelador MG, capacitando-o a realizar modelagens geométricas para elementos finitos usando multi-regiões e superfícies paramétricas.

1.3 – Organização da Tese

Este trabalho divide-se em 6 capítulos. Este capítulo mostrou uma visão geral da tese, bem como descreveu a sua organização.

O Capítulo 2 descreve detalhes referentes à metodologia de modelagem adotada neste trabalho. Esse capítulo mostra, ainda, como essa metodologia está implementada no modelador MG.

O Capítulo 3 descreve a estrutura de dados híbrida desenvolvida neste trabalho e mostra as adaptações necessárias à sua implementação no modelador MG.

O Capítulo 4 descreve a versão original do algoritmo de interseção de superfícies paramétricas utilizado no MG, bem como a sua nova implementação que considera casos patológicos (especiais) não-tratados na versão original.

O Capítulo 5 ilustra o ambiente apresentado neste trabalho com diversos exemplos, detalhando-os em cada etapa do processo de simulação (modelagem geométrica, definição dos atributos e geração da malha de elementos finitos). O capítulo mostra ainda exemplos numéricos mostrando a validade do ambiente proposto.

Finalmente, no Capítulo 6 são feitas as considerações finais deste trabalho e são apresentadas sugestões para futuros trabalhos.

2. Modelagem Geométrica para Elementos Finitos

Em muitos problemas de engenharia a solução analítica é desconhecida ou de difícil obtenção. Nesses problemas, simulações numéricas usando o método dos elementos finitos (MEF) [40,82] têm sido realizadas com o objetivo de obter resultados aproximados para o problema. O uso do MEF se deve, principalmente, à sua grande versatilidade, à qualidade dos resultados e à relativa facilidade na implementação computacional. Umas das etapas importantes e essenciais para o uso desse método refere-se à definição da malha de elementos finitos utilizada. Essa malha pode ser definida como a subdivisão ou "discretização" do domínio (geometria) do problema. A definição deste domínio, no entanto, não é um processo trivial. Diversos tipos de problemas em engenharia são de grande porte (por exemplo, sistemas flutuantes, turbinas, compressores, etc.), e dependem de um modelador capaz de reproduzir as diferentes formas geométricas do problema e definir a malha de elementos finitos associadas a essas formas com precisão para as simulações numéricas. Nesse contexto, modelagem de superfícies (ou cascas) e sólidos tornam-se itens importantes e devem ser considerados em um processo de simulação computacional usando o MEF.

Uma forma de se construir modelos complexos consiste em combinar várias superfícies ou sólidos construídos individualmente em uma única representação. No caso particular das superfícies, isso pode ser feito recortando as regiões excedentes determinadas pelas interseções, como pode ser visto na Figura 2.1. Para que esse tipo de modelagem seja produtivo, o problema de interseção entre retalhos de superfícies deve ser tratado de forma eficiente e confiável [19,20].

A determinação de multi-regiões (sólidos) também é importante no contexto do processo de modelagem ao qual este trabalho está inserido. Existem aspectos que são decisivos nas diversas formas de abordagem dos problemas de modelagem de sólidos.

Um desses aspectos refere-se à interface gráfica com o usuário [17]. Muitas vezes, a tarefa de determinar, por exemplo, as superfícies do contorno que delimitam uma região ou as suas superfícies geradoras não é trivial (ver, por exemplo, a Figura 2.2). Existem situações onde uma ou mais superfícies localizam-se em posições geométricas de difícil visualização para o usuário, dificultando a tarefa de indicá-las como superfícies formadoras da região desejada. Outro problema é a necessidade de determinar multi-regiões, trabalho que pode ser bastante laborioso e passível de erros, como também pode ser visto na Figura 2.2. Situações como essas determinam a necessidade de se fazer um reconhecimento automático das multi-regiões existentes no problema. Esse é um dos tópicos importantes deste trabalho e será tratado mais detalhadamente no decorrer deste e do próximo capítulo.

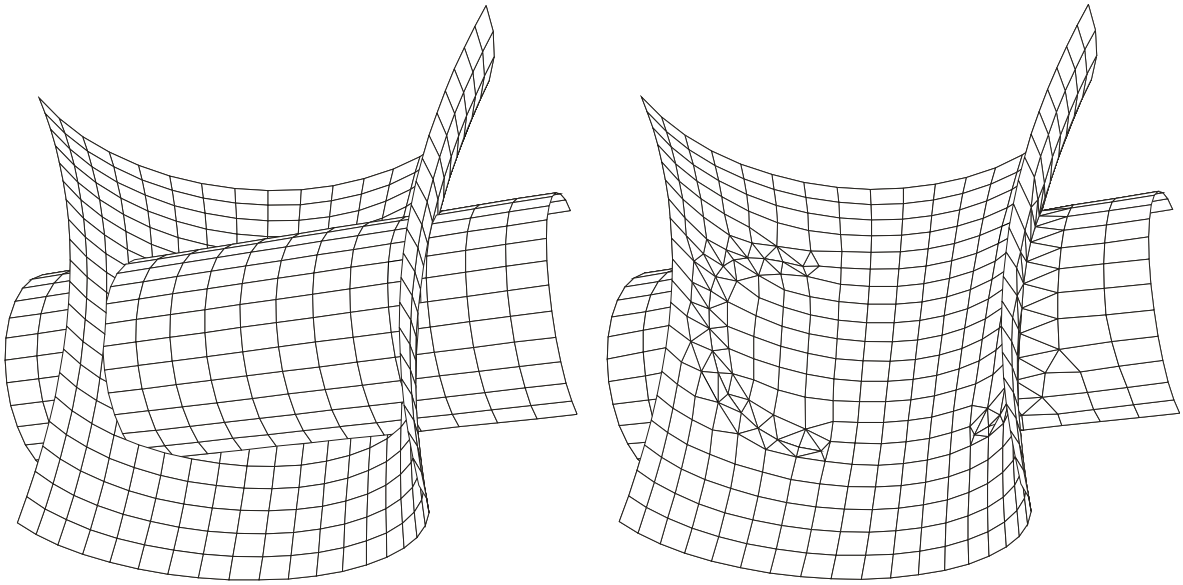


Figura 2.1 – Recortes de regiões excedentes em interseções de superfícies.

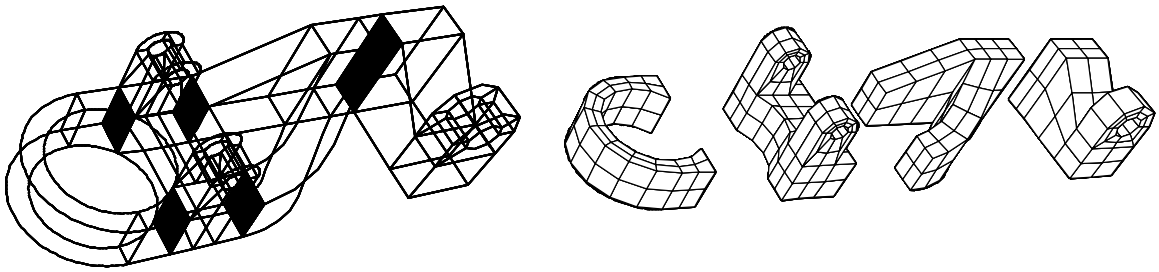


Figura 2.2 – Dificuldade para detecção de multi-regiões.

Deve-se observar que, tanto na modelagem de superfícies quanto na modelagem de sólidos, uma metodologia deve ser usada para descrever o modelo completo com uma representação única da malha de elementos finitos associada a todos os objetos.

Este capítulo expõe, de uma forma geral, as etapas de modelagem geométrica de superfícies e sólidos utilizadas neste trabalho. Ele trata não só das questões relacionadas ao processo de modelagem propriamente dito, mas também da geração de malhas de elementos finitos e da definição dos atributos associados a essas malhas.

2.1 - Representação Paramétrica de Superfícies

Os dois métodos mais comuns para representar superfícies em modelagem geométrica são as formas implícitas e as equações paramétricas [28]. A forma implícita de uma superfície é dada pela equação $f(x,y,z) = 0$, onde x, y e z formam o sistema de eixos no espaço Euclidiano. Uma representação paramétrica de uma superfície é dada por $S(u,v) = (x(u,v), y(u,v), z(u,v))$, onde u e v formam o sistema de eixos no espaço paramétrico da superfície. A Figura 2.3 mostra o exemplo de uma superfície definida no espaço Euclidiano e o seu espaço paramétrico.

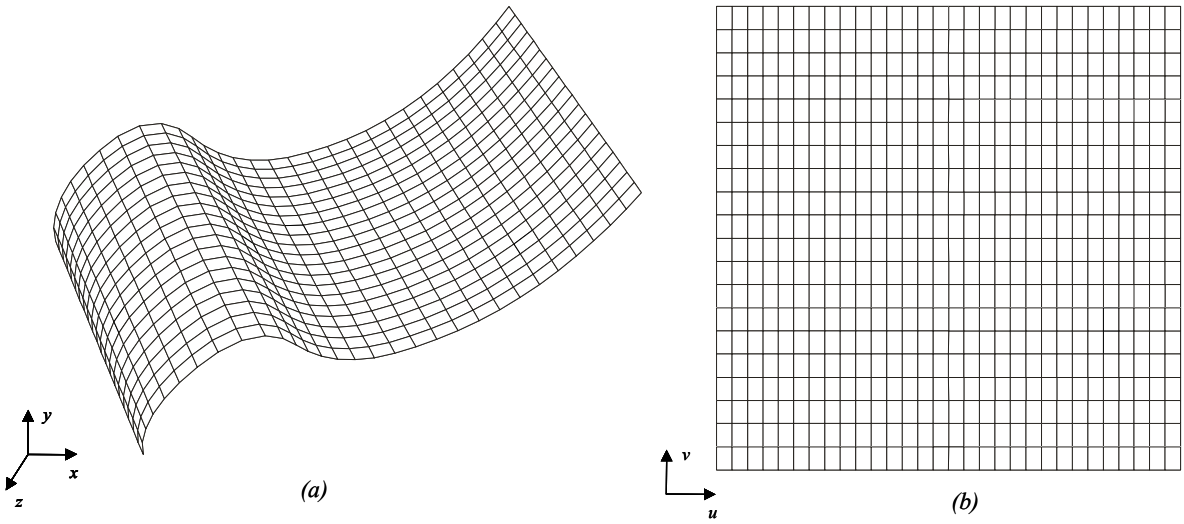


Figura 2.3 – Representações de uma superfície: a) espaço Euclidiano; b) espaço paramétrico.

Definir qual das representações é melhor em uma modelagem geométrica não é uma tarefa fácil. Ambas têm suas vantagens e desvantagens. No escopo deste trabalho, a forma paramétrica é usada porque é possível tirar proveito de algumas de suas potencialidades, além de ser a mais adequada na representação de objetos formados por seções transversais. Dois critérios são considerados. O primeiro refere-se ao algoritmo utilizado no MG para a geração de malhas não-estruturadas em superfícies. Esse algoritmo, proposto por Miranda [44], é aplicado na geração de malhas triangulares em superfícies com geometria arbitrária utilizando a sua descrição paramétrica. Essa descrição é usada por ser mais comum, pois utiliza os algoritmos de triangulação bidimensional, com correções de distorções métricas da geometria da superfície. Além disso, a geração de malhas diretamente na superfície 3D requer maiores critérios para a validação dos elementos triangulares, tornando o algoritmo mais lento. Assim, a superfície 3D é mapeada para uma superfície 2D (ou seja, a sua representação paramétrica), e então é realizada a triangulação bidimensional, corrigindo-se as distâncias e os ângulos distorcidos na transformação da malha do espaço 3D para o espaço paramétrico. Na seqüência, a malha resultante desta triangulação é reconduzida ao espaço 3D. Esse processo pode ser visualizado na Figura 2.4.

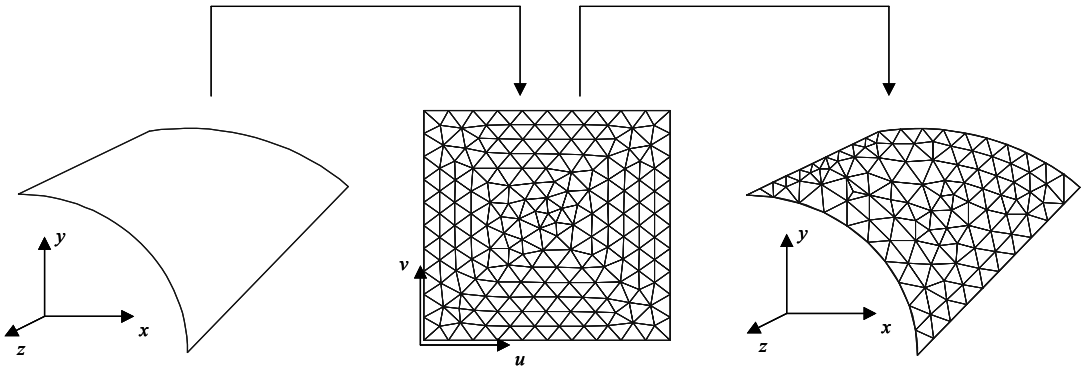


Figura 2.4 – Geração de malhas em superfícies.

Um segundo critério usado na escolha da representação paramétrica está relacionado com o algoritmo para interseção de superfícies usado neste trabalho. Esse algoritmo é baseado na interseção entre malhas sobre as superfícies paramétricas. As buscas necessárias ao cálculo das curvas de interseção e à reconstrução das malhas são apoiadas por uma estrutura de dados topológica cujas principais características são o uso

do espaço paramétrico das superfícies para a orientação das entidades e o uso de árvores *B-trees* [21] e *R*-trees* [7] para o armazenamento destas entidades.

2.2 - Modelagem Geométrica Usando NURBS

Este trabalho incorpora ao seu escopo as representações de superfícies conhecidas como NURBS (*Non Uniform Rational B-Splines*) [53,54]. Algumas das vantagens oferecidas por esse tipo de representação são:

- NURBS provê uma base matemática única para representação das formas analíticas, tais como seções cônicas e superfícies quádricas, bem como superfícies com formas quaisquer (por exemplo, cascos de navios ou carenagem de carros);
- a modelagem usando NURBS é intuitiva; quase todas as ferramentas e algoritmos geométricos têm interpretações de fácil compreensão;
- as superfícies NURBS são invariantes quando submetidas a transformações geométricas afins (translação, rotação, projeções, etc.);
- as superfícies NURBS são generalizações de superfícies *B-Splines* e *Béziers*.

As seções seguintes descrevem resumidamente as principais propriedades das NURBS, bem como a sua representação matemática. Uma visão mais detalhada sobre NURBS pode ser encontrada no livro *NURBS Book* [54].

2.2.1 - Definição e Propriedades

Uma curva NURBS de grau p é uma curva polinomial por partes definida pela expressão

$$C^w(u) = \sum_{i=0}^{n} N_{i,p}(u) P_i^w \ , \ 0 \le u \le 1 \qquad (2.1)$$

onde, P_i^w são os pontos de controle da curva em coordenadas homogêneas e $N_{i,p}(u)$ são as funções base *B-Splines*. Essas funções são definidas sobre o vetor[1]

$$U = \left\{ \underbrace{0,...,0}_{p+1}, u_{p+1},...,u_{r-p-1}, \underbrace{1,...,1}_{p+1} \right\},$$

onde $r = n + p + 1$.

As funções $N_{i,p}(u)$ são determinadas pelas seguintes expressões:

$$N_{i,1}(u) = \begin{cases} 1, \text{se } u_i \le u \le u_{i+1} \\ 0, \text{caso contrário} \end{cases}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p-1} - u_i} N_{i,p-1}(u) + \frac{u_{i+p} - u}{u_{i+p} - u_{i+1}} N_{i+1,p-1}(u) \qquad (2.2)$$

Cada ponto de controle da curva possui pesos, definidos por w_i, associados a ele. Assim, P_i^w é dado por $P_i^w = (w_i x_i, w_i y_i, w_i z_i, w_i)$.

Superfícies NURBS de grau *p* na direção *u* e grau *q* na direção *v* são funções polinomiais racionais por partes. Uma superfície NURBS pode ser tratada como uma extensão da definição dada acima para curvas NURBS. Assim, a descrição de uma superfície NURBS de graus (*p,q*) é dada por:

$$S^w(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(u) N_{j,q}(v) P_{i,j}^w, \ 0 \le u,v \le 1 \qquad (2.3)$$

onde, $P_{i,j}^w$ formam uma rede de pontos de controle da superfície em coordenadas homogêneas, e $N_{i,p}(u)$ e $N_{j,p}(v)$ são as funções base *B-Splines* definidas sobre os vetores de *knots*

[1] Esses vetores são conhecidos na literatura como *knots* e serão assim referenciados nesse trabalho.

$$U = \left\{ \underbrace{0,...,0}_{p+1}, u_{p+1},...,u_{r-p-1}, \underbrace{1,...,1}_{p+1} \right\}$$

$$V = \left\{ \underbrace{0,...,0}_{q+1}, v_{q+1},...,v_{s-q-1}, \underbrace{1,...,1}_{q+1} \right\},$$

onde $r = n + p + 1$ e $s = m + q + 1$.

A Figura 2.5 [54] mostra a rede com os pontos de controle de uma superfície NURBS, enquanto que a Figura 2.6 [54] mostra a superfície NURBS associada a essa rede.

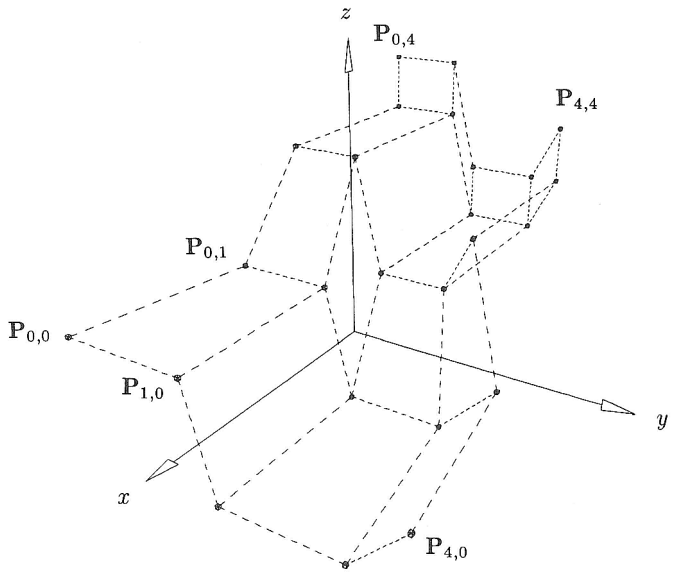


Figura 2.5 – Exemplo de uma rede com os pontos de controle de uma superfície NURBS.

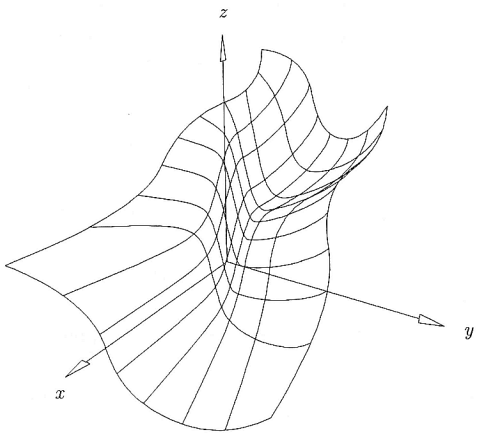


Figura 2.6 – Exemplo de superfície NURBS.

Algumas propriedades geométricas importantes associadas às superfícies NURBS são:

- Pontos de interpolação extremos: $S(0,0) = P_{0,0}$, $S(1,0) = P_{n,0}$, $S(0,1) = P_{0,m}$ e $S(1,1) = P_{n,m}$;
- Invariância afim: uma transformação afim é aplicada à superfície aplicando essa transformação diretamente nos seus pontos de controle;
- Modificação local: se $P_{i,j}$ é movido ou $w_{i,j}$ modificado, então a forma da superfície só é modificada no intervalo $[u_i, u_{i+p+1}) \times [v_j, v_{j+q+1})$;
- Superfícies *B-Spline* e *Bézier* são casos especiais de superfícies NURBS;
- Diferenciabilidade: $S(u,v)$ é $p-k$ vezes diferenciável com relação a *u*, e $q-k$ vezes diferenciável com relação a *v*, onde k é a multiplicidade dos vetores *knots*.

2.2.2 – Biblioteca Computacional NURBS++

Atualmente, existem diversas bibliotecas computacionais que implementam as representações de curvas e superfícies do tipo NURBS, disponibilizando um conjunto de funções para a sua manipulação. Algumas dessas bibliotecas encontram-se disponíveis na literatura. Como exemplo, pode-se citar as bibliotecas Nlib [65], Machester Library [68], DT_NURBS [23] GS_LIB [27] e NURBS++ [30]. A maioria dessas bibliotecas é utilizada em empresas ou em programas comerciais de modelagem, em engenharia ou CAGD (*Computer-Aided Geometric Design*), como ferramenta no

seu processo de modelagem. Exemplos dessas empresas e/ou programas que utilizam tais bibliotecas são Boing Company [8] e Rhinoceros [59].

Esses programas, em sua maioria, são bastante robustos, pois são utilizados no dia-a-dia da modelagem. Isso, de certa forma, é uma maneira de validar o uso de tais bibliotecas, garantindo confiabilidade aos seus usuários. Neste trabalho, optou-se por utilizar a biblioteca NURBS++. Essa biblioteca é implementada em C++ [66] usando o conceito de programação orientada a objetos. Essa é uma de suas vantagens, pois permite a expansão da estrutura de classes, aumentando a capacidade de utilização das curvas e superfícies NURBS. Além disso, o código-fonte também está disponível, o que torna possível a implementação de novas funcionalidades na biblioteca. Essas características possibilitaram a implementação, por exemplo, de superfícies de *Coons* na biblioteca NURBS++. Esse tipo de superfície, muito importante no contexto de modelagem utilizada neste trabalho, não está implementado na versão original da biblioteca.

A biblioteca NURBS++ possui duas classes básicas. A primeira classe contém informações necessárias para a definição de uma curva NURBS. Um objeto dessa classe armazena o grau de interpolação da curva, o vetor de *knots* e os seus pontos de controle. Esses dados são suficientes para definir uma curva NURBS. Para se criar um objeto dessa classe, pode-se usar o seu construtor[2] padrão, fornecendo exatamente as informações descritas acima. No entanto, em muitas situações, tais informações não são fáceis de serem obtidas. A classe, então, possui construtores que criam automaticamente diversos tipos de curvas conhecidas, tais como, retas, arcos e *splines*. Esses construtores trabalham com informações mais intuitivas em modelagem e, internamente, calculam a representação NURBS associada à curva desejada. Por exemplo, um construtor responsável pela criação do objeto NURBS do tipo arco recebe informações do centro desse arco, dos seus pontos inicial e final e do plano que o contém, como pode ser visto na Figura 2.7. A Figura 2.8 resume a descrição da classe curva da biblioteca NURBS++.

[2] Construtores, no contexto de POO, são métodos de uma classe responsáveis pela criação dos objetos.

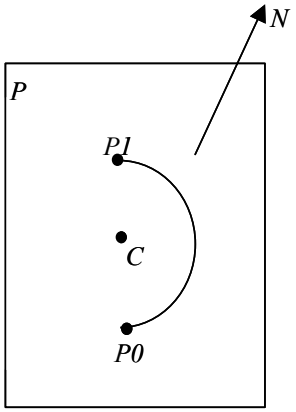


Figura 2.7 – Representação de um arco usando NURBS: *C* é o centro do arco, *P0* e *P1* são os pontos inicial e final, e *P* é o plano que contém o arco e definido pela normal *N*.

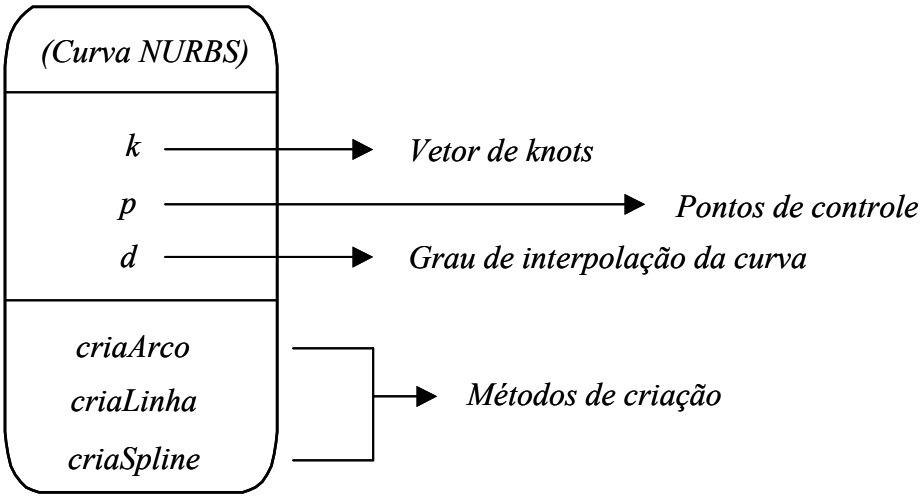


Figura 2.8 – Representação de curvas na biblioteca NURBS++.

A segunda classe básica da biblioteca NURBS++ é a classe para representação de superfícies. Um objeto dessa classe armazena os graus de interpolação da superfície nas direções paramétricas *u* e *v*, o vetor de *knots* em ambas direções e os seus pontos de controle, armazenados de forma matricial. Esses dados são suficientes para definir uma superfície NURBS. Para criar um objeto dessa classe, pode-se também usar o seu construtor padrão, fornecendo exatamente as informações citadas acima, ou usar os construtores que criam automaticamente diversos tipos de superfícies conhecidas, tais como, *sweep*, *Gordon* e revolução. A Figura 2.9 resume a descrição da classe superfície da biblioteca NURBS++.

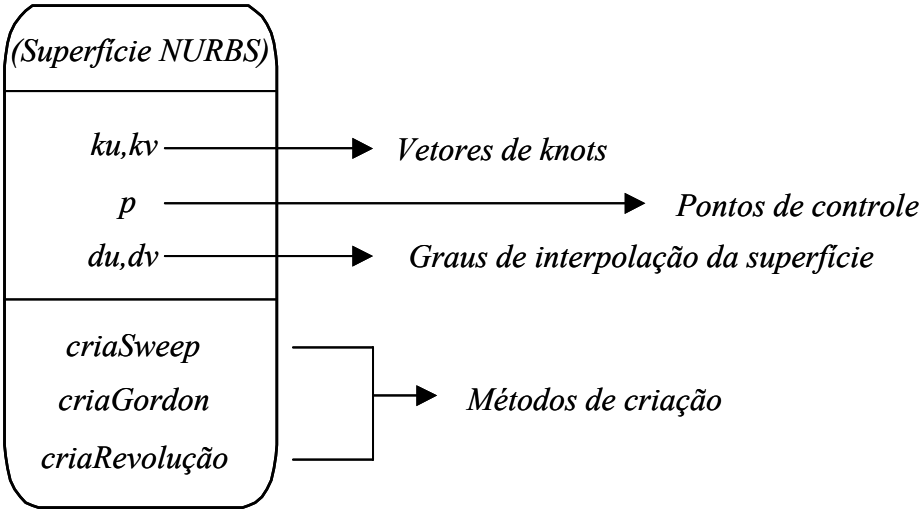


Figura 2.9 – Representação de superfícies na biblioteca NURBS++.

2.2.3 – Incorporação da Biblioteca NURBS++ no Modelador MG

O uso de NURBS no MG foi motivado, principalmente, pela sua capacidade de representar o espaço paramétrico das superfícies. Na versão original do MG, nem todos os tipos de superfícies implementados possuíam a descrição do seu espaço paramétrico.

No processo de incorporação da biblioteca NURBS++ na nova versão do MG, uma hierarquia de classes foi criada para representar os tipos de superfícies utilizados pelo modelador, como pode ser visto na Figura 2.10. Essa hierarquia possui, basicamente, três níveis. O primeiro nível (ou nível superior) é representado pela classe *mgSurface*, responsável pela definição de métodos virtuais puros que devem ser redefinidos pelas suas classes filhas.

A classe *mgSurfaceNURBS*, localizada no nível intermediário da hierarquia, representa uma interface entre o modelador MG e a biblioteca NURBS++. Essa classe possui uma referência para o objeto NURBS correspondente, como pode ser visto na Figura 2.11, pelo qual é possível acessar os dados da superfície armazenados na classe correspondente da biblioteca NURBS++. Todos os dados comuns aos tipos de superfícies definidos no MG são acessados através de métodos definidos nessa classe.

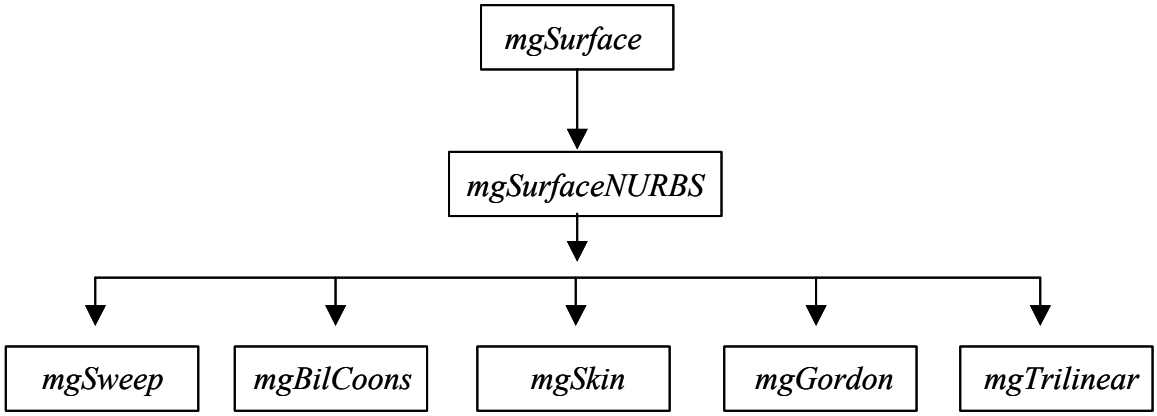


Figura 2.10 – Organização das classes de superfícies no modelador MG.

As classes *mgSweep*, *mgBilCoons*, *mgSkin, mgTrilinear* e *mgGordon*, representam os tipos de superfícies implementados nesta versão do MG. Essas são as classes, relacionadas às superfícies, que podem ser instanciadas pelo usuário através da interface gráfica do modelador MG (não é possível criar um objeto diretamente da classe *mgSurfaceNURBS*). Algumas informações podem ser especificas para cada tipo de superfície. Assim, essas classes também implementam métodos para acessar as informações contidas na biblioteca NURBS++. A formulação referente a cada um desses tipos de superfícies é descrita detalhadamente na seção seguinte.

O fluxo de comunicação entre o MG e a biblioteca NURBS++ é unidirecional. Ou seja, o MG tem acesso às informações contidas na biblioteca NURBS++, enquanto que essa biblioteca não possui acesso aos dados do MG, conforme é mostrado na Figura 2.11.

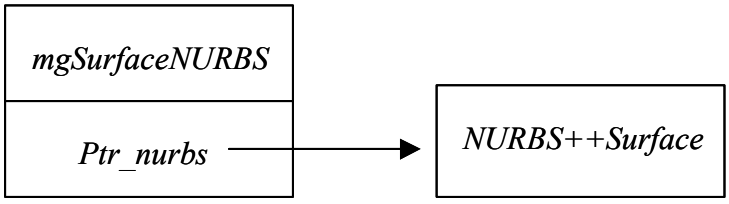


Figura 2.11 – Comunicação entre uma superfície do MG e da biblioteca NURBS++.

Uma hierarquia de classes semelhante é adotada para a modelagem de curvas, como pode ser visto na Figura 2.12. Da mesma forma, a classe *mgCurveNURBS* é a responsável pelo acesso da maioria dos dados referentes às curvas definidas na biblioteca NURBS++, enquanto que as classes *mgSpline*, *mgArc* e *mgPLine* representam os tipos de curvas definidos no modelador MG.

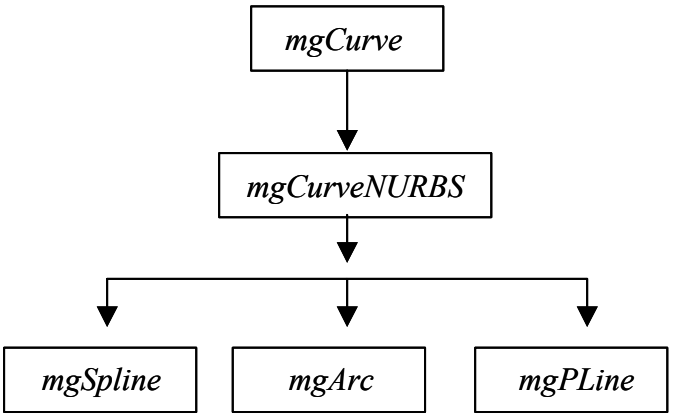


Figura 2.12 – Organização das classes de curvas no modelador MG.

2.3 - Técnicas de Modelagem para Geração de Curvas e Superfícies

Dentre os vários enfoques utilizados em uma modelagem de curvas e superfícies, dois aspectos são determinantes: a metodologia de interface com o usuário e a representação interna usada nos algoritmos geométricos. A interface com o usuário usada no MG é baseada em técnicas de manipulação direta e consiste de três aspectos fundamentais: criação de formas, especificação de transformações tridimensionais, e modificação dos parâmetros de visualização. A estrutura de dados e os algoritmos geométricos usam a idéia de "poligonalização" adaptativa para representação de curvas e superfícies paramétricas.

No MG, a definição das coordenadas 3D necessárias para a construção das curvas é feita com um plano de interface auxiliar, mostrado na Figura 2.13, que também é utilizado na especificação das transformações geométricas. A técnica de manipulação direta usada para implementar essas transformações no espaço 3D é baseada em idéias apresentadas por Emmerik [24]. Essa técnica, implementada por Coelho [17] no modelador MG, oferece facilidades tais como a atração para pontos no plano de interface (*snap-to-grid*) e a atração para outros objetos já definidos.

A técnica de modelagem adotada no MG é baseada na criação de curvas que formam a base para a geração das superfícies desejadas. Para criar uma curva no ambiente, o usuário escolhe uma forma a partir de três opções: linha poligonal, arco de círculo, ou

interpolação por *B-Splines*. Os pontos são fornecidos posicionando o cursor na área de desenho da interface. Cada ponto fornecido é adicionado na descrição geométrica da curva. Pontos da curva que está sendo criada podem ser atraídos para pontos extremos de curvas anteriormente definidas, facilitando a criação de curvas adjacentes entre si. Uma tolerância, controlada pelo usuário através do cursor, determina a área de atração. A Figura 2.13 mostra a área de desenho durante o processo de criação de uma curva *B-Spline*.

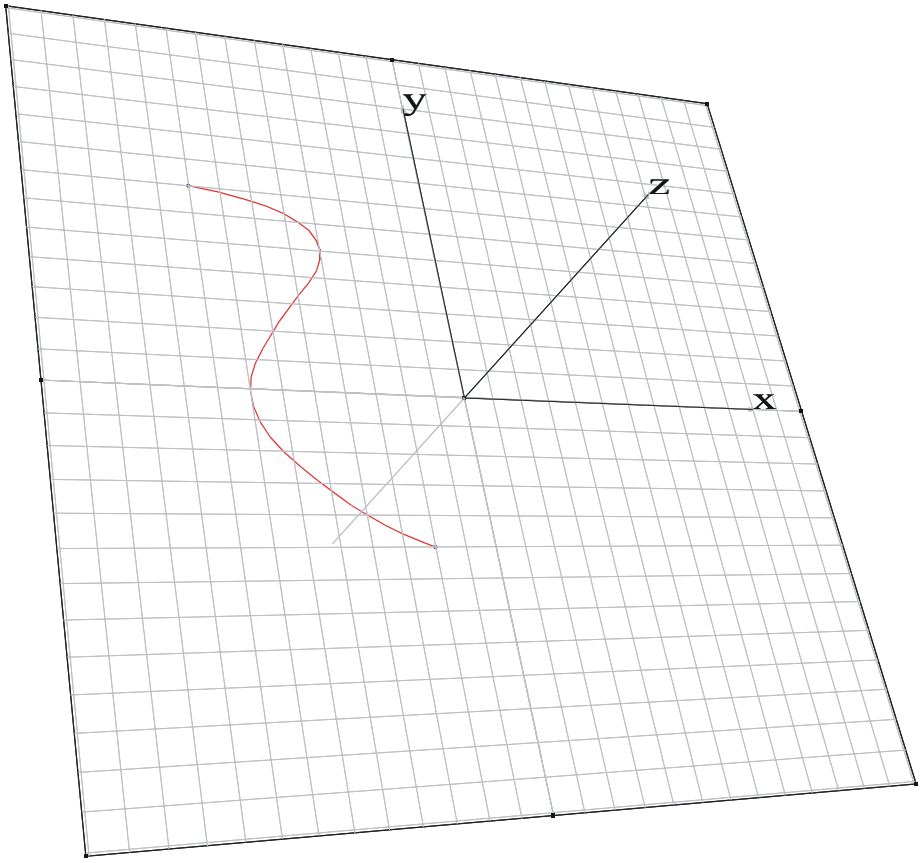


Figura 2.13 – Área de desenho durante o processo de criação de uma curva no MG.

Ao final da edição, os pontos são passados para a biblioteca NURBS++ que gera a curva NURBS correspondente. O ponteiro para essa curva NURBS é armazenado na estrutura de dados MG, juntamente com uma representação da poligonal equivalente adaptativa associada a ela. Essa poligonal, implementada com base no trabalho apresentado por Figueiredo [26], é utilizada, por exemplo, em eventos de seleção e desenho das curvas geométricas correspondentes. A representação da poligonal adaptativa é comum para todos os tipos de curvas e os algoritmos geométricos utilizados na sua manipulação são simples e objetivos.

A criação de superfícies é feita utilizando as curvas existentes. O usuário escolhe um método de construção de superfícies, pré-seleciona as curvas que definem completamente a superfície desejada (por exemplo, na criação de uma superfície de Coons, deve-se definir quais são as curvas de bordo que delimitarão tal superfície) e cria a superfície pressionando o botão correspondente. Nesse processo, os dados referentes ao tipo de superfície desejada são passados para a biblioteca NURBS++ que gera a superfície NURBS correspondente. A referência para essa superfície NURBS é armazenada no MG, juntamente com uma discretização da superfície (malha de elementos finitos), utilizada em eventos de seleção e desenhos das superfícies geométricas, bem como em algoritmos geométricos (por exemplo, interseção de superfícies). Essa malha ou discretização é gerada de acordo com o tipo de mapeamento escolhido pelo usuário (triangular ou quadrilateral, por exemplo).

Para exemplificar o processo de criação de uma superfície, a Figura 2.14 mostra a geração de um toro (*torus*) usando a técnica de *sweep* (descrita com detalhes mais adiante) [18]. A Figura 2.14a mostra os dois arcos criados para este propósito. A primeira curva selecionada define a forma da superfície, enquanto que a segunda curva estabelece a trajetória pela qual a primeira curva será transladada. A Figura 2.14b mostra a superfície gerada.

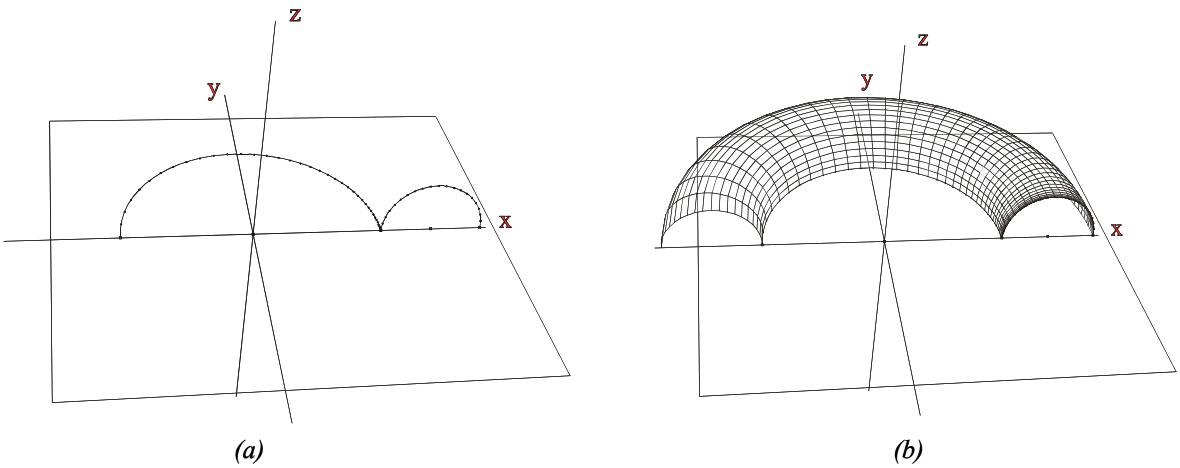


Figura 2.14 – Superfície gerada pela técnica de *sweep*: a) curvas bases; b) superfície gerada.

2.4 – Modelagem de Superfícies

Esta seção descreve a metodologia e formulação utilizadas na criação de cada tipo de superfície no modelador MG. Como foi dito na seção anterior, essa metodologia é baseada na construção das superfícies a partir de curvas espaciais pré-selecionadas pelo usuário do MG.

Deve-se observar que nem todas as metodologias utilizadas na construção das superfícies necessitam das informações de suas curvas de bordo. Por outro lado, a modelagem proposta neste trabalho requer que tais curvas sejam definidas. Assim, sempre que uma superfície é construída e as curvas de bordo não estão completamente definidas, elas são criadas automaticamente pelo MG. Essa tarefa é facilitada, pois na biblioteca NURBS++ existem métodos que determinam as *isocurvas* relacionadas a uma determinada coordenada paramétrica. Assim, é possível, por exemplo, obter as curvas de bordo de uma superfície definida no espaço paramétrico $0 \le u \le 1$ e $0 \le v \le 1$.

As seções subseqüentes apresentam as metodologias e formulações utilizadas na construção de superfícies no MG. Os tópicos descritos nessas seções são encontrados nos livros de Piegl e Tiller [54] e Farin [25], onde informações mais detalhadas sobre o assunto podem ser obtidas.

2.4.1 - Superfícies Bilineares

Superfícies *bilineares* são completamente definidas por um circuito de quatro pontos. Uma representação NURBS dessa superfície é obtida pela interpolação linear entre os segmentos formados por estes pontos. Assim, se $P_{0,0}$, $P_{1,0}$, $P_{0,1}$ e $P_{1,1}$ são quatro pontos no espaço tridimensional, a superfície NURBS *bilinear* é definida por:

$$S(u,v) = \sum_{i=0}^{1}\sum_{j=0}^{1} N_{i,1}(u) N_{j,1}(v) P_{i,j} \ , \qquad (2.4)$$

com

$$U = V = \{0,0,1,1\}$$

onde, $N_{i,j}$ são as funções bases *B-Spline* usuais.

Essa equação representa uma interpolação linear simples entre as linhas do contorno opostas em cada uma das direções.

2.4.2 - Superfícies Geradas por Skins

Superfícies *skin*, também conhecidas como *loft*, são aquelas obtidas pela interpolação de um conjunto de curvas ao longo de uma direção. Essas curvas representam as seções transversais da superfície gerada e não são necessariamente planas. Considere um conjunto de curvas dada pela expressão (2.5). Essas curvas são as seções transversais de uma superfície gerada por *skin*.

$$C_k^w(u) = \sum_{i=0}^{n} N_{i,p}(u) P_{i,k}^w, \; k = 0,...,K \tag{2.5}$$

Se todas as curvas $C_k^w(u)$ são definidas sobre o mesmo vetor de *knots*, *U*, e elas possuem o mesmo grau de interpolação *p*, então, para a direção *v*, um grau de interpolação *q* é escolhido, e um vetor de knots, *V*, é calculado. Esse vetor *V* é usado para fazer *n+1* interpolações curvas através dos pontos de controle das curvas das seções, obtendo os pontos de controle $Q_{i,j}^w$ das superfícies *skin* geradas. Assim, $Q_{i,j}^w$ é o *j*-ésimo ponto de controle da curva interpolante através dos pontos $P_{i,0}^w,...,P_{i,k}^w$.

A construção desse tipo de superfície no MG é realizada a partir de uma seqüência de curvas espaciais definidas pelo usuário, como pode ser visto na Figura 2.15. Não é necessário definir curvas no bordo livre. Na direção desses bordos, uma interpolação linear é feita pela biblioteca NURBS++, gerando uma superfície linear ao longo dessa direção. As curvas geradoras fornecidas são respeitadas na construção da superfície. Assim, a superfície criada tem essas curvas como suas seções transversais.

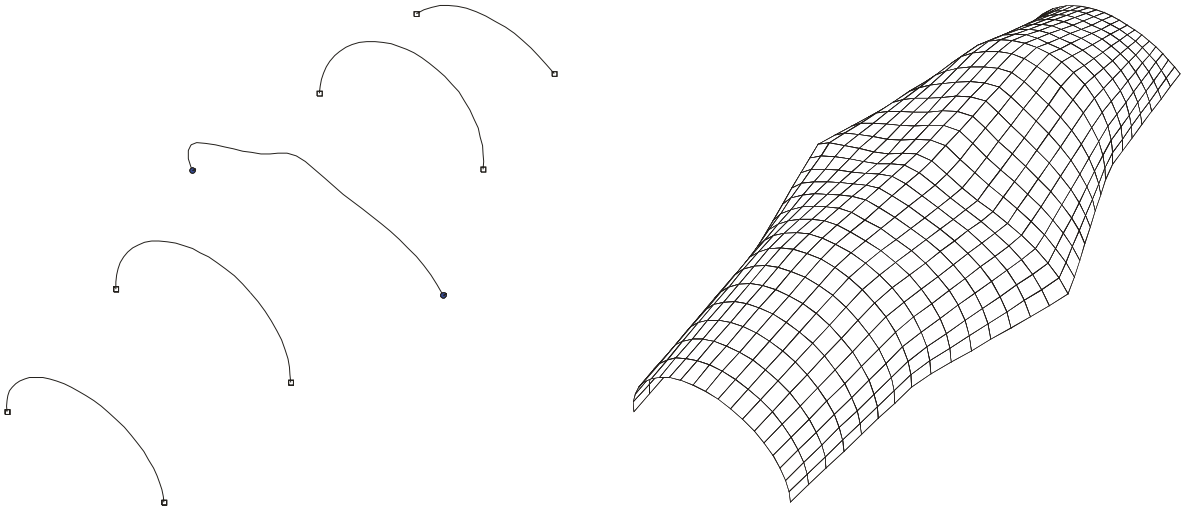


Figura 2.15 – Exemplo de superfícies geradas por *Skin* e suas curvas geradoras.

Este tipo de superfícies é importante, pois serve como base para a geração de outros tipos de superfícies, como será visto nas seções seguintes.

2.4.3 - Superfícies de Coons

Uma extensão das superfícies *bilineares* é a superfície conhecida como *Coons*. Esse tipo de representação usa os pontos definidos ao longo dos segmentos do contorno como pontos de controle da superfície. As superfícies de *Coons* são mais interessantes, pois não existe limitação em relação ao grau de interpolação das curvas do contorno que definem a superfície.

Desta forma, se $C_0(u)$, $C_1(u)$, $C_0(v)$ e $C_1(v)$ são quatro curvas que delimitam o contorno de uma superfície de *Coons*, a formulação para ela é dada por:

$$S(u,v) = R_1(u,v) + R_2(u,v) - T(u,v) \tag{2.6}$$

onde $R_1(u,v)$ e $R_2(u,v)$ são superfícies geradas por *skins* entre as curvas $C_0(u)$ e $C_1(u)$, e $C_0(v)$ e $C_1(v)$, respectivamente. $T(u,v)$ é uma superfície gerada pelo produto tensorial bilinear dada por:

$$T(u,v) = \begin{bmatrix} 1 & u \end{bmatrix} \cdot \begin{bmatrix} S_{0,0} & S_{0,1} \\ S_{1,0} & S_{1,1} \end{bmatrix} \cdot \begin{bmatrix} 1 \\ v \end{bmatrix}$$

A construção, no modelador MG, desse tipo de superfície é realizada a partir da definição das quatro curvas de bordo que limitam a superfície. Essas curvas podem ser de qualquer tipo, não sendo necessariamente retas. A Figura 2.16 mostra um exemplo de uma superfície de *Coons* e suas curvas geradoras.

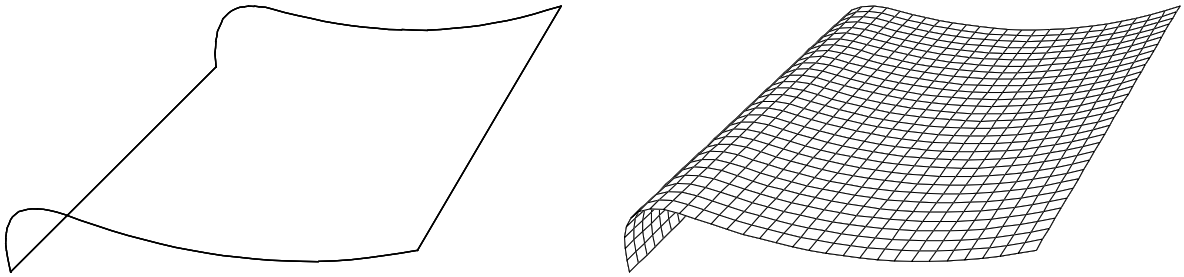


Figura 2.16 – Exemplo de superfície *Coons* e suas curvas geradoras.

2.4.4 – Superfícies Geradas por Sweeps

Superfícies geradas por *sweeps* são aquelas que podem ser obtidas pelo arrasto de uma seção curva ao longo de uma trajetória, também curva. Assim, se uma trajetória for denotada por $T(v)$ e a seção curva por $C(u)$, uma formulação geral para o *sweep* pode ser escrita como

$$S(u,v) = T(v) + M(v)C(u) \qquad (2.7)$$

onde $M(v)$ é uma matriz *3x3* que incorpora rotação e escalas não-uniformes de $C(u)$ como uma função de *v*.

No MG, as superfícies geradas por *sweeps* podem ser obtidas de três formas diferentes. Uma forma de construção é definida pela indicação de uma curva base para o *sweep* e pela direção ao qual essa curva deve se propagar. Esse tipo de geração de superfície é

conhecido como *sweep* de translação. A segunda técnica utilizada para criação de uma superfície do tipo *sweep* é baseada na rotação de uma curva base em torno de uma direção. Esse tipo de geração é conhecido como *sweep* de rotação. Existe ainda o *sweep* genérico, onde uma curva base é arrastada ao longo de uma outra curva que define a trajetória do *sweep*. Um exemplo de superfície gerada por um *sweep* genérico é mostrado na Figura 2.17.

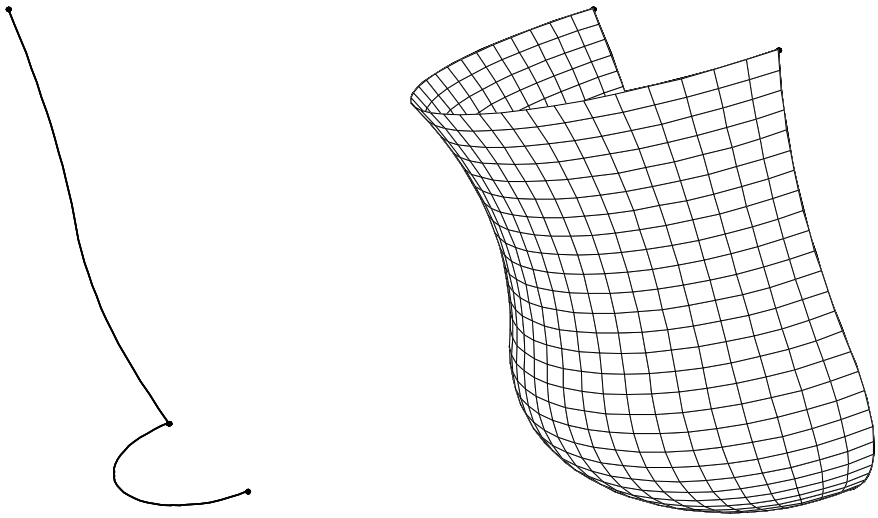


Figura 2.17 – Exemplo de uma superfície gerada por *sweep* genérico e suas curvas geradoras.

2.4.5 – Superfícies de Gordon

Uma superfície de *Gordon* é aquela gerada por uma rede bidirecional de curvas espaciais. Se

$$C_k(u) = \sum_{i=0}^{n} N_{i,p}(u) P_{k,i} \ , \ k = 0,...,r$$

e

$$C_l(v) = \sum_{j=0}^{m} N_{j,q}(v) P_{l,j} \ , \ l = 0,...,s \tag{2.8}$$

são dois conjuntos de curvas NURBS, a construção de superfícies de *Gordon* é dada pela composição de três superfícies, como descrito na Equação 2.9. Nessa equação,

$L_1(u,v)$ e $L_2(u,v)$ são superfícies geradas por *skin* das curvas em cada uma das direções, e $T(u,v)$ é uma superfície obtida pelo produto tensorial das curvas.

$$S(u,v) = L_1(u,v) + L_2(u,v) - T(u,v) \tag{2.9}$$

No MG, a construção desse tipo de superfície é realizada a partir de uma seqüência de curvas espaciais definidas pelo usuário, como pode ser visto na Figura 2.18. Deve-se indicar quais são as curvas geradoras em cada uma das direções paramétricas da superfície. As curvas geradoras fornecidas são respeitadas na construção da superfície. Elas funcionam como seções transversais, em ambas direções, da superfície gerada.

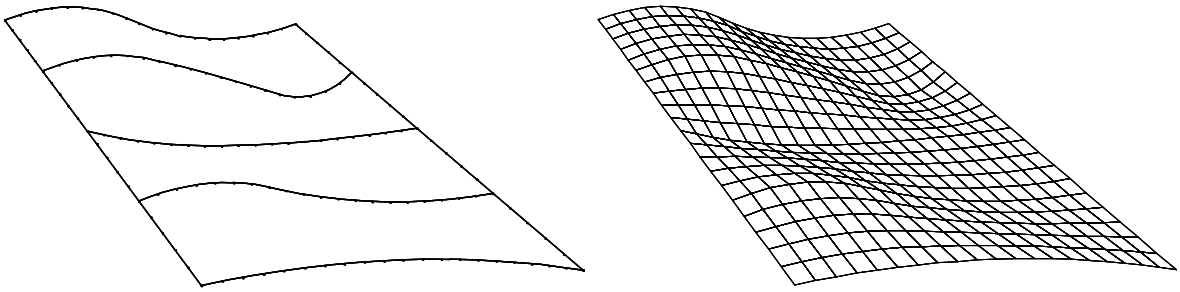


Figura 2.18 – Exemplo de superfícies de *Gordon* e suas curvas geradoras.

Pode-se notar que esse tipo de superfície é uma generalização das superfícies de *Coons*. Enquanto as superfícies de *Coons* são geradas exatamente por uma rede de duas curvas em cada direção, as superfícies de *Gordon* não possuem tal limitação.

2.4.6 – Superfícies Triangulares

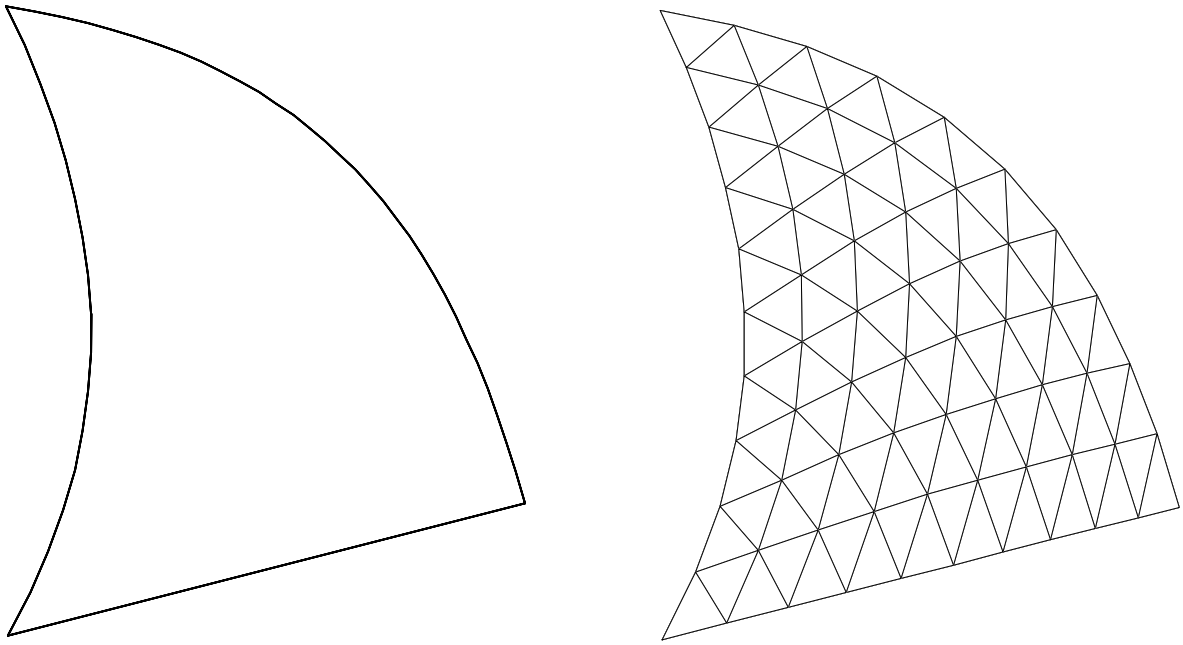
Superfícies triangulares são definidas por um circuito de três curvas. A representação dessas superfícies usa os pontos localizados ao longo dos segmentos do contorno como seus pontos de controle.

O tratamento dado às superfícies triangulares é diferente do tratamento dado às outras superfícies, pois não se conhece uma formulação matemática NURBS para representar

essas superfícies. Representação convencional de superfícies por NURBS é aplicada somente na geração de retalhos de superfícies retangulares no seu espaço paramétrico.

Neste trabalho, essa limitação é parcialmente resolvida com a expansão da biblioteca NURBS++, utilizada pelo MG, de tal forma que superfícies triangulares sejam aproximadas por superfícies cúbicas do tipo Bézier [40]. Essa expansão é realizada derivando-se a classe base que representa superfícies na biblioteca NURBS++, redefinindo apenas os métodos necessários para a completa definição das superfícies triangulares. É importante notar que essa implementação apresenta restrições, pois nem todas as curvas geométricas usadas na definição de uma superfície triangular podem ser representadas por curvas Bézier.

A construção, no modelador MG, desse tipo de superfície é realizada a partir da definição das três curvas de bordo que limitam a superfície. A Figura 2.19 mostra um exemplo de uma superfície triangular gerada pelo MG e suas curvas geradoras.



(a) *(b)*

Figura 2.19 – Exemplo de superfícies triangulares e suas curvas geradoras.

2.5 - Modelagem de Sólidos

No MG, as técnicas utilizadas para modelagem de sólidos se confundem com o próprio mapeamento associado a cada sólido. Ou seja, gerar um sólido no MG é o mesmo que gerar uma malha 3D de elementos finitos, onde essa malha é utilizada para representar o sólido criado (por exemplo, no desenho ou seleção do objeto correspondente).

Existem quatro técnicas utilizadas no MG para a modelagem de sólidos (e suas respectivas malhas de elementos finitos). Uma dessas técnicas descreve uma metodologia para geração de malhas sólidas, não-estruturadas, em domínios arbitrários. As outras três técnicas usam o *sweep* para o mapeamento dos sólidos. Na técnica de *sweep*, a geração de sólidos é feita através do "arrasto" de uma seção transversal bidimensional (retalho de superfície) ao longo de uma trajetória no espaço. Essas técnicas são descritas nas seções seguintes. Maiores detalhes sobre elas podem ser vistos nos trabalhos de Miranda [42,44].

É importante observar que a consistência topológica exigida quando dois retalhos de superfícies são adjacentes entre si também é requerida aqui. Ou seja, dois sólidos adjacentes devem ter pelo menos um retalho de superfície comum a ambos, localizado na interface entre eles, como pode ser visto na Figura 2.20. A Figura 2.20a representa um modelo gerado com dois sólidos adjacentes, enquanto que a Figura 2.20b mostra esse modelo explodido, ilustrando a necessidade da existência do retalho de superfície comum localizado na interface entre os sólidos *A* e *B*. Essa necessidade exige que retalhos de superfícies sejam criados automaticamente pelo modelador MG para delimitar o contorno dos sólidos gerados e garantindo a consistência topológica nesta etapa da modelagem.

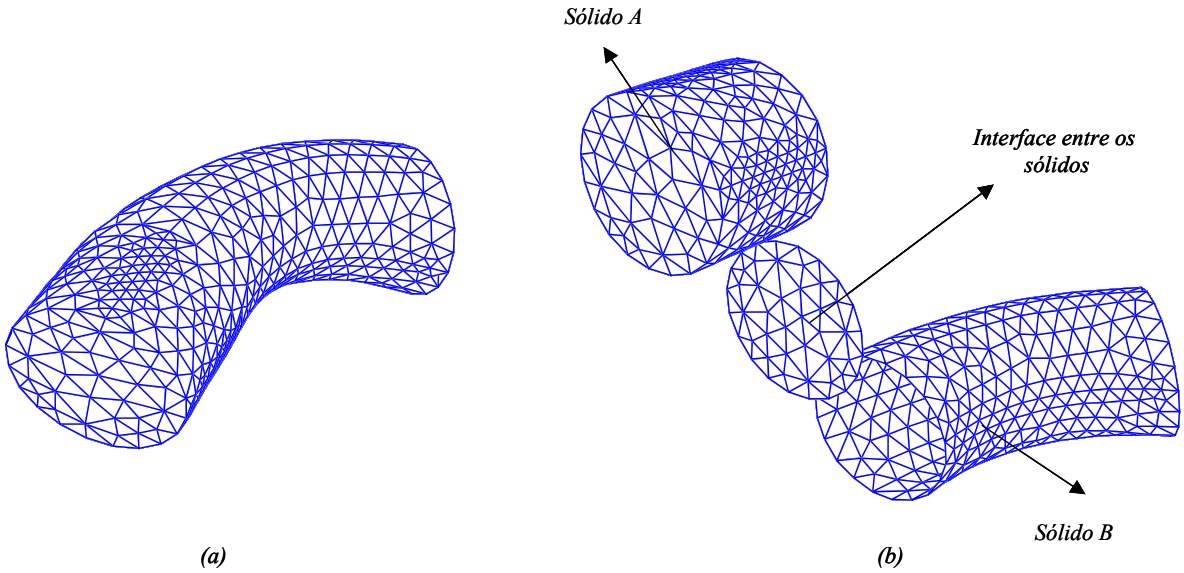


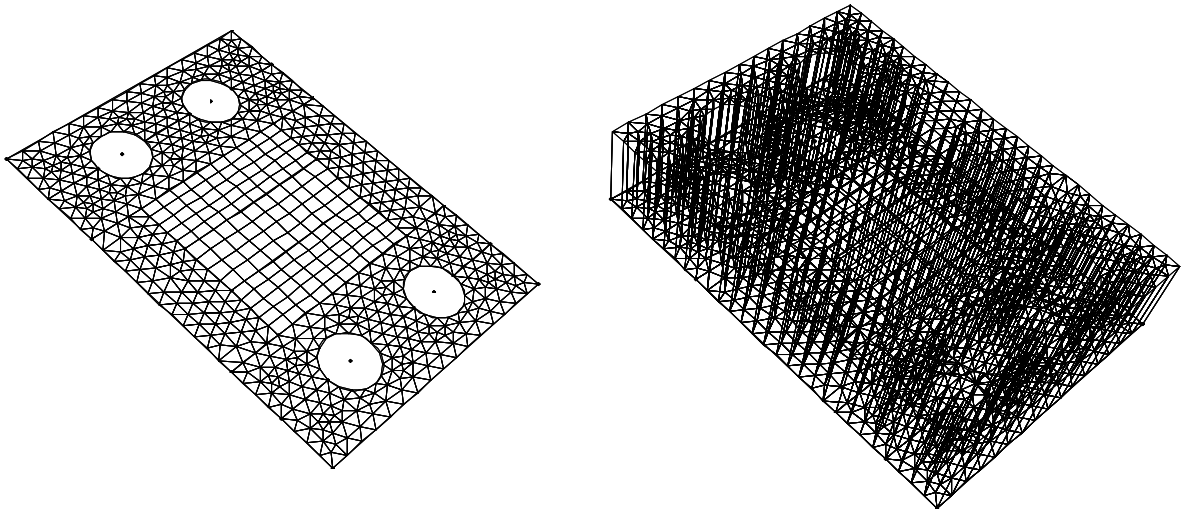
Figura 2.20 – Interface entre sólidos adjacentes: a) modelo completo; b) modelo explodido – visualização do retalho de superfície na interface entre os dois sólidos.

2.5.1 – Sólidos Gerados por Sweeps

Esta seção apresenta as técnicas de *sweep* utilizadas pelo MG para o mapeamento de sólidos.

2.5.1.1 – Extrusão

Uma das técnicas mais simples de *sweep* para o mapeamento de sólidos é o translacional, também chamado de *extrusão*. Nesse tipo de técnica, o "arrasto" da área (superfície representando uma seção transversal do sólido) é feito ao longo de uma linha reta no espaço tridimensional (definido por um vetor no MG). Os elementos finitos sólidos gerados podem ser pentaédricos e/ou hexaédricos. A Figura 2.21 mostra um exemplo desse tipo de mapeamento [42].

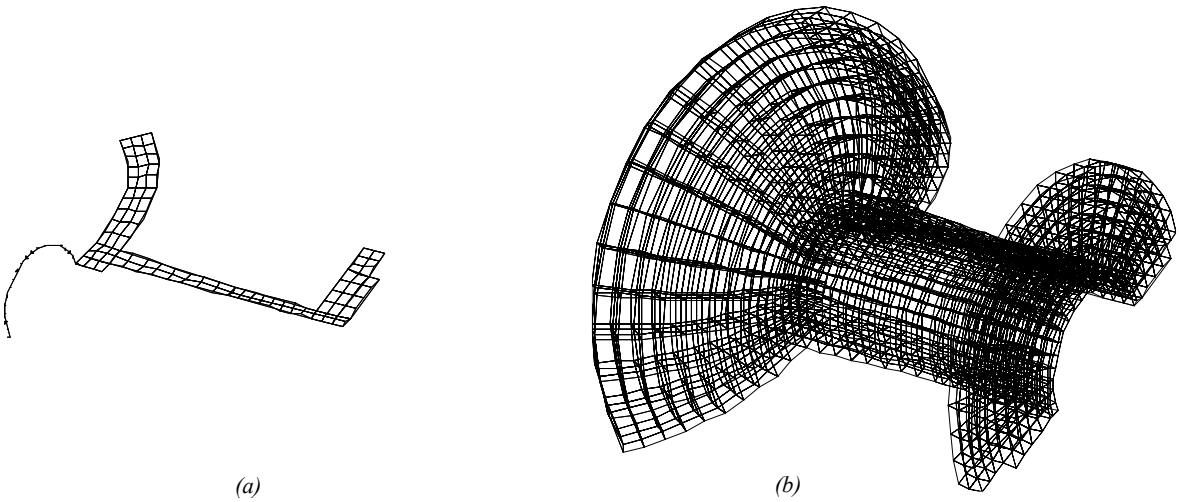


(a) *(b)*

Figura 2.21 – Mapeamento de sólidos por extrusão: a) entidades geradoras do mapeamento; b) sólido gerado pela extrusão.

2.5.1.2 – Sweep Curvo

Essa técnica é semelhante à anterior. No entanto, o arrasto da seção transversal (superfície) é feito ao longo de uma curva no espaço, como pode ser visto na Figura 2.22. Essas curvas indicam a trajetória do *sweep*. A malha associada à superfície pode ser formada por elementos triangulares ou quadrilaterais. Os vetores tangentes à trajetória do *sweep* devem seguir a mesmo sentido da trajetória do *sweep.* Os elementos finitos sólidos gerados podem ser pentaédricos e/ou hexaédricos [42].

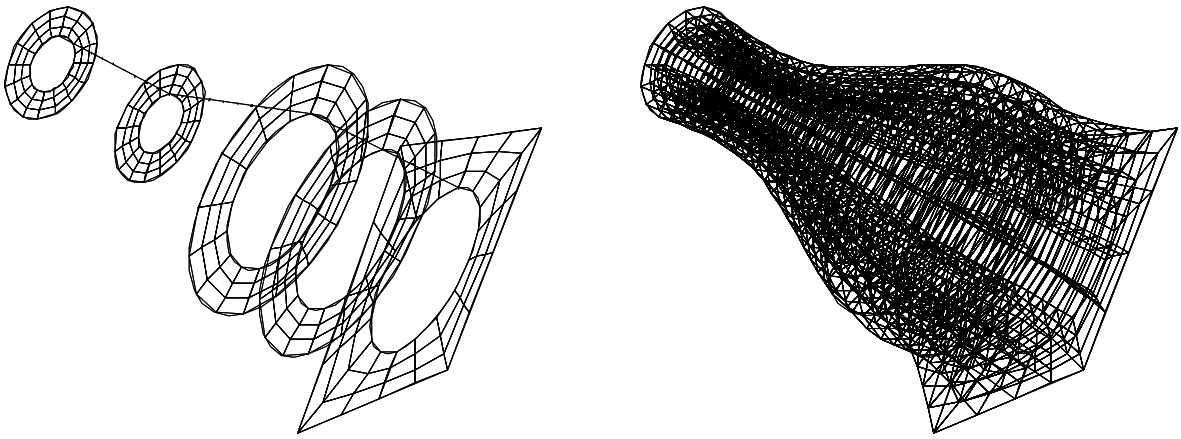


(a) *(b)*

Figura 2.22 – Mapeamento de sólidos pelo *sweep* de uma superfície ao longo de uma curva no espaço: a) entidades geradoras do *sweep*; b) sólido gerado pelo *sweep*.

2.5.1.3 – Mapeamento Transfinito Tridimensional

Essa técnica representa um procedimento para a construção automática de malhas sólidas a partir das seções transversais (superfícies) de um modelo a ser discretizado em elementos finitos. Essas seções transversais devem estar ligadas por uma curva que permite identificar os pontos bases de cada seção transversal e o número de seções intermediárias entre seções, necessários para a utilização do algoritmo. Os pontos bases são os pontos onde as curvas e seções transversais se interceptam. O número de seções intermediárias entre seções transversais é o número de divisões da curva que liga essas seções. O reconhecimento da ordem das seções transversais é feito de modo automático, sem que haja necessidade de intervenção do usuário [42]. Essa técnica é capaz de gerar elementos sólidos pentaédricos e/ou hexaédricos. A Figura 2.23 ilustra esse tipo de mapeamento [42].



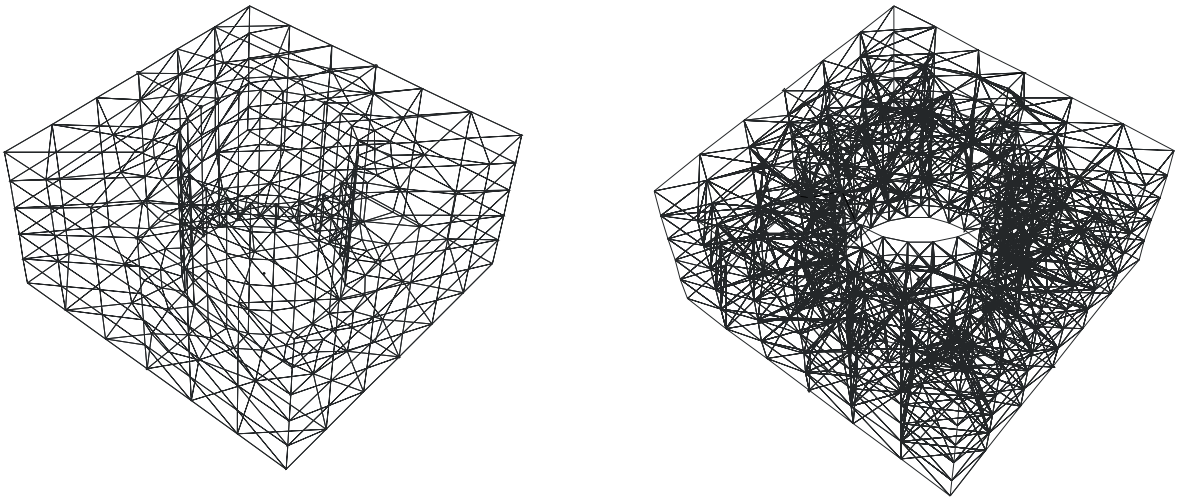
(a) *(b)*

Figura 2.23 – Sólido gerado pelo mapeamento transfinito tridimensional: a) entidades geradoras do mapeamento; b) sólido gerado pelo mapeamento.

2.5.2 – Geração de Sólidos em Domínio Arbitrário

Essa técnica usa um algoritmo para geração de malhas volumétricas não-estruturadas de tetraedros para domínios arbitrários [13,14]. O domínio dos sólidos gerados é definido pelas superfícies que os delimitam. Essas superfícies devem ser indicadas como entrada de dados para o algoritmo responsável pela geração de tais sólidos. As malhas associadas a cada uma dessas superfícies são definidas usando elementos triangulares. Tais malhas são convertidas em uma representação única de elementos finitos, onde a topologia desses elementos define se uma região delimitada por tais elementos é fechada, bem como posiciona-se as orientações das normais de tal forma que elas apontem para dentro da região formada.

Se uma região é identificada neste processo, a representação única de elementos finitos é passada como entrada de dados para o algoritmo que gera a malha não-estruturada de elementos tetraédricos na região correspondente. Essa malha é a própria representação do sólido com elementos tetraédricos. O algoritmo utilizado na geração da malha é descrito na seção seguinte. A Figura 2.24 ilustra um exemplo de utilização dessa técnica.



(a) *(b)*

Figura 2.24 – Geração de sólidos com domínio arbitrário.

2.6 – Geração de Malhas Não-Estruturadas

Na modelagem de problemas de engenharia pelo MEF, o passo seguinte à descrição da geometria é a geração do modelo numérico para a análise por elementos finitos. Como foi dito anteriormente, o modelo numérico é composto pela malha de elementos finitos e pelos atributos associados a ela. Essa seção trata dos algoritmos utilizados no MG para geração de malhas não-estruturadas de elementos finitos.

O objetivo de descrever especificamente os algoritmos para geração de malhas não-estruturadas é que eles são essenciais para simulações genéricas de problemas tridimensionais usando o MEF, pois podem ser utilizados em domínios arbitrários. Além disso, esses algoritmos também são úteis em processos adaptativos, onde as malhas são refinadas de acordo com critérios (numéricos ou geométricos) definidos na simulação.

No MG, os algoritmos utilizados para geração de malhas não-estruturadas volumétricas e de superfícies combinam técnicas de avanço de fronteira [55,70] com decomposição espacial recursiva [63,73]. Dentro dessa filosofia, a principal motivação para o uso destes algoritmos é a necessidade de atender alguns requisitos específicos que não são tratados diretamente (ou não são eficientes) em outros algoritmos encontrados na

literatura. Inicialmente, esses algoritmos devem produzir elementos com boa forma evitando, sempre que possível, a geração de elementos de má qualidade. Esses algoritmos utilizam um procedimento que procura melhorar localmente a qualidade dos elementos gerados. Isso é baseado em parâmetros que qualificam a forma de um elemento finito triangular ou tetraédrico.

Esses algoritmos devem gerar malhas que sejam conformes com as discretizações existentes no contorno do domínio. No caso do algoritmo para geração de malhas volumétricas, ele deve gerar malhas que sejam conformes com as malhas triangulares das superfícies do contorno do modelo. Essa característica é importante quando se deseja gerar localmente novas malhas em processos adaptativos.

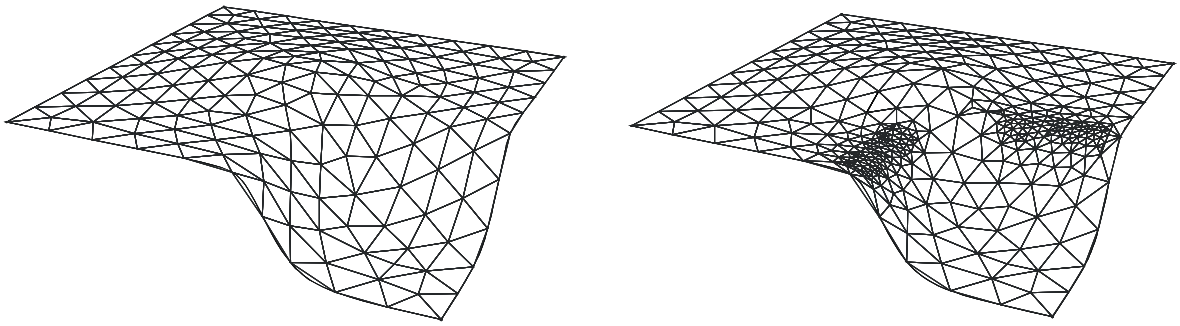
Outra característica que esses algoritmos apresentam é a geração de malhas com uma boa transição entre regiões com diferentes tamanhos de elementos.

Por fim, esses algoritmos devem tratar casos onde as superfícies possuem curvaturas acentuadas. Nesses casos, os algoritmos devem refinar a malha localmente nas regiões onde ocorrem estas curvaturas. Este procedimento evita, por exemplo, que elementos vizinhos formem locais pontiagudos, gerando uma situação indesejável em uma análise de elementos finitos.

Nas seções seguintes, os algoritmos utilizados neste trabalho para geração de malhas não-estruturadas volumétricas e em superfícies são apresentados. É importante observar que tais algoritmos são aplicados localmente em cada região (superfície ou sólido) do modelo. Uma etapa seguinte à geração local das malhas refere-se à geração de uma malha única de elementos finitos. Essa etapa é realizada atendendo aos princípios básicos de elementos finitos (por exemplo, manutenção da compatibilidade entre elementos adjacentes).

2.6.1 – Geração de Malhas em Superfícies

Esta seção descreve o algoritmo para a geração de malhas de superfícies utilizado no MG. Esse algoritmo foi desenvolvido por Miranda [44] e é aplicado na geração de malhas triangulares em superfícies com geometria arbitrária utilizando a sua descrição paramétrica. Como foi dito anteriormente, essa descrição é utilizada por ser mais comum, pois utiliza os algoritmos de triangulação bidimensional, com correções das distorções da geometria da superfície. Além disso, a geração de malhas diretamente na superfície 3D requer maiores critérios para a validação dos elementos triangulares, tornando o algoritmo mais lento. Utilizando o espaço paramétrico da superfície, a superfície 3D é mapeada para uma superfície 2D e então a triangulação é realizada com correções das distorções geométricas. Em seguida, a malha resultante desta triangulação é reconduzida para o espaço 3D. Este processo pode ser visualizado na Figura 2.4. A Figura 2.25 mostra um exemplo de malha em superfície 3D.



(a) *(b)*

Figura 2.25 – Exemplo de malha em uma superfície 3D: a) sem considerar distorções; b) considerando distorções.

O algoritmo se baseia em técnicas de avanço da fronteira com decomposição espacial recursiva. Essa decomposição espacial é realizada utilizando-se uma árvore quaternária (*quadtree*) para armazenar métricas e desenvolver diretrizes locais usadas para definir o tamanho dos elementos gerados. A técnica de avanço de fronteira é baseada em um procedimento padrão encontrado na literatura, com algumas modificações, descritas a seguir.

Os dados de entrada para geração de malhas de superfícies utilizando este algoritmo são descritos por uma lista de nós, definidos por suas coordenadas paramétricas, e uma lista com o número de arestas de cada circuito (porção conexa da fronteira) do modelo.

Todos os detalhes referentes a este algoritmo podem ser vistos em [44].

2.6.2 – Geração de Malhas Volumétricas

O algoritmo utilizado no MG para geração de malhas volumétricas não-estruturadas é o proposto por Cavalcante Neto [13,14]. Assim como o algoritmo utilizado na geração de malhas em superfícies, esse algoritmo também se baseia em uma técnica de avanço da fronteira combinado com uma técnica de decomposição espacial recursiva. No caso desse algoritmo, essa técnica é uma árvore octária (*octree*), utilizada para desenvolver diretrizes locais usadas para definir o tamanho dos elementos a serem gerados. A técnica de avanço de fronteira utilizada nesse algoritmo é baseada em um procedimento padrão encontrado na literatura [36,45,52] com duas fases adicionais, utilizadas para garantir a geração de uma malha volumétrica válida para praticamente qualquer domínio.

Os dados de entrada para geração de malhas volumétricas utilizando este algoritmo são fornecidos por uma lista de nós definindo suas coordenadas e uma lista de faces definidas por sua conectividade nodal. Dentre as vantagens dessa estrutura de dados, está o poder de representação de geometrias quaisquer, incluindo furos, cavidades e trincas, de uma maneira simples, podendo ser facilmente incorporada em qualquer sistema de elementos finitos.

Este algoritmo está organizado em três passos. O primeiro é a geração da *octree*. Como dito anteriormente, a *octree* tem como funcionalidade 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.

O segundo passo seguinte desse algoritmo é o procedimento de avanço da fronteira. Esse processo começa com a superfície que limita o domínio a ser preenchido com a malha volumétrica. Elementos volumétricos são extraídos do domínio um por vez. Quando cada elemento é extraído, a superfície limitante é atualizada e o processo se repete. O procedimento termina quando a malha é gerada sobre todo domínio ou quando uma ou mais cavidades internas permanecem não discretizadas, de onde elementos válidos não podem ser extraídos. Nesse algoritmo, para garantir a geração de elementos válidos, o processo de avanço de fronteira é dividido em três fases. Na primeira fase, a geração de elementos é realizada baseando-se na geometria. Nessa fase, tenta-se gerar elementos com formas ótimas. Quando não é mais possível gerar estes elementos inicia-se, então, a segunda fase deste procedimento: a geração de elementos baseado na topologia. Tenta-se, então, criar elementos válidos, mas não necessariamente de boa forma. Na última fase, um procedimento de "volta-passo" (*backtracking*) é utilizado para eliminar algumas faces dos elementos que estão impedindo a algoritmo de completar a malha.

O procedimento de avanço da fronteira pode gerar elementos tetraédricos com formas não-ótimas. Para tentar evitar tal problema, esse algoritmo possui um último passo que busca uma melhoria local da malha. Esse passo possui duas fases. A primeira é a utilização de uma técnica de suavização convencional que consiste na relocação de nós baseado na média das coordenadas nodais, com testes de validação. O segundo passo é um procedimento de "volta-passo", similar àquele da última fase do procedimento de avanço de fronteira. Esse procedimento remove faces de elementos de forma ruim para criar uma região onde elementos com melhor forma possam ser gerados.

Todos os detalhes referentes a este algoritmo podem ser encontrados em [13,14].

2.7 - Atributos

Como foi dito anteriormente, na maioria dos problemas de engenharia a metodologia adotada na discretização do domínio é comum para os diversos tipos dos problemas que

podem ser analisados pelo MEF. Os atributos, entretanto, são específicos para cada tipo de simulação.

Em muitas situações, é interessante que os modeladores sejam capazes de atender aos diversos tipos de análise em engenharia (por exemplo, estrutural, naval, termodinâmica, etc.). No entanto, esses modeladores implementam diversas funcionalidades e alterações em seu código-fonte não são tarefas fáceis de ser realizadas. Então, em muitas situações é conveniente que os atributos possam ser configurados para atender aos diversos tipos de problemas sem, necessariamente, haver alterações no código computacional do modelador.

Baseado nessas necessidades, o modelador MG usa um sistema, denominado ESAM (*Extensible System Attributes Managment*) [9,34], que é responsável pelo gerenciamento e configuração de todos os atributos de uma simulação.

No ESAM, a configuração dos atributos é feita através de arquivos que contém instruções em uma linguagem de programação extensível (linguagens utilizadas para estender ou configurar uma aplicação para fins particulares) bastante simples. Esta linguagem é interpretada, permitindo que a configuração dos atributos seja feita sem a necessidade de recompilação do sistema. Esse sistema utiliza uma abordagem conhecida como modelagem baseada em geometria. Nesse tipo de abordagem, os atributos são associados às entidades geométricas do modelo e a discretização dessa geometria (malha de elementos finitos) herda, automaticamente, estes atributos.

A modelagem baseada em geometria oferece diversas vantagens, dentre as quais pode-se citar um melhor suporte para a automação do processo de modelagem, incluindo geração automática de malha e análise adaptativa (os atributos são associados às entidades geométricas do modelo e não precisam ser redefinidos no processo de refinamento da malha de elementos finitos), e a simplificação da geração do modelo para a simulação [64].

A Figura 2.26 mostra alguns diálogos definidos no MG e que podem ser usados pelo usuário do programa para capturar dados relacionados a alguns atributos utilizados em uma análise de tensões pelo método dos elementos finitos.

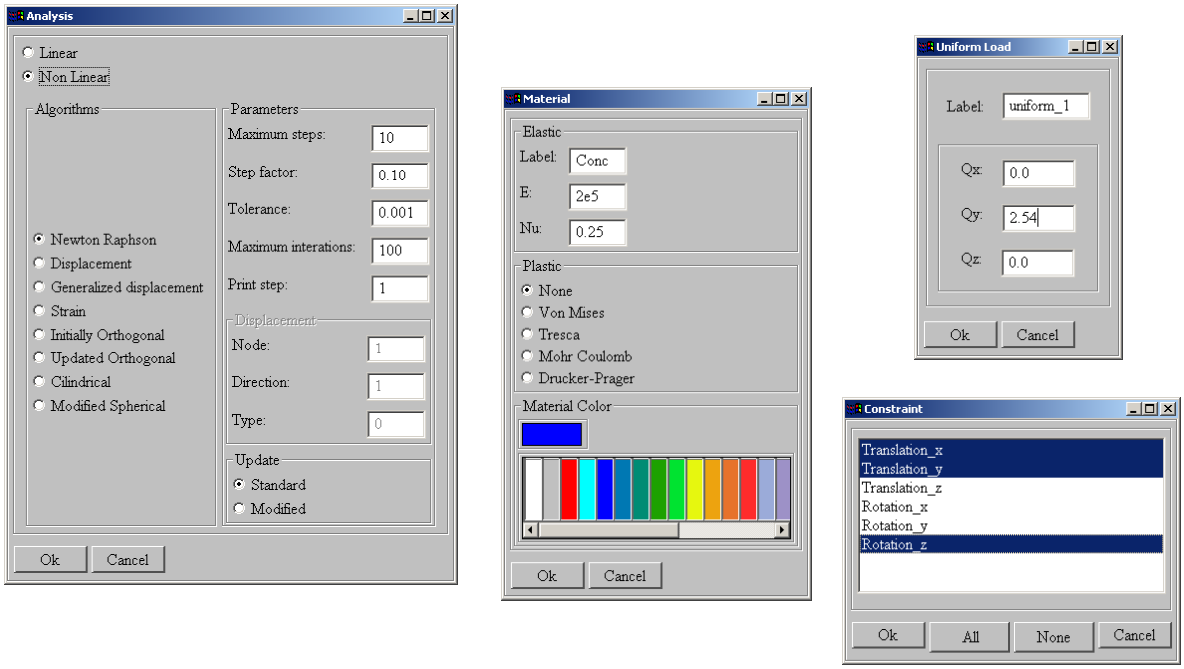


Figura 2.26 – Diálogos utilizados para a captura de dados dos atributos definidos no MG.

Todos os detalhes referentes à descrição do sistema ESAM podem ser vistos em [9,34], inclusive com as especificações necessárias à sua utilização.

3.Estrutura de Dados Híbrida

Este capítulo descreve uma estrutura de dados híbrida baseada em uma organização de classes, no contexto da programação orientada a objetos (POO), de uma nova versão do modelador MG que, mantendo as suas características originais (interface com o usuário simples e eficiente), provê capacidades para a detecção automática de regiões com retalhos de superfícies paramétricas representados por NURBS.

Para isso, um enfoque híbrido baseado na combinação da representação CGC e na estrutura de dados do modelador MG é adotado, onde a representação CGC não é mantida durante todo o processo de modelagem. Nesse enfoque, a interseção de superfícies é realizada usando o algoritmo implementado no modelador MG (descrito no capítulo seguinte), enquanto que a representação CGC é responsável justamente pelo reconhecimento de multi-regiões.

A versão anterior da estrutura de dados do MG foi estendida, permitindo que um modelo CGC possa ser criado, em qualquer instante, quando solicitado pelo usuário para realizar a detecção automática de regiões. Dependendo da complexidade do modelo, o processo de modelagem pode envolver uma série de passos intermediários e a manutenção da consistência entre geometria e topologia após cada etapa desse processo pode ser muito ineficiente. É muito difícil alcançar um grau razoável de eficiência na interface com o usuário se essa consistência for forçada após cada tarefa realizada pelo usuário.

A nova organização de classes também provê suporte para a geração automática de malhas em superfícies e sólidos. A geração de malhas em superfícies é feita no espaço paramétrico de cada superfície. Neste enfoque, o suporte para geração dessas malhas é alcançado pela criação de novas entidades topológicas na estrutura de dados do MG e pelo conceito de *uso* de uma entidade topológica por uma superfície.

3.1 – Representações CGC (Complete Geometric Complex)

Em geral, existem duas estratégias principais para a representação de um modelo geométrico tridimensional: um esquema explícito e um esquema implícito. O mais comum esquema implícito é a CSG (*Constructive Solid Geometry*) [58], cujo objeto alvo é reproduzido por um conjunto de operações *booleanas* aplicadas aos objetos primitivos. No esquema explícito, a geometria do objeto é definida por um conjunto de retalhos de superfícies. Estes retalhos podem ser criados através de uma interface gráfica iterativa. No entanto, a definição completa do modelo sólido requer relações combinatórias entre os vários retalhos de superfícies, que irão resultar na definição das regiões interiores, contorno do modelo e outras informações topológicas. Este tipo de representação de modelos sólidos é chamado B-REP (*Boundary Representation*) [28,38].

As técnicas tradicionais de CSG e B-REP aplicam-se a objetos que decompõem o espaço em três partes: interior, exterior e fronteira. Essa classe de objetos é referida como objetos *manifolds*, porque suas fronteiras constituem-se em variedades de duas dimensões (2-*manifold*) no espaço tridimensional [28]. Isto significa que as técnicas tradicionais não poderiam modelar objetos com multi-regiões.

No entanto, muitas aplicações em engenharia precisam modelar esses objetos com multi-regiões ou objetos que apresentam outras partes *non-manifolds* (isto é, cujas fronteiras não são superfícies 2-*manifold*); por exemplo, faces ou arestas soltas. Por esta razão, muitos trabalhos na literatura têm proposto esquemas de modelagem *non-manifold* [16,22,31,32,60,61,74,75].

A geração de um modelo B-REP *non-manifold* consistente a partir de um conjunto de retalhos de superfícies não é uma tarefa simples e pode envolver interseção de superfícies e detecção de regiões. Considerando os problemas de interface com o usuário, é desejável que a criação do modelo B-REP a partir de um conjunto de retalhos de superfícies seja realizada automaticamente. O modelador deve ser capaz, então, de gerar automaticamente curvas de interseção entre superfícies e as relações topológicas destas entidades como a definição geométrica.

Trabalhos anteriores [16] do grupo de pesquisa no qual este trabalho está inserido propuseram um enfoque *non-manifold* para modelagem de multi-regiões. Uma metodologia geral para a criação e manipulação de uma subdivisão espacial em células com forma e geometria arbitrária foi desenvolvida. A subdivisão espacial pode ser criada através da inserção, uma a uma, de retalhos de superfícies planas, permitindo a inserção de novos retalhos em tempo real. O objeto resultante desta decomposição é classificado como uma CGC (*Complete Geometric Complex*) porque ela é um caso especial de um Complexo Geométrico [60] que ocupa todo o espaço tridimensional (a região externa ilimitada também é representada na subdivisão). A Figura 3.1 mostra dois exemplos de objetos *non-manifold*. A Figura 3.1a apresenta um objeto com duas regiões internas e uma externa. A Figura 3.1b ilustra a criação de um modelo CGC a partir da inserção e eliminação de retalhos de superfícies.

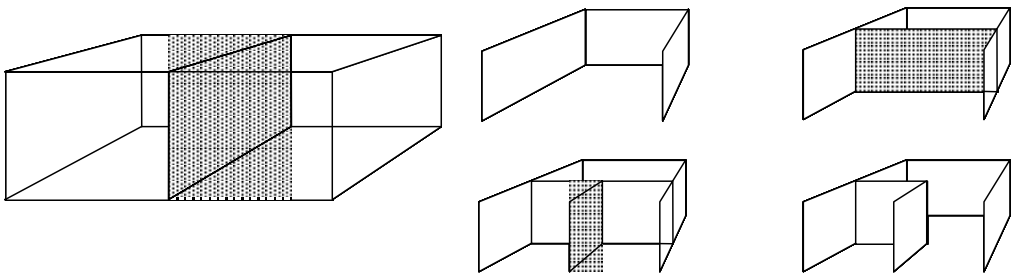


Figura 3.1 – Exemplos de objetos *non-manifold*.

Vários trabalhos têm apresentado métodos para representar subdivisões espaciais. Rossignac e O'Connor [60] têm tratado o problema geral de representação de objetos com *n* dimensões, possivelmente com estruturas internas. Algumas estruturas de dados usadas em modelagem de sólidos *non-manifolds* [22,31,32,75] representam, de uma forma geral, as relações de adjacências dos objetos tridimensionais não necessariamente homogêneos no espaço. Na presente implementação da CGC, foi adotada a estrutura de dados *Radial-Edge* (RED), proposta por Weiler [74,75].

Essa estrutura de dados é conhecida como *Radial-Edge* porque ela armazena explicitamente a lista de faces ordenadas radialmente ao redor de uma aresta (Figura

3.2). A estrutura de dados *Radial-Edge* foi concebida para modelagem *non-manifold* e Weiler provou sua completitude, o que significa que todas as relações de adjacência entre entidades topológicas podem ser obtidas a partir dessa representação.

A fim de descrever a topologia de uma subdivisão espacial, a representação *Radial-Edge* emprega o conceito de *uso* de um elemento topológico. Um uso pode ser visto como a ocorrência de um elemento topológico em uma relação de adjacência com um elemento de dimensão superior. Assim, a estrutura RED armazena explicitamente os dois usos (lados) de uma face pelas duas regiões (não necessariamente distintas) que a compartilham. Cada uso de face é limitado por um ou mais usos de ciclo (*loop*), que por sua vez são formados por uma seqüência alternada de usos de aresta e usos de vértices. Um ciclo, neste contexto, é uma porção conexa da fronteira de uma face e é formado por uma seqüência alternada de vértices e arestas.

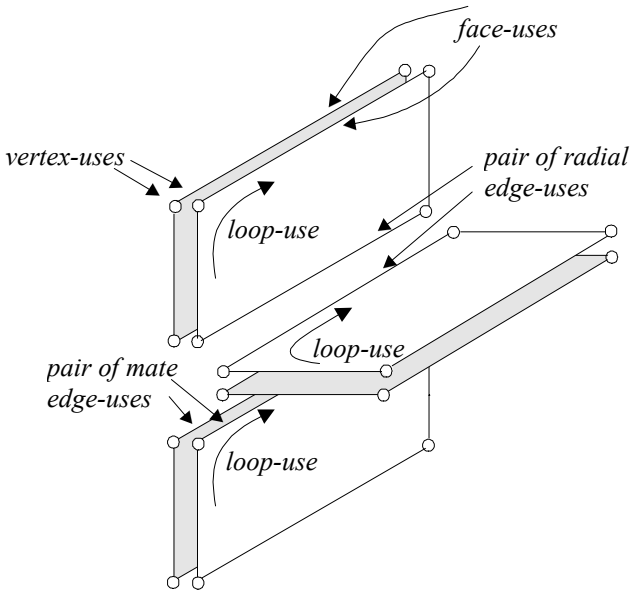


Figura 3.2 – Usos dos elementos topológicos na RED.

A estrutura RED é uma descrição hierárquica de uma subdivisão espacial, iniciando no nível mais alto (regiões), e terminando nos níveis mais baixo (vértices) (Figura 3.3). Os elementos topológicos são mantidos em listas duplamente encadeadas e têm ponteiros para seus atributos.

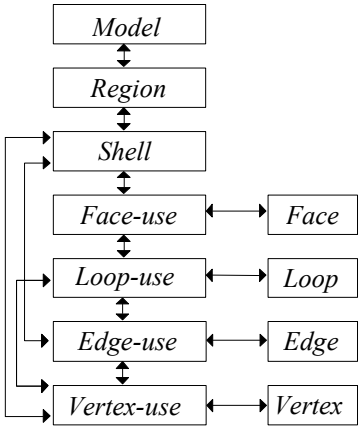


Figura 3.3 – Hierarquia das entidades topológicas da RED.

Estrutura de dados topológicas são complexas e não devem ser manipuladas diretamente. Weiler introduziu um conjunto de operadores que são métodos de alto nível para acessar a estrutura *Radial-Edge*. Esses operadores são divididos em dois grupos. O primeiro grupo tem operadores que atuam sobre as faces de uma subdivisão espacial e são análogos aos operadores *manifolds* apresentados por Mäntylä [38]. O segundo grupo tem operadores que são capazes de criar arestas e adicionar faces. Esses são os operadores *non-manifolds*. Considerações sobre um conjunto mínimo de operadores podem ser encontrados no trabalho de Wu [69].

A presente representação CGC é implementada como uma biblioteca de classes C++. Essas classes provêm um conjunto de operadores de alto nível que manipulam uma subdivisão espacial. Esses operadores recebem como dados de entrada informações geométricas dos retalhos das superfícies que são inseridas na subdivisão espacial. Estas informações geométricas são automaticamente transferidas para as informações topológicas requeridas pelos operadores *manifolds* e *non-manifolds* de Weiler.

A capacidade de modelagem da CGC é um dos aspectos chaves do esquema de modelagem de elementos finitos proposto neste trabalho. Entretanto, a implementação original desta representação apenas considera retalhos de superfícies planares. Uma das tarefas deste trabalho é estender a metodologia para considerar superfícies com geometrias curvas usando NURBS. Detalhes sobre essa metodologia serão mostrados mais adiante.

3.2 – Enfoque de Modelagem Híbrida

Como foi dito anteriormente, o esquema de representação dos dados proposto, e que é adotado na nova versão do modelador MG, é baseado em um enfoque híbrido. A idéia básica é ter duas representações separadas do mesmo modelo. Uma representação é armazenada na estrutura de dados do modelador. Esta é a representação que é mantida na memória do computador durante o processo de modelagem. A estrutura de dados do modelador é também armazenada permanentemente em disco quando o modelo é salvo. A outra representação é uma conversão temporária da estrutura de dados do modelador em uma representação CGC. Nessa conversão, a topologia do modelo é determinada tal que diferentes regiões possam ser reconhecidas. Em um estágio intermediário dessa conversão, o algoritmo para interseção de superfícies é usado para calcular as curvas e retalhos de superfícies resultantes da interseção. As entidades topológicas na estrutura de dados do modelador também são atualizadas para refletir possíveis mudanças da interseção de superfícies.

A principal vantagem desta abordagem é que ela une a capacidade de geração de malhas e interseção do MG com a poderosa representação topológica e robustez da CGC. Existe uma comunicação nos dois sentidos entre as duas representações, como pode ser visto na Figura 3.4. A CGC pode ser vista como um "costurador topológico" que gera informações de adjacência de um modelo que é consistente com a sua geometria. As entidades topológicas identificadas pela representação CGC são passadas de volta para a representação MG, incluindo as regiões detectadas automaticamente. Na Figura 3.4, pode-se observar também que a descrição geométrica das entidades topológicas é comum para ambas as representações. Essa descrição geométrica é armazenada em um módulo separado baseado na representação NURBS [54].

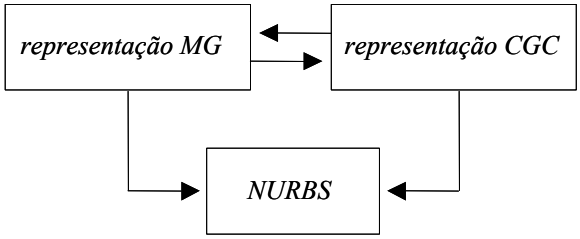


Figura 3.4 – Módulos gerais na representação do enfoque de modelagem híbrido proposto.

Os três módulos mostrados na Figura 3.4 são implementados como classes POO. A organização das classes CGC é descrita por Cavalcanti [15] e a organização das classes NURBS pode ser encontrada em [30]. A organização de classes do modelador MG é descrita na próxima seção.

Os passos a seguir resumem a metodologia adotada.

Metodologia de Modelagem Híbrida

(a) O usuário gera retalhos de superfícies simples que podem se interceptar;

(b) O modelador MG determina as interseções usando o algoritmo para interseção de superfícies, gerando novas curvas, segmentos de curvas e retalhos de superfícies;

(c) O usuário seleciona os retalhos de superfícies que irão ser usados no modelo final, removendo partes indesejáveis;

(d) O módulo CGC identifica as regiões fechadas e retorna essas informações para a estrutura de dados do MG;

(e) O modelador MG calcula a malha final combinando as malhas dos retalhos e sólidos individuais.

As interseções de superfícies são realizadas antes do reconhecimento de regiões. Novos segmentos de curvas e retalhos de superfícies podem ser criados neste passo. A informação básica passada da estrutura de dados do modelador para a estrutura de dados da CGC consiste de um conjunto de retalhos de superfícies definidos pelo usuário através da interface gráfica do MG ou resultantes da interseção de superfícies. Para cada

retalho de superfície inserido na representação CGC, uma face topológica na *Radial-Edge* é gerada. Na representação CGC, os retalhos de superfícies apenas se interceptam entre si nas suas fronteiras. Aqui, um retalho de superfície é representado pelas suas curvas do contorno e pelas suas descrições geométricas NURBS. Esses parâmetros são suficientes para determinar um retalho de superfície como entrada de dados para a representação CGC.

Esse módulo processa os retalhos de superfícies passados pelo MG e gera um modelo topológico consistente com detecção de multi-regiões. Esse novo modelo é então convertido de volta para a representação MG, atualizando todas as relações topológicas na estrutura de dados do MG. Essa conversão é feita, basicamente, percorrendo todas as regiões e faces geradas pelo módulo CGC e transformando-as em entidades da representação MG. Cada face na representação CGC corresponde a um retalho de superfície na representação MG. Dois retalhos podem está sobre uma mesma superfície geométrica, em situações quando uma curva de interseção subdivide o retalho de superfície inicial. Analogamente, cada aresta na representação CGC corresponde a um segmento de curva na representação do modelador, e esses segmentos podem pertencer à mesma curva geométrica se eles forem originados por uma divisão da curva. Finalmente, cada região na CGC gera um sólido na representação MG.

3.3 – Organização das Classes do Modelador

A estrutura de dados principal do MG foi concebida com o objetivo de oferecer suporte à modelagem de superfícies (cascas) e sólidos para elementos finitos. Neste trabalho, a estrutura de dados do MG é reorganizada para considerar a distinção entre entidades topológicas e geométricas, que não é feito na sua versão original. O esquema *Radial-Edge* poderia ser adotado na reorganização dessa estrutura de dados. Entretanto, isso requereria que a consistência entre geometria e topologia fosse forçada em cada etapa da modelagem. Além disso, a organização de classes da estrutura de dados do MG é mais simples e leve que a estrutura de dados *Radial-Edge*. Apesar disso, a estrutura de dados do MG mantém as adjacências requeridas entre as entidades topológicas necessárias para representar um modelo *non-manifold* com multi-regiões. A Figura 3.5

mostra a organização de classes POO do módulo MG. O diagrama de classes que é mostrado nesta figura, bem como em outras figuras deste trabalho, segue a nomenclatura da OMT (*Object Modeling Technique*) [62].

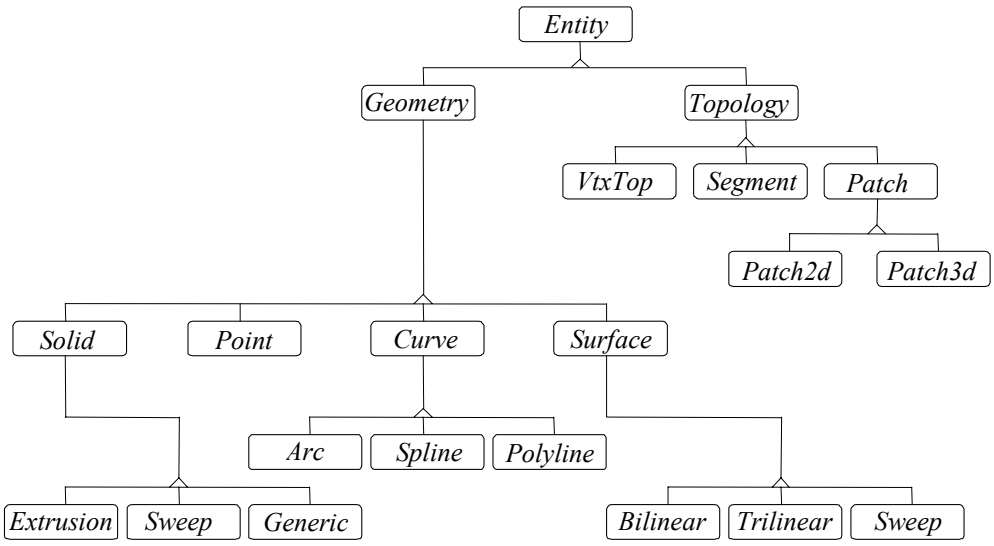


Figura 3.5 – Organização geral das classes do modelador MG.

A classe *Entity* é subdividida em duas subclasses. A primeira, denominada classe *Geometry*, refere-se às entidades geométricas do modelo. A outra, chamada classe *Topology*, representa as entidades que contêm as informações topológicas.

Na estrutura de dados do MG, um vértice no espaço tem uma instância topológica, representada por um objeto da classe *VtxTop*, e uma instância geométrica, representada por um objeto da classe *Point*. Um objeto da classe *VtxTop* contém uma lista de referências para os segmentos de curvas adjacentes (objetos *Segment*) e uma referência para o objeto *Point* correspondente. Como será visto mais adiante, um objeto da classe *VtxTop* também possui uma lista de *usos* associados a ele. Por sua vez, um objeto da classe *Point* armazena a posição geométrica de um vértice no espaço Euclidiano 3D e tem uma de referência para o objeto *VtxTop* correspondente. As Figuras 3.6 e 3.7 mostram as relações geométricas e topológicas dos objetos das classes *VtxTop* e *Point*, respectivamente.

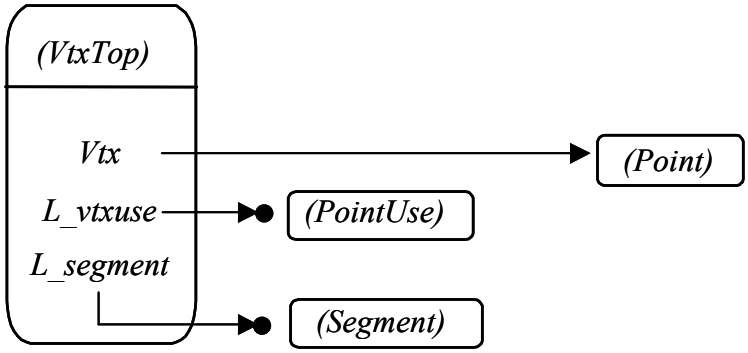


Figura 3.6 – Relações dos objetos da classe *VtxTop*.

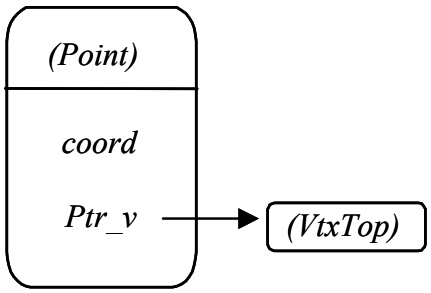


Figura 3.7 – Relações dos objetos da classe *Point*.

Analogamente, um objeto *Segment* representa a topologia de uma porção (segmento) de uma curva geométrica. Um objeto desta classe armazena as informações topológicas (por exemplo, ponteiros para os vértices adjacentes) e uma referência para sua descrição geométrica (objeto *Curve*). A Figura 3.8 ilustra as relações entre uma curva (geometria) e seus dois segmentos (topologia). Nesse caso, a curva *c1* tem uma referência para a sua representação geométrica (um objeto NURBS) e uma lista de referências para objetos *Segment* que estão ao longo dela (*s1* e *s2*). Estes objetos *Segment* contêm referências para seus vértices extremos (*v1*, *v3*) e (*v3*, *v2*), respectivamente, e para a curva *c1*. Um objeto *Curve* deve ter pelo menos um objeto *Segment*. As Figuras 3.9 e 3.10 mostram as relações geométricas e topológicas dos objetos das classes *Segment* e *Curve*, respectivamente. Na Figura 3.10 os parâmetros *t0* e *t1* representam os limites extremos, no espaço paramétrico da curva, dos objetos da classe *Segment*, e a variável *L_seguse*, descrito mais adiante, representa a lista de *usos* associados ao segmento correspondente.

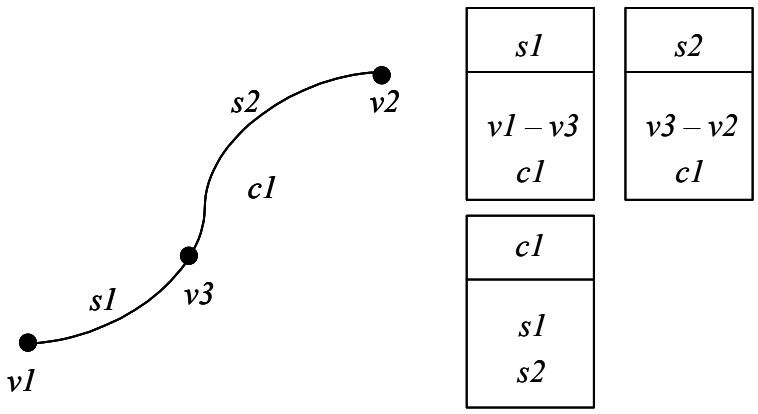


Figura 3.8 – Relações *Curve-Segment*.

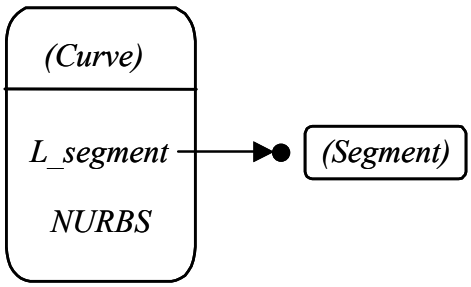


Figura 3.9 – Relações dos objetos da classe *Curve*.

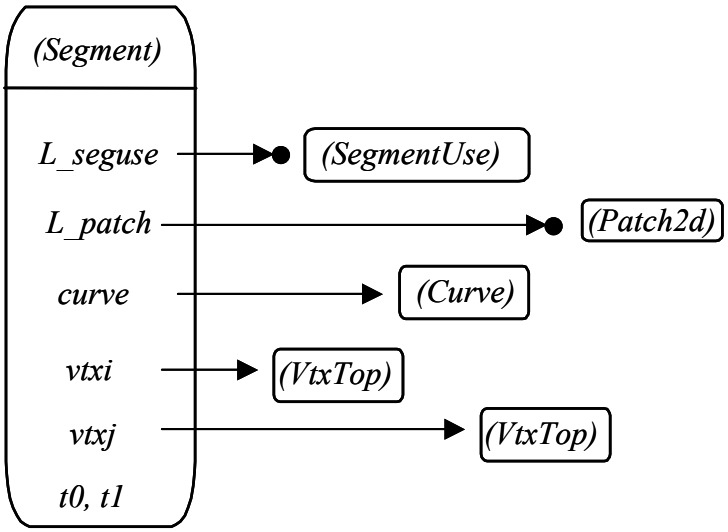


Figura 3.10 – Relações dos objetos da classe *Segment*.

Outra importante classe no contexto da estrutura de dados do MG é a classe *Patch2d*, que representa a topologia de uma porção de uma superfície. Um objeto da classe *Patch2d* armazena referências para suas curvas do contorno e para a superfície que descreve a sua geometria, a qual é representado pela classe *Surface*. Um objeto *Surface*

contém uma referência para a sua representação geométrica, ou seja, um ponteiro para um objeto NURBS, e uma lista de referências para objetos *Patch2d* que estão ao longo dele. A Figura 3.11 ilustra as relações entre a superfície *s1* e dois objetos *Patch2d* correspondentes, *p1* e *p2*. Cada objeto *Patch2d* contém informações topológicas, que consistem principalmente de uma lista de objetos *Segment* no seu contorno. Além disso, um objeto *Patch2d* armazena uma malha de elementos finitos de superfície que eventualmente é gerada sobre ela.

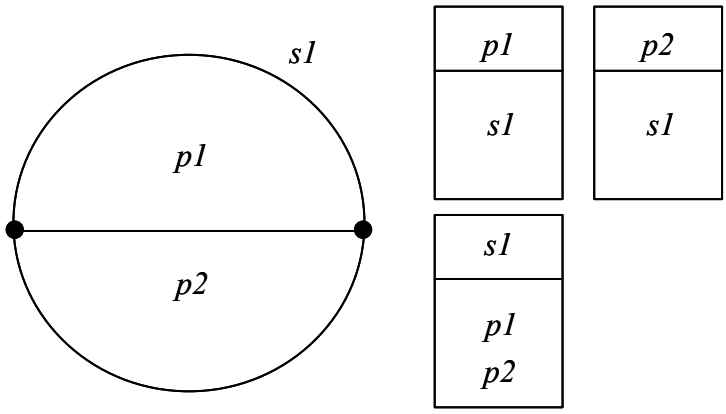


Figura 3.11 – Relações *Surface-Patch2d*.

Um objeto da classe *Surface* deve ter pelo menos um objeto *Patch2d* associado a ele. Um objeto *Surface* também contém informações geométricas relacionados ao processo que foi utilizado na sua geração iterativa. Atualmente, uma superfície pode ser gerada pelas curvas do contorno de um mapeamento *bilinear*, mapeamento *trilinear* ou *sweep* [19]. Então, cada objeto *Surface* contém informações sobre seus métodos de criação e uma lista de referências das suas curvas geradoras (curvas que foram usadas na sua criação). A Figura 3.12 mostra as relações geométricas e topológicas dos objetos da classe *Surface*.

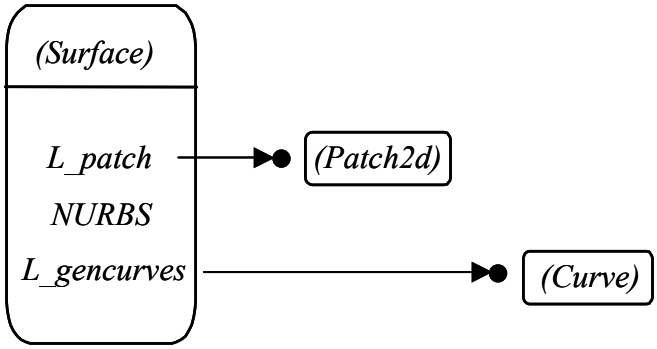


Figura 3.12 – Relações dos objetos da classe *Surface*.

Um objeto *Patch2d* também tem referência para objetos *Patch3d* adjacentes. Um objeto da classe *Patch3d* é a representação topológica de uma região sólida. Um objeto *Patch3d* tem uma lista de referências para os objetos *Patch2d* que formam o seu contorno e um ponteiro para o objeto *Solid* correspondente. Além disso, um objeto *Patch3d* armazena um malha de elementos finitos que eventualmente pode ser gerada sobre ele. As relações topológicas e geométricas dos objetos da classe *Patch2d* podem ser vistas na Figura 3.13. Já a Figura 3.14 mostra as relações geométricas e topológicas referentes aos objetos da classe *Patch3d*. Nessas Figuras, as variáveis *mesh* e *vmesh* representam ponteiros para as malhas de superfície e volumétrica associadas aos objetos das classes *Patch2d* e *Patch3d*, respectivamente.

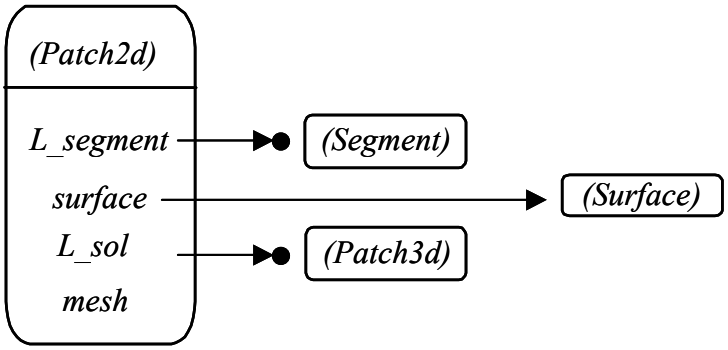


Figura 3.13 – Relações dos objetos da classe *Patch2d*.

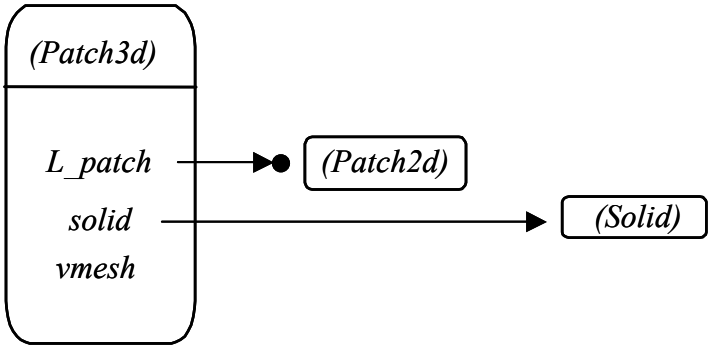


Figura 3.14 – Relações dos objetos da classe *Patch3d*.

A versão geométrica da classe topológica *Patch3d* é a classe *Solid*. Um objeto *Solid* (ver Figura 3.15) contém informações geométricas sobre o método que foi utilizado na sua geração: extrusão, *sweep* ou geração genérica [43]. Uma geração genérica de um sólido pode ser realizada explicitamente pelo usuário (pela simples seleção dos retalhos de superfícies do sólido desejado) ou automaticamente pela representação CGC quando uma região sólida é detectada. Então, cada objeto *Solid* contém informações sobre o seu método de criação e uma lista de referências para suas curvas e superfícies geradoras (curvas e superfícies que foram usadas na sua criação).

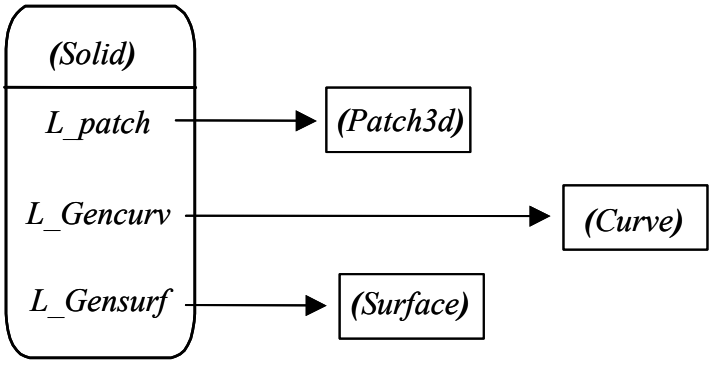


Figura 3.15 – Relações dos objetos da classe *Solid*.

Um importante assunto que é tratado no presente esquema de representação dos dados é o suporte para geração automática de malhas. Como mencionado anteriormente, geração de malhas de elementos finitos é uma área de pesquisa ativa e tem sido estudada pelo grupo de pesquisa ao qual este trabalho está inserido [14,20,43,44]. Um dos objetivos dessa linha de pesquisa é o desenvolvimento de um sistema completo para modelagem geométrica e simulação adaptativa 3D usando o método dos elementos finitos. A metodologia utilizada na geração de malhas adaptativas é baseada no refinamento de

cada entidade topológica no seu espaço paramétrico. Por exemplo, geração de malhas em um retalho de superfície (objeto *Patch2d*) é baseado em um refinamento anterior dos segmentos do seu contorno no espaço paramétrico da superfície. Da mesma forma, geração de malhas sólidas em uma região 3D requer o refinamento anterior dos seus retalhos de superfícies do contorno.

A implementação dessa metodologia para geração de malhas é acoplada na presente estrutura de dados através do conceito de *uso* de uma entidade topológica por uma superfície. O conceito de *uso* na estrutura de dados do MG é diferente daquele apresentada na estrutura de dados *Radial-Edge*. No MG, *uso* simplesmente significa a informação geométrica de uma determinada entidade no espaço paramétrico de uma superfície, enquanto que na *Radial-Edge usos* são responsáveis pelas principiais relações topológicas. Na estrutura de dados do MG, as entidades topológicas descritas acima prezam as relações de adjacência e mantém entidades *usos* para armazenar as coordenadas paramétricas. Por exemplo, um vértice que está sobre duas superfícies adjacentes tem dois *usos*. A Figura 3.16 ilustra este exemplo. O *uso* do vértice topológico *v1* refere-se à superfície *s1*, com coordenadas paramétricas *u=0* e *v=0*. O vértice topológico *v5* tem dois *usos*, um em *s1*, com coordenadas paramétricas *u=1* e *v=1*, e outro em *s2*, com coordenadas paramétricas *u=1* e *v=0*. A mesma idéia é adotada para os segmentos. Nesse caso, cada *uso* de um segmento tem as coordenadas paramétricas dos seus pontos extremos.

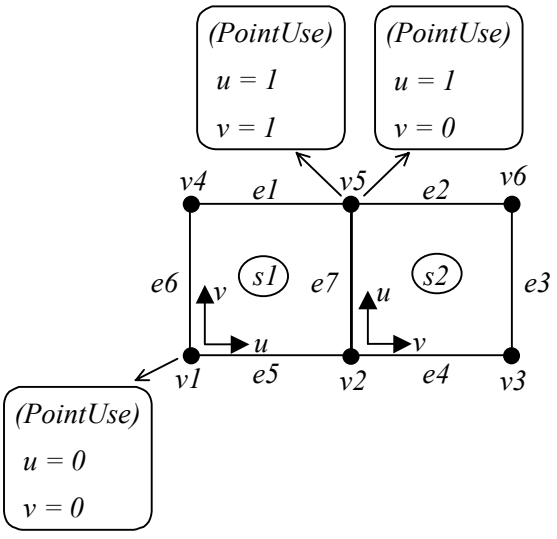


Figura 3.16 – Usos de vértices em duas superfícies adjacentes.

Então, um objeto *VtxTop* tem uma lista de *usos* associados a ele além das outras informações topológicas descritas anteriormente. O *uso* de um vértice é um objeto da classe *PointUse*. Um objeto *PointUse* tem uma referência para o seu objeto *Point* correspondente e uma outra referência para o objeto *Surface* correspondente. Um objeto *PointUse* também contém duas variáveis para armazenar as coordenadas paramétricas de um ponto em uma superfície. As relações dessa classe podem ser vistas na Figura 3.17. Aqui, as variáveis *u* e *v* são as coordenadas paramétricas do ponto na superfície correspondente.

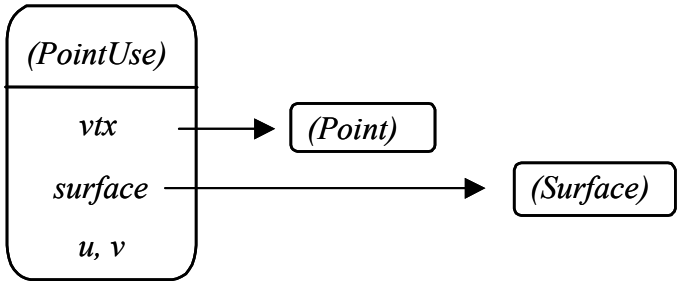


Figura 3.17 – Relações dos objetos da classe *PointUse*.

Da mesma forma, um objeto *Segment* tem uma lista de *usos*, que são objetos da classe *SegmentUse*. Um objeto *SegmentUse* (Figura 3.18) tem uma referência para o objeto *Segment* correspondente e uma outra referência para o objeto *Surface* correspondente. Um objeto *SegmentUse* também contém variáveis que representam as coordenadas paramétricas de um segmento de curva em uma superfície. Na Figura 3.18, a variável *vec_uv* representa o vetor com as coordenadas paramétricas do segmento na superfície correspondente.

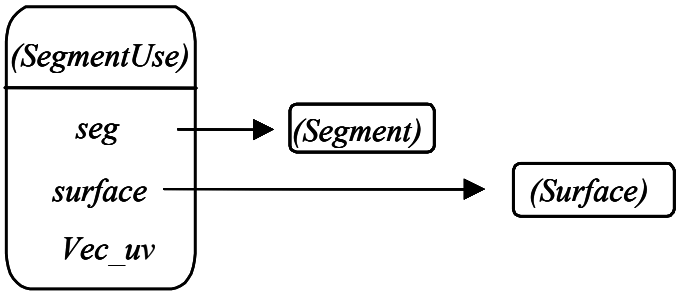


Figura 3.18 – Relações dos objetos da classe *SegmentUse*.

3.4 – Implementação de NURBS na Representação CGC

A implementação original da CGC trata apenas retalhos de superfícies planares (objetos poliedrais). No entanto, um dos objetivos deste trabalho é considerar modelos CGC com geometria curvas, o qual é alcançado com a incorporação de geometria NURBS.

O módulo CGC neste trabalho é usando justamente para detecção de regiões e recebe como dados de entrada retalhos de superfícies que apenas se interceptam entre si nos seus contornos. Conseqüentemente, a extensão do módulo CGC foi relativamente simples, porque a estrutura de dados CGC não depende da geometria considerada, ou seja, não existe diferença, por exemplo, se uma aresta é considerada como uma linha reta ou uma curva. Da mesma forma, uma face pode ser indistintamente tratada como plana (polígono) ou curva.

Na representação CGC, objetos responsáveis pela definição das entidades topológicas (*regiões, faces, arestas* e *vértices*) apontam para objetos que descrevem a geometria e os atributos associados a elas (objetos das classes *RegionAttr, FaceAttr, EdgeAttr* e *VertexAttr*). Além disso, a CGC implementa uma outra classe que concentra todos os algoritmos geométricos. Essa classe, denominada *GeomPck*, ajuda na construção de um modelo geométrico, calculando o volume de uma certa região, a interseção entre uma superfície e uma curva, a localização de um ponto com relação a uma região, etc. Os métodos dessa classe tratam das informações geométricas associadas a cada entidade topológica. Então, as modificações causadas pela adição de um novo tipo de geometria na implementação CGC são limitadas a um pequeno grupo de classes que definem e manipulam a geometria do modelo.

Os objetos *FaceAttr* e *EdgeAttr* são aqueles que contêm as descrições geométricas das *faces* e *arestas*. Essa descrição é agora armazenada em duas novas classes, como pode ser visto na Figura 3.19. A classe *Surface* define a representação geométrica das superfícies do modelo, enquanto que a classe *Curve* define a geometria das curvas. Ambos objetos *Curve* e *Surface* têm seus próprios espaços paramétricos. Também existem subclasses específicas para certos tipos de geometria (por exemplo, a classe *Arc* descreve geometricamente um arco de círculo e a classe *Gordon*, uma superfície de

Gordon). Os métodos nessas classes manipulam as informações geométricas correspondentes. Por exemplo, existe um método que a partir das coordenadas paramétricas em uma superfície, obtêm as coordenadas Cartesianas do ponto correspondente.

A representação geométrica de curvas e superfícies no modelador MG usa a biblioteca NURBS++, no contexto da POO, de domínio público [30] e baseada em NURBS. Essa biblioteca tem a capacidade de representar vários tipos de curvas e superfícies, incluindo as curvas cônicas e superfícies quádricas.

A conexão entre as representações MG e CGC e a biblioteca NURBS++ é simples. Os objetos *Curve* e *Surface* contêm referências para seus correspondentes objetos NURBS, como pode ser visto na Figura 3.20.

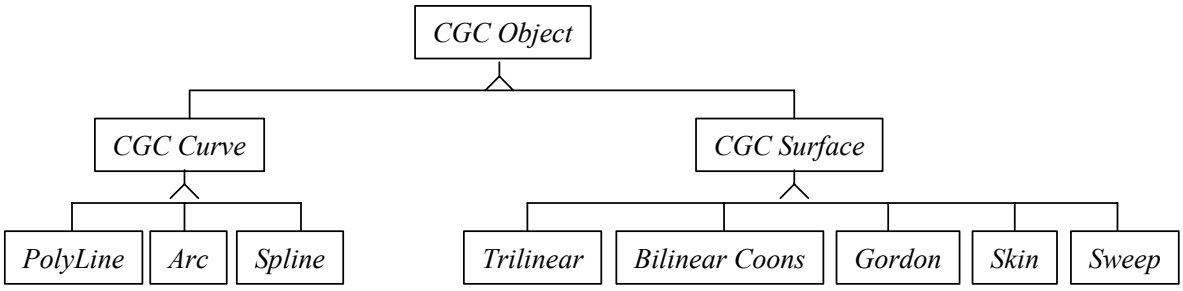


Figura 3.19 – Novas classes implementadas na representação CGC.

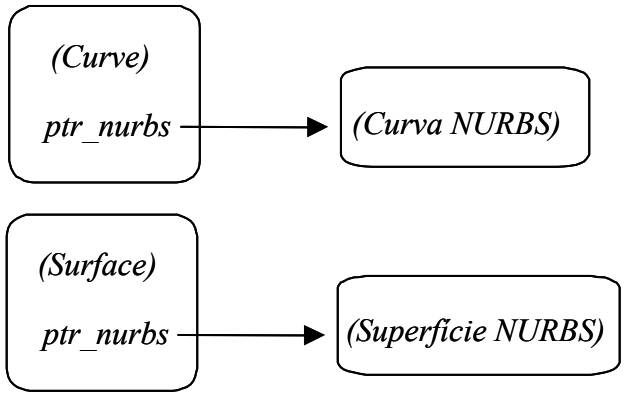


Figura 3.20 – Relações entre as classes *Curve* e *Surface* da representação CGC com a biblioteca NURBS++.

4.Interseção de Superfícies

Uma forma de se construir modelos complexos consiste em combinar várias superfícies paramétricas construídas individualmente em uma única representação. Isso pode ser feito recortando as regiões excedentes determinadas pelas interseções. Se essa técnica de modelagem é adotada, então o problema de interseção entre dois retalhos de superfícies deve ser resolvido de forma eficiente e robusta.

Poucas interseções de superfícies paramétricas podem ser calculadas analiticamente. Mesmo quando isso é possível, pode não ser prático. Portanto, a maioria dos problemas que envolvem interseções entre superfícies é solucionada usando-se técnicas numéricas.

Os métodos para solução do problema de interseção pertencem a duas classes [4]: métodos de *marcha* e *subdivisão*. Os métodos de *marcha*, também chamados de *continuação*, determinam as curvas de interseção no espaço tridimensional do objeto pela marcha na direção do seu vetor tangente [2,4,5,67]. Os métodos de *subdivisão* ou *decomposição* determinam as curvas de *trimming*[3] no espaço paramétrico bidimensional de cada uma das superfícies, por subdivisão recursiva em cada passo [29].

Os problemas considerados neste trabalho tratam não só de questões relacionadas à modelagem, mas também da definição das malhas sobre cada uma das superfícies do modelo. Dessa forma, o problema de interseção entre superfícies também envolve o problema de reconstrução de *malhas*. Assim o objetivo é não só determinar as curvas geradas pela interseção entre superfícies, mas também as novas malhas definidas sobre essas superfícies.

Várias soluções para o problema de interseção entre superfícies funcionam relativamente bem em muitos casos, mas não tratam o problema de interseção de malhas

[3] curvas resultantes da interseção entre superfícies com representações nos seus espaços paramétricos e no espaço tridimensional.

como descrito acima. Uma exceção é o trabalho de Lo [35]. Esse trabalho apresenta um algoritmo simples para interseção de malhas triangulares que redefinem automaticamente as faces, adaptando-as às curvas de interseção encontradas. Essa solução não usa a representação contínua da superfície, e então pode ser usada mais facilmente em sistemas de modelagem. Por outro lado, em regiões com grandes curvaturas, a determinação dos pontos de interseção pode não se localizar sobre a superfície original, o que não é interessante para a modelagem. Então, é desejável que tanto a parametrização das superfícies quanto as malhas sejam consideradas em um algoritmo para o cálculo de interseções. Mais ainda, as discretizações no contorno das superfícies devem ser compatíveis, para que as malhas localizadas em superfícies adjacentes sejam compatíveis e consistentes entre si.

Baseado nessas idéias e no trabalho proposto por Lo, Coelho [20] apresentou um algoritmo que resolve o problema de interseção entre malhas de superfícies. As buscas requeridas para a determinação das curvas de interseção e a reconstrução das malhas são suportadas por uma estrutura de dados topológica cujas principais características são a simplicidade e o armazenamento das entidades topológicas em estruturas espaciais *B-trees* [21] e *R*-trees* [7], em vez de listas encadeadas. O uso dessas estruturas espaciais aumenta a eficiência do algoritmo.

Esse algoritmo foi incorporado à versão original do modelado MG, sendo bastante utilizado em modelagens de cascas. O algoritmo resolve diversos tipos de problemas relacionados a interseções entre superfícies, como pode ser visto na Figura 4.1. No entanto, existem casos patológicos que não são tratados pela implementação do algoritmo, tais como a interseção entre superfícies onde uma delas já foi interceptada anteriormente e a nova curva de *trimming* intercepta uma já existente (ver Figura 4.2). São casos que aparentemente raros, mas que podem surgir na modelagem de problemas mais complexos de engenharia.

Baseado nestes aspectos, a contribuição original deste trabalho, em relação ao algoritmo de interseção de superfícies apresentado por Coelho, é o tratamento desses casos especiais não considerados na implementação da sua versão original, além do suporte consistente que a nova estrutura de dados apresentada no capítulo anterior dá as subdivisões que podem ocorrer. Deve-se notar que a metodologia apresentada pela

versão original do algoritmo foi mantida integralmente. Isso significa que as idéias centrais apresentadas por Coelho não foram alteradas.

Neste capítulo, as primeiras seções fazem um resumo do algoritmo de interseção de superfícies apresentado por Coelho. As últimas seções tratam das contribuições deste trabalho em relação ao algoritmo de interseção. Os detalhes sobre a versão original podem ser encontrados em [19,20].

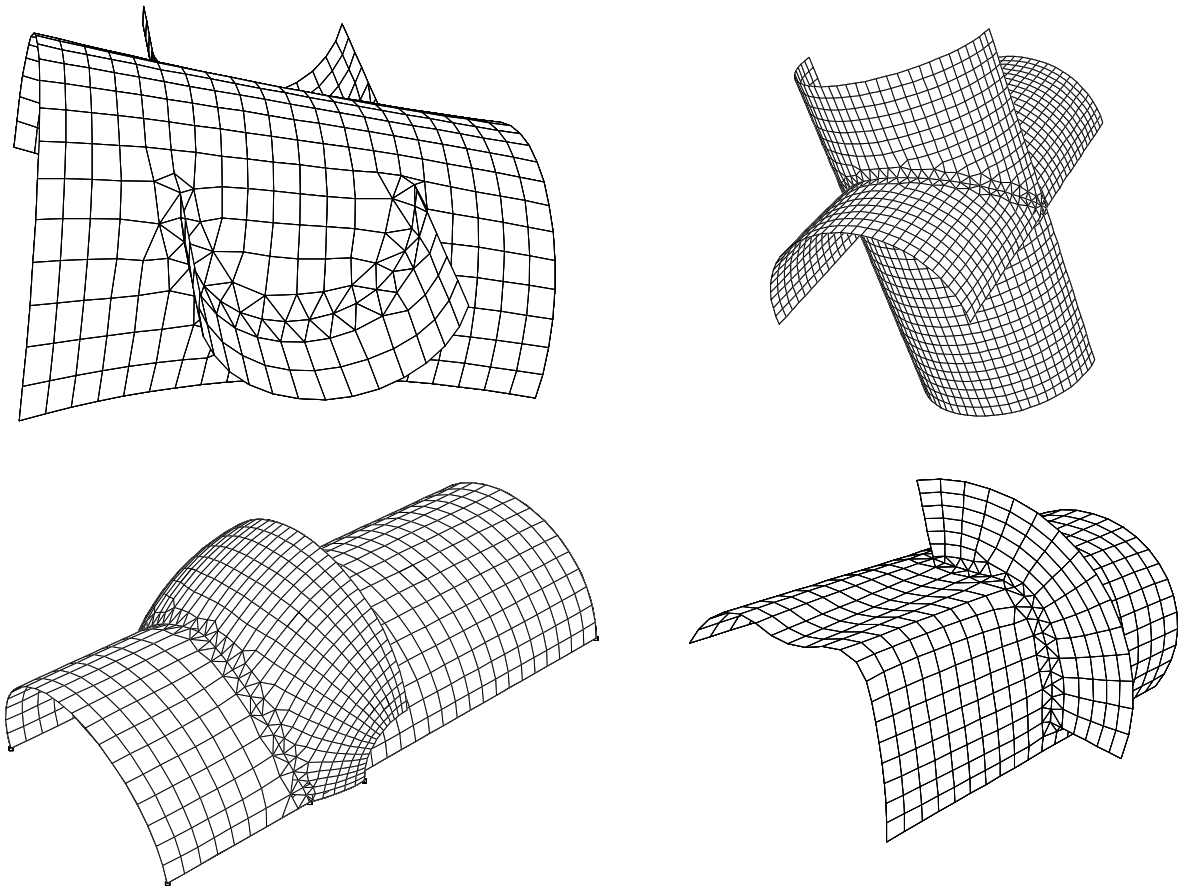


Figura 4.1 – Exemplos de interseções entre superfícies.

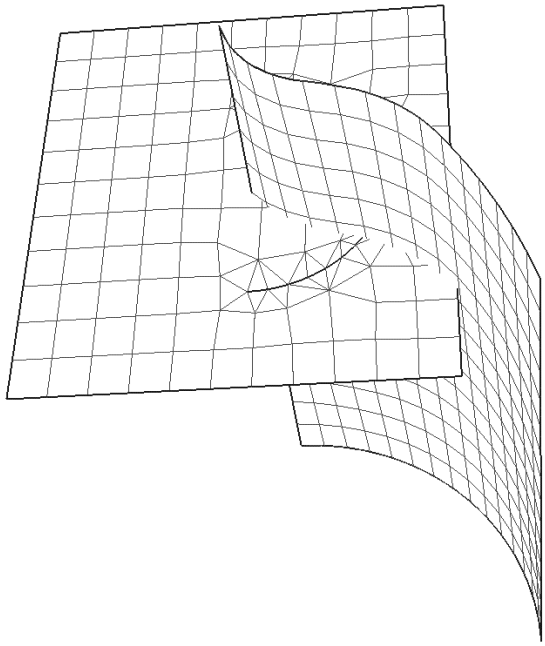


Figura 4.2 – Interseção entre superfícies: um caso especial não tratado na versão original do algoritmo implementado por Coelho [19].

4.1 – O Problema de Interseção entre Superfícies

Idealmente, uma boa solução para o problema de interseção entre superfícies usando as suas malhas de elementos finitos deve satisfazer os seguintes critérios:

(a) *Exatidão*: As curvas de interseção calculadas devem estar sobre as superfícies. Se as curvas de interseção estão apenas sobre as faces das malhas mas não estão sobre as superfícies, então a análise por elementos finitos pode fornecer resultados inaceitáveis. A forma mais fácil de garantir a *exatidão* é calcular as curvas de *trimming* no espaço paramétrico e então mapeá-las para o objeto espacial.
(b) *Fidelidade*: A geometria da superfície resultante deve refletir, fielmente, a geometria das superfícies originais, porque elas representam a intenção do responsável pela modelagem. Em particular, novos retalhos paramétricos não devem ser definidos usando as curvas de *trimming* como curvas do contorno, porque isso poderia gerar uma geometria diferente.
(c) *Precisão*: Pontos no espaço paramétrico correspondendo ao mesmo ponto de interseção devem, quando mapeados para 3D, estar a uma distância tão pequena quanto

uma tolerância selecionada pelo usuário (usualmente, uma fração do tamanho do menor elemento na malha original).

(d) *Automatismo*: As duas malhas devem ser automaticamente redefinidas ao incorporar as curvas de interseção e possíveis regiões de *trimming*. Não deve haver necessidade de intervenção manual do usuário nessa redefinição.

(e) *Qualidade*: Os elementos gerados durante a reconstrução das malhas devem ter boas formas geométricas e tamanhos médios semelhantes. Quando as malhas originais têm elementos com tamanhos muitos diferentes, a malha resultante pode ter elementos distorcidos e pontiagudos, se nenhuma correção for realizada.

(f) *Localidade*: Apenas elementos sobre a região de interseção devem ser modificados durante a reconstrução da malha. Elementos fora dessa região devem permanecer inalterados. Além disso, se uma suavização é realizada para melhorar a qualidade da malha, então ela também deve ser local.

(g) *Regiões*: As regiões definidas no espaço paramétrico pelas curvas de *trimming* devem ser identificadas automaticamente.

(h) *Eficiência*: O tempo e espaço requeridos para calcular a interseção devem ser lineares no número de elementos em torno da região interceptada. Algoritmos que são quadráticos no número total de elementos nas malhas são lentos para grandes malhas e não são apropriados para modelagem iterativa.

(i) *Robustez*: Um número arbitrário de curvas de interseção podem ser geradas, com diferentes geometrias, topologias e pontos de interpolação.

4.2 – O Algoritmo Utilizado

Atendendo aos itens descritos na seção acima, o algoritmo proposto por Coelho para determinar a interseção entre as malhas das superfícies *A* e *B* tem três passos básicos:

(1) *Determinação dos pontos de interseção*:

(1a) Cálculo e armazenamento das interseções das arestas em *A* contra as faces em *B*; e

(1b) Cálculo e armazenamento das interseções das arestas em *B* contra as faces em *A*.

(2) *Determinação das curvas de trimming*:

(2a) Conexão dos pontos de interseção em linhas poligonais representando as curvas de *trimming*;

(2b) Interpolação das curvas paramétricas passando pelos pontos da linha poligonal;

(2c) Determinação de novos pontos com espaçamento adequado nessas curvas;

(2d) Projeção desses novos pontos em cada superfície.

(3) *Reconstrução da topologia*:

(3a) Determinação das regiões de *trimming* removendo vértices e arestas baseadas na linha poligonal;

(3b) Inserção de novas arestas sobre as curvas de *trimming* usando os novos pontos definidos no passo (2c);

(3c) Triangulação das regiões de *trimming* em ambas superfícies; e

(3d) Suavização das malhas.

Para evitar o teste de todas as arestas contra todas as faces no passo (1), as entidades topológicas são armazenadas em árvores de indexação espacial, como descrito na seção 4.2.1. Como as arestas e faces são curvas no espaço tridimensional, é necessário um procedimento numérico para determinar os pontos de interseção, como é mostrado na seção 4.2.2. No final do passo (1), arestas em uma malha estão ligadas com as faces que elas interceptam na outra malha, e vice-versa. Para cada par aresta/face, são armazenados também as coordenadas paramétricas, em ambas superfícies, do ponto de interseção.

No passo (2a), calcula-se as curvas de *trimming* no espaço paramétrico pela conexão e interpolação dos pontos de interseção determinados no passo (1). Nos passos (2b-d), calcula-se representações contínuas para as curvas de *trimming* no espaço paramétrico. Pontos igualmente espaçados são então calculados e relaxados para as superfícies originais. O passo (2) é descrito na seção 4.2.3.

No passo (3a) as regiões de *trimming* são identificadas. Essas regiões são faces da estrutura de dados topológica geradas pela eliminação de algumas arestas. No final do

passo (3a) existem, em cada superfície, tantas regiões quanto o número de porções conexas de curvas de *trimming*. O passo (3b) consiste da inserção das arestas que representam as curvas de *trimming*. Como será mostrada mais adiante, essa inserção subdivide a face de *trimming* em duas. No passo (3c), cada face de *trimming* é triangulada pela inserção de arestas, de acordo com o critério geométrico descrito na seção 4.2.4.

Finalmente, para melhorar a qualidade nas formas das faces geradas na interseção, é usada uma técnica de suavização Laplaciana padrão no espaço paramétrico (passo (3d)). Nessa técnica, as coordenadas paramétricas de cada vértice são modificadas para uma média entre as coordenadas adjacentes. Essa média é repetida várias vezes, até que a qualidade dos elementos atinja níveis desejáveis. Vértices do contorno e sobre as curvas de *trimming* não são movidos.

4.2.1 – Estrutura de Dados

As malhas associadas a cada superfície são armazenadas em uma variante da estrutura de dados DCEL (*Doubly Connected Edge List*) [56], estendida para conectar duas topologias. Além disso, as entidades topológicas (vértices, arestas e faces) são armazenadas em árvores, ao invés de listas ou vetores. Esse armazenamento reduz os tempos de buscas necessários à determinação dos pontos de interseção, construção da malha inicial e da sua reconstrução.

Usa-se *B-trees* [21] para armazenar vértices e arestas e *R*-trees* [7] para armazenar faces. Os vértices são inseridos na *B-tree* que usa as coordenadas paramétricas dos pontos para buscas em ordem lexicográfica. As arestas são consideradas linhas retas no espaço paramétrico, conectando pares de vértices. As arestas são orientadas do vértice com menor índice até o vértice com o índice maior, e são armazenadas em uma *B-tree* que tem esses índices como chave para as buscas necessárias.

As faces são inseridas em uma *R*-tree* usando a sua caixa envolvente (*bounding box*) tridimensional como chave de inserção e buscas.

Na descrição das arestas e faces existe uma variável, denominada *intersection*, que é usada para determinar as curvas de *trimming* e para reconstruir a topologia. Ela também é a chave para a conexão entre as duas estruturas de dados DCEL das superfícies.

4.2.2 – Pontos de Interseção

No passo (1) do algoritmo, as arestas em uma superfície são testadas contra as faces na outra superfície que são candidatas a interceptá-las. Isso é feito testando as *bounding boxes* para interseção. A face é procurada na *R*-tree* usando-se a *bounding box* da aresta.

Para cada par aresta/face selecionado, encontra-se o ponto de interseção resolvendo o problema de minimização $F(u,v,t) = S(u,v) - R(t)$, como pode ser visto na Figura 4.3 [20]. Aqui, o método de Newton-Raphson é utilizado para resolver esse problema de minimização. O algoritmo pára quando a distância entre o ponto da aresta com valor paramétrico *t* e o ponto da face com coordenadas *(u,v)* é menor que uma dada tolerância. Essa tolerância corresponde a uma fração do menor comprimento das arestas envolvidas na interseção. Na Figura 4.3, o parâmetro *t*, que define a variação ao longo da aresta interceptada, pode ser expresso como uma interpolação linear dada pelo segmento de reta no espaço paramétrico *(w,h)* que conecta V_0 e V_1.

Após a convergência do algoritmo de minimização, dois pares paramétricos, (u_f, v_f) na superfície da face e (u_e, v_e) na superfície da aresta, são identificados. A face da superfície interceptada que contém o ponto de interseção é determinada por testes no espaço paramétrico. Um algoritmo para localização de ponto em polígono verifica se o ponto é interno, externo, sobre a fronteira, ou sobre um vértice da face em questão. Se o ponto está fora da face nenhuma atualização é feita na estrutura de dados. Se o ponto está sobre uma aresta, além do par aresta/face, a aresta identificada pelo algoritmo também deve ser atualizada. Quando a interseção está localizada sobre um vértice da face, todas as arestas adjacentes a esse vértice devem ser atualizadas.

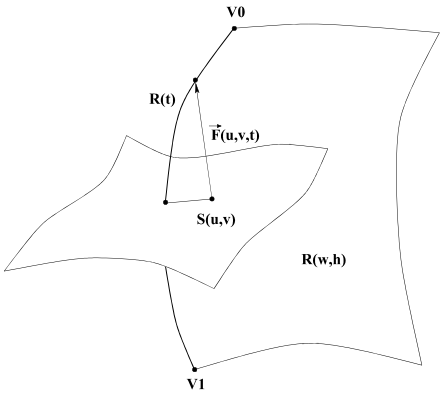


Figura 4.3 – Problema de interseção.

Para completar a determinação dos pontos de interseção e para preencher todos os campos necessários à construção das curvas de *trimming*, todos os procedimentos descritos nesta seção devem ser repetidos trocando-se as superfícies, ou seja, testando-se as arestas da segunda superfície contra as faces da primeira. No final deste passo, todas as informações topológicas necessárias para determinar as curvas de *trimming* estão armazenadas nas estruturas de dados DCEL das duas superfícies.

4.2.3 – Curvas de Interseção

A determinação das curvas de interseção é realizada percorrendo as faces interceptadas da primeira superfície, propagando-se as interseções pelas faces adjacentes a arestas interceptadas, identificadas pela variável *intersection*. Em cada ponto, referências para as arestas interceptadas em cada superfície são armazenadas. Quando todas faces interceptadas forem visitadas, todos os componentes conectados da interseção foram encontrados.

Dois tipos de linhas poligonais são usados: *poligonais locais*, que conectam os pontos de interseção dentro de uma determinada face; e *poligonais globais*, que são compostas de poligonais locais e representam uma curva de *trimming* completa. Para converter as poligonais globais em curvas de interseção contínuas, os pontos de interseção são interpolados usando *splines* cúbicas no espaço Cartesiano.

Uma vez determinadas as representações contínuas para as curvas de interseção, deve-se amostrar pontos igualmente espaçados ao longo da curva, usando uma distância que corresponde a uma fração do tamanho médio das arestas interceptadas nas duas malhas iniciais. Esses pontos amostrados definem vértices representando as curvas de interseção na malha combinada. Mesmo considerando que os pontos de interpolação estão localizados sobre ambas superfícies segundo uma dada tolerância, os pontos amostrados podem não estar. Assim, esses pontos amostrados devem ser relaxados, usando um procedimento similar ao apresentado no passo (1) (seção 4.2.2), de forma que as novas posições se aproximem das duas superfícies (passo (2d)). Como resultado desse processo, pares de coordenadas paramétricas em ambas superfícies são determinados para os pontos amostrados.

4.2.4 – Reconstrução das Topologias (Malhas)

Para encontrar as regiões de *trimming* no passo (3a), todas as arestas conectadas aos vértices definidos por cada ponto de interseção encontrado em (1) são removidas. As arestas pertencentes a um contorno ou a uma curva de *trimming* não podem ser removidas. Essas arestas são subdivididas pela inserção de um novo vértice na posição de cruzamento. A Figura 4.4 [19] mostra um exemplo com as faces de *trimming* identificadas pela remoção e subdivisão de arestas.

No passo (3b), as arestas representando as curvas de *trimming* são inseridas, dividindo necessariamente as faces em duas: uma do lado direito, e outra do lado esquerdo de cada curva. Em situações onde a face de *trimming* não é subdividida por essa inserção de arestas (ver Figura 4.4b), os vértices extremos das curvas de *trimming* são unidos a vértices localizados no contorno da face de *trimming*, forçando a subdivisão dessas faces. A busca dos vértices no contorno é feita por proximidade.

No passo (3c), as regiões de *trimming* são trianguladas pela inserção de arestas atendendo os seguintes critérios geométricos:

- *Consistência*: arestas devem estar completamente contidas nas região paramétrica limitada pelas arestas, sem outras interseções;
- *Proximidade*: arestas devem conectar dois vértices que estão próximos um ao outro;
- *Curvatura*: a área do setor formado pela corda e pela curva que segue a superfície em 3D é a menor entre todas as outras;
- *Delaunay*: arestas devem obedecer ao critério de Delaunay, onde elas são projetadas em um plano cujo vetor normal é a média dos vetores das faces adjacentes.

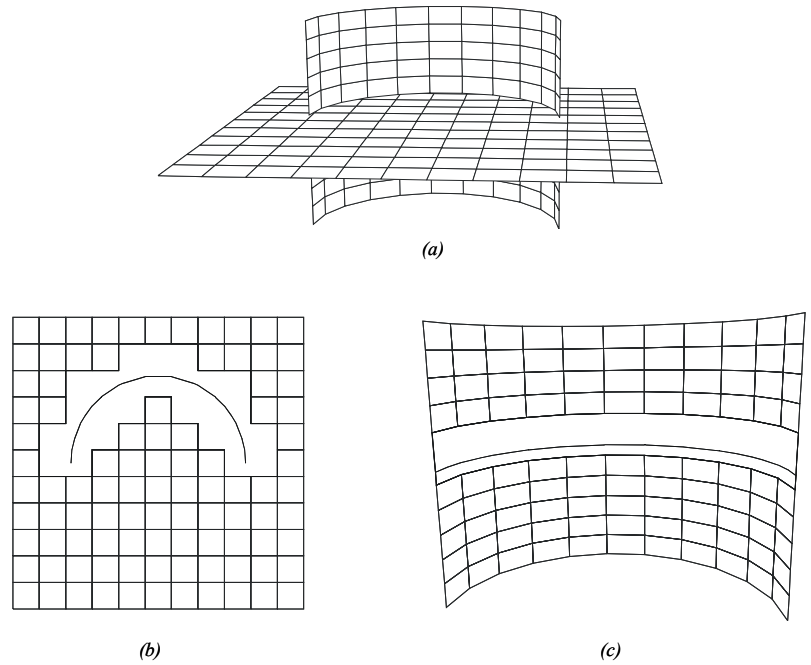


Figura 4.4 – Faces de *trimming* em um exemplo de interseção: a) Interseção visual; b) e c) Faces e curvas de *trimming*.

4.3 – Casos Especiais

O algoritmo proposto por Coelho para interseção de superfícies, e descrito nas seções anteriores, pode ser aplicado em uma grande quantidade de situações que aparecem na modelagem de problemas de engenharia. No entanto, alguns casos patológicos não eram tratados de forma eficiente na implementação original desse algoritmo. A idéia desta seção é descrever alterações realizadas nessa implementação com o objetivo de fornecer

mais robustez e confiabilidade ao algoritmo apresentado por Coelho. Deve-se observar que as idéias centrais do algoritmo permanecem praticamente inalteradas. As principais modificações no algoritmo estão relacionadas com a sua implementação.

4.3.1 – Interseções Aresta/Aresta e Aresta/Vértice

O primeiro caso especial tratado neste trabalho é mostrado na Figura 4.5. Esse caso refere-se à situação onde uma aresta (*e1*) da superfície *Sa* intercepta uma aresta (*e2*) da superfície *Sb*.

No passo (1) do algoritmo de interseção, o campo *intersecion* das arestas e faces é preenchido. Assim, na aresta *e1*, *intersection* armazena apenas uma face de *Sb* (*Sb1* ou *Sb2*). Da mesma forma, na aresta *e2* esse campo armazena apenas uma face de *Sa* (*Sa1* ou *Sa2*).

No passo (2) do algoritmo, esses campos são utilizados para fazer o entrelaçamento das duas estruturas DCEL. Esse entrelaçamento é utilizado na determinação das curvas de interseção. A determinação dessas curvas é realizada percorrendo-se as faces interceptadas da superfície *Sa*, propagando as interseções pelas faces adjacentes a arestas interceptadas, identificadas pela variável *intersection*. Como cada variável *intersection* armazena uma única aresta ou face por vez, pode existir uma combinação onde não é possível dar seqüência à determinação das curvas de interseção no passo (2), ou seja, algumas arestas ou faces das estruturas DCEL podem não ser percorridas no algoritmo. Ocorrendo essa situação, não existe a garantia que o algoritmo de interseção vai ou não obter o resultado esperado.

Para resolver casos onde essas situações aparecem, é realizada uma alteração na entidade aresta da estrutura DCEL. A variável *intersection*, na aresta, não mais armazena uma única face da superfície interceptada por ela. Nesta nova implementação do algoritmo, tal variável guarda em uma lista todas as faces interceptadas por ela, evitando que determinadas faces não sejam percorridas no passo (2) do algoritmo.

Nesta nova implementação, os passos (1) e (2) do algoritmo foram modificados. No passo (1), é feito o preenchimento da lista *intersection*, enquanto que no passo (2) tal lista é utilizada na determinação das curvas de interseção.

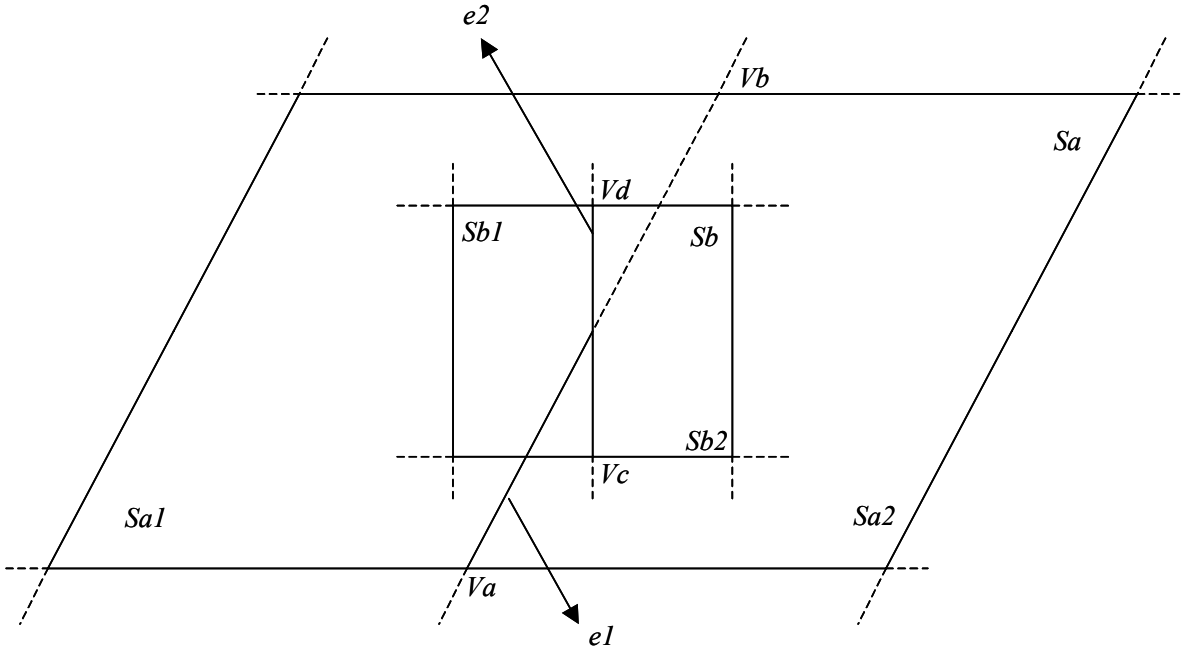


Figura 4.5 – Caso especial de interseção de malhas: aresta interceptando aresta.

Uma segunda situação não tratada na versão original do algoritmo ocorre quando a aresta de uma superfície intercepta um vértice da outra, como pode ser visto na Figura 4.6. O problema aqui é exatamente equivalente ao descrito acima nas situações onde ocorre a interseção aresta/aresta. Assim, se a aresta *e1* de *Sa* intercepta o vértice *Vc* de *Sb*, a variável *intersection* de *e1* armazena apenas uma face de *Sb* (*Sb1* ou *Sb2*). Essa situação pode impedir a correta geração das curvas de interseção. A mesma solução adotada para resolver o problema de interseções aresta/aresta também resolve o problema de interseções aresta/vértice, não existindo a necessidade de se fazer outras alterações na implementação do algoritmo.

A Figura 4.7 mostra um exemplo de interseção de superfícies onde aparece uma situação dos problemas citados nesta seção.

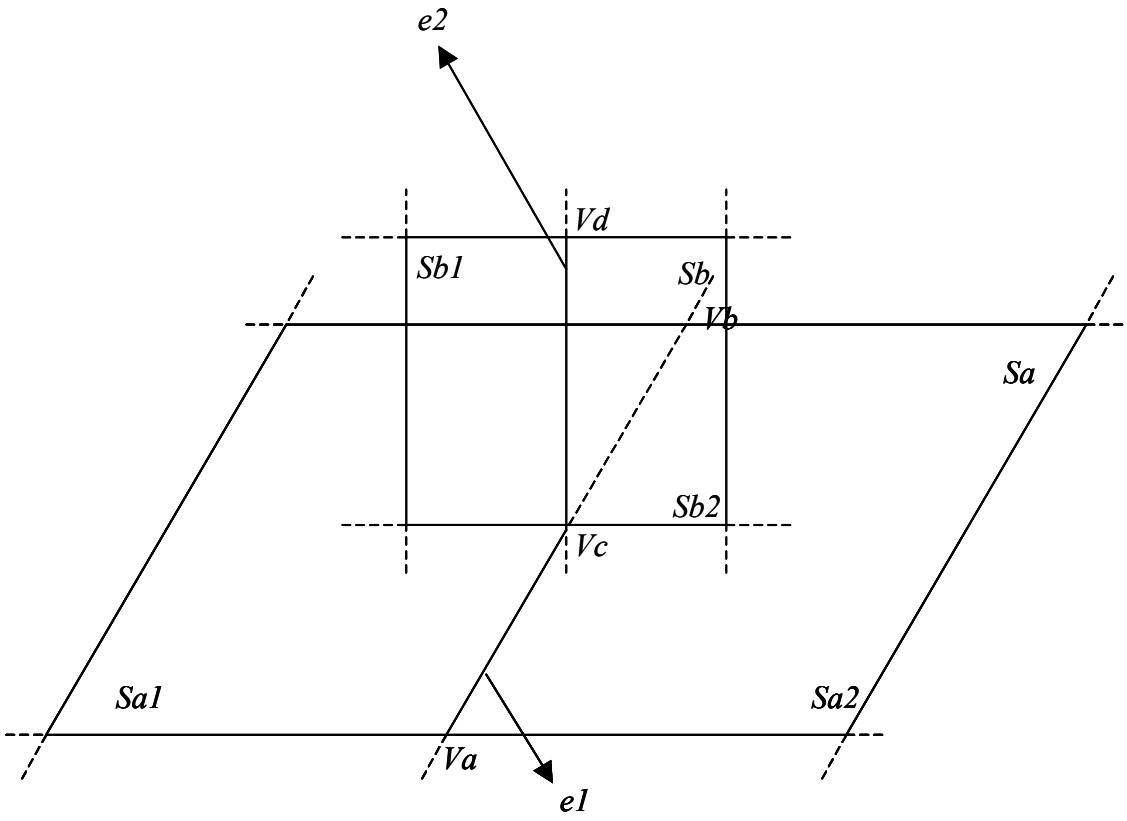


Figura 4.6 – Caso especial de interseção de malhas: aresta interceptando vértice.

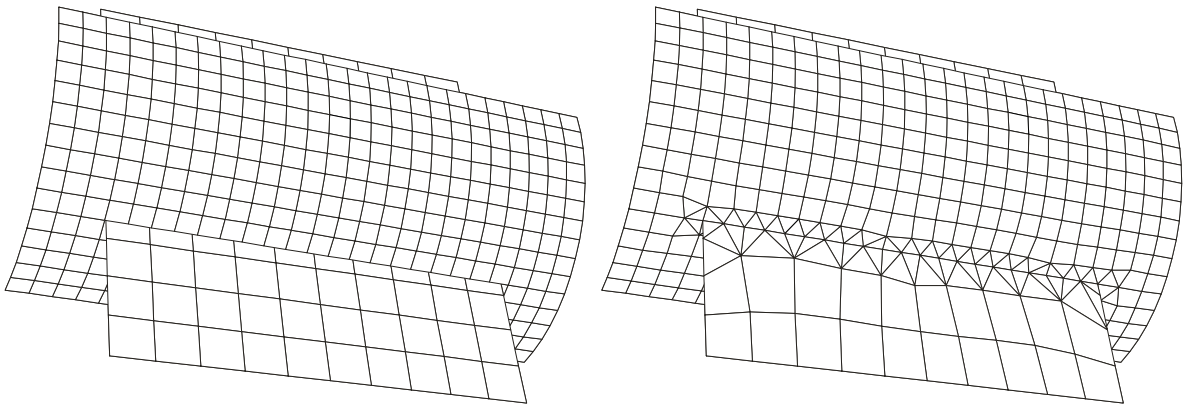


Figura 4.7 – Exemplo de interseção de malhas dos casos especiais aresta/aresta e aresta/vértice.

4.3.2 – Curvas Interceptando Curvas já Existentes

Outra situação especial no uso do algoritmo de interseção refere-se à interseção de superfícies onde a curva de *trimming* intercepta uma outra já existente, como pode ser visto na Figura 4.2. Essa situação é problemática porque no passo (3a) do algoritmo, tenta-se remover arestas e vértices, formando as regiões que devem ser trianguladas. Essas regiões são formadas a partir da propagação de uma face inicial, adjacente a primeira aresta que vai ser removida. No entanto, arestas localizadas sobre as curvas de *trimming* já existentes, conhecidas como arestas de restrição, não podem ser removidas. Quando estas arestas são encontradas pelo algoritmo, elas podem impedir essa propagação. Nessas situações, regiões de *trimming* indesejadas podem ser geradas.

A solução encontrada para resolver tal problema é simples. Em uma etapa intermediária entre os passos (2d) e (3a), verifica-se se as novas curvas de interseção interceptam arestas de restrição. Se isso ocorrer, tais curvas são divididas no ponto de interseção encontrado. Esse procedimento evita que os problemas citados acima ocorram. A Figura 4.8 mostra o resultado da interseção da Figura 4.2. Nessa figura, a seta indica o local geométrico de uma curva resultante da interseção de superfícies realizada em uma etapa anterior da modelagem.

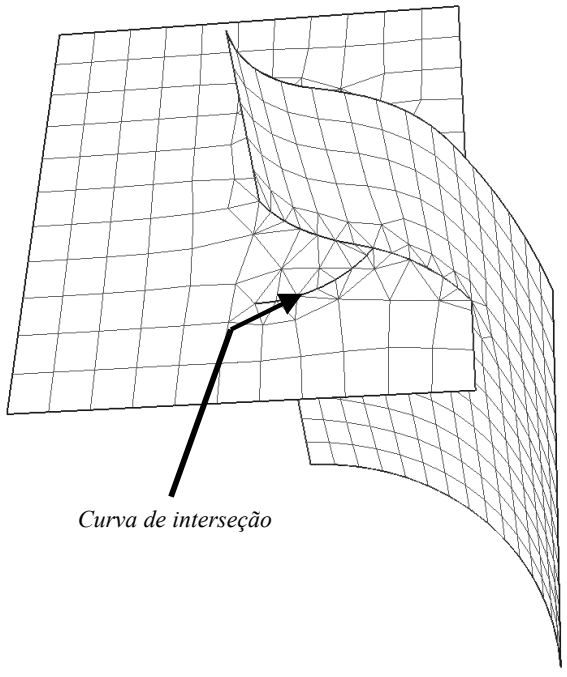


Figura 4.8 – Resultado da interseção do problema mostrado na Figura 4.2.

4.3.3 – Interseção de Superfícies Não-Retangulares

Na modelagem geométrica em problemas de engenharia, a maioria das superfícies utilizadas é retangular no seu espaço paramétrico. No entanto, existem casos onde a modelagem requer a descrição de outros tipos superfícies como, por exemplo, superfícies triangulares (ver Figura 4.9).

A versão original do algoritmo não considera tais problemas, pois a construção inicial da estrutura de dados DCEL trata apenas os casos específicos de superfícies retangulares.

Neste trabalho, tal restrição não existe mais. Um dos avanços alcançados aqui é que, desde que uma superfície possua uma descrição paramétrica, é possível utilizar o algoritmo para realizar interseções entre superfícies. Maiores detalhes sobre este assunto podem ser vistos na seção 4.4, onde é mostrado como o algoritmo de interseção é incorporado a nova versão do modelador MG. A Figura 4.10 mostra a interseção obtida para o exemplo mostrado na Figura 4.9.

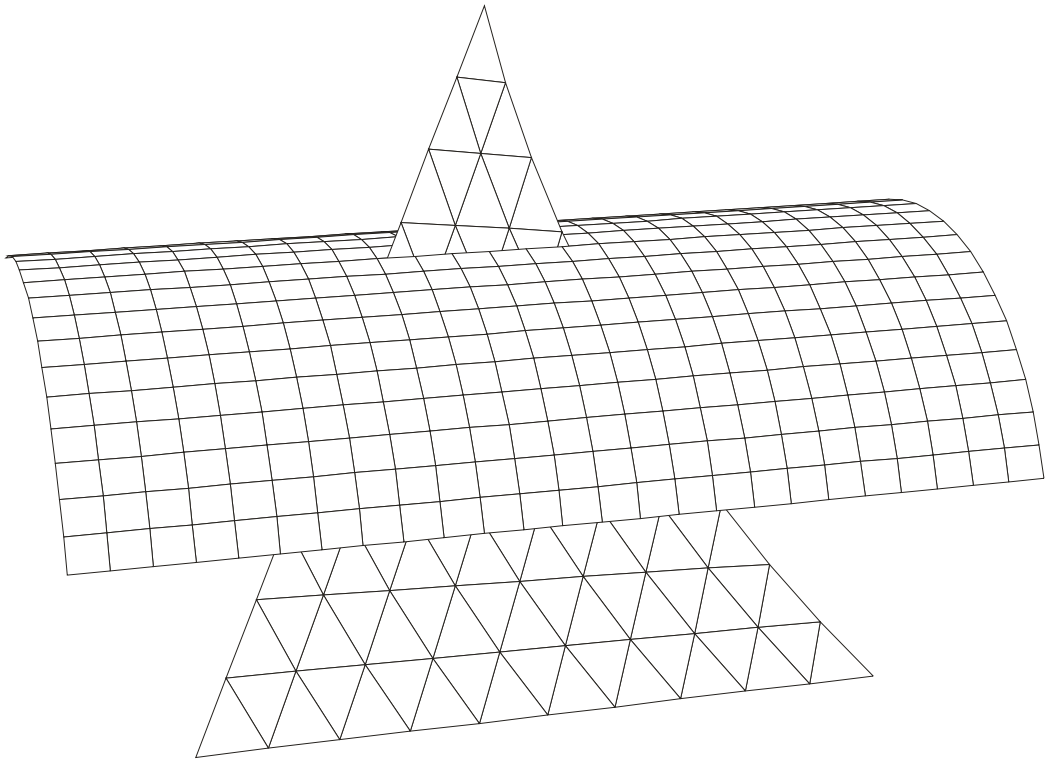


Figura 4.9 – Modelagem com superfícies não-retangulares.

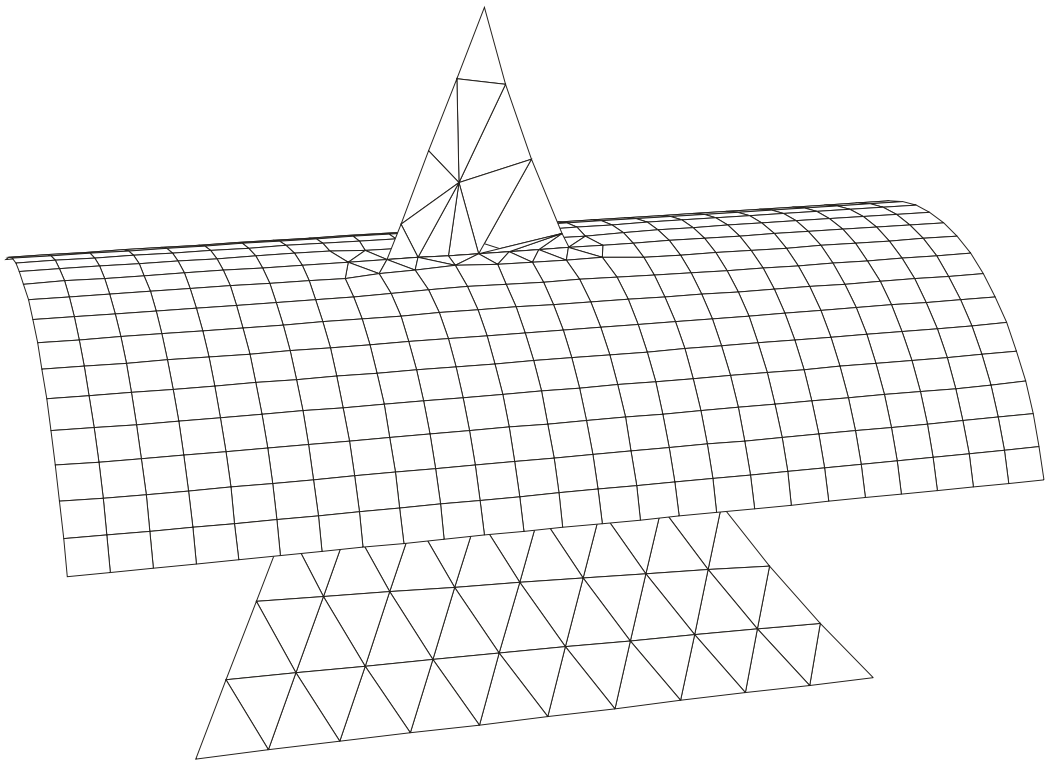


Figura 4.10 – Interseção de superfícies não-retangulares.

4.3.4 – Reconstrução de Malhas em Superfícies Incidentes às Curvas de Interseção

Existem situações onde as superfícies submetidas à interseção possuem superfícies incidentes às curvas de interseção resultantes, como pode ser visto na Figura 4.11. Aqui, a interseção entre as superfícies *A* e *B* geram malhas inconsistentes com aquela definida na superfície *C*. Quando isso ocorre, é necessário fazer uma reconstrução da malha da superfície incidente, evitando o surgimento de situações como a mostrada na Figura 4.12. Nesse caso, tal inconsistência pode gerar uma representação única da malha de elementos finitos inválida.

Para evitar tais situações, é realizada uma reconstrução das malhas nas superfícies incidentes às curvas resultantes da interseção. Isso é feito utilizando-se a mesma estrutura de dados DCEL do algoritmo de interseção. Os passos (1) e (2) do algoritmo não são usados, pois não existe a necessidade de calcular os pontos de interseção nesta superfície e nem as curvas de interseção. No lugar desses passos, usa-se um procedimento onde os pontos da curva de *trimming* gerados pela interseção das

superfícies *A* e *B* (ponto *Pi*) e que tocam a superfície *C*, são inseridos na estrutura DCEL. Nessa inserção, duas situações podem ocorrer: o ponto inserido corresponde a um vértice já existente; ou o ponto inserido está sobre uma aresta. Se o ponto inserido corresponde a um vértice já existente, então as arestas que chegam nesse vértice são removidas, exceto aquelas que pertencem ao contorno ou são de restrição. Se o ponto inserido está sobre uma aresta, esta aresta é dividida.

Os passos (3c) e (3d) do algoritmo de interseção são, então, utilizados para fazer a triangulação das novas regiões e a suavização da malha gerada, respectivamente. A Figura 4.13 mostra o resultado final deste procedimento.

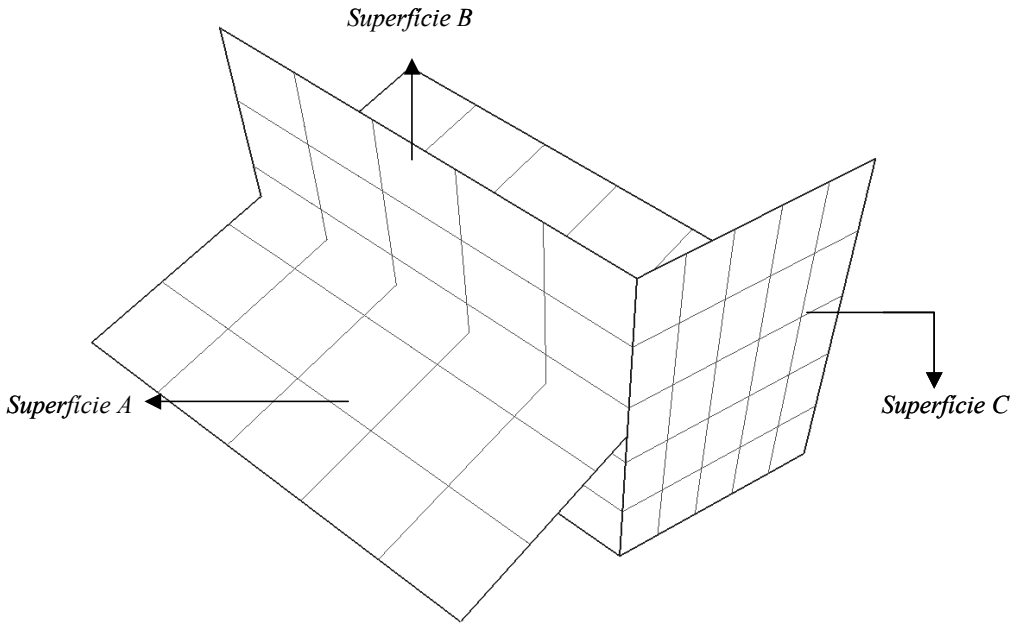


Figura 4.11 – Problema de interseção de superfícies.

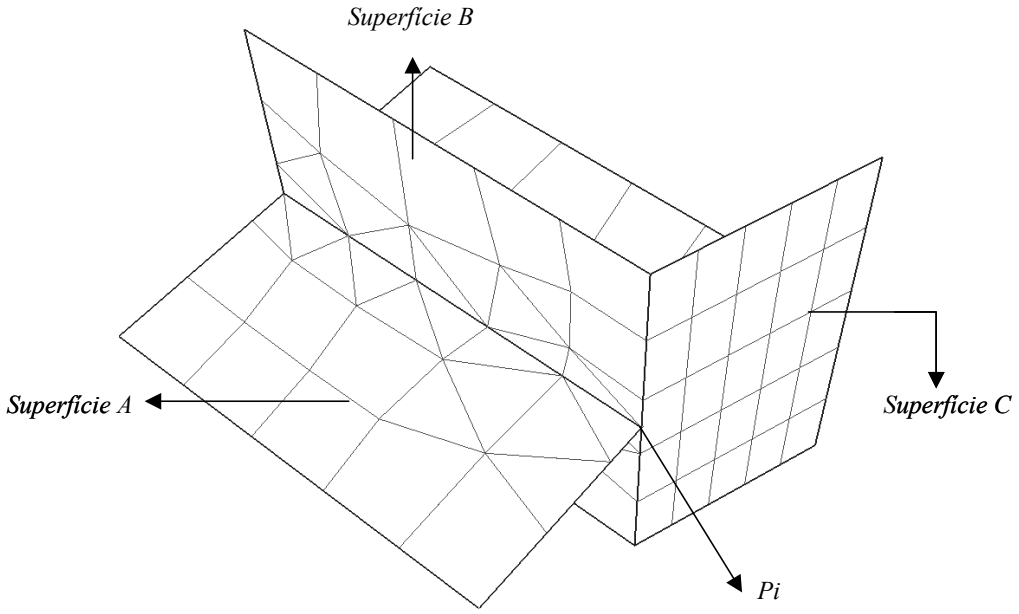


Figura 4.12 – Inconsistência entre malhas de superfícies incidentes às curvas de interseção.

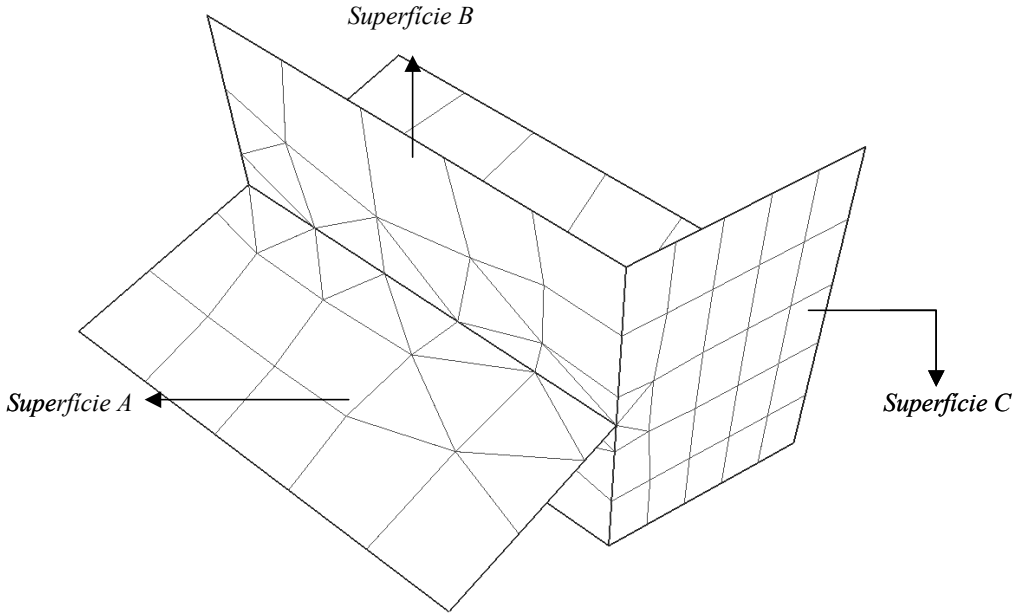


Figura 4.13 – Solução do problema de interseção apresentado na Figura 4.11.

4.4 – Comunicação entre o Algoritmo de Interseção e o Ambiente de Modelagem

O algoritmo para interseção de superfícies descrito neste capítulo foi transformado em uma biblioteca de classes, no contexto da programação orientada a objetos, usando a linguagem C++. A idéia de tratar o algoritmo como uma biblioteca permite que ele seja usado em outros programas de modelagem, além do MG. A estrutura de classes adotada é bastante simples. Existem classes para representar cada uma das entidades topológicas da estrutura DCEL (classes *dcelVertex*, *dcelEdge*, *dcelFace*, que descrevem, respectivamente, vértices, arestas e faces topológicas).

O gerenciamento do algoritmo é realizado pela classe *dcel*. Outras duas classes importantes são: *dcelTrimming*, responsável pela execução do passo (2) do algoritmo, ou seja, construção das curvas de interseção; e classe *dcelTriangulate*, responsável pela etapa de reconstrução da malha, ou seja, o passo (3) do algoritmo de interseção.

Essa biblioteca possui, ainda, uma classe, denominada *dcelClient*, que descreve os métodos genéricos que devem ser implementados pelo modelador que vai usar a biblioteca. A implementação desses métodos genéricos é a única tarefa que deve ser realizada para incorporar a biblioteca de interseção ao modelador. A Figura 4.14 mostra uma organização de classes da biblioteca de interseção de superfícies, mostrando a fluxo de comunicação entre os objetos destas classes.

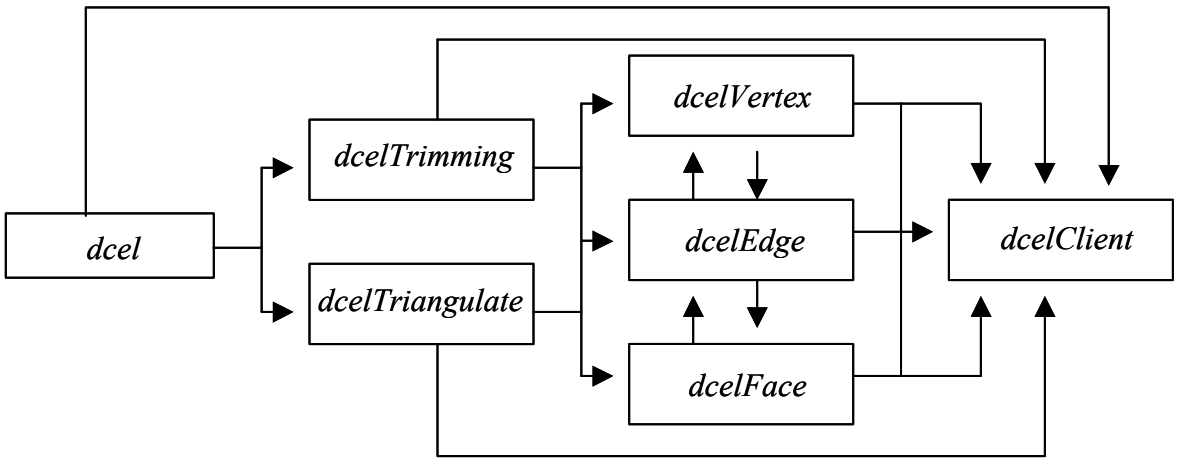


Figura 4.14 – Fluxograma de comunicação entre as classes da biblioteca para interseção de superfícies.

Os métodos genéricos são:

(a) *getParametricMesh*: este método recebe uma malha de superfícies inicial que é convertida na estrutura de dados topológica usada pelo algoritmo. Esse método é chamado no início do processo de interseção de superfícies;
(b) *getConstraint*: este método recebe as restrições associadas à malha da superfície; por exemplo, as arestas sobre as curvas de interseção obtidas em uma etapa anterior. Esse método é chamado no início do processo de interseção;
(c) *evalSurface*: dado uma coordenada no espaço paramétrico, este método calcula as coordenadas 3D correspondentes e suas derivadas parciais. Esse método é chamado durante todo o processo de interseção;
(d) *closestSurfacePoint*: dado um ponto 3D na superfície, este método determina as coordenadas paramétricas correspondentes. Esse método é chamado durante todo o processo de interseção;
(e) *getTrimmingCurve*: dado um conjunto de pontos de interpolação obtidos pelo algoritmo de interseção ao longo de uma curva de *trimming*, este método retorna uma referência de uma representação paramétrica desta curva definida pelo modelador;
(f) *getCurveSub*: dado uma curva de *trimming* e um tamanho característico, este método calcula pontos igualmente espaçados ao longo da curva. Esse tamanho característico é definido pelo tamanho médio das arestas interceptadas da malha inicial;
(g) *newPatchMeshes*: para cada retalho de superfície inicial, este método passa para o modelador um conjunto de malhas (definidas sobre o retalho correspondente) resultantes da interseção.

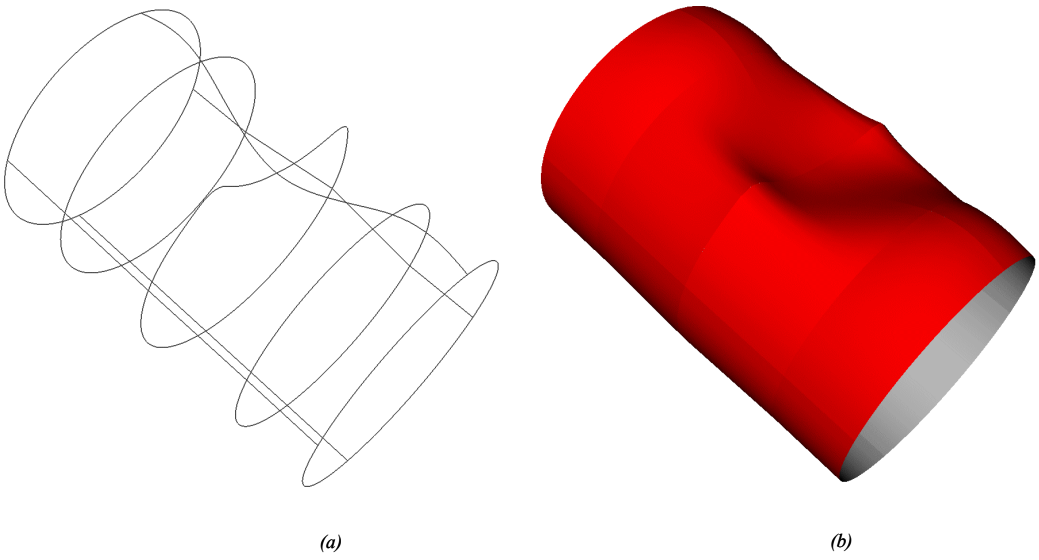
5. Exemplos

Este capítulo mostra exemplos das técnicas de modelagem apresentadas neste trabalho e implementadas no modelador MG. Tais exemplos ilustram diversas etapas envolvidas em uma modelagem. Essas etapas vão desde a modelagem de superfícies, até o reconhecimento de multi-regiões, passando pelo problema de interseção entre superfícies. O capítulo apresenta ainda resultados provenientes de análise numérica de um problema de engenharia usando MEF, cujos modelo geométrico e a malha de elementos finitos foram gerados pelo MG. Tais resultados servem como validação das técnicas aqui apresentadas.

5.1 - Modelagem Geométrica com NURBS

Nesta seção, dois exemplos de problemas reais de engenharia são apresentados para ilustrar o uso, na modelagem de superfícies, da biblioteca NURBS++. Ambos exemplos mostram vantagens da utilização da biblioteca relacionadas à implementação de novos tipos de superfícies no MG.

O primeiro exemplo refere-se à modelagem de um duto amassado, como pode ser visto na Figura 5.1. As curvas utilizadas na geração dos vários retalhos de superfícies de Coons, podem ser vistas na Figura 5.1a. A Figura 5.1b mostra as superfícies obtidas nessa modelagem. Observa-se que existem descontinuidades nas junções desses retalhos, bem como regiões pontiagudas no modelo.



(a) (b)

Figura 5.1 – Exemplo de modelagem de um duto amassado: a) curvas geradoras; b) retalhos de Coons.

A superfície de Gordon é um dos novos tipos de superfícies implementados no MG a partir da incorporação da biblioteca NURBS++. O mesmo problema do duto amassado foi modelado utilizando-se esse tipo de superfície, como pode ser visto na Figura 5.2. Aqui, as curvas utilizadas na geração da superfície são as mesmas apresentadas na Figura 5.1a. Um único retalho de superfície é gerado em cada quadrante, evitando o surgimento das descontinuidades observadas na Figura 5.1b.

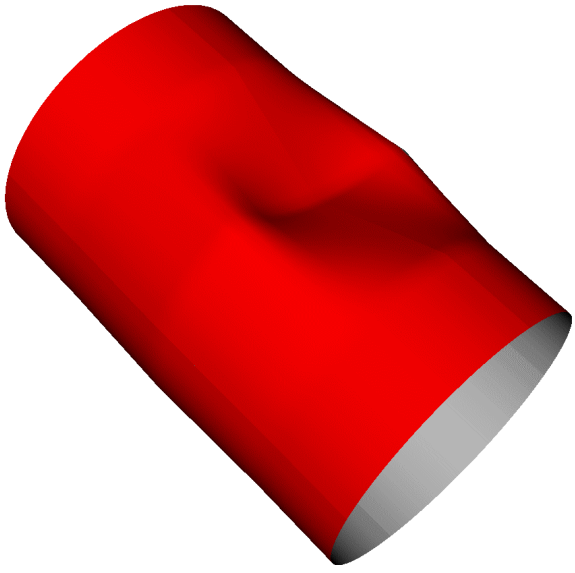
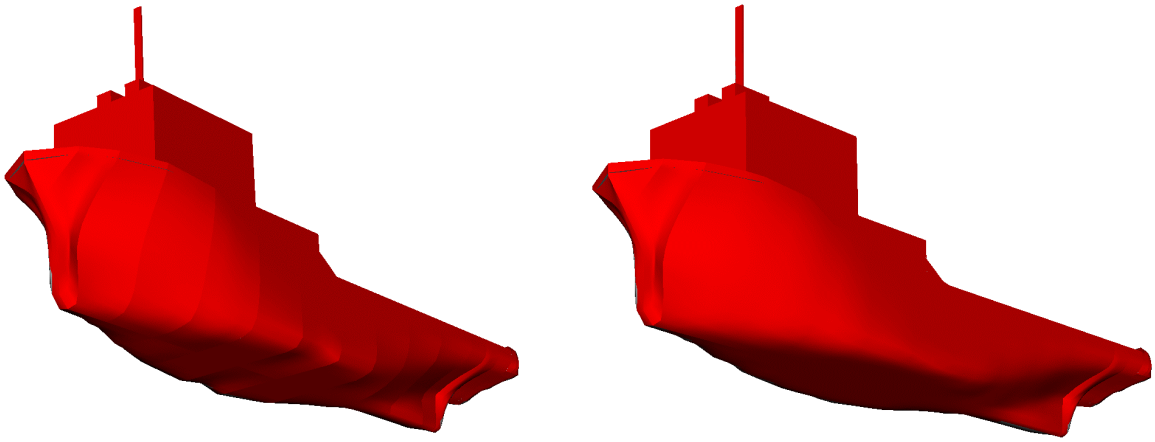


Figura 5.2 – Modelagem de um duto amassado usando superfícies de Gordon.

O mesmo problema apresentado no exemplo acima, também pode ser visto no exemplo de modelagem de cascos de navios, como apresentado na Figura 5.3. A Figura 5.3a mostra o casco modelado por superfícies de Coons, enquanto que a Figura 5.3b mostra o uso de superfícies de Gordon modelando o mesmo problema.



(a) *(b)*

Figura 5.3 – modelagem de cascos de navios: a) superfícies de Coons; b) superfícies de Gordon.

5.2 – Interseção de Superfícies

Esta seção ilustra alguns exemplos de interseção entre superfícies no MG, usando o algoritmo proposto por Coelho e modificado neste trabalho. O primeiro exemplo mostra a interseção de dois cilindros. A Figura 5.4a mostra as malhas originais nas duas superfícies, enquanto que a Figura 5.4b representa as malhas resultantes da interseção. Na Figura 5.4c, alguns retalhos resultantes da interseção são removidos, ilustrando a capacidade do MG de gerar modelos a partir da composição de vários componentes de modelagem.

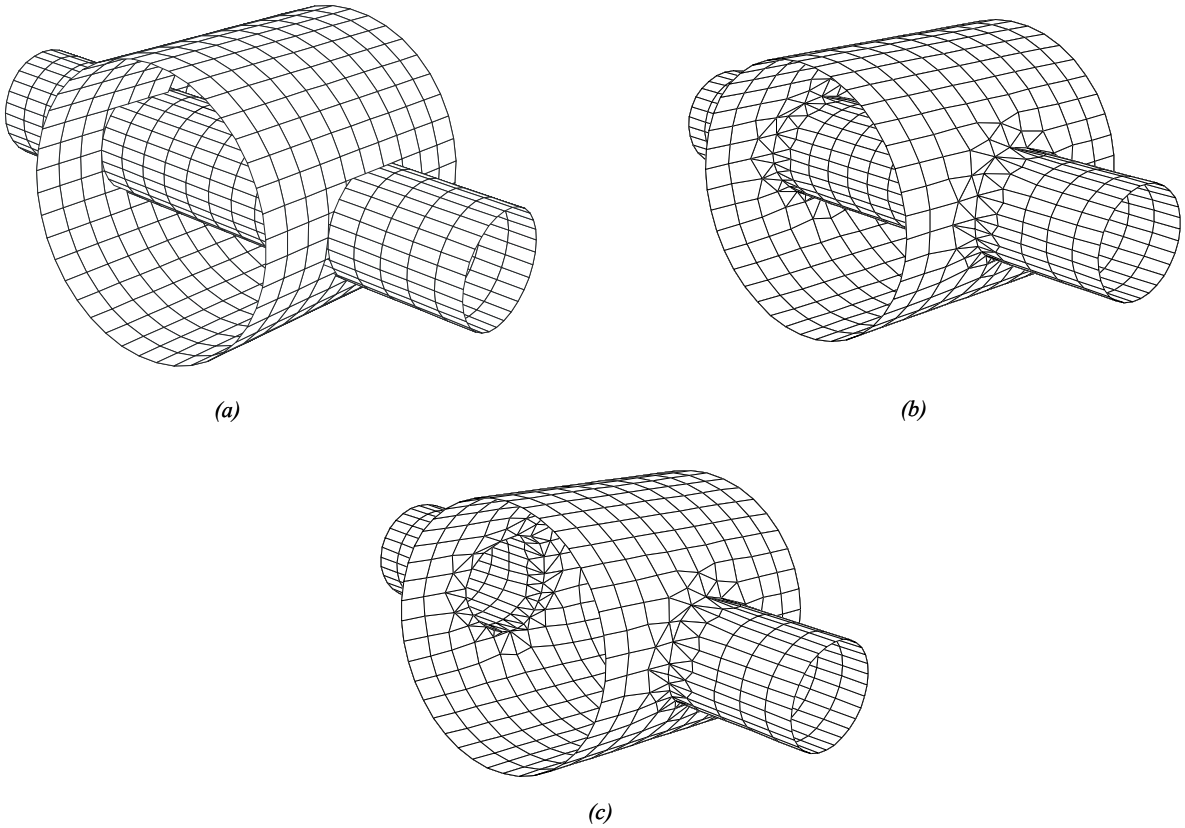


Figura 5.4 – Exemplo de interseção entre superfícies: a) malhas originais; b) malhas resultantes da interseção; c) manipulação das superfícies interceptadas.

O exemplo mostrado na Figura 5.5 ilustra a interseção de um toro (*torus*) com um cilindro. A Figura 5.5a mostra as malhas originais das superfícies, enquanto que a Figura 5.5b mostra as malhas resultantes da interseção. A Figura 5.5c mostra as curvas geradoras das superfícies e a curva de *trimming* (em destaque) resultante da interseção. Este exemplo serve para ilustrar a capacidade do algoritmo de interseção em determinar curvas de *trimming* complexas na interseção entre pares de superfícies.

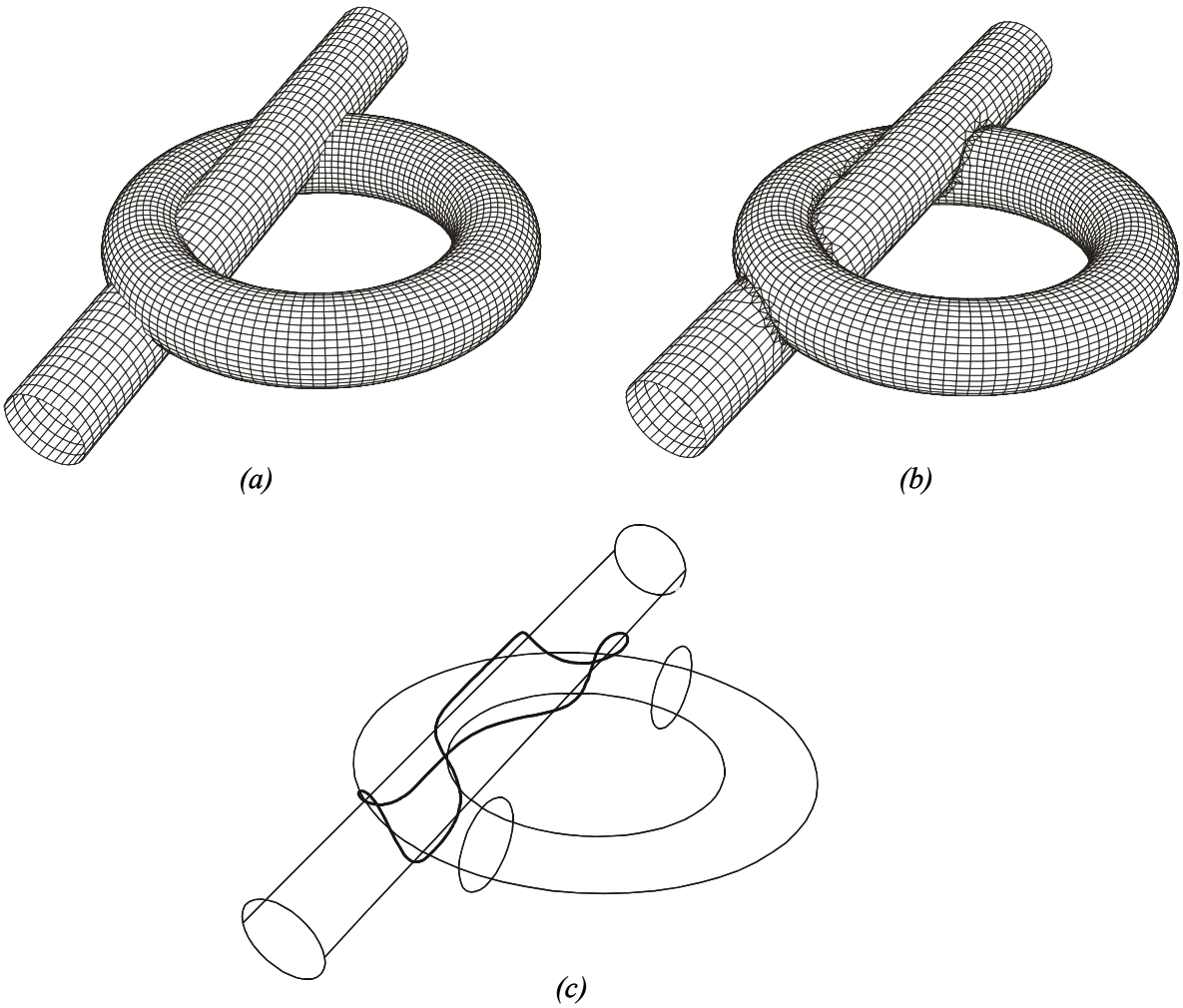


Figura 5.5 – Exemplo de curvas de *trimming* resultantes da interseção entre superfícies: a) malhas originais; b) malhas resultantes da interseção; c) curvas de *trimming*.

A Figura 5.6 ilustra um dos principais problemas considerados neste trabalho: a interseção entre superfícies que já haviam sido interceptadas e cujas novas curvas de *trimming* cortam outras já existentes. A Figura 5.6a mostra, em uma etapa inicial, três superfícies que vão se interceptar (superfícies *A*, *B* e *C*). A Figura 5.6b mostra as malhas resultantes da interseção entre as superfícies *A* e *C*. Por fim, a Figura 5.6c mostra a interseção da superfície B com a superfície *A* já interceptada. Deve-se observar que as curvas de *trimming* geradas se cortam no exemplo citado.

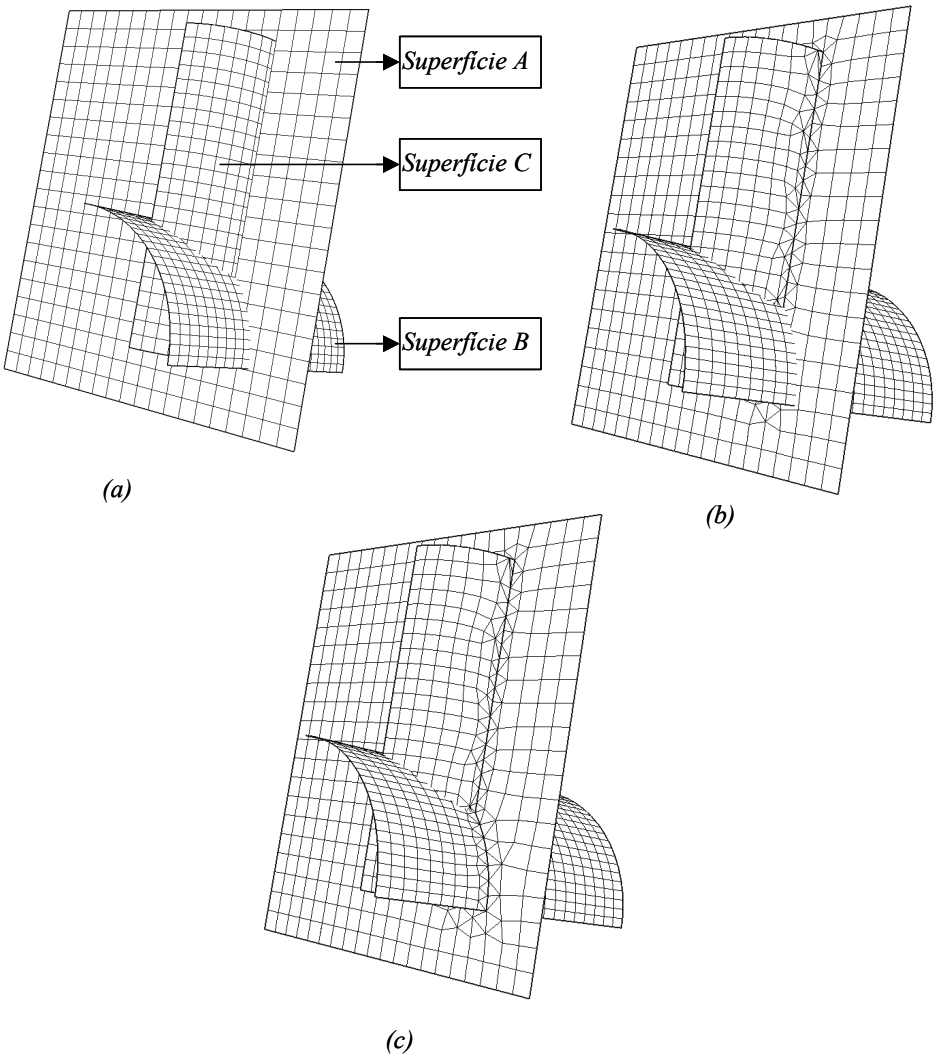


Figura 5.6 – Interseção entre superfícies cujas curvas de *trimming* se cortam: a) malhas originais; b) malhas resultantes da interseção entre duas superfícies; c) malhas resultantes da interseção de uma terceira superfície com o resultado mostrado em b).

O exemplo mostrado na Figura 5.7 ilustra um problema onde a interseção de superfícies pode gerar inconsistência geométrica e topológica entre as entidades do modelo. Nesse exemplo, a curva de *trimming* resultante da interseção entre as superfícies *A* e *B* toca uma terceira superfície *C*. Após a interseção, a malha na superfície *C* é inconsistente com a malha da superfície *B* resultante da interseção com *A*. Em situações como essas, malhas nas superfícies incidentes às curvas de interseção são reconstruídas automaticamente. Aqui, a Figura 5.7a mostra as malhas originais nas superfícies *A*, *B* e *C*, enquanto que a Figura 5.7b mostra as malhas resultantes nessas três superfícies. A Figura 5.7c mostra um detalhe, antes da interseção, da região mostrada na Figura 5.7b. A Figura 5.7d mostra a região detalhada na Figura 5.7b após a interseção.

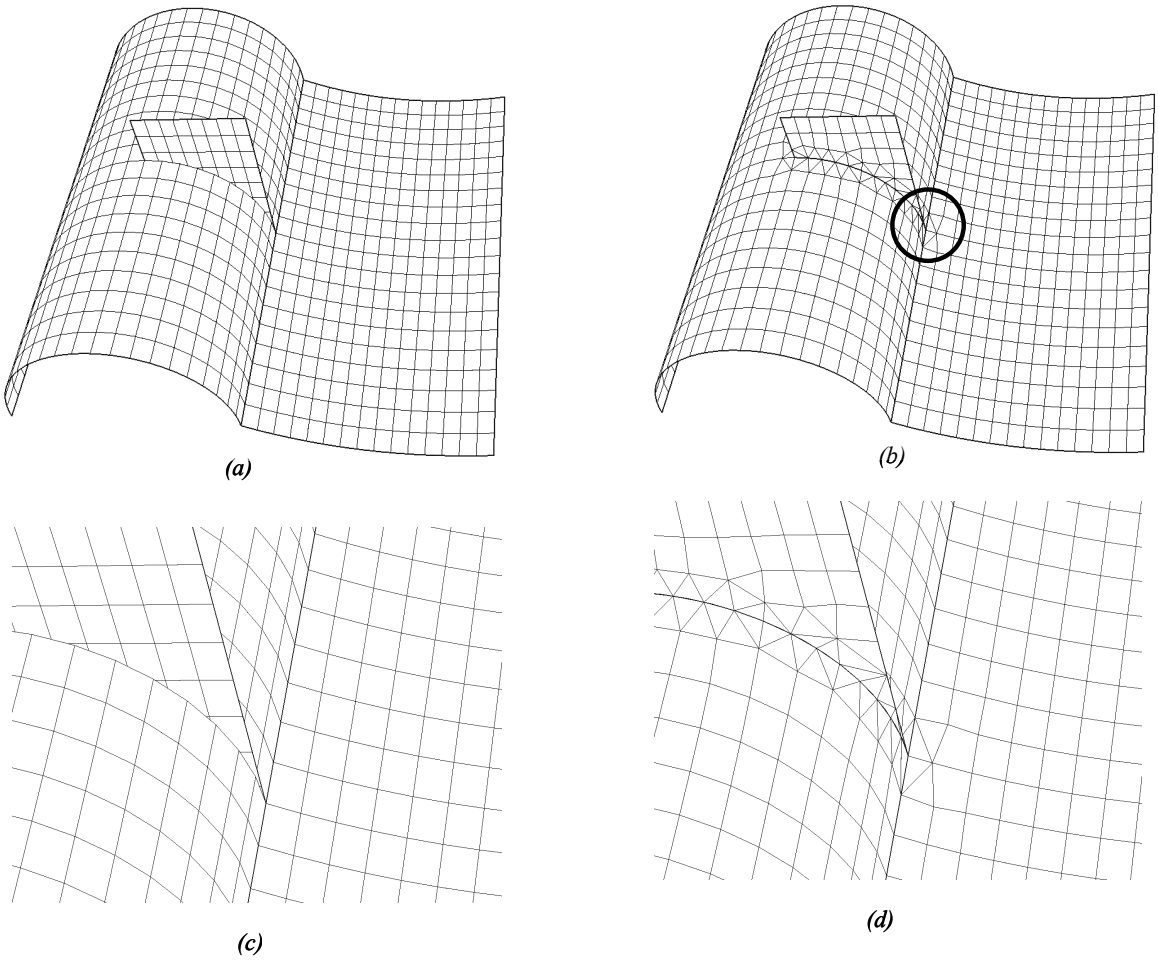
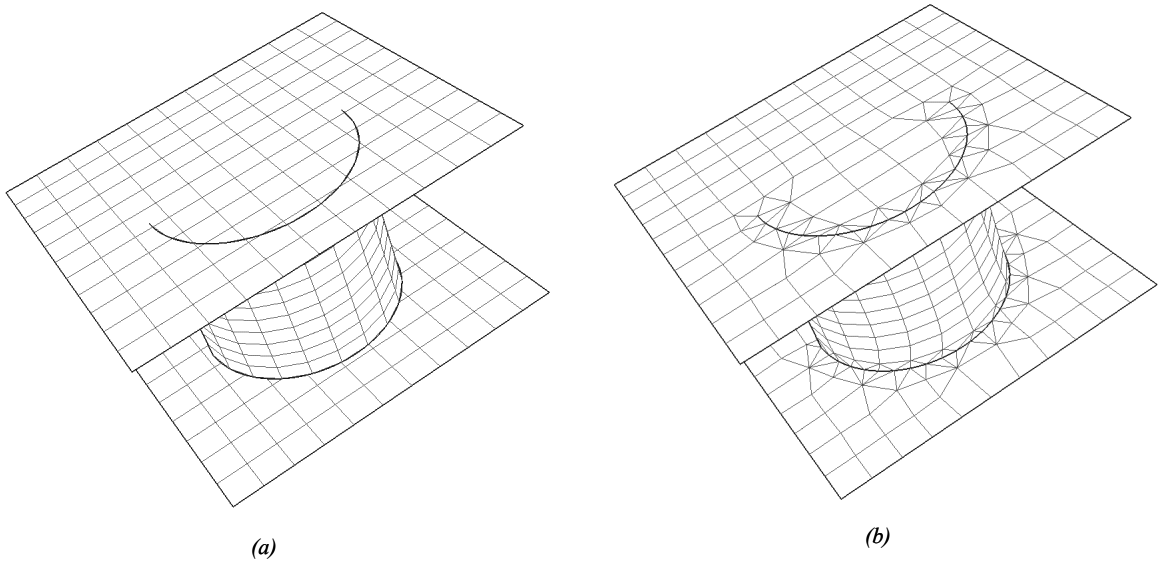


Figura 5.7 – Reconstrução de malhas em superfícies incidentes à curva de interseção: a) malhas originais; b) malhas resultantes da interseção; c) detalhe antes da interseção; d) detalhe após a interseção.

A Figura 5.8 mostra um exemplo de interseção entre um plano e um cilindro. Essa situação é interessante, pois as duas superfícies apenas se tocam em seus bordos. A curva de interseção gerada é uma curva de bordo em uma das superfícies. Em situações como essas, as malhas nas superfícies também são reconstruídas para forçar a consistência entre elas.



(a) *(b)*

Figura 5.8 – Interseção entre superfícies que se tocam em seus bordos: a) malhas originais; b) malhas resultantes da interseção.

5.3 – Reconhecimento Automático de Regiões

Esta seção mostra exemplos de modelagem onde a técnica para reconhecimento automático de multi-regiões, apresentada neste trabalho, é utilizada. O primeiro exemplo refere-se à modelagem de parte de uma plataforma usada na exploração de petróleo em águas profundas, conhecida como TLP (*Tension Leg Plataform*). Nesse exemplo, o modelo original da TLP foi modificado permitindo a visualização mais clara dos resultados obtidos com o uso do procedimento para detecção automática de regiões fechadas. A Figura 5.9 mostra as curvas utilizadas na geração dos retalhos de superfícies. A modelagem dessas superfícies, mostrada na Figura 5.10, é realizada usando superfícies de Coons e superfícies do tipo *skin*. A Figura 5.11 mostra o modelo completo da plataforma TLP. A Figura 5.12 mostra o modelo explodido, onde as regiões foram detectadas automaticamente pela técnica descrita neste trabalho.

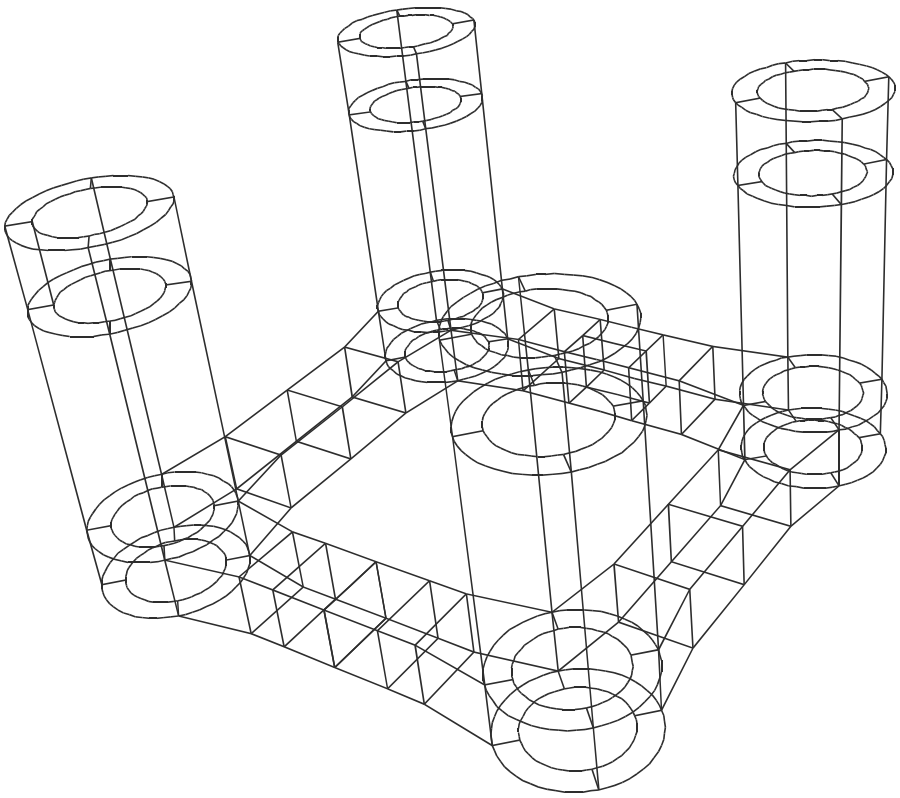


Figura 5.9 – Curvas utilizadas na geração dos retalhos de superfícies da TLP.

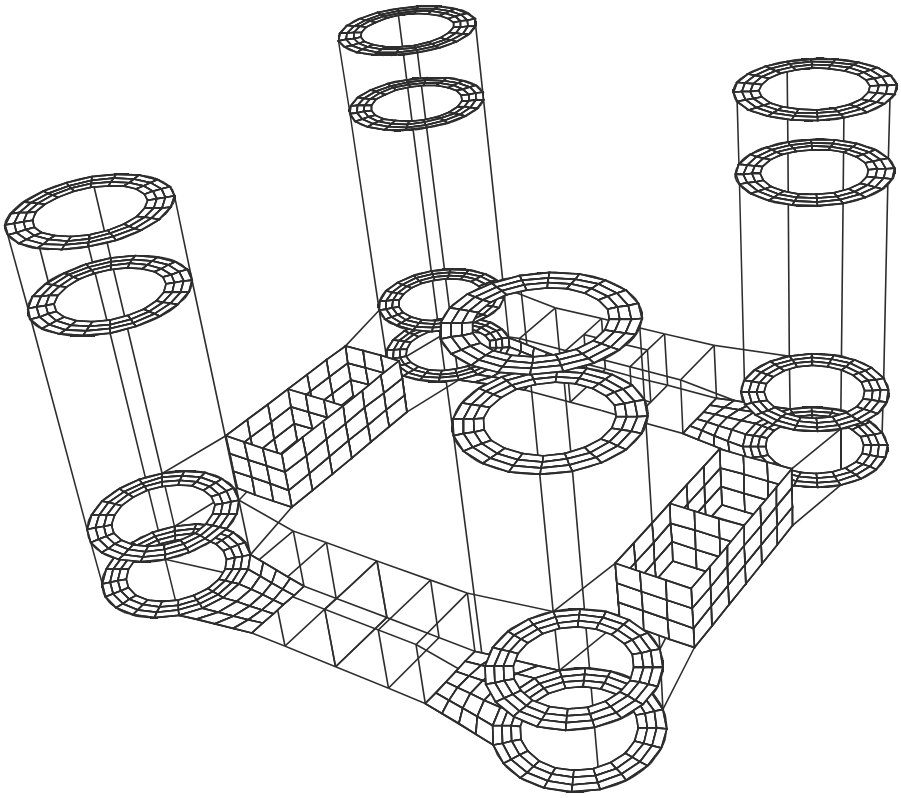


Figura 5.10 – Modelagem das superfícies da TLP.

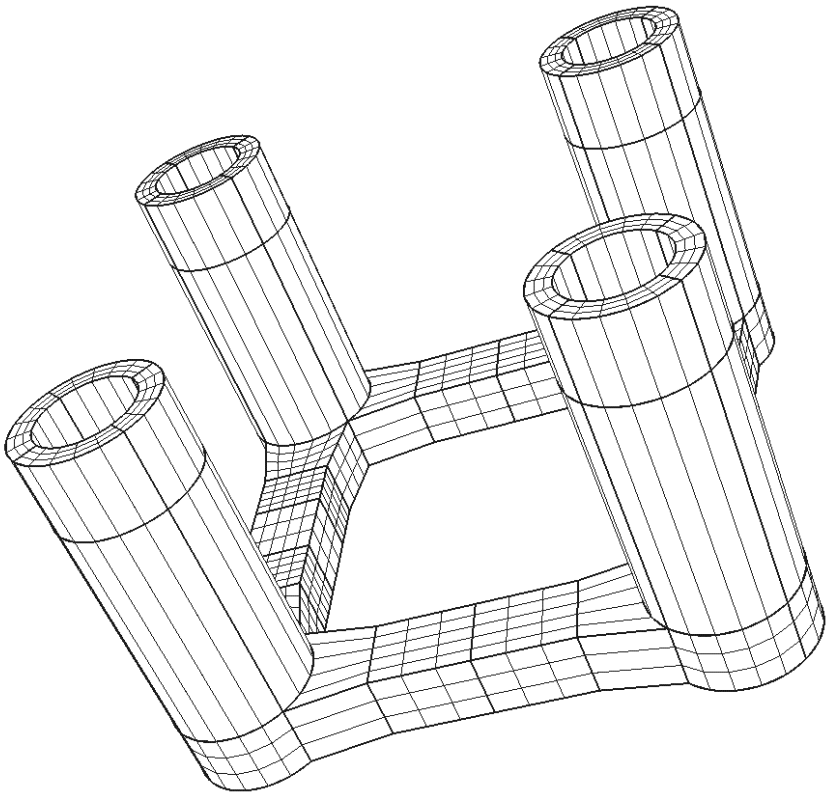


Figura 5.11 – Modelo Final da TLP.

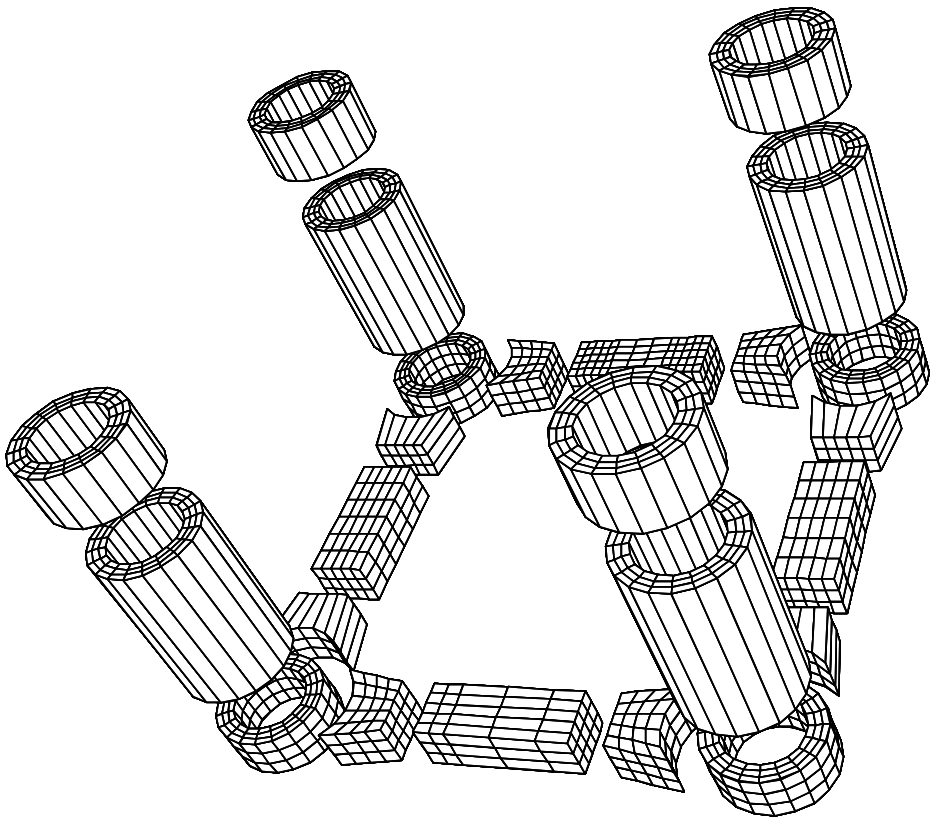
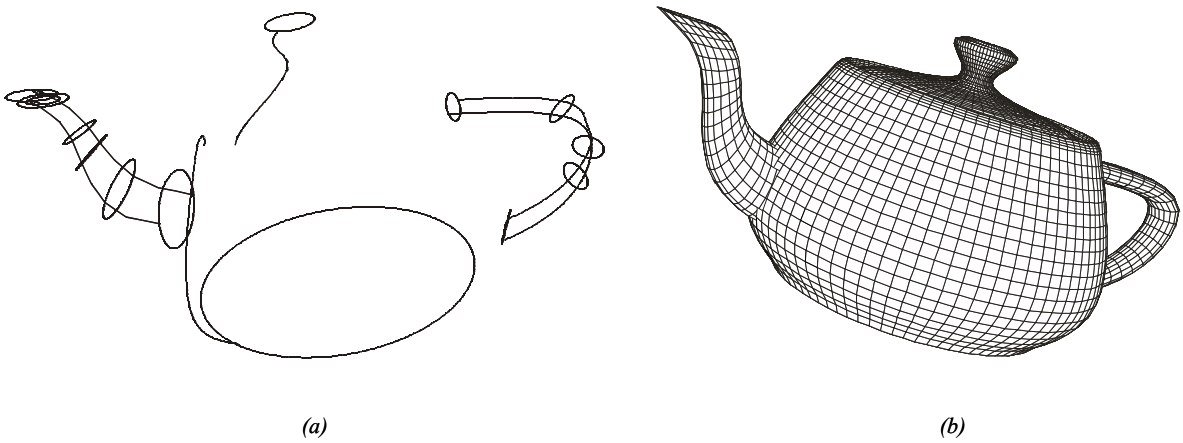


Figura 5.12 – Modelo explodido: regiões detectadas automaticamente pela técnica descrita neste trabalho.

O segundo exemplo refere-se à modelagem de um bule (*teapot*). A Figura 5.13a mostra as curvas primitivas usadas para gerar os retalhos de superfícies iniciais do bule. Esses retalhos de superfícies são mostrados na Figura 5.13b. A Figura 5.14 mostra os retalhos de superfícies usados na modelagem do bule explodidos. As superfícies do corpo e da tampa do bule são geradas usando a técnica de *sweep* rotacional. As superfícies do bico e da alça são modeladas usando retalhos de Gordon. Tampas planares são usadas para fechar o modelo nas partes superior e inferior do bule. As Figuras 5.13 e 5.14 mostram as malhas de elementos finitos iniciais que são geradas sobre os retalhos de superfícies do modelo.



(a) *(b)*

Figura 5.13 – Modelagem do bule: a) curvas primitivas; b) retalhos de superfícies.

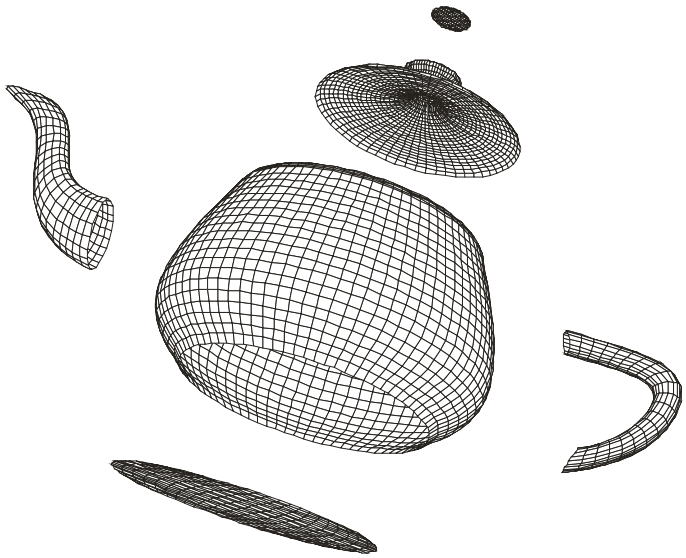


Figura 5.14 – Modelo do bule explodido.

As Figuras 5.15, 5.16 e 5.17 são usadas para exemplificar a interseção de superfícies e a capacidade de reconhecimento de regiões do modelador MG. As Figuras 5.15 e 5.16 mostram um detalhe da conexão entre o bico e o corpo do bule antes da interseção de superfícies. Na Figura 5.16, pode-se observar que a superfície do bico entra no corpo do bule.

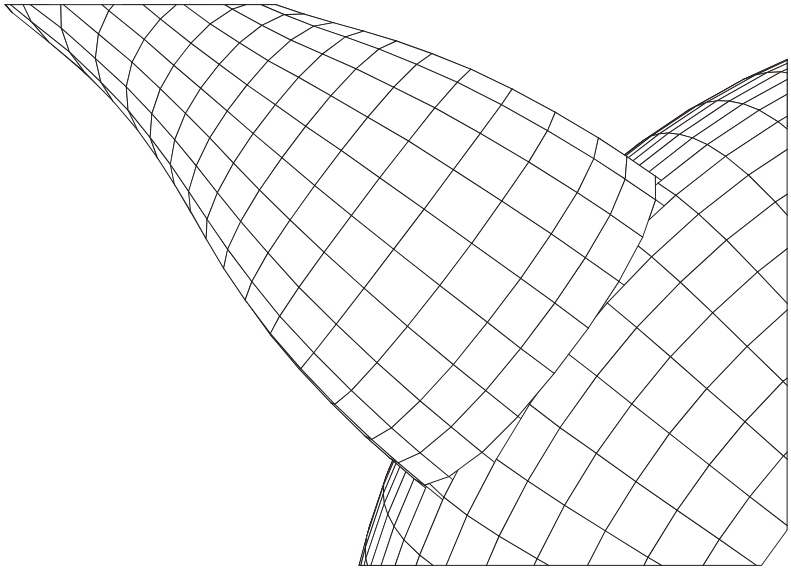


Figura 5.15 – Detalhe da malha de elementos finitos do bico e do corpo do bule.

A Figura 5.16 mostra um detalhe da malha de elementos finitos resultante da interseção entre o bico e corpo do bule. Neste caso, um remalhamento local foi realizado, ou seja, apenas os elementos próximos à curva de interseção são afetados pela interseção. Novos nós são gerados no espaço paramétrico de cada superfície, indicando que eles estão exatamente sobre as superfícies, incluindo nós da curva de interseção.

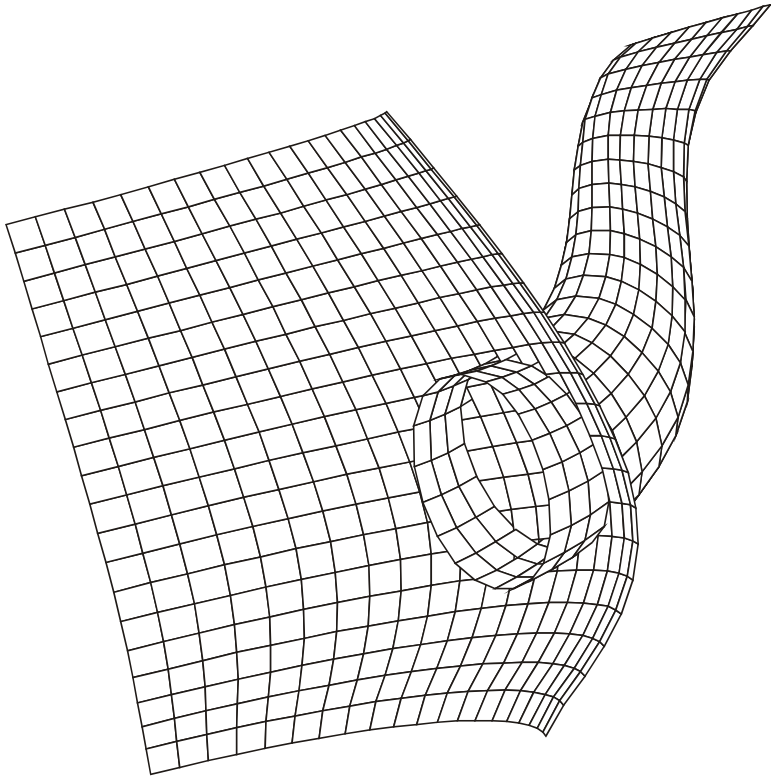


Figura 5.16 – Detalhe da superfície do bico original e do corpo do bule.

A detecção automática de regiões do bico do bule após a interseção com o corpo é mostrada na Figura 5.17. Pode-se observar que a porção da superfície do bico que estava dentro do corpo do bule é eliminada. Isso é uma tarefa trivial no corrente ambiente de modelagem porque esta porção é detectada automaticamente como um retalho independente. Na Figura 5.18, também pode-se notar que a malha localizada no retalho de superfície do corpo que ficou sobre o bico foi opcionalmente re-gerada como um todo, ao invés de manter a malha modificada localmente.

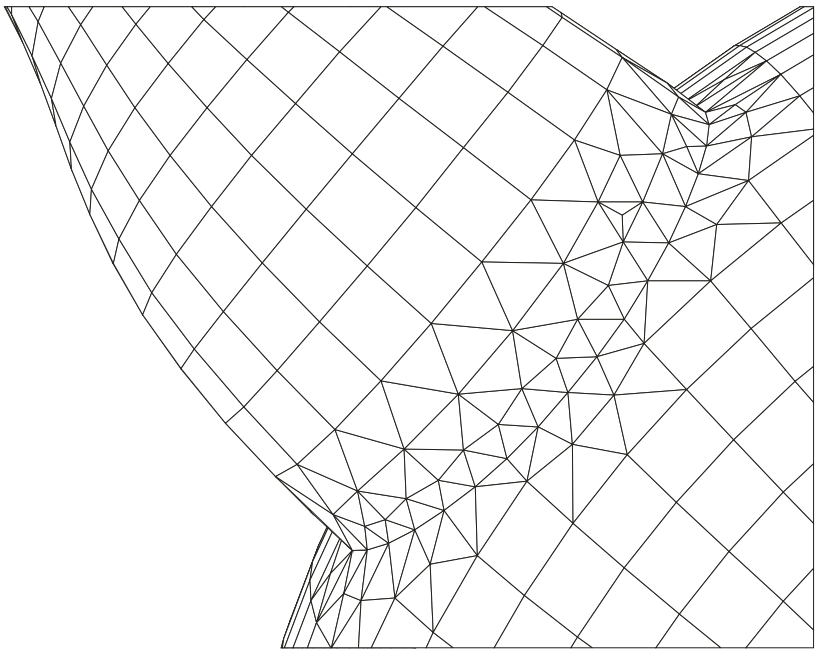


Figura 5.17 – Detalhe da malha de elementos finitos resultante da interseção do bico e do corpo do bule.

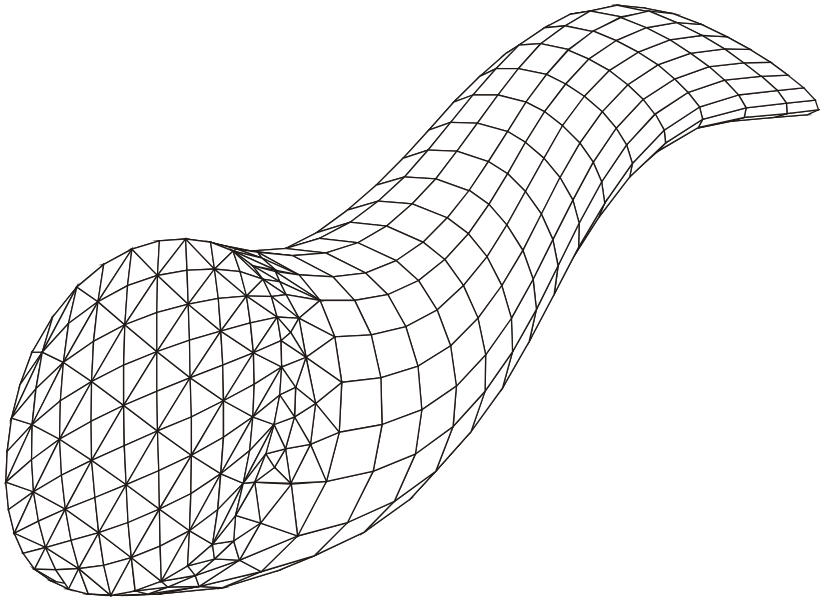


Figura 5.18 – Região detectada automaticamente do bico do bule após a interseção com o corpo.

5.4 – Análise Usando Elementos Finitos

Com o objetivo de validar as ferramentas propostas neste trabalho, esta seção apresenta um exemplo de problema real de engenharia cuja análise de tensões é realizada usando o método dos elementos finitos. O exemplo é uma turbina circular com doze pás. A Figura 5.19 mostra as curvas que são usadas para gerar os retalhos de superfícies iniciais. Essas superfícies, mostradas na Figura 5.20, são modeladas por superfícies de Coons. A Figura 5.21 mostra o modelo explodido. Todas as regiões foram detectadas automaticamente. A malha de elementos finitos sólidos, mostrada na Figura 5.22, foi gerada em cada região das pás separadamente usando o algoritmo para geração de malhas volumétricas não-estruturadas mencionado no Capítulo 2. O desenho de uma componente de tensão resultante da análise pelo método dos elementos finitos é mostrado na Figura 5.23. Neste exemplo, as condições de contorno associadas ao modelo são pressão aplicada a uma das pás da turbina e restrições no interior do disco impedindo o seu deslocamento longitudinal. As propriedades do material associado ao modelo são consideradas elásticas.

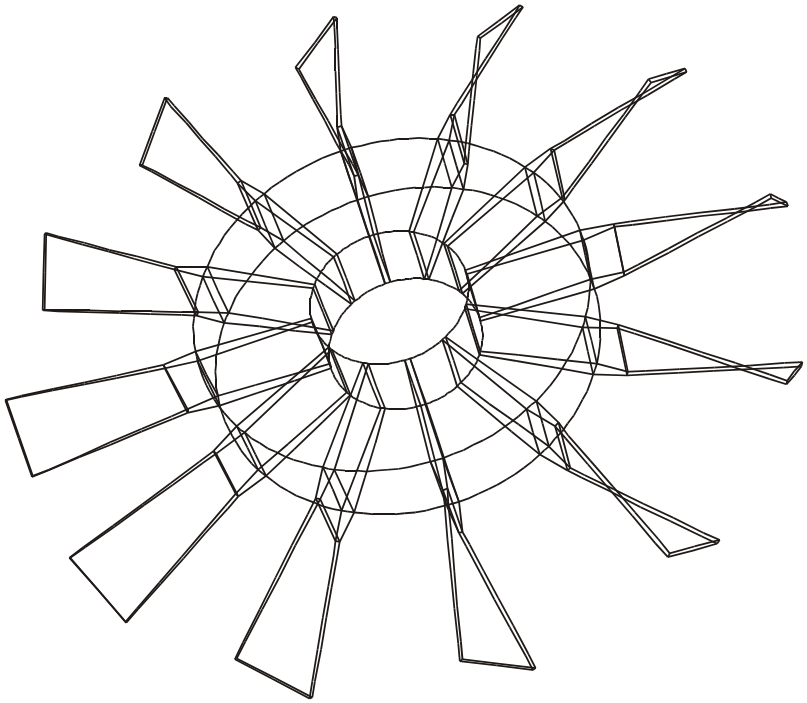


Figura 5.19 – Curvas primitivas de uma turbina circular com doze pás.

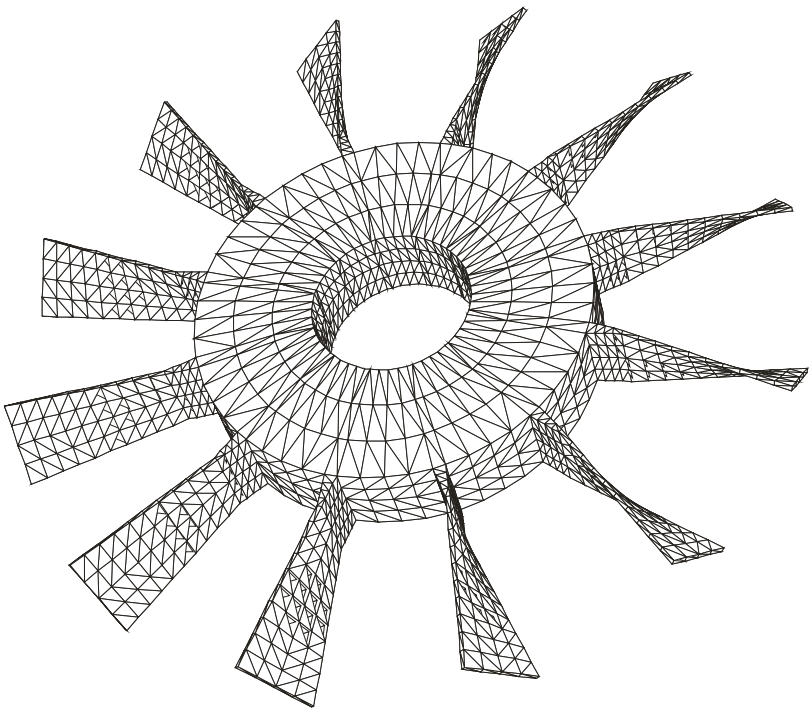


Figura 5.20 – Retalhos de superfícies da turbina circular.

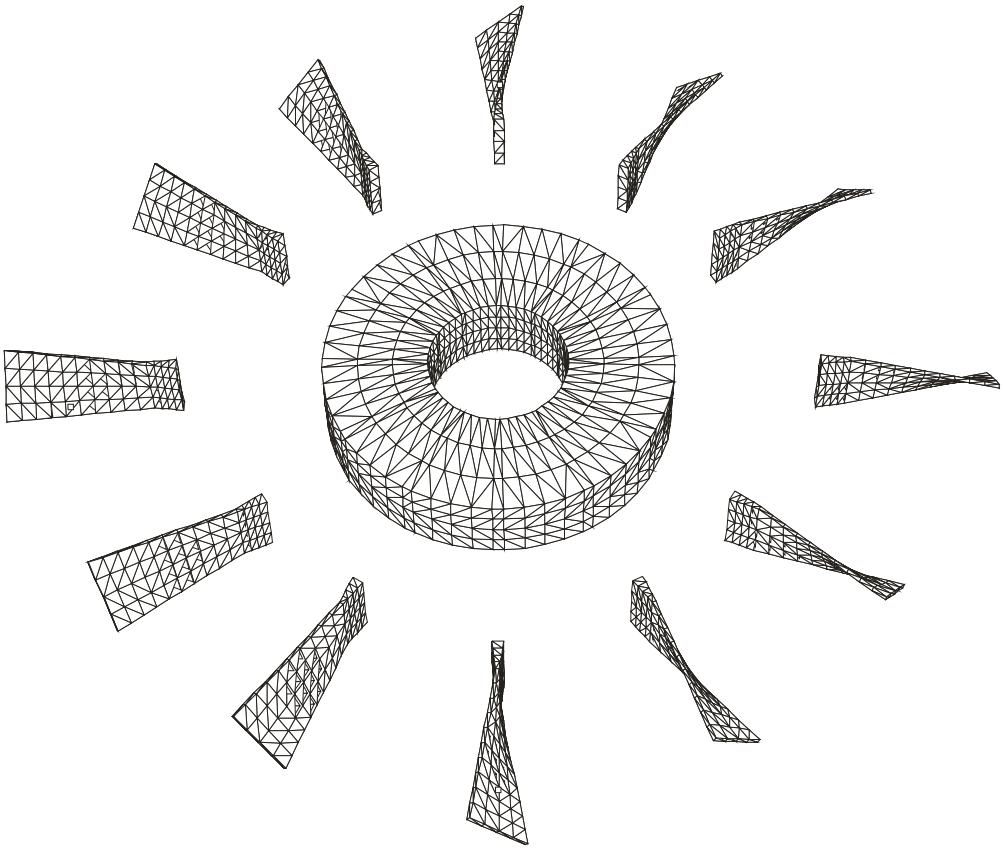


Figura 5.21 –Modelo da turbina circular explodido.

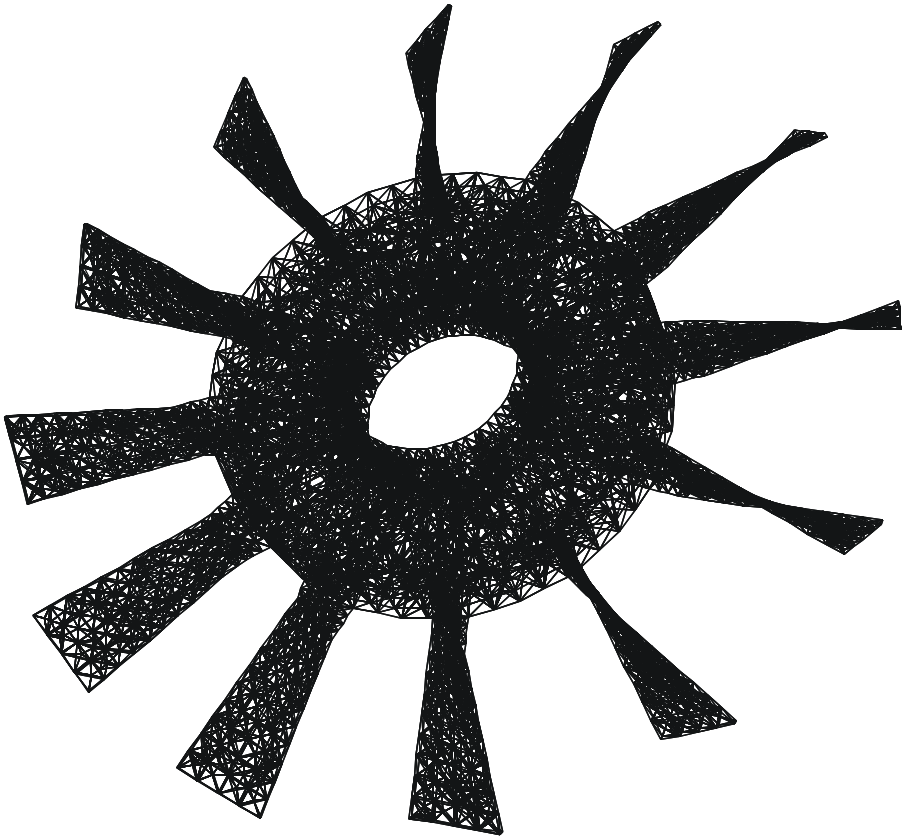


Figura 5.22 – Malha de elementos finitos sólida da turbina circular.

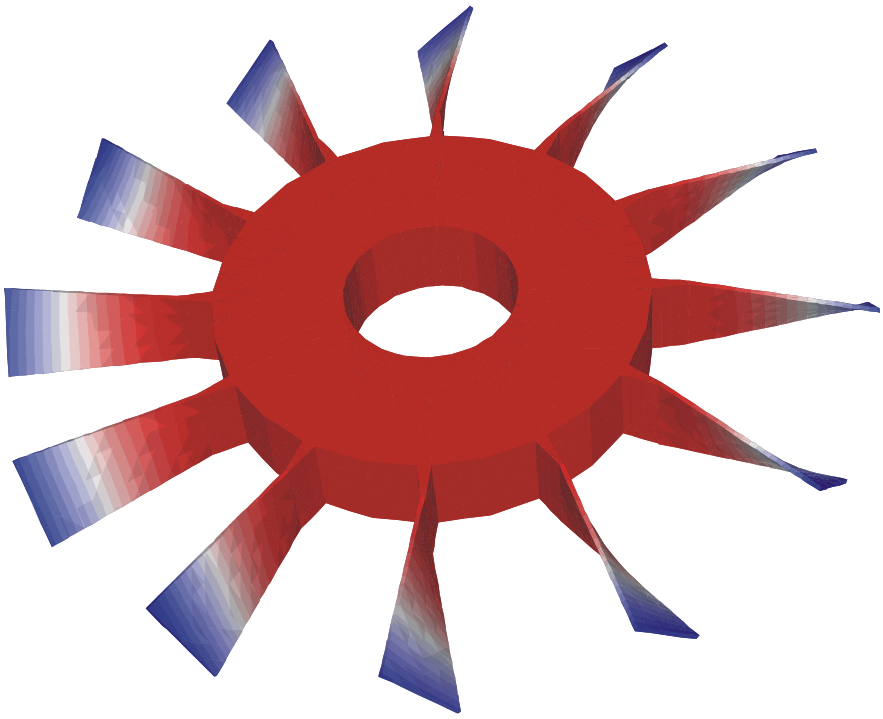


Figura 5.23 – Isofaixas de uma componente de tensões da turbina circular.

6. Conclusões

Este trabalho apresenta um ambiente computacional para modelagem geométrica aplicada à análise por elementos finitos usando multi-regiões e superfícies paramétricas definidas por NURBS. O principal objetivo aqui é gerar modelos 3D para serem usados em simulações numéricas baseadas no Método de Elementos Finitos (MEF). Nessa proposta, a metodologia adotada consiste na combinação de alguns aspectos da modelagem geométrica tal qual a detecção automática de regiões e interseção de superfícies com geração de malhas de elementos finitos. Esses aspectos são integrados em um ambiente que serve como um sistema para modelagem geométrica genérica 3D de problemas de engenharia usando o MEF.

Uma organização de classes, no contexto da programação orientada a objetos, de uma nova versão da estrutura de dados do MG foi apresentada. Essa organização mantém a simplicidade e eficiência de interface com o usuário da versão original, e provê capacidades para detecção automática de regiões e geração de malhas de elementos finitos para modelos com retalhos de superfícies curvos representados por NURBS.

Uma estrutura de dados híbrida é adotada. Nessa estrutura, uma representação CGC do modelo é usada como uma estrutura auxiliar para a modelagem *non-manifold* com detecção de multi-regiões. Essa representação CGC não é mantida durante cada etapa da modelagem geométrica. Para isso, a estrutura de dados anterior do MG foi estendida tal que um modelo CGC possa ser criado em qualquer momento, quando requisitado pelo usuário para detecção de regiões com interseção entre superfícies.

A descrição geométrica das entidades topológicas é comum em ambas representações. Essa descrição geométrica é armazenada em um módulo separado baseado em uma representação NURBS. Tal representação é importante, pois permite ao modelador descrever diversos tipos de curvas e superfícies, inclusive formas cônicas, bem como

possui as descrições dos espaços paramétricos dessas entidades. Em relação ao módulo CGC, a incorporação da representação NURBS permitiu a descrição de arestas e faces curvas. Na versão original da CGC, apenas faces planas são consideradas.

Em relação à interseção entre superfícies, o algoritmo utilizado neste trabalho resolve o problema de interseção usando as malhas definidas no espaço paramétrico das superfícies. Esse algoritmo é baseado em um procedimento onde as malhas de superfícies existentes são convertidas em uma estrutura de dados topológica auxiliar que permite a construção de curvas de *trimming* de forma eficiente. Essa informação topológica auxiliar também é usada para remalhar localmente as malhas das superfícies interceptadas. O algoritmo de interseção gera elementos resultantes da interseção com boas qualidades geométricas, adequados para análise de elementos finitos.

Além do remalhamento local resultante da interseção de superfícies, a metodologia de modelagem da nova versão do MG permite o remalhamento total do modelo de uma forma consistente. A CGC provê consistência entre entidades geométricas e topológicas de um modelo *non-manifold* com multi-regiões. A completa informação topológica e geométrica permite o remalhamento, individualmente ou globalmente, de qualquer retalho de superfície ou região sólida do modelo.

Uma nova versão do algoritmo de interseção entre superfícies foi desenvolvida neste trabalho. Nessa versão, alguns casos especiais não-tratados na implementação original foram considerados. As principais alterações foram realizadas na implementação do algoritmo, mantendo praticamente inalteradas as idéias centrais propostas na versão original.

A modelagem geométrica dá suporte à geração de malhas de elementos finitos. Neste trabalho, foram utilizados algoritmos para geração de malhas não-estruturadas em superfícies e volumes desenvolvidos no mesmo grupo ao qual o trabalho está inserido. Esses algoritmos são baseados em técnicas de decomposição espacial recursiva combinadas com técnicas de avanço da fronteira. A principal motivação para o uso desses algoritmos está na necessidade de atender alguns requisitos específicos que não são tratados diretamente em outros algoritmos encontrados na literatura. Esses algoritmos produzem elementos com boa forma evitando a geração de elementos de má

qualidade. Além disso, os algoritmos geram malhas que são conformes com as discretizações existentes no contorno do domínio. Outra característica que os algoritmos apresentam é a geração de malhas com uma boa transição entre regiões com diferentes tamanhos de elementos, considerando ainda casos onde as superfícies possuem curvaturas acentuadas. O uso, neste trabalho, de tais algoritmos pode ser justificado pela suas capacidades de gerar malhas em domínios arbitrários, importante na modelagem de problemas complexos de engenharia.

As ferramentas apresentadas neste trabalho foram implementadas no modelador MG. Essas ferramentas permitem ao MG modelar uma grande quantidade de problemas tridimensionais (cascas e/ou sólidos), considerando interseções entre superfícies e reconhecimento automático de multi-regiões, bem como a geração de modelos numéricos (malhas e atributos) para análise pelo método dos elementos finitos. Os atributos são gerenciados por um sistema, denominado ESAM, que permite a configuração do modelador para o seu uso em diversos tipos de simulações em engenharia.

6.1 - Principais Contribuições

Baseado nos aspectos descritos acima, as principais contribuições deste trabalho foram:

- O desenvolvimento de um esquema de dados híbrido que oferece suporte à modelagem geométrica de sólidos e provê capacidade para realizar a detecção automática de multi-regiões e interseções de superfícies.
- Incorporação ao modelador MG de uma biblioteca de funções com representações de curvas e superfícies do tipo NURBS, permitindo a implementação de uma grande variedade de tipos de superfícies, bem como a descrição dessas nos seus espaços paramétricos. Essa descrição paramétrica dá suporte à geração de malhas no espaço paramétrico das superfícies e à utilização do algoritmo de interseção de superfícies paramétricas.
- Extensão da biblioteca CGC, que gerencia a modelagem de subdivisões espaciais em 3D, incorporando curvas e superfícies paramétricas do tipo NURBS.

- O tratamento de casos especiais no algoritmo de interseção de superfícies usando malhas paramétricas apresentado por Coelho, permitindo a sua utilização, de forma confiável e eficiente, em uma grande quantidade de problemas de engenharia.
- A inclusão no modelador MG de um sistema configurável para gerenciamento de atributos (ESAM), permitindo o uso do MG na simulação de uma grande quantidade de problemas de engenharia nas mais diversas áreas.
- A integração dos itens descritos acima com algoritmos para geração de malhas, estruturadas e não-estruturadas, de elementos finitos (de superfícies e sólidos), em uma nova versão do modelador MG, provendo-o da capacidade de realizar modelagens geométricas para elementos finitos usando multi-regiões e superfícies paramétricas.

6.2 – Sugestões para Trabalhos Futuros

Como foi dito anteriormente, a idéia central deste trabalho foi propor um ambiente para a modelagem geométrica de problemas de engenharia. As ferramentas propostas nesse ambiente foram implementadas no modelador MG, tornando-o capaz de realizar modelagens de superfícies e sólidos, considerando os problemas de interseção entre superfícies e reconhecimento automático de regiões, e dando suporte à geração de modelos numéricos (malhas e atributos) para elementos finitos.

No entanto, a geração de malhas de elementos finitos não é uma tarefa fácil. Em geral, o processo de geração de malhas demanda tempo e é bastante cansativo, além de exigir uma certa experiência do profissional responsável por esta tarefa. Neste contexto, algoritmos adaptativos para geração de malhas têm se tornado bastante úteis para aumentar a confiabilidade dos procedimentos de análise numérica pelo MEF. Uma proposta para trabalhos futuros é a implementação de uma estratégia que possibilite a geração adaptativa de malhas de elementos finitos no modelador MG. Deve-se observar que as ferramentas implementadas no MG, tais como a estrutura de dados híbrida e os algoritmos para geração de malhas não-estruturadas, já dão suporte a essa geração adaptativa de malhas, pois as malhas de superfícies são construídas nos espaços

paramétricos e as malhas sólidas arbitrárias são construídas a partir das malhas de superfície.

Em três dimensões, o desenvolvimento de análises adaptativas não é fácil, pois nas simulações, a geração de malhas volumétricas e a estimativas de erros numéricos são tarefas relativamente complexas. No entanto, existem alguns trabalhos que apresentam metodologias bidimensionais para considerar os aspectos relacionados à geração de uma forma eficiente. Por exemplo, Cavalcante Neto *et al* [10,11,12], desenvolveu uma estratégia auto-adaptativa bidimensional capaz de fazer a simulação, envolvendo a geração de malhas e métodos adaptativos de uma forma automática. Uma sugestão é estender tal estratégia bidimensional para modelos tridimensionais.

Dessa forma, a estratégia adaptativa seria baseada no refinamento das curvas, malhas nas superfícies e malhas nos sólidos do modelo, respectivamente. Em uma primeira etapa, as arestas do sólido seriam refinadas com base na estimativa de erros do passo anterior e com o auxílio de uma estrutura de dados de árvore binária. Em seguida, seria feito o refinamento das malhas nas superfícies de bordo do sólido usando uma estrutura de dados de subdivisões espaciais recursivas no espaço paramétrico. Nesse passo, o algoritmo para geração de malhas não-estruturadas de superfícies citado no capítulo 2 deve ser usado, pois ele já apresenta ferramentas para a sua utilização em processos adaptativos. No último estágio da estratégia adaptativa, o domínio do sólido seria discretizado com base no algoritmo para geração de malhas volumétricas não-estruturadas mencionado neste trabalho. Esse algoritmo deve ser adaptado para que o refinamento da *octree* auxiliar também seja dependente da estimativa de erros do passo anterior.

O critério para refinamento das malhas adotado nessa estratégia deve ser baseado no erro numérico associado às discretizações. Diversas técnicas são apresentadas na literatura [3,77,78]. Uma dessas técnicas define o estimador de erro baseado na obtenção de valores de tensão melhorados usando alguns processos de recuperação de tensões disponíveis [79,80,81].

Essas técnicas de estimadores de erros necessárias em uma análise adaptativa estão implementadas no programa para análise numérica pelo método dos elementos finitos

Femoop [40]. Esse programa poderia, então, ser utilizado na determinação dos erros numéricos necessários à geração adaptativa de malhas.

Por outro lado, o aumento no tamanho dos modelos tridimensionais tende a consumir grandes recursos computacionais desses programas para análise numérica pelo método dos elementos finitos. Modelos grandes e complexos requerem uma grande quantidade de memória que pode não se ajustar em um computador. Além disso, o alto tempo consumido para obter a solução do sistema de equações pode não ser desejável. Nesse sentido, uma forma de aumentar a velocidade de análise e reduzir o espaço para armazenamento de memória é paralelizar essa etapa computacional. Assim, esquemas paralelos para obter a solução do sistema de equações lineares foram implementados no Femoop [46,47,48]. Então, uma sugestão é utilizar, em grandes modelos, a versão paralela do programa Femoop na obtenção dos erros numéricos necessários para a geração adaptativa de malhas.

Outro ponto que deve ser observado é a geração de modelos sólidos usando o algoritmo para geração de malhas não-estruturadas utilizado neste trabalho. Quando o problema requer um refinamento muito grande da malha, o algoritmo torna-se lento. A solução desse problema é a utilização de sistemas distribuídos (paralelos) para a geração dessas malhas [49].

A capacidade de modelar superfícies usando outros tipos de representações devem ser implementadas no MG. Em particular, superfícies planares, definidas por um número ilimitado de curvas no seu contorno, e superfícies triangulares, definidas por três curvas de bordo, devem ser consideradas. No caso das superfícies triangulares, a implementação usada neste trabalho deve ser substituída por outra que resolva os problemas apresentados quando as curvas de bordo geradoras dessas superfícies não podem ser representadas por curvas do tipo Bézier.

Em relação à geração de malhas em superfícies, deve-se observar que, com exceção do algoritmo para geração de malhas não-estruturadas apresentado neste trabalho, todos os outros utilizados no MG não consideram distorções geométricas no mapeamento da malha gerada no espaço paramétrico da superfície para o espaço tridimensional. Com isso, modelos com uma geometria mais complexa e curvaturas não-uniformes nas suas

superfícies, apresentam distorções nas malhas geradas pelos algoritmos estruturados. Um critério semelhante ao critério que corrige distorções no algoritmo para geração de malhas não-estruturadas também deve ser adotado nesses casos.

Tanto na modelagem de superfícies quanto na modelagem de sólidos, um tópico importante abordado neste trabalho é a obtenção da consistência topológica e geométrica do modelo. Diversos pontos foram discutidos ao longo do trabalho, incluindo a necessidade de garantir a consistência entre entidades do mesmo tipo e adjacentes entre si. No caso dos retalhos de superfícies, quando uma metodologia utilizada na geração da superfície não requer que as curvas de bordo sejam fornecidas como dados de entrada, estas são geradas automaticamente pelo modelador. A determinação dessas curvas de bordo de um retalho de superfície não é uma tarefa difícil, pois a biblioteca NURBS++ permite o cálculo de *isocurvas* associadas às superfícies modeladas.

No caso da modelagem de sólidos, situações onde o sólido é gerado a partir da técnica adotada na geração de elementos tetraédricos em domínios arbitrários, o problema de determinação dos retalhos de superfícies do contorno do sólido é minimizado, pois a entrada de dados requerida pelo algoritmo que implementa essa metodologia é justamente os retalhos das superfícies do bordo do sólido que vai ser gerado. No entanto, as técnicas para geração de sólidos baseadas no *sweep* de retalhos de superfícies, não exigem que o usuário defina *a priori* quais são os retalhos de superfícies que delimitam o sólido. Nestes casos, após a geração dos sólidos, é necessário que os retalhos das superfícies de bordo sejam determinadas automaticamente pelo modelador. Isso, no entanto, não é um processo trivial na metodologia de modelagem de sólidos adotada pelo MG. Esse é um problema que não é considerado neste trabalho. O autor sugere que, em trabalhos futuros, um algoritmo para a determinação dos retalhos de superfícies do bordo de um sólido seja implementado, evitando o surgimento de inconsistências topológicas que aparecem em situações como as descritas acima. A solução para este problema não é fácil, pois provavelmente tais retalhos devem ser calculados a partir da malha de elementos 3D associada ao sólido. Inicialmente, deve-se determinar quais são as faces (lados) dos elementos 3D que possuem apenas um elemento finito adjacente. Essas faces determinadas são aquelas que estão no contorno da malha sólida, devendo-se utilizar as informações topológicas e

geométricas associadas a essas faces para estabelecer uma técnica para geração dos retalhos de superfícies correspondentes. Deve-se observar ainda que este processo deve ser feito de forma ótima, provavelmente utilizando árvores de ordenação espacial para auxiliar nas buscas necessárias.

7.Referências Bibliográficas

[1] *ACIS 3D Geometric Modeler.* *http://www.spatial.com/products/3D/modeling/ACIS.html*.

[2] *Andrade L.N e Ting W.S., "Caminhando sobre uma Interseção de Superfícies com Passos Circulares", Anais do IX SIBGRAPI, pp. 151-158, 1996.*

[3] *Babuska, I. and Rheinboldt, W.C, "A-posteriori Error Estimates for Finite Element Problems", International Journal for Numerical Methods in Engineering, vol. 12, pp. 1597-1615, 1978.*

[4] *Barnhill, R., Farin, G., Jordan, M. and Piper, B.R.; "Surface/surface intersection"; Computer Aided Geometric Design, vol. 4, pp. 3-16, 1987.*

[5] *Barnhill, R. and Kersey S.; "A marching method for parametric surface/surface intersection"; Computer Aided Geometric Design, vol. 7, pp. 257-280, 1990.*

[6] *Bathe K. J, "Finite Element Procedures in Engineering Analysis", Pretice Hall, 1982.*

[7] *Beckmann N., Kriegel H.P., Schneider R. and Seeger B.; "The R*-Tree: An Efficient and Robust Acess Method for Points and Rectangles", In Proceedings of the ACM SIGMOD Conference on Management of Data, pp. 322-332, May 1990.*

[8] *Boing Company. http://ocean.dt.navy.mil/dtnurbs/dtnurbs.htm.*

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

[10] *Cavalcante, Neto, J. B.; "Simulação Auto-Adaptativa Baseada em Enumeração Espacial Recursiva de Modelos Bidimensionais de Elementos Finitos", Dissertação de Mestrado, Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Civil, 1994.*

[11] *Cavalcante Neto, J.B.; Martha, L.F.; Menezes, I.F.M. and Paulino, G.H., "A Methodology for Self-Adaptive Finite Method Analysis Using an Object Oriented Approach", In IV-WCCM: Fourth World Congress on Computational Mechanics, Bueno Aires, Argentina, 20 pages, 1998 (available in CD ROM).*

[12] *Cavalcante Neto, J. B. e Martha, L. F.; "Um Protótipo para um Sistema de Análise Adaptativa em Três Dimensões", Anais do XX CILAMCE, São Paulo, Brasil, vol. 1, pp. 1-9, 1999.*

[13] *Cavalcante, Neto, J. B.; "Geração de Malha e Estimativa de Erro para Modelos Tridimensionais de Elementos Finitos com Trincas", Tese de Doutorado, Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Civil, 1998.*

[14] *Cavalcante Neto, J. B.; Wawrzynek P.A.; Carvalho, M.T.M.; Martha, L.F. e Ingraffea, A.R., "An Algorithm for Three-Dimensional Mesh Generation for Arbitrary Regions with Cracks", In Fifth US National Congress on Computational Mechanics, Colorado, USA, 1999.*

[15] *Cavalcanti, P. R.; "Criação e Manutenção de Subdivisões do Espaço", Tese de Doutorado, Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 1992.*

[16] *Cavalcanti, P.R., Carvalho, P.C.P. and Martha, L.F., "Non-manifold Modeling: An Approach Based on Spatial Subdivision", Computer-Aided Design, vol. 29, no. 3, pp. 209-220, 1997.*

[17] *Coelho, L.C.G e de Souza, C.S.; "Comunicação de Problemas e Soluções Geométricas em uma Interface 3D"; Anais do VIII SIBGRAPI, pp. 233-240, 1995.*

[18] *Coelho, L.C.G., Gattass, M. and Martha, L.F.; "Modeling Techniques to Generate 3D Meshes"; Anais of the XVI CILAMCE, pp. 1270-1274, 1996.*

[19] *Coelho, L. C. G.; "Modelagem de Cascas com Interseções Paramétricas", Tese de Doutorado, Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 1998.*

[20] *Coelho, L. C. G., Gattass, M. and Figueiredo, L. H.; "Intersecting and Trimming Parametric Meshes on Finite-Element Shells", International Journal for Numerical Methods in Engineering, vol. 47, no. 4, pp. 777-800, 2000.*

[21] *Comer D.; "The Ubiquitous B-tree", ACM Computing Surveys, vol. 11, no. 2, pp. 121-131, 1979.*

[22] *Dobkins, D.P. and Laszlo, M.J., "Primitives for the Manipulation of Three-dimensional Subdivisions" Third ACM Symposium on Computational Geometry, pp. 86-99, Waterloo, 1987.*

[23] *DT_NURBS Library.* http://ocean.dt.navy.mil/dtnurbs/dtnurbs.htm.

[24] *Emmerik, M.J.G.M.V.; "A Direct Manipulation Technique for Specifying 3D Object Transformations with a 2D Input Device"; Computer Graphics Forum, vol. 9, pp. 355-362, 1990.*

[25] *Farin, G.E.; "Curves and Surfaces for Computer Aided Geometric Design – A Practical Guide", 3rd ed., Boston: Academic Press, 1993.*

[26] de *Figueiredo, L.H.; "Adaptive Sampling of Parametric Curves"; Graphics Gems V, pp. 173-178, 1995.*

[27] *GSLIB: Geometric Solver Library.* http://www.integrityware.com/iwframe.htm.

[28] *Hoffmann, C. M.; "Geometric & Solid Modeling: An Introduction", Purdue University, Indiana, 1989.*

[29] *Hougthon, E., Emnett, E., Factor, R. and Sabharwal, L.; "Implementation of a Divide-and-Conquer-Method for the Intersection of Parametric Meshes"; Computer Aided Geometric Design, vol. 2(1-3), pp. 173-184, 1985.*

[30] *Lavoie, P. "The Nurbs++ Package - User's Reference Manual – Version 3.0", http://yukon.genie.uottawa.ca/~lavoie/software/nurbs/, Ottawa, 1999.*

[31] *Laszlo, M.J., "A Data Structure for Manipulating Three-dimensional Subdivisions", PhD Thesis, Department of Computer Science, Princeton University, 1987.*

[32] *Lienhardt, P., "Extension of the Notion of Map Subdivisions of a Three-dimensional Space", In STACS'88 Proceedings of the Cinquième Symposium sur les Aspects Thèoriques de L'Informatique, Bordeaux, 1988.*

[33] *Lin, F. and Hewitt, W.; "Expressing Coons-Gordon Surfaces as NURBS", Computer Aided Design, vol. 26, no. 2, pp. 145-155, 1994.*

[34] *Lira W. W. M., "Um Sistema Integrado Configurável para Simulações de Problemas em Mecânica Computacional", Dissertação de Mestrado, PUC-Rio, Departamento de Engenharia Civil, 1998.*

[35] *Lo, S.H.; "Automatic mesh generation over intersecting surfaces"; International Journal for Numerical Methods in Engineering, vol. 38, pp. 943-954, 1995.*

[36] *Lohner, R. and Parikh, P., "Generation of Three-Dimensional Unstructured Grids by the Advancing-Front Method", International Journal for Numerical Methods in Fluids, vol. 8, pp. 1135-1149, 1988.*

[37] *MacNeal, R.H.; "MSC/PATRAN 6 & 7 FEA USER'S MANUAL"; MacNeal Swhwendler Corporation, 1996. http://www.macsch.com:80/bookstore/patran.html.*

[38] *Mäntylä, M., "An Introduction to Solid Modeling Computer", Science Press, Rockville, Maryland, 1988.*

[39] *Martha, L. F. C. R.; "Topological and Geometrical Modeling Approach to Numerical Discretization and Arbitrary Fracture Simulation in Three-Dimensions", PhD Thesis, Cornell University, Ithaca, N.Y., 1989.*

[40] *Martha, L. F.; Menezes, I. F. M.; Lages, E. N.; Parente Jr., E. and Pitangueira, R. L. S., "An OOP Class Organization for Materially Nonlinear Finite Element Analysis", XVII CILAMCE, Pádova, Itália, pp. 229-232, 1996.*

[41] *Mello, U.T. and Cavalcanti, P.R.; "A Topologically-Based Framework for Simulating Complex Geological Processes", Research Report RC21339, IBM T.J. Watson Research Center, Yorktown Heights, New York, 1998.*

[42] *Miranda, A.C. e Martha, L.F., "Mapeamento Transfinito Tridimensional", XX CILAMCE, São Paulo, Brasil, vol. 1, pp. 1-13, 1999.*

[43] *Miranda A. C. O., "Integração de Algoritmos de Geração de Malhas de Elementos Finitos", Dissertação de Mestrado, PUC-Rio, Departamento de Engenharia Civil, 1999.*

[44] *Miranda, A. C. e Martha L.F., "Uma Biblioteca Computacional para Geração de Malhas Bidimensional e Tridimensional de Elementos Finitos", XXI CILAMCE, Rio de Janeiro, Brasil,* vol. 03, pp.09.1-09.15, 2000.

[45] *Moller, P. and Hansbo, P., "On Advancing Front Mesh Generation in Three Dimensions", International Journal for Numerical Methods in Engineering, vol. 38, pp. 3551-3569, 1995.*

[46] *Moretti, C.O.; Bittencourt, T.N. and Martha, L.F. "A Low Cost Distributed System for FEM Parallel Structural Analysis", VECPAR'98 – 3rd International Meeting on Vector and Parallel Processing, Porto, Portugal, pp. 1063-1075, 1998.*

[47] *Moretti, C.O.; Bittencourt, T.N. and Martha, L.F. "A Multiplatform Distributed FEM Analysis System Using PVM and MPI", VECPAR'2000 – 4th International Meeting on Vector and Parallel Processing, Lisboa, Portugal, pp. 819-828, 2000.*

[48] *Moretti, C.O.; Cavalcante Neto, J.B, Bittencourt, T.N. and Martha, L.F. "A Parallel Environment for Three-Dimensional Finite Element Method Analysis", Developments in Engineering Computational Technology, B.H.V. Topping (Editor), Civil-Comp Press, Edinburgh, UK, pp. 283-287, 2000.*

[49] *Moretti, C.O.; "Um Sistema Computacional Paralelo Aplicado à Simulação de Propagação Tridimensional de Fissuras", Tese de Doutorado, Escola Politécnica da Universidade de São Paulo, Departamento de Engenharia de Estruturas e Fundações, 2001.*

[50] *O'Leary, A.; "ANSYS Modeling and Meshing Guide – Realese 5.4"; SAS IP Incorporated, 1998. http://www.ansys.com/ServSupp/Library/library.html.*

[51] *Parasolid: Powering the Digital Enterprise. http://www.plmsolutions-eds.com/products/parasolid.*

[52] *Peraire, J.; Peiro, J.; Formaggia, L.; Morgan, K. and Zienkiewicz, O.C., "Finite Euler Computation in Three-Dimensions", International Journal for Numerical Methods in Engineering, vol. 26, pp. 2135-2159, 1988.*

[53] *Piegl, L.; "On NURBS: A Survey", IEEE Comput. Graph. and Appl., vol. 10, no. 1, pp. 55-71, 1991.*

[54] *Piegl, L. e Tiller, W., "The Nurbs Book", 2nd ed. Springer-Verlag, 1999.*

[55] *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.*

[56] *Preparata, F.P. and Shamos, M.I.; "Computational Geometry: An Introduction"; Springer Verlag, New York, 1990.*

[57] *Pro/ENGINEER API Toolkit. http://www.ptc.com/products/proe/app_toolkit.htm.*

[58] *Requicha, A.G. and Voelcker, H.B., "Constructive Solid Geometry", Technical Report Technical Memo no. 25, Production Automation Project, University of Rochester, Rochester, New York, 1977.*

[59] *Rhinoceros, NURBS Modeling for Windows. http://www.rhino3d.com/.*

[60] *Rossignac, J.R. and O'Connor, M.A., "A Dimensional-independent Model for Pointsets with Internal Structures and Incomplete Boundaries", Geometric Modeling for Product Engineering, pp. 145-180, North Holland, 1990.*

[61] *Rossignac, J.R. and Requicha, A.G., "Constructive Non-regularized Geometry. Computer Aided Design", vol. 23, no. 1, pp. 21-32, 1991.*

[62] *Rumbaugh J., "Object-Oriented Modeling and Design", Prentice-Hall, Englewood Cliffs, 1991.*

[63] *Samet, H., "The Quadtree and Related Hierarchical Data Structure", ACM Computer Surveys, vol. 16, no. 2, pp. 187-260, 1984.*

[64] *Shephard, M. S.; "The Specification of Physical Attribute Information for Engineering Analysis.", Engng. with Comput., vol. 4, pp. 145-155, 1988.*

[65] *Solid Modeling Solutions. http://www.smlib.com/NURBS.htm.*

[66] *Stroustrup B.; "The C++ Programming Language", 3 ed., Addison-Wesley, 1997.*

[67] *Stoyanov, Tz. E.; "Marching along surface/surface intersection curves with an adaptive step length"; Computer Aided Geometric Design, vol. 9, pp. 485-489, 1992.*

[68] *The Manchester Nurbs Library.* http://mvc.man.ac.uk/research/nurbs/library/.

[69] *Ting, W.S., "Considerations about a Minimal Set of Non-manifold Operations", Technical Memo, Technische Hochshule Darmstadt – GRIS, Wilhelminen-strabe, 7 DA-6100, FRG, 1990.*

[70] *Vianna, A.C., "Modelagem Geométrica Completa para Modelos Bidimensionais de Elementos Finitos", Dissertação de Mestrado, PUC-Rio, Departamento de Engenharia Civil, 1992.*

[71] *Yerry, M.A. and Shephard, M.S., "Automatic Three-Dimensional Mesh Generation by Mofified-Octree Technique", International Journal for Numerical Methods in Engineering, vol. 20, pp. 1965-1990, 1984.*

[72] *Watson, D.F., "Computing the n-dimensional Delaunay Triangulation with Application to Voronoi Polytopes", The Computer Journal, vol. 24, no. 2, pp. 167-172, 1981.*

[73] *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.*

[74] *Weiler, K.; "Topological Structures for Geometric Modeling", Ph.D. Thesis, Rensselear Polytechnic Institute, Troy, N.Y., 1986.*

[75] *Weiler, K.; "The Radial-Edge Structure: A Topological Representation for Non-Manifold Geometric Boundary Representation", Geometric Modeling for CAD Applications, North Holland, pp. 3-36, 1988.*

[76] *Wong, V. S.; "Qualification and Management of Analysis Attributes with Application to Multi-Procedural Analysis for Multichip Modules.", Master Thesis, Rensselaer Polytechnic Institute, Troy, New York, 1994.*

[77] *Zienkiewicz O. C.; Kelly, D. W.; Gago, J. and Babuska I., "Hierarchical Finite Element Approaches, Error Estimator and Adaptive Refinement", in J. Whiteman (ed.), Mathematics of Finite Elements and Applications (IV), Academic Press, New York-NY, pp. 311-346, 1982.*

[78] *Zienkiewicz O. C. and Zhu, J.Z., "A Simple Error Estimator and Adaptive Procedure for Practical Engineering Analysis", International Journal for Numerical Methods in Engineering, vol. 24, pp. 337-357, 1987.*

[79] *Zienkiewicz O. C. and Zhu, J.Z., "The Superconvergent Patch Recovery and A Posteriori Error Estimates. Part 1: The Recovery Technique", International Journal for Numerical Methods in Engineering, vol. 33, pp. 1331-1364, 1992.*

[80] *Zienkiewicz O. C. and Zhu, J.Z., "The Superconvergent Patch Recovery and A Posteriori Error Estimates. Part 2: Error Estimates and Adaptivity", International Journal for Numerical Methods in Engineering, vol. 37, pp. 3195-3196, 1994.*

[81] *Zienkiewicz O. C. and Zhu, J.Z., "A Posteriori Error Estimates and Three-Dimensional Automatic Mesh Generation", Finite Elem. Anal. Des., vol. 25, pp. 167-184, 1997.*

[82] *Zienkiewicz, O.C. and Taylor, R.L., "The Finite Element Method", Fifth ed.. vols. 1 and 2, Butterworth-Heinemann, 2000.*

[83] *"MG: Manual do Usuário – Versão 3.0", Grupo de Tecnologia em Computação Gráfica, Pontifícia Universidade Católica do Rio de Janeiro, http://www.tecgraf.puc-rio.br/~lula/mg, Abril, 1996.*