MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

MODELAMENTO PARAMÉTRICO E GERAÇÃO DE MALHA EM SUPERFÍCIES PARA APLICAÇÕES EM ENGENHARIA

por

Fábio Gonçalves Teixeira

Tese para obtenção do Título de
Doutor em Engenharia

Porto Alegre, novembro de 2003

MODELAMENTO PARAMÉTRICO E GERAÇÃO DE MALHA EM SUPERFÍCIES PARA APLICAÇÕES EM ENGENHARIA

por

Fábio Gonçalves Teixeira
Mestre em Engenharia

Tese submetida ao Corpo Docente do Programa de Pós-Graduação em Engenharia Mecânica, PROMEC, da Escola de Engenharia da Universidade Federal do Rio Grande do Sul, como parte dos requisitos necessários para a obtenção do Título de

Doutor em Engenharia

Área de Concentração: Mecânica dos Sólidos

Orientador: Prof. Dr. Guillermo J. Creus

Co-orientador: Prof. Dr. Eduardo Bittencourt

Aprovada por:

Prof. Dr. Luiz Fernando Martha

Profa. Dra. Carla M. Dal Sasso Freitas

Prof. Dr. Armando Miguel Awruch

Prof. Dr. João Luís Dihl Comba

Prof. Dr. Jun Sérgio Ono Fonseca
Coordenador do PROMEC

Porto Alegre, 27 de novembro de 2003.

A minha família

AGRADECIMENTOS

À minha querida esposa, Marlise, pelo apoio e compreensão ao longo dos cinco anos de desenvolvimento deste trabalho e à minha filha, Helena, que tinha apenas 1 ano quando iniciei o doutorado e cresceu ouvindo falar diariamente em Tese.

Aos meus Pais, Deny e Dedé, que me proporcionaram uma formação sólida e sempre apoiaram e incentivaram o meu trabalho.

Aos professores Gillermo J. Creus e Eduardo Bittencourt pela orientação e pelo incentivo na realização deste trabalho.

Ao professor José Luís Aymone pela colaboração na elaboração e análise dos exemplos de hidroconformação e pelo apoio e incentivo no decorrer de todo o trabalho.

Às professoras Anelise Hoffmann e Jocelise Jacques pelo incentivo e apoio, quando me substituíram em várias aulas de graduação para que fosse possível finalizar a Tese.

Ao acadêmico Fernando Bruno pelo incentivo e apoio ao longo de todo o desenvolvimento do trabalho, além da ajuda que prestou no final do trabalho, quando elaborou imagens e sugestões para a apresentação.

RESUMO

Este trabalho apresenta um conjunto de técnicas para a modelagem paramétrica e geração de malhas de superfícies para uso em sistemas de análise e simulações numéricas pelo Método dos Elementos Finitos. Foram desenvolvidos algoritmos para a geração paramétrica de superfícies, para a determinação das curvas de interseções entre superfícies e para a geração de malhas em superfícies curvas e recortadas.

Foram implementas linhas e curvas paramétricas básicas, a partir das quais são geradas superfícies paramétricas de vários tipos que proporcionam uma grande flexibilidade de modelamento geométrico. Curvas e superfícies são geradas e manipuladas de forma interativa. São apresentadas técnicas que simplificam a implementação de linhas e superfícies paramétricas.

Foi desenvolvido um algoritmo para determinar as curvas de interseção entre superfícies paramétricas, que são utilizadas como linhas de recorte (*trimming lines*) para obter geometrias complexas e compostas de várias superfícies. O algoritmo desenvolvido emprega técnicas de subdivisão adaptativa, por *quadtrees*, em função da curvatura local das superfícies. Primeiramente, obtém-se uma aproximação das curvas de interseção no espaço 3D, através da aproximação por triângulos. Os resultados iniciais são refinados e projetados sobre as duas superfícies envolvidas com algoritmos que permitem obter grande precisão. As curvas de interseção finais são mapeadas nos espaços paramétricos das duas superfícies, porém com uma parametrização única, o que facilita a junção com superfícies adjacentes.

Um algoritmo de geração de malha foi desenvolvido para gerar malhas triangulares de qualidade sobre as superfícies curvas e recortadas. O algoritmo utiliza um processo de subdivisão adaptativa por *quadtrees*, similar ao utilizado no algoritmo de interseção, para definir tamanhos de elementos em função da curvatura local. Em seguida, aplica-se um algoritmo tipo *advancing front* para gerar a malha sobre a superfície.

Os algoritmos foram implementados e testados em um ambiente gráfico interativo especialmente desenvolvido para este trabalho. São apresentados vários exemplos que comprovam a eficiência das técnicas e algoritmos propostos, incluindo exemplos de matrizes de conformação mecânica para uso com código de análise METAFOR, análise de sensibilidade para otimização de pré-formas e de modelagem de superfícies compostas recortadas com geração de malhas de qualidade, para uso em análise por Elementos Finitos ou como contorno para geração de elementos tridimensionais.

ABSTRACT

"Parametric modeling and mesh generation on surfaces and engineering applications"

This work presents a set of techniques for parametric modeling and mesh generation on surfaces, to be used with numerical simulations through the Finite Element Method. Algorithms to model parametric surfaces, the intersection between parametric surfaces and to mesh the curved trimmed parametric surfaces are developed and implemented.

A basic set of parametric curves is used to generate parametric surfaces of several types, which provide a great flexibility in geometric modeling. Curves and surfaces are generated and manipulated in interactive way. Techniques to simplify the implementation of parametric curves and surfaces are presented.

An algorithm to compute the intersection curves between parametric surfaces is developed. These curves are used as trimming lines to create complex geometries combining several trimmed surfaces. The algorithm uses an adaptive subdivision technique, by quadtrees, driven by the local curvature of the surfaces. First, an approximation of intersection curves is obtained in 3D space through intersection between triangles that represent locally the surfaces. This approximation is refined and projected onto both surfaces by an algorithm that provides great accuracy. The final intersection curves are mapped in the parametric spaces of the two surfaces involved. The two representations of the same curve in the two surfaces have the same parametric form, made using the actual arc length of the curves. This facilitates the junction with adjacent surfaces and assures the continuity of the meshes.

A mesh generation algorithm is developed to generate high quality triangular meshes on trimmed curved surfaces. The algorithm uses an adaptive subdivision process, by quadtrees, similar to that used in the intersection procedure, to define element sizes according to the local curvature. Finally, an advancing front type algorithm is applied to mesh the surface.

The algorithms were implemented and tested in a graphic interface specially developed. Several examples are presented that confirm the efficiency of the techniques and algorithms proposed. The examples include die-shapes modeling to use with Metafor, sensitivity analysis for preform optimization and trimmed composed surfaces with high quality meshing for Finite Element analyses. The surface meshes can be used as a boundary for the generation of tetrahedral unstructured meshes.

ÍNDICE

LISTA DE SÍMBOLOS

$\hat{A}_j$	Ângulo entre dois vetores tangentes de dois pontos consecutivos sobre uma curva
A_{Max}	Tolerância angular admissível
$\mathbf{B}$	Vetor da base ortonormal do Frame da superfície Sweep
$B_i(t)$	Função de peso
$\mathbf{C}(t)$	Curva paramétrica
$\mathbf{C}_i$	i-ésima curva paramétrica
$\mathbf{C}_i(u)$	Curva de bordo de superfície correspondente à direção *u*
$\mathbf{C}_i(v)$	Curva de bordo de superfície correspondente à direção *v*
D_{Max}	Dimensão máxima dentre as duas superfícies envolvidas na interseção
$\mathbf{D}_t(t)$	Função derivada de uma curva paramétrica
$\mathbf{D}_u(u,v)$	Função derivada de uma superfície paramétrica na direção *u*
$\mathbf{D}_v(u,v)$	Função derivada de uma superfície paramétrica na direção *v*
$\boldsymbol{e}$	Eixo de rotação
E_i	Erro relativo dos pontos de interseção
$f_C(t)$	Equação paramétrica de linha de interseção no espaço paramétrico de uma superfície
f_{RP}	Fator local de correção do espaço real para o espaço paramétrico
$\mathbf{G}$	Matriz de transformação para o Sistema de Referência Global
$\mathbf{L}$	Matriz de transformação para um Sistema de Referência Auxiliar (Local)
L_j	Comprimento do j-ésimo segmento de linha paramétrica
L_{Max}	Tamanho máximo de Elemento na geração de malha
L_n	Comprimento de uma linha de interseção até o n-ésimo ponto
$\mathbf{N}$	Vetor da base ortonormal do Frame da superfície Sweep
$\mathbf{N}(u,v)$	Vetor normal a uma superfície nas coordenadas *u* e *v*
$\mathbf{N}_i$	Vetor normal à superfície no centro do trecho correspondente à i-ésima célula da *quadtree*
n_u	Número de divisões da superfície na direção *u*
n_v	Número de divisões da superfície na direção *v*
$\mathbf{N}_{ij}$	Vetor normal à superfície nos vértices do trecho correspondente à i-ésima célula da *quadtree*

$\mathbf{P}$	Primeiro ponto de um segmento do Front
$\mathbf{P}_i$	Representação vetorial do i-ésimo ponto de uma série
$\mathbf{P}_{\mathrm{S}}^{\mathrm{p}}$	Ponto projetado sobre uma superfície em coordenadas paramétricas
$\mathbf{P}_{S1}^{i}$	i-ésimo ponto de interseção projetado sobre S_1
$\mathbf{P}_{S2}^{i}$	i-ésimo ponto de interseção projetado sobre S_2
$\hat{P}_i$	Planura de um trecho de superfície correspondente à i-ésima célula da *quadtree*
$\hat{P}_{Min}$	Planura mínima
$\mathbf{Q}$	Segundo ponto de um segmento do Front
$\mathbf{R}$	Ponto que completa um Elemento da malha gerada (**PQR**)
R	Raio de arco ou de circunferência
$\mathbf{R}_1$	Ponto que define o limite inferior de volume envolvente
$\mathbf{R}_2$	Ponto que define o limite superior de volume envolvente
$\mathbf{S}(u,v)$	Representação de uma superfície paramétrica
t	Parâmetro de uma curva paramétrica
$\mathbf{T}$	Vetor da base ortonormal do Frame da superfície Sweep
T_1	Triângulo de um trecho da superfície S_1 (primeira a ser clicada na interseção)
T_2	Triângulo de um trecho da superfície S_2 (segunda a ser clicada na interseção)
t_c	Parâmetro t corrigido para uma curva paramétrica específica
t_i	Parâmetro t do i-ésimo ponto de uma curva paramétrica
t_m	Parâmetro t modificado
$\mathbf{T}(u)$	Matriz de transformação paramétrica (ex: Rotação)
$\mathbf{T}_u(u,v)$	Vetor tangente à superfície na direção u e nas coordenadas u e v
$\mathbf{T}_v(u,v)$	Vetor tangente à superfície na direção v e nas coordenadas u e v
u	Parâmetro de superfície paramétrica
v	Parâmetro de superfície paramétrica
x	Eixo de coordenadas x
$x(u,v)$	Equação paramétrica de uma superfícies correspondente ao eixo x
$\mathbf{X}_C$	Vetor tangente a uma curva sobre uma superfície no espaço paramétrico desta
$\mathbf{X}_{\mathrm{i}}$	Vértice de *patch* de superfície
$\mathbf{X}_i^n$	Série de nós incidentes em um nó da malha
$\mathbf{X}_0$	Nó da malha a ser rearranjado em função dos nós incidentes

x_L	Eixo *x* local
y	Eixo de coordenadas *y*
$y(u,v)$	Equação paramétrica de uma superfícies correspondente ao eixo *y*
y_L	Eixo *y* local
$\mathbf{Y}_M$	Direção de propagação da malha
z	Eixo de coordenadas *z*
$z(u,v)$	Equação paramétrica de uma superfícies correspondente ao eixo *z*
z_L	Eixo *z* local
$\mathbf{Z}_P$	Direção *z* do espaço paramétrico
$\overline{\alpha}$	Fator de qualidade geométrica médio de uma malha
α_{Min}	Fator de qualidade geométrica mínimo de uma malha
α_{90}	Percentual de Elementos com fator de qualidade igual ou superior a 0,9
$\alpha(t)$	Função paramétrica para um ângulo
α_i	Ângulo inicial
α_f	Ângulo final
$\phi(u)$	Função paramétrica para ângulo de rotação
ϕ_T	Ângulo total de rotação

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

1. INTRODUÇÃO

A análise numérica e a simulação computacional, baseadas no Método dos Elementos Finitos, tornaram-se ferramentas tão importantes no auxílio ao projeto de Engenharia, que a industria moderna não pode mais prescindir de seu uso no desenvolvimento de novos projetos e produtos. Bons exemplos disto estão nas industrias: metal-mecânica, automobilística, naval e aeroespacial, que utilizam procedimentos de análise e simulação numérica desde o projeto até os processos de fabricação das peças.

Neste contexto, a modelagem geométrica computacional (CAGD[1]) ocupa um lugar importante, pois constitui a primeira etapa do processo de simulação numérica, onde a geometria dos objetos envolvidos, na forma de malhas, constitui um dos principais dados de entrada de um sistema de análise por Elementos Finitos.

Os processos de geração de malha de Elementos Finitos, normalmente, necessitam de um modelo geométrico (superfícies ou sólidos) que define os domínios sobre os quais a malha deve ser gerada. No entanto, o modelamento geométrico e a geração de malhas constituem, cada um, campos ativos de pesquisa há décadas.

É comum os desenvolvedores de códigos de análise recorrerem a pacotes gráficos comerciais ou mesmo a programas de análise comerciais, que possuem pré-processadores, para gerar modelos e malhas. Sistemas como ANSYS (Moaveni, 1999), Gid (Rodriguez, 2000), PATRAN (MSC Software, 2001), I-DEAS (Shih, 2000) e NASTRAN (Schaeffer, 1998) são exemplos de programas que realizam as tarefas de geração de malha e análise.

Se por um lado a utilização desses sistemas muitas vezes viabiliza as pesquisas e o desenvolvimento de novas tecnologias, por outro pode limitar tais pesquisas às capacidades de geração dos programas utilizados. Assim, os pesquisadores são levados a adaptar suas metodologias de pesquisa para que possam utilizar o *software* disponível em seus laboratórios e instituições. Além disso, muitas vezes os modeladores e geradores comerciais não publicam dados do modelo geométrico, em seus arquivos de exportação, que podem ser importantes em determinadas situações. Um bom exemplo disto é o código para simulação de processos de conformação mecânica METAFOR (Bittencourt, 1994), que não tem um pré-processador, e normalmente utiliza malhas geradas por pacotes comerciais. No entanto, o METAFOR também necessita da definição dos modelos geométricos das Matrizes de Conformação, através de superfícies paramétricas. Dados estes que os geradores citados não publicam. O usuário do

[1] Computer Aided Geometric Design

METAFOR é obrigado, então, a entrar manualmente com estes dados através de um editor de textos, o que é um processo lento e impreciso, pois o modelamento não se dá de forma interativa.

Outro aspecto a ser analisado, é que a dependência do uso de pacotes comerciais pode inviabilizar determinados projetos em função do custo cada vez mais alto das licenças. A transferência de tecnologia para o setor produtivo também fica prejudicada devido à impossibilidade de distribuição, mesmo que limitada, de programas comerciais.

Portanto, o domínio e o desenvolvimento de novas tecnologias de modelamento geométrico tridimensional e de geração de malhas é importante para o desenvolvimento da Pesquisa na área de simulação e análise numérica para Engenharia, permitindo o desenvolvimento de técnicas e de ferramentas adaptadas às realidades e objetivos específicos de cada problema.

É neste contexto que se insere esta tese. Foram analisadas e desenvolvidas técnicas para o modelamento geométrico e geração de malha de Elementos Finitos em superfícies paramétricas. Tais técnicas envolvem: o modelamento paramétrico de curvas e superfícies, algoritmo para cálculo das curvas de interseções entre superfícies e algoritmo para a geração de malha não estruturadas em superfícies curvas recortadas. Com o objetivo de implementar os algoritmos desenvolvidos neste trabalho, optou-se pela construção de um ambiente computacional gráfico e interativo, o T-CADE. Todos os algoritmos desenvolvidos neste trabalho foram implementados e testados no programa, o que proporcionou grande agilidade no processo de desenvolvimento.

O trabalho começou com o desenvolvimento do ambiente gráfico, que utiliza conceitos básicos de computação gráfica para visualizar e manipular objetos em 3D. Em seguida foram implementas as linhas e superfícies paramétricas que podem ser modeladas de forma interativa. Nesta etapa, cumpriu-se um dos primeiros objetivos deste trabalho que é a geração de matrizes de conformação mecânica, para uso no METAFOR, de forma visual, interativa e precisa [Teixeira, 2000].

Na etapa seguinte, desenvolveu-se e implementou-se o algoritmo para a determinação de interseção entre superfícies paramétricas. Utilizando técnicas de subdivisão adaptativa e algoritmos de refinamento e mapeamento paramétrico, foi possível desenvolver um algoritmo que determina as linhas de interseção entre duas superfícies paramétricas quaisquer dispostas de forma arbitrária no espaço. Os resultados são obtidos com grande precisão mapeados nos espaço paramétrico das duas superfícies, permitindo a utilização das linhas de interseção como linhas de recorte (*trimming*), para definir sub-domínios para o algoritmo de geração de malha. O algoritmo determina, ainda, possíveis interseções entre as linhas resultantes

de múltiplas interseções. O algoritmo para cálculo de interseções é baseado em técnicas existentes em trabalhos anteriores, mas possui importantes contribuições originais em todas as suas etapas.

A última etapa de desenvolvimento deste trabalho consistiu no desenvolvimento de um algoritmo para a geração de malhas não-estruturadas em superfícies paramétricas. A idéia era que o processo de geração da malha fosse sensível às diferentes curvaturas das superfícies. Foi desenvolvido, então, uma algoritmo tipo Frontal que gera malhas em domínios arbitrários, definidos por recortes de interseção ou pelo usuário, sobre superfícies curvas ou planas. A qualidade da malha é controlada por uma malha de fundo tipo *quadtree*, que controla os tamanhos dos elementos em função da curvatura local. O algoritmo é baseado nos trabalhos de Lee e Hobbs (1999), Miranda e Martha (2002) e Lau e Lo (1996), mas também utiliza procedimentos originais.

Esta tese descreve os algoritmos desenvolvidos e implementados para o modelamento geométrico interativo de linhas e superfícies paramétricas, o algoritmo para a determinação de linhas de interseção entre superfícies paramétricas e o algoritmo para a geração de malha em superfícies paramétricas recortadas. Tudo isto com o objetivo fundamental de modelar dados para análise de problemas de Engenharia pelo método dos Elementos Finitos, incluindo a análise não-linear de problemas de grandes deformações, que é o caso da conformação mecânica. São apresentados vários exemplos que demonstram o potencial dos algoritmos desenvolvidos.

1.1. REVISÃO BIBLIOGRÁFICA

Nesta seção, será apresentada uma revisão bibliográfica sobre os principais tópicos abordados nesta tese que são: Superfícies Paramétricas, Interseções de Superfícies e Geração de Malha em Superfícies.

1.1.1. Superfícies Paramétricas

A representação paramétrica de curvas e superfícies constitui uma ferramenta corrente em computação gráfica, principalmente em programas CAD. Técnicas que foram desenvolvidas para modelar fuselagens de aviões e carrocerias de automóveis são utilizadas hoje em diversas áreas da computação gráfica.

Uma das mais conhecidas representações paramétricas de superfícies é o Bézier *patch*[2], desenvolvido de forma independente, em 1962, por Pierre Bézier [Bézier, 1966 e 1972], para uso no projeto das carrocerias dos carros da Renault, e por De Casteljau, em 1959, para a Citroën. O nome foi atribuído a Bézier por seu trabalho ter sido divulgado antes. Seu CAD, o UNISURF [Bézier, 1971], foi utilizado no projeto da maioria dos modelos de carros da Renault nos anos 70.

Posteriormente, foram desenvolvidas outras formulações para representação paramétrica de superfícies. *Patches* B-Splines [DeBoor,1972 e Riesenfeld, 1973] e *Non-Uniform Rational B-Splines*, conhecidas como NURBS [Versprille, 1973, e Tiller, 1983 e 1986], são exemplos da evolução da representação paramétrica de superfícies e permitem o modelamento de geometrias complexas, principalmente superfícies do tipo *free-form*. As superfícies Coons [Coons, 1964 e 1967] são outra forma de representação paramétrica que utilizam interpolações entre curvas de bordo que podem ser de qualquer tipo, desde que possuam representação paramétrica.

A geração recursiva de *Patches* B-Spline apresentada em Catmull e Clark, 1978, representou uma grande evolução na geração de superfícies com múltiplos *Patches* a partir de uma malha de pontos. Vários sistemas de modelagem comerciais como o 3D Studio e Maya têm implementado em seus códigos este tipo de superfície.

Os *patches* gerados por *Sweep* [Bloomenthal and Riesenfeld, 1991; Coquillart, 1987; Rogers, 1990] são outra forma de representação paramétrica de superfícies. Neste caso, a superfície é gerada pelo deslocamento de uma curva perfil sobre uma curva caminho. O tipo de deslocamento efetuado pela curva define o tipo de *Sweep*. Siltanen e Woodward, 1992, e Ueng e LAI, 1998, também abordam esse tema. As superfícies de revolução podem ser consideradas como um caso particular de *Sweep*.

Neste trabalho, foi implementado um conjunto básico de superfícies paramétricas constituído pelos seguintes tipos de superfícies: Coons, Sweep, Revolução, Regradas, Bilinear e Planar. Destas, somente a Bilinear não está baseada em curvas de bordo, também paramétricas. O conjunto de superfícies implementado proporciona uma capacidade de modelamento compatível com os objetivos deste trabalho.

---

[2] *Patch* é o nome usual para porções limitadas de superfícies paramétricas, normalmente correspondente ao domínio paramétrico completo. Neste trabalho, trechos e *patches* têm o mesmo significado. *Pacthes* recortados são denominados de retalhos.

1.1.2. Interseções entre Superfícies

O problema de interseção de superfícies é um tema complexo e tem sido um campo ativo de pesquisas por mais de três décadas. Existem diversos trabalhos que tratam deste assunto, com abordagens específicas que podem depender dos tipos das superfícies envolvidas e das aplicações a que se destinam. Segundo Hoschek e Lasser, 1993, um algoritmo de interseção ideal deveria ter as seguintes características:

- **Precisão numérica** – deve ser adequada a aplicação a que se destina;
- **Robustez** – para determinar todas as linhas de interseção, *loops* e singularidades, quando houver, independentemente do tipo e posição das superfícies;
- **Velocidade de processamento** – deve ser compatível com a aplicação;
- **Autocontrole** – não deve necessitar qualquer interatividade ou ajuda do usuário para a sua correta execução.

No entanto, tais características são conflitantes. No desenvolvimento de um algoritmo de interseção de superfícies, deve-se procurar um equilíbrio adequado entre estas características de acordo com a aplicação desejada e para que o mesmo tenha utilidade prática. Os trabalhos de Hoschek e Lasser, 1993, Krishnan *et al.*, 1994, e Andrade, 1998, descrevem, pelo menos, cinco grandes grupos de tipos algoritmos para determinar as linhas de interseção entre superfícies: algoritmos analíticos, algoritmos de discretização, algoritmos de continuação, algoritmos de caminhada, algoritmos de subdivisão. Há, também, técnicas que combinam características de diferentes categorias de algoritmos e são genericamente designadas de algoritmos híbridos [Coelho, 1998; Coelho *et al*., 2000; Lira *et al.*, 2002].

1.1.2.1. Algoritmos algébricos ou analíticos

Os métodos analíticos procuram resolver o problema de interseção através da solução analítica do problema:

$$f - g = 0 \tag{1.1}$$

onde f e g são as funções que definem as duas superfícies. A solução é relativamente simples se as formulações das duas superfícies estão descritas por funções do tipo $z = f(x, y)$ e $z = g(x, y)$. Neste caso, é possível obter as projeções das linhas de interseção sobre os planos xy, xz e yz por eliminações algébricas. A partir daí podem-se obter representações paramétricas das linhas de interseção.

Se as formulações são implícitas ($f(x, y, z) = 0$ e $g(x, y, z) = 0$), o problema de interseção resulta em um sistema de equações não-lineares que pode ser resolvido por algum método numérico, como o de Newton-Raphson, técnicas de geometria diferencial [Asteasu, 1988] ou técnicas algébricas [Owen *et al.*, 1987]. Quando apenas uma superfície possui representação implícita e a outra é paramétrica, o problema ainda é de simples solução. Deve-se inserir a formulação paramétrica na formulação implícita para se chegar, novamente, a um sistema de equações não-lineares. No caso de superfícies paramétricas, procura-se tornar implícita a representação de, pelo menos, uma das superfícies. A idéia é, através de transformações algébricas, remover os parâmetros u e v da formulação [Sederberg, 1987]. Com exceção de superfícies muito simples, este tipo de algoritmo tem custo computacional elevado, o que torna difícil sua aplicação prática [Hoschek e Lasser, 1993]. Portanto, este tipo de método revela-se impróprio quando se utilizam superfícies genéricas e complexas no modelamento tridimensional.

1.1.2.2. Algoritmos de discretização

As técnicas de discretização (*lattice evaluation*) reduzem o grau de complexidade do problema de interseção, pois determinam as interseções de linhas de uma superfície com a outra superfície para determinar pontos sobre a curva interseção [Barnhill *et al.*, 1987]. As linhas de uma superfície são determinadas fixando uma das coordenadas paramétricas. Deste modo, é possível construir famílias de curvas para diferentes constantes paramétricas. Assim, é possível determinar conjuntos de pontos da curva de interseção pela solução de um sistema não-linear do tipo $f(u,Cv)=g(s,t)$, utilizando técnicas numéricas. A constante C deve ser substituída nas coordenadas paramétricas das duas superfícies, gerando um conjunto de sistemas não-lineares que representam grades de curvas sobre as superfícies: $f(Cu,v)=g(s,t)$, $f(u,Cv)=g(s,t)$, $f(u,v)=g(Cs,t)$ e $f(u,v)=g(s,Ct)$. Cada um dos C's terá uma faixa de variação que dependerá do método específico a ser utilizado. A precisão deste tipo de método depende do incremento aplicado na obtenção de C e das técnicas numéricas aplicadas para solucionar os sistemas não-lineares.

1.1.2.3. Algoritmos de continuação

Os métodos de continuação utilizam sistemas de equações diferenciais, obtidos a partir das equações paramétricas das superfícies e de suas características geométricas, para determinar as linhas de interseção. Os sistemas de equações são resolvidos por técnicas

numéricas e o desempenho do método depende das aproximações iniciais e da complexidade da curva interseção. Problemas como singularidades e ramificações devem ser tratados de forma particular. Patrikalakis, 1991, e Abdel-Malek e Yeh, 1996, utilizam técnicas de continuação em seus trabalhos.

1.1.2.4. Algoritmos de marcha

Os métodos de marcha utilizam técnicas de avanço incremental sobre a curva de interseção. Estes métodos necessitam de pontos iniciais sobre a curva de interseção a partir dos quais são obtidos novos pontos em função da direção da tangente da curva que deve ser avaliada. A obtenção de pontos iniciais é uma das etapas críticas destes métodos e pode ser feita através de outros métodos de interseção, como o de subdivisões, utilizado por Barnhill e Kersey, 1990, e Andrade, 1998, ou através de algoritmos específicos. O processo de marcha é a outra etapa crítica deste tipo de método. Alguns trabalhos utilizam uma aproximação do vetor tangente da linha de interseção para determinar a direção de marcha. Stoyanov, 1992, aproxima localmente a curva de interseção por uma parábola cujos coeficientes são obtidos através de derivadas parciais das superfícies, gerando sistemas de equações não-lineares. Outra abordagem é a utilização de um círculo osculador, Wu e Andrade, 1999, para a determinação do próximo ponto de interseção.

1.1.2.5. Algoritmos de subdivisão

Os algoritmos de subdivisão dividem as duas superfícies em muitas partes, em seguida procuram-se aquelas partes que se interceptam. O cálculo dos pontos de interseção pode ser feito por aproximações lineares, considerando-se que os trechos resultantes são quase planos. Deste modo, a interseção se reduz, localmente, ao caso de interseção plano/plano. Os métodos diferem na forma como a subdivisões são feitas e em como são determinadas as interseções em cada trecho. Os critérios de parada do processo constituem outro aspecto importante dos algoritmos de subdivisão.

Este tipo de algoritmo envolve normalmente três etapas: subdivisões recursivas das superfícies até atingir um determinado nível (que depende do processo), determinação dos pontos das linhas de interseção e reordenação dos pontos para formarem as linhas de interseção. Cada uma destas etapas pode ter características e desdobramentos que dependem do algoritmo utilizado. A etapa de subdivisão pode ter diversas abordagens. Os primeiros algoritmos deste tipo realizavam subdivisões uniformes ao longo de todas as superfícies [Griffiths, 1975]. Este tipo de algoritmo era custoso e ineficiente, praticamente, inviabilizando o seu uso. Para reduzir memória

e tempo de processamento, em geral, utilizam-se *quadtrees* não-uniformes para subdividir as superfícies no espaço paramétrico e *bounding boxes* ou volumes envolventes (*bounding volumes*) que envolvem cada trecho correspondente no espaço 3D real. Com isto, é possível identificar aqueles trechos onde há potencial de ocorrer interseção. Somente aqueles trechos cujos volumes envolventes possuem interseção com os volumes envolventes da outra superfície são candidatos a conterem as linhas de interseção. Os outros trechos podem ser descartados, o que restringe os locais de busca pelas linhas de interseção. Os trechos cujos volumes interferem com os volumes da outra superfície são subdivididos nas duas superfícies e o processo se repete, verificando-se interferências e eliminando-se os trechos sem ocorrências. Este é o princípio *Divide-and-Conquer*[3]. Este processo é denominado de algoritmo de subdivisão adaptativa não uniforme. Depois de várias subdivisões sucessivas, o conjunto de volumes restantes envolve estreita e completamente as linhas de interseção. O tamanho destes volumes envolventes restantes pode ser tão pequeno quanto se queira, podendo até ser utilizado como critério de parada do processo de subdivisão. Outro critério de parada de subdivisão pode ser a curvatura local do trecho [Houghton *et al.*, 1985]. Assim, se a curvatura é menor que um certo valor, aquele trecho não é mais subdivido.

Os volumes envolventes (VE) podem ser, basicamente, de dois tipos: alinhados com os eixos coordenados ou orientados segundo o trecho correspondente da superfície. Os VE's alinhados aos eixos são também denominados *min/max box* [Houghton *et al.*, 1985]. Este tipo de VE é muito simples de calcular e de computar as interferências. Os VE's orientados, também denominados como *tight bounding volumes* (volumes envolventes justos) no trabalho de Barth e Huber, 1999, podem melhorar o processo em termos de velocidade de convergência, mas são custosos em termos computacionais. Apesar de otimizar o número de subdivisões, o custo computacional para montar os volumes envolventes justos e computar as interferências entre os mesmos é elevado, o que acaba não refletindo em vantagem significativa no tempo de processamento. O trabalho de Barth-Huber cita uma redução de apenas 7% no tempo total de processamento.

A forma como é controlado o processo de subdivisão varia. Gleicher e Kass, 1992, utilizaram a aritmética intervalar (AI) para controlar o processo de subdivisão. Figueiredo, 1996, adaptou o algoritmo proposto por Gleicher-Kass com o uso da aritmética afim (AA) [Comba e Stolfi, 1993], e obteve resultados melhores em termos de velocidade de convergência e de número de subdivisões em comparação com os resultados obtidos com AI.

---

[3] Dividir e conquistar.

O traçado de linha de interseção após o processo de subdivisão pode ser feito através de uma aproximação plana dos trechos restantes da subdivisão. Estes trechos são divididos em triângulos, que são sempre planos. As interseções entre os triângulos dos trechos cujos VE's se interceptam determinam aproximações das linhas de interseção das duas superfícies. Dependendo da tolerância utilizada nos critérios de parada, é necessário utilizar algum processo numérico para refinar os resultados obtidos. Outra abordagem, é obter os pontos de interseção diretamente a partir dos trechos, tal como os centros dos mesmos, desde que os critérios de precisão tenham sido atingidos. Estes critérios variam, mas um exemplo pode ser a distância entre os centro de trechos próximos das duas superfícies [Andrade, 1998].

A grande vantagem deste tipo de algoritmo é a total independência em relação aos tipos de superfícies envolvidas e à complexidade da linha de interseção, além de não necessitar de aproximações iniciais. Estas razões foram determinantes na escolha deste tipo de algoritmo para ser implementado neste trabalho.

1.1.3. Geração de Malha Não-Estruturada em Superfícies Paramétricas

Entre os primeiros trabalhos de geração de malhas não-estruturadas bi e tridimensionais podem-se citar: George, 1971; Carnet, 1978; Löhner e Parikh, 1988 e Peraire *et al*., 1988. Estes trabalhos abordam a geração de malhas planas de triângulos e 3D com tetraedros (sólidos) e introduziram a técnica *advancing front* ou Frontal, um procedimento interativo que gera malhas em um domínio a partir de sua fronteira. Outro tipo de técnica para a geração de malhas não-estruturadas é a triangularização de Delaunay [Cavendish *et al*., 1985; Lo, 1989; Rebay, 1991; Marcum e Weatherill, 1995b; Mavriplis, 1995]. A maioria dos trabalhos utilizam estas técnicas para a geração de malhas em domínios planos e sólidos. Os primeiros trabalhos que trataram da geração de malha em superfícies curvas foram: Zienkiewicz e Phillips, 1971; Cohen, 1980; Ghansemi, 1982; Haber e Abel, 1982 e George, 1991.

Mais recentemente, diversos trabalhos têm tratado da geração de malhas em superfícies paramétricas, empregando abordagens tipo Delanay [George e Borouchaki, 1998] e Frontal, como Lau e Lo, 1996, que gera malhas em retalhos de superfícies paramétricas projetando triângulos sobre as superfícies. Cuillière, 1998, trabalha com técnicas adaptativas, definidas inicialmente para domínios planos, e estendida para superfícies paramétricas. Marcum e Gaither, 1999, utilizam a técnica Frontal com um esquema para a regularização de domínios compostos por múltiplos *patches*, trabalhando como um único *Patch*.

Alguns trabalhos utilizam malhas de fundo (*backgraund mesh*) para controlarem o tamanho dos elementos durante o processo de geração Frontal, é caso de Lau e Lo, 1996; Canann

et al., 1997; Lee e Hobbs, 1999; Owen e Saigal, 1999 e Miranda e Martha, 2002. Há ainda, alguns algoritmos que utilizam analogias físicas para controlar o tamanho dos elementos. Shimada, 1998, apresenta uma técnica de geração de malha baseada em *bubble packing*, que utiliza uma campo de bolhas no espaço paramétrico e uma analogia física de equilíbrio entre as mesmas para gerar as malhas triangulares. O trabalho de Borouchaki *et al*., 2000, é semelhante ao de Shimada, e emprega um esquema para a minimização da variação dos tamanhos dos elementos, gerando malhas de grande qualidade.

Nesta tese, optou-se por um algoritmo tipo Frontal devido, principalmente, à capacidade de gerar malhas em domínios de contornos arbitrários e com vários recortes.

1.2. ORGANIZAÇÃO DA TESE

Esta tese está organizada em 6 Capítulos. O Capítulo 2 apresenta as técnicas representação paramétricas das curvas e superfícies desenvolvidas e implementadas no T-CADE, assim como o cálculo de derivadas numéricas e de vetores normais.

O Capítulo 3 apresenta o algoritmo para o cálculo das linhas de interseção entre superfícies paramétricas, incluindo os vários procedimentos envolvidos nas 4 etapas que constituem o algoritmo. São apresentados vários exemplos que comprovam a eficiência do algoritmo.

O Capítulo 4 descreve o algoritmo de geração de malha, baseado na técnica Frontal, para superfícies paramétricas recortadas planas ou curvas. São apresentadas as 5 etapas que constituem o algoritmo. No final, uma bateria de exemplos mostra resultados de geração de malha para vários tipos de superfícies.

O Capítulo 5 trata da implementação computacional e da interface do T-CADE, incluindo as principais classes utilizadas.

No Capítulo 6, apresentam-se exemplos de modelamento geométrico e geração de malha de superfícies compostas de vários *Patches* com problemas de interseção e múltiplos domínios, geração de matrizes de conformação e análise de sensibilidade (METAFOR), para uso em otimização de forma.

Finaliza-se o trabalho com o Capítulo 7, onde são apresentadas as principais conclusões e contribuições, além de sugestões para a continuidade da pesquisa.

2. MODELAMENTO GEOMÉTRICO DE CURVAS E SUPERFÍCIES

Neste Capítulo, são abordadas as representações paramétricas das curvas e superfícies implementadas no T-CADE, para o modelamento geométrico tridimensional de superfícies. Primeiramente, são apresentadas representações paramétricas de curvas e, em seguida, as representações paramétricas de superfícies. Propriedades, como vetores tangentes e normais, também são definidas neste Capítulo. Aspectos relativos à implementação computacional destes objetos são tratados no Capítulo 5.

A representações paramétricas constituem uma forma robusta para a representação computacional de objetos geométricos como curvas e superfícies. O uso de representações paramétricas é uma importante ferramenta no modelamento geométrico tridimensional para a análise de problemas de Engenharia, aliando precisão geométrica, com possibilidades praticamente infinitas de geometrias, a uma grande simplicidade de implementação e manipulação computacional.

2.1. REPRESENTAÇÃO PARAMÉTRICA DE CURVAS

As representações paramétricas de curvas, em geral, utilizam pontos de controle e funções de peso (*blending functions*) que estabelecem relações de proporcionalidade entre os mesmos através de um único parâmetro (t). Uma representação genérica de uma curva seria:

$$\mathbf{C}(t) = \sum_{i=0}^{n} B_i(t) \cdot \mathbf{P}_i \tag{2.1}$$

onde $B_i(t)$ são as funções de peso, $\mathbf{P}_i$ os pontos de controle utilizados para representar a curva e t é um parâmetro que varia dentro de uma determinada faixa, em geral [0,1]. Representações como a da Eq. 2.1 consistem de 3 equações, uma para cada dimensão (x, y e z). Para este trabalho, foram implementados 4 tipos de curvas paramétricas: Reta, Arco/Círculo, Spline e Polilinhas.

2.1.1. Representação Paramétrica de um Segmento de Reta

A representação paramétrica da Reta é muito simples, mas torna possíveis representações de segmentos de Reta dispostos em qualquer posição no espaço tridimensional de forma muito robusta. A parametrização é feita através de funções lineares que interpolam dois pontos de controle, que constituem as duas extremidades do segmento.

Pode-se representar um segmento de reta na seguinte forma paramétrica:

$$\mathbf{C}(t) = (1-t)\cdot\mathbf{P}_1 + t\cdot\mathbf{P}_2 \tag{2.2}$$

onde $t \in [0,1]$ e $\mathbf{P}_1$ e $\mathbf{P}_2$ são os pontos de controle que coincidem com as extremidades do segmento (Fig. 2.1). Devido à natureza geométrica da reta, valores de *t* fora do intervalo original também são aplicáveis à Eq. 2.2, o que permite o uso da mesma parametrização para extrapolações, quando isto se fizer necessário.

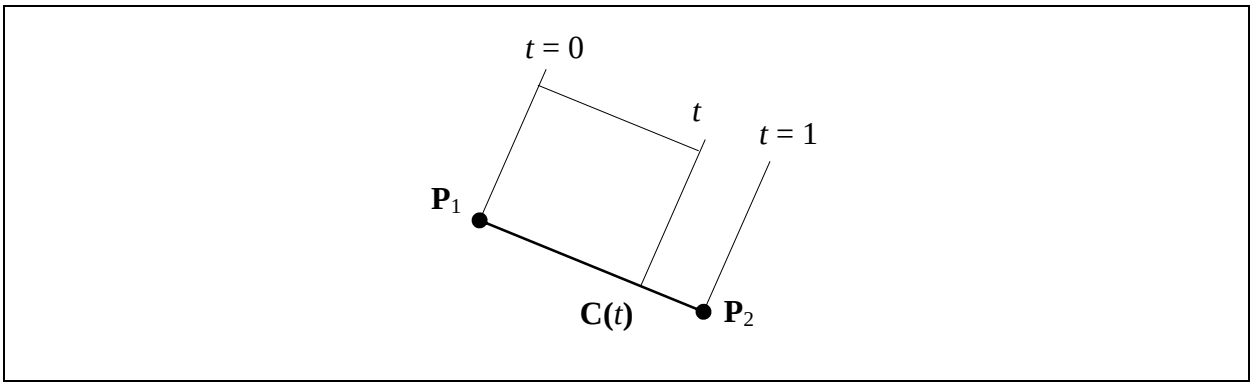


Figura 2.1 – Parametrização de um segmento de reta.

Esta representação paramétrica da Reta é utilizada repetidas vezes nos diversos algoritmos desenvolvidos para este trabalho, destacando-se a interseção entre retas, que é utilizada em diversas etapas do algoritmo de interseção de superfícies (ver Capítulo 3), em etapas do algoritmo de geração de malha (ver Capítulo 4) e em diversas outras situações.

2.1.2. Parametrização de Circunferência e de Arco de Circunferência

A equação paramétrica de uma Circunferência no plano *xy* apresenta uma forma simples, que vem diretamente das definições das funções trigonométricas seno e co-seno. No entanto, uma representação paramétrica tridimensional não é trivial. Mas o problema pode tornar-se simples com a utilização de um Sistema de Referência Auxiliar (SRA) acoplado ao plano da circunferência. Considerando-se um Arco de Circunferência no plano paralelo a *xy*, com centro na origem, pode-se definir a seguinte representação:

$$\mathbf{C}_0(t) = [R\cdot Cos\,\alpha(t) \quad R\cdot Sen\,\alpha(t) \quad z] \tag{2.3}$$

$$\alpha(t) = \alpha_i + (\alpha_f - \alpha_i)\cdot t \tag{2.4}$$

onde R é o raio do arco, α_i e α_f são, respectivamente, os ângulos inicial e final do arco e $t \in [0,1]$. Para representar um arco com centro em posição distinta da origem, basta somar à Eq. 2.3 o ponto do centro do arco, resultando na seguinte representação:

$$\mathbf{C}(t) = \mathbf{O}_A + \mathbf{C}_0(t) \tag{2.5}$$

onde $\mathbf{O}_A$ é o centro do Arco. A Eq. 2.5 é a representação paramétrica de um arco no plano *xy* do Sistema de Referência Global (SRG).

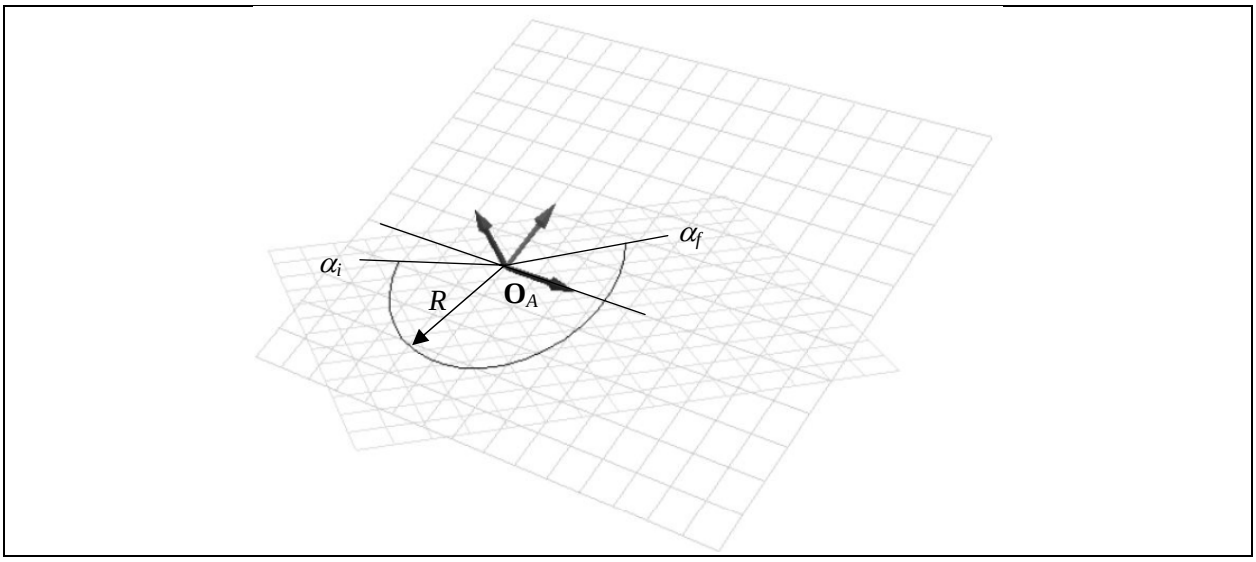


Figura 2.2 – Esquema de parametrização de um Arco de Circunferência no espaço tridimensional.

Para uma representação no espaço 3D, define-se um Sistema de Referência Auxiliar acoplado ao plano do arco e cuja Origem está no centro do Arco, o que permite o uso da Eq. 2.3 em conjunto com uma transformação de coordenadas do SRA para o SRG. Assim, pode-se considerar a seguinte representação paramétrica de um arco no espaço 3D:

$$\mathbf{C}(t) = \mathbf{G} \cdot \mathbf{C}_0(t) \tag{2.6}$$

onde **G** é a matriz de transformação que converte as coordenadas do Sistema de Referência local do arco (SRA) para o Sistema de Referência Global e $\mathbf{C}_0(t)$ é definido pela Eq. 2.3, que define um arco paramétrico com centro na Origem de um Sistema de Referência. Utiliza-se a Eq. 2.4 para definir o ângulo em função do parâmetro *t*. A representação de uma Circunferência completa é feita utilizando-se ângulos inicial e final tais que a diferença entre eles seja de 360º.

Desta forma, com a escolha apropriada de Sistemas de Referência[1], é possível representar arcos e circunferências em qualquer posição no espaço (Fig. 2.2), o que é muito importante para o modelamento de superfícies que utilizam curvas de bordo, como as Regradas, Coons e de Revolução.

2.1.3. Spline 3D Interpoladora

O uso de curvas para representar geometrias de objetos de Engenharia pode exigir a representação de curvas interpoladoras no espaço 3D. As curvas do tipo Spline, em geral, permitem representações 3D, mas, na maioria dos casos, utilizam pontos de controle que não estão sobre a curva. É possível construir uma Spline interpoladora tipo B-Spline, obtendo os pontos de controle correspondentes a partir da solução de um sistema de equações que fica bem determinado com a definição de condições de contorno [Rogers e Adams, 1990].

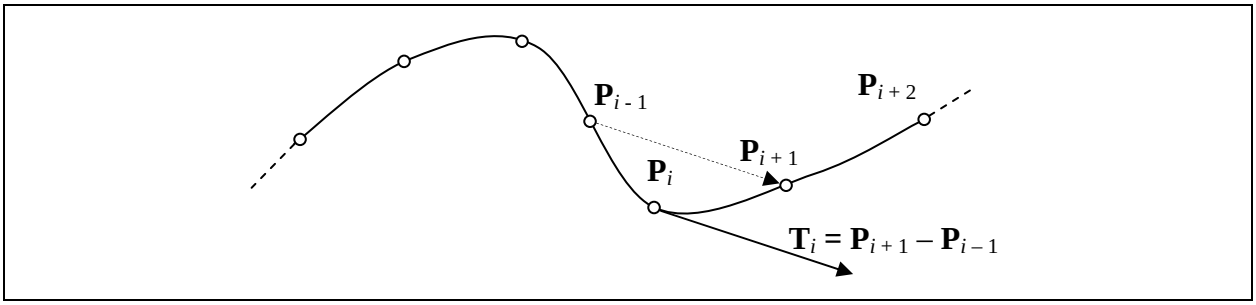


Figura 2.3 – Esquema de parametrização de uma Spline Catmull-Rom.

A Spline desenvolvida por Catmull e Rom, 1974, possui os pontos de controle sobre a curva, com exceção do primeiro e do último ponto que definem as tangentes nas extremidades. A direção da tangente à curva em um ponto de controle P_i é dada pelo vetor $\mathbf{P}_{i+1} - \mathbf{P}_{i-1}$. A equação paramétrica é cúbica e pode ser escrita da seguinte forma:

$$\mathbf{C}(t) = \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \cdot \frac{1}{2} \cdot \begin{bmatrix} 0 & 2 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 2 & -5 & 4 & -1 \\ -1 & 3 & -3 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{P}_{i-1} \\ \mathbf{P}_i \\ \mathbf{P}_{i+1} \\ \mathbf{P}_{i+2} \end{bmatrix} \tag{2.7}$$

onde $\mathbf{P}_{i-1}$, $\mathbf{P}_i$, $\mathbf{P}_{i+1}$ e $\mathbf{P}_{i+2}$ são os 4 pontos de controle que definem um trecho de curva entre os pontos $\mathbf{P}_i$ e $\mathbf{P}_{i+1}$, sendo os outros pontos responsáveis pelas direções das tangentes nas duas

[1] Foi implementada no T-CADE a Classe TSis juntamente com uma interface amigável, o que permite criar

extremidades do trecho (Fig. 2.3). Uma curva com mais de 4 pontos de controle pode ser gerada pelo acoplamento de trechos sucessivos utilizando a mesma equação. Assim, uma curva com *n* pontos de controle terá $n-3$ trechos. Curvas fechadas também são possíveis, utilizando-se o primeiro ponto também com último. Desta forma, considerando uma curva fechada com *n* pontos de controle, o último trecho da curva fica definido pelos pontos: $\mathbf{P}_{n-2}$, $\mathbf{P}_{n-1}$, $\mathbf{P}_1$ e $\mathbf{P}_2$.

A Spline Catmull-Rom foi escolhida para implementação pela sua capacidade de interpolação, e, principalmente, pela sua formulação explícita, o que facilita a implementação e garante velocidade durante uma modelagem interativa.

2.1.4. Polilinhas Paramétricas

Um recurso importante no modelamento geométrico é a união de várias curvas paramétricas em uma única linha paramétrica, permitindo a criação de geometrias complexas, aumentando as possibilidades de modelamento. Estas linhas compostas são denominas Polilinhas e são usuais em programas CAD.

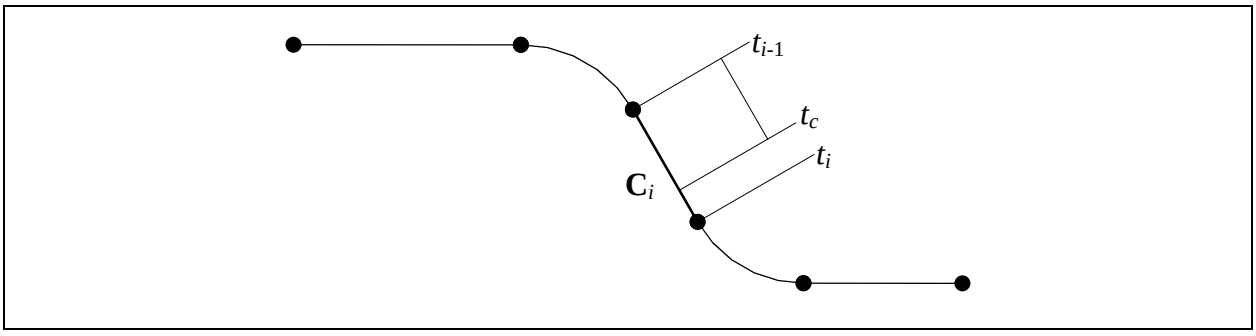


Figura 2.4 – Esquema de parametrização de uma polilinha.

A forma de parametrização escolhida utiliza como peso a relação entre os comprimentos das linhas individualmente e o comprimento total da Polilinha. Inicialmente, define-se o parâmetro *t* para o ponto final[2] de cada curva da seguinte forma:

$$t_i = \frac{C_i}{\sum_{i=1}^{n} C_i} \tag{2.8}$$

onde C_i é o comprimento da linha *i* e *n* é o número total de linhas. A Eq. 2.8 considera que cada Linha está conectada pelas extremidades e em ordem crescente de 1 a *n*. A partir disto, para um

---

Sistemas de Referência Auxiliares de forma fácil e flexível (ver Capítulo 5).

[2] Considerando que o sentido é definido pela ordem em que as Linhas são clicadas para formar a Polilinha.

determinado *t*, é possível calcular um *t* corrigido (t_c) para o segmento correspondente da seguinte forma:

$$t_c = \frac{t - t_{i-1}}{t_i - t_{i-1}} \tag{2.9}$$

onde t_{i-1} e t_i correspondem aos limites paramétricos do segmento ou linha que contém o *t* onde pretende-se avaliar a polilinha. A equação paramétrica de uma Polilinha fica reduzida à equação da curva correspondente a t_c:

$$\mathbf{C}(t) = \mathbf{C}_i(t_c) \tag{2.10}$$

onde t_c é definido pela Eq. 2.9 e $\mathbf{C}_i$ corresponde à equação paramétrica do segmento ou curva que contém *t* (Fig. 2.4). As Eqs. 2.9 e 2.10 permitem definir Polilinhas com segmentos paramétricos de qualquer tipo, desde que o intervalo paramétrico válido seja o mesmo: [0,1]. A disposição espacial das curvas pode variar, desde que estejam conectadas pelas extremidades.

2.2. REPRESENTAÇÃO PARAMÉTRICA DE SUPERFÍCIES

A representação paramétrica de superfícies é uma alternativa extremamente prática e conveniente para aplicação computacional. De forma idêntica à representação paramétrica de curvas, as superfícies são representadas por equações paramétricas independentes para cada dimensão. Nesse caso, o espaço paramétrico é bidimensional e a representação genérica de uma superfície fica da forma:

$$\mathbf{S}(u,v) = \sum_{i=0}^{n} B_i(u,v) \cdot \mathbf{P}_i \tag{2.11}$$

onde $B_i(u,v)$ são as funções de peso, $\mathbf{P}_i$ os pontos de controle utilizados para representar a superfície e *u* e *v* são parâmetros que, normalmente, variam entre 0 e 1. Essa forma de representação relaciona dois espaços: o espaço paramétrico (bidimensional) e o espaço real (tridimensional). No espaço paramétrico, a superfície é um quadrado com uma unidade de lado. No espaço real, a superfície assume forma descrita nas equações paramétricas, no entanto, seu domínio permanece quadrilátero. Uma maneira usual de fugir dessa limitação é recortar a superfície, criando sub-domínios de contornos arbitrários, incluindo linhas internas e furos. As superfícies recortadas (*trimmed surfaces*) apresentam como propriedades funções que definem as linhas de recorte (*trimming lines*). Essas funções devem ser definidas no espaço paramétrico e podem ser obtidas a partir da interseção entre superfícies ou inseridas pelo usuário a fim de

representar determinadas geometrias.

Para este trabalho, foram desenvolvidas e implementadas no T-CADE representações de superfícies paramétricas de domínios quadriláteros que são utilizadas na geração de modelos geométricos para análise ou simulação de problemas de Engenharia. As superfícies implementadas são dos seguintes tipos: Planas, Bilineares, Regradas, Revolução, Coons e Sweep. Estas superfícies sempre utilizam linhas paramétricas em sua definição e também para a definição de linhas de recorte.

2.2.1. Superfície Plana

As superfícies denominadas aqui de Planas constituem um tipo de superfície básico para a definição de superfícies Planas Recortadas. A sua definição é muito simples, porém muito útil para a definição de regiões planas com contornos arbitrários. A parametrização consiste em definir uma área retangular cujos limites envolvem um determinado conjunto de curvas paramétricas previamente selecionadas (Fig. 2.5). Utilizam-se as coordenadas mínimas e máximas (*x* e *y* locais) medidas sobre o plano das linhas pré-definidas.

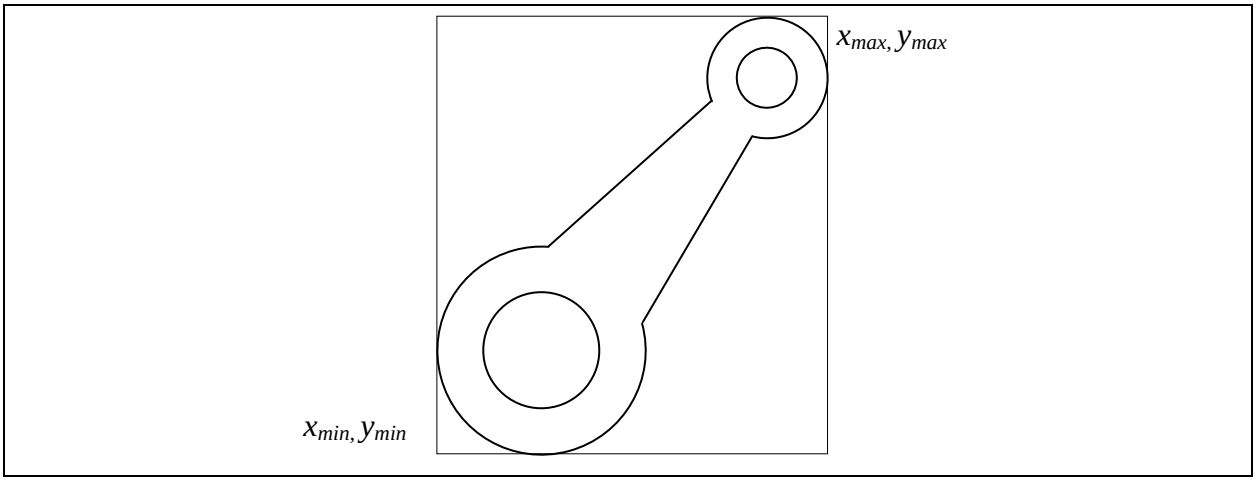


Figura 2.5 – Parametrização de uma superfície plana.

A representação paramétrica de uma área retangular em uma região plana pode ser assim definida:

$$\mathbf{S}(u,v) = \mathbf{G} \cdot \left[(1-u) \cdot x_{\min} + u \cdot x_{\max} \quad (1-v) \cdot y_{\min} + v \cdot y_{\max} \quad 0\right] \tag{2.12}$$

onde x_{min}, y_{min} *e* x_{max}, y_{max} são as coordenadas da diagonal do menor retângulo que envolve as linhas medidas nos eixos *x* e *y* locais do plano das linhas; **G** é a matriz que transforma as coordenadas do SRA que contém o plano das linhas para o SRG. Neste caso, a área retangular

correspondente à superfície plana é sempre alinhada aos eixos *x* e *y* do SRA da superfície. As curvas são utilizadas como *trimming lines* e a hierarquização das conexões entre elas deve ser estabelecida para determinar como a superfície será recortada pelas mesmas, definindo-se sub-domínios[3] com contornos, furos e linhas internas. Isto tem especial importância para a geração de malha, pois irá determinar em que regiões da superfície serão gerados os elementos da malha.

A implementação destas superfícies Planas no T-CADE possibilitou a geração de malhas em domínios planos de contornos arbitrários definidos pelo usuário, que não era o objetivo inicial deste trabalho, mas resultou como subproduto desta forma paramétrica e do algoritmo de geração de malha do Capítulo 4.

2.2.2. Superfície Bilinear

As superfícies denominadas aqui de superfícies Bilineares são obtidas pela interpolação bilinear de um conjunto de quatro pontos ($\mathbf{P}_i$, i = 1 .. 4). Por ser linear, a forma paramétrica é uma expansão da representação do segmento de reta nas duas direções paramétricas:

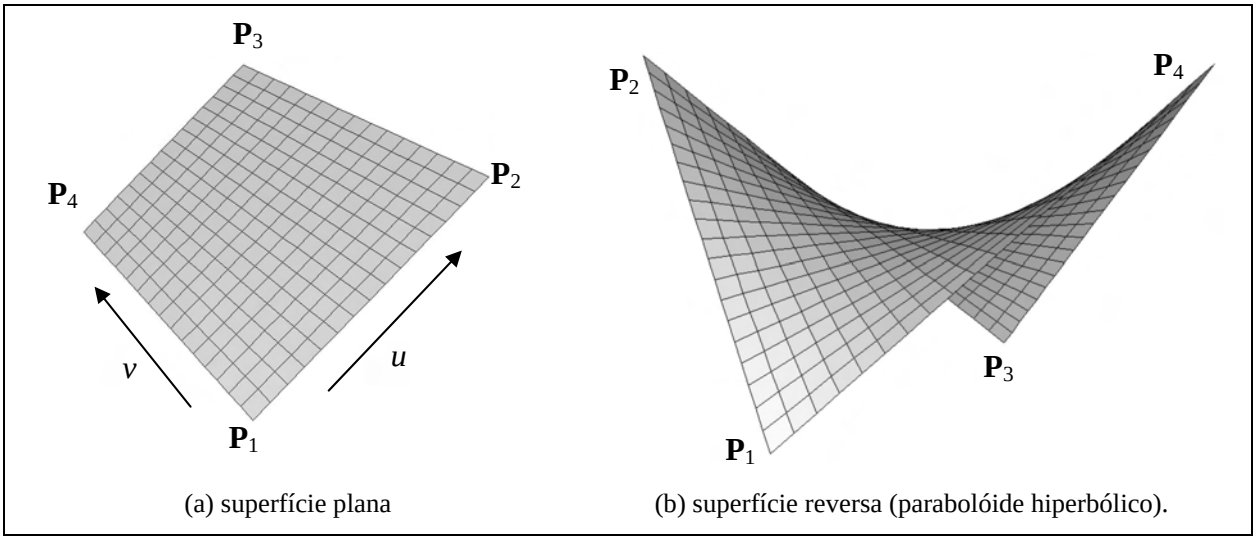
$$\mathbf{S}(u,v) = \begin{bmatrix}(1-u)(1-v) & u(1-v) & uv & (1-u)v\end{bmatrix}\begin{bmatrix}\mathbf{P}_1\\ \mathbf{P}_2\\ \mathbf{P}_3\\ \mathbf{P}_4\end{bmatrix} \tag{2.13}$$

onde $\mathbf{P}_1$ a $\mathbf{P}_4$ devem estar ordenados no sentido horário ou anti-horário, de tal forma que coincidam com as coordenadas paramétricas da seguinte forma: $\mathbf{P}_1$ – (0,0), $\mathbf{P}_2$ – (1,0), $\mathbf{P}_3$ – (1,1) e $\mathbf{P}_4$ – (0,1), o que pode ser facilmente comprovado substituindo estes valores na Eq. 2.13. Os pontos de controle podem estar em um mesmo plano, gerando uma superfície plana (Fig. 2.6a) ou podem não formar plano, gerando superfícies reversas e curvas, como o Parabolóide Hiperbólico[4] (Fig. 2.6b).

---

[3] Este assunto será abordado no Capítulo 4, que trata da geração de malha sobre superfícies recortadas, e no Capítulo 5, que aborda aspectos da implementação computacional (seção 5.1).

[4] O Parabolóide Hiperbólico é uma superfície bilinear com as linhas de bordo retas. Portanto, pode ser gerado como Superfície Bilinear, Superfície Regrada (Seção 2.2.3), e até como Superfície Coons (Seção 2.2.4).



(a) superfície plana (b) superfície reversa (parabolóide hiperbólico).

Figura 2.6 – Parametrização de uma superfície linear.

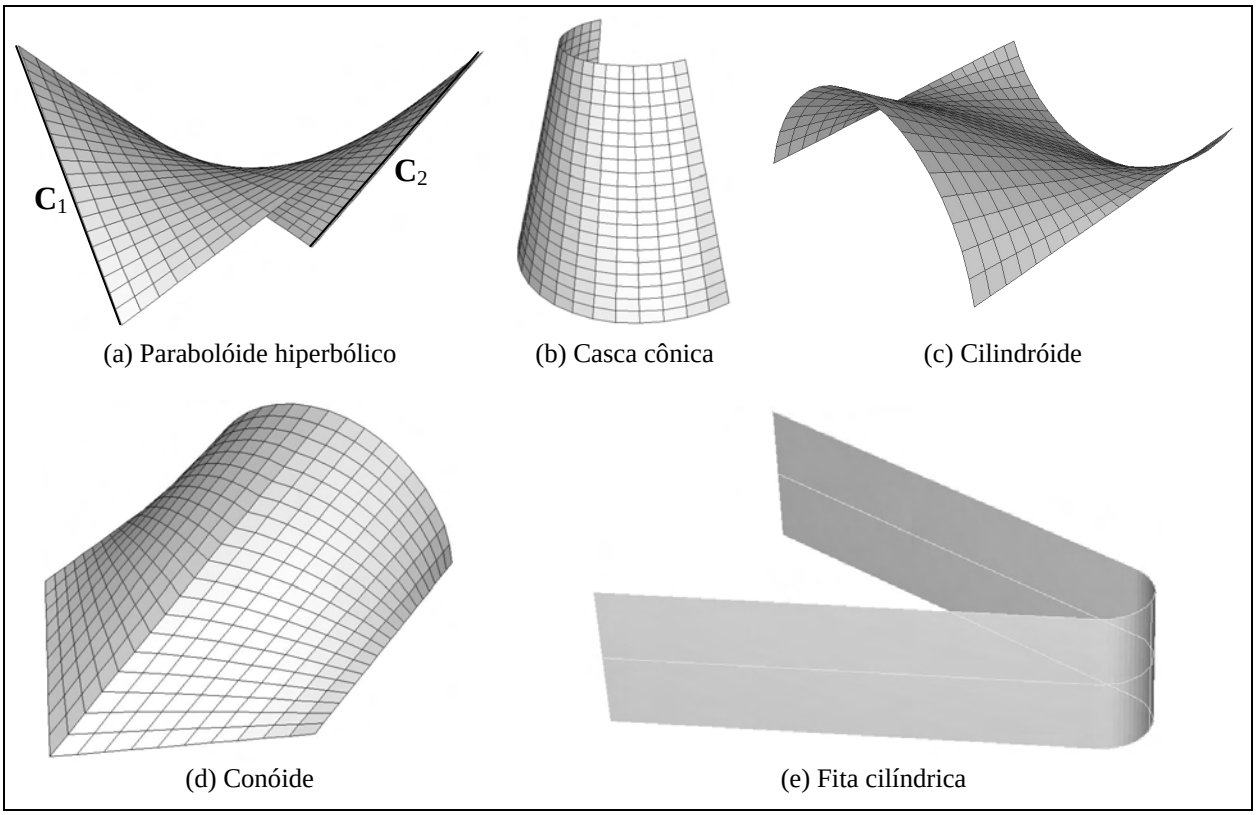
2.2.3. Superfície Regrada

As superfícies Regradas, na Geometria Descritiva, são geradas pelo deslocamento de uma reta (geratriz) apoiada em duas curvas (diretrizes). A parametrização é baseada diretamente neste conceito, realizando uma interpolação linear (geratrizes) entre duas curvas de bordo (diretrizes). Assim, a equação paramétrica das superfícies Regradas é derivada da representação do segmento de reta, onde as equações das curvas de bordo substituem os pontos de controle na equação paramétrica da Reta (Eq. 2.3), resultando na seguinte equação:

$$\mathbf{S}(u,v) = (1 - v)\cdot \mathbf{C}_1(u) \; + \; v\cdot \mathbf{C}_2(u) \tag{2.14}$$

onde $\mathbf{C}_1$ e $\mathbf{C}_2$ são as representações paramétricas das curvas de bordo. A forma da superfície gerada irá depender da forma das curvas de bordo e de suas posições. Muitas formas conhecidas podem ser geradas com este tipo, tais como: cascas cilíndricas e cônicas, cilindróides e conóides, entre muitos outros, o que torna este tipo de representação extremamente útil em modelagem 3D, como é comprovado por vários exemplos ao longo deste trabalho.

Além disso, a simplicidade da representação paramétrica das superfícies Regradas facilita a implementação computacional e permite agilidade na determinação de interseções e na geração de malha, pois os algoritmos desenvolvidos para estes processos utilizam, de forma intensiva, a avaliação das superfícies através de suas representações paramétricas. A Figura 2.7 mostra exemplos de superfícies Regradas geradas com diferentes curvas de bordo, evidenciando a versatilidade de representação geométrica inerente a este tipo superfície.



(a) Parabolóide hiperbólico (b) Casca cônica (c) Cilindróide

(d) Conóide (e) Fita cilíndrica

Figura 2.7 – Exemplos de superfícies Regradas com diferentes curvas de bordo.

2.2.4. Superfície Coons

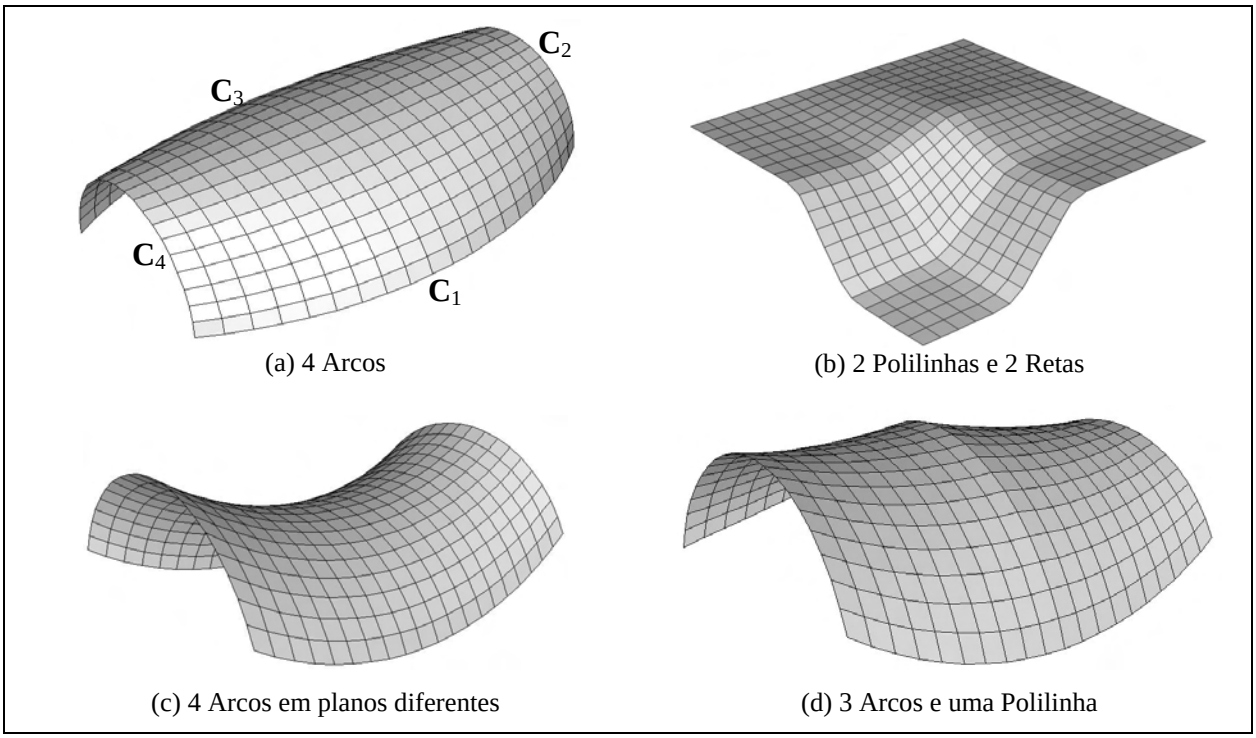
As superfícies Coons [Coons, 1964 e 1967] são superfícies paramétricas geradas a partir de quatro Curvas de bordo, utilizando interpolações duas a duas. Cada uma dessas quatro Curvas deve estar conectada às 2 Curvas adjacentes pelos pontos das extremidades. As Curvas de bordo podem ter parametrizações genéricas, contanto que os domínios sejam compatíveis.

As superfícies Coons podem ser lineares e quadráticas, nas quais é possível controlar a direção dos vetores tangentes nos vértices [Rogers e Adams, 1990]. Para este trabalho, foi implementada somente a Coons linear cuja representação paramétrica é a seguinte:

$$\begin{aligned}\mathbf{S}(u,v) = {} & (1-v)\,\mathbf{C}_1(u) + v\,\mathbf{C}_3(u) + (1-u)\,\mathbf{C}_4(v) + u\,\mathbf{C}_2(v) - \\ & -(1-u)(1-v)\,\mathbf{X}_1 - u\,(1-v)\,\mathbf{X}_3 - (1-u)\,v\,\mathbf{X}_2 - u\,v\,\mathbf{X}_4\end{aligned} \tag{2.15}$$

onde $\mathbf{C}_i$ são as equações paramétricas das curvas de bordo e $\mathbf{X}_i$ são os vértices do contorno. A Eq. 2.15 representa superfícies Coons lineares com domínios quadrados, mas existem superfícies Coons com domínios triangulares. As superfícies Coons são utilizadas em CAGD na modelagem 3D de superfícies devido à facilidade de criação de superfícies complexas a partir das curvas de contorno. A Figura 2.8 mostra alguns exemplos de superfícies Coons modeladas

no T-CADE. É possível observar que são diversas as geometrias possíveis, que dependem, fundamentalmente das geometrias das curvas de bordo.



(a) 4 Arcos (b) 2 Polilinhas e 2 Retas

(c) 4 Arcos em planos diferentes (d) 3 Arcos e uma Polilinha

Figura 2.8 – Exemplos de superfícies Coons com diferentes curvas de bordo.

2.2.5. Superfície de Revolução

Superfícies de revolução ocorrem com muita freqüência em diversos tipos de peças e dispositivos fabricados pelo homem, tais como elementos de máquinas e veículos, na industria mecânica e automobilística e aeroespacial, além de elementos construtivos, na industria da construção civil. Assim, é indispensável a implementação de superfícies de revolução em um programa que seja utilizado para o modelamento de objetos e problemas de Engenharia.

Nesse tipo de superfície, uma curva Perfil sofre uma revolução em torno de um eixo. Considerando-se o eixo vertical como o eixo *z* do Sistema de Referência, é possível definir a equação paramétrica de uma superfície de revolução da seguinte forma:

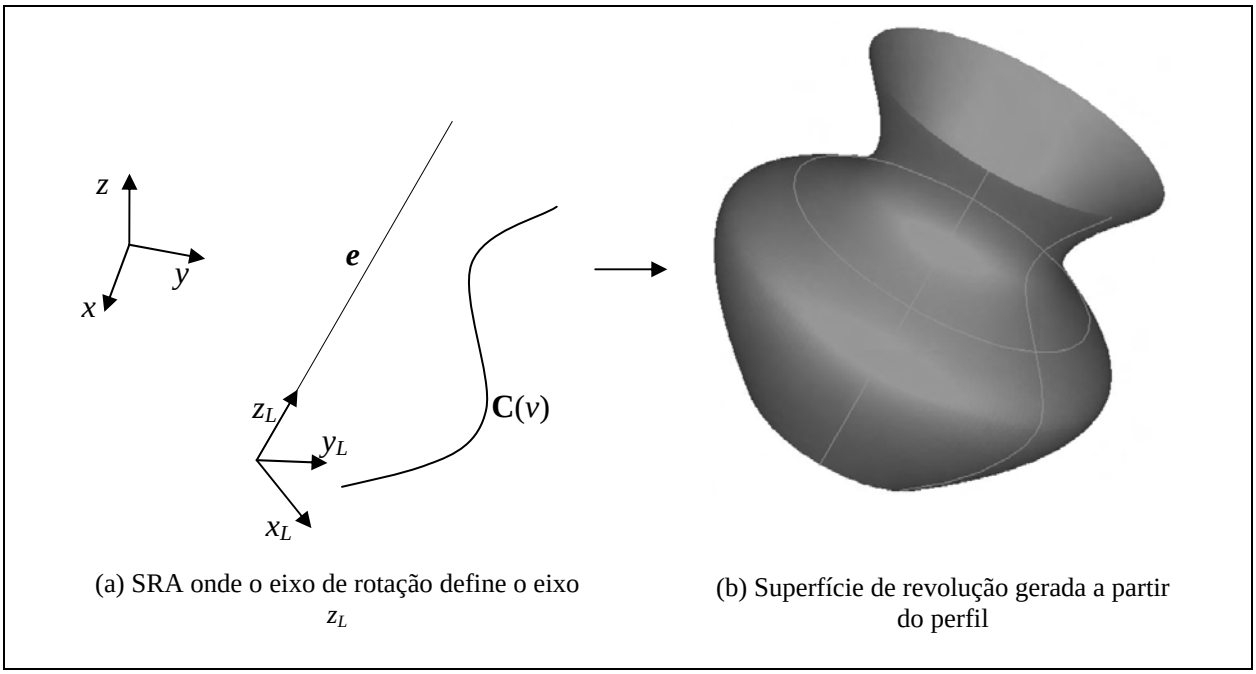
$$\mathbf{S}_z(u,v) = \mathbf{C}(v) \cdot \mathbf{T}(u) \tag{2.16}$$

onde $\mathbf{C}(v)$ é a equação paramétrica da linha de perfil que sofre a revolução e $\mathbf{T}(u)$ é a função paramétrica que realiza uma transformação de revolução em $\mathbf{C}(v)$. Neste caso, $\mathbf{T}(u)$ se resume à matriz de rotação no plano:

$$\mathbf{T}(u) = \begin{bmatrix} Cos\phi(u) & Sen\phi(u) & 0 \\ -Sen\phi(u) & Cos\phi(u) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.17}$$

$$\phi(u) = \phi_T \cdot u \tag{2.18}$$

onde $\phi(u)$ é a função que define o ângulo de rotação em função do parâmetro u e ϕ_T é o ângulo de revolução. O processo torna-se complexo quando o eixo de revolução adquire uma posição genérica. A abordagem tradicional para este caso [Rogers e Adams, 1990] utiliza três rotações e uma translação para criar uma matriz de transformação genérica.



(a) SRA onde o eixo de rotação define o eixo z_L

(b) Superfície de revolução gerada a partir do perfil

Figura 2.9 – Procedimento para criar uma superfície de revolução.

A alternativa proposta aqui é a utilização de um Sistema de Referência Auxiliar acoplado ao eixo de rotação, o que pode tornar o processo quase tão simples quanto a rotação em torno do eixo *z* (Fig. 2.9a). A transformação de rotação é realizada no SRA e o resultado é transferido para o SRG. A equação paramétrica da superfície de revolução fica assim definida:

$$\mathbf{S}(u,v) = \mathbf{G} \cdot \mathbf{T}(u) \cdot \mathbf{L} \cdot \mathbf{C}(v) \tag{2.19}$$

onde **C**(v) é a curva paramétrica de Perfil; **L** é a matriz de transformação do SRG para o SRA acoplado ao eixo de revolução, de tal forma que o eixo *z* local coincide com o eixo de revolução; **T**(*u*) é matriz de rotação (Eqs. 2.17 e 2.18) função do parâmetro *u*; **G** é a matriz de

transformação do SRA da superfície para SRG. A Eq. 2.19 toma um determinado ponto sobre a curva perfil **C**(*v*), transforma para o SRA com **L**, em seguida, aplica-se a rotação **T**(*u*) e, finalmente, transforma o resultado para o Sistema Global com **G**. É possível utilizar revolução completas (360º) ou parciais. As superfícies de Revolução completas apresentam uma linha de *costura* no fechamento da revolução que correspondem a valores de $u = 0$ e $u = 1$. Isto deve ser levando em conta no algoritmo de geração de malha para que haja continuidade da malha em todo o domínio da superfície.

2.2.6. Superfície Sweep

A superfícies denominadas aqui de Sweep são resultado de uma extrusão de uma curva de Forma através de uma curva Caminho, que corresponde a um Sweep genérico na literatura [Coelho, 1998]. Superfícies de Revolução também são consideradas superfícies Sweeps. As superfícies Sweep implementadas neste trabalho utilizam qualquer tipo de curva como forma e o Caminho pode ser de qualquer tipo, desde que seja uma curva plana. Apesar desta limitação, este tipo de superfície permite grande flexibilidade na criação de formas tridimensionais, as quais dificilmente poderiam ser geradas com outro tipo de superfícies paramétrica de forma mais econômica.

Um modo de representação usual de Sweeps é utilizar um SRA ou Frame [Watt, 1993] que se desloca ao longo da curva Caminho, de tal forma que o eixo *x* local seja coincidente com a direção da tangente à curva. O processo se completa transferindo as coordenadas da curva de Forma para estes Sistemas Locais e, em seguida, transpondo para o sistema Global. A representação paramétrica pode ser expressa pela seguinte equação:

$$\mathbf{S}(u,v) = \mathbf{G}(u) \cdot \mathbf{L}_0 \cdot \mathbf{C}(v) \tag{2.20}$$

onde **C**(*v*) é a representação paramétrica da curva de forma, $\mathbf{L}_0$ é matriz de transformação para coordenadas locais do Sistema de Referência Auxiliar localizado no primeiro do ponto da curva Caminho, de tal forma que seu eixo x_L coincide com a tangente da curva caminho neste ponto, **G**(*u*) é a matriz de transformação para coordenadas globais de um sistema de referência auxiliar que se desloca ao longo da curva caminho, sendo que seu eixo x_L é coincidente com a direção da tangente da curva caminho em *u*.

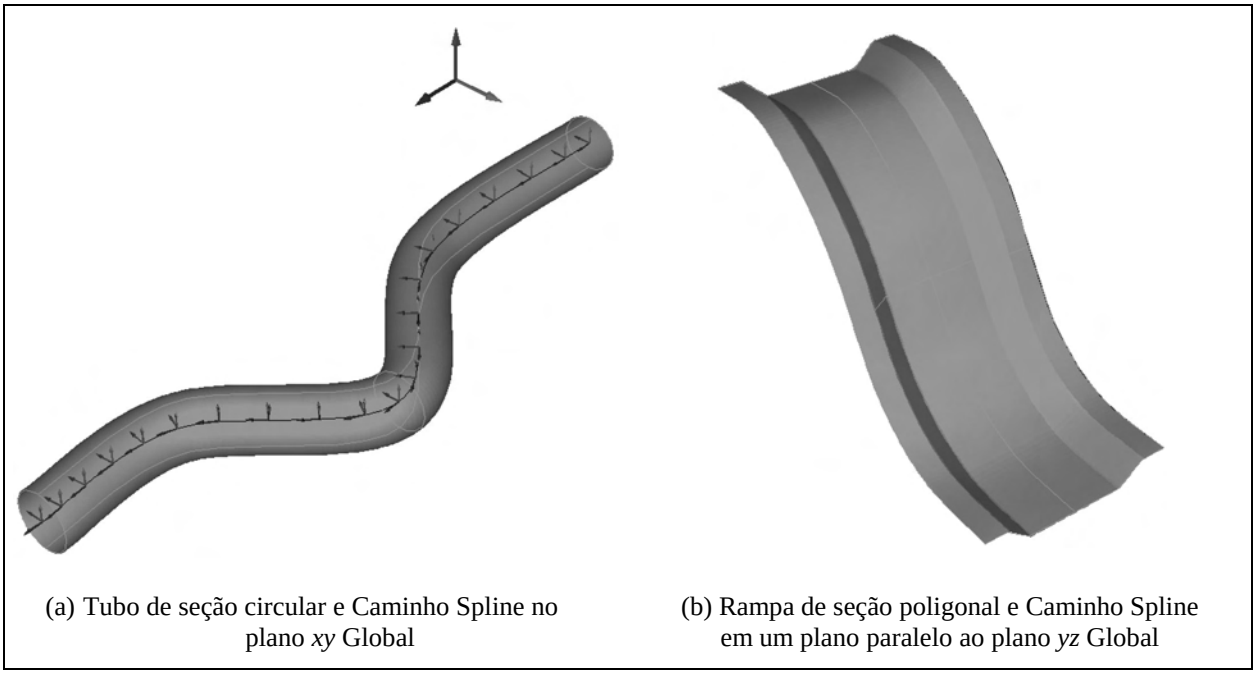
O Frame é um Sistema de Referência Auxiliar que pode ser montado de muitas maneiras. Uma forma usual é o Frenet Frame [Watt, 1993], cuja origem está em **C**(*t*) e possui como base ortonormal os vetores **T**, **N** e **B**. Onde **T** é o vetor unitário tangente em **C**(*t*). **N** é

calculado em função da derivada segunda da curva. E **B** é calculado pelo produto vetorial de **T** e **N**. O problema, é que o Frenet Frame tende mudar bruscamente de orientação em pontos de inflexão da curva Caminho [Coelho, 1998]. Neste trabalho, optou-se por restringir o problema de tal forma que a curva Caminho seja obrigatoriamente plana. Assim, utiliza-se um vetor **B** que coincide com a direção *z* local do Sistema de Referência cujo plano *xy* é coincidente com o plano da curva Caminho (Fig. 2.10a). Assim, o vetor **B** pode ser assim expresso da seguinte forma:

$$\mathbf{B} = \mathbf{G}_C \cdot \mathbf{k}_C \tag{2.21}$$

onde $\mathbf{k}_C$ é em vetor cujas componentes são: (0, 0, 1) no Sistema de Referência da curva Caminho e $\mathbf{G}_C$ é a matriz de transforma do Sistema de Referência da curva Caminho para o SRG. Com a determinação de **B**, **N** também fica determinado pelo produto vetorial:

$$\mathbf{N} = \mathbf{T} \times \mathbf{B} \tag{2.22}$$



(a) Tubo de seção circular e Caminho Spline no plano *xy* Global

(b) Rampa de seção poligonal e Caminho Spline em um plano paralelo ao plano *yz* Global

Figura 2.10 – Exemplos de superfície Sweep.

Esta uma solução robusta e que, mesmo com a limitação de ter Caminhos planos, permite a geração de superfícies de geometrias variadas e complexas, como as que ocorrem na modelagem geométrica voltada para a simulação e análise de problemas de Engenharia. A Figura 2.10 mostra dois Exemplos de Sweeps gerados no T-CADE com esta formulação. A Figura 2.10a apresenta um tubo de seção circular cujo Caminho é uma Spline contida no plano *xy* Global. Neste Exemplo, são mostrados os Frames ao longo da curva Caminho, comprovando que não há mudanças bruscas na sua orientação, mesmo nos pontos de inflexão da curva. A Figura

2.10b mostra um Sweep de uma poligonal em Caminho tipo Spline contido em um plano paralelo ao plano *yz* Global.

2.3. PROPRIEDADES DE CURVAS E SUPERFÍCIES

Várias situações na modelagem geométrica requerem a determinação de direções tangentes e normais a curvas e superfícies. A determinação destas propriedades exige o uso de derivadas parciais em alguma etapa do processo. Uma vez que são utilizadas várias formas de parametrizações, o caminho natural para representações genéricas é o uso de derivadas numéricas, uma vez que a determinação de derivadas de forma analítica deveria considerar cada forma paramétrica de maneira individual.

2.3.1. Derivadas Numéricas de Curvas e Superfícies Paramétricas

A derivada de uma função *f*(*x*) para $x = a$ pode ser aproximada pela seguinte expressão:

$$f'(a) \cong \frac{f(a+h) - f(a)}{h} \tag{2.23}$$

desde que *h*, que deve ser próximo de 0, tenha um valor adequado à precisão requerida pela aplicação. Esta expressão também pode ser utilizada para definição de derivadas parciais. No caso de linhas paramétricas, utiliza-se a derivada parcial em relação ao parâmetro *t* para determinar a direção da tangente a uma curva:

$$\mathbf{D}_t(t) = \frac{\partial}{\partial t}\mathbf{C}(t) = \frac{\mathbf{C}(t+h) - \mathbf{C}(t)}{h} \tag{2.24}$$

No caso de superfícies paramétricas, utilizam-se as derivadas parciais em relação aos parâmetros *u* e *v*, para determinar as direções dos vetores tangentes de um ponto nas duas direções paramétricas:

$$\mathbf{D}_u(u,v) = \frac{\partial}{\partial u}\mathbf{S}(u,v) = \frac{\mathbf{S}(u+h,v) - \mathbf{S}(u,v)}{h} \tag{2.25}$$

$$\mathbf{D}_v(u,v) = \frac{\partial}{\partial v}\mathbf{S}(u,v) = \frac{\mathbf{S}(u,v+h) - \mathbf{S}(u,v)}{h} \tag{2.26}$$

Neste trabalho, utiliza-se $h = 10^{-6}$ (avaliado no espaço paramétrico), que é um valor mais do que suficiente [Andrade, 1998] para se atingir a precisão requerida para maioria das

aplicações de CAGD. As Equações 2.24, 2.25 e 2.26 fornecem vetores não unitários e, muitas vezes, pode ser necessário o uso de vetores unitários[5]. Neste caso, ainda seria necessário normalizar os resultados destas Equações. A determinação de vetores tangentes a curvas e superfícies é de fundamental importância para o algoritmo de interseção e para o algoritmo de geração de malha, como será mostrado.

2.3.2. Determinação de Vetor Normal a uma Superfície Paramétrica

Os vetores normais à superfícies são utilizados intensivamente em computação gráfica. A determinação de curvaturas locais, a determinação de ângulos entre faces de malhas, a determinação de ângulo de incidência de linhas sobre uma superfície, entre outros, são exemplos onde se utilizam vetores normais a uma superfície.

A determinação de um vetor normal a uma superfície paramétrica pode ser feita utilizando-se o produto vetorial entre os vetores tangentes em um ponto especificado. O vetor normal (**N**) à uma superfície pode ser assim descrito:

$$\mathbf{N}(u,v) = \mathbf{T}_u(u,v) \times \mathbf{T}_v(u,v) \tag{2.27}$$

onde $\mathbf{T}_u(u,v)$ e $\mathbf{T}_v(u,v)$ são vetores tangentes à superfície, obtidos pelas derivadas parciais em relação às coordenadas paramétricas (Eqs. 2.25 e 2.26) e normalizados:

$$\mathbf{T}_u(u,v) = \frac{\mathbf{D}_u(u,v)}{\left\|\mathbf{D}_u(u,v)\right\|} \tag{2.28}$$

$$\mathbf{T}_v(u,v) = \frac{\mathbf{D}_v(u,v)}{\left\|\mathbf{D}_v(u,v)\right\|} \tag{2.29}$$

2.4. CONSIDERAÇÕES FINAIS

Em trabalhos futuros, pretende-se implementar curvas e superfícies tipo NURBS, e outras, a fim de tornar mais genérica e robusta a representação de objetos geométricos. Além disso, pretende-se aproveitar o T-CADE como plataforma de desenvolvimento para novas representações paramétricas. No entanto, os objetos geométricos já implementados no T-CADE

[5] Como no cálculo de ângulos entre vetores, medido pelo produto escalar que coincide com o co-seno do ângulo quando os vetores são unitários.

satisfazem plenamente os objetivos deste trabalho, permitindo o modelamento de geometrias complexas como as que ocorrem em problemas de Engenharia, tais como elementos de máquinas, ou mesmo objetos mais complexos, como será mostrado mais adiante. Neste caso, a determinação de interseções entre superfícies paramétricas é crucial, pois permite recortar as superfícies a fim de representar as mais variadas geometrias. O Capítulo 6 apresenta várias aplicações de modelamento de superfícies compostas que utilizam as curvas e superfícies apresentadas aqui, em conjunto com o algoritmo de interseção de superfícies apresentado no Capítulo 3 e o algoritmo de geração de malha apresentado no Capítulo 4.

3. INTERSEÇÕES ENTRE SUPERFÍCIES PARAMÉTRICAS

A determinação de interseções entre superfícies consiste em um dos temas mais importantes no modelamento geométrico computacional de superfícies e sólidos, com aplicações em computação gráfica e no projeto assistido por computador (CAD e CAE). Várias são as aplicações onde é necessário determinar as linhas de interseção entre duas superfícies, podendo-se destacar: modelamento geométrico para a geração de malhas de Elementos Finitos em cascas e sólidos, representação B-Rep a partir de CSG, determinação de silhuetas de superfícies, operações Booleanas, construção de superfícies de concordância entre duas superfícies, determinação de caminhos de ferramentas (CAM), detecção de interferências e colisões.

As técnicas de determinação de interseção têm especial importância em Engenharia, principalmente nas industrias automobilística, aeroespacial, naval e metal-mecânica, onde se utiliza intensivamente a simulação computacional nas etapas de projeto, na manufatura e análise de desempenho de componentes.

3.1. O ALGORITMO DE INTERSEÇÃO PROPOSTO

O método de interseção proposto e implementado neste trabalho utiliza subdivisão adaptativa em função da curvatura local das superfícies. As superfícies são subdivididas em etapas sucessivas até que não existam trechos com curvatura superior a um determinado limite. Desta forma, é possível reduzir o problema de interseção, localmente, ao caso de interseção entre dois planos. A grande vantagem deste método é a total independência de tipo e forma das superfícies, assim como da forma e complexidade das linhas de interseção. Estes fatores são importantes para garantir a robustez e generalidade do método, características fundamentais em CAGD. A velocidade de processamento, que é apontada como uma característica negativa deste tipo de método [Andrade, 1998], pode ser otimizada com a utilização critérios adequados na subdivisão, na computação de interferências e na determinação dos pontos de interseção.

No entanto, o uso exclusivo de subdivisão adaptativa não garante um nível de precisão adequado e, tão pouco, uniforme ao longo das linhas de interseção. Este problema torna-se mais crítico quando as superfícies envolvidas possuem curvatura variável, o que é comum em diversos tipos de superfícies. Tais limitações são resolvidas utilizando um algoritmo que realiza um refinamento dos resultados, garantindo uma precisão elevada e uniforme ao longo de todas as linhas de interseção obtidas. Este algoritmo obtém os pontos das linhas de interseção mapeados em coordenadas paramétricas sobre as duas superfícies envolvidas com uma precisão de 10^{-12}

das dimensões das mesmas. Esta é uma contribuição importante deste trabalho, pois definindo as linhas de interseção no espaço paramétrico, é possível definir sub-domínios sobre as superfícies, o que irá facilitar o processo de geração de malha em recortes das superfícies.

O algoritmo completo desenvolvido neste trabalho consiste de quatro etapas consecutivas e complementares:

- Subdivisão adaptativa – As duas superfícies são subdivididas em trechos quase planos, analisando-se as interferências dos volumes envolventes dos mesmos;
- Interseção entre trechos – São calculados os segmentos de interseção no espaço 3D real que correspondem a uma aproximação inicial das linhas de interseção;
- Refinamento dos resultados – Refinam-se os resultados em um processo iterativo que faz o mapeamento no espaço paramétrico das superfícies;
- Reordenação dos segmentos e parametrização – Conectam-se os segmentos consecutivos para criar linhas contínuas de interseção, as quais são parametrizadas.

3.1.1. Subdivisão Adaptativa

A subdivisão adaptativa implementada utiliza uma estrutura de *quadtrees* não uniformes para subdividir as superfícies em função das curvaturas locais. O objetivo é chegar a trechos planos o suficiente para determinar os pontos de interseção através da interseção entre dois planos (trecho a trecho). O processo de subdivisão é iterativo e orientado pelas interferências dos volumes envolventes (*bounding volumes*) dos trechos de cada superfície e pela curvatura local das superfícies. O processo de subdivisão é realizado no espaço paramétrico, mas é governado por parâmetros do espaço real.

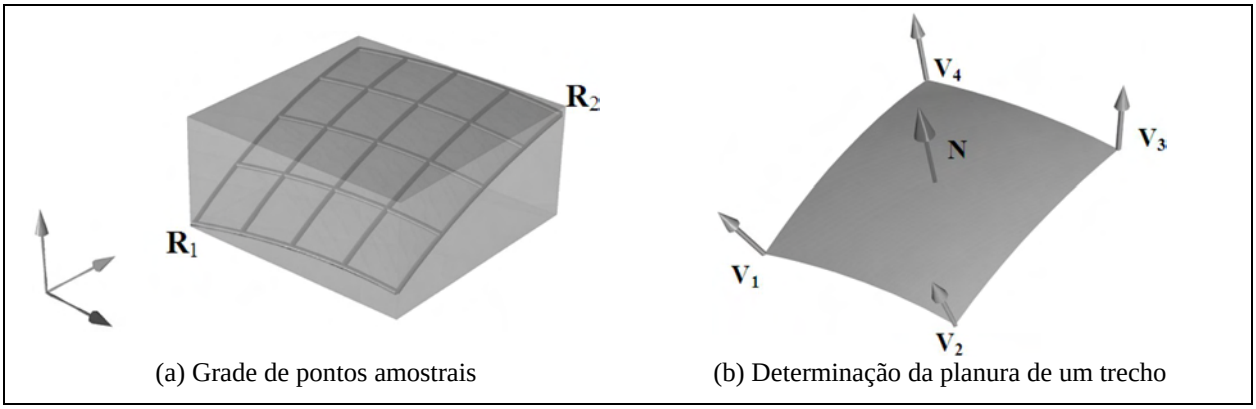
O algoritmo de subdivisão utiliza uma estrutura de dados que armazena os trechos de superfície com seus volumes envolventes. A cada iteração, esta estrutura de dados é atualizada em função da verificação de interferências entre os volumes envolventes das duas superfícies e das subdivisões decorrentes da curvatura local da superfície. A montagem desta estrutura de dados, bem como os procedimentos de verificação de interferência e do processo completo de subdivisão serão descritos a seguir.

3.1.1.1. Construção dos volumes envolventes (*bounding volumes*)

A subdivisão por *quadtree* gera células, que são regiões quadradas no espaço paramétrico às quais correspondem trechos de superfície no espaço 3D. Considera-se o volume

envolvente (VE) de um trecho um paralelepípedo que o encerra justa e totalmente. Optou-se pelo uso de volumes envolventes alinhados com os eixos coordenados devido à velocidade na construção e no cálculo de interferências inerentes a este tipo de VE. Os chamados volumes justos ou orientados podem reduzir o número de iterações do processo de subdivisão [Houghton *et al.*, 1985; Barth e Huber, 1998], porém esta redução deve compensar a redução de desempenho devido ao alto custo computacional da montagem e do cálculo de interferências.

Cada célula da *quadtree* é descrita por dois pontos, $\mathbf{P}_1$ e $\mathbf{P}_2$, no espaço paramétrico e dois pontos no espaço 3D, $\mathbf{R}_1$ e $\mathbf{R}_2$ (Fig. 3.1a). Os pontos no espaço paramétrico correspondem aos limites da célula e definem a diagonal que une o ponto inferior esquerdo ao ponto superior direito. Os pontos no espaço real correspondem às extremidades da diagonal do paralelepípedo que envolve o trecho de superfície correspondente à célula da *quadtree*. Este paralelepípedo é o volume envolvente do trecho da superfície. Os pontos $\mathbf{R}_1$ e $\mathbf{R}_2$ correspondem, então, às coordenadas máximas e mínimas do volume envolvente.



(a) Grade de pontos amostrais (b) Determinação da planura de um trecho

Figura 3.1 – Determinação do volume envolvente (a) em função da planura (b).

Para garantir que o volume envolvente contenha completamente o trecho de superfície correspondente, tomam-se pontos amostrados do interior do trecho. O número de pontos utilizados depende da planura do trecho, que pode ser expressa pelo ângulo entre as normais dos vértices do trecho com a normal do centro do trecho (Fig. 3.1b). Considerando um trecho de superfície *i*, a sua planura ($\hat{P}_i$) pode ser definida pela seguinte expressão:

$$\hat{P}_i = Min\left(\frac{\mathbf{N}_i}{\|\mathbf{N}_i\|} \cdot \frac{\mathbf{N}_{ij}}{\|\mathbf{N}_{ij}\|} \right)_{j=1..4} \tag{3.1}$$

onde $\mathbf{N}_i$ é o vetor normal do centro do trecho e os $\mathbf{N}_{ij}$ correspondem às normais à superfície nos

quatro vértices do trecho. Considerando que estes vetores normais são unitários, o menor valor corresponde ao co-seno do maior ângulo entre as normais. Trechos completamente planos apresentam $\hat{P}_i = 1$. Utiliza-se, então, um valor $\hat{P}_{Min}$, que corresponde a uma tolerância angular (A_{Max}). Para valores superiores a $\hat{P}_{Min}$, considera-se o trecho suficientemente plano.

Para o cálculo do volume envolvente, o número de pontos amostrados decresce com o aumento de $\hat{P}$. O trecho é subdividido em *n*x*n* partes (Fig. 3.1a) de tal forma que a planura medida entre os pontos amostrados seja superior a $\hat{P}_{Min}$. Nos exemplos apresentados na seção 3.2, utilizou-se $\hat{P}_{Min} = 0{,}999048$ que corresponde a um $A_{Max} = 2{,}5^{o}$. Esta é a metade da tolerância angular utilizada no processo de subdivisão, que é 5^{o}. A malha de amostragem mínima é 2 x 2. No começo do processo de subdivisão, *n* é alto pois os trechos ainda são pouco planos, mas, à medida que o processo avança, *n* cai rapidamente até que, no final do processo, aproxima-se de 2. Este processo de computação para os volumes envolventes é simples, rápido e eficiente e consiste em uma das contribuições do algoritmo proposto.

3.1.1.2. Verificação de Interferência entre dois Volumes Envolventes

A determinação de interferências entre dois volumes envolventes de dois trechos de superfícies distintas é relativamente simples quando estes são alinhados com os eixos coordenados. Projetam-se os volumes em cada um dos planos *xy*, *yz* e *xz*, resultando em retângulos (Fig. 3.2). Para que haja interferência entre dois volumes, devem haver interferências entre as suas projeções em todos os planos. Assim, analisando-se as três projeções dos dois volumes envolvidos, é possível determinar se há ou não interferência entre os mesmos.

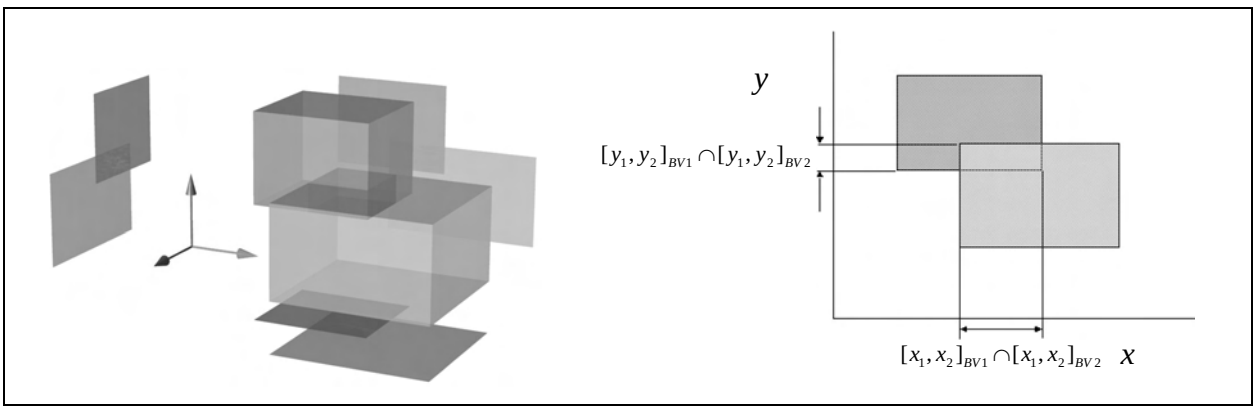


Figura 3.2 – Determinação de interferência entre dois volumes envolventes. São analisadas as coordenadas nos 3 eixos para verificar se há interferência entre as projeções dos mesmos.

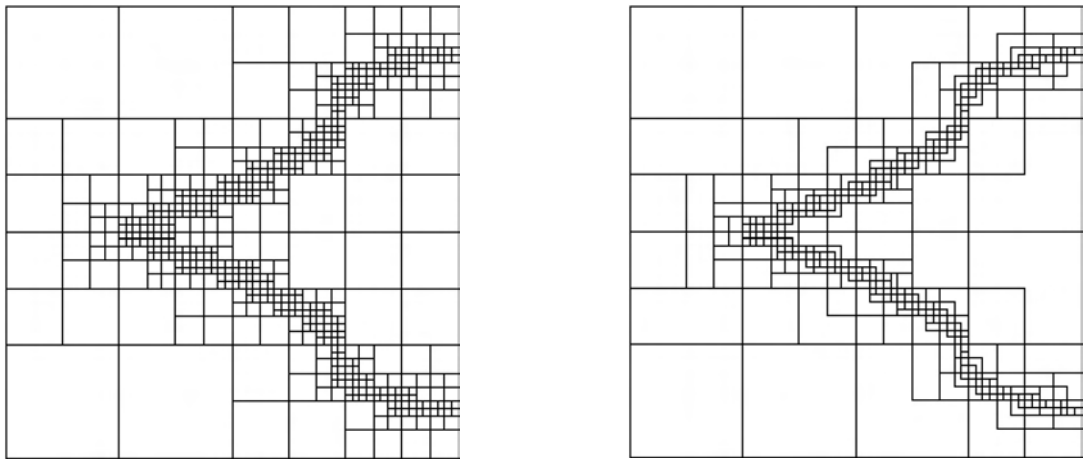
A verificação de interferências em cada plano é feita comparando-se as coordenadas limites dos retângulos nos respectivos planos, o que pode ser resumido da seguinte forma:

- $[x_1, x_2]_{VE1} \cap [x_1, x_2]_{VE2}$, verifica a interseção no eixo *x*;
- $[y_1, y_2]_{VE1} \cap [y_1, y_2]_{VE2}$, verifica a interseção no eixo *y*;
- $[z_1, z_2]_{VE1} \cap [z_1, z_2]_{VE2}$, verifica a interseção no eixo z.

Os pares de coordenadas correspondem às componentes dos pontos extremos dos VE ($\mathbf{R}_1$ e $\mathbf{R}_2$). Se estas três condições forem verdadeiras, pode-se afirmar que há interseção entre os dois volumes envolventes. Neste caso, são acrescentadas às estruturas de dados os vínculos entre as células correspondentes das *quadtrees* das duas superfícies.

3.1.2. A Subdivisão de Um Trecho de Superfície

O algoritmo de subdivisão tem como instância fundamental a subdivisão de uma célula, que pode ser devida à planura menor que a tolerância ou por questões topológicas, que serão descritas oportunamente. Cada célula subdividida dá origem a quatro outras células filhas, com dimensões paramétricas iguais à metade das dimensões da célula mãe. As células filhas herdam os vínculos da célula mãe, que é removida da estrutura de dados da *quadtree*.



(a) Sem verificação de falsos vínculos (b) Com verificação de falsos vínculos

Figura 3.3 – *Quadtree* de subdivisão.

Antes de as células filhas serem adicionadas à estrutura de dados, uma verificação de interferência é feita entre os VE destas e os VE das células da outra superfície vinculadas à sua célula mãe. São adicionadas à base de dados somente aquelas células filhas com verificação positiva. Este procedimento evita verificações desnecessárias entre células com vínculos falsos,

aumentando a taxa de convergência e o desempenho do algoritmo, além de reduzir consideravelmente o número de verificações de interferências nas gerações seguintes. A Figura 3.3 mostra dois exemplos de *quadtree*: sem verificação de vínculos falsos (Fig. 3.3a) e com a verificação de vínculos falsos (Fig. 3.3b). Pode-se observar a sensível redução de células que a verificação proporciona. Isto repercute diretamente no desempenho do algoritmo, pois uma célula de uma superfície pode estar vinculada a várias outras da outra superfície e, sem a remoção dos vínculos falsos, a cada geração o número de descendentes multiplica-se por 4 e o número de vínculos aumenta com o quadrado disto (16). Assim, qualquer célula eliminada previamente, elimina a necessidade de, pelo menos, 16 verificações. Considerando-se o conjunto das células, isto pode ser muito significativo, mas a redução varia conforme a geometria e a topologia das superfícies.

3.1.3. O Algoritmo de Subdivisão

O algoritmo de subdivisão atua simultaneamente nas duas superfícies que se interceptam, utilizando as interferências dos volumes envolventes das mesmas e a curvatura (planura) local para orientar o processo. A seguir são descritas as principais etapas do algoritmo.

i. As duas superfícies são subdivididas, inicialmente, em uma malha 2x2, resultando em 4 trechos, em cada uma, de dimensões iguais em coordenadas paramétricas. Para cada trecho, é a acrescentado à base de dados o VE correspondente. Neste passo inicial, todos os VE de uma superfície são vinculados a todos VE da outra superfície.

ii. Computam-se as interferências entre os volumes envolventes das células das duas superfícies segundo os seus vínculos. Aquelas células cujos volumes não possuem interferência são descartadas da base de dados. As células cujos volumes possuem interferências são candidatas potenciais a conterem interseções. Estas são mantidas na base de dados e são vinculadas àquelas células da outra superfície cujos VEs interferem com os seus.

iii. Verificam-se as planuras ($\hat{P}_i$) de todas as células restantes das duas superfícies. Caso uma célula possua $\hat{P}_i > \hat{P}_{Min}$ (tolerância), a mesma não é subdividida. Caso contrário, a célula é subdividida e as células filhas herdam seus vínculos com as células da outra superfície.

iv. Uma verificação adicional de interferência é feita entre os VE das células filhas e os VE das células vinculadas a estas por herança. Somente as células filhas com interferência positiva são acrescentadas à base de dados em substituição à célula mãe.

v. Verificam-se as diferenças de níveis de subdivisão entre as células vizinhas. Só é permitido um nível de diferença. Caso existam células vizinhas com mais de um nível de diferença, realizam-se subdivisões adicionais para que a diferença final seja de apenas um nível.

vi. Verificam-se as planuras de todas as células restantes das duas superfícies:

- Se existe alguma célula com $\hat{P}_i < \hat{P}_{Min}$, retorna-se ao passo *ii*.
- Se todas as células restantes têm $\hat{P}_i > \hat{P}_{Min}$, o processo de subdivisão é encerrado.

Na implementação computacional, optou-se por uma tolerância angular (utilizada para definir uma planura aceitável) de 5^o, o que corresponde a um $\hat{P}_{Min} = 0{,}99616$. Testes numéricos mostraram que valores angulares menores tornam o custo computacional elevado sem uma contrapartida significativa em precisão, além de aumentar muito o número de pontos de interseção. Como a precisão do processo de interseção por subdivisão não é uniforme, pois depende da forma das superfícies, seria praticamente inviável obter uma precisão numérica aceitável em função da tolerância angular. Para que os resultados atinjam uma precisão uniforme e dentro da faixa requerida, é necessário utilizar um algoritmo de refinamento (ver Seção 3.1.5). Assim, utiliza-se uma tolerância angular que não é excessivamente baixa, mas que garante resultados que são uma boa aproximação inicial para o processo de refinamento.

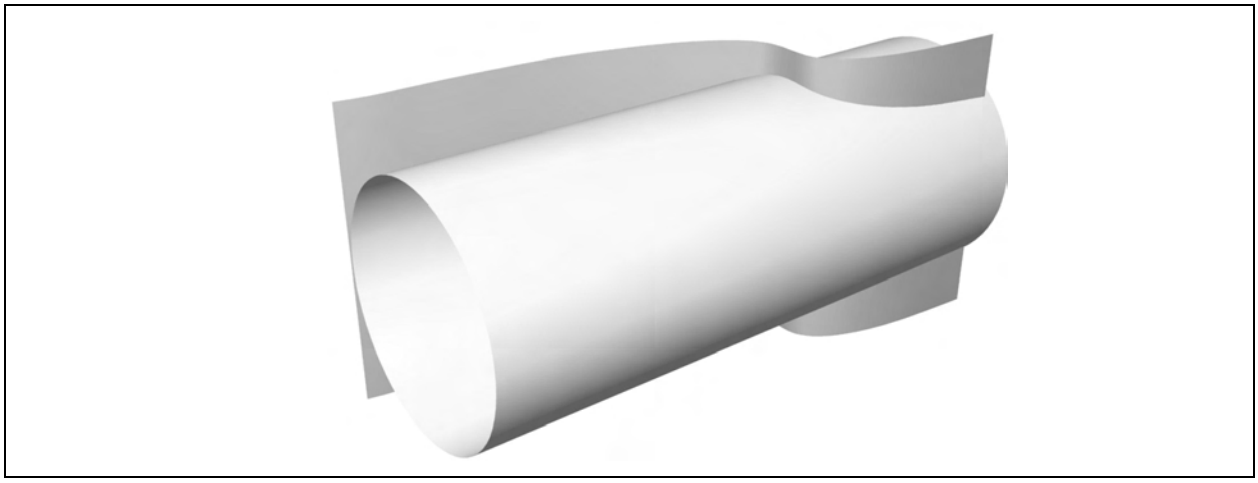


Figura 3.4 – Exemplo 3.1: uma superfície Sweep e um cilindro de revolução.

Para ilustrar o processo de subdivisão, será utilizado o Exemplo 3.1 (Fig. 3.4) que contém duas superfícies com interseção positiva. A Figura 3.5 ilustra como o algoritmo de subdivisão adaptativa atua sobre as duas superfícies a cada passo de iteração. É possível observar os trechos com seus volumes envolventes no espaço real e a configuração equivalente no espaço paramétrico. Cada coluna corresponde a um passo de iteração. As linhas superior e inferior

mostram as *quadtrees* no espaço paramétrico para cada superfície. Nas linhas centrais, os volumes envolventes das células ativas são mostrados sobre a superfície no espaço 3D. As células hachuradas representam trechos de superfície cujos VE interferem. Apenas estas permanecem para a iteração seguinte. O número de iterações depende da forma das superfícies, mas, em geral, 5 ou 6 iterações são suficientes[1]. É possível verificar que, nas duas superfícies, as subdivisões convergem para uma solução comum no espaço real. Ao final do processo de subdivisão, obtém-se, para cada superfície, uma lista de células que possuem interferências (indicadas por seus vínculos). Na próxima etapa, serão verificadas e calculadas as interseções entre os trechos vinculados, as quais constituirão a primeira aproximação dos segmentos de interseção no espaço 3D.

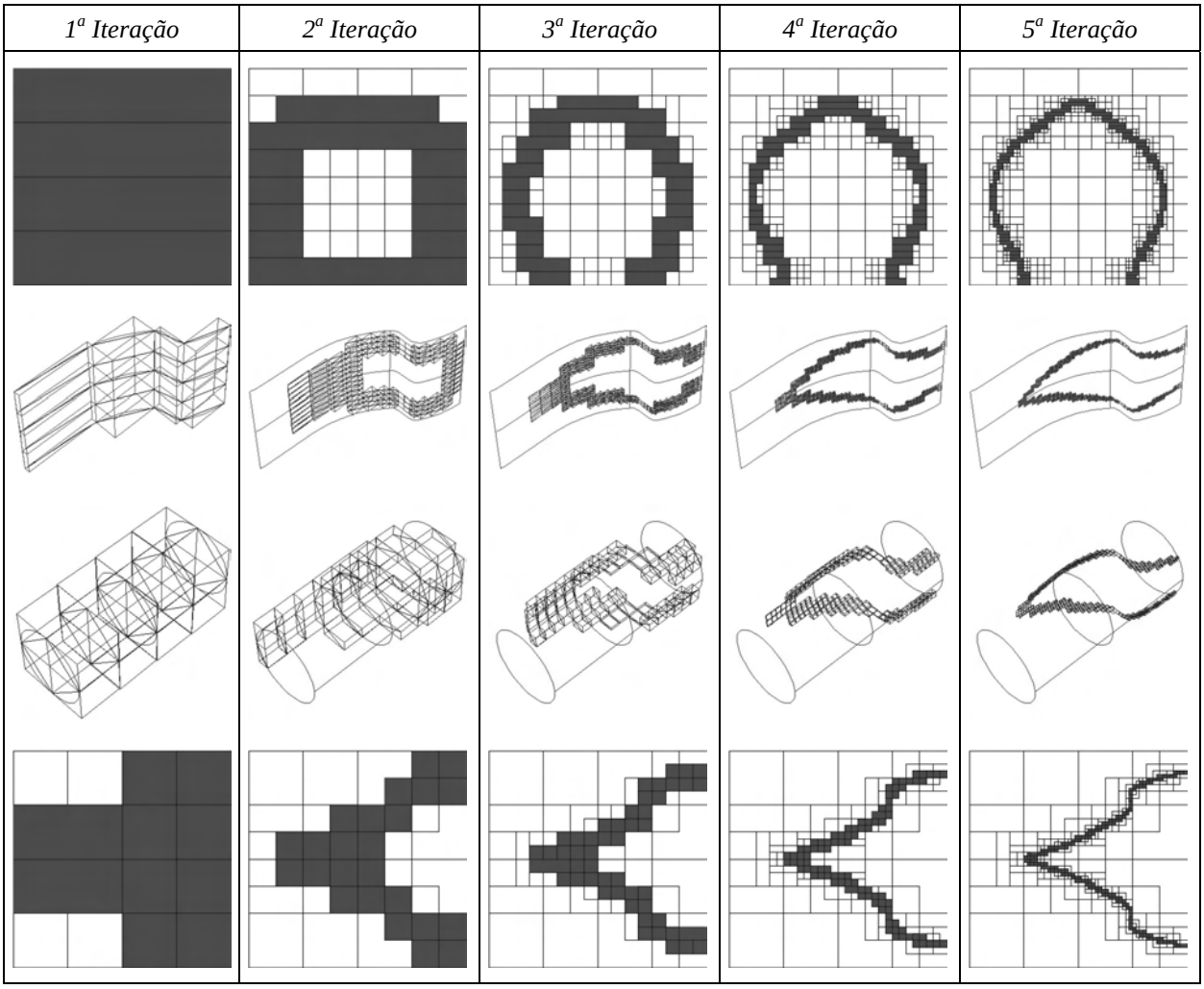


Figura 3.5 – Processo de subdivisão adaptativa. Cada coluna corresponde às subdivisões em cada iteração no espaço paramétrico e no espaço real com os volumes envolventes. As porções em escuras correspondem aos trechos onde há interferência em cada iteração.

[1] Este número de iterações irá depender da tolerância angular utilizada.

3.1.4. Determinação dos Segmentos de Interseção no Espaço 3D

Nesta etapa, determinam-se os segmentos das linhas de interseção no espaço 3D através da interseção dos trechos das duas superfícies correspondentes às células vinculadas. Os trechos são subdivididos em triângulos, aproximando localmente as superfícies por regiões planas. Assim, o cálculo dos segmentos de interseção é feito determinado-se as interseções entre os triângulos (dois a dois) dos trechos vinculados das duas superfícies.

3.1.4.1. Divisão em triângulos

A subdivisão do domínio obtida na etapa anterior não é uniforme, podendo conter células com diferentes números de células adjacentes (vizinhas). Se a divisão em triângulos não for feita com uma topologia adequada, poderão ocorrer falhas (*gaps*) ao longo da linha de interseção, como a que aparece na Figura 3.6. O padrão de divisão em triângulos irá depender da topologia da *quadtree* de cada superfície, de tal forma que se garanta a continuidade das linhas de interseção.

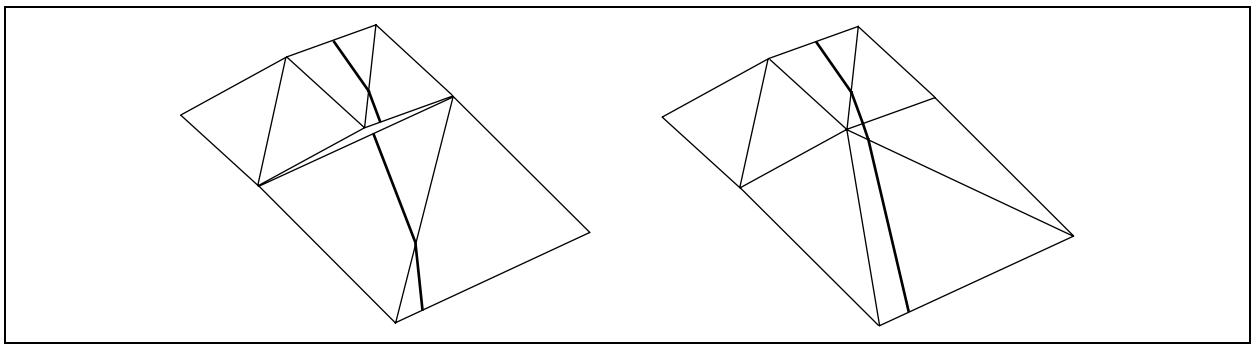


Figura 3.6 – Cada célula restante do processo de subdivisão é dividida em triângulos em função da topologia da *quadtree*, com o objetivo de prevenir a ocorrência de *gaps*.

Computam-se, para todas as células, o número de células adjacentes aos seus lados. Este número pode ser 1 ou 2, pois o algoritmo de subdivisão não permite transições com diferença superior a um nível, o que corresponde a 2 células adjacentes em um lado. Foram estabelecidas cinco (5) configurações possíveis de topologia de divisão em triângulos, conforme o número células adjacentes. Quando os quatro lados de uma célula possuem apenas um vizinho por lado, o trecho correspondente é dividido em dois triângulos (Fig. 3.7a). Quando um dos lados da célula possui dois vizinhos, o trecho é dividido em três triângulos, criando-se um vértice intermediário (na metade do lado célula) em que há dois vizinhos (Fig. 3.7b). Se dois lados

possuem dois vizinhos, podem ocorrer duas situações: os lados com dois vizinhos são adjacentes ou não. Em ambos os casos, são criados quatro triângulos, mas as configurações topológicas são distintas (Fig. 3.7c-d). Há, ainda, a possibilidade de haver uma célula com três lados com dois vizinhos. Neste caso, são necessários cinco triângulos, com três novos vértices intermediários (Fig. 3.7e). Estas cinco configurações asseguram que a malha de triângulos não tenha falhas na região da linha de interseção.

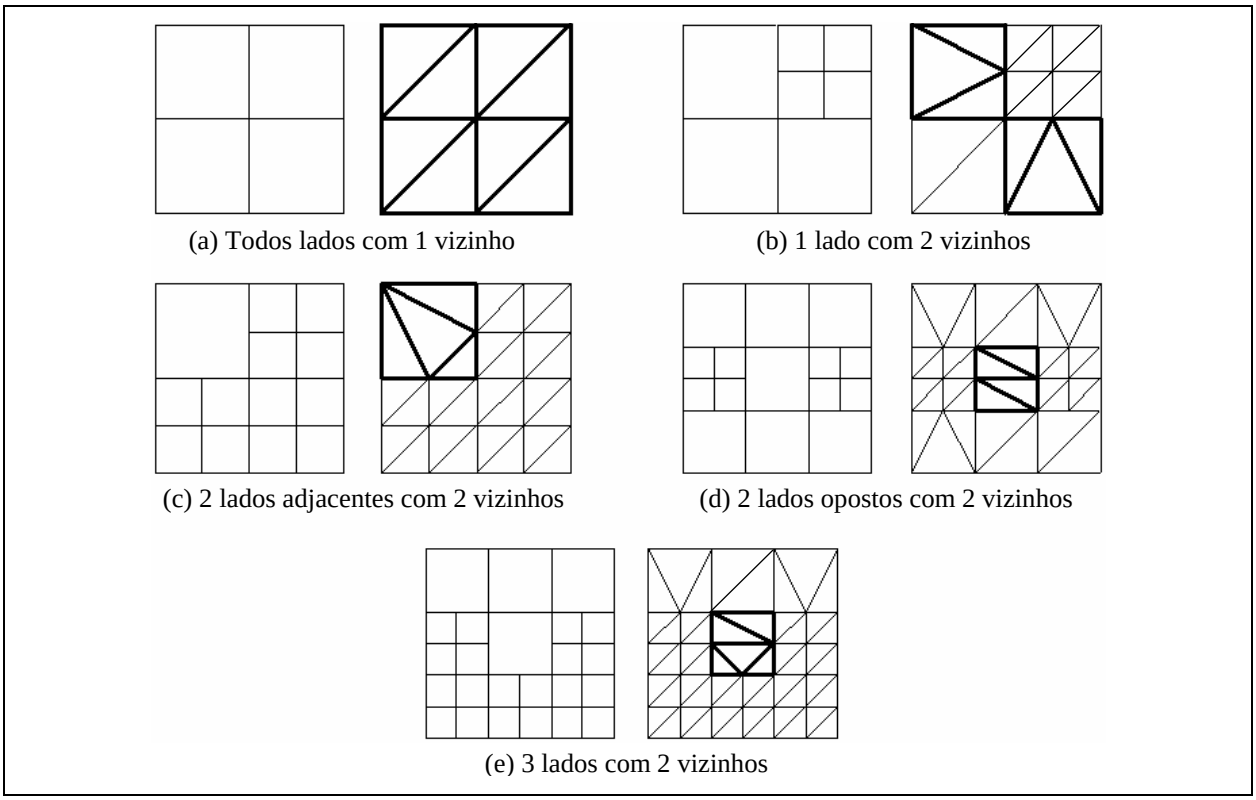


Figura 3.7 – Topologias possíveis para composição triangular dos trechos de superfície.

Cada segmento da linha de interseção resulta da interseção entre dois triângulos das duas superfícies. Um único trecho, correspondente a uma célula, pode conter vários segmentos resultantes de interseções entre vários triângulos, podendo chegar a dezenas, dependendo do número de trechos vinculados e do número de triângulos em cada um (função da topologia da *quadtree*). Por exemplo: considerando que um trecho pode ser dividido em até 5 triângulos, seriam necessárias 25 verificações de interseções para determinar a interseção entre apenas dois trechos e, ainda, se este trecho possui 4 vínculos, o número de interseções entre triângulos a serem computadas para um único trecho seria de 100. No entanto, isto não chega a comprometer o desempenho do algoritmo, pois o procedimento para determinar as interseções entre os triângulos garante velocidade ao processo.

3.1.4.2. Cálculo da Interseção entre Dois Triângulos no Espaço 3D

A interseção entre dois triângulos no espaço 3D envolve transformações de coordenadas e análises topológicas. O algoritmo desenvolvido utiliza um sistema de referência auxiliar acoplado a um dos triângulos para facilitar o processo de cálculo da interseção. Utilizam-se representações paramétricas dos lados dos triângulos como parte do procedimento para determinar o segmento de interseção.

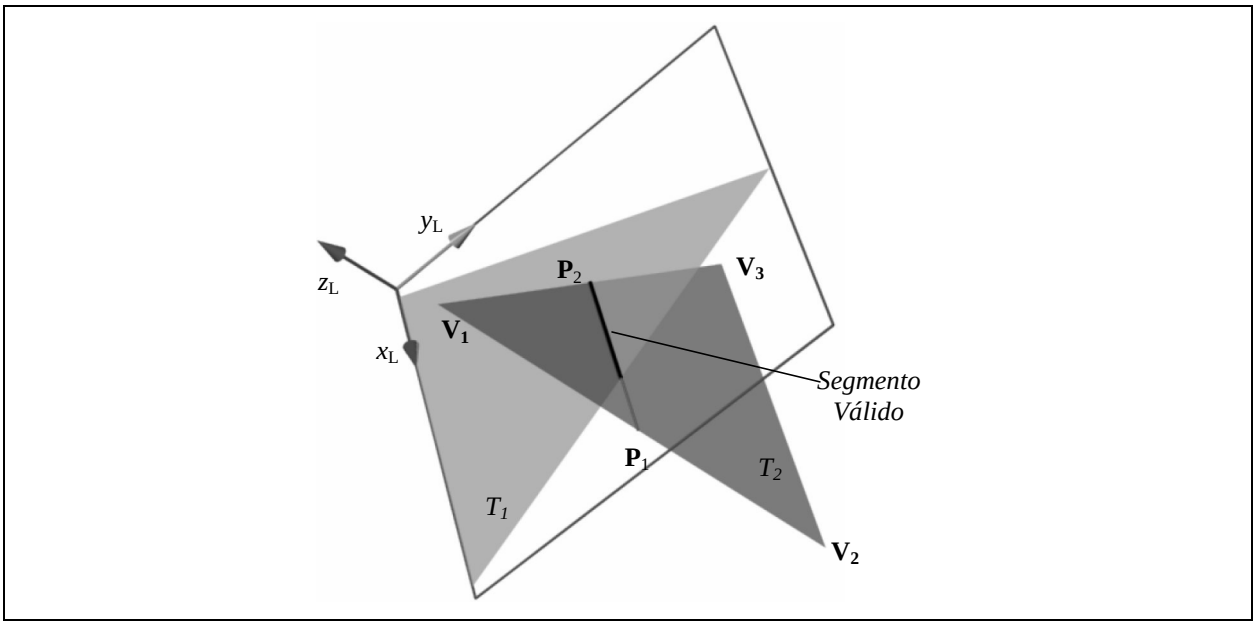


Figura 3.8 – Interseção entre dois triângulos (T_1 e T_2) no espaço 3D.

A seguir, são descritos os principais passos do algoritmo para cálculo da interseção entre dois triângulos T_1 e T_2 (ver Fig. 3.8).

i. Um Sistema de Referência Auxiliar (x_L, y_L, z_L) é acoplado a T_1. O plano do triângulo fica coincidente com o plano *xy* local.

ii. Os vértices de T_2 têm suas coordenadas transformadas para o SRA de T_1.

iii. Comparam-se os sinais das coordenadas z_L dos vértices de T_2:

- Caso todos os sinais sejam iguais, não há interseção e Fim.
- Se houver sinais diferentes[2], significa que T_2 penetra o plano de T_1.

iv. Os pontos de interseção entre T_2 e o plano de T_1 ocorrem em $z_L = 0$. Dois pontos, $\mathbf{P}_1$ e $\mathbf{P}_2$, resultam da interseção de dois lados de T_2 com o plano de T_1 (*xy* local). As representações paramétricas dos dois lados de T_2, aqueles que cruzam o plano x_Ly_L, são utilizadas para

[2] Esta é uma condição necessária, mas não suficiente, para que haja interseção entre T_2 e T_1.

computar $\mathbf{P}_1$ e $\mathbf{P}_2$: $\mathbf{L}_1(t) = (1-t).\mathbf{V}_1 + t.\mathbf{V}_2$ e $\mathbf{L}_2(s) = (1-s).\mathbf{V}_1 + s.\mathbf{V}_3$. $\mathbf{P}_1$ e $\mathbf{P}_2$ são encontrados com os valores de *t* e *s* que tornam $z_L = 0$. Esta operação é mostrada a seguir:

$$\begin{aligned} 0 &= (1-t).z_1 + t.z_2\ , \quad t = \frac{z_1}{z_1 - z_2}\ , \\ 0 &= (1-s).z_1 + s.z_3\ , \quad s = \frac{z_1}{z_1 - z_3}\ , \end{aligned} \tag{3.2}$$

onde z_1, z_2, e z_3 são os valores locais de *z* dos vértices de T_2. Os valores válidos de *t* e *s* devem estar no intervalo [0,1]. As coordenadas x_L e y_L são calculadas substituindo os valores de *t* e *s* nas equações paramétricas correspondentes. Assim, $\mathbf{P}_1$ e $\mathbf{P}_2$ são obtidos analiticamente com valores exatos.

v. Se o segmento definido pelos pontos $\mathbf{P}_1$ e $\mathbf{P}_2$ estiver total o parcialmente contido em T_1, então existe interseção entre T_1 e T_2.

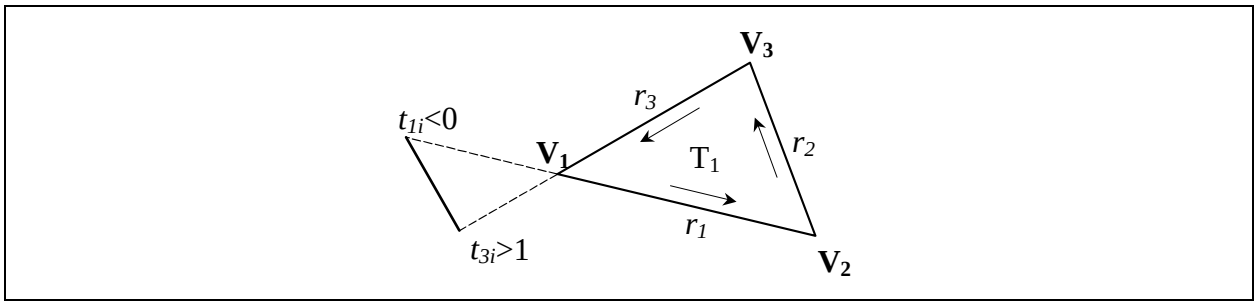


Figura 3.9 – Quando os valores dos parâmetros t_{ji} estão fora de [0,1], não há interseção.

Um segmento de interseção válido deve pertencer a ambos os triângulos. As equações paramétricas dos segmentos são utilizadas para resolver o problema. São analisados quatro segmentos: os três lados de T_1 e o segmento definido por $\mathbf{P}_1$ e $\mathbf{P}_2$. São computadas as interseções entre o segmento ($\mathbf{P}_1\mathbf{P}_2$) e os lados de T_1. O resultado é um parâmetro que *t* para cada segmento. Isto permite a avaliação da posição do segmento $\mathbf{P}_1\mathbf{P}_2$ em relação a T_1. $r_i(t_i)$ e $r_j(t_j)$, com j = 1 .. 3, são, respectivamente, as equações paramétricas do segmento interseção e dos lados de T_1. A interseção entre r_i e r_j, com resultados em termos de r_i, é expressa por:

$$t_{ik} = \text{Int}(r_i, r_k) \text{ e } t_{il} = \text{Int}(r_i, r_l) \tag{3.3}$$

e a interseção entre r_j e r_i, com resultados em termos de r_j, é expressa por:

$$t_{ki} = \text{Int}(r_k, r_i), \text{ e } t_{li} = \text{Int}(r_l, r_i) \tag{3.4}$$

onde Int(.,.) é a função que determina a interseção entre duas retas em termos paramétricos e r_k e r_l são os dois lados de T_1 tais que $0 \le t_{ki} \le 1$ e $0 \le t_{li} \le 1$. Esta é a condição necessária e suficiente para que exista interseção entre os triângulos. Quando dois valores de t_{ji} estão fora do intervalo [0,1], não existe interseção entre T_1 and T_2 (Fig. 3.9).

Os parâmetros t_{ij} e t_{ik} são analisados para determinar se o segmento de interseção está fora de T_1, total ou parcialmente dentro de T_1. Quando há interseção, uma das seguintes situações ocorre:

i. $t_{il} < 0$ e $t_{ik} > 1$: O segmento interseção está totalmente dentro de T_1 (Fig. 3.10a);

ii. $0 < t_{ik} < 1$ e $0 < t_{il} < 1$: O segmento cruza T_1, mas as extremidades estão fora (Fig. 3.10b);

iii. $0 < t_{ik} < 1$ e ($t_{il} > 1$ ou $t_{il} < 0$): O segmento está parcialmente dentre de T_1 (Fig. 3.10c).

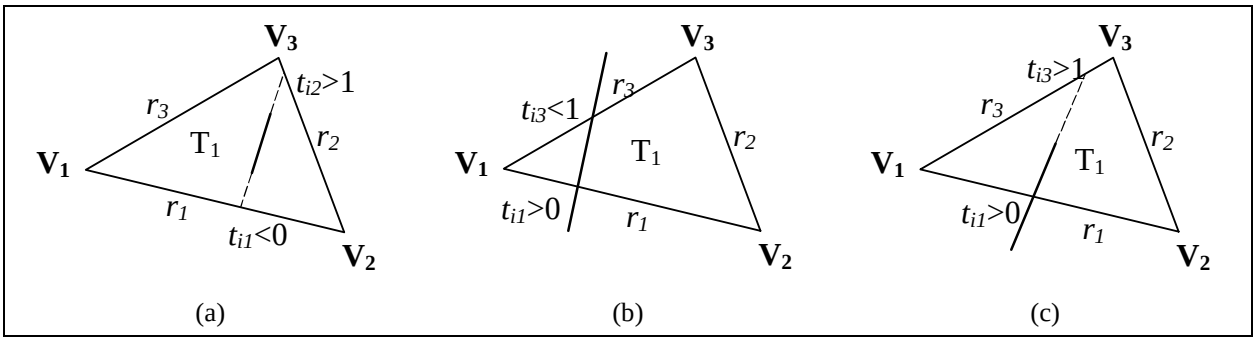


Figura 3.10 – Situações em que ocorre interseção entre T_1 e T_2.

O segmento válido de interseção corresponde à porção que está dentro de T_1. Desta forma, pode ser necessário corrigir as coordenadas da extremidades do segmento de interseção utilizando as coordenadas paramétricas t_{ij}. Para a situação *i* (Fig. 3.10a), nada muda e $\mathbf{P}_1$ e $\mathbf{P}_2$ permanecem os mesmos. Na situação *ii* (Fig. 3.10b), as duas extremidades devem ser atualizadas: $\mathbf{P}_1 = r_i(t_{il})$ e $\mathbf{P}_2 = r_i(t_{jk})$. Na situação *iii* (Fig. 3.10c), é necessário ajustar uma das extremidades: $\mathbf{P}_k = r_i(t_{il})$. Neste caso, *k* é a extremidade (1 ou 2) que está fora de T_1 e t_{il} deve estar contido entre 0 e 1.

Após determinar o segmento de interseção entre os dois triângulos, os pontos são convertidos para o sistema de referência do Universo e armazenados em um arranjo próprio. O segmento é armazenado em um arranjo de conectividades que contém ponteiros para o arranjo de pontos. Após calcular a interseção entre os triângulos de todos os trechos das duas superfícies, o resultado é um conjunto de segmentos conectado pelas extremidades, os quais representam uma aproximação das linhas de interseção entre as duas superfícies no espaço 3D (Fig. 3.11). Na

próxima etapa, realiza-se o refinamento e a projeção sobre as superfícies dos pontos de interseção.

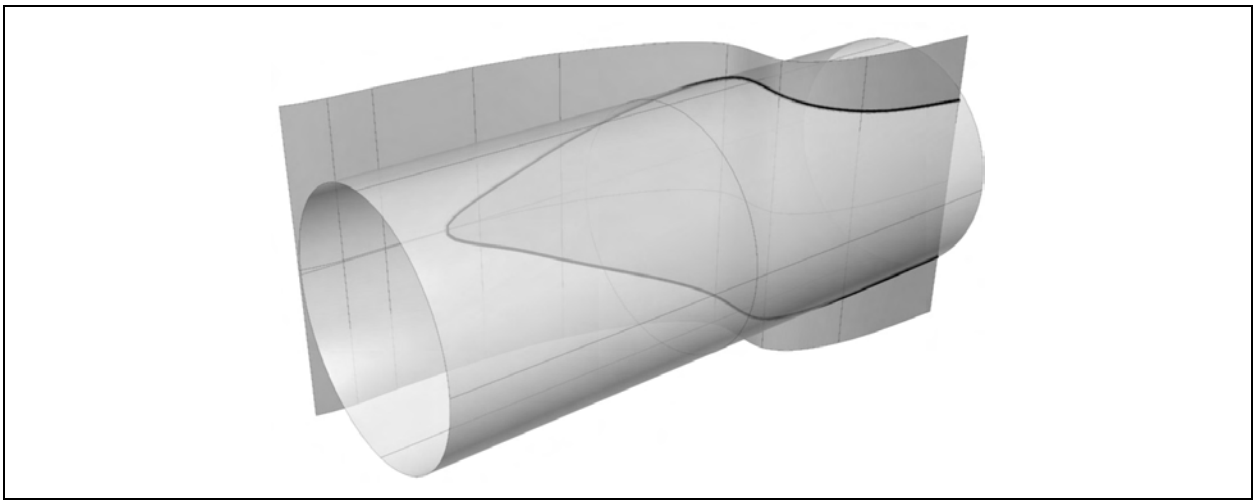


Figura 3.11 – Primeira aproximação da linha de interseção no espaço real.

3.1.5. Refinamento dos Resultados e Mapeamento Paramétrico dos Pontos de Interseção

Como mencionado anteriormente, a precisão dos pontos de interseção resultantes do processo de subdivisão não é uniforme e haveria um grande custo computacional se a tolerância angular fosse excessivamente reduzida. Utilizou-se um processo de refinamento que garante uniformidade e a precisão requerida nos pontos de interseção. Os resultados que comprovam a eficiência do processo de refinamento estão na Seção 3.3 de exemplos. O algoritmo de refinamento desenvolvido é do tipo Newton-Raphson e utiliza as direções das tangentes das duas superfícies para incrementar a precisão dos pontos de interseção obtidos. O processo é baseado no trabalho de Houghton *et al.*, 1985, mas ao contrário deste, determina os pontos em coordenadas paramétricas sobre as duas superfícies. A precisão é da ordem 10^{-12} da dimensão máxima das superfícies envolvidas. Os pontos são mapeados em coordenadas paramétricas, através de um algoritmo especialmente desenvolvido, que obtém resultados com um erro máximo de 10^{-15} em coordenadas absolutas.

3.1.5.1. Erro relativo

A precisão mencionada acima refere-se ao valor máximo admissível para o erro relativo, o qual é calculado pela distância entre dois pontos das duas superfícies, resultantes da projeção sobre as superfícies do mesmo ponto de interseção, relacionada com a dimensão

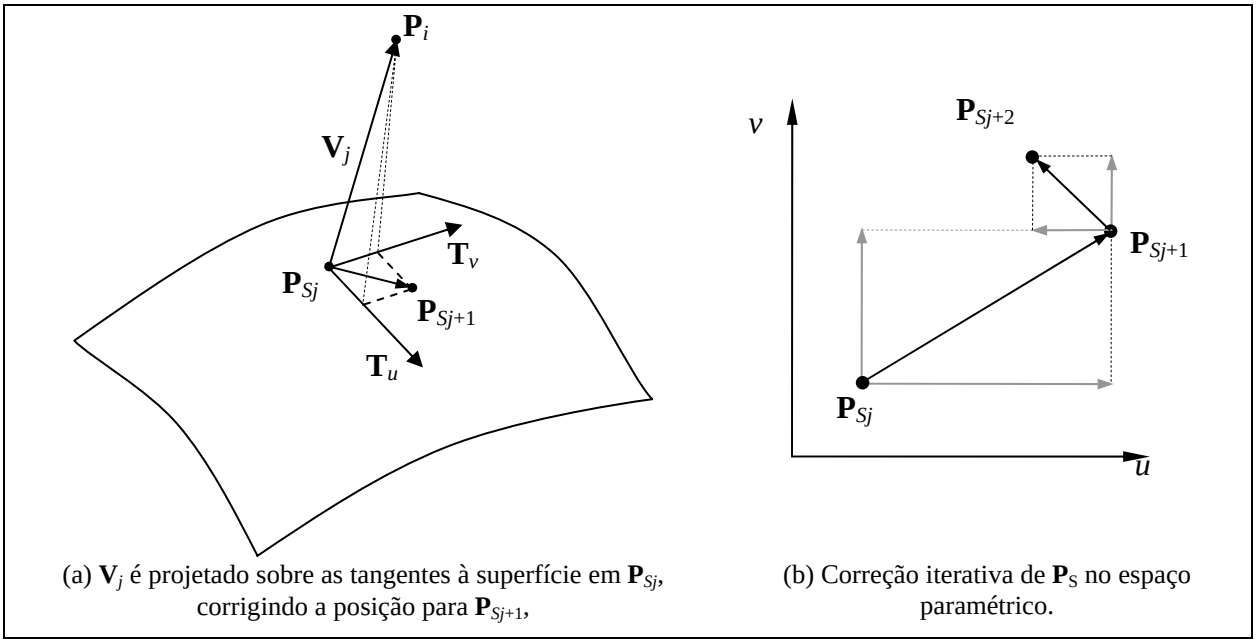
máxima das superfícies envolvidas. Assim, o erro relativo (E_i) de um ponto de interseção $\mathbf{P}^i$ é definido por:

$$E_i = \frac{\left\| \mathbf{P}_{S1}^i - \mathbf{P}_{S2}^i \right\|}{D_{Max}} \tag{3.5}$$

onde $\mathbf{P}_{S1}^i$ e $\mathbf{P}_{S2}^i$ são as projeções de $\mathbf{P}^i$ sobre as duas superfícies e D_{Max} é a maior dimensão, considerando-se as duas superfícies envolvidas. O algoritmo de refinamento realiza a minimização deste erro para melhorar a qualidade dos resultados.

3.1.5.2. Mapeamento Paramétrico

O algoritmo de mapeamento paramétrico projeta um ponto $\mathbf{P}_i$ do espaço 3D sobre uma superfície paramétrica. O resultado é um ponto sobre a superfície em coordenadas paramétricas. Não é possível estabelecer uma solução analítica genérica quando utilizam-se diferentes tipos de parametrizações. Assim, emprega-se um processo numérico iterativo, que utiliza as direções das tangentes sobre a superfície para acelerar a convergência para o ponto procurado.



(a) $\mathbf{V}_j$ é projetado sobre as tangentes à superfície em $\mathbf{P}_{Sj}$, corrigindo a posição para $\mathbf{P}_{Sj+1}$,

(b) Correção iterativa de $\mathbf{P}_S$ no espaço paramétrico.

Figura 3.12 – Projeção de um ponto sobre uma superfície.

O algoritmo baseia-se no fato de que a projeção de um ponto sobre uma superfície é feita a partir de uma normal à superfície. Consequentemente, o vetor definido pelo ponto e sua

projeção e os vetores tangentes à superfície no ponto de projeção devem ter produtos internos nulos. Quando estes não são nulos, podem ser utilizados para deslocar o ponto projetado para uma nova posição que, se espera, seja mais próxima da posição correta (Fig. 3.12).

Considerando que o ponto será projetado sobre um trecho conhecido da superfície, o ponto central deste trecho $\mathbf{P}_{S1}$ é utilizado como uma aproximação inicial da projeção de $\mathbf{P}_i$ sobre a superfície e dará início ao processo iterativo. As direções das tangentes em $\mathbf{P}_{S1}$ são computadas a partir das derivadas parciais em relação a *u* e *v*, obtidas numericamente. O ponto $\mathbf{P}_i$ é projetado sobre as tangentes e estas projeções, convertidas para o espaço paramétrico, corrigem a aproximação inicial. O processo é repetido até que se atinja a precisão desejada.

Considerando-se um ponto $\mathbf{P}_i$ no espaço 3D e um trecho de superfície paramétrica S_i próxima a $\mathbf{P}_i$, o algoritmo realiza as seguintes operações:

i. Um fator de conversão local (f_{RP}) entre as medidas paramétricas e 3D é calculado:

$$f_{RP} = D_P / D_R \,, \tag{3.6}$$

onde D_P é a distância paramétrica sobre a superfície e D_R é a distância 3D correspondente.

ii. O ponto central de $\mathbf{S}_i$ é computado no espaço paramétrico ($\mathbf{P}^{p}_{Sj}$, $j = 1$), e no espaço 3D ($\mathbf{P}_{Sj}$). Esta é a primeira aproximação da projeção de $\mathbf{P}_i$ sobre a superfície.

iii. Determina-se o vetor $\mathbf{V}_j = \mathbf{P}_i - \mathbf{P}_{Sj}$.

iv. Calculam-se os vetores tangentes ($\mathbf{T}_u$, $\mathbf{T}_v$) em $\mathbf{P}_{Sj}$ nas direções paramétricas *u* e *v*.

v. O vetor $\mathbf{V}_i$ é projetado sobre $\mathbf{T}_u$ e $\mathbf{T}_v$ e estas projeções são utilizadas para atualizar $\mathbf{P}^{p}_{Sj}$:

$$\Delta_u = \mathbf{V}_j \bullet \mathbf{T}_u \tag{3.7}$$

$$\Delta_v = \mathbf{V}_j \bullet \mathbf{T}_v \tag{3.8}$$

$$\mathbf{\Delta}_p = [\Delta_u, \Delta_v] \tag{3.9}$$

vi. A nova aproximação é dada por:

$$\mathbf{P}^{p}_{S\,j+1} = \mathbf{P}^{p}_{S\,j} + \mathbf{\Delta}_p \cdot f_{RP} \tag{3.10}$$

$$\mathbf{P}_{S\,j+1} = \mathbf{S}(\mathbf{P}^{p}_{S\,j+1}) \tag{3.11}$$

onde **S** é a função paramétrica da superfície.

vii. Se (Δ_s e Δ_t) < *Tolerância* então $\mathbf{P}_S$ é a projeção de $\mathbf{P}_i$, caso contrário retorna ao passo *iii*.

No final do processo, $\mathbf{P}_S$ é a projeção ortogonal de $\mathbf{P}_i$ sobre a superfície e $\mathbf{P}_S^p$ é a projeção em coordenadas paramétricas. A tolerância adotada de 10^{-15} corresponde ao resíduo máximo aceitável no valor das projeções sobre os vetores tangentes. Este elevado grau de precisão é necessário para garantir a convergência e a precisão do algoritmo de refinamento. A Figura 3.12 ilustra as etapas do algoritmo de projeção.

3.1.5.3. O Algoritmo de Refinamento

Um ponto $\mathbf{P}_i$ do espaço 3D projetado sobre duas superfícies resulta em dois outros pontos ($\mathbf{P}_{S1}$, $\mathbf{P}_{S2}$) que são as duas projeções. Se o ponto $\mathbf{P}_i$ está sobre a linha de interseção a distância entre $\mathbf{P}_{S1}$ e $\mathbf{P}_{S2}$ é nula. O algoritmo de refinamento utiliza os vetores tangentes às superfícies em $\mathbf{P}_{S1}$ e $\mathbf{P}_{S2}$ para deslocar $\mathbf{P}_i$ com o objetivo de minimizar a distância entre $\mathbf{P}_{S1}$ e $\mathbf{P}_{S2}$. A idéia do algoritmo é utilizar as direções das tangentes às duas superfícies sobre o plano que contém $\mathbf{P}_i$, $\mathbf{P}_{S1}$ e $\mathbf{P}_{S2}$ para determinar uma nova aproximação de $\mathbf{P}_i$.

A seguir são descritas as principais etapas do algoritmo de refinamento considerando um ponto $\mathbf{P}_i$ próximo à interseção de duas superfícies:

i. O ponto $\mathbf{P}_i$ é projetado sobre as duas superfícies em $\mathbf{P}_{S1}^p$ e $\mathbf{P}_{S2}^p$ em coordenadas paramétricas e $\mathbf{P}_{S1}$ e $\mathbf{P}_{S2}$ no espaço real.

ii. Se o erro relativo E_i < *Tolerância* (Eq. 3.5), vai para *v*.

iii. Dois vetores tangentes ($\mathbf{T}_1$ em $\mathbf{P}_{S1}$ e $\mathbf{T}_2$ em $\mathbf{P}_{S2}$) às duas superfícies são determinados de tal forma que pertençam ao plano definido pelos pontos $\mathbf{P}_i$, $\mathbf{P}_{S1}$ e $\mathbf{P}_{S2}$.

iv. A posição de $\mathbf{P}_i$ é atualizada para a interseção entre as retas definidas por $\mathbf{T}_1$ e $\mathbf{T}_2$. Retorna ao passo *i*.

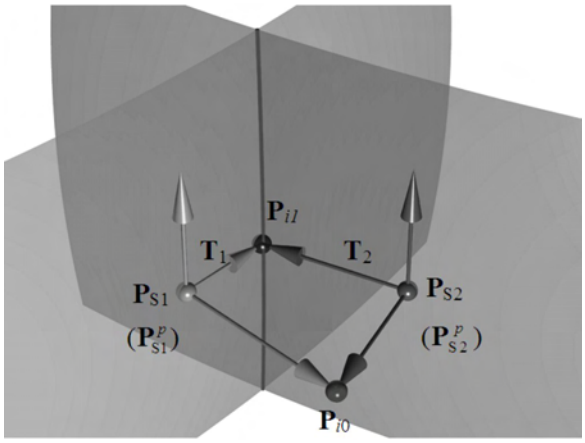
v. As posições paramétricas $\mathbf{P}_{S1}^p$ e $\mathbf{P}_{S2}^p$ são armazenadas em arranjos específicos para cada superfície. O arranjo de conectividades permanece inalterado, pois o refinamento não gera mudanças topológicas.

A Figura 3.13a mostra o esquema de funcionamento do algoritmo de refinamento para 1 iteração. A convergência é, em geral, muito rápida (1 a 2 iterações) para superfícies de curvatura constante. Em superfícies de curvatura variável, podem ser necessárias mais iterações, mas geralmente não são necessárias mais do que 4 de iterações. Um tolerância de 10^{-12} relativa à máxima dimensão das superfícies envolvidas foi adotada para que os resultados possam ser utilizados em diferentes aplicações de CAGD. O tempo de processamento desta etapa é

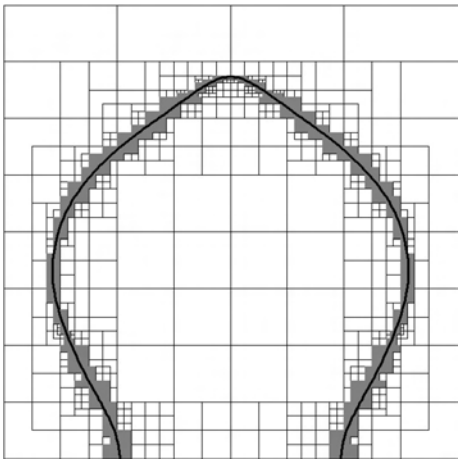
dominante em relação às demais etapas do algoritmo de interseção devido, principalmente, ao grande número operações vetoriais e de ponto flutuante necessárias ao processo.

Depois de o algoritmo de refinamento ser aplicado em todos os pontos de interseção calculados, as linhas de interseção ficam definidas em segmentos retos no espaço paramétrico de cada superfície. Estes segmentos já constituem, em seu conjunto, a solução para o problema de interseção com um alto grau de precisão. No entanto, para que seja possível manipular as linhas de interseção como objetos geométricos, é necessário reordenar os segmentos e, ainda, criar representações paramétricas das curvas de interseção.

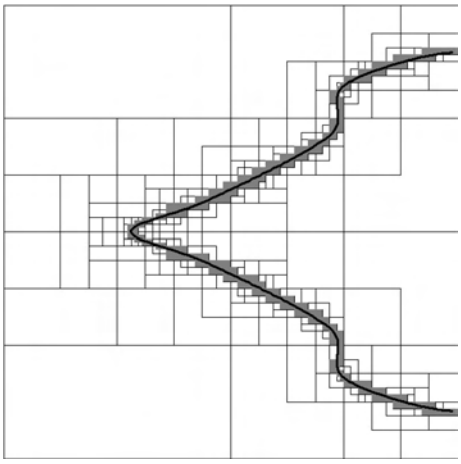
As Figuras 3.13b e 3.13c apresentam as representações, no espaço paramétrico, dos segmentos de interseção para o Exemplo 3.1 obtidos a partir do refinamento, com mapeamento paramétrico, da aproximação inicial no espaço 3D (Figs. 3.4, 3.5 e 3.11).



(a) Esquema de refinamento em 1 iteração.



(b) Fita Sweep



(c) Cilindro de Revolução

Figura 3.13 – Esquema de refinamento (a) e curva de interseção representada no espaço paramétrico das duas superfícies (b e c) do Exemplo 3.1.

3.1.6. Reordenação dos Segmentos e Parametrização das Curvas de Interseção

O objetivo desta etapa é transformar o conjunto de segmentos em um conjunto ordenado de pontos sucessivos de definem linhas contínuas. As conectividades dos segmentos são analisadas para determinar as extremidades das linhas de interseção e para identificar os segmentos adjacentes. Isto já permite determinar alguns aspectos da topologia das linhas de interseção. Linhas abertas possuem duas extremidades e linhas fechas não possuem extremidades. As extremidades são utilizadas para iniciar o processo de montagem das linhas. Nas curvas fechadas os pontos iniciais são escolhidos arbitrariamente.

O algoritmo é simples. Cada ponto adicionado à curva é chamado de Pivô. O primeiro Pivô é uma extremidade, que é um ponto com um único segmento incidente. O algoritmo encontra um ponto conectado ao Pivô por um segmento. Este ponto é o novo Pivô. O processo é repetido até a outra extremidade da linha.

Depois da reordenação, as curvas são definidas por um conjunto de pontos ordenados que definem segmentos retos nos espaços paramétricos das duas superfícies. Entretanto, esta não é a melhor representação para uma manipulação eficiente das curvas de interseção como objetos geométricos. Uma representação paramétrica tipo $f(t)$ seria muito mais apropriada. Esta representação é obtida utilizando as distâncias reais entre os pontos para definir coordenadas paramétricas dos pontos em função do comprimento total da curva. Como no espaço 3D a curva é única, é possível utilizar a mesma representação paramétrica das curvas para as duas superfícies. A representação paramétrica de uma linha é determinada de seguinte forma:

i. O comprimento total da curva (L_n) é calculado adicionando as distâncias entre os pontos consecutivos no espaço 3D:

$$L_n = \sum_{i=1}^{n-1} \left\| \mathbf{P}_{i+1} - \mathbf{P}_i \right\| \tag{3.12}$$

onde n é o número total de pontos da curva.

ii. Assim, o comprimento acumulado até um ponto k é L_k.

iii. A coordenada paramétrica de um ponto é dada por:

$$t_i = \frac{L_i}{L_n} \tag{3.13}$$

iv. A função paramétrica da curva utiliza uma interpolação linear no espaço paramétrico da superfície para calcular pontos sobre a curva:

$$\mathbf{f}_C(t) = \mathbf{P}_j^p.(1-t_m(t)) + \mathbf{P}_{j+1}^p.t_m(t) \tag{3.14}$$

$$t_m(t) = \frac{t - t_j}{t_{j+1} - t_j} \tag{3.15}$$

Depois de calculados os comprimentos parciais para cada ponto sucessivo da linha de interseção, calculam-se as coordenadas paramétricas correspondentes a cada ponto em função do comprimento total da linha de interseção. Desta forma, cada um dos pontos da linha tem definida a sua posição paramétrica ($0 \leq t \leq 1$) em função do comprimento da linha. Assim, quando é necessário determinar um ponto para uma coordenada *t* específica, buscam-se os dois pontos cujas coordenadas limitam o valor procurado. O ponto final é obtido pela interpolação linear, no campo paramétrico, entre estes dois pontos limitantes.

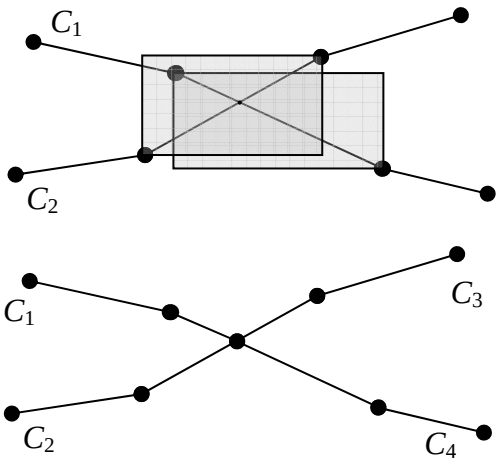
A equação (3.14) fornece as coordenadas paramétricas (*u*, *v*) para a curva na superfície correspondente. Todas as curvas de interseção ficam representadas no espaço paramétrico das duas superfícies. Isto permite manipular de forma independente cada uma das superfícies. Como a parametrização é função do comprimento no espaço 3D, apesar das linhas estarem sobre superfícies diferentes, para os mesmos valores do parâmetro *t* fornecem o mesmo ponto no espaço 3D. Isto é fundamental para o processo de geração de malha em domínios compostos por sub-domínios adjacentes de superfícies diferentes, onde deve ser garantida a continuidade das malhas. Esta característica falha quando as linhas apresentam topologias diferentes nas duas superfícies. Isto acontece quando em uma superfície a curva é aberta e na outra é fechada, ou quando em uma superfície a curva é inteira e na outra corresponde a 2 ou mais. Seria necessário compatibilizar as topologias das duas superfícies para sempre garantir uma coincidência de parametrização.

3.2. INTERSEÇÕES ENTRE VÁRIAS SUPERFÍCIES

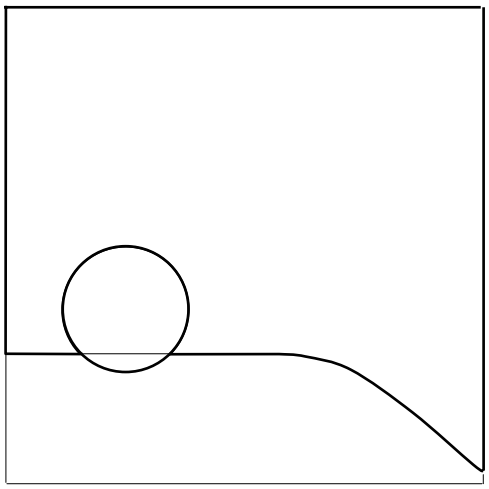
O algoritmo desenvolvido determina a interseção entre duas superfícies. Mas é comum ocorrerem situações em que há mais de duas superfícies com interseções entre si. E, em muitos casos, há interseção entre próprias linhas de interseção. Devido a importância deste tipo de situação, é necessário tratar também este problema, o que incrementa o potencial de uso do algoritmo proposto no modelamento geométrico de objetos complexos.

A interseção simultânea entre várias superfícies é calculada como mesmo algoritmo de interseção e as superfícies são analisadas duas a duas, não havendo necessidade de alterar o

algoritmo. No entanto, as linhas de interseção são armazenadas cumulativamente na estrutura de dados (Propriedades) das superfícies, de tal forma que cada superfície pode armazenar um conjunto ilimitado de linhas de interseção, provenientes de interseções com superfícies diferentes.



(a) Interseção entre linhas de interseção



(b) Exemplo de cruzamentos de linhas em um domínio com um sub-domínio definido

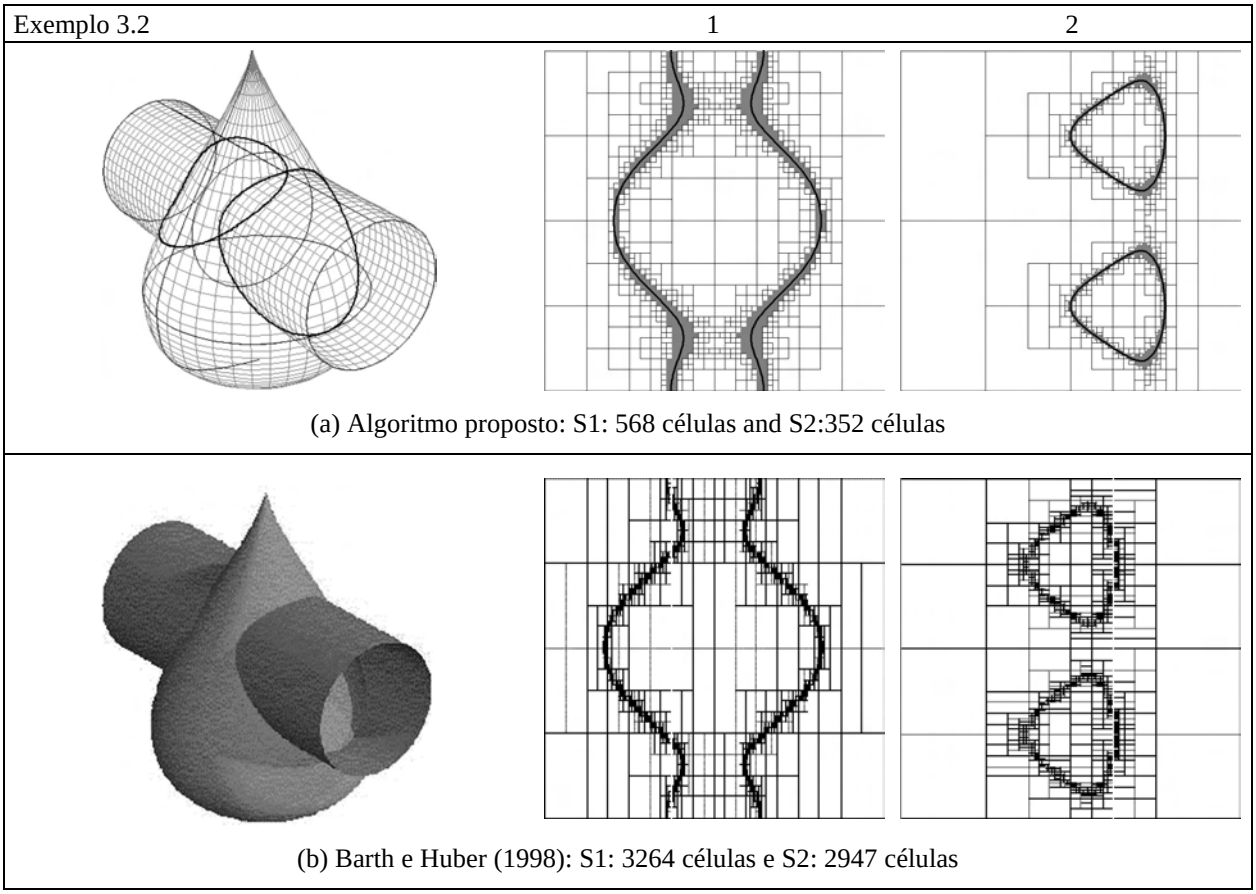
Figura 3.14 – Determinação de interseção entre superfícies de interseção.

Com várias linhas de interseção sobre a mesma superfície, pode surgir o problema do cruzamento de linhas. Para resolver este problema, foi desenvolvido uma versão simplificada do algoritmo de subdivisão, agora aplicada a curvas no espaço paramétrico de uma superfície. O algoritmo utiliza áreas envolventes[3] para todos os trechos das linhas de interseção da superfície no espaço paramétrico. Verificam-se aqueles segmentos cujos volumes possuem interferência, determinando-se o ponto de interseção entre as duas[4] linhas (Fig. 3.14a). Este processo é simples, pois os segmentos que constituem as linhas de interseção são retos no espaço paramétrico. Quando uma interseção entre as linhas é determinada, estas são cortadas nestes pontos. Isto torna possível criar seqüências de linhas conectadas pelas extremidades, o que permite a definição de sub-domínios compostos por linhas provenientes da interseção com várias superfícies diferentes (Fig. 3.14b).

[3] Para o caso 2D, os VE são reduzidos a retângulos que envolvem os segmentos.
[4] O processo sempre verifica as linhas duas a duas.

3.3. EXEMPLOS DE INTERSEÇÕES

Nesta seção, são apresentados vários exemplos, todos modelados e analisados com o T-CADE, envolvendo diversos tipos de superfícies para testar a eficiência e a versatilidade do algoritmo de interseção. A Tabela 3.1 mostra um resumo dos resultados obtidos para os exemplos de 1 a 15. Os Exemplos são apresentados nas Figuras 3.15 a 3.19, sendo que a situação 3D é mostrada a esquerda e a situação no espaço paramétrico de cada superfície e mostrada no lado direito.



(a) Algoritmo proposto: S1: 568 células and S2:352 células

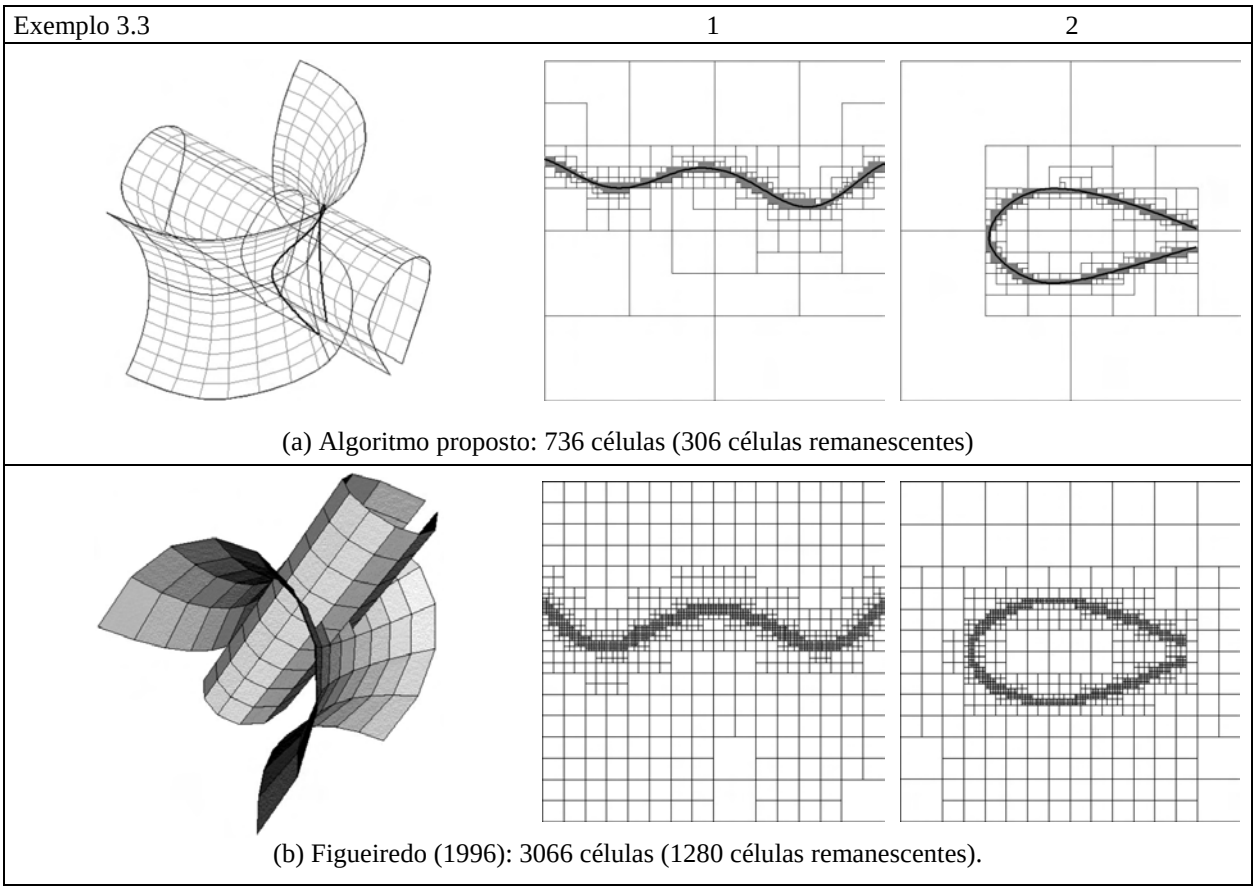
(b) Barth e Huber (1998): S1: 3264 células e S2: 2947 células

Figura 3.15 – Exemplo 3.2.

O Exemplo 3.1 foi utilizado na descrição do algoritmo (Fig. 3.4). O Exemplo 3.2 (Fig. 3.15a) é utilizado para comparar o algoritmo proposto com o trabalho de Barth e Huber, 1998, e Huber, 1999, (Fig. 3.15b). As configurações das subdivisões obtidas são muito semelhantes, graficamente, às obtidas por Barth-Huber, que utiliza volumes envolventes justos. Seu trabalho não utiliza tolerâncias angulares e sim a condição: $A_{Di} = A_{D0}/10000$, onde A_{Di} é a área da célula i no espaço paramétrico e A_{D0} é a área da superfície no espaço paramétrico. Esta é a razão para a grande profundidade nas subdivisões das superfícies (um total de 6211 células). O

algoritmo proposto resolve o mesmo problema de interseção com 920 células e volumes envolventes orientados pelos eixos globais. Os modelos dos Exemplos 3.2 e 3.3 foram construídos por semelhança a partir das referências mencionadas. Os resultados foram comparados com as mesmas referências.

O Exemplo 3.3 compara resultados com o trabalho de Figueiredo, 1996, o qual utiliza uma variação do algoritmo de Gleicher e Kass, 1992. O algoritmo de Figueiredo utiliza a Aritmética Afim para computar os volumes envolventes enquanto que o algoritmo de Gleicher e Kass utiliza a Aritmética Intervalar. O algoritmo de Figueiredo necessitou de 3066 subdivisões (células), restando 1280 células no final do processo. O algoritmo original de Gleicher-Kass necessita de 8038 células, sendo 5508 remanescentes. O algoritmo proposto utiliza um total de 736 células, sendo 306 remanescentes.



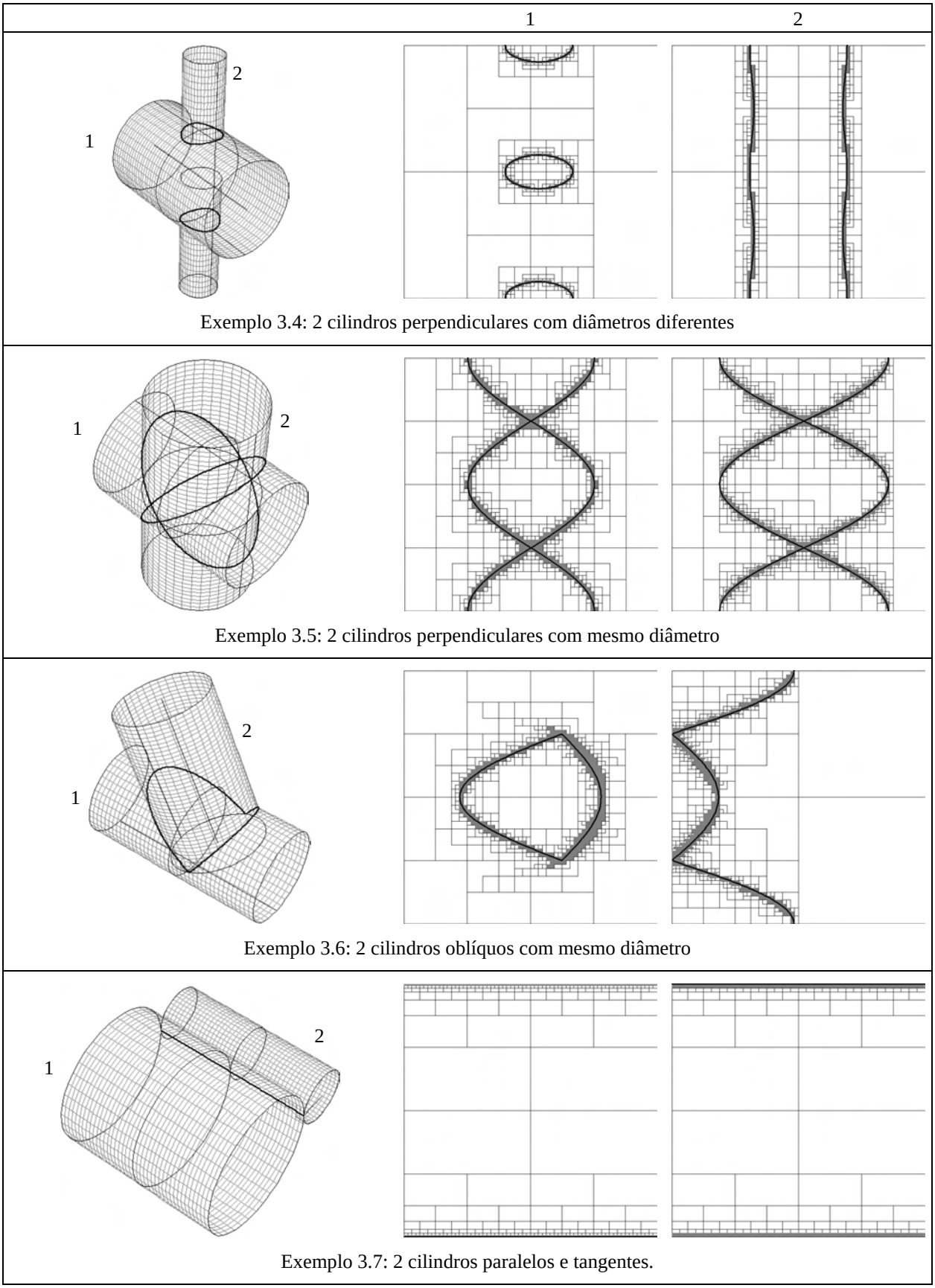
(a) Algoritmo proposto: 736 células (306 células remanescentes)

(b) Figueiredo (1996): 3066 células (1280 células remanescentes).

Figura 3.16 – Exemplo 3.3.

Os próximos Exemplos, 3.4 a 3.15 (Fig. 3.19 – 3.21), demonstram o uso do algoritmo em diversas situações de geometria e topologia. Os Exemplos de 3.4 a 3.7 mostram interseções entre cilindros, gerados como superfícies de revolução, dispostos em várias posições diferentes, simulando situações que podem ocorrer em problemas de Engenharia, tais como

tubulações industriais.



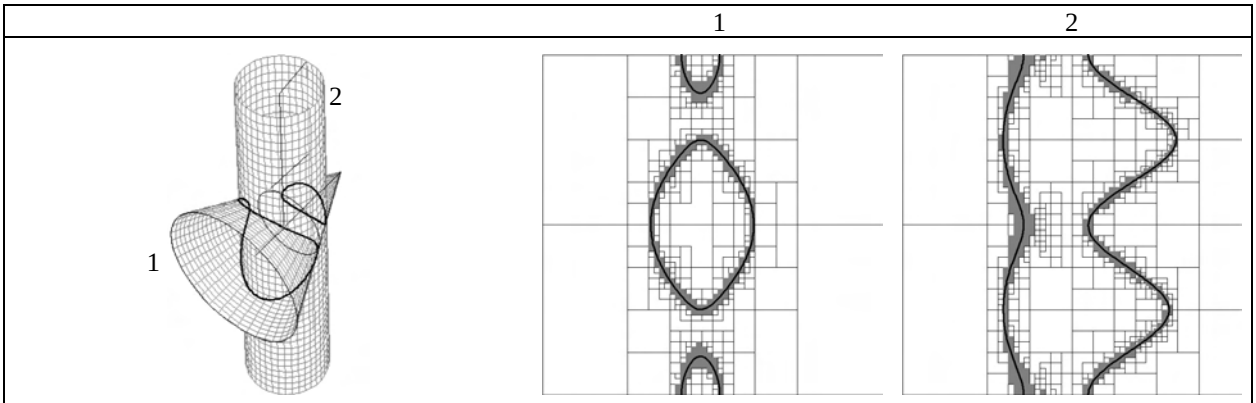
Exemplo 3.4: 2 cilindros perpendiculares com diâmetros diferentes

Exemplo 3.5: 2 cilindros perpendiculares com mesmo diâmetro

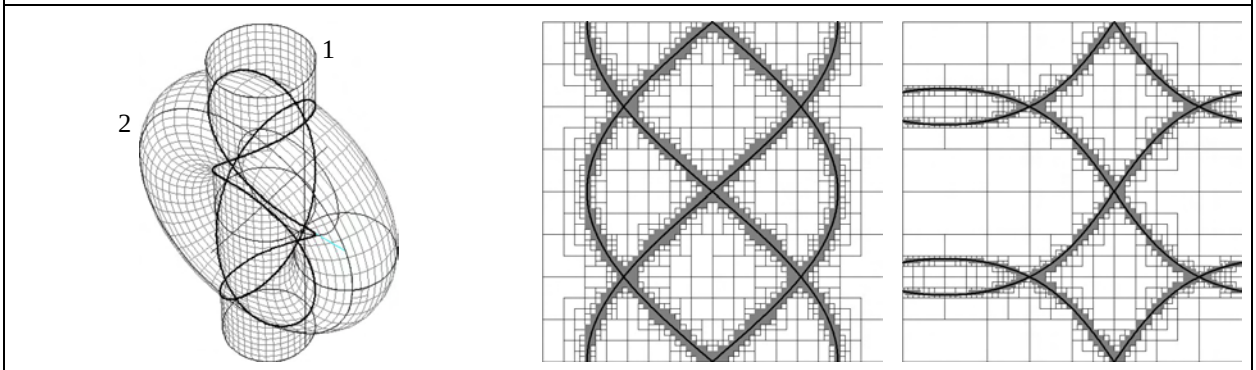
Exemplo 3.6: 2 cilindros oblíquos com mesmo diâmetro

Exemplo 3.7: 2 cilindros paralelos e tangentes.

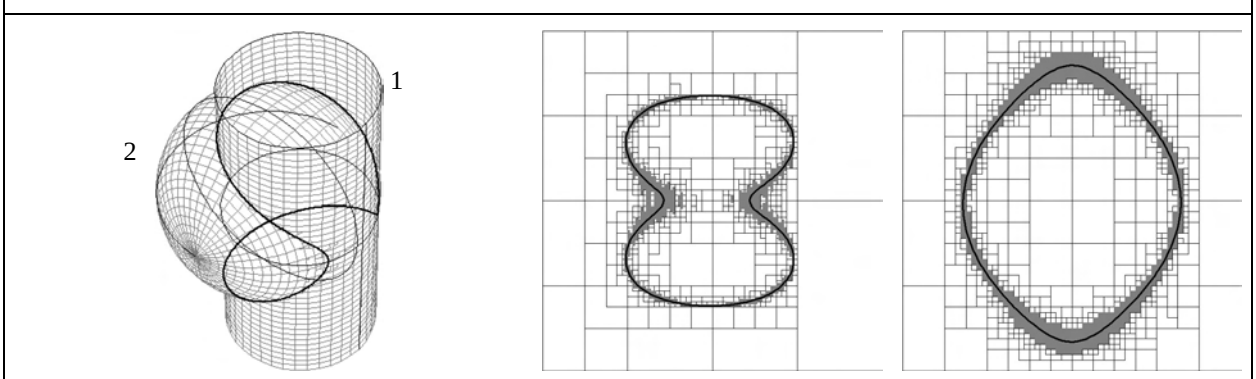
Figura 3.17 – Exemplos 3.4 a 3.7.



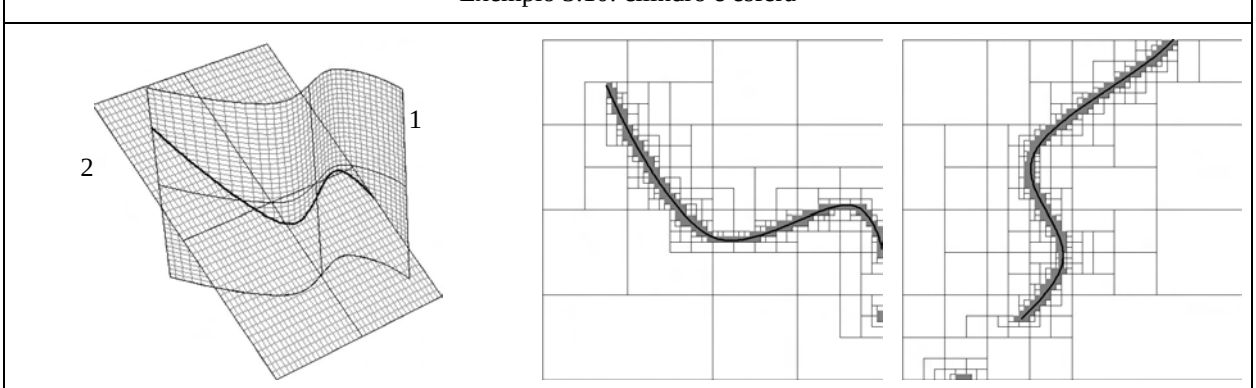
Exemplo 3.8: cilindro e cone com eixos perpendiculares



Exemplo 3.9: cilindro e toro com mesmo diâmetro de tubo

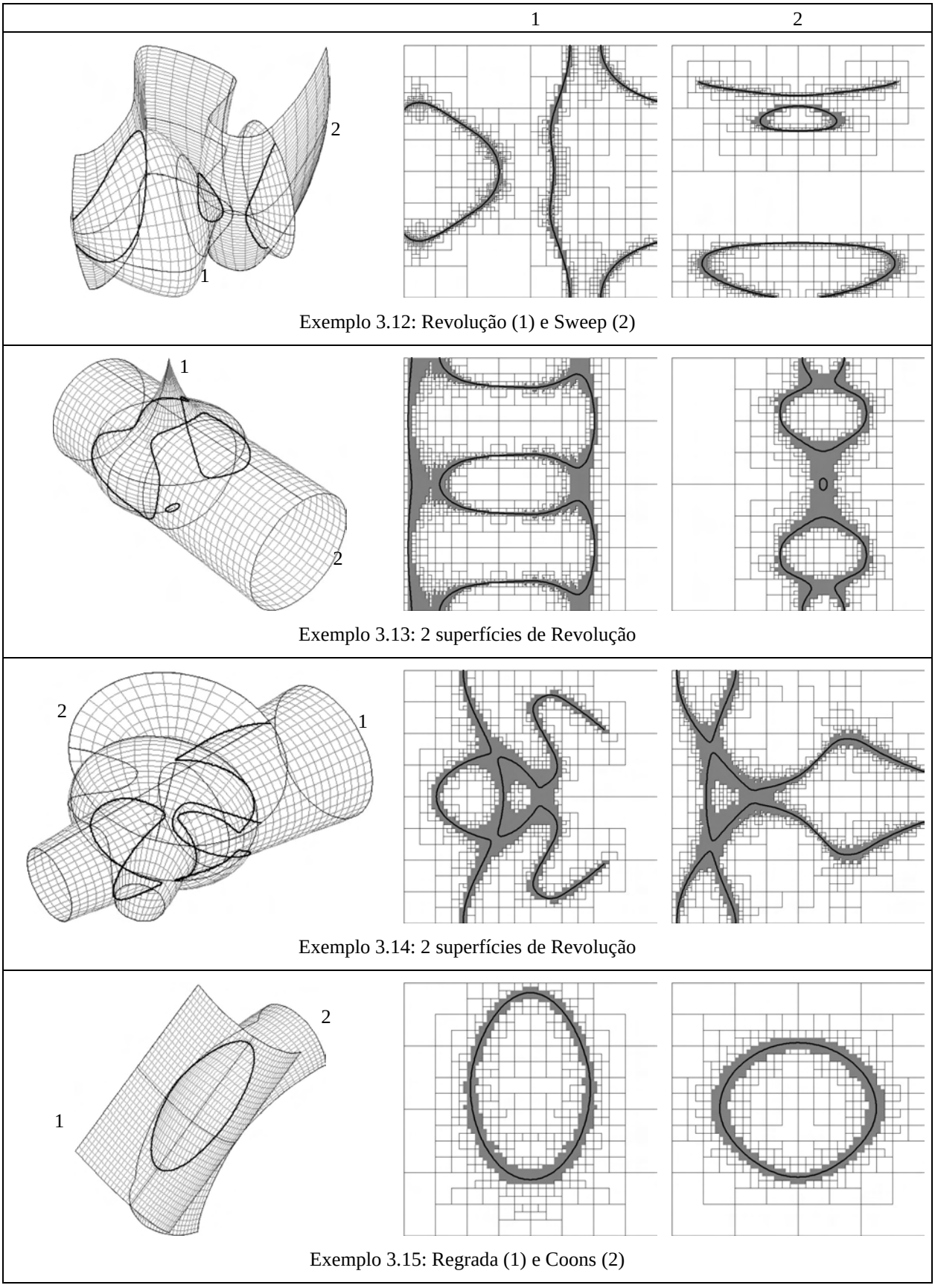


Exemplo 3.10: cilindro e esfera



Exemplo 3.11: Plano (Regrada) e superfície Sweep

Figura 3.18 – Exemplos 3.8 a 3.11.



Exemplo 3.12: Revolução (1) e Sweep (2)

Exemplo 3.13: 2 superfícies de Revolução

Exemplo 3.14: 2 superfícies de Revolução

Exemplo 3.15: Regrada (1) e Coons (2)

Figura 3.19 – Exemplos 3.12 a 3.15.

O Exemplo 3.7 testa a robustez do algoritmo calculando a interseção entre duas superfícies tangentes. Os exemplos de 3.8 a 3.10 testam o cálculo de interseções entre superfícies geométricas primitivas: cilindro, cone, toro e esfera. Os exemplos de 3.12 a 3.15 mostram interseções entre superfícies de geometrias complexas, com vários tipos de parametrizações, que testam a capacidade do algoritmo em situações limite.

A Tabela 3.1 mostra os resultados para os exemplos de 3.1 a 3.15, incluindo, Erro (máximo e médio), número de pontos calculados, número de células ou subdivisões utilizadas, número de curvas e o tempo de processamento. Na grande maioria dos Exemplos o erro máximo foi inferior a 10^{-10} e o erro médio foi inferior a 10^{-11}, com exceção do Exemplo 14, onde o erro máximo foi da ordem de 10^{-5} e o erro médio em torno de 10^{-8}. Como a precisão usual em programas CAD é da ordem de 10^{-6} a 10^{-8}, a precisão obtida é, em geral, equivalente ou superior. O tempo de processamento refere-se a um Pentium IV 2.4 GHz com 256MB. A etapa de refinamento é a que consome a maior parte do tempo. A velocidade de processamento cresce sensivelmente com a redução da tolerância. O Exemplo 3.13 consome 36 segundos de CPU com tolerância 10^{-12} e cai para 25 segundos (-30,6%) quando a tolerância é de 10^{-8}. A tolerância pode ser customizada na implementação em função da precisão requerida pela aplicação.

Tabela 3.1 – Resultados obtidos com o algoritmo proposto para os Exemplos de 3.1 a 3.15.

	Nº Células S1		Nº Células S2		Núm.	Curvas		Erro Relativo		Tempo
Ex.	Total	Rem.	Total	Rem.	Pontos	S1	S2	Max	Med.	CPU (s)
3.1	986	348	516	222	608	1	1	2.157009e-11	2.088392e-12	7
3.2	1116	568	1178	352	832	2	2	3.443295e-11	2.759666e-12	5
3.3	411	173	325	133	372	1	1	1.288117e-10	4.007308e-11	13
3.4	536	260	468	144	616	3	2	1.788262e-11	3.192538e-13	3
3.5	2471	1339	951	334	1242	6	6	2.154993e-11	2.166294e-12	6
3.6	658	235	1031	523	541	1	1	2.027047e-11	2.766512e-12	3
3.7	508	256	252	128	129	1	1	4.898425e-16	4.898425e-16	1
3.8	552	209	1004	442	679	3	2	2.426449e-11	1.156698e-12	3
3.9	1789	690	1772	922	1488	12	14	2.509866e-11	2.162789e-12	8
3.10	1195	416	1604	872	890	1	1	1.738579e-11	2.159538e-12	5
3.11	346	134	434	165	329	1	1	2.047152e-11	1.067454e-12	4
3.12	1721	622	2017	936	1472	4	3	7.022341e-10	2.883347e-11	36
3.13	1468	690	2634	1412	1225	2	3	4.471619e-11	2.518051e-12	11
3.14	1250	538	2856	1576	1216	2	3	1.918307e-05	7.844887e-08	11
3.15	1036	566	1004	544	510	1	1	4.239043e-12	7.908833e-13	5

3.4. CONSIDERAÇÕES FINAIS

Os resultados obtidos nos 15 exemplos apresentados comprovam a eficiência do algoritmo proposto para determinar a interseção entre superfícies de diversos tipos de parametrização, com formas e posições variadas. O tipo de parametrização pode influir na velocidade de processamento, em função da complexidade das equações paramétricas, o que afeta a avaliação de pontos sobre a superfície e, consequentemente, diversas etapas do algoritmo. A forma das superfícies também afeta a velocidade de processamento, pois é determinante na etapa de subdivisão, o que influi no número de pontos de interseção, afetando a etapa de refinamento, que é dominante em relação ao tempo total de processamento.

No entanto, o algoritmo é suficientemente robusto para obter as linhas de interseção independentemente das formas ou singularidades que estas possam ter. Esta é uma das principais características dos métodos de subdivisão. Mesmo em situações críticas como superfícies que se tangenciam, ou quase se tangenciam, o algoritmo determinou as linhas de interseção com grande precisão e a velocidade de processamento mostrou-se compatível com aplicações de modelamento interativo.

As curvas paramétricas obtidas no espaços paramétricos das duas superfícies são utilizadas para manipular a geometria das superfícies através de sub-domínios. Desta forma, é possível construir modelos complexos unindo recortes de superfícies para utilizar em processos de geração de malha (ver Cap. 4 e 6) para as diversas aplicações do modelamento geométrico, tais como simulações e análises numéricas por Elementos Finitos.

4. GERAÇÃO DE MALHAS EM SUPERFÍCIES PARAMÉTRICAS RECORTADAS

A geração de malhas consiste em uma das etapas mais importantes deste trabalho, pois são as malhas os dados fundamentais para definir a geometria na análise pelo Método dos Elementos Finitos. O objetivo nesta etapa é gerar malhas de elementos triangulares sobre retalhos de superfícies paramétricas que podem ter contornos arbitrários, permitindo o modelamento de superfícies complexas.

O algoritmo proposto neste trabalho é do tipo Frontal (*advancing front*) e utiliza uma malha de fundo (*backgrownd mesh*), construída através de subdivisões recursivas, para regular o tamanho dos elementos em função da curvatura local das diferentes regiões da superfície. A escolha por um método Frontal é devida, principalmente, à característica deste tipo de método de permitir a construção de malhas em domínios com contornos arbitrários, como os que ocorrem após a determinação das linhas de interseção entre duas superfícies.

4.1. O ALGORITMO PROPOSTO

O algoritmo utiliza tolerâncias angulares aplicadas aos vetores normais das curvas e da superfície, de modo a levar em conta as curvaturas locais, tanto na discretização do contorno, como na geração da malha de fundo. O algoritmo está organizado em cinco etapas:

- Determinação de sub-domínios;
- Discretização das linhas de contorno;
- Geração da malha de fundo, para estabelecer o tamanho dos elementos, em função da curvatura e do contorno, através da subdivisão do domínio;
- Geração da malha não estruturada com o método Frontal;
- Suavização da malha.

4.1.1. Definição de Sub-domínios

A malha é gerada sobre todo o domínio ou em sub-domínios de uma superfície paramétrica, que podem ser resultados de interseções entre superfícies, ou mesmo, definidos pelo usuário. Um sub-domínio é sempre definido no espaço paramétrico e pode ser composto por até três classes de linhas: linhas de contorno, linhas de furos e linhas internas.

Uma linha de contorno é um conjunto de segmentos conectados que define as fronteiras de um sub-domínio sobre a superfície. A malha se propaga do contorno para dentro do sub-domínio. Um furo é conjunto de segmentos conectados pelas extremidades que forma uma

região interna a um contorno. Neste caso, a malha propaga-se para fora da linha de furo. Linhas internas são conjuntos de segmentos conectados formando linhas abertas ou fechadas e são sempre internas aos sub-domínios. Na prática, as linhas internas servem para forçar a criação de nós em posições específicas como regiões de interface entre superfícies e zonas onde há condições de contorno determinadas, como vínculos e carregamentos. A Figura 4.1 mostra um esquema com os tipos de linha de um domínio e a forma de propagação da malha.

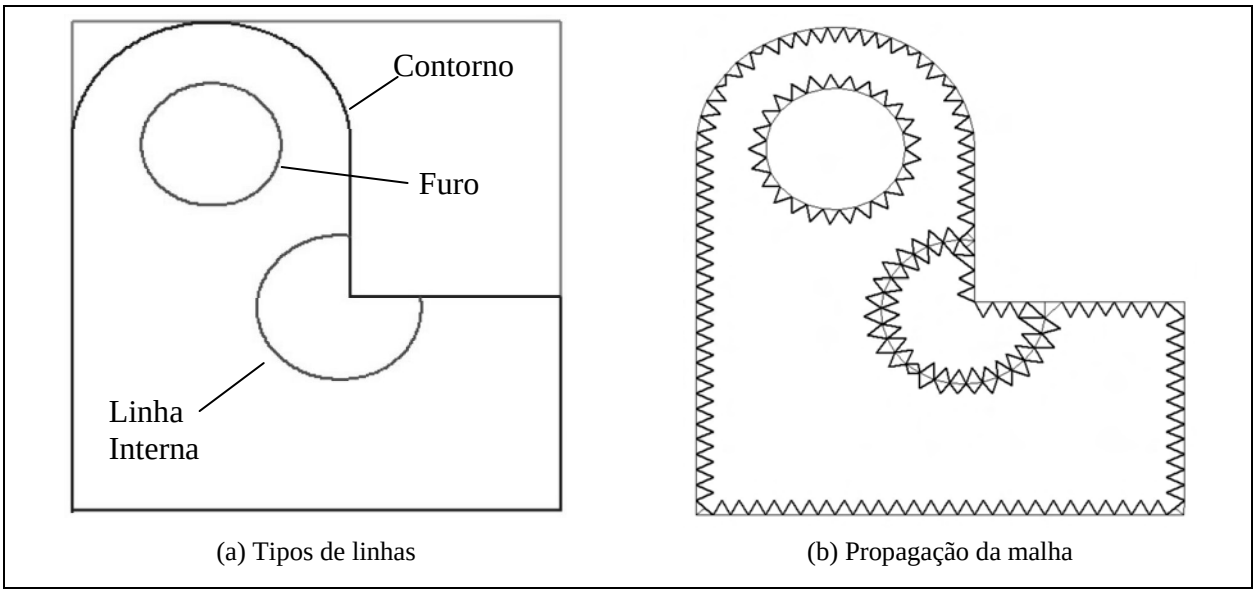


Figura 4.1 – Relação entre tipos de linhas e forma de propagação da malha.

Neste trabalho, optou-se pela definição manual dos sub-domínios através de uma interface gráfica, que é mostrada na Seção 5.1.5. Após a determinação dos domínios, realiza-se a discretização dos contornos para a obtenção dos pontos que irão definir os segmentos do Front inicial, o qual servirá como base para dar a partida no algoritmo Frontal.

4.2.1 Discretização das Linhas de Contorno

Uma discretização adequada das linhas de contorno é crucial para o desempenho do algoritmo de geração de malha e para a qualidade dos elementos, principalmente daqueles próximos ao contorno. O algoritmo de discretização do contorno leva em conta a curvatura das linhas do contorno variando o comprimento dos segmentos de contorno. Por outro lado, é importante que as variações de comprimento sejam suaves para evitar transições abruptas, o que também iria repercutir na qualidade da malha gerada.

A curvatura da linha de contorno é determinada em função do ângulo entre vetores tangente sobre a linha. O ângulo cresce com o aumento da curvatura. Assim, estabeleceu-se uma

tolerância angular ($\hat{A}_{Max}$) que limita o comprimento dos segmentos do contorno, de forma que as extremidades tenham vetores tangentes com ângulo menor ou igual a $\hat{A}_{Max}$. A tolerância angular, assim como os tamanhos máximos dos lados do elemento (L_{Max}), são os parâmetros limitantes do processo de subdivisão do contorno.

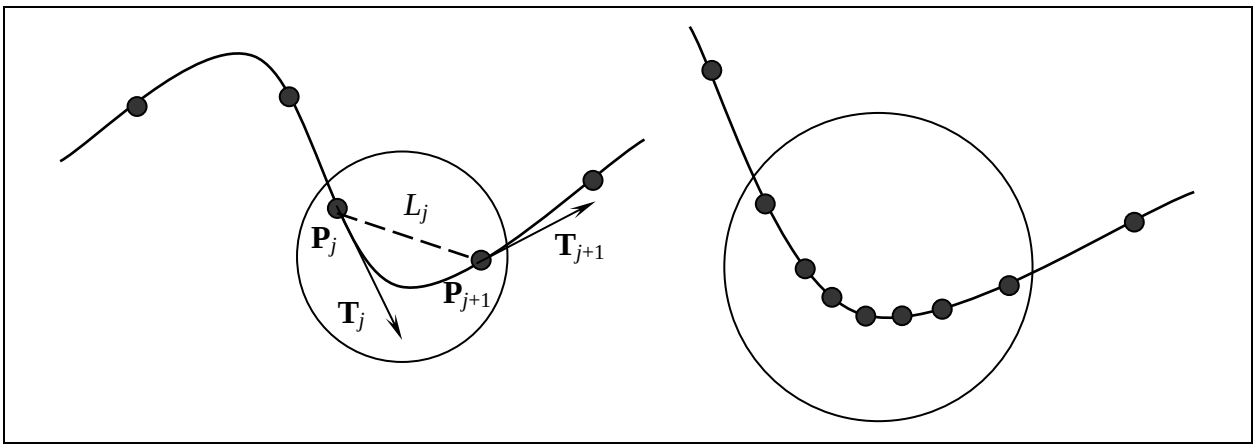


Figura 4.2 – Discretização do contorno em função da curvatura e do comprimento máximo de elemento.

A discretização do contorno é feita através de subdivisões sucessivas até que a tolerância angular seja satisfeita e que não haja nenhum segmento com comprimento maior que o tamanho máximo predefinido. O algoritmo para a divisão de uma curva C_i do contorno pode ser assim resumido:

i. Subdivide a linha em um número igual de partes n_i de tal forma que:

$$n_i = \frac{Comp(C_i)}{L_{Max}} \tag{4.1}$$

$$t_i = \frac{1}{n_i} \tag{4.2}$$

onde $Comp(C_i)$ é o comprimento real da linha C_i e t_i é o espaçamento paramétrico inicial entre os pontos da linha.

ii. Para cada segmento de C_i, Determinam-se os ângulos ($\hat{A}_j$) entre os vetores tangentes das extremidades e o comprimento dos mesmos (L_j):

$$Cos\,\hat{A}_j = \frac{\mathbf{T}_j}{\left\|\mathbf{T}_j\right\|} \cdot \frac{\mathbf{T}_{j+1}}{\left\|\mathbf{T}_{j+1}\right\|} \tag{4.3}$$

$$L_j = \left\| \mathbf{P}_{j+1} - \mathbf{P}_j \right\| \tag{4.4}$$

iii. Se em todos os segmentos $A_j < A_{Max}$ e $L_j < L_{Max}$, a subdivisão está completa, caso contrário, vai para o passo *iv*.

iv. Cada segmento onde a condição *iii* não foi satisfeita é subdividido de tal forma que:

$$t_j^{k+1} = t_j^k / 2 \tag{4.5}$$

onde t_j^k é o comprimento paramétrico do segmento *j* antes da subdivisão e t_j^{k+1} é o comprimento paramétrico dos dois segmentos resultantes depois da subdivisão.

v. Retorna ao passo *ii.*

O resultado do algoritmo de subdivisão é um conjunto de parâmetros *t* no espaço paramétrico das linhas, os quais correspondem a pontos no espaço paramétrico da superfície (*u*, *v*). A Figura 4.2 mostra o esquema de subdivisão do contorno e a Figura 4.3 apresenta a discretização do contorno para uma superfície Coons que constitui o Exemplo 4.1.

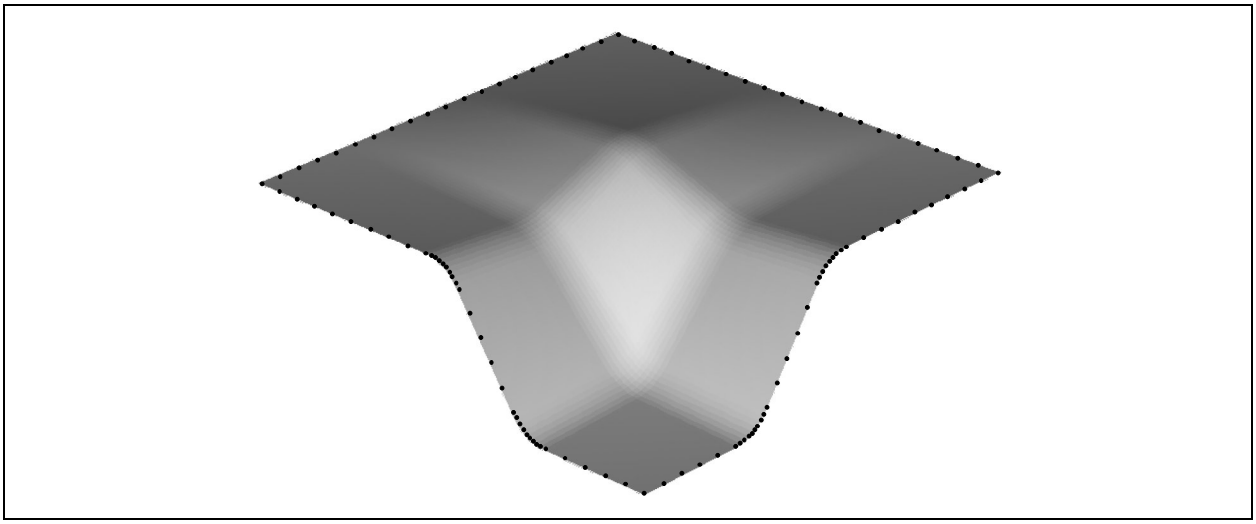


Figura 4.3 – Exemplo 4.1 (superfície Coons) com o contorno discretizado através de subdivisões sucessivas em função da curvatura e do comprimento máximo.

4.1.2. Geração da Malha de Fundo

Um algoritmo para geração de malhas não estruturadas deve conter mecanismos que garantam a qualidade da malha e, ao mesmo tempo, levem em conta propriedades geométricas do domínio, controlando o tamanho de elementos. Alguns algoritmos utilizam funções matemáticas que definem a distribuição de densidade de elementos ao longo do domínio [Lau e

Lo, 1996]. No entanto, em domínios complexos torna-se inviável definir funções explícitas para a densidade nodal. Neste caso, é mais prático o uso de malhas de fundo, que definem a densidade nodal diretamente sobre o domínio. As malhas de fundo podem ser definidas pelo usuário [Lee e Hobbs, 1999] ou geradas automaticamente. Os trabalhos de Owen e Saigal, 1999, e Canann et al, 1997, utilizam malhas de fundo baseadas numa triangulação tipo Delaunay. O trabalho de Miranda e Martha, 2002, utiliza uma malha de fundo tipo *quadtree* gerada com base na curvatura da superfície e obtém ótimos resultados. Este tipo de malha é muito eficiente para controlar o tamanho dos elementos em função da geometria do superfície. Esta razão, aliada ao fato de um algoritmo de subdivisão deste tipo já estar implementado para o algoritmo de interseção de superfícies, foram determinantes na escolha deste tipo malha de fundo para a implementação neste trabalho.

A geração da *quadtree* é feita com base em alguns critérios sugeridos por Miranda e Martha, 2002, e em critérios definidos com a experiência da implementação. A malha de fundo é gerada no espaço paramétrico, mas utiliza características geométricas da superfície real para controlar o processo. A profundidade inicial da *quadtree* é determinada em função das dimensões reais da superfície e de um tamanho máximo de elementos. Como o espaço paramétrico é sempre 1x1, há a necessidade de ajustar a grade da quadtree para que esta reflita as proporções reais da superfície. A grade inicial da quadtree (n_uxn_v) é determinada pelas expressões:

$$n_u = \frac{Comp(C_u)}{L_{Max}} \tag{4.6}$$

$$n_v = \frac{Comp(C_v)}{L_{Max}} \tag{4.7}$$

onde C_u e C_v são linhas sobre a superfície tal que $C_u = \mathbf{S}(u,0.5)$ e $C_v = \mathbf{S}(0.5,v)$, $Comp(..)$ refere-se aos comprimentos reais destas linhas e L_{Max} é o tamanho máximo de elemento para a malha de Elementos Finitos. Com isto, o primeiro nível de subdivisão contém células com $H \cong L_{Max}$, onde H é a dimensão real do lado de uma célula. Assim, a grade inicial da *quadtree* não é, necessariamente, formada por células quadradas no espaço paramétrico, pois depende das dimensões da superfície na espaço real nas direções *u* e *v*. Se uma superfície é um retângulo de dimensões reais 10x5 e o tamanho máximo dos elementos da malha de elementos finitos é 0.5, a grade inicial será 20x10.

Em seguida, a malha de fundo é refinada para satisfazer determinados critérios que têm por objetivo garantir a qualidade da malha de Elementos Finitos. Os seguintes critérios devem ser garantidos para a geração da *quadtree*:

i. Cada célula da *quadtree* pode conter, no máximo, um nó de contorno. Assim, o nível de subdivisão é controlado, localmente, pelos comprimentos dos segmentos do contorno. As células que contêm mais de um nó, são subdivididas até que esta condição seja atendida.

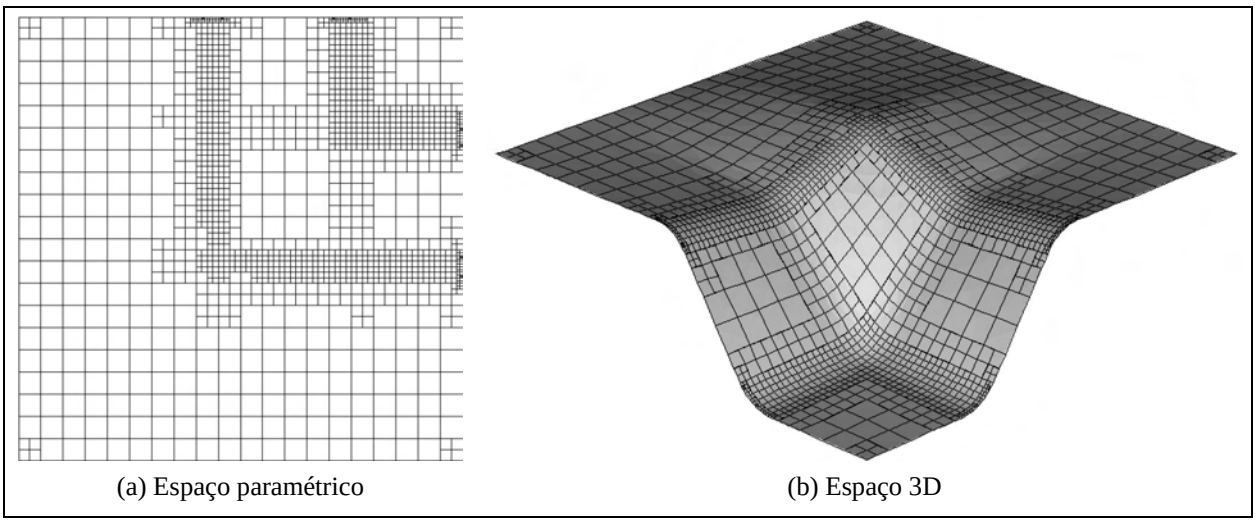
ii. A *quadtree* é refinada até que a curvatura de cada célula seja menor que uma dada tolerância.

iii. As células internas devem ter nível de subdivisão maior ou igual às células externas (que contêm o contorno).

iv. Deve haver, no máximo, um nível de subdivisão de diferença entre duas células adjacentes.

v. Cada célula deve possuir tamanho (*H*), medido no espaço 3D, menor ou igual ao tamanho máximo de elemento requerido.

vi. A subdivisão de cada célula é feita sempre utilizando-se coordenadas paramétricas médias da célula, gerando quatro células filhas com as mesmas dimensões paramétricas.



(a) Espaço paramétrico (b) Espaço 3D

Figura 4.4 – Malha de fundo para o Exemplo 4.1.

Um vez que o processo de subdivisão é similar ao utilizado no algoritmo de interseção (Cap. 3), para medir o nível curvatura da superfície emprega-se o mesmo conceito de planura ($\hat{P}$) já definido pela Eq. 3.1, que é o co-seno do máximo ângulo entre as normais dos vértices e a normal do centro da célula. Na prática, este cálculo é feito através do produto escalar entre os vetores normais normalizados. Utiliza-se uma tolerância angular (A_{Max}), de tal forma que as células que não satisfazem a condição $A_i \leq A_{Max}$ são subdivididas.

O algoritmo para a geração da *quadtree* é iterativo e termina quando todos critérios mencionados acima são satisfeitos. A Figura 4.4 apresenta um exemplo de malha de fundo correspondente à superfície Coons do Exemplo 4.1. A Figura 4.4a mostra a malha de fundo no espaço paramétrico e a Figura 4.4b mostra a malha de fundo no espaço 3D. Neste exemplo, é possível observar claramente a concentração da malha de fundo nos trechos de maior curvatura da superfície. Verifica-se, também, que não existe diferença maior que um nível entre duas células adjacentes.

4.1.3. Montagem do Front Inicial

O Front é um conjunto de segmentos conectados pelas extremidades, definidas pela discretização do contorno, que serve como base para a criação de todos os elementos da malha e é atualizado continuamente, durante o processo de geração, à medida que avança sobre o domínio da superfície. A sua configuração inicial determina, em parte, como o domínio será preenchido pela malha. Portanto, é importante que a montagem do Front inicial atenda aos critérios de topológicos requeridos.

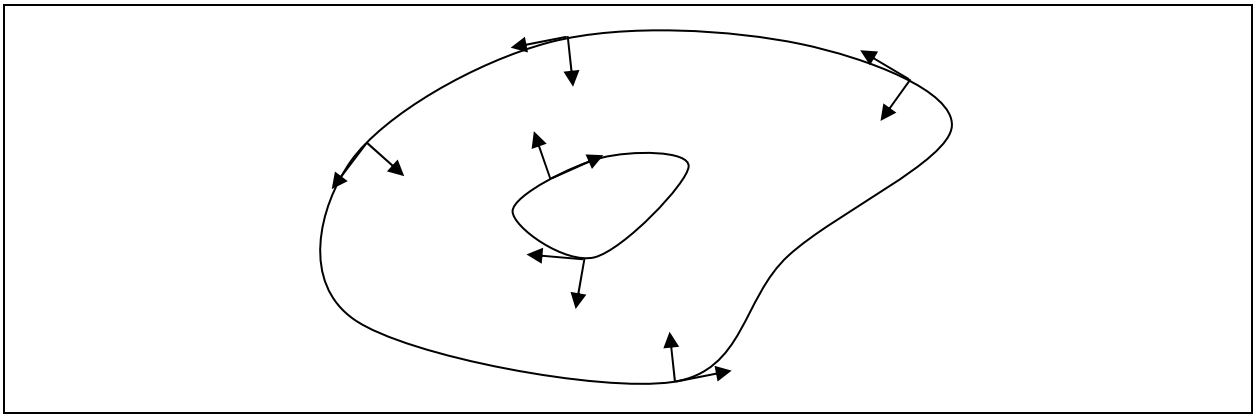


Figura 4.5 – Orientação vetorial das linhas conforme o sentido desejado de propagação.

Os segmentos do Front inicial são criados a partir dos pontos obtidos na discretização do contorno e seguindo uma ordenação que irá depender do tipo de linha de domínio. As linhas do domínio podem estar orientadas no espaço paramétrico de três formas conforme a sua classificação. As linhas de contorno ficam orientadas no sentido anti-horário. As linhas de furo ficam orientadas segundo o sentido horário. As linhas internas apresentam orientações duplas, nos dois sentidos. Os pontos gerados sobre cada linha são conectados em ordem seguindo estas orientações, o que irá determinar a direção e o sentido de propagação da malha no algoritmo Frontal.

A direção e o sentido de propagação da malha em relação a uma linha do domínio no espaço paramétrico podem ser definidos pelo vetor $\mathbf{Y}_m$ definido da seguinte forma:

$$\mathbf{Y}_M = \mathbf{Z}_P \times \mathbf{X}_C \tag{4.9}$$

onde $\mathbf{X}_C$ é o vetor tangente à curva, no espaço paramétrico, com sentido coincidente à orientação da mesma e $\mathbf{Z}_P$ corresponde à direção *z* no espaço paramétrico, considerando-se *u* e *v* como as direções *x* e *y* locais. A Figura 4.5 mostra a direção de propagação da malha em um contorno genérico.

4.1.5. A Geração da Malha de Elementos Finitos

O algoritmo de geração de malha parte de um Front inicial e se propaga até preencher todo o sub-domínio. A partir do Front, são gerados Elementos com novos Nós ou com Nós do próprio Front, de tal forma que os segmentos do Front formem lados dos Elementos que serão criados. Toda vez que um Elemento novo é gerado, o Front é atualizado e o processo continua até que Front esteja vazio.

O algoritmo inicia pelo segmento de menor comprimento no Front [Lee e Hobbs, 1999], em seguida uma série de verificações são realizadas para determinar como será criado um Elemento a partir do segmento escolhido, podendo-se criar um novo Nó ou utilizar um Nó existente no Front ou ambos. Qualquer que seja a situação, para que um novo Elemento seja criado a partir de um segmento $\overline{\mathbf{AB}}$ e um Nó **C**, as seguintes condições devem ser satisfeitas:

- O triângulo **ABC** não deve conter em seu interior nenhum Nó da malha;
- O ponto **C** não pode estar contido no interior de nenhum Elemento da malha;
- A distância de **C** aos demais segmentos do Front deve ser superior a um determinado valor considerado mínimo;
- $\overline{\mathbf{AC}}$ e $\overline{\mathbf{BC}}$ não devem, em hipótese alguma, possuir interseção com outro segmento do Front;
- A área do triângulo **ABC** deve ser positiva.

Se todas estas condições são satisfeitas considera-se *Verifica*(**ABC**)=*Verdadeiro*, em caso contrário, *Verifica*(**ABC**)=*Falso*. Somente um triângulo com *Verifica*(**ABC**)=*Verdadeiro* pode ser aceito para integrar à malha.

Um processo de geração de malha também deve verificar a qualidade dos elementos gerados. Um maneira simples de avaliar esta qualidade é o conceito de qualidade α introduzido por Lo, 1989, e também utilizado por Lee e Hobbs, 1999.

A expressão para calcular α é dada por:

$$\alpha(\mathbf{A},\mathbf{B},\mathbf{C}) = 2\sqrt{3}\,\frac{\|\mathbf{AB}\times\mathbf{AC}\|}{\|\mathbf{AB}\|^2 + \|\mathbf{BC}\|^2 + \|\mathbf{AC}\|^2} \tag{4.10}$$

onde **AB**, **AC** e **BC** são os vetores definidos pelos vértices **A**, **B** e **C** de um determinado triângulo. Esta métrica relaciona a área do triângulo com a soma dos quadrados de seus lados. O valor de α é 1 para triângulos equiláteros e 0 quando os vértices estão alinhados. Todos os outros tipos de triângulos apresentam valores de α menores que 1. Considera-se de boa qualidade uma malha com α médio maior ou igual a 0,90 [Lee e Hobbs, 1999]. A Figura 4.6 apenas ilustra a variação de α para diferentes relações *H/L* para triângulos isósceles. No entanto, não existe uma correlação geral entre α e *H/L*, podendo haver triângulos distorcidos com α baixo e *H/L* próximo de $\sqrt{3}/2$.

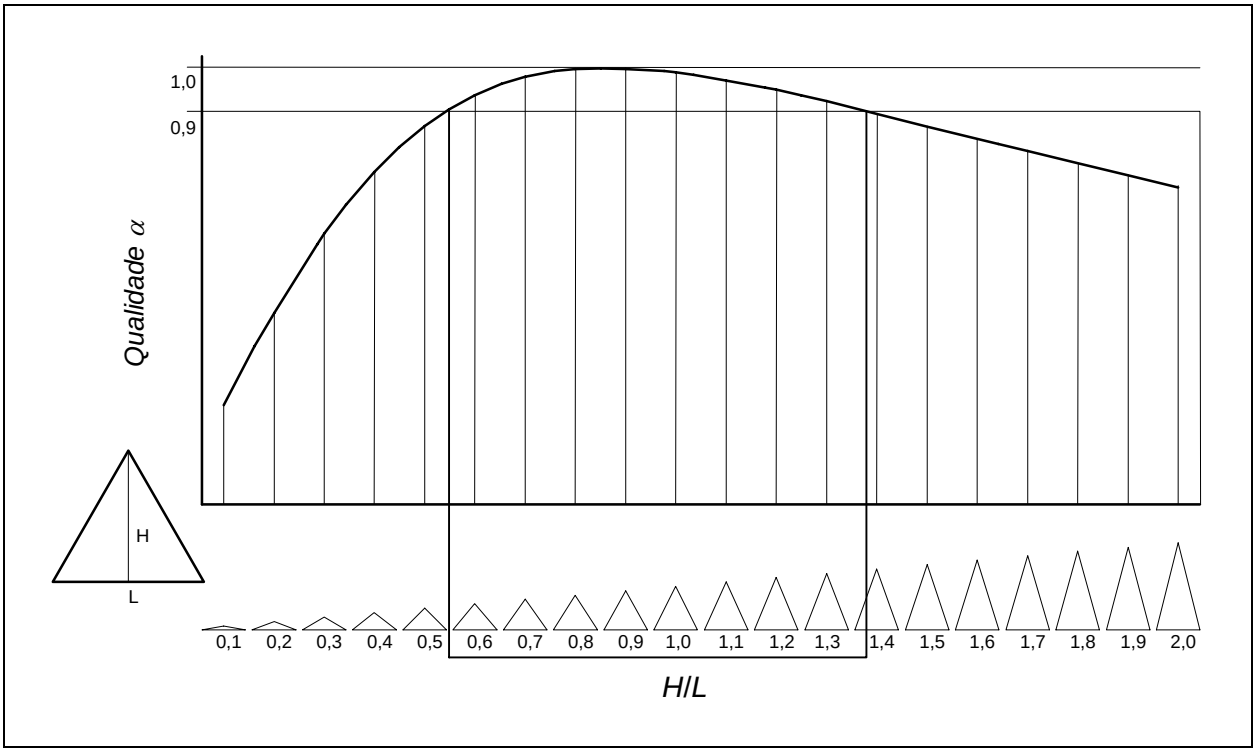


Figura 4.6 – Valores de qualidade α para diferentes relações *H*/*L* de triângulos.

A qualidade geral das malhas geradas será avaliada pelos seguintes parâmetros relacionados a α:

- $\overline{\alpha}$ Fator de qualidade α médio, que deve ser superior a 0,90;
- α_{90} Percentual de elementos com $\alpha \geq 0{,}90$;
- α_{Min} Menor valor de α na malha.

A seguir são detalhados os principais procedimentos do algoritmo durante o processo de avanço do Front:

i. Toma-se o menor segmento do Front em comprimento real que será denominado $\overline{\mathbf{PQ}}$;

ii. Calcula-se o ponto médio ($\mathbf{M}^p$) do segmento no espaço paramétrico:

$$\mathbf{M}^p = \frac{\mathbf{P}^p + \mathbf{Q}^p}{2} \tag{4.11}$$

iii. Determina-se *H*, a dimensão da célula da quadtree que contém $\mathbf{M}^p$.

iv. Calcula α_E e α_D, ângulos com os segmentos à esquerda e à direita de $\overline{\mathbf{PQ}}$, respectivamente;

v. Se $\alpha_E < 80^o$ e/ou $\alpha_D < 80^o$, vai para *vi*, caso contrário, vai para *vii*.

vi. Toma-se o vértice oposto (**R**) do segmento com menor ângulo e os dois segmentos incidentes darão origem a 1 ou 2 triângulos:

- Caso a distância entre extremidade esquerda do segmento da esquerda e a extremidade direita do segmento da direita seja menor ou igual *H*:
  - Se *Verifica*(**PQR**) = *Verdadeiro*, cria-se um triângulo **PQR**, sem a criação de um novo vértice (Fig. 4.7). Vai para *xvi*.

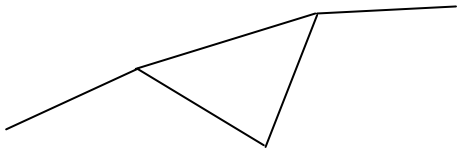


Figura 4.7 – Cria novo elemento sem criar novo nó.

  - Se *Verifica* (**PQR**) = *Falso*, vai para *vii*.
- Caso a distância seja maior que *H*, cria-se um novo nó (**S**) na posição central entre as duas extremidades e 2 novos triângulos podem ser criados:
  - Se *Verifica*(**PSR**) = *Verdadeiro* e/ou *Verifica*(**PQS**) = *Verdadeiro*, criam-se os novos triângulos (Fig. 4.8) e vai para *xvi*.

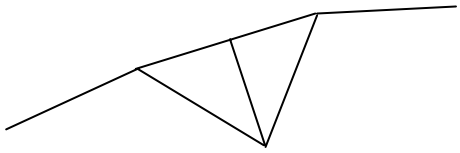


Figura 4.8 – Cria 2 novos elemento com um novo nó.

  - Se *Verifica*(**PSR**) = *Falso* e *Verifica*(**PQS**) = *Falso*, vai para *vii*.

vii. Um novo nó (**R**) será criado. Calcula-se uma aproximação inicial de $\mathbf{R}^p$ considerando-se um triângulo equilátero no espaço paramétrico:

$$\mathbf{T}^p = \mathbf{P}^p - \mathbf{Q}^p \tag{4.12}$$

$$\mathbf{V}^p = \frac{(0,0,1) \times \mathbf{T}^p}{\left\| (0,0,1) \times \mathbf{T}^p \right\|} \tag{4.13}$$

$$\mathbf{R}_0^p = \mathbf{M}^p + \left\| \mathbf{T}^p \right\| \cdot \frac{\sqrt{3}}{2} \cdot \mathbf{V}^p \tag{4.14}$$

viii. Determina-se um Sistema de Referência Auxiliar cujo plano xy_L é $\mathbf{PQR}_0$ no espaço real. A direção x_L deste SRA é dada pelo segmento **PQ**. O SRA é utilizado para facilitar a construção de um triângulo de altura *H*, $\mathbf{PQR}_1$, neste mesmo plano:

$$\mathbf{T} = \mathbf{P} - \mathbf{Q} \tag{4.15}$$

$$\mathbf{M} = \frac{(\mathbf{P} + \mathbf{Q})}{2} \tag{4.16}$$

$$\mathbf{V} = \frac{(0,0,1) \times \mathbf{T}}{\left\| (0,0,1) \times \mathbf{T} \right\|} \tag{4.17}$$

$$\mathbf{R}_1 = \mathbf{M} + H \cdot \mathbf{V} \tag{4.18}$$

ix. O ponto $\mathbf{R}_1$ deve ser projetado sobre a superfície. Aqui, utiliza-se o mesmo algoritmo apresentado no Capítulo 3 (ver Seção 3.1.5.2) para realizar o mapeamento paramétrico dos pontos de interseção. Assim, obtém-se $\mathbf{R}_2^p$, no espaço paramétrico, e $\mathbf{R}_2$, sua representação no espaço real (ver Fig. 4.9).

x. Se a distância entre $\mathbf{R}_2$ e **M** for menor ou igual a *H*, $\mathbf{R}_2^p$ é o ponto procurado e o processo continua no passo *xii*.

xi. Se a distância é maior que *H*, a equação 4.18 é reaplicada com um valor de $H_i = 0{,}9 \cdot H_{i-1}$ até que a distância entre $\mathbf{R}_i$ e **M** seja menor ou igual a *H*.

xii. Define-se um círculo, no espaço paramétrico, com centro em $\mathbf{R}^P$ e raio H^P;

xiii. Caso existam nós do Front no interior do círculo tais que *Verifica* ($\mathbf{PQS}_i$) = *Verdadeiro*, toma-se aquele ($\mathbf{S}_i$) que resulta em um triângulo com maior qualidade. Cria-se, assim, um triângulo **PQS** e vai para *xvi*.

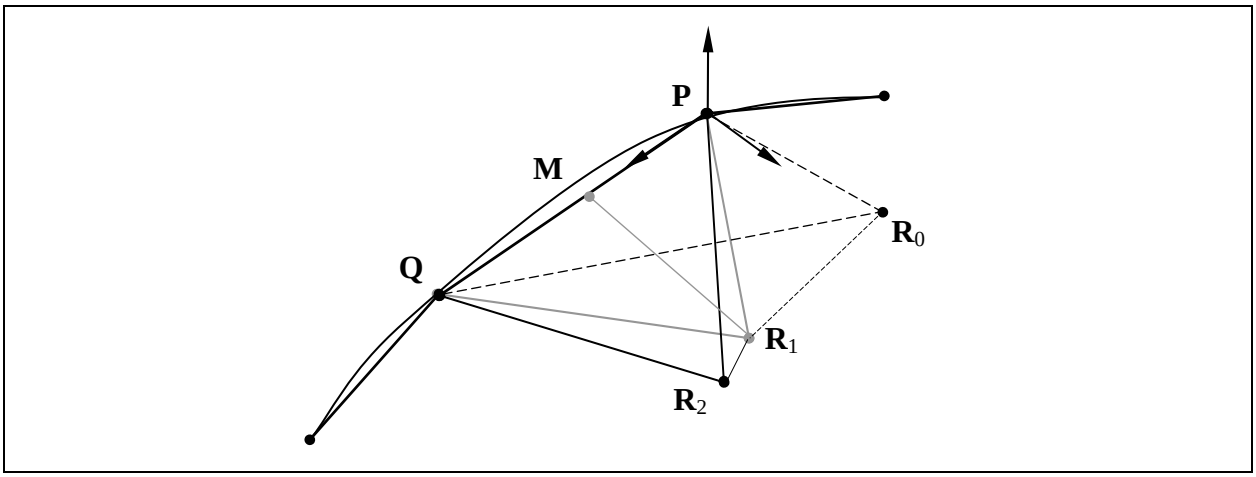


Figura 4.9 – Criação de um novo ponto a partir de um segmento PQ do Front.

xiv. Caso não existam nós do Front no interior círculo:

- Se *Verifica*(**PQR**)=*Verdadeiro*, cria o triângulo **PQR** e vai para *xvi*.
- Se *Verifica*(**PQR**)=*Falso*, vai para *xv*.

xv. Se não foi possível criar nenhum elemento com o segmento $\overline{\mathbf{PQ}}$, este é configurado como inativo até que todos os segmentos do nível atual sejam processados. Retorna ao passo *i*.

xvi. Atualiza o Front, removendo os segmentos existentes que se tornaram lados de triângulos e acrescentando os novos lados dos novos triângulos. Os novos segmentos adicionados ao Front ficam inativos até que todos os segmentos do nível anterior tenham sido processados.

xvii. Caso todos os segmentos tenham sido processados, verifica-se se há segmentos inativos restantes:

- Caso não existam segmentos restantes o processo está completo.
- Caso existam segmentos inativos restantes, todos os segmentos são remarcados como ativos e retorna-se ao passo *i*.
- Caso o processo não avance, relaxam-se as restrições da função *Verifica*(.), de tal forma que os elementos são criados função apenas da topologia, e retorna-se ao passo *i*.

A Figura 4.10 apresenta uma seqüência de imagens com a configuração do Front ao longo do processo de geração da malha para o Exemplo 4.1. Pode-se observar que o Front avança por camadas. Isto ocorre devido à restrição, introduzida no algoritmo, que torna os novos segmentos adicionados ao Front inativos até que o nível (ou geração) de segmentos atual seja totalmente processado. A partir daí, toda a nova geração de segmentos do Front é ativada e o processo recomeça. Isto garante uniformidade na propagação do Front. Se, ao contrário, todos os segmentos do Front fossem mantidos ativos desde a sua criação, a propagação se daria de forma aleatória, criando situações onde geram-se os elementos excessivamente distorcidos.

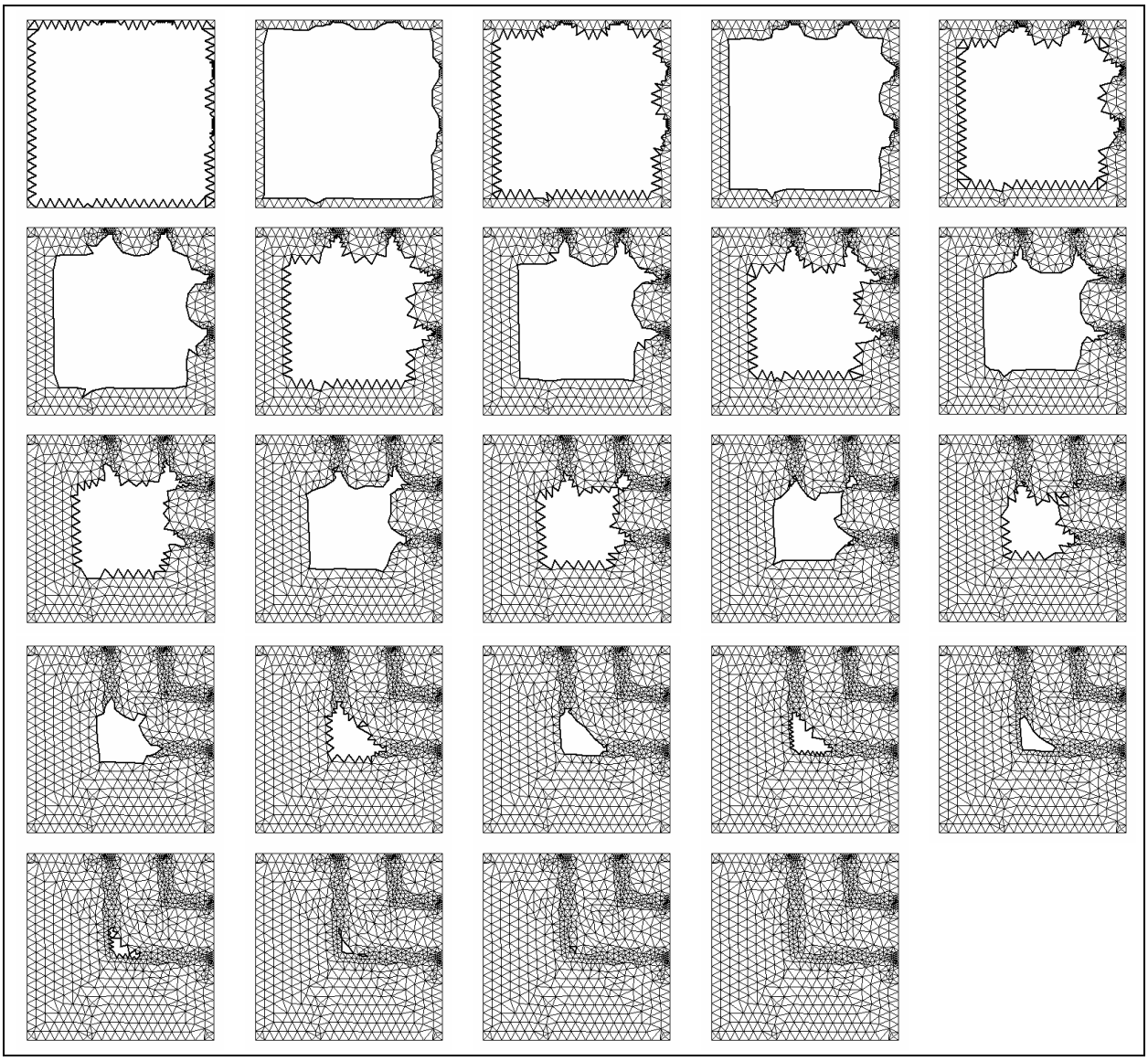
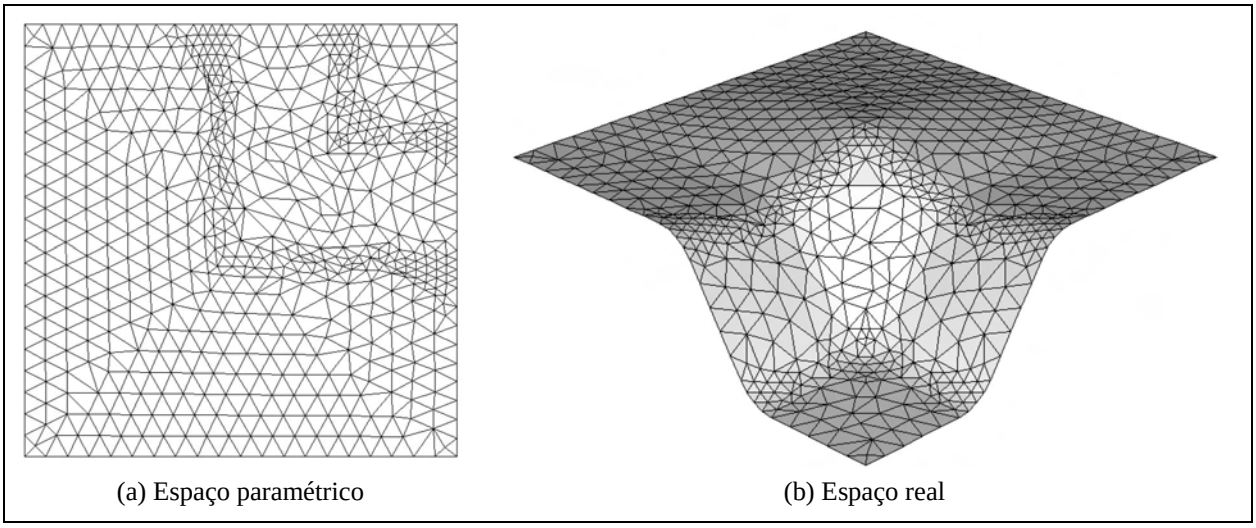


Figura 4.10 – Configuração do Front durante o processo de geração de malha.



(a) Espaço paramétrico (b) Espaço real

Figura 4.11 – Malha gerada sobre a superfície ($\overline{\alpha} = 0{,}93$).

O próximo passo é suavizar a malha para que os elementos tenham a menor distorção possível. A Figura 4.11 apresenta uma malha gerada na superfície do Exemplo 4.1 com L=0,5 e A=10º, onde é possível notar que a qualidade dos Elementos é baixa em algumas regiões.

4.1.6. Suavização da Malha

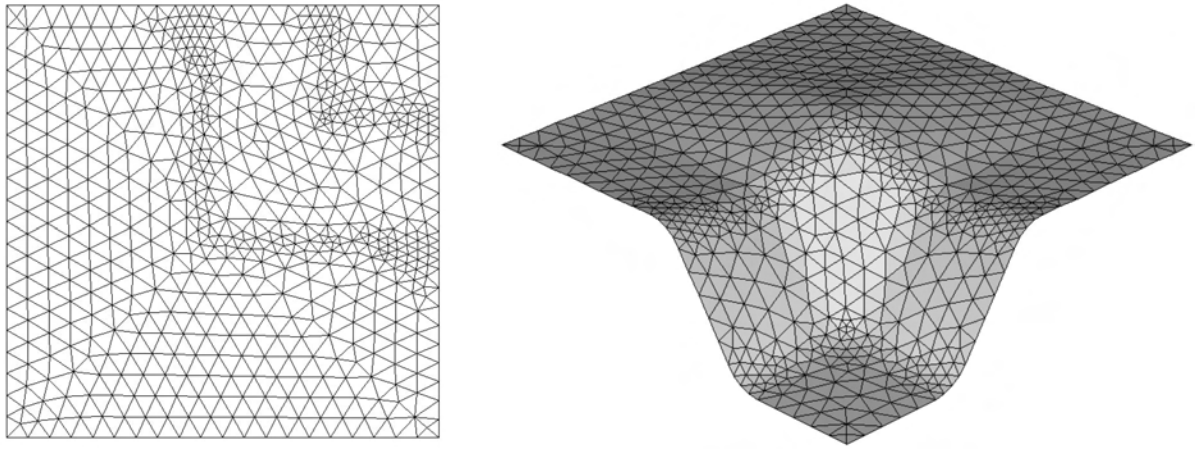
Normalmente, após uma geração de malha não estruturada, é necessário realizar algum tipo de suavização para que a malha final tenha uma qualidade adequada à aplicação a que se destina. A suavização Laplaciana é a forma mais usual para melhorar a qualidade de malhas não estruturadas, reposicionando cada nó de tal forma que este passe a ser o centróide do polígono formado pelos nós vizinhos.

Neste trabalho, optou-se por utilizar a forma modificada do operador Laplaciano que utiliza um fator que relaciona as medidas do espaço real e do espaço paramétrico [Miranda e Martha, 2002], o que evita distorções, pois o processo é aplicado no espaço paramétrico. A equação utilizada é a seguinte:

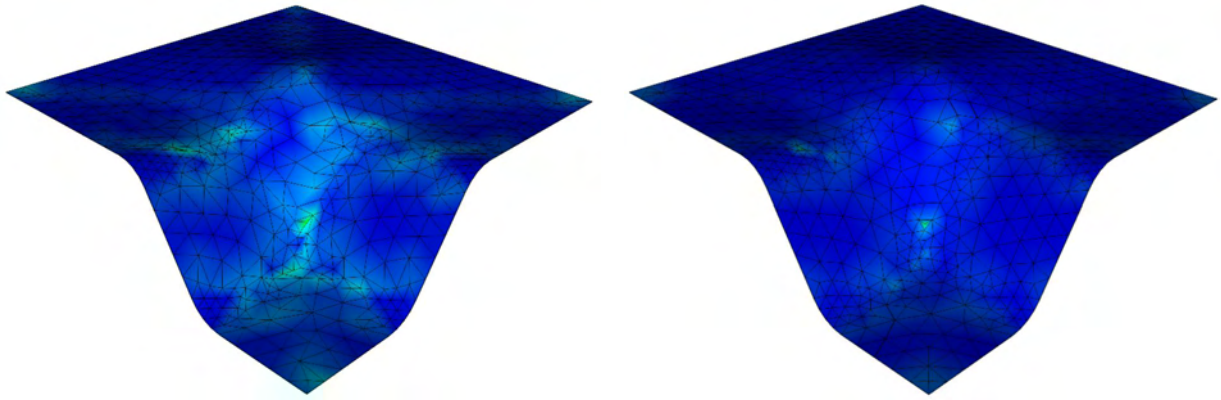
$$\mathbf{X}_0^{n+1} = \mathbf{X}_0^n + \phi \frac{\sum_{i=1}^{m} w_{i0}(\mathbf{X}_i^n + \mathbf{X}_0^n)}{\sum_{i=1}^{m} w_{i0}} \tag{4.19}$$

onde $\mathbf{X}_0$ é o vetor posição do nó 0 no qual incidem os nós i ($\mathbf{X}_i$), m é o número de nós incidentes, w_{i0} é um fator que relaciona a distância real e a distância paramétrica entre o nó i e o nó 0, e ϕ é um fator de relaxação que pode estar no intervalo [0,1]. Aplica-se a Eq 4.19 em um processo iterativo em todos os nós interiores do domínio. O número de iterações pode variar, mas, em geral, 3 são suficientes.

A Figura 4.12 apresenta a transformação da malha da Figura 4.11 após a suavização Laplaciana. Neste caso, foram aplicados 5 passos de suavização consecutivos, o que foi adotado como padrão no T-CADE. É possível observar uma sensível melhora da qualidade, tanto na malha do espaço real, como no espaço paramétrico. O $\overline{\alpha}$ da malha antes da suavização era de 0,93 e passou a 0,97 após a suavização. O α_{Min} passou de 0,28 para 0,68. As Figuras 4.10b e 4.10c comparam a distribuição do fator α na malha antes e após a suavização. O azul corresponde a valores de α próximos de 1.



(a) Malha da Figura 4.9 (L=0,5 e A=10º) após a suavização Laplaciana ($\overline{\alpha} = 0{,}97$)



(b) Sem suavização (c) Com suavização

Figura 4.12 – Malha final (a) e comparação das distribuições do fator alfa nas malhas antes e após a suavização Laplaciana (b e c).

4.2. EXEMPLOS DE GERAÇÃO DE MALHA

Nesta seção é apresentada uma série de exemplos para mostrar a eficiência do algoritmo em criar malhas de qualidade em diversas situações. Além do qualidade geral da malha, que pode ser avaliada pelo seu aspecto visual, também são utilizados os parâmetros: $\overline{\alpha}$, α_{90}, α_{Min}.

Foram geradas malhas com diferentes tamanhos máximos de elemento (L) e tolerâncias angulares (A) para verificar se a qualidade geral da malha se mantinha acima do mínimo aceitável, mesmo em situações críticas (ex. L alto e A baixo). Outro aspecto avaliado é capacidade de criar malhas de qualidade em domínios recordados, com furos e linhas internas. Os resultados estão na tabela 4.1 que apresenta o número de Nós e Elementos gerados nas malhas, os fatores de qualidade e o tempo de processamento em uma máquina Pentium IV 2.4 GHz e 512MB de RAM).

4.2.1. Malhas em Superfícies Não Recortadas

O Exemplo 4.1, uma superfície Coons, é apresentado nas Figuras 4.9 e 4.10 com três malhas diferentes obtidas a partir de três combinações de *L* e *A*. A superfície tem dimensão máxima de 10 unidades. Na primeira malha (Fig. 4.10), utilizou-se a combinação: L = 0,5 e $A = 10^{o}$. A segunda malha (Fig. 4.11a) foi obtida com uma combinação de L = 0,5 e $A = 5^{o}$. Na terceira (Fig. 4.11b), a combinação utilizada foi $L = 0{,}3$ e $A = 3^{o}$.

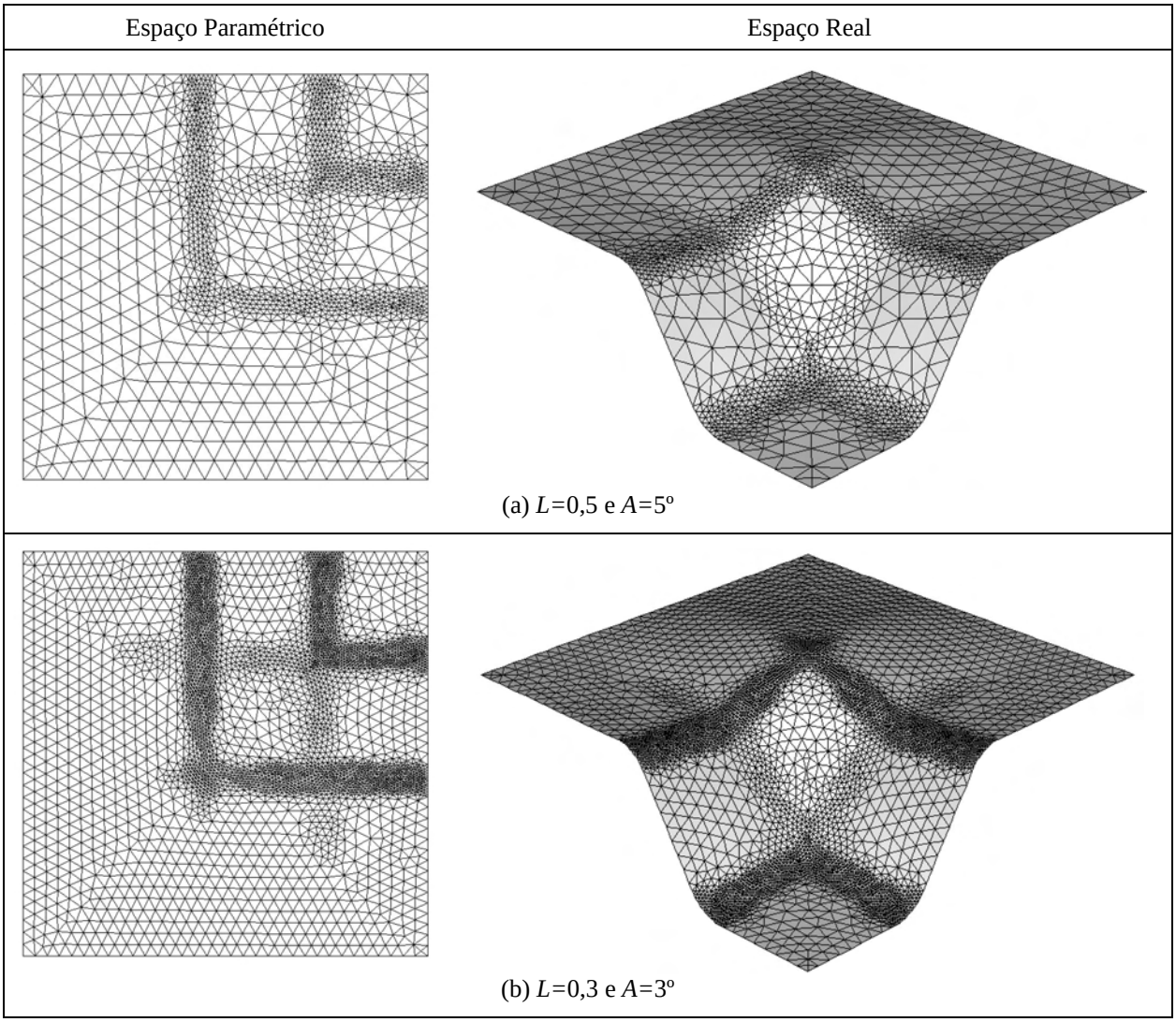


Figura 4.13 – Malhas obtidas no Exemplo 4.1 com 2 combinações de *L* e *A*.

Nos três casos, a qualidade mínima ($\overline{\alpha} \geq 0{,}9$) foi satisfeita com folga. Além disso, foram obtidos altos índices de elementos com α superior a 0,9 (α_{90}) nas três malhas, o que é um bom indicador da qualidade das malhas obtidas. Pode-se notar a maior densidade de elementos nas regiões da superfície onde há maior curvatura. Em superfícies curvas a densidade dos

elementos cresce com a diminuição da tolerância angular *A*. Este refinamento permite aproximar a malha da superfície teórica em todas as suas regiões.

Os Exemplos 4.2 a 4.6 apresentam superfícies com diferentes tipos de curvatura, onde pode-se verificar a eficiência do algoritmo em preencher o domínio com uma malha fiel à geometria do modelo. As ilustrações (Fig. 4.11 a 4.13) apresentam, para cada Exemplo, a malha de fundo no espaço paramétrico e no espaço real (linha superior) e a malha de triângulos no espaço paramétrico e no espaço real (linha superior).

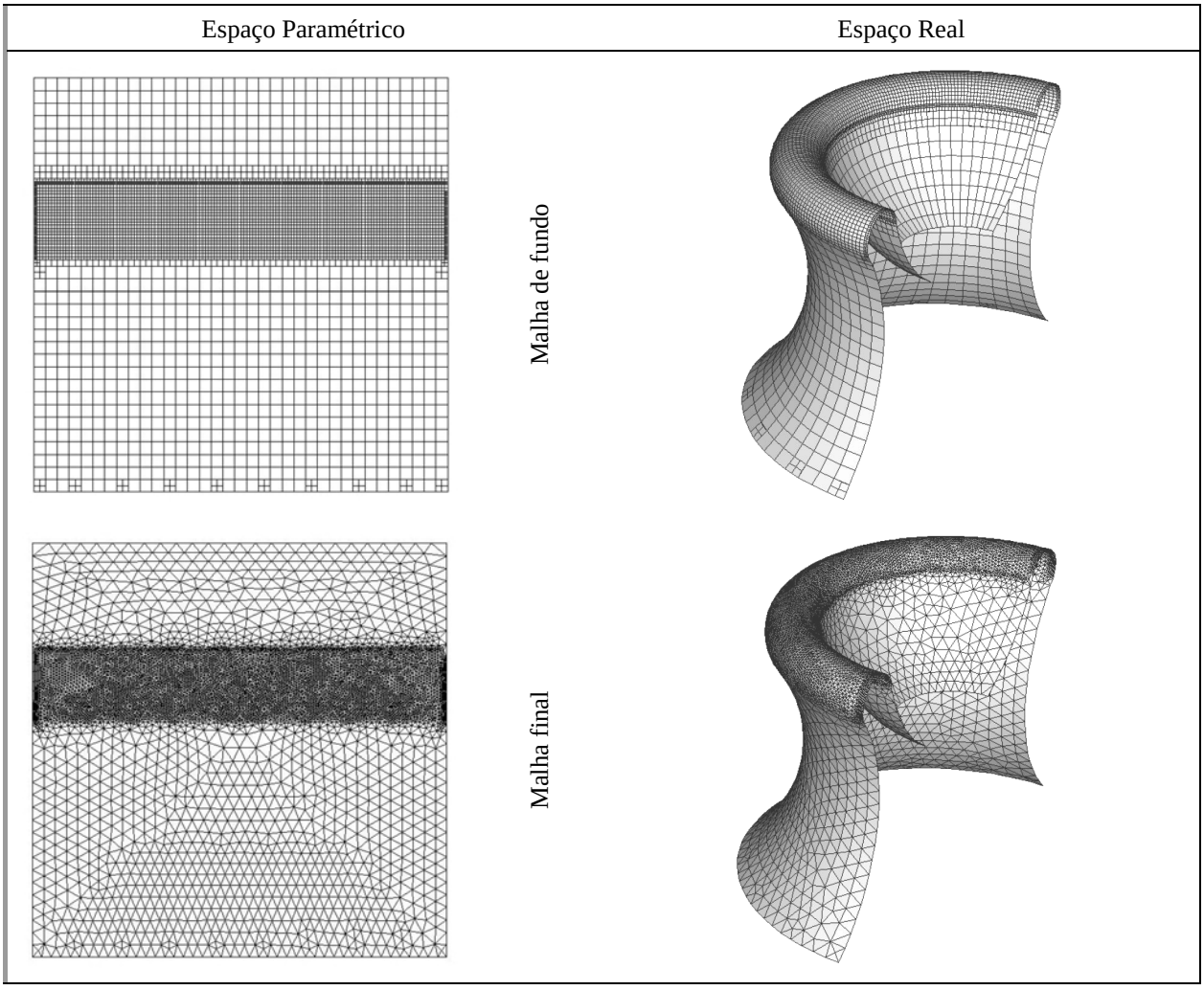


Figura 4.14 – Exemplo 4.2: Superfície de revolução.

O Exemplo 4.2 (Fig.4.14) é uma superfície de revolução (180^o) obtida a partir de uma Polilinha composta arcos de circunferência. Neste caso, as dimensões paramétricas e reais são relativamente proporcionais, portanto a malha de fundo no espaço paramétrico é dividida em células quadradas. Comparando-se a malha de fundo nos espaços paramétrico e real, observa-se uma relação direta entre o tamanho das células e curvatura local da superfície. A malha triangular final apresenta-se com a mesma densidade da malha de fundo, como era de se esperar.

A qualidade de malha gerada fica patente já no aspecto visual e, também, através do parâmetro α médio superior a 0,95 (ver Tab. 4.1).

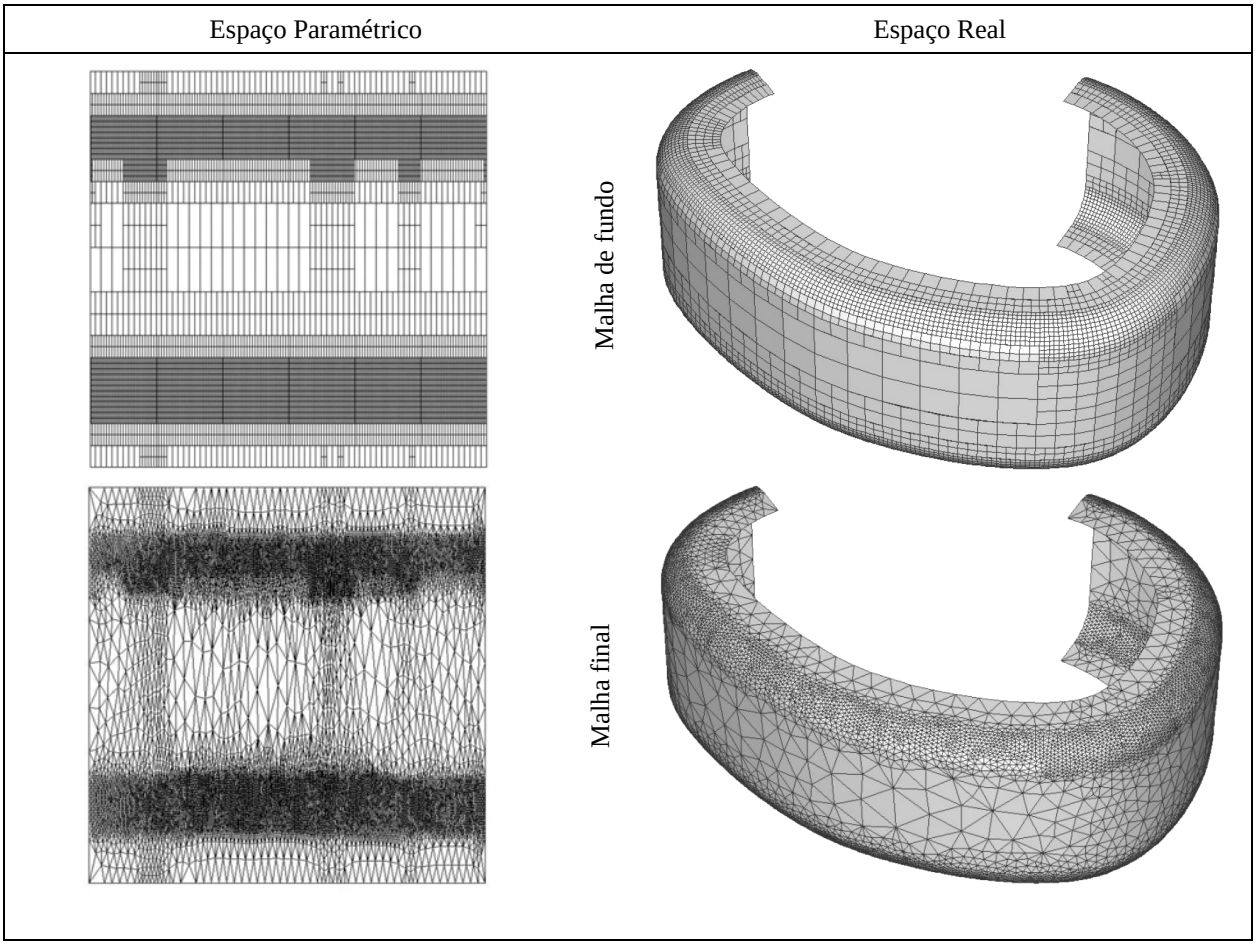


Figura 4.15 – Exemplo 4.3: Sweep

O Exemplo 4.3 (Fig. 4.15) consiste em uma superfície tipo Sweep gerada a partir de uma Polilinha em *C* e um caminho tipo Spline. Neste exemplo, com *L*=1 e *A*=5^o, a malha de fundo apresenta células retangulares no espaço paramétrico devido à distorção existente entre as dimensões da superfície no espaço real e no espaço paramétrico, o que permite reduzir as distorções das células no espaço 3D, que ficam próximas à forma quadrada. Isto é importante durante o processo Frontal, que é realizado no espaço paramétrico, mas utiliza dimensões do espaço 3D. A malha final, com α médio de 0,95, apresenta ótima qualidade, o que pode também ser verificado pelo aspecto visual.

O Exemplo 4.4 (Fig. 4.16) é uma superfície Regrada obtida a partir de uma reta e de uma Spline. Este exemplo é interessante, pois a curvatura da superfície varia ao longo de toda a superfície. Para a geração da malha, foram utilizados *L*=1 e *A*=5^o. Neste caso, a malha de fundo no espaço paramétrico apresenta células quase quadradas, devido à pequena distorção entre as

proporções dos espaços paramétrico e real. Pode-se observar a maior densidade da malha nas regiões de maior curvatura, mas com uma variação suave de tamanho de elemento. O α médio para este Exemplo foi de 0,95.

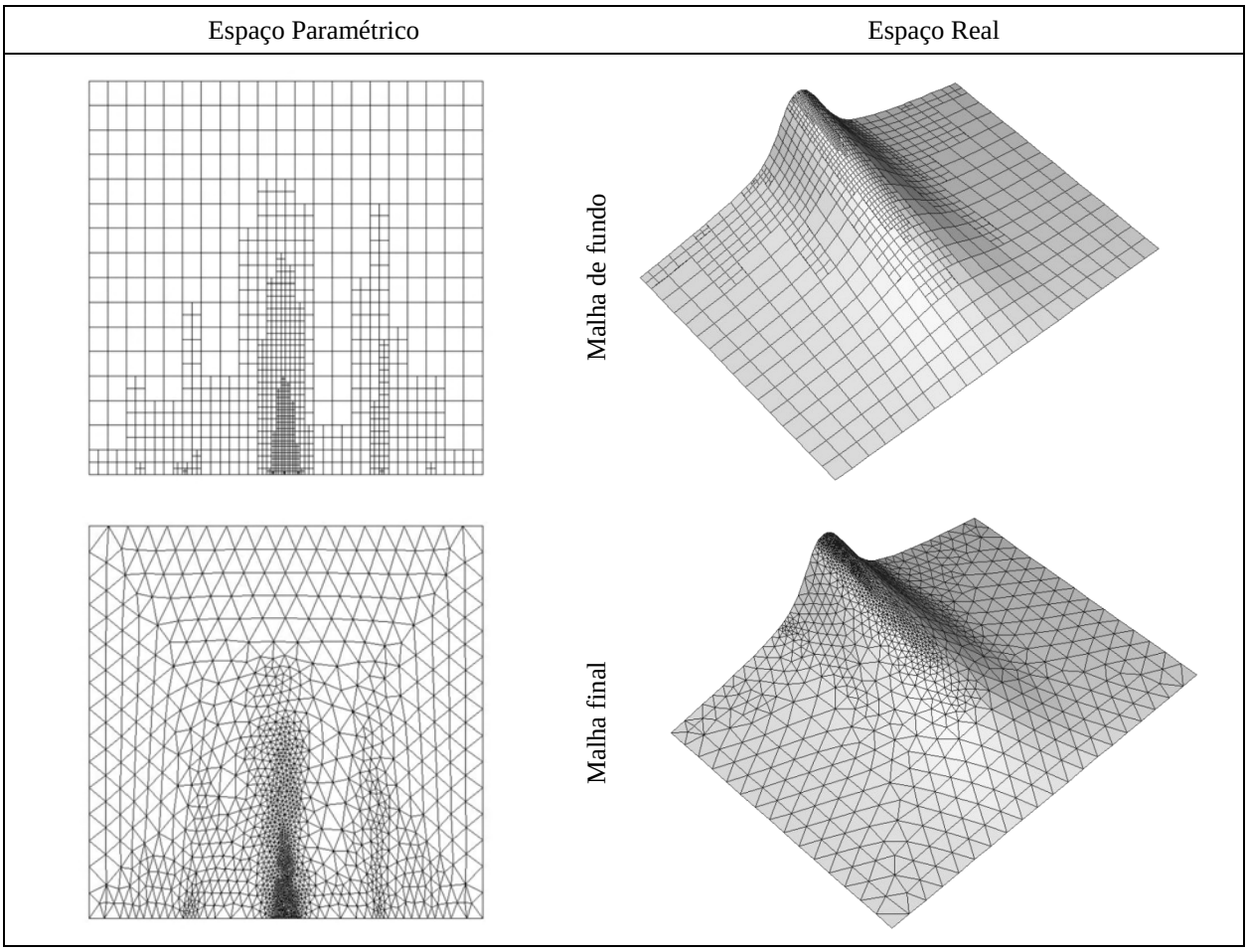
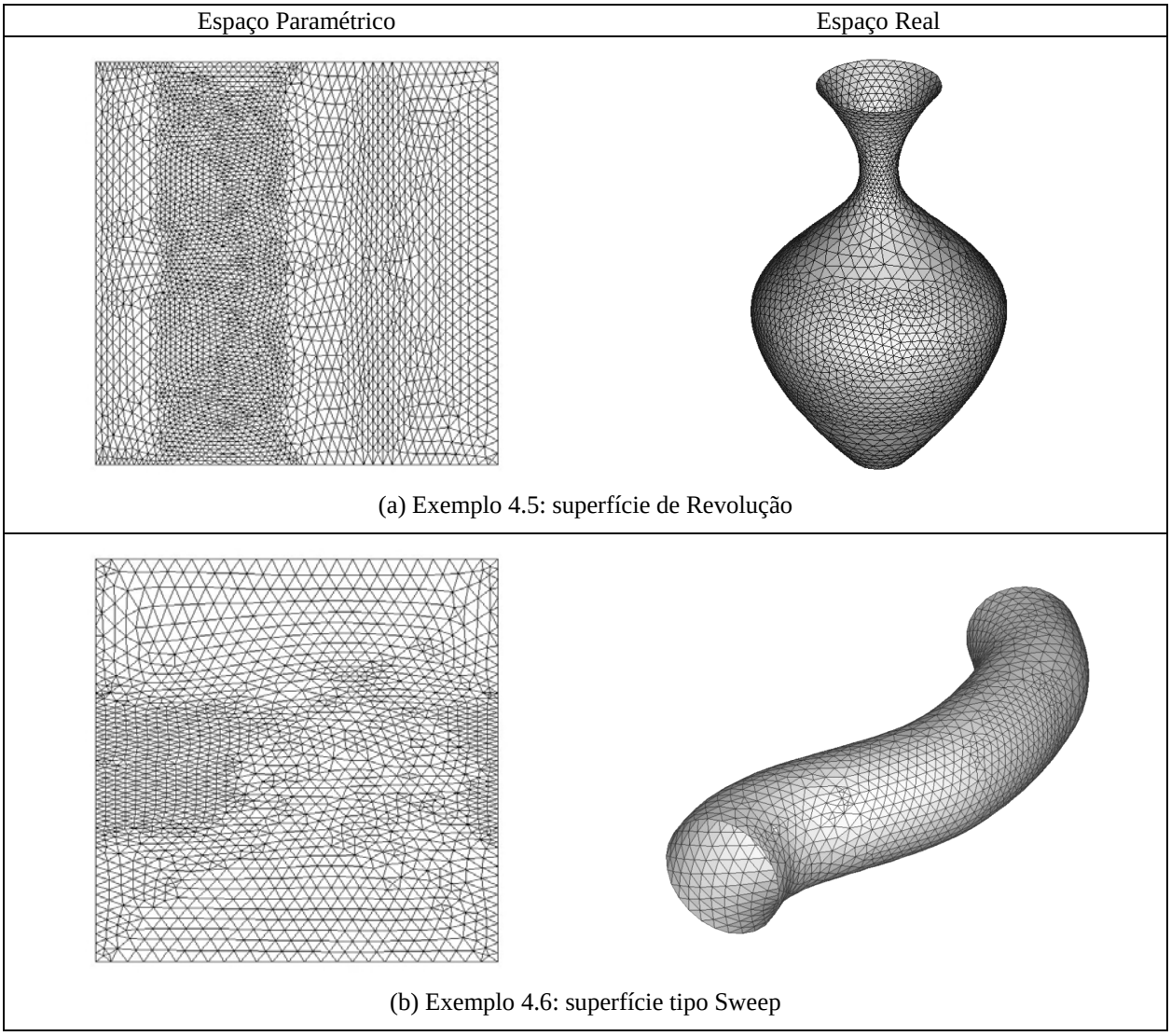


Figura 4.16 – Exemplo 4.4: Superfície Regrada.

O Exemplo 4.5 (Fig. 4.17a) é uma superfície de revolução (360^o) obtida a partir de uma *spline*. Este exemplo é interessante para verificar a eficiência do algoritmo em situações limite. Neste caso, devido à geometria do modelo, há uma variação muito grande número de Elementos ao longo do altura do modelo, mas sem, com isto, comprometer a qualidade da malha, que apresenta um excelente α médio de 0,97.

O Exemplo 4.6 (Fig. 4.17b) é constituído por uma superfície Sweep obtida a partir de uma circunferência, como perfil, e uma Spline, como caminho, formando um tubo. O algoritmo de geração de malha foi utilizado com L=0,5 e A=10^o e a malha obtida apresenta um α médio de 0,96. A forma do modelo é quase orgânica, no entanto a malha gerada possui boa qualidade com transições suaves em todas as regiões.



(a) Exemplo 4.5: superfície de Revolução

(b) Exemplo 4.6: superfície tipo Sweep

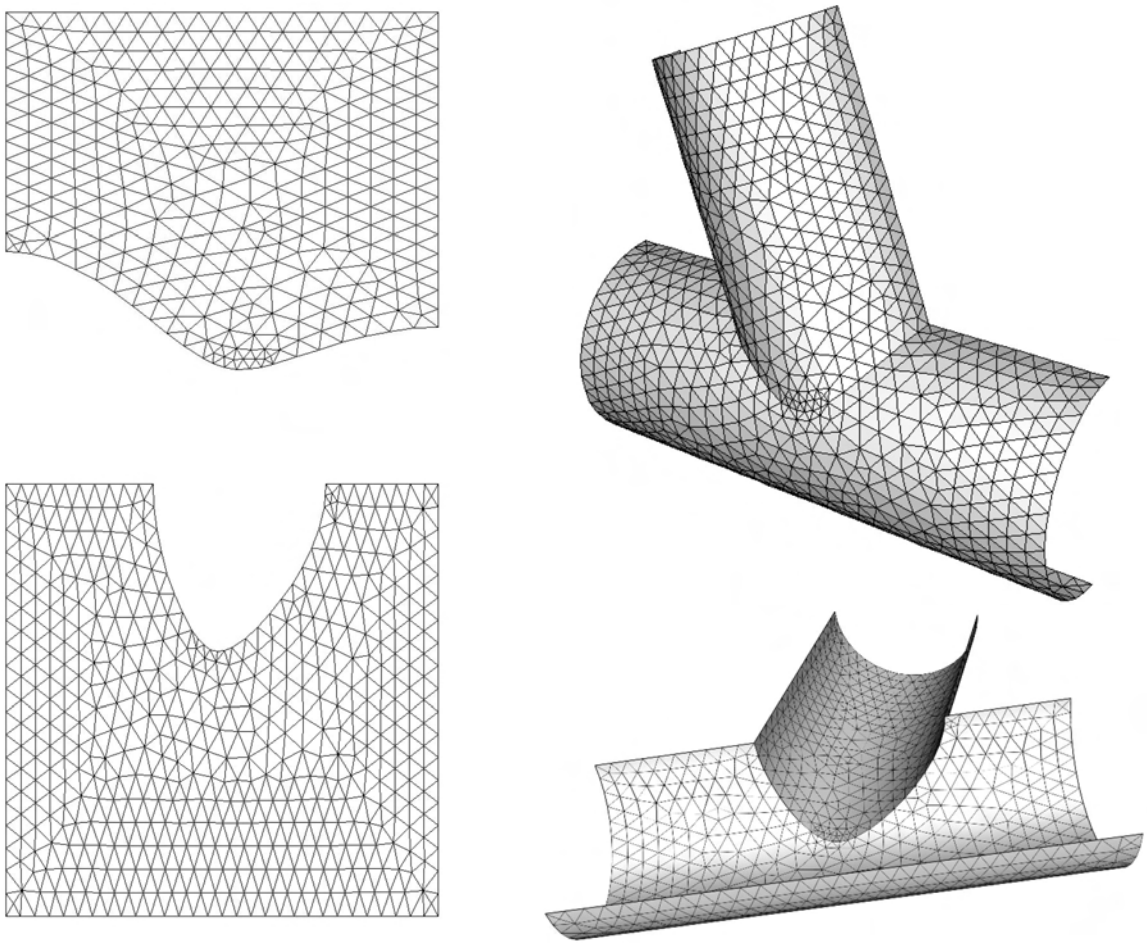
Figura 4.17 – Malhas geradas nos Exemplos 4.5 e 4.6.

4.2.2. Malhas em Superfícies Recortadas por Interseções

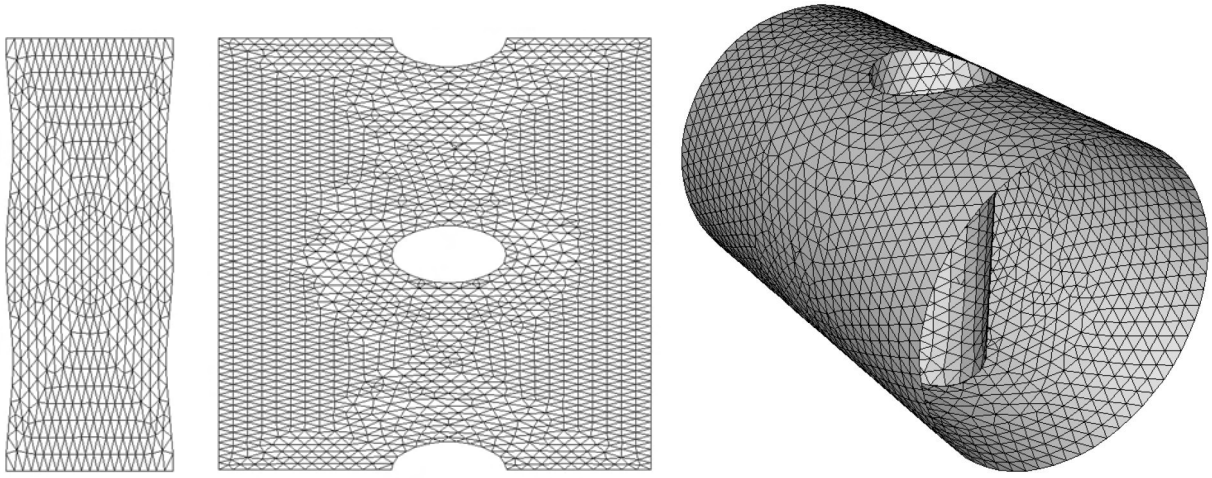
Os Exemplos 4.7 a 4.13 consistem em superfícies recortadas obtidas a partir da determinação das interseções[1] entre superfícies. São apresentadas superfícies de diversos tipos de parametrização e em posições variadas.

O Exemplo 4.7 é composto por duas superfícies cilíndricas geradas por revolução de 180^o, formando uma junção em *Y*. A Figura 4.18a apresenta a malha gerada sobre as duas superfícies no espaço paramétrico e no espaço 3D. É possível verificar, através das vistas mostradas, que as superfícies estão realmente recortadas e, na transição das superfícies, há continuidade na malha, com a coincidência dos Nós das duas superfícies. A qualidade da malha não foi afetada significativamente pelo recorte, uma vez que o α médio foi superior 0,98.

[1] As interseções foram todas calculadas pelo algoritmo descrito no capítulo anterior.



(a) Exemplo 4.7: 2 superfícies de Revolução recortadas pela linha de interseção

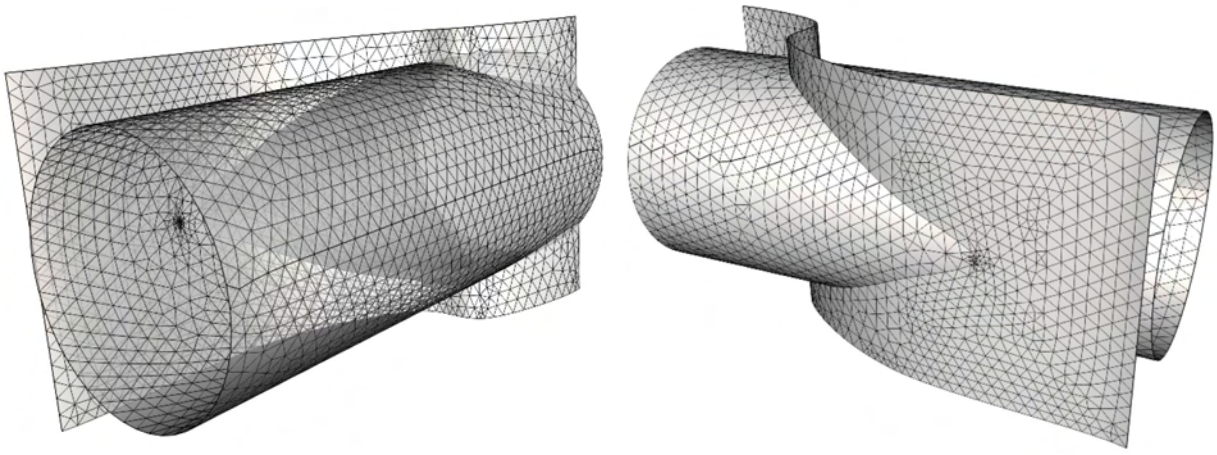


(b) Exemplo 4.8: 2 cilindros de revolução

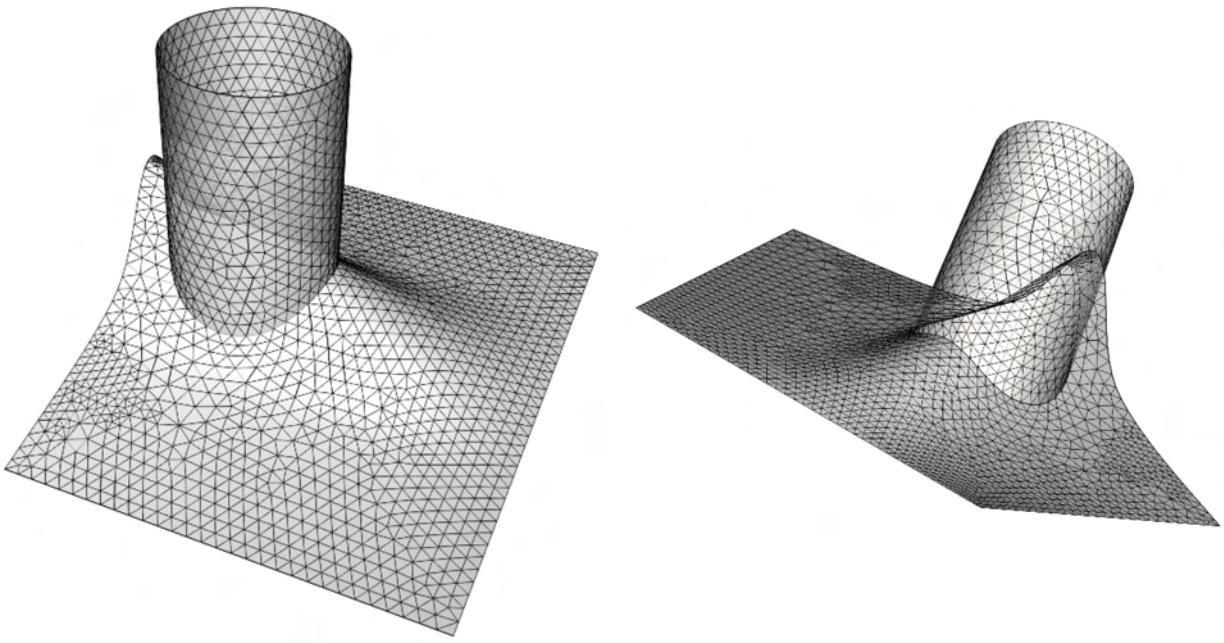
Figura 4.18 – Exemplos 4.7 (a) e 4.8 (b).

O Exemplo 4.8 (Fig. 4.18b) é formado por dois cilindros perpendiculares. O cilindro de diâmetro menor fura o cilindro maior. No cilindro menor, a malha é gerada somente na porção que está interna ao cilindro maior. O Exemplo 4.9 (Fig. 4.19a) é composto pelas mesmas

superfícies do Exemplo 3.1 do Capítulo 3 e contém uma superfície cilíndrica de Revolução e uma Sweep. A malha foi gerada em todo o cilindro, contudo foram gerados nós sobre a linha de interseção entre as superfícies. Na superfícies Sweep foi removida a porção interna ao cilindro.



(a) Exemplo 4.9: Malha sobre recorte do Exemplo 3.1

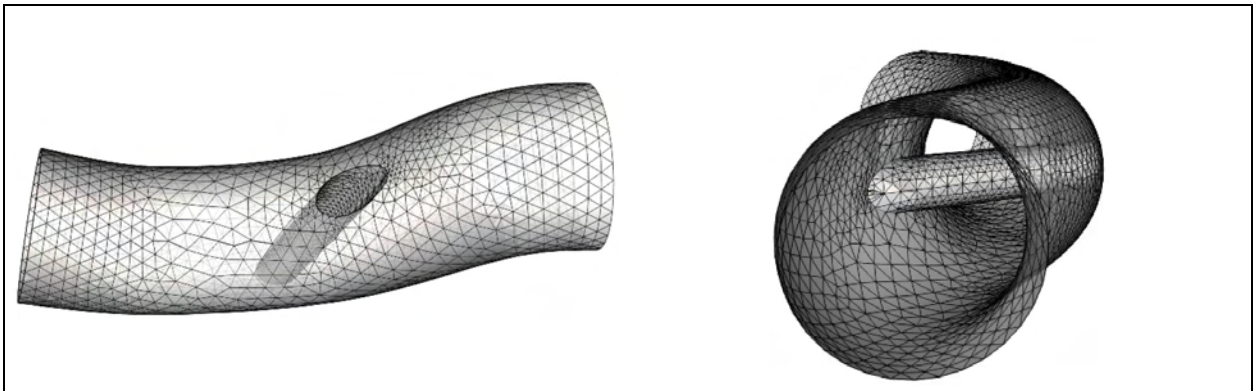


(b) Exemplo 4.10: Regrada e Revolução

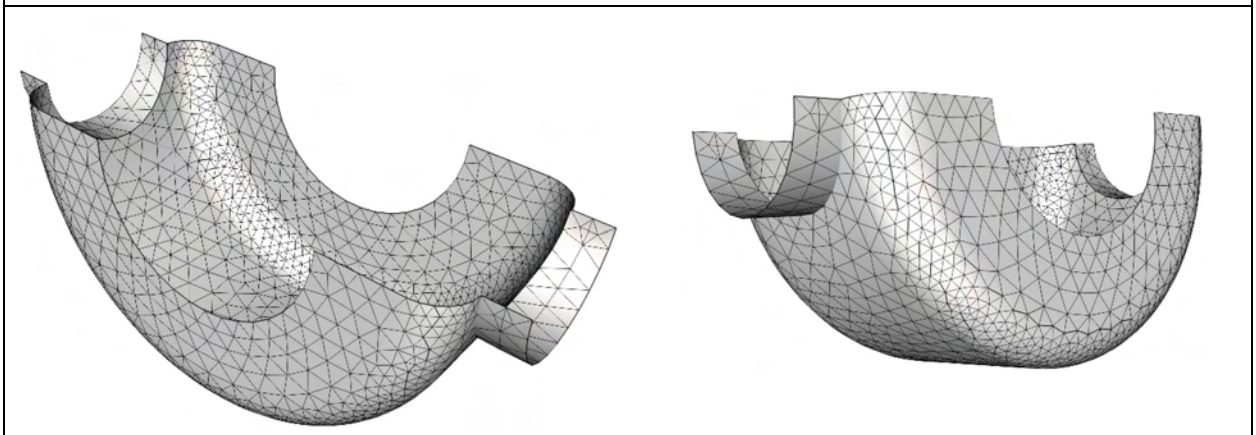
Figura 4.19 –Malhas em superfícies recortadas por interseções.

O Exemplo 4.10 (Fig. 4.19b) contém uma superfície Regrada (a mesma do Exemplo 4.4) e um cilindro reto gerado por Revolução. No cilindro, a malha foi gerada na porção acima da linha de interseção. Na superfície Regrada, foi feito um furo pela interseção do cilindro. O Exemplo 4.11 (Fig. 4.20a) é composto pelo tubo do Exemplo 4.4 e cilindro gerado como superfície Regrada. O tubo é furado pelo cilindro que recebeu malha somente na porção que é interno ao tubo.

O Exemplo 4.12 (Fig.4.20b) é constituído por uma superfície de Revolução e uma superfície Regrada. A superfície Regrada recebe malha nas duas porções das extremidades e a porção interna foi removida.



(a) Exemplo 4.11: Sweep e Regrada



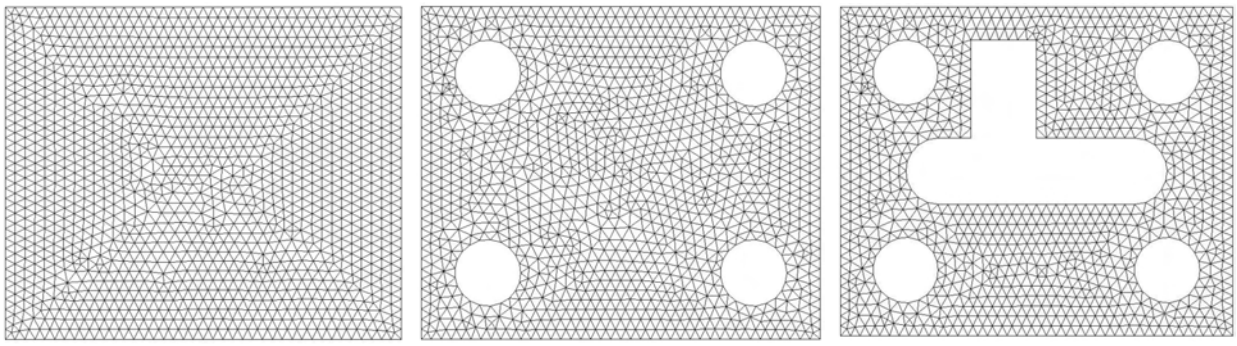
(b) Exemplo 4.12: Revolução e Regrada

Figura 4.20 – Malhas em superfícies recortadas por interseções.

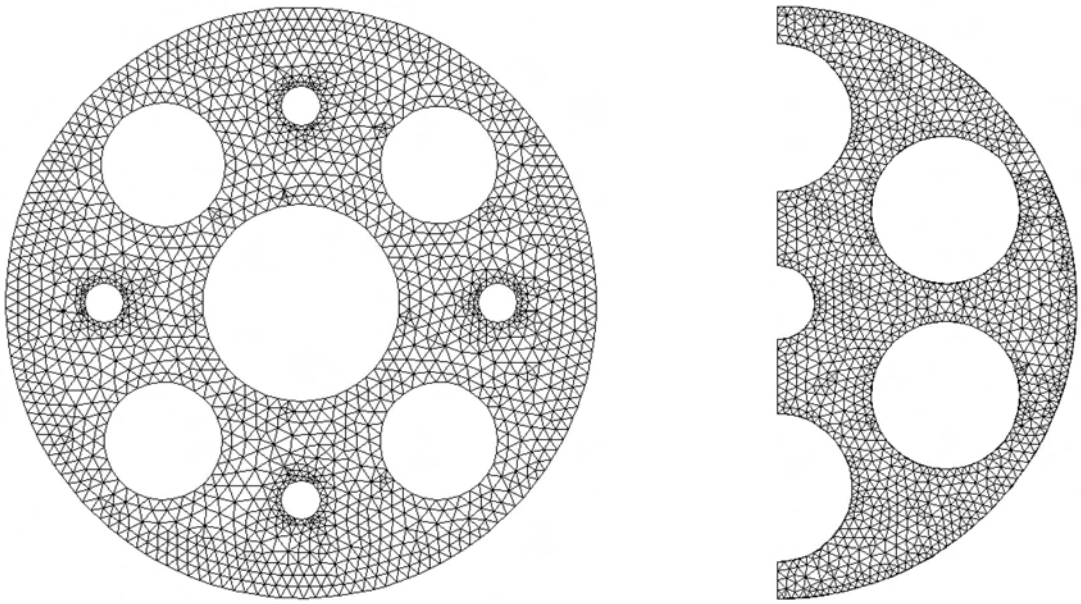
4.2.3. Superfícies Planas com Recortes Definidos pelo Usuário

Finalmente, os Exemplos 4.13 a 4.20 (Fig. 4.21) apresentam superfícies planas com recortes definidos pelo usuário. Todos estes Exemplos foram gerados diretamente a partir de linhas (Retas, Arcos e Círculos) definidos pelo usuário, que foram utilizadas para construir superfícies Planas (ver Cap. 2, Seção 2.2.1). Os domínios, sobre os quais foram geradas as malhas, também definidos pelo usuário.

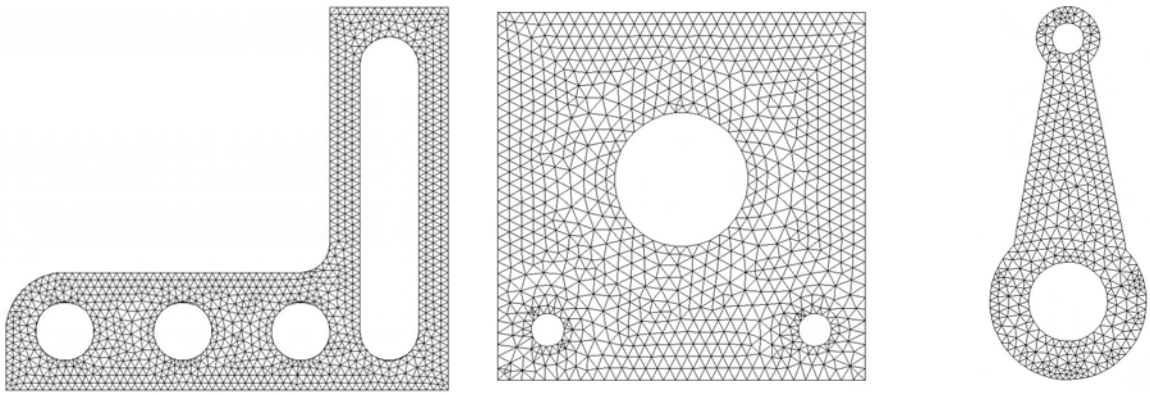
Embora o objetivo inicial no desenvolvimento do algoritmo fosse a geração de malhas em superfícies curvas, estes exemplos comprovam a eficiência (ver Tab. 4.1) do algoritmo na geração de malha em superfícies planas, recortadas ou não. Como a geração da malha é baseada somente na geometria do modelo, não há concentração de malha em zonas em que, eventualmente, isto poderia ser necessário, dependendo da aplicação.



(a) Exemplo 4.13 (b) Exemplo 4.14 (c) Exemplo 4.15



(d) Exemplo 4.16 (e) Exemplo 4.17



(f) Exemplo 4.18 (g) Exemplo 4.19 (h) Exemplo 4.20

Figura 4.21 – Malhas em superfícies planas recortadas por linhas definidas pelo usuário.

Tabela 4.1 – Resultados da geração de malha nos Exemplos 4.1 a 4.20.

Exemplo	*L*	*A*	*NN*	*NE*	$\overline{\alpha}$	α_{90}	α_{Min}	*CPU* (s)
	0,5	10	609	1122	0,96	91,9	0,49	2,3
4.1	0,5	5	1353	2594	0,95	86,5	0,43	9,3
	0,3	3	3820	7456	0,96	91,6	0,47	54,3
4.2	0,5	5	6905	13589	0,95	87,7	0,33	127,6
4.3	1,0	5	8990	17794	0,95	86,7	0,20	731,7
4.4	1,0	5	1380	2648	0,95	85,6	0,60	9,5
4.5	0,5	10	3077	5987	0,97	94,9	0,52	41,9
4.6	0,5	10	1593	3136	0,97	93,4	0,38	37,8
4.7	0,5	15	881	1576	0,98	96,9	0,59	7,2
4.8	0,5	15	2656	4959	0,97	99,6	0,40	18,7
4.9	0,5	10	2949	5498	0,98	94,4	0,52	47,2
4.10	0,5	15	2115	3894	0,98	95,9	0,47	20,3
4.11	0,5	15	2616	4928	0,97	93,0	0,20	58,2
4.12	0,5	15	1536	2890	0,97	93,9	0,61	8,4
4.13	0,3	15	1384	2620	0,99	99,7	0,76	7,8
4.14	0,3	15	1257	2282	0,98	97,0	0,66	7,9
4.15	0,3	15	1093	1876	0,97	95,8	0,66	9,9
4.16	0,3	15	1766	3133	0,97	92,4	0,36	18,3
4.17	0,25	10	1248	2150	0,97	92,0	0,58	13,7
4.18	0,25	15	1187	1988	0,98	97,9	0,75	17,3
4.19	0,35	15	940	1688	0,97	93,2	0,37	7,6
4.20	0,25	25	454	752	0,96	87,6	0,67	2,3

4.3. CONSIDERAÇÕES FINAIS

O algoritmo gera malhas triangulares em recortes de superfícies planas ou curvas e foi implementado no T-CADE. As malhas gerada são sensíveis às curvaturas das superfícies e dos contornos dos sub-domínios. É possível gerar malhas em mais de um sub-domínio em cada superfície, o que aumenta o potencial de geração do programa. A qualidade de todas as malhas apresentadas nos exemplos é satisfatória, com $\overline{\alpha} \geq 0{,}95$ em todos os Exemplos apresentados, como pode ser visto na Tabela 4.1.

Além disso, a taxa de Elementos com qualidade maior 0,9 é sempre alta e, na maioria dos casos, passa de 90%. Mesmo nos casos em que está abaixo deste patamar, o histograma da qualidade mostra que a imensa maioria dos Elementos ainda apresenta uma boa qualidade, como mostra o gráfico da Figura 4.22, que compara os Exemplos 4.2, 4.3 e 4.4.

A Figura 4.23 apresenta gráficos da distribuição do fator α médio para os mesmos exemplos, onde a cor azul representa valores de α próximos de 1. O vermelho corresponderia a $\alpha = 0$. Nota-se que há uma predominância do azul, o que traduz a qualidade das malhas obtidas.

O Exemplo 4.3 é que apresenta áreas maiores com valores de α<0,9 (azul claro), mas, ainda sim a malha gerada é de boa qualidade, com α médio de 0,95 . A qualidade geral da malha poderia ser facilmente melhorada se fosse utilizado um valor de *L* menor ou um valore de *A* maior.

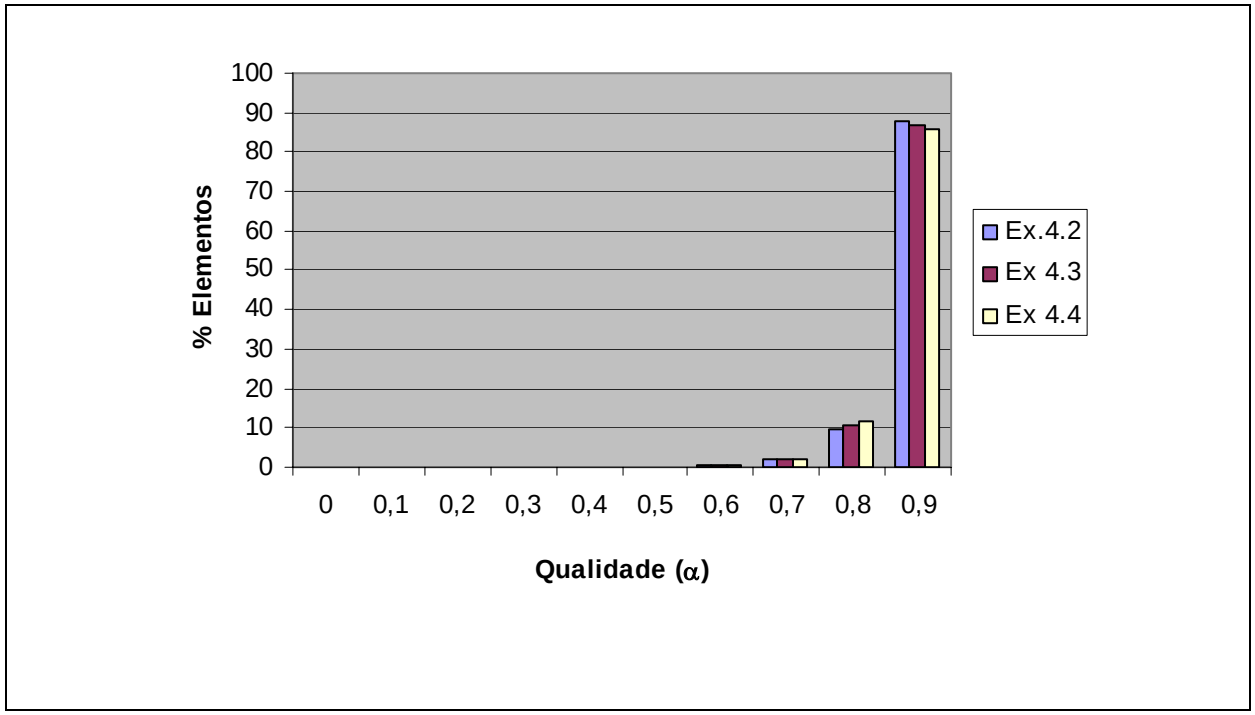


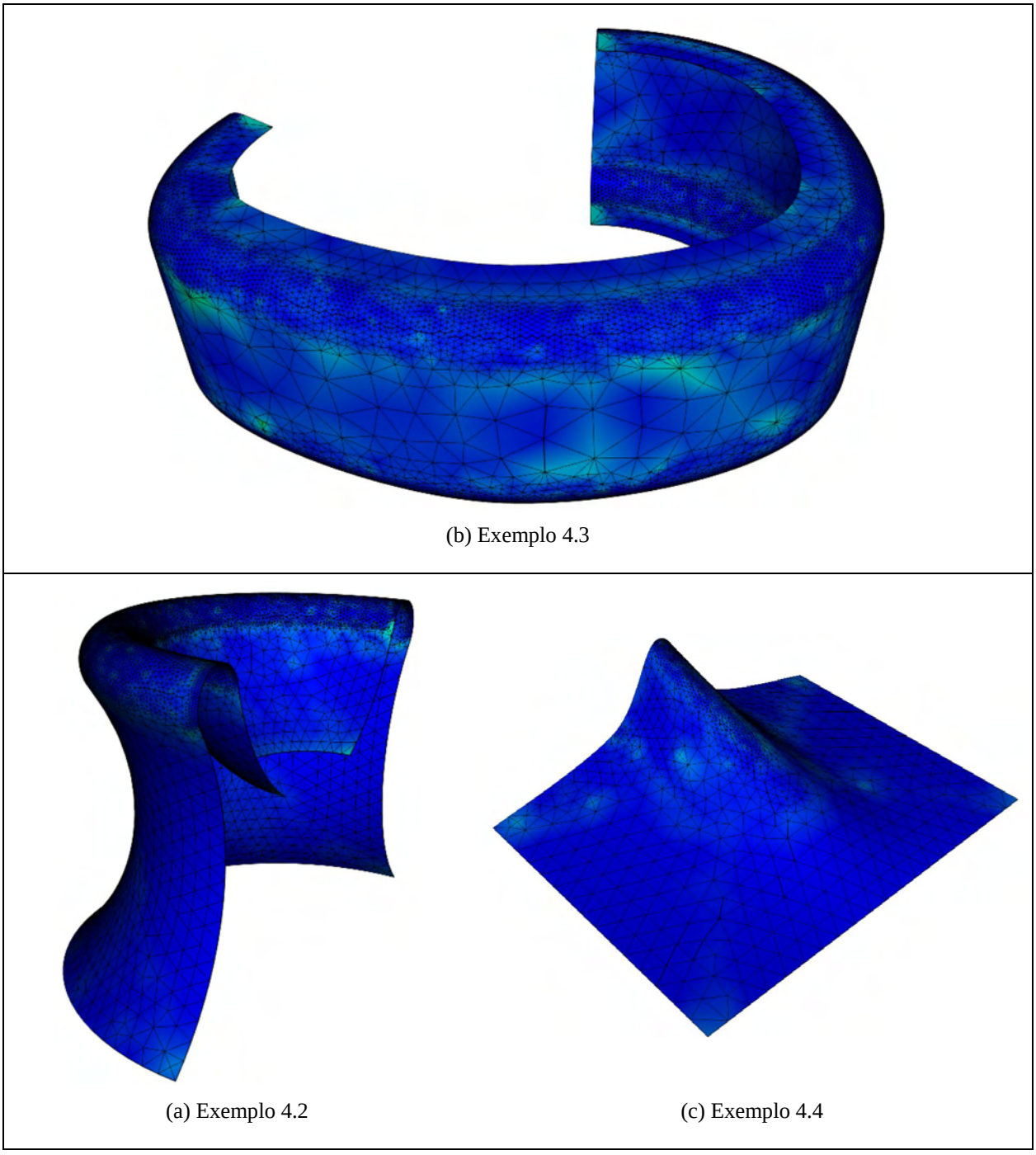
Figura 4.22 – Histograma da qualidade dos Elementos para os Exemplos 4.2, 4.3 e 4.4.

Pode-se melhorar muito a qualidade das malhas variando-se os parâmetros *L* e *A*, aumentando a precisão geométrica da malha. Em alguns casos, dependendo das condições de contorno e do tipo de análise efetuada, pode ser necessário aumentar a densidade das malhas em determinadas regiões ou, ainda, refazer toda a malha. Mesmo nestes casos, o potencial do gerador implementado pode ser aproveitado com algumas modificações. Um exemplo seria utilizar a distribuição de erro para redefinir o tamanho dos elementos em diferentes posições da geometria, que é um atributo da malha de fundo. Atualmente, este atributo está relacionado somente à curvatura da superfície, mas seria simples adicionar atributos relacionados a outros parâmetros, como o erro relativo. De qualquer forma, o algoritmo de geração permaneceria o mesmo em sua essência.

Quanto ao desempenho, apesar dos tempos de processamento registrados serem compatíveis com o modelamento iterativo, é necessário otimizar o desempenho numérico do algoritmo para que possa ser utilizado para a geração de problemas com grande número de elementos. A Figura 4.24 apresenta um gráfico de tempo de processamento de CPU relacionado com o número de elementos gerados, incluindo todas as etapas do algoritmo. O gráfico foi feito

para uma superfície Coons, mas o tempo final é dependente do tipo de superfície, pois a complexidade da função paramétrica da superfície é um parâmetro importante no tempo de processamento.

O algoritmo de geração de malha proposto pode ser utilizado para a geração de malhas para análise de problemas de Engenharia por Elementos Finitos, para a geração de malha de contorno para a geração de Elementos sólidos (tetraedros) e, ainda, para a visualização de superfícies (*rendering*).



(b) Exemplo 4.3

(a) Exemplo 4.2

(c) Exemplo 4.4

Figura 4.23 – Distribuição do fator de qualidade α nas malhas.

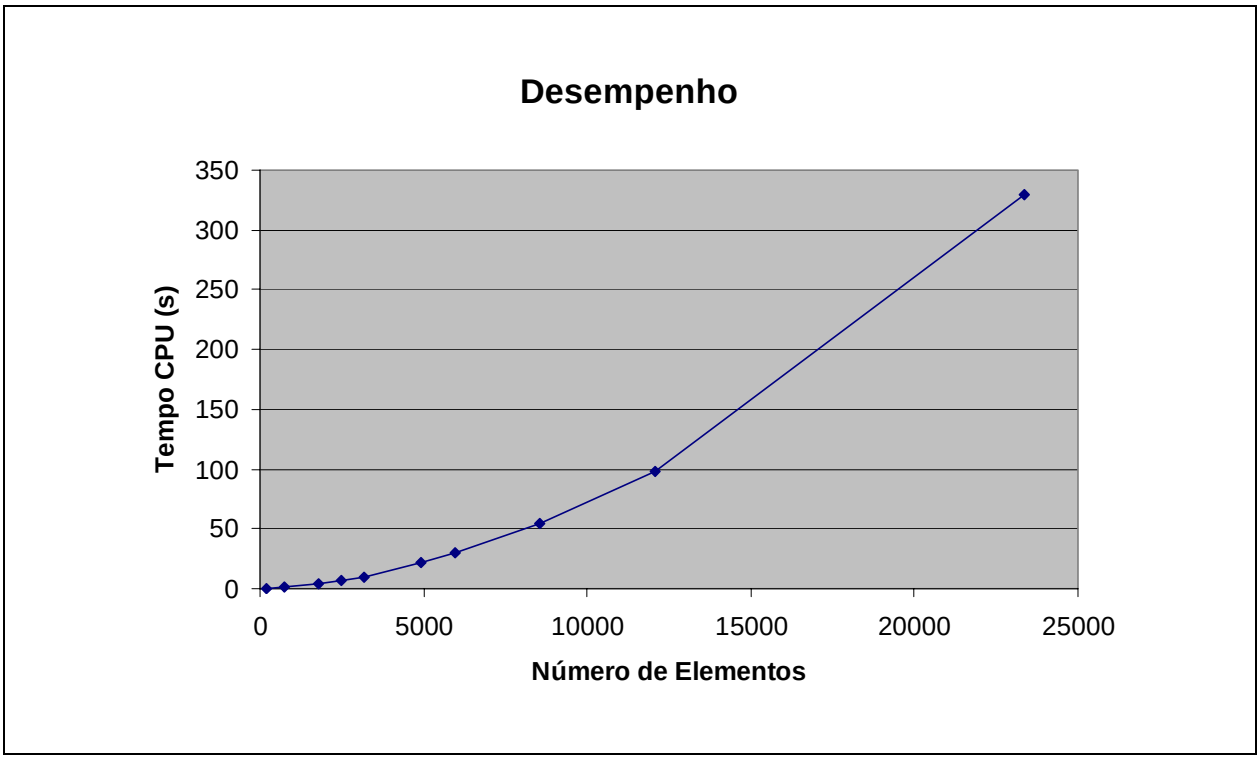


Figura 4.24 – Gráfico de desempenho do algoritmo de geração de malha.

5. IMPLEMENTAÇÃO COMPUTACIONAL

Com o objetivo de implementar os algoritmos desenvolvidos neste trabalho, optou-se pela construção de um ambiente computacional gráfico e interativo, o T-CADE. A idéia original era testar o desempenho dos algoritmos desenvolvidos. Mas, no decorrer das pesquisas, o ambiente mostrou-se uma ferramenta indispensável para o desenvolvimento dos algoritmos, pois agilizou o processo de implementação e evitou a necessidade de adaptar os algoritmos à plataforma de desenvolvimento, o que é comum quando se utilizam programas comerciais. Com o T-CADE, quando o algoritmo em desenvolvimento era restringido pela plataforma, modificava-se a plataforma. Isto foi crucial no desenvolvimento desta Tese. O T-CADE é uma interface que permite o modelamento interativo, a visualização 3D e a manipulação direta de objetos através do dispositivo apontador (*mouse*). Esta interface permitiu a implementação e testes dos algoritmos com uma agilidade que não seria possível de outra forma.

O ambiente consiste de um conjunto de ferramentas de visualização e manipulação de objetos em 3D, ferramentas de edição de objetos e de posicionamento de Sistemas de Referência. A linguagem de programação utilizada foi o Delphi [Cantù, 1999], que permite orientação a objetos e apresenta ótimo desempenho, tanto em aplicações numéricas, como em aplicações gráficas. A seguir serão descritas as principais etapas envolvidas na implementação do T-CADE.

5.1. CLASSES DE OBJETOS GEOMÉTRICOS

Os objetos geométricos foram organizados em quatro classes fundamentais que definem hierarquia e complexidade: *TPonto*, *TObjetoGeométrico*, *TLinha* e *TSuperfície*[1]. Os objetos *TPonto* definem posição no espaço 3D e são fundamentais entre outras coisas como propriedades dos objetos mais complexos. A classe *TObjetoGeométrico* encerra as propriedades e métodos básicos de todo objeto geométrico e são herdados pelas classes filhas, que são: *TLinha* e *TSuperfície*. A classe *TLinha* contém as propriedades e métodos gerais das linhas e engloba as classes: *TReta*, *TArco*, *TSpline* e *TPolilinha*, as quais têm propriedades e métodos específicos, além dos herdados. As superfícies têm em *TSuperfície* a sua classe genérica da qual derivam: *TPlanar*, *TBilinear*, *TRegrada*, *TCoons*, *TRevolução* e *TSweep*. A Figura 5.1 apresenta um gráfico desta organização hierárquica de classes.

[1] Por padrão, o Delphi utiliza sempre a letra *T* para iniciar um nome de classe. Adotou-se aqui a mesma notação para que fique claro que trata-se efetivamente de uma classe.

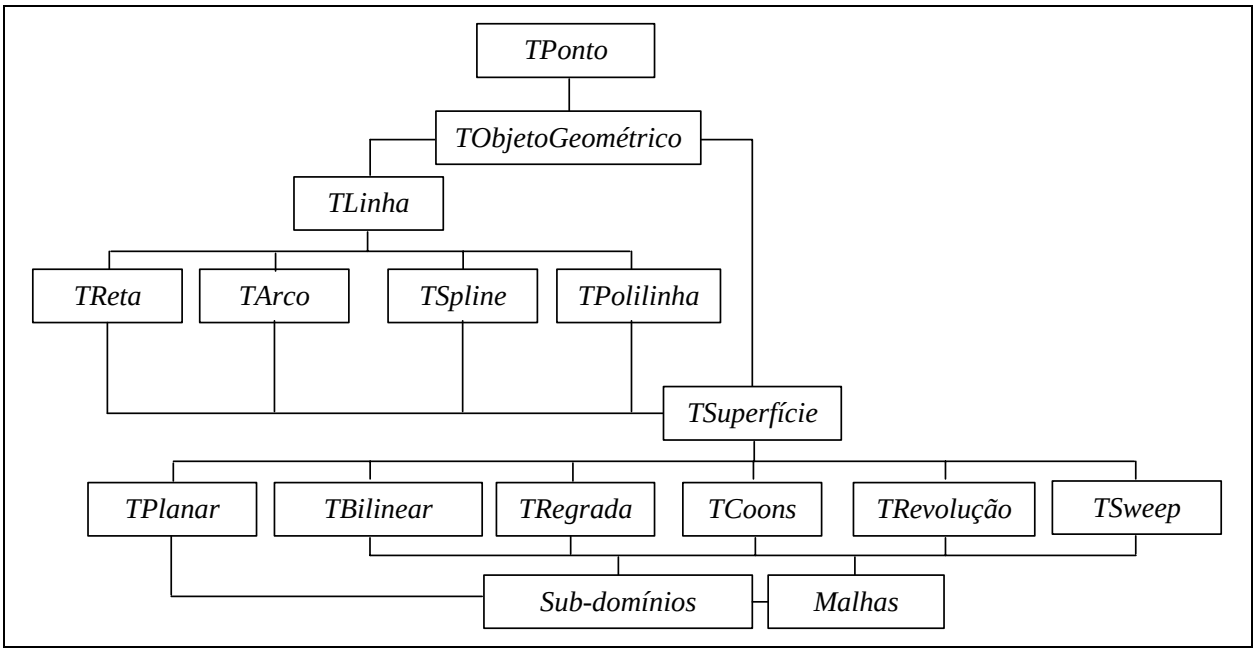


Figura 5.1 – Hierarquia de classes dos objetos geométricos.

Esta forma de organização facilita a implementação de novos tipos de parametrizações de linhas e superfícies, sem, muitas vezes, a necessidade de rescrever códigos específicos para um determinado tipo de objeto. Um exemplo disto é a função que encontra o ponto inicial e o ponto final de uma linha, que é uma função da classe mãe *TLinha* e não precisa ser rescrita em nenhuma das classes filhas, qualquer que seja a parametrização. Esta idéia também é utilizada na determinação de interseções e na geração de malha, funções que são executadas a partir da classe *TSuperfície*. As linhas de recorte geradas pela interseção entre superfícies, assim como uma malha gerada sobre uma superfície são propriedades da classe *TSuperfície* e independem da parametrização, que é definida na classe filha.

A geração das Linhas e Superfícies é feita de forma interativa, utilizando o *mouse* e ferramentas de restrição coordenadas, ferramentas de atração e de manipulação do Sistemas de Referência (ver Seção 5.1.3). A estratégia de modelamento consiste em construir primeiro as Linhas, que servirão de base para a construção das Superfícies. As Superfícies são geradas utilizando as Linhas como curvas de bordo ou como curvas de recorte (*Trimming Lines*), no caso de Superfícies Planas.

5.1.1. Classe *TPonto*

A classe *TPonto* representa um ponto com as suas coordenadas 3D. É uma classe fundamental para o sistema de posicionamento, visualização e geração de todos os objetos, incluindo os Sistemas de Referência, câmeras e objetos geométricos. Esta classe não possui

nenhum método associado, possuindo somente as propriedades correspondentes às componentes em cada eixo (*x*, *y*, *z*).

Tabela 5.1 – Classe *TPonto*.

Nome	*TPonto*	
Classe Mãe	*não tem*	
Propriedades		
Nome	**Descrição**	**Tipo**
x	*Coordenada na direção X*	*Real*
y	*Coordenada na direção Y*	*Real*
z	*Coordenada na direção Z*	*Real*

5.1.2. Classe *TObjetoGeometrico*

A classe *TObjetoGeometrico* apresenta as propriedades e métodos fundamentais comuns a todas as classes de objetos gráficos. Essa classe fundamental possui duas grandes classes filhas: classe TLinha e classe TSuperficie. Essas, por sua vez, possuem diversas classes filhas, uma para cada objeto gráfico específico. A tabela 3.1 apresenta as propriedades e métodos da classe Objeto Geométrico.

Tabela 5.2 – Classe *TObjetoGeometrico*.

Nome	*TObjetoGeometrico*	
Classe Mãe	*não tem*	
Propriedades		
Nome	**Descrição**	**Tipo**
Id	*Código de identificação interno*	*Inteiro*
Cor	*Código da cor do objeto*	*Inteiro*
Tipo	*Código do tipo de objeto. Existe um código para cada tipo de objeto gráfico.*	*Inteiro*
Classe	*Classe de objetos*	*Classe*
Métodos		
Nome	Descrição	Tipo
Clic	*Verifica se um objeto é clicado com o mouse.*	*TPonto/Abstrato*
Desenha	*Desenha o objeto no dispositivo VPData.*	*Abstrato*
Ext	*Encontra uma extremidade do objeto.*	*TPonto/Abstrato*
Med	*Encontra um ponto médio do objeto.*	*TPonto/Abstrato*
Prox	*Encontra um ponto sobre o objeto.*	*TPonto/Abstrato*

Todos os objetos geométricos criados são armazenados em um arranjo de objetos da classe TObjetoGeometrico. Quando os objetos devem ser desenhados, esse arranjo é varrido e o método *Desenha* de cada objeto é acionado. O método *Desenha* da classe TObjetoGeometrico é Abstrato. Isso significa que não existe um código específico para esse método nessa classe, mas, ao ser acionado, esse método executa o código correspondente ao mesmo método da classe filha. Assim, conforme o tipo de objeto, métodos específicos são acionados sem que seja necessário criar estruturas condicionais para selecionar tais métodos em função das classes de objetos envolvidas. Apesar dessa propriedade ser típica de uma linguagem de programação orientada a objetos, como é o Delphi, é necessário um certo planejamento para tirar o máximo de proveito de sua implementação.

5.1.3. Classe *TLinha*

A classe *TLinha* é a classe mãe da qual derivam todas as demais classes de linhas e contém métodos comuns a todas as linhas. A Tabela 5.3 apresenta os principais métodos da classe *TLinha*.

Tabela 5.3 – Classe *TLinha*.

Nome	*TLinha*	
Classe Mãe	*TObjetoGeometrico*	
Métodos		
Nome	**Descrição**	**Tipo**
Ft(t)	*Equação paramétrica (abstrata)*	*TPonto/Abstrato*
DFt(t)	*Derivada na direção t normalizada*	*TPonto/Abstrato*
Comp	*Calcula o comprimento do objeto.*	*Real*
Cen	*Encontra o centro de curvatura (arcos e círculos)*	*TPonto*
Per	*Encontra o ponto na perpendicular a linha*	*TPonto/Abstrato*

Os métodos *Ft*(*t*) e *DFt(t)* são Abstratos, portanto cada classe filha de *TLinha* deve ter métodos *Ft*(*t*) e *DFt(t)* específicos para cada tipo de linha implementado. A vantagem de utilizar uma classe mãe *TLinha* é que, em qualquer operação onde um objeto das classes filhas é necessário, é possível utilizar um objeto da classe *TLinha* como um objeto linha genérico. Um exemplo é a superfície regrada que tem como propriedades dois objetos *TLinha*. Dessa forma, qualquer objeto de classes derivadas da classe *TLinha* (*TReta*, *TArco*, *TSpline* e *TPolilinha* – Tabelas 5.4 a 5.7) pode ser utilizado para criar uma superfície regrada.

Tabela 5.4 – Classe *TReta*.

Nome	*TReta*	
Classe Mãe	*TLinha*	
Propriedades		
Nome	**Descrição**	**Tipo**
Ponto1	*Ponto de início.*	*TPonto*
Ponto2	*Ponto de fim.*	*TPonto*

Tabela 5.5 – Classe *TArco*.

Nome	*TArco*	
Classe Mãe	*TLinha*	
Propriedades		
Nome	**Descrição**	**Tipo**
Ang1	*Ângulo inicial*	*Real*
Ang2	*Ângulo final*	*Real*
Centro	*Centro de curvatura*	*TPonto*
Raio	*Raio de curvatura*	*Real*
Ponto1	*Ponto de início (Ang1)*	*TPonto*
Ponto2	*Ponto de fim (Ang2)*	*TPonto*
Plano	*Sistema de coordenadas do objeto*	*TSis*

Tabela 5.6 – Classe *TSpline*.

Nome	*TSpline*	
Classe Mãe	*TLinha*	
Propriedades		
Nome	**Descrição**	**Tipo**
NPontos	*Número de pontos de controle*	*Inteiro*
Pontos[ ]	*Arranjo com todos os pontos de controle da curva*	*TPonto*

Tabela 5.7 – Classe *TPolilinha*.

Nome	*TPolilinha*	
Classe Mãe	*TLinha*	
Propriedades		
Nome	**Descrição**	**Tipo**
NLinhas	*Número de linhas que compõem a polilinha.*	*Inteiro*
Linhas[ ]	*Arranjo com todos as linhas que compõem a Polilinha*	*TLinha*

5.1.4. Classe *TSuperficie*

Uma classe genérica *TSuperficie* apresenta as principais propriedades e métodos herdados pelas demais classes de superfície. A classe *TSuperficie* contém uma malha de quadriláteros utilizada especificamente para operações de sombreamento. Assim, todas as superfícies podem ser visualizadas com efeitos de sombreamento. Essa malha pertence à classe *TMalha* que será abordada ao longo deste trabalho. Além disso, a classe genérica *TSuperficie* contém métodos para determinar vetores tangentes e normais à superfície. A Tabela 5.8 apresenta as principais propriedades e métodos da classe *TSuperficie*.

Tabela 5.8 – Classe *TSuperficie*

Nome	*TSuperficie*	
Classe Mãe	*TObjetoGeometrico*	
Propriedades		
Nome	**Descrição**	**Tipo**
NVértices	*Número de vértices*	*Inteiro*
Vértices[]	*Arranjo que contém os vértices da borda da superfície*	*TPonto*
Malha	*Malha de utilizada para operações de sombreamento.*	*TMalha*
Métodos		
Nome	**Descrição**	**Tipo**
Ft(u,v)	*Equação paramétrica*	*TPonto*
Du(u,v)	*Derivada à superfície em relação a u*	*TPonto*
Dv(u,v)	*Derivada à superfície em relação a v*	*TPonto*
N(u,v)	*Vetor normal à superfície*	*TPonto*
GMalha	*Gera malha em um sub-domínio*	–

Tabela 5.9 – Classe *TPlanar*.

Nome	*TPlanar*	
Classe Mãe	*TSuperficie*	
Propriedades		
Nome	**Descrição**	**Tipo**
P1	*Primeiro Ponto*	*TPonto*
P2	*Segundo Ponto*	*TPonto*
P3	*Terceiro Ponto*	*TPonto*
P4	*Quarto Ponto*	*TPonto*

Cálculo de interseções e geração de malha são operações realizadas diretamente sobre a classe *TSuperficie*, o que abrange todas as classes derivadas, não importando o tipo de parametrização. As Tabelas 5.9 a 5.14 apresentam as principais propriedades das classes derivadas de *TSuperficie*: *TPlanar* (Tab. 5.9), *TBilinear* (Tab. 5.10), *TRegrada* (Tab. 5.11), *TCoons* (Tab. 5.12), *TRevolucao* (Tab. 5.13) *e TSweep* (Tab. 5.14).

Tabela 5.10 – Classe *TBilinear*.

Nome	*TBilinear*	
Classe Mãe	*TSuperficie*	
Propriedades		
Nome	**Descrição**	**Tipo**
P1	*Primeiro Ponto*	*TPonto*
P2	*Segundo Ponto*	*TPonto*
P3	*Terceiro Ponto*	*TPonto*
P4	*Quarto Ponto*	*TPonto*

Tabela 5.11 – Classe *TRegrada*.

Nome	*TRegrada*	
Classe Mãe	*TSuperficie*	
Propriedades		
Nome	**Descrição**	**Tipo**
Linha1	*Primeira linha de base da superfície.*	*TLinha*
Linha2	*Segunda linha de base da superfície.*	*TLinha*

Tabela 5.12 – Classe *TCoons.*

Nome	*TCoons*	
Classe Mãe	*TSuperficie*	
Propriedades		
Nome	**Descrição**	**Tipo**
Lado1	*Primeira curva do bordo*	*TLinha*
Lado2	*Segunda curva de bordo (adjacente a Lado1)*	*TLinha*
Lado3	*Terceira curva de bordo (adjacente a Lado2)*	*TLinha*
Lado4	*Quarta curva de bordo (adjacente a Lado3)*	*TLinha*

Tabela 5.13 – Classe *TRevolucao.*

Nome	*TRevolucao*	
Classe Mãe	*TSuperficie*	
Propriedades		
Nome	**Descrição**	**Tipo**
Perfil	*Linha do perfil a sofrer a revolução em torno do eixo.*	*TLinha*
P0	*Primeiro ponto que define o eixo de revolução.*	*TPonto*
P1	*Segundo ponto que define o eixo de revolução.*	*TPonto*
Ang	*Ângulo de revolução*	*Real*

Tabela 5.14 – Classe *TSweep.*

Nome	*TRevolucao*	
Classe Mãe	*TSuperficie*	
Propriedades		
Nome	**Descrição**	**Tipo**
Perfil	*Linha que define o perfil a sofrer o Sweep*	*TLinha*
Caminho	*Linha que define o caminho do Sweep.*	*TLinha*

5.2. VISUALIZAÇÃO 3D

O objetivo desta etapa do trabalho é obter um ambiente onde seja possível visualizar objetos 2D e 3D durante e após o modelamento. Assim, foi desenvolvido um ambiente utilizando conceitos já bem conhecidos em computação gráfica, como Câmera Sintética, transformações de coordenadas e cálculo de projeções. Estes temas são amplamente abordados na literatura. Como exemplos de referência podem-se citar: Foley et al, 1992; Castier et al, 1994; Watt, 1993. Como o ambiente gráfico em si não é o objetivo final deste trabalho, e sim um meio para o seu desenvolvimento, foram implementadas rotinas gráficas com simplificações, a fim de agilizar esta etapa do processo. Além disso, bibliotecas gráficas como OpenGL [Woo, 1999] e Direct 3D [Trujillo, 1997] possuem soluções completas em termos de visualização, incluindo sombreamento de alta qualidade para superfícies. Em etapas futuras, pretende-se enriquecer a implementação atual utilizando-se, em princípio, OpenGL, devido à sua maior portabilidade para várias plataformas.

Por questões de tempo e por não ser o foco principal deste trabalho, foi implementada uma câmera simplificada que realiza projeções ortogonais cilíndricas e cônicas. No entanto, não foi implementado recorte 3D (*clipping*). A câmera é definida pela sua posição,

alvo e distância focal. O eixo *x* do Sistema de Referência da Câmera é mantido sempre paralelo ao plano *xy* global. Na interface com o usuário, utilizam-se como parâmetros de posicionamento da câmera os ângulos de rotação (θ), inclinação (ϕ) e distância ao alvo (Fig. 5.2a). O alvo é, inicialmente, localizado na origem do Sistema Global. O usuário também pode controlar a posição da câmera, arrastando o *mouse* sobre a tela. Desta forma, o deslocamento na direção horizontal incrementa o ângulo de rotação da câmera e o deslocamento vertical incrementa o ângulo de inclinação da câmera. Além disso, é possível escolher dentre vistas pré-definidas, o que facilita o posicionamento rápido da câmera. A câmera foi implementada através de uma classe específica (*TCam* – Tab. 5.15) e a sua manipulação se dá pela alteração das suas propriedades, principalmente a posição e orientação. A Figura 5.1a apresenta o esquema de posicionamento da câmera e a Figura 5.2b mostra a tela do T-CADE com a interface de posicionamento de câmera.

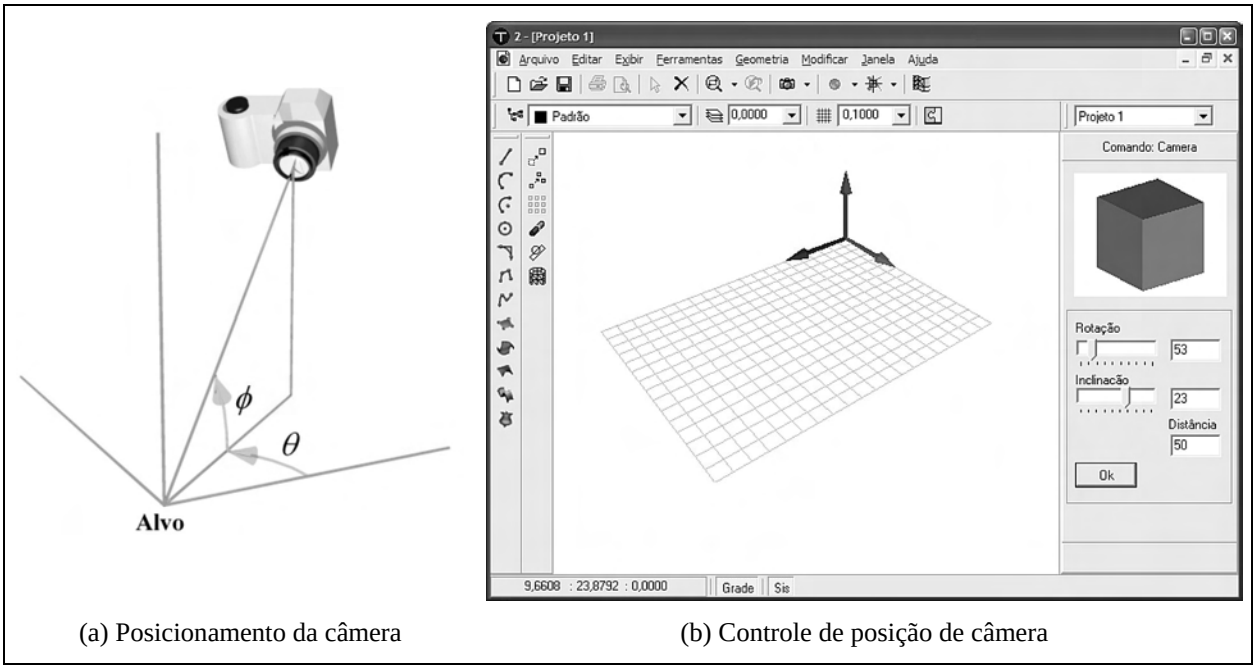
Tabela 5.15 – Classe *TCam*.

Nome	*TCam*	
Classe Mãe	*não tem*	
Propriedades		
Nome	***Descrição***	***Tipo***
Posição	*Posição da câmera no SCU*	*TPonto*
Alvo	*Posição do alvo da câmera no SCU*	*TPonto*
Cônica	*Determina se a projeção será cônica ou cilíndrica.*	*Booleana*
Métodos		
Nome	***Descrição***	***Tipo***
Tela	*Coordenadas de tela de um Ponto*	*P 2D*
Global	*Coordenadas do Universo de um ponto da tela*	*TPonto*

A câmera implementada é parte do mecanismo que proporciona interatividade com os objetos gráficos, permitindo a manipulação de objetos pelo *mouse*, pois estão embutidas na classe *TCam* as transformações de coordenadas nos dois sentidos (Câmera – Objeto e Objeto – Câmera) tanto para Sistema o Referência Global como para Sistemas de Referência Auxiliares (ver Seções 5.1.2 e 5.1.3). Isto possibilita ao usuário interagir com os objetos utilizando o *mouse*, realizar operações de desenho e editar objetos com precisão em qualquer posição do espaço 3D virtual.

Para a visualização das malhas geradas, foi implementado um sistema de visualização por realidade virtual, que utiliza o padrão VRML [Ames, 1996 e Hartman, 1996] e

um componente do Delphi que consiste em um *Web Browser*. Para que este tipo de visualização possa ser utilizada, é necessário que haja algum *plug-in* de visualização de VRML instalado no sistema operacional. Apesar de a interatividade estar limitada à manipulação da câmera, a qualidade da visualização proporcionada é realmente muito boa, com iluminação e render de qualidade, incluindo transparências e interpolação de cores.



(a) Posicionamento da câmera (b) Controle de posição de câmera

Figura 5.2 – Câmera sintética implementada no T-CADE.

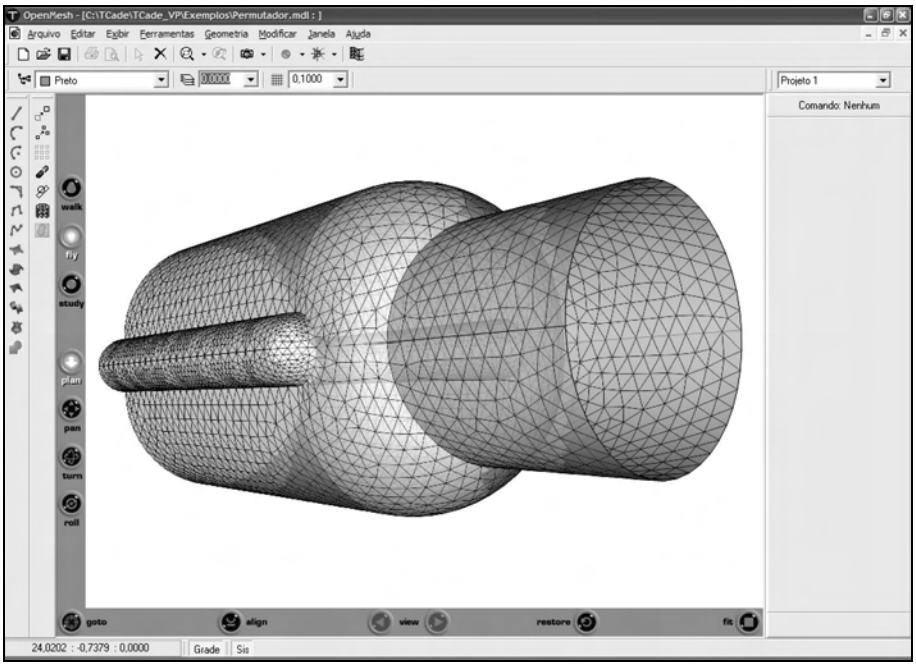


Figura 5.3 – Interface para visualização de malhas em VRML.

A Figura 5.3 apresenta a interface do programa com uma malha, gerada em um dos exemplos analisados na Seção 5.3, com efeito de transparência. Pode-se observar a interface do *plug-in* de visualização nas bordas da janela. O *plug-in* utilizado aqui é o Cortona (*www.parallelgraphics.com*), mas existem outros como o Microsoft VRML *Viewer* (*www.microsoft.com*) e o Cosmo Player (*http://ca.com/cosmo*). Todos possuem versões gratuitas.

A interatividade, proporcionada pela Câmera Sintética simplificada, e a qualidade de visualização de malhas, obtida com o VRML, formam um conjunto que permite agilidade nos processos de modelamento interativo e no controle da qualidade das malhas geradas. Várias figuras de exemplos apresentados no Capítulo 4, e neste, foram geradas com a visualização em VRML do T-CADE.

5.2.1. Sistemas de Referência

O T-CADE utiliza um Sistema de Referência Global (SRG), onde todos os objetos e câmera são referenciados, mas também permite a definição de Sistemas de Referência Auxiliares (SRA), os quais podem ser utilizados tanto pelo usuário, no auxílio ao modelamento tridimensional, como pelo próprio sistema, como uma ferramenta extremamente importante em diversas implementações.

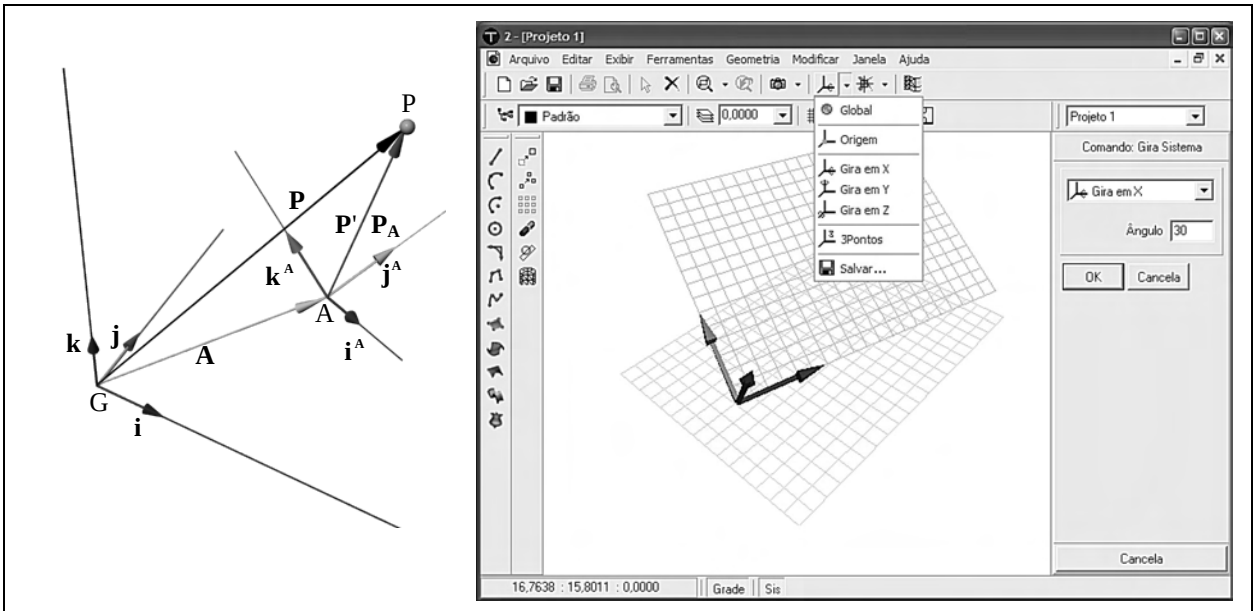
Tabela 5.16 – Classe *TSis*.

Nome	*TSis*	
Classe Mãe	*não tem*	
Propriedades		
Nome	***Descrição***	***Tipo***
Origem	*Origem do Sistema de Coordenadas Auxiliar.*	*TPonto*
PontoX	*Ponto que define a direção e o sentido positivo do novo eixo X*	*TPonto*
PontoY	*Ponto que define a direção e a orientação do eixo Y do novo sistema de coordenadas. Caso o PontoY não defina uma direção perpendicular ao eixo X, sua direção é corrigida.*	*TPonto*
Métodos		
Nome	***Descrição***	***Tipo***
Local	*Transforma de SRU para SRA*	*TPonto*
Global	*Transforma de SRA para SRU*	*TPonto*

As seguintes fontes abordam o tema de transformação de Sistemas de Referência:

Spiegel, 1973; Foley et al, 1992, e Watt, 1993. Da forma com foi implementado, o usuário pode realizar três tipos de alterações do Sistema de Referência: mudança de origem, rotação em relação aos eixos *x*, *y* e *z*, e mudança de plano de trabalho, fornecendo-se três pontos. Nos dois primeiros tipos, parte-se da situação corrente do Sistema de Referência. Já na definição por três pontos, cria-se um novo Sistema de Referência em função de parâmetros totalmente definidos pelo usuário.

Para facilitar o uso de sistemas de coordenadas auxiliares, foi criada uma classe *TSis* (Tab. 5.16) que encapsula as operações de transformações de coordenadas tanto no sentido Global – Local, como no sentido Local – Global. As propriedades básicas desta classe, e suficientes para definir um sistema de referência, são a origem e os vetores unitários (**i**, **j** e **k**) que definem a sua base ortonormal. Assim, toda vez que há uma alteração destas propriedades, altera-se o Sistema de Referência.



(a) Sistemas de Referência Auxiliares (b) Mudança de sistema de referência no ambiente gráfico

Figura 5.4 – Sistemas de Referência no T-CADE.

Considerando-se um ponto qualquer **P** (Fig. 5.4a) no sistema de referência global (*G*), a transformação para o um Sistema de Referência Auxiliar com origem em um ponto **A** fica assim definida:

$$\mathbf{P}_A = \mathbf{T}_A \cdot [\mathbf{P} - \mathbf{A}] \tag{5.1}$$

onde $\mathbf{T}_A$ é a matriz de transformação cujas linhas correspondem às componentes dos vetores $\mathbf{i}^A$, $\mathbf{j}^A$ e $\mathbf{k}^A$. A transformação inversa fica:

$$\mathbf{P} = \mathbf{T}_A^T \cdot \mathbf{P}_A + \mathbf{A} \tag{5.2}$$

onde $\mathbf{T}_A^T$ é a matriz transposta de $\mathbf{T}_A$, que coincide com a sua inversa, pois $\mathbf{T}_A$ é ortogonal. A Figura 5.4b apresenta a interface do ambiente gráfico com um Sistema de Referência Auxiliar definido pelo usuário.

5.2.2. Interatividade

O ambiente gráfico do T-CADE permite ao usuário desenhar de forma interativa utilizando *mouse*. Também é possível selecionar os objetos diretamente com o *mouse*, mover e copiar objetos com visualização em tempo real, além de obter coordenadas de objetos existentes (extremidades, ponto médio, quadrante e centro de arcos e círculos e perpendicular) para serem utilizadas no modelamento interativo de curvas e superfícies.

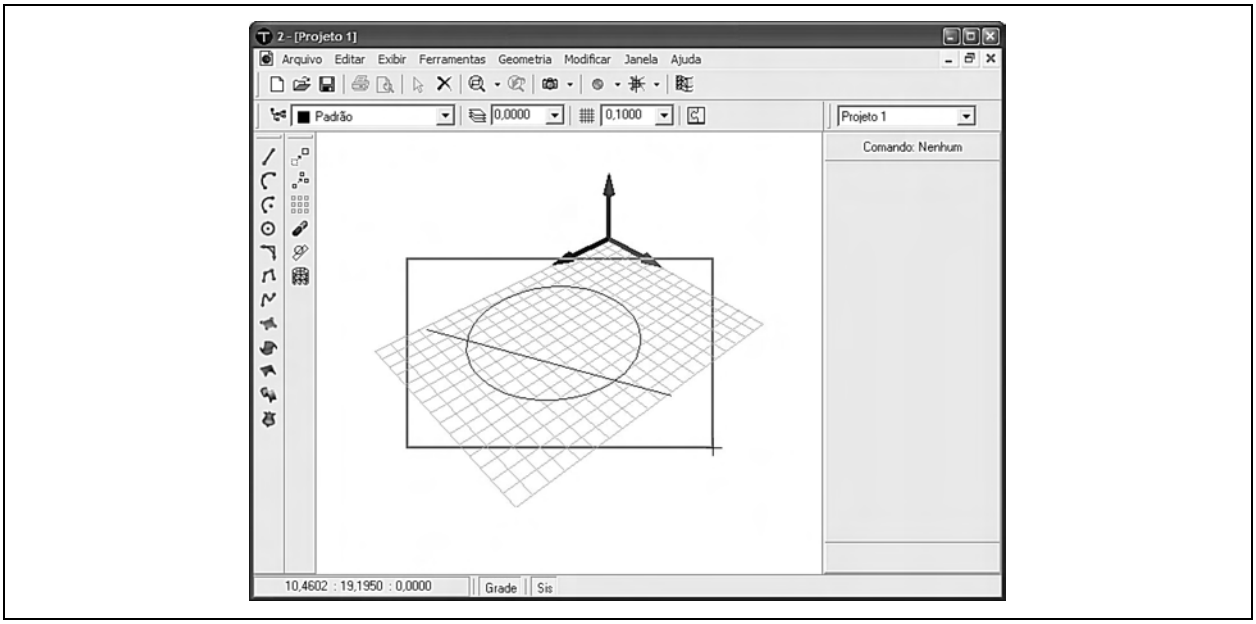


Figura 5.5 – Exemplo de seleção de objetos por janela definida pelo usuário.

5.2.2.1. Seleção de objetos

A seleção direta de objetos na tela utiliza as coordenadas de tela para verificar se e qual objeto foi selecionado. A classe *TObjetoGeometrico*, que abrange todos as linhas e superfícies no programa, têm como uma de suas propriedades um arranjo de retas de tela, as quais pertencem à classe *TRetaTela*. Quando se clica na tela, verifica-se a proximidade das coordenadas do cursor a todos objetos desta classe existentes. Quando a distância é menor que

uma dada tolerância (em geral, 5 pixels), um evento *Clic*, do objeto geométrico ao qual a reta de tela pertence, é executado. Um outra forma de seleção é feita quando uma janela é definida pelo usuário. Neste caso, verificam-se quais objetos geométricos possuem retas de tela no interior da janela de seleção. Estes são então selecionados.

A seleção de objetos apresenta grande velocidade, mesmo quando há muitos objetos para serem verificados, principalmente porque não há transformação de coordenadas, nem operações de ponto flutuante, pois as coordenadas da tela são inteiras. A Seleção de objetos é utilizada para escolher objetos a editar e para construir novos objetos (como superfícies) a partir de objetos existentes (linhas).

5.2.2.2. Determinação de pontos sobre objetos

A determinação de pontos sobre objetos gráficos, tais como extremidades, ponto médio e centro, chamada de ferramenta de atração, utiliza o mesmo evento *Clic* descrito no item anterior (5.1.4.1). Ao mover o cursor do *mouse* próximo a objetos gráficos, se houver alguma ferramenta de atração ligada, ao ser disparado um evento *Clic*, verifica-se a proximidade do cursor aos pontos selecionados. Toma-se aquele ponto mais próximo da posição do cursor. O que é feito com grande precisão, pois utilizam-se as equações paramétricas dos objetos. Por exemplo, quando ativa-se a ferramenta *Extremidades* de uma curva, toma-se dentre os pontos $\mathbf{C}(0)$ e $\mathbf{C}(1)$ aquele que estiver mais próximo do ponto de *Clic*. Como é possível perceber, é necessário verificar a coordenada paramétrica de onde ocorreu o evento *Clic*, o que é feito pela interpolação a partir das posições paramétricas das extremidades das retas de tela do objeto. Este tipo de implementação facilita muito o processo de desenho interativo, pois evita a entrada manual de coordenadas já existentes e proporciona grande precisão na manipulação de objetos gráficos.

5.2.2.3. Visualização em tempo real

Cópia e movimentação de objetos com visualização em tempo real são características que proporcionam grande interatividade na manipulação de objetos. O uso de programação orientada a objetos facilita muito a implementação deste tipo de característica. Uma das propriedades comuns a todos os objetos da classe *TObjetoGeometrico* é o ponto de inserção. A movimentação de um objeto geométrico é feita simplesmente alterando-se esta propriedade fundamental e, como a coordenada utilizada é a mesma do cursor sobre o plano de trabalho, o movimento se dá em tempo real (Fig. 5.5). No entanto, é necessário converter as coordenadas de

tela do cursor do *mouse* para as coordenadas globais do ambiente gráfico, para que o deslocamento possa ser realizado corretamente no espaço 3D virtual. Isto é feito pela Câmera Sintética, que contém um objeto da classe *TSis*, já mencionado na Seção 5.1.2.

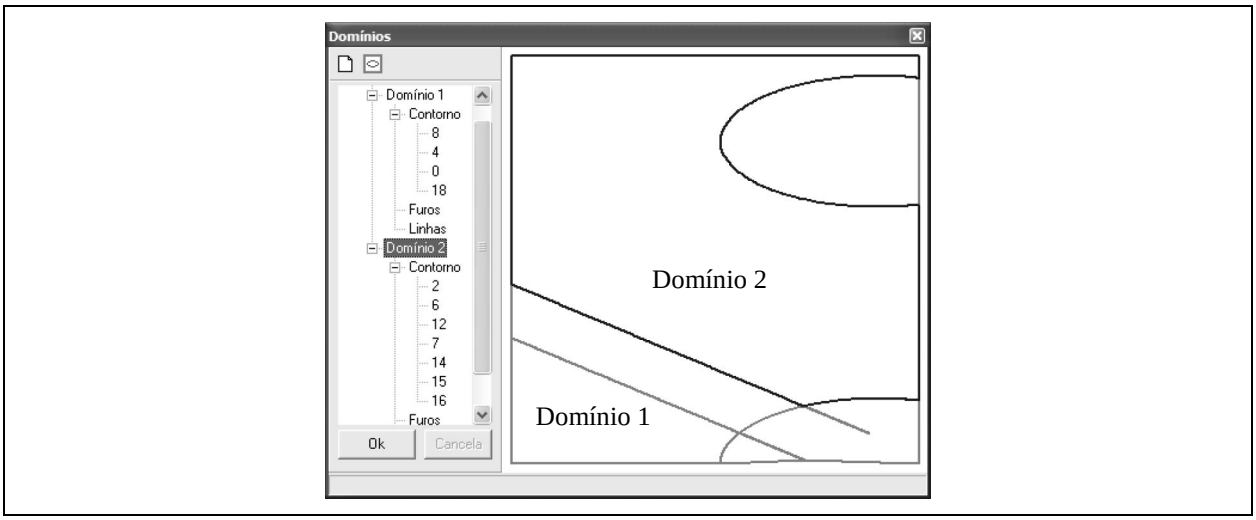


Figura 5.6 – Interface para a definição de sub-domínios válidos para a geração de malha.

5.2.3. Interface para a Definição de Domínios

A definição de sub-domínios, nas superfícies, sobre os quais devem ser geradas as malhas é feita através de uma interface gráfica interativa na qual o usuário define os sub-domínios selecionando diretamente as linhas com o *mouse*. A Figura 5.6 apresenta a interface para a definição de domínios de uma superfície. No lado esquerdo, mostra-se a organização lógica das linhas dos domínios. No lado direito[2], há uma representação gráfica das linhas no espaço paramétrico onde o usuário escolhe, por meio de cliques, as linhas que irão compor contornos, furos e linhas internas.

Em superfícies não recortadas, não há necessidade de definição de domínios, pois todas as superfícies, quando são geradas, são definidas em um domínio fundamental, que corresponde a um quadrado de lado 1 no espaço paramétrico. Em superfícies onde há linhas de interseção e em superfícies planas, onde há recortes definidos pelo usuário, é necessário utilizar esta interface para que as linhas de recorte sejam utilizadas no processo de geração de malha.

[2] Os domínios definidos na Figura 5.6 correspondem ao exemplo da Seção 5.2.3. Trata-se, especificamente, da parte plana superior do modelo de flutuador da plataforma semi-submersível, com as linhas de interseção resultantes da interseção com outras 3 superfícies.

6. APLICAÇÕES

Este Capítulo apresenta uma série de aplicações a fim de demonstrar o potencial de modelamento geométrico e de geração de malha em problemas reais de Engenharia. São apresentados exemplos de modelamento de superfícies complexas compostas de vários recortes, modelos de matrizes de conformação para o METAFOR e geração de campos de sensibilidade.

6.1. GERAÇÃO DE MALHA EM MODELOS COMPOSTOS E COM INTERSEÇÕES

Nesta seção, são apresentados exemplos modelos compostos por várias superfícies, onde é necessário determinar interseções, determinando recortes sobre as superfícies e gerar malhas sobre estes recortes, a fim de demonstrar o potencial dos algoritmos desenvolvidos para gerar modelos complexos para análise por Elementos Finitos. São três os exemplos mostrados, sendo que o primeiro descreve todas as etapas de modelamento.

6.1.1. Modelamento de uma Conexão de Tubulações

Conexões entre tubulações consistem em um problema comum de Engenharia, sobretudo nas industrias: de extração de petróleo, química e petroquímica. No entanto, o modelamento geométrico destas conexões é, quase sempre, crítico, pois envolve a análise de interseção entre duas ou mais superfícies. A geração de malha sobre o modelo geométrico também pode consistir em um problema complexo, pois envolve múltiplos sub-domínios.

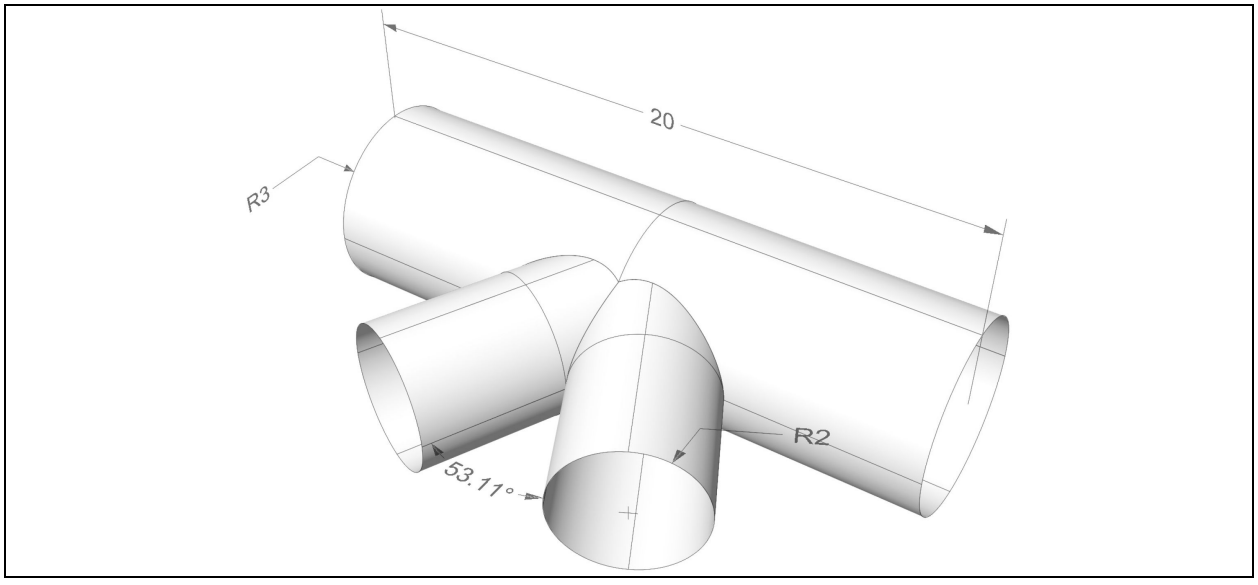


Figura 6.1 – Exemplo de conexão de tubulação.

Apresenta-se um exemplo de modelamento, utilizando o T-CADE, de uma conexão de três tubos em todas as etapas: modelamento das superfícies dos tubos, determinação das interseções entre os três tubos, definição dos sub-domínios onde serão geradas as malhas. A Figura 6.1 mostra o Exemplo citado com as suas dimensões principais. As dimensões não estão expressas em unidades, apenas definem as proporções entre as peças do modelo. Os dois tubos menores têm o mesmo diâmetro e estão dispostos de forma simétrica em relação ao tubo de diâmetro maior. Os dois tubos de menor diâmetro conectam-se com o de maior diâmetro através de uma abertura cuja forma é determinada pela linha de interseção entre os tubos de menor diâmetro e o tubo de maior diâmetro. Além disso, a forma dos dois tubos de menor diâmetro também é determinada pela interseção destes com o tubo de maior diâmetro. A Figura 6.2 mostra uma perspectiva explodida da conexão em estudo, onde é possível verificar a forma das interseções.

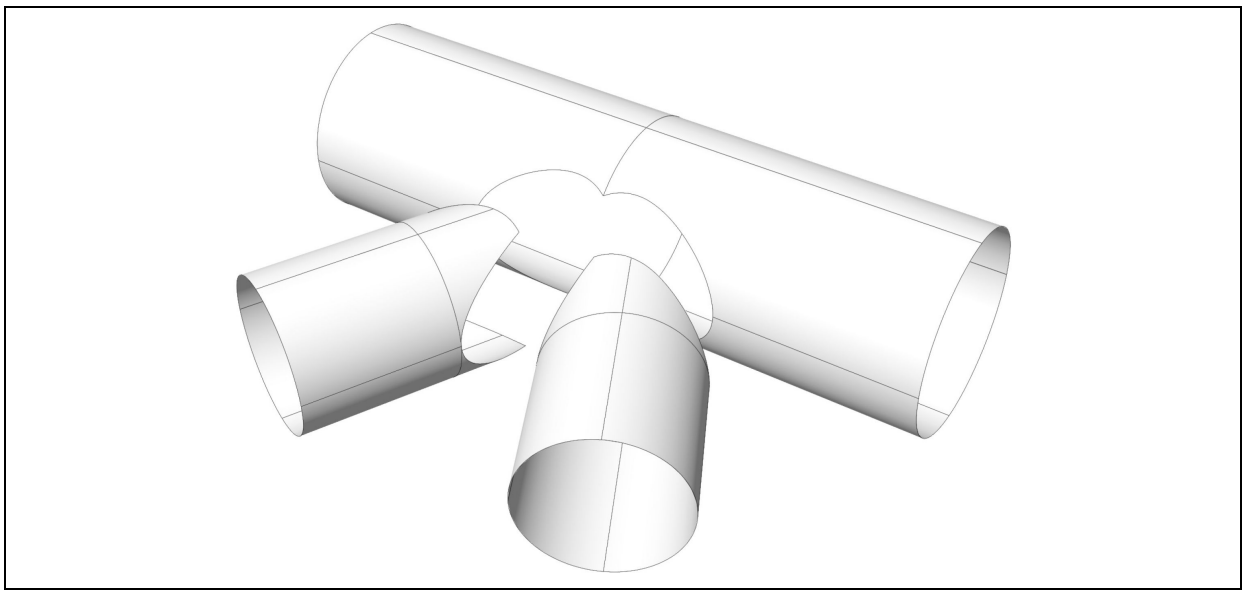
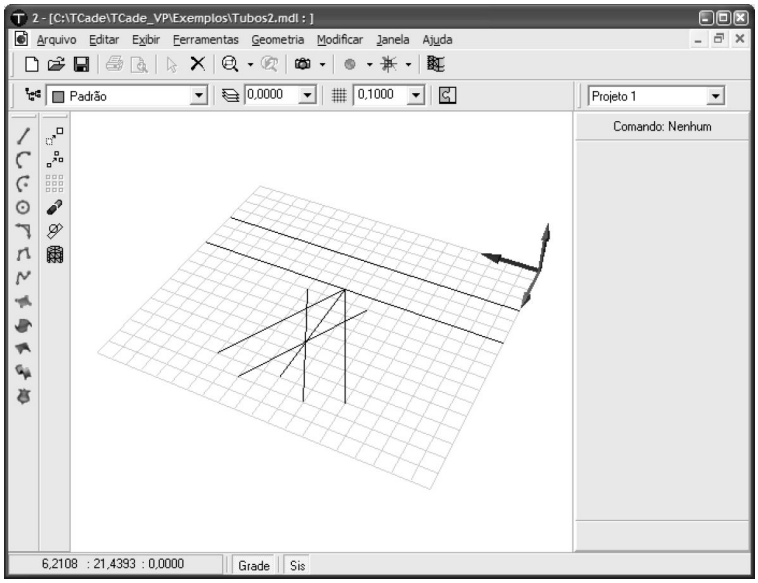


Figura 6.2 – Detalhe da interface da conexão de tubulação.

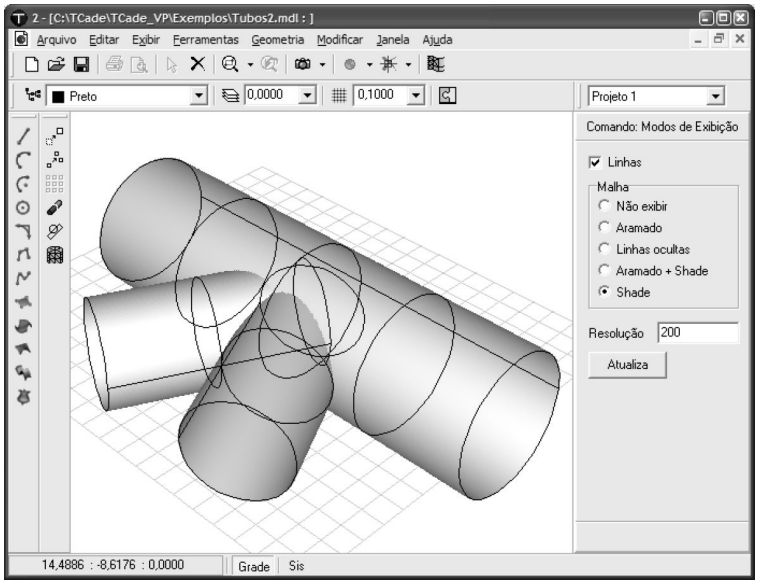
6.1.1.1. Modelamento dos tubos

O modelamento do tubos é a etapa mais simples, pois os tubos são facilmente gerados no T-CADE como superfícies de Revolução. Primeiramente, devem ser construídas as retas Geratrizes os Eixos de revolução dos tubos. A definição das geratrizes poder ser auxiliada pelo uso de uma ferramenta que restringe o movimento do cursor para que o mesmo salte em determinados incrementos de coordenadas. No caso, foi utilizado incremento de 1 unidade para o cursor gráfico. As Geratrizes e os Eixos, que servem de base para a geração dos tubos, são

geradas com linhas Retas paramétricas, através da definição dos pontos de extremidade. Os comprimentos das retas Geratrizes já definem os comprimentos dos tubos. No caso, as Geratrizes dos dois tubos menores são prolongadas até as proximidades do eixo do tubo maior, para garantir que haja interferência entre os mesmos.



(a) Eixos e Geratrizes dos tubos já representados com retas paramétricas



(b) Os três tubos já representados

Figura 6.3 – Interface do T-CADE mostrando etapas da geração dos tubos.

O tubo maior é gerado em duas peças, com a metade do comprimento do tubo. Esta é uma precaução para garantir que as linhas de interseção entre o tubo de maior diâmetro e os tubos de menor diâmetro tenham compatibilidade topológica e, com isto, haja continuidade dos nós das malhas nas interfaces entre as superfícies. A Figura 6.3a apresenta a interface do

programa com as Geratrizes e Eixos já representados sobre o plano *xy* Global.

Após a definição das Geratrizes e Eixos, criam-se as superfícies de Revolução[1] correspondentes a cada tubo. O procedimento é muito simples, bastando clicar sobre a Geratriz e, em seguida, sobre o Eixo. Isto é repetido para a geração das 4 superfícies correspondentes aos 3 tubos. A Figura 5.9b mostra a interface do programa com os 3 tubos representados[2]. É possível observar, pelos contornos dos tubos, que há interferência entre os mesmos.

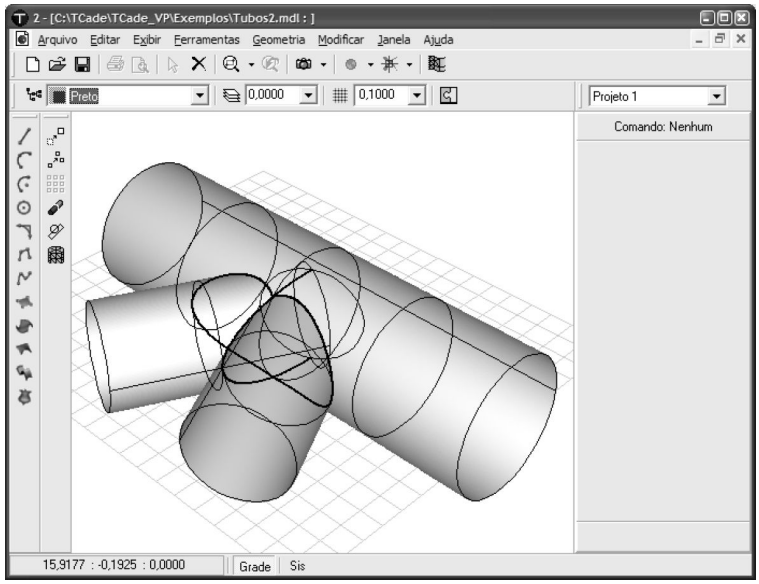


Figura 6.4 – Cálculo das linhas de interseção entre os quatro tubos.

6.1.1.2. Cálculo das interseções entre os tubos

O cálculo de interseções deve ser feito sempre entre duas superfícies. Como existem 4 superfícies que interferem entre si, será necessário utilizar o procedimento de cálculo de interseções 3 vezes. Ao final do processo, existem três conjuntos de linhas de interseção, um sobre cada tubo. O resultado das 3 interseções pode ser observado na Figura 6.4, onde as linhas de interseção estão destacadas. Nesta etapa, as linhas de interseção ainda estão inteiras, pois ainda não foram definidos os sub-domínios sobre os quais serão geradas as malhas, situação em que determinadas porções destas linhas e das superfícies são descartadas.

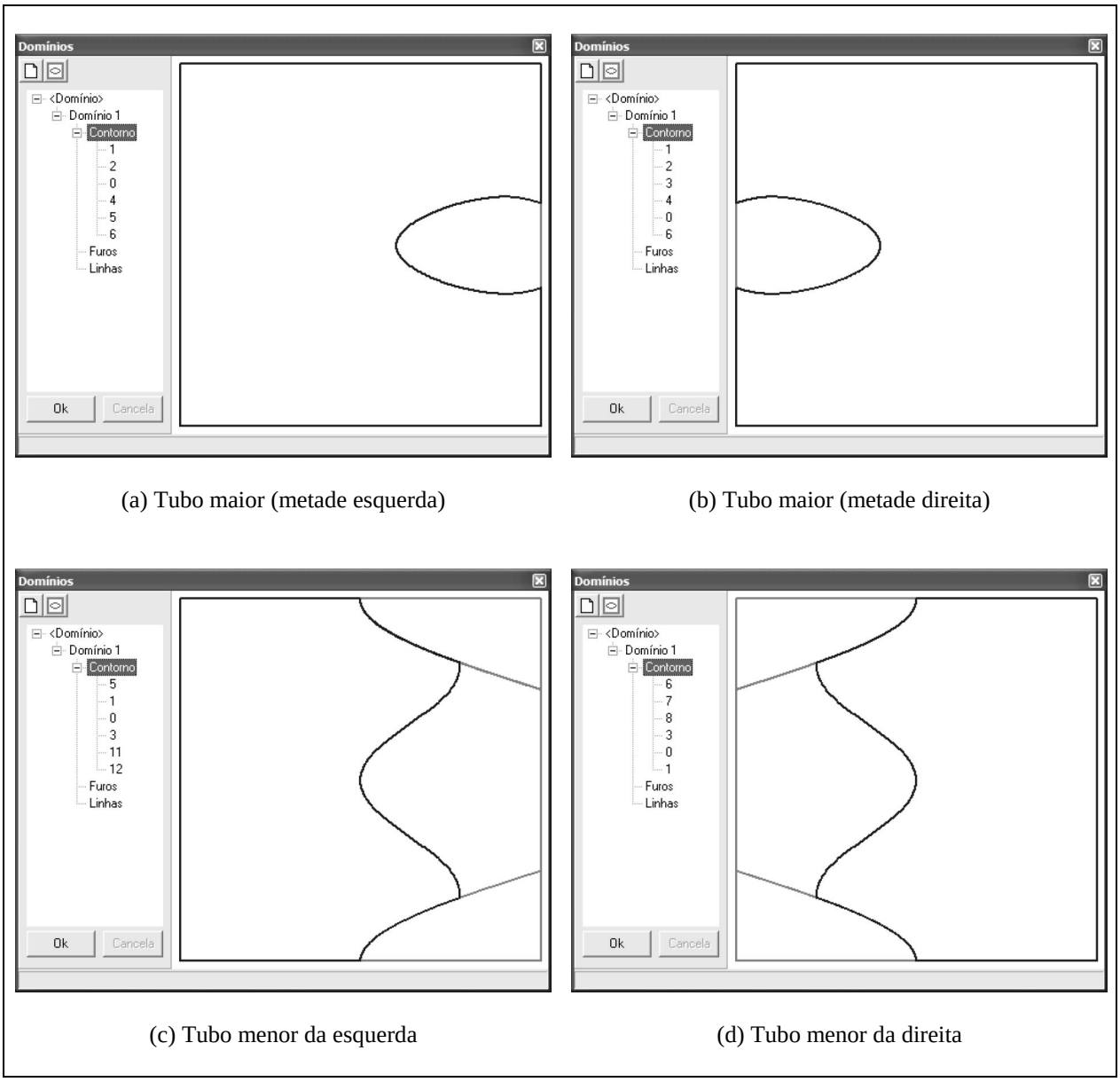
---

[1] Os mesmos tubos poderiam ser criados como superfícies Regradas, a partir das circunferências das extremidades. Em outros Exemplos, como o Permutador de Calor da Seção 6.1.2, a geração das superfícies cilíndricas é feita com o uso de superfícies Regradas.

[2] Sempre que uma superfície é criada, uma malha de visualização com elementos quadriláteros é construída para criar efeitos básicos de visualização, como o sombreamento. A resolução desta malha pode ser alterada pelo usuário, de acordo com o nível de precisão desejado na visualização. As superfícies podem então ser visualizadas com sombreamento através da câmera do T-CADE ou por VRML, como foi mostrado na Seção 5.1.2.

6.1.1.3. Determinação dos sub-domínios válidos

A determinação dos domínios válidos para a geração de malha utiliza a interface descrita da seção 5.1.5. O usuário deve selecionar as linhas que definem os sub-domínios para cada superfície. A Figura 6.5a e 6.5b apresentam a interface com as linhas do domínio do tubo maior em suas duas metades. As linhas escolhidas para definir a área válida (mais escuras) correspondem ao contorno externo do furo, em cada metade, provocado pela penetração dos dois tubos menores individualmente.



(a) Tubo maior (metade esquerda)

(b) Tubo maior (metade direita)

(c) Tubo menor da esquerda

(d) Tubo menor da direita

Figura 6.5 – Definição dos sub-domínios para a geração da malha nas duas metades do tubo maior e nos dois tubos menores.

As Figura 6.5c e 6.5d mostram o domínio paramétrico dos dois tubos menores, que se apresentam simétricos. É possível observar que há interferências entre as linhas resultantes das duas interseções, de forma que há interrupções de algumas destas linhas nestes domínios, o que gera mudanças topológicas nos sub-domínios resultantes. As linhas selecionadas definem a porção de cada tubo que não interpenetra os outros dois. Após a definição dos sub-domínios, a interseção entre os três tubos fica bem delimitada (Fig. 6.6).

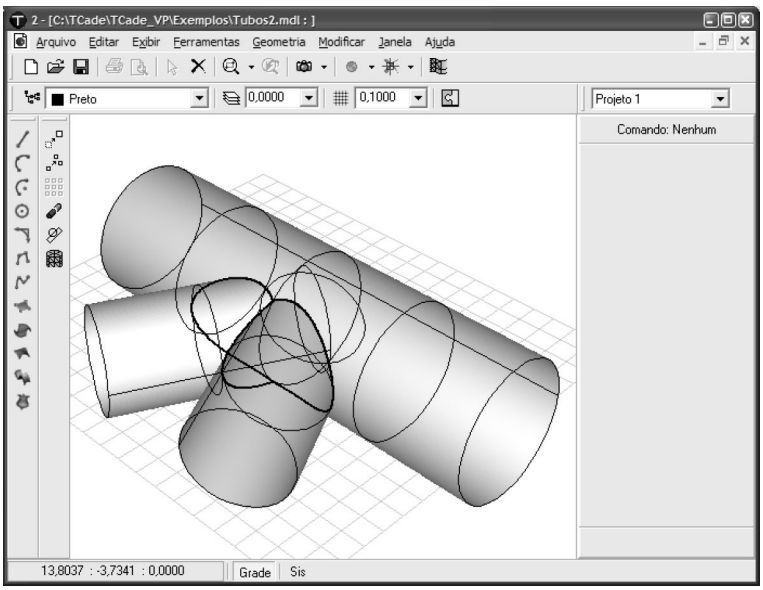
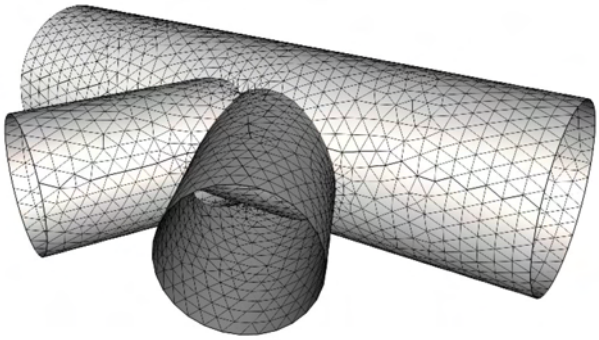


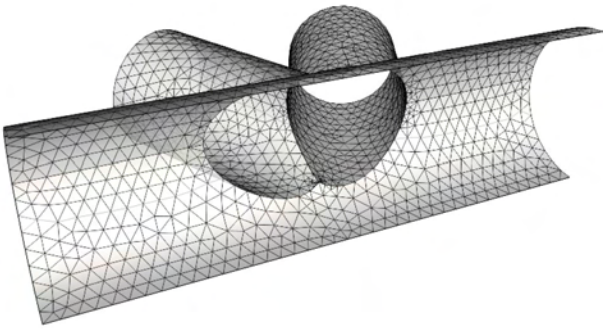
Figura 6.6 – Linhas de interseção em suas porções válidas, após a definição dos sub-domínios.

6.1.1.4. Geração de malha sobre os tubos

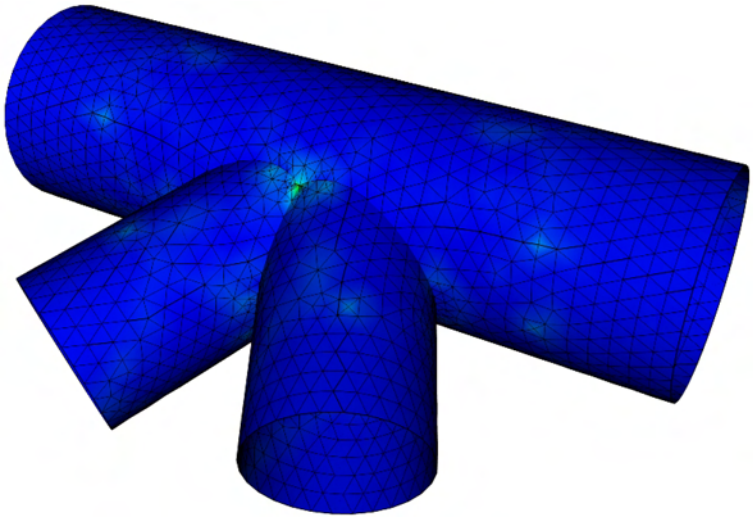
Com os sub-domínios válidos delimitados sobres os tubos, pode-se iniciar o processo de geração de malha. As malhas são geradas individualmente, sobre cada superfície a partir de dois parâmetros básicos: Tamanho de Elemento (*L*) e Tolerância Angular (*A*). Para este exemplo, foi utilizado *L*=0,5 e *A*=10º. A Figura 6.7 mostra detalhadamente a malha resultante sobre as três superfícies. Mostra-se a malha completa (Fig. 6.7a) e o detalhe da continuidade da malha na junção dos três tubos (Fig. 6.7b). Mostra-se também a malha com a distribuição da qualidade α sobre toda a malha (Fig. 6.7c). Pode-se observar que a cor azul, que corresponde a fatores α maiores que 0,9, é predominante em toda a malha, com poucas regiões tendendo para o verde, que corresponderia à faixa entre 0,8 e 0,9, onde a qualidade ainda é boa. A qualidade geral da malha é excelente, o que pode ser traduzido pelo α médio de 0,98. Mostra-se, ainda, uma perspectiva explodida (Fig. 6.7d) dos 3 tubos com a malha gerada.



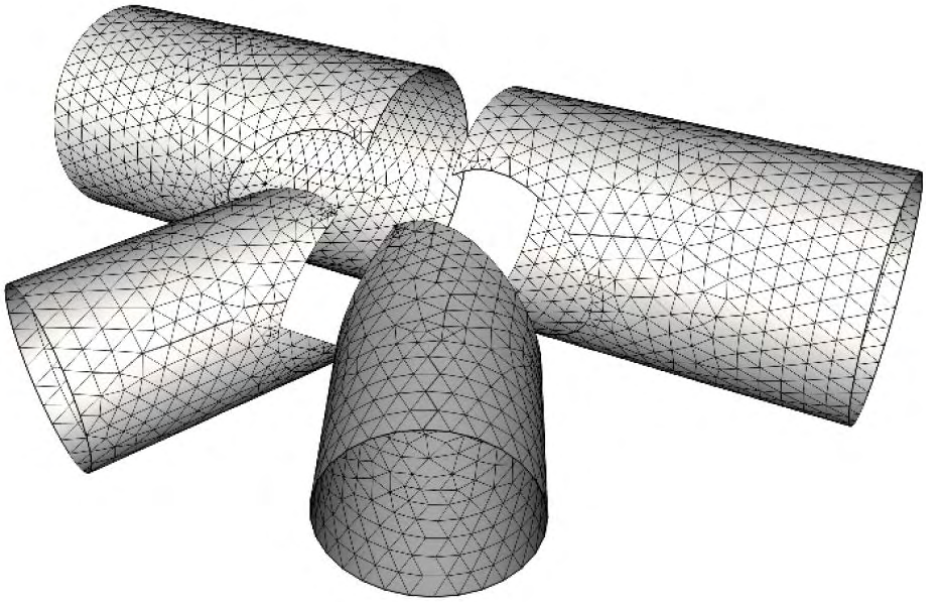
(a) Vista geral da malha



(b) Detalhe da conexão



(c) Distribuição do fator α



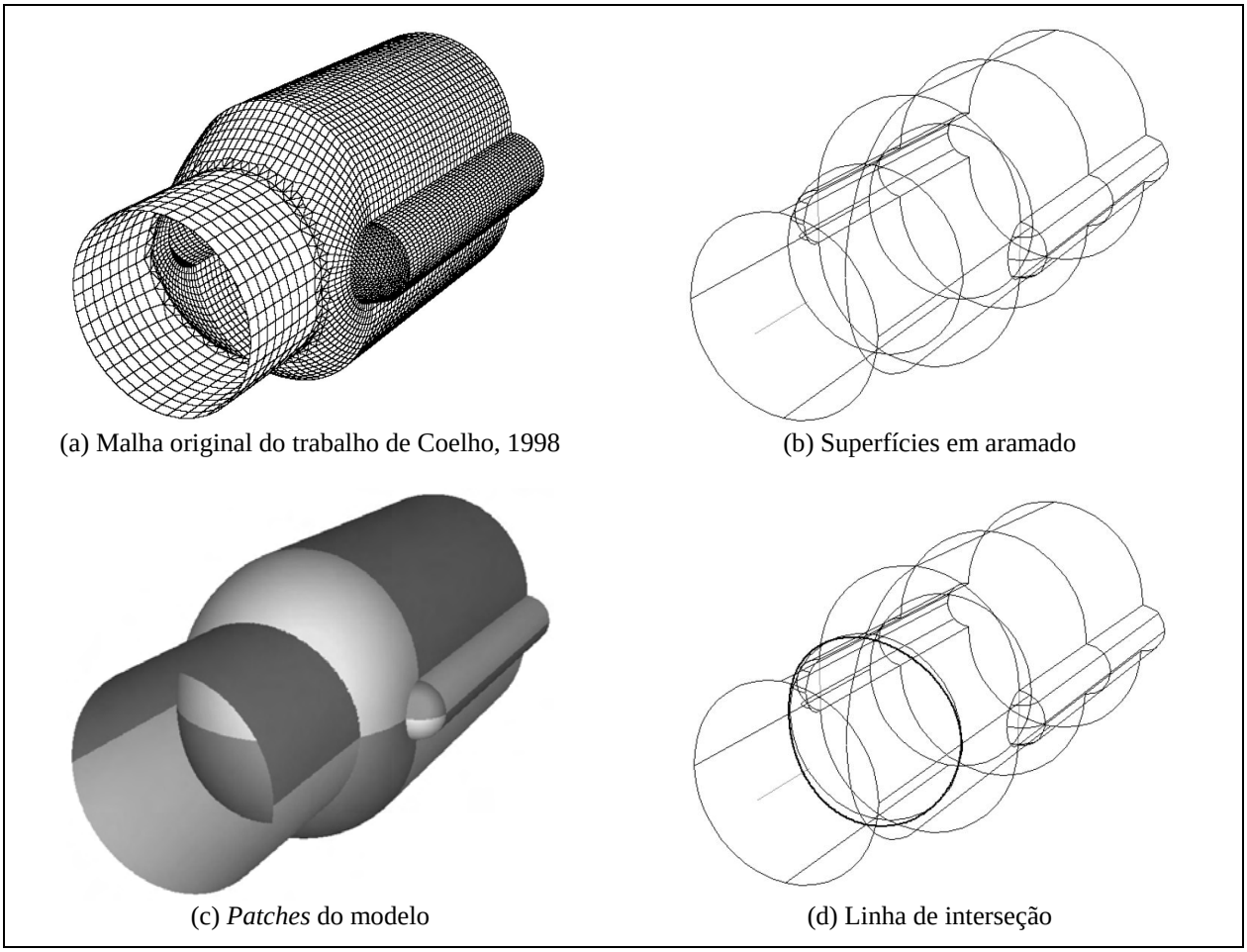
(e) Perspectiva explodida

Figura 6.7 – Malha gerada sobre as superfícies

6.1.2. Trocador de Calor

O próximo exemplo é baseado em um dos exemplos trabalho de Coelho, 1998. Trata-se de uma carcaça de um trocador de calor utilizado em refinarias de petróleo, cuja malha original pode ser vista na Figura 6.8a. No T-CADE, o modelo foi gerado em proporções aproximadas, utilizando superfícies Regradas e superfícies de Revolução a partir de arcos de circunferência (Fig. 6.8c-d). Foi necessário determinar a interseção entre o cilindro central (de diâmetro menor) e a porção esférica, definida com superfície de revolução (Fig. 6.8d).

Removeram-se porções das duas superfícies na definição dos sub-domínios. O resultado pode ser visto na Figura 6.9, que mostra o aspecto final da malha gerada no modelo de trocador de calor. Pode-se observar o interior do modelo revelado pela abertura na superfície esférica obtida pela interseção e definição de sub-domínios. É possível comparar a malha gerada com a do modelo original de Coelho, 1998, na Figura 6.8a.



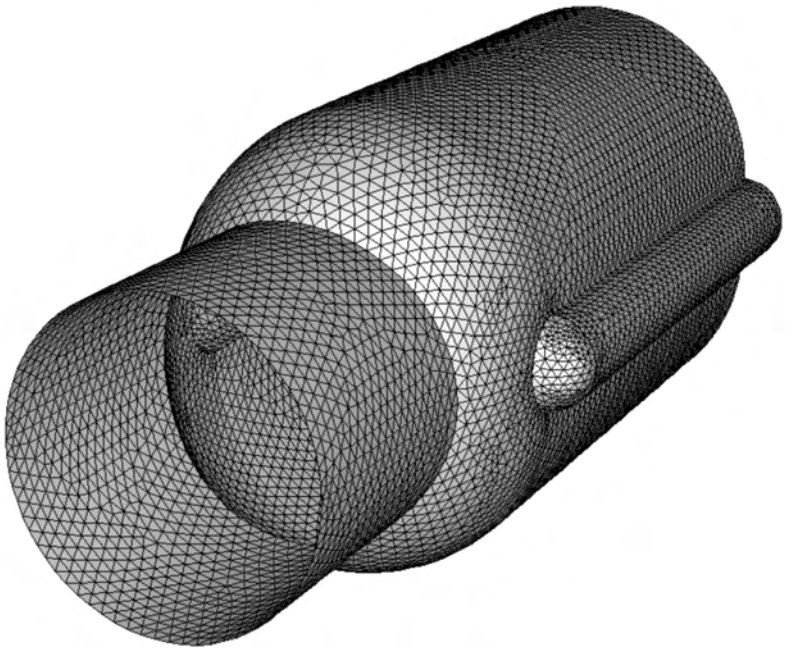
(a) Malha original do trabalho de Coelho, 1998

(b) Superfícies em aramado

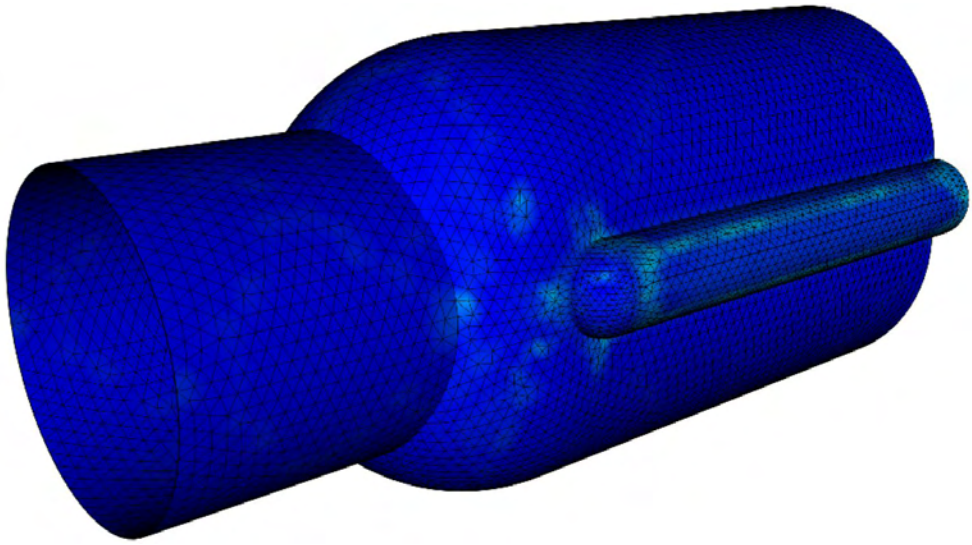
(c) *Patches* do modelo

(d) Linha de interseção

Figura 6.8 – Modelo de trocador de calor.



(a) Malha final com a porção central da superfície esférica e a porção posterior do cilindro central removidas



(b) Distribuição do fator de qualidade a na malha do Trocador de Calor

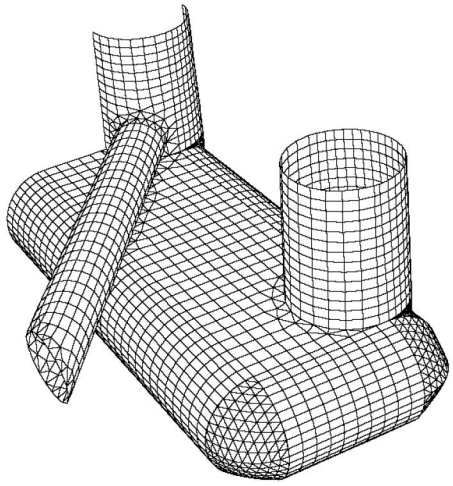
Figura 6.9 – Malha final do trocador de calor.

Para este exemplo, foram gerados 8038 Nós e 15752 Elementos, utilizando-se um *L* de 0,3 e *A* de 10º. A dimensão máxima de peça é da ordem de 20 unidades. O fator de qualidade α médio foi de 0,97 para esta malha e a distribuição do fator α pode ser observada diretamente sobre a malha na Figura 6.9b. Valores de α=1 ou muito próximo disto são as regiões em azul. Em algumas regiões há tons mais claros de azul, mas no aspecto geral pode-se dizer que a qualidade geométrica da malha é muito boa.

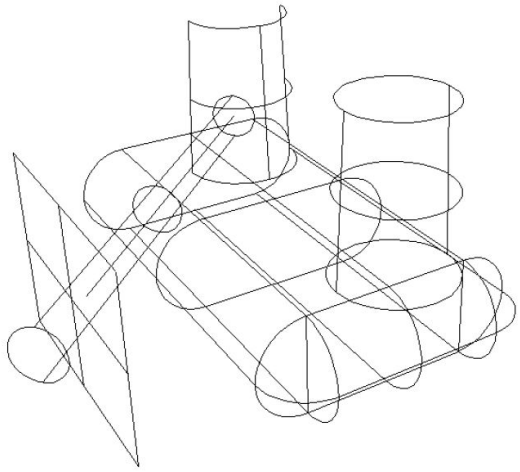
6.1.3. Plataforma semi-submersível

Este exemplo também é baseado em Coelho, 1998. Trata-se de um modelo de um flutuador de uma plataforma de petróleo semi-submersível[3]. O modelo, que corresponde a ¼ da peça real devido à dupla simetria, foi criado com 9 *patches* de superfícies Regradas e 2 *patches* de superfícies de Revolução (os arredondamentos dos cantos).

A Figura 6.10a mostra o modelo original [Coelho, 1998] e as Figuras 6.10b e 6.11a mostram as etapas principais do modelamento com o T-CADE. Primeiramente, são criadas as superfícies a partir de linhas de bordo (Fig. 6.10b), que são, basicamente, circunferências, arcos de circunferência e segmentos de retas. Em seguida, são determinadas as interseções entre os pares de superfícies, utilizando o algoritmo descrito no Capítulo 3.



(a) Malha original do trabalho de Coelho, 1998



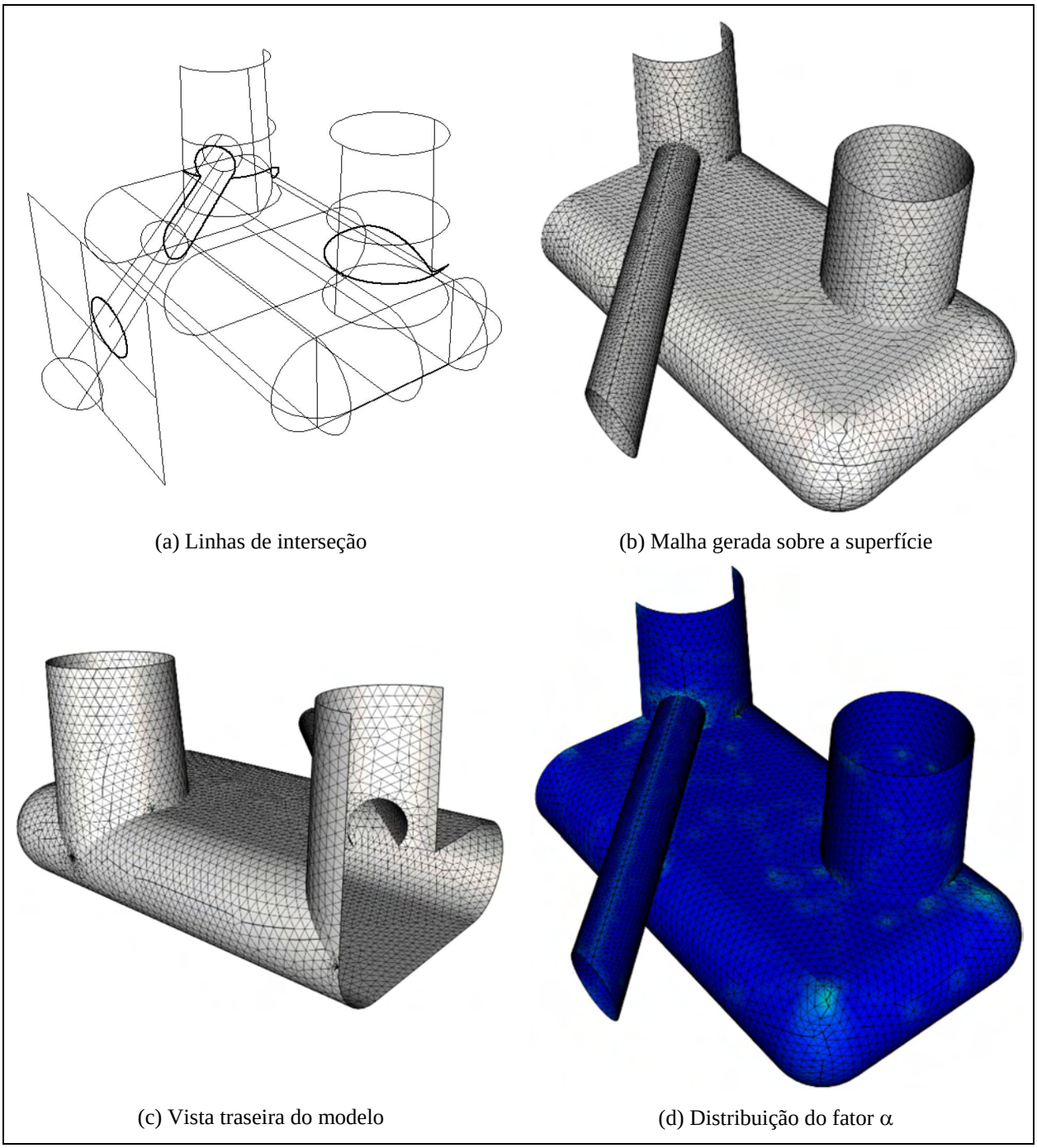
(a) Superfícies com as linhas de bordo

Figura 6.10 – Modelamento geométrico do flutuador.

A Figura 6.11a apresenta as linhas de interseção (mais grossas) entre todas as superfícies do modelo. Na próxima etapa, determinam-se os sub-domínios das superfícies sobre os quais será gerada a malha. Finalmente, gera-se a malha sobre todos os sub-domínios criados. A Figura 6.11b apresenta a malha final. A Figura 6.11c apresenta a vista por trás do modelo onde é possível ver detalhes da geometria, como os recortes. Este é um exemplo que mostra bem o potencial das técnicas implementadas nesta Tese, pois há superfícies com múltiplas interseções e com mais de um sub-domínio onde deve ser gerada a malha.

[3] O trabalho de Coelho refere-se à plataforma P-27 da Petrobrás, mas como o seu trabalho não publica as dimensões, foi feito aqui um modelamento por semelhança, que não é exatamente o mesmo modelo.

Nas superfícies com múltiplas interseções, o algoritmo que determina as interseções das linhas de interseção, segmentando-as, funcionou perfeitamente e permitiu a geração adequada da malha sobre o modelo.



(a) Linhas de interseção

(b) Malha gerada sobre a superfície

(c) Vista traseira do modelo

(d) Distribuição do fator α

Figura 6.11 – Modelo de flutuador com a malha gerada.

A malha gerada com, L=0,5 e A=10º, resultou em 8056 Nós e 15723 Elementos. A qualidade geral da malha é muito boa, como pode ser visto na Figura 6.11d, que mostra a distribuição do fator α, com poucas regiões com α<0,9 (em verde) e pelo α médio que chegou a

0,97. O que demonstra que é possível utilizar o gerador e o modelador, desenvolvidos e implementados no T-CADE, para gerar malhas em modelos complexos com a união de várias superfícies recortadas. Portanto, o potencial de uso como pré-processador para códigos de análise de Elementos Finitos é bastante grande, o que pode ser implementado com a compatibilização dos formatos dos arquivos de dados.

6.2. APLICAÇÕES PARA O METAFOR

Nesta seção, é apresentada uma breve descrição do código de análise METAFOR e dois tipos de aplicações do T-CADE específicas para o METAFOR: modelamento e geração de matrizes de conformação mecânica e cálculo de sensibilidades para aplicações em otimização de pré-forma.

6.2.1. O METAFOR

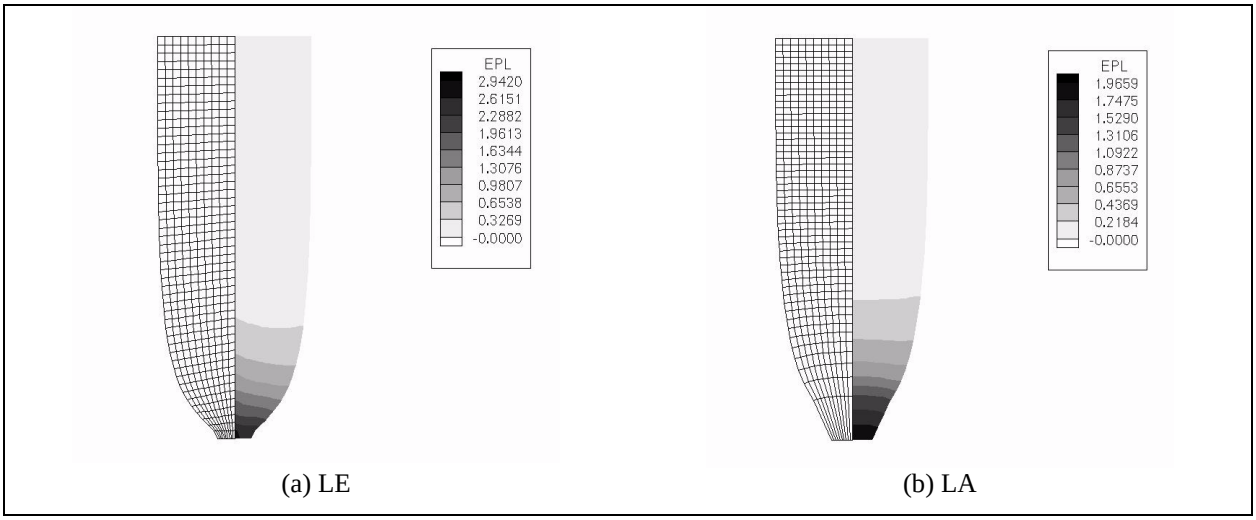
O programa METAFOR foi desenvolvido inicialmente por Ponthot, 1995, e consiste em um módulo de análise de problemas não lineares através do método dos Elementos Finitos. O sistema de equações de equilíbrio pode ser resolvido através de métodos implícitos (Newton-Raphson, Newmark), ou explícitos (método das diferenças centrais).

Diferentes relações constitutivas podem ser empregadas. Para materiais metálicos emprega-se uma relação do tipo hipo-elástica, na qual uma decomposição aditiva das taxas de deformação elástica e plástica são empregadas. As deformações plásticas são integradas empregando um algoritmo de retorno radial. A garantia da objetividade de tal processo de integração em grandes deformações é conseguida pelo fato de toda a integração ser feita em eixos chamados "co-rotacionais", onde apenas deformação pura se desenvolve, livre de todo efeito de rotação de corpo rígido. Ainda no caso de materiais metálicos, relações plásticas do tipo independentes da pressão são empregadas (von Mises) bem como relações que dependem da pressão. Este último tipo de relação é necessário pois em processos de grandes deformações, uma considerável deterioração do material pode ocorrer devido, fundamentalmente, à formação e coalescimento de micro vazios que conduzem a trincas macroscópicas. Tal relação constitutiva permite a utilização avançada dos materiais, ou seja, pode-se empregar o material até o seu limite, sabendo que um certo nível de dano não vai por em risco a resistência global da estrutura. Em conformação mecânica, permite prever a qualidade final do produto e os defeitos de fabricação, bem como otimizar processos de corte ou usinagem. Para materiais tais como

polímeros e borrachas, relações do tipo hiper-elásticas são empregadas.

Problemas de grandes deformações são, normalmente, acompanhados de uma forte distorção de malha que prejudicam a precisão do método dos Elementos Finitos, quando formulações do tipo Lagrangeanas Atualizadas (LA) são empregadas. O programa trata o problema basicamente de duas formas diferentes:

- Emprego de uma formulação Lagrangeana-Euleriana (LE): Desvincula-se o movimento da malha da matéria, de modo que é possível ter-se uma malha ideal (Fig. 6.12a), sem distorção. O método consiste em, a cada passo de tempo, solucionar o problema de forma LA. Uma vez atingido o equilíbrio, uma nova malha, não distorcida, é criada. Faz-se então uma transferência de dados da malha velha (ou matéria) para a malha nova. Para o novo passo de tempo esta malha nova está ligada a matéria, tendo reinicio o processo.



(a) LE (b) LA

Figura 6.12 – Estricção de uma barra axissimétrica.

- Emprego de remalhamento clássico interativo [Bittencourt *et al.*, 1997]: Mesmo as formulações LE são incapazes de evitar em certos casos a formação de elementos finitos muito distorcidos. Uma forma drástica mas definitiva para eliminação da distorção dos elementos é a redefinição de uma malha totalmente nova, uma vez atingido certos níveis de distorção da malha (Fig. 6.12b),. Um dos problema neste caso é a passagem dos dados da malha velha (distorcida) para a outra nova (não distorcida). Normalmente é empregada uma técnica de inversão paramétrica para localizar os nós da malha nova no interior de um elemento da malha antiga. Os dados nos nós da nova malha são obtidos então por interpolação.

O programa original METAFOR sofreu modificações, descritas em Bittencourt e Creus, 1998, com a implementação de Elementos tridimensionais e contato e atrito

tridimensional. Vale lembrar que praticamente todo tipo de transmissão de carga entre partes de uma estrutura ocorre através do fenômeno do contato e atrito. Portanto, a correta simulação destes fenômenos permite uma maior fidelidade da representação da situação física real. O estudo do contato e do atrito torna-se fundamental quando o objetivo é simular problemas de conformação mecânica ou determinar a resistência de estruturas ao impacto, quando pode ocorrer auto-contato. Além disto, é a única forma de determinar tensões e deformações quando dois corpos flexíveis entram em contato. O processo de integração das forças de atrito segue metodologia similar àquela, descrita acima, para deformações plásticas. Diferentes formas geométricas de matrizes rígidas de contato podem ser empregadas pelo programa.

6.2.2. Geração de Matrizes de Conformação

Um dos objetivos iniciais deste trabalho é a modelamento geométrico de matrizes de conformação mecânica para uso no programa METAFOR. Neste caso, os modelos devem conter somente superfícies Planas, Regradas, Coons e Esféricas, que são os tipos de superfícies disponíveis no METAFOR. O programa permite a utilização de dois tipos de matrizes: rígidas e flexíveis. As matrizes rígidas são definidas como superfícies paramétricas. As matrizes flexíveis são definidas por malhas. A entrada de dados geométricos de matrizes de conformação do METAFOR é, originalmente, feita diretamente em arquivo de texto, o que não permite nenhuma interatividade, nem uma verificação da precisão da geometria gerada através de algum tipo de visualização. Os algoritmos de modelamento paramétrico e de geração de malha implementados no T-CADE, permitem a geração destes dois tipos de matrizes de forma precisa e interativa. Os dois exemplos apresentados a seguir, utilizam matrizes rígidas.

6.2.2.1. Modelo de Matriz para Hidroconformação de uma Peça em Forma de T

Nos últimos anos, a indústria automobilística tem procurado o desenvolvimento de processos de fabricação capazes de produzir peças mais leves e resistentes e com menor variação dimensional, reduzindo custos de produção, principalmente das ferramentas [Button e Bortolussi, 2001]. Neste contexto, o processo de hidroconformação de tubos vem sendo utilizado para a produção de peças da estrutura da carroceria e de componentes agregados, como suporte do motor e eixos dianteiros e traseiros. Este processo utiliza a combinação de deslocamento de pistão e pressão para a conformação das peças até a sua forma final (Fig. 6.13).

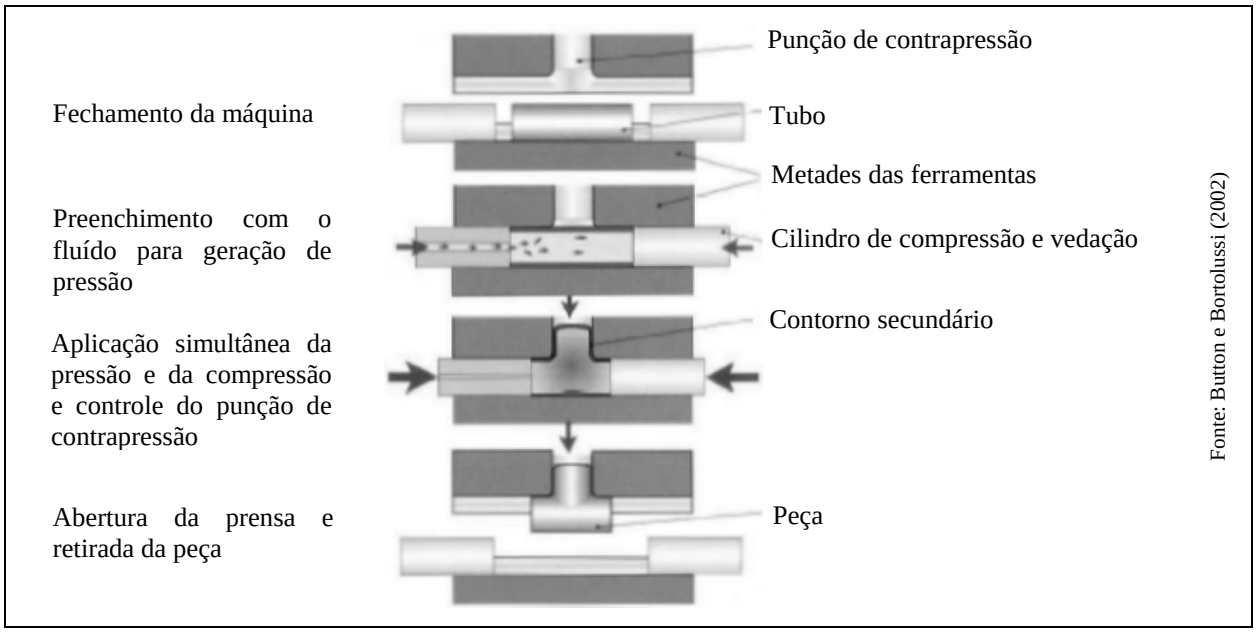
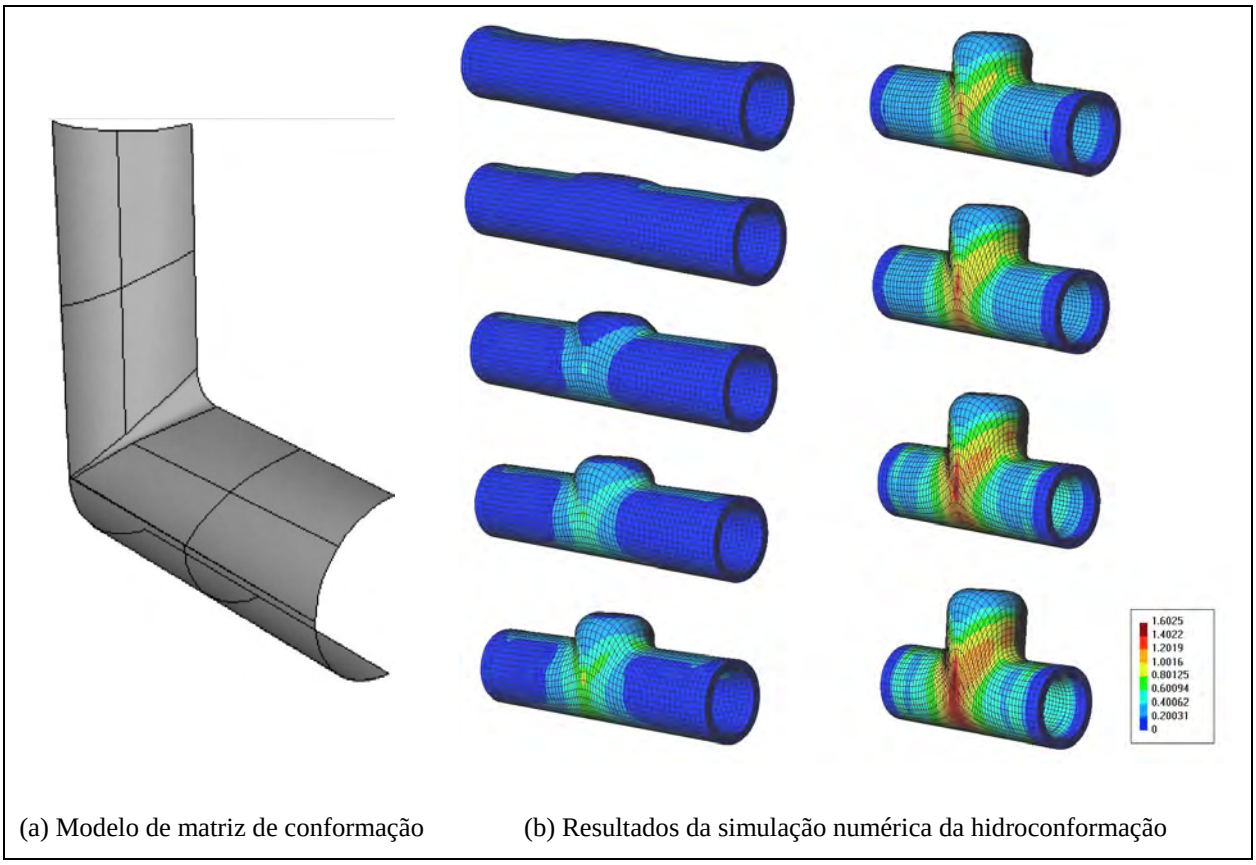


Figura 6.13 – Esquema de hidroconformação de uma peça em T.

Este exemplo foi utilizado para testar a capacidade do METAFOR em simular o processo de hidroconformação. Trata-se de um exemplo, utilizado no trabalho de Aymone *et al*., 2000, onde uma peça na forma de T é gerada a partir de um tubo cilíndrico por hidroconformação. A matriz de conformação utilizada é um casca em forma de T. Foi modelado apenas ¼ da geometria original devido à simetria do problema. As superfícies utilizadas foram: Regradas, nas porções cilíndricas, e Coons, na concordância dos dois tubos. A Figura 6.14a a matriz de conformação modelada com o T-CADE.

As superfícies Regradas, em um total de 3, foram geradas utilizando os arcos como curvas de bordo. As curvas próximas à superfície de concordância foram aproximadas por arcos, pois o METAFOR não tem outro tipo de curva implementado. A superfície de concordância é Coons com domínio quadrilátero, embora pareça triangular devido a um dos lados ser bem pequeno. Foi utilizada, ainda, uma superfície plana tipo Quadrilátero que, para efeitos de análise, é uma matriz móvel que controla a forma do tubo que sobe verticalmente durante o processo de conformação. A entrada de dados geométricos de matrizes de conformação do METAFOR é, originalmente, feita diretamente em arquivo de texto, o que não permite nenhuma interatividade, nem uma verificação da precisão da geometria gerada através de algum tipo de visualização.

Com a utilização do T-CADE, foi possível modelar a matriz rapidamente e com a precisão apropriada, possibilitando a simulação numérica com o METAFOR deste problema de hidroconformação. A Figura 6.14b apresenta o resultado da simulação numérica da hidroconformação. As isofaixas correspondem às taxas de deformação plástica sobre a malha em vários instantes da simulação.



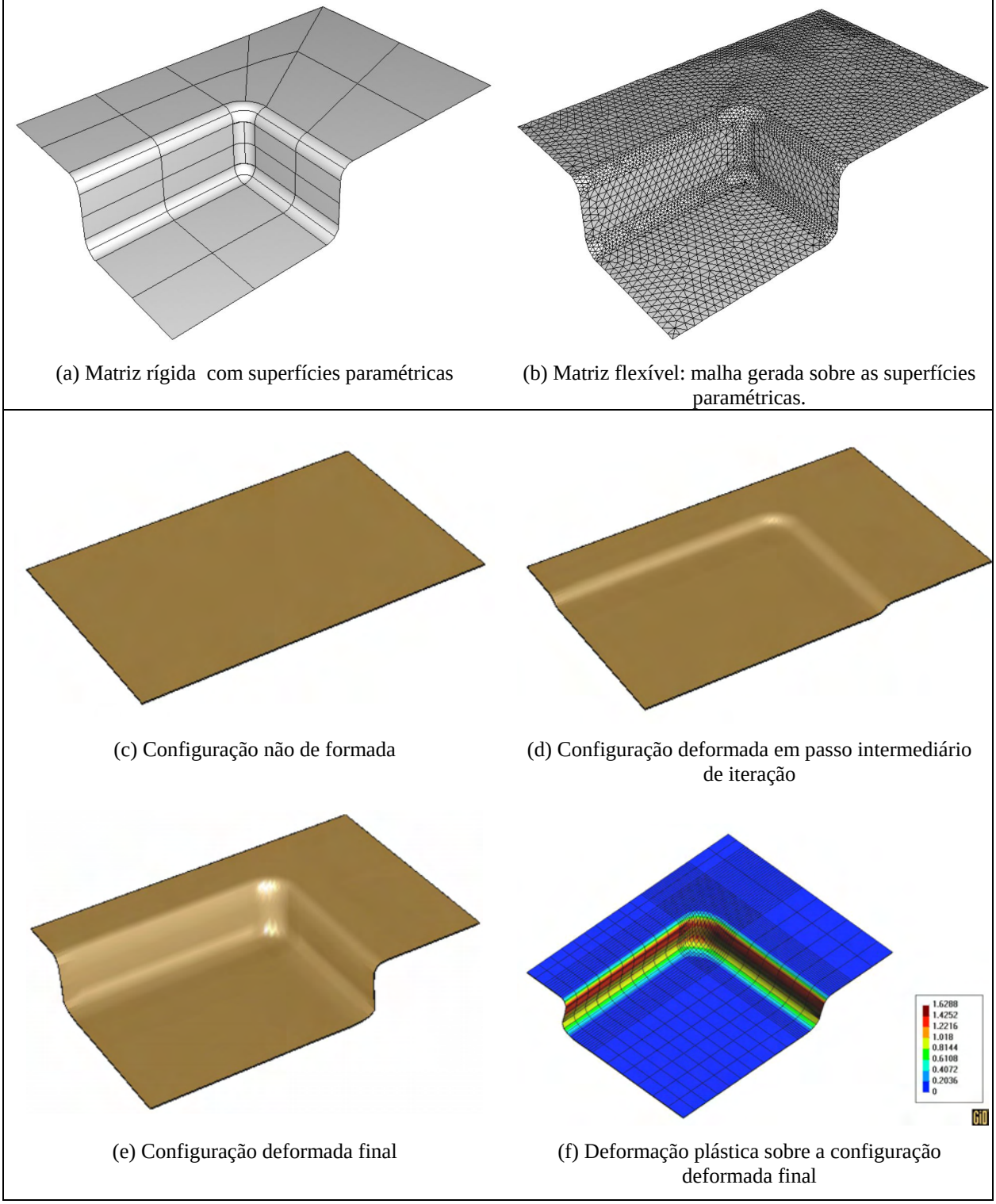
(a) Modelo de matriz de conformação (b) Resultados da simulação numérica da hidroconformação

Figura 6.14 – Exemplo de hidroconformação.

6.2.2.2. Modelo de Matriz para Conformação de Pia Metálica

Neste exemplo, o objetivo era testar a capacidade de modelamento do T-CADE e sua compatibilidade com os padrões do METAFOR. Pretendia-se, também, testar o próprio METAFOR em sua capacidade de análise em problemas de hidroconformação. Trata-se de uma pia metálica, da qual foi modelado apenas ¼, por questões de simetria. A tarefa com o T-CADE foi a de modelar, especificamente, a matriz de conformação.

A matriz de conformação com *patches* de superfícies Regradas (11) e Coons (2) para a simulação de conformação de uma pia metálica no METAFOR. Novamente, simulou-se o processo de hidroconformação. A Figura 6.15a apresenta a matriz rígida gerada para análise no METAFOR, com todos *patches* destacados. A Figura 6.15b apresenta uma malha, gerada sobra as superfícies da Figura 5.20a, que poderia ser utilizada como matriz flexível em substituição à matriz rígida. No entanto, para ser usada com o METAFOR, as matrizes flexíveis devem ser constituídas de Elementos volumétricos (sólidos), o que não está implementado no T-CADE, mas que seria uma tarefa relativamente simples partindo-se da situação atual. Esta será uma das implementações que deverão ser realizadas nos próximos trabalhos de pesquisa.



(a) Matriz rígida com superfícies paramétricas

(b) Matriz flexível: malha gerada sobre as superfícies paramétricas.

(c) Configuração não de formada

(d) Configuração deformada em passo intermediário de iteração

(e) Configuração deformada final

(f) Deformação plástica sobre a configuração deformada final

Figura 6.15 – Simulação do processo de conformação no METAFOR mostrando a matriz de conformação e os diferentes estágios da malha.

A matriz foi modelada com sucesso e a simulação numérica foi realizada pelo METAFOR, como mostra a Figura 6.15(c-f)[4], que apresenta o resultado da simulação com 3

[4] As imagens das Figuras 6.14b e 6.15c-f foram obtidas com o auxílio de uma versão acadêmica do programa GiD.

imagens (Fig. 6.15c-e) de instantes diferentes do processo de conformação, onde a malha aparece com diferentes estágios de deformação. A Figura 6.15f apresenta uma visualização da distribuição das deformações plásticas sobre a malha já no final do processo.

6.2.3. Cálculo de Campos de Velocidades em Processos de Otimização

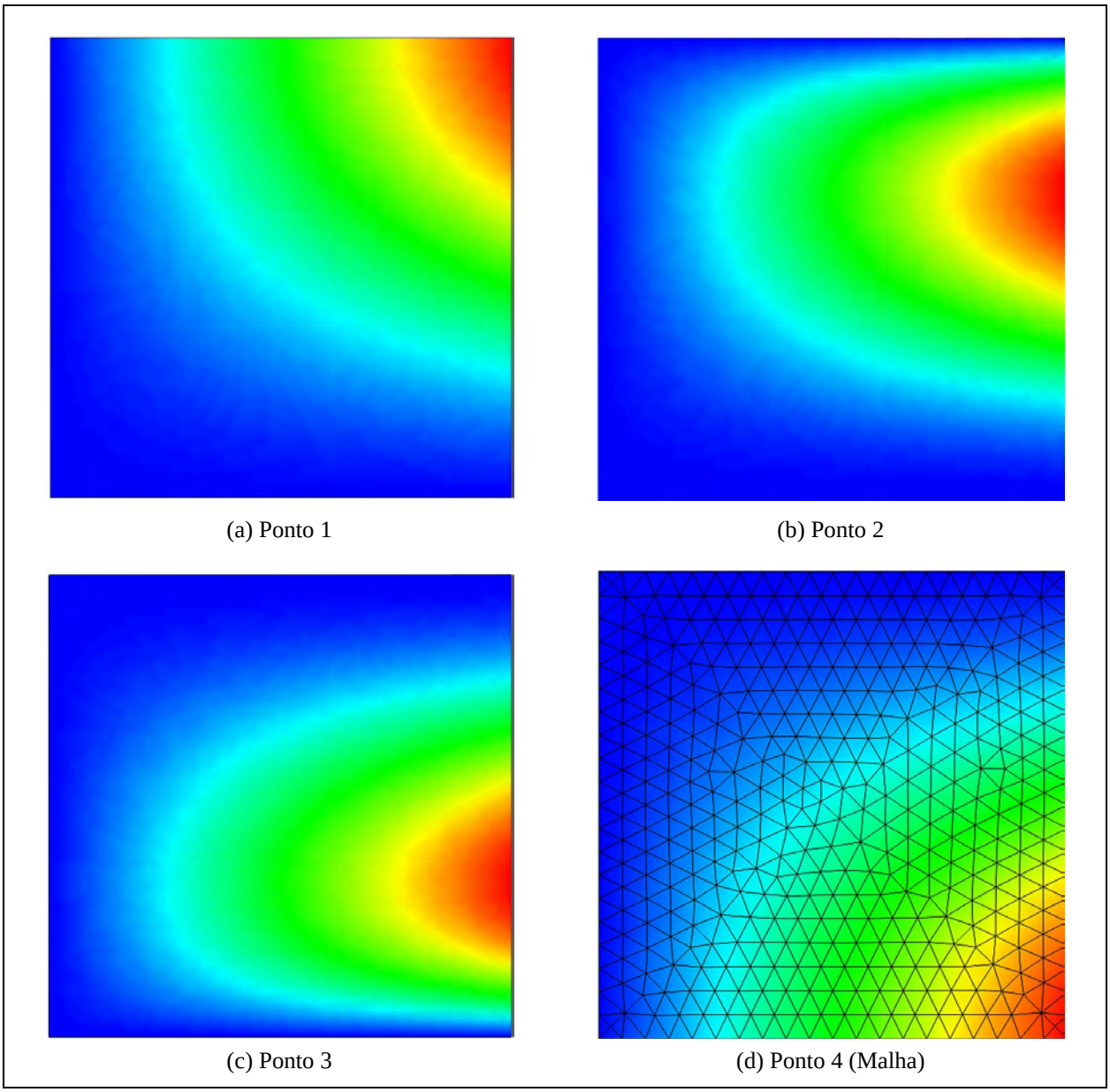
Os métodos de otimização de forma, necessitam calcular os chamados campos de sensibilidade, que são as taxas de variação de posição de pontos do domínio em relação a perturbações no contorno. Estas taxas são as derivadas da função posição nestes pontos em relação às variáveis de projeto, que no caso de otimização de forma, são as que determinam a geometria do contorno. Quando o domínio a ser analisado possui uma representação paramétrica, o cálculo das derivadas pode ser feito de forma analítica diretamente a partir das equações paramétricas. Caso contrário, é necessário utilizar algum processo numérico, como diferenças finitas e, ainda, utilizar técnicas especiais para determinar os campos de velocidades no interior do domínio. Estes processos são custosos e envolvem a solução de um sistema de equações lineares da ordem do número de graus de liberdade de toda a malha para cada variável de projeto.

Rojas, 2003, implementou um código de otimização de pré-forma no METAFOR que determina a sensibilidade de forma analítica no contorno, pois sua parametrização é conhecida, mas utiliza perturbações finitas no contorno e suavização Laplaciana no interior da malha, para determinar os campos de velocidades nos pontos da malha, pois suas coordenadas paramétricas não são conhecidas.

É neste contexto que o gerador de malha baseado em superfícies paramétricas implementado no T-CADE pode ser utilizado. Uma vez que as malhas são geradas diretamente sobre as superfícies paramétricas, é possível determinar os campos de velocidades de forma analítica. Para verificar a potencialidade de uso do T-CADE neste tipo de análise, determinou-se o campo de velocidades para uma superfície Coons linear considerando as curvas de bordo 3 Retas e uma Spline (Catmull-Rom). O problema é relativamente simples, pois basta derivar a equação da superfície Coons (Eq. 2.15) em relação aos pontos de controle da Spline. Como a superfície Coons é linear e apenas um dos termos da Eq. 2.15 multiplica a Eq. 2.7, as derivadas vão ficar reduzidas ao coeficiente correspondente da Eq. 2.15 que multiplica o termo da Eq. 2.7, correspondente ao ponto de controle analisado. Considerando que na parametrização a Spline corresponde à curva de bordo C_1, a derivada em relação a um dos pontos de controle ficaria reduzida a:

$$\frac{\partial}{\partial X_i}\mathbf{S}(u,v) = (1-v).\frac{\partial}{\partial X_i}\mathbf{C}_1(u) \tag{6.1}$$

onde X_i é o ponto de controle da Spline onde se pretende analisar a sensibilidade. A derivada de C_1 ficará reduzida ao coeficiente do ponto de controle analisado (ver Eq. 2.7).



(a) Ponto 1 (b) Ponto 2

(c) Ponto 3 (d) Ponto 4 (Malha)

Figura 6.16 – Campos de velocidades obtidos de forma analítica em superfície.

Para verificar a validade da Eq. 6.1, foi analisada uma superfície Coons quadrada, com a uma Spline com 4 pontos de controle sobre a curva. Foram calculados os campos de velocidades referentes a estes 4 pontos e os resultados são mostrados na Figura 6.16. Pode-se observar que os campos resultaram extremamente bem definidos e, como são obtidos de forma

analítica, independem da configuração da malha. A Figura 6.16d apresenta, além do campo de velocidades, a configuração da malha utilizada.

Este resultados[5] comprovam o potencial de utilização do gerador do T-CADE em processos de otimização. No entanto, para uso com o METAFOR, há a necessidade de malhas com Elementos quadriláteros, pois os Elementos triangulares criam um problema de *locking* da malha em processos de grandes deformações, travando a análise. De qualquer forma, existem vários algoritmos que permitem a transformação de uma malha triangular em uma malha de quadriláteros, sem a necessidade de uma nova geração, e que poderiam vir a ser implementados a fim de explorar este potencial.

[5] As imagens foram geradas com a visualização em VRML descrita na Seção 5.1.1.

7. CONCLUSÕES

Foram desenvolvidos e implementados algoritmos para o modelamento paramétrico de superfícies, incluindo um algoritmo para determinação de linhas de interseção entre superfícies e para a geração de malhas em superfícies curvas e recortadas. Os algoritmos foram implementados em um ambiente gráfico, o T-CADE, desenvolvido como parte deste trabalho, onde foi possível testar todas as técnicas empregadas. A seguir são destacadas as principais contribuições e conclusões desta Tese.

1) As linhas e superfícies paramétricas implementadas mostraram-se eficientes para a modelagem de objetos complexos, como é possível observar nos vários exemplos apresentados. É possível representar uma grande variedade de formas, incluindo as que ocorrem em problemas de Engenharia, que é o principal objetivo deste trabalho.

2) O algoritmo de cálculo de interseções entre superfícies paramétricas é uma das principais contribuições desta tese. O algoritmo emprega uma técnica de subdivisão adaptativa em função das curvaturas locais da superfície utilizando um esquema de *quadtree*. Os trechos onde há potencial de ocorrerem interseções são determinados pela existência de interferências entre os volumes envolventes (*Bounding Boxes*). Uma técnica de pontos amostrais foi introduzida para a computação destes volumes, o que tornou a etapa de subdivisão bastante rápida. O processo de subdivisão é encerrado quando um determina tolerância angular, que corresponde a uma planura (*Flatness*) local mínima aceitável. Calcula-se, então, uma aproximação inicial das linhas de interseção no espaço 3D, utilizando uma técnica que subdivide as células remanescentes em triângulos. A aproximação é determinada pela interseção entre os triângulos das duas superfícies, gerando um conjunto de pontos e segmentos. São introduzidas restrições topológicas que garantem a continuidade das linhas de interseção. Em seguida, os resultados são refinados por um algoritmo que projeta os pontos de interseção sobre os espaços paramétricos das duas superfícies. Os segmentos são reordenados e as linhas de interseção são parametrizadas em função do comprimento real. Isto permite uma mesma parametrização para curvas em diferentes espaços paramétricos, o que é útil para a geração a continuidade de malhas em superfícies na transição entre dois *Patches*. Foram acrescentadas técnicas que permitem a determinação da interseção entre múltiplas superfícies, determinado-se os pontos de interseção entre as curvas de interseção, gerando quebras nas linhas. Isto permite gerar sub-domínios que levem em conta todas as interseções e suas interações. A precisão dos

resultados obtidos é elevada e pode ser controlada por parâmetros como a tolerância angular e a o erro relativo admissível, dependendo da aplicação requerida. Na implementação realizada, utilizou-se tolerância angular de 5º e um erro relativo admissível 10^{-12} da dimensão máxima das superfícies envolvidas. O que é mais do que suficiente para todas as aplicações de modelamento geométrico.

3) O algoritmo de geração automática de malhas não-estruturadas em superfícies paramétricas recortadas curvas ou planas consiste em outra importante contribuição deste trabalho. Numa primeira etapa, é feita a discretização das linhas do contorno, que podem ser as linhas de bordo do *patch*, linhas provenientes da interseção com outras superfícies ou, ainda, definidas pelo usuário, o que pode ser feito em superfícies planas. A discretização do contorno é uma etapa importante, pois condiciona a qualidade da malha no interior do domínio. Foi desenvolvido um algoritmo de subdivisão orientado pelos ângulos entre os vetores tangentes das linhas de interseção. Após a discretização do contorno, uma malha de fundo é gerada através de subdivisões recursivas em uma estrutura tipo *quadtree*, para estabelecer uma distribuição do tamanho de elemento em função das curvaturas locais. Utilizou-se uma técnica modificada do trabalho de Miranda e Martha (2002), que introduziram uso de *quadtrees* como malha de fundo, em que a proporção das células da quadtree no espaço paramétrico é função das dimensões da superfície no espaço 3D. Em seguida, utiliza-se um algoritmo Frontal com uma série de restrições topológicas conhecidas da literatura internacional, principalmente no trabalhos de Lee e Hobbs (1999), e algumas contribuições originais, tais como a inserção iterativa de novos Nós a partir da projeção sobre a superfície. Após o algoritmo preencher todo o domínio especificado para a geração, realiza-se um processo de suavização Laplaciana que garante a qualidade da malha gerada. A malha é gerada sobre as superfícies em coordenadas paramétricas e transformada para o espaço 3D a partir das funções paramétricas das superfícies. Como é possível verificar nos exemplos apresentados, as malhas geradas apresentam grande qualidade geométrica, com transições suaves nas zonas de grande curvatura.

4) Mesmo em domínios complexos, constituídos de várias superfícies recortadas por linhas de interseção, foi possível gerar malhas de qualidade, como é possível verificar nas aplicações apresentadas. O uso do algoritmo em domínios planos recortados também apresenta bons resultados, permitindo a geração de malha em contornos planos arbitrários definidos pelo usuário, o que tem grande aplicação no modelamento de problemas de Engenharia.

5) A disponibilidade das coordenadas paramétricas de cada ponto das malhas, bem como das funções paramétricas das superfícies, permite o cálculo preciso e eficiente de sensibilidades empregadas em processos de otimização de forma. Isto tem implicações importantes na eficiência e no desempenho dos códigos de otimização de forma.

7.1. SUGESTÕES PARA TRABALHOS FUTUROS

Muitas das técnicas empregadas nesta Tese podem ser aprofundadas em futuras pesquisas, afim de melhorar o desempenho e o potencial de aplicação. A seguir são apresentadas sugestões para futuros trabalhos.

1) Em termos de modelamento paramétrico, poderiam ser introduzidas linhas e superfícies NURBS e superfícies de subdivisão, afim de aumentar a capacidade e generalidade no modelamento.

2) No algoritmo de interseção, poderiam ser introduzidas restrições ao processo de subdivisão a fim de melhorar os resultados em superfícies geradas com polilinhas com vértices que produzem arestas. Além disso, poderia ser implementado um algoritmo para automatizar a detecção dos sub-domínios, atualmente definidos pelo usuário.

3) No algoritmo para a geração de malha, vale a mesma observação quanto ao caso das superfícies geradas por polilinhas. Além disso, o fundamental seria o desenvolvimento de uma estratégia geral para geração de malhas adaptativas (2D e 3D) baseada no algoritmo de geração de malhas atual.

4) Por fim, junto às técnicas já desenvolvidas, poderia ser agregada uma estrutura de dados para criar um sistema de modelagem por B-Rep e CSG, aumentando significativamente o potencial de modelamento geométrico.

5) Implementar os predicados geométricos de Shewchuk, 1997, com o objetivo de tornar robustos os algoritmos em termos de precisão numérica.

REFERÊNCIAS BIBLIOGRÁFICAS

Abdel-Malek, K., Yeh, H. J., 1996. “Determining intersection curves between surfaces of two solids”, **Computer-Aided Design**, vol. 28:6/7, pp. 539-549.

Abdel-Malek, K., Yeh, H. J., 1997. “On the determination of starting points for parametric surface intersections”, **Computer-Aided Design**, vol. 29:1, pp. 21-35.

Ames, A.L. Nadeau, D. R. Moreland, J. L., 1996. **The VRML Source Book**. John Wilet & Sons, New York, 650p.

Andrade, L. N., 1998. “**Traço de interseção de superfícies com passos circulares**”, Tese de Doutorado, FEEC, Unicamp, Campinas.

Andrade, L.N., Wu, S-T., 1996. “Caminhando sobre uma interseção de superfícies com passos circulares”, In: **Anais do IX SIBGRAPI**, pp.151-158.

Asteasu, C., 1988. “Intersection of arbitrary surfaces”, **Computer-Aided Design**, vol. 20:9, pp. 533-538.

Aymone, J.L.F., Bittencourt, E., Creus, G.J., 2001. “Simulation of 3D metal-forming using an arbitrary lagrangian-eulerian finite element method”, **Journal of Materials Processing Tecnology**, vol. 110:2, pp. 218-232.

Aymone, J. L. F., Teixeira, F. G., Bittencourt, E., Creus, G.J., 2000. “Numerical simulation of the hydroforming process”, In: **Anais do SAE Brasil 2000**, São Paulo, pp. 1-8.

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

Barnhill, R., Kersey, S., 1990. “A marching method for parametric surface/surface intersection”, **Computer-Aided Design**, vol. 7, pp. 257-280.

Barth, W., Huber, E., 1999. “Computations with tight bounding volumes for general parametric surfaces”, In: **Proceedings of the 15th European Workshop on Computational Geometry - CG'99**, pp. 123-126, Antibes, France.

Bézier, P. E., 1966, “Définition numérique des courbes et surfaces”, **Automatisme**, vol. XI, pp. 625-632.

Bézier, P. E., 1971. “Example of an existing system in the motor industry: The Unisurf System”, In: **Proc. Royal Soc. of Loindon**, vol. A321, pp. 197-205, London.

Bézier, P. E., 1972. "**Numerical Control: Mathematics and Applications**",John Wiley, London,.

Bittencourt, E., 1994. "**Tratamento do Problema de Contato e Impacto em Grandes Deformações pelo Método dos Elementos Finitos**", Tese (Doutorado em Engenharia) – Escola de Engenharia, Curso de Pós-Graduação em Engenharia Civil, UFRGS, Porto Alegre.

Bittencourt, E. e Creus, G. J., 1998. "Finite Element Analysis of Three-dimensional Contact and Impact in Large Deformation Problems." **Computers & Structures**, vol. 69, nº 2, pp. 219-234.

Bittencourt, E., Olmi, F., Creus, G. J., 1997. "An interactive remeshing technique applied to two dimensional problems involving large elasto-plastic deformations", In: **Proceedings of V COMPLAS**, Barcelona.

Bloomenthal, M., and Riesenfeld R., 1991. "Approximation of Sweep Surfaces by Tensor Product NURBS", **SPIE Curves and Surfaces in Computer Vision and Graphics II**, Vol. 1610, pp. 132-144.

Borouchaki, H., Lafon, P., Laug, P., George, P. L., 2000. "Minimal variational surfaces and quality triangular meshes", In: **Proceedings, 9th International Meshing Roundtable**, Sandia National Laboratories, pp. 217-225.

Button, S. T., Bortolussi, R., 2001. "Simulação do processo de hidroconformação de tubos", in: **Anais do XVI Congresso Brasileiro De Engenharia Mecânica**, Uberlândia.

Canann, S. A., Liu Y. C., Mobley, A. V., 1997. "Automatic 3D Surface Meshing to Address Today's Industrial Needs", **Finite Elements in Analysis and Design**, vol. 25, pp. 185 – 198.

Cantù, M., 2000. "**Dominando o Delphi 5, A Bíblia**", Makron Books, São Paulo.

Carnet, J., 1978. **Une méthode heuristique de maillage dans le plan pour la mise en oeuvre des éléments finits**, Thèse, Paris.

Castier, B., Martha, L. F., Gattass, M., 1994. "Uma taxonomia para manipulação interativa e visualização de objetos 3D", In: **Anais do SIBGRAPI 94 – VII Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens**, Curitiba, pp. 9-12.

Catmull, E. e Clark, J., 1978. "Recursively generated B-spline surfaces on arbitrary topological meshes"**, Computer-Aided Design**, Volume 10, Issue 6, November, pp 350-355.

Catmull, E. e Rom, R., 1974. A class of local interpolation splines. In: Barnhill R.E and R.F. Riesenfled (eds.), **Computer Aided Geometric Design**, Academic Press, New York.

Cavendish, J. C., Field, D. A. Frey, W. H., 1985. "An approach to automatic three-dimensional finite element mesh generation", **International Journal in Numerical Methods in Engineering**, vol. 21, pp. 329-347.

Comba, J. L. D. e Stolfi, J., 1993. "Affine Arithmetic and its Applications to Computer Graphics", In: **Anais do SIBGRAPI 93 – VI Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens**, Recife, pp. 9-18.

Coelho, L. C. G., 1998. "**Modelagem de Cascas com Interseções Paramétricas**", Tese de doutorado, PUC-RJ, Rio de Janeiro.

Coelho, L. C. G., Gattass, M., Figueiredo, L. H., 2000. "Intersecting and trimming parametric meshes on finite element shells", **International Journal in Numerical Methods in Engineering**, vol. 47, pp. 777-800.

Cohen, H. D., 1980. "A method for automatic generation of triangular elements on a surface", **International Journal in Numerical Methods in Engineering**, vol. 15, pp. 470-476.

Coons, S. A., 1964. "**Surfaces for Computer Aided Design**", Technical Report, Design Division, Mech. Engin. Dept., M.I.T. Cambridge, Massachusetts.

Coons, S. A., 1967. "**Surfaces for Computer-Aided Design of Space Forms**", M.I.T. Proj. MAC, MAC-TR-41.

Coquillart, S., 1987. "A control-point-based sweeping technique", **IEEE Computer Graphics and Applications**, vol. 7:11, pp. 36 – 45.

Cuillière, J. C., 1998. "An adaptive method for the automatic triangulation of 3D parametric surfaces", **Computer Aided Design**, vol. 30:2, pp. 139-149

DeBoor, C., 1972. "On calculating with B-Splines", **Journal of Approximation Theory**, vol. 6(1), pp. 50–62

Figueiredo, L. H., 1996. "Surface intersection using affine arithmetic", In: **Proceedings of Graphics Interface '96**, pp.161 – 170.

Foley, J. D., van Dam, J., Feiner, S. L., Hughes, J. F., 1992. "**Computer Graphics – Principles and Practice**", 2nd edition, Addison-Wesley.

Frey, P. J., Borouchaki, H., George, P.L., 1996. **"**Delaunay Tetrahedralization using an Advancing-Front Approach", In: **Proceedings of 5th International Meshing Roundtable**, Sandia National Laboratories, pp. 31-46.

George, P.L., 1991. "**Automatic Mesh Generation – Application to Finite Element Methods**", John Wiley and Sons, New York.

George, P.L. e Borouchaki, H., 1998. "**Delaunay Triangulation and Meshing Application to Finite Elements**", Hermes, Paris.

George, P.L., Hecht, F., Saltel, E., 1991. **"**Automatic Mesh Generator with Specified Boundary", **Computer Methods in Applied Mechanics and Engineering**, North-Holland, vol. 92, pp. 269 – 288.

Ghansemi, F., 1982. "Automatic mesh generation scheme for a two or three dimensional triangular curved surface", **Computer & Structures**, vol. 15, pp. 613-626.

GT Strudl – Automation Data Generation and GT Modeler. CASE - Computer Aided Structural Engineering Center, Georgia Institute of Teachnology.

Gleicher, M. e Kass, M., 1992. "An interval refinement technique for surface intersection", In: **Proceedings of Graphics Interface '92**, pp. 242-249.

Griffiths, J. G., 1975. "A data structure for elimination of hidden-surface algorithms", **Computer-Aided Design**, vol. 11, pp. 71-78.

Haber R. e Abel, J.F., 1982. "Discrete transfinite mappings for the description and meshing of three-dimensional surface using interactive computer graphics", **International Journal in Numerical Methods in Engineering**, vol. 18, pp. 41-66.

Hartman, J. Wernecke, J., 1996. **The VRML 2.0 Handbook – Building Moving Worlds on the Web**. Addison-Wesley, New York, 412p.

Hoschek, J. e Lasser, D., 1993. **Fundamentals of Computer Aided Geometric Design**, A. K. Peters, Wellesley.

Houghton, E. G., Emnett, R. F., Factor, J. D., Sabharwal, C. L., 1985. "Implementation of a divide-and-conquer method for intersection of parametric surfaces", **Computer Aided Geometric Design**, vol. 2, pp. 173 – 183.

Huber, E., 1998. "Intersecting general parametric surfaces using bounding volumes**",** In: **Proceedings of the 10th Canadian Conference on Computational Geometry - CCCG'98**, Montreal, Canada.

Krishnan, S. e Manocha, D., 1994. "**An efficient surface intersection algorithm based on lower dimensional formulation**", Tchnical Report, Department of Computer Science, University of North Carolina.

Lau, T. S. e Lo, S. H., 1996. "Finite element mesh generation over analytical curved surfaces", **Computer & Structures**, vol. 59: 2, pp. 301-309.

Lee, C. K. e Hobbs, R. E., 1999. "Automatic adaptive finite element mesh generation over arbitrary two-dimensional domain using advancing front technique". **Computer & Structures**, vol. 71, pp. 9 – 34.

Lira, W. W. M., Coelho, L. C. G., Cavalcanti, P. R., Martha, L. F., 2002. "A modeling methodology for finite element mesh generation of multi-region models with parametric surfaces", **Computer & Graphics**, vol. 26:6, pp. 907-918.

Lo, S. H., 1989. "Delaunay triangulation of non-convex planar domains", **International Journal in Numerical Methods in Engineering**, vol. 28, pp. 2695-2707.

Löhner, R. e Parikh, P., 1988. "Three-dimensional grid generation by the advancing front method", **International Journal in Numerical Methods in Fluids**, vol. 8, pp. 1135-1149.

Marcum, D.L. e Weatherill, N.P., 1995a. **"**Unstructured Grid Generation Using Iterative Point Insertion and Local Reconnection", **AIAA Journal,** vol. 33:9, pp. 1619-1625.

Marcum, D.L. e Weatherill, N.P., 1995b. "A procedure for efficient generation of solution adapted unstructured grid", Comp. Methods Appl. Mech. Engng., vol 127, pp. 259-268.

Marcum, D.L. e Gaither, J. A., 1999. "Unstructured surface grid generation using global mapping end physical space approximation", In: **Proceedings, 8th International Meshing Roundtable**, South Lake Tahoe, CA, USA, pp. 397-406.

Mavriplis, D. J., 1995. "An advancing front Delaunay triangulation algorithm designed for robustness", **Journal of Computational Physics**, vol. 117, pp. 90-101.

Miranda, A. C. O., Martha, L. F., 2002. “Mesh generation on high-curvature surfaces based on background quadtree structure”, In: **Proceedings of 11th International Meshing Roundtable**, Sandia National Laboratories, pp. 333-342.

Moaveni, S., 1999. “**Finite Element Analysis: Theory and Application with ANSYS**”, Prentice Hall.

Möller, P. and Hansbo, P., 1995. “On advancing front mesh generation in three dimensions”, **International Journal in Numerical Methods in Engineering**, vol. 38, pp. 3551-3569.

MSC Software. MSC.Patran, 2001. **The world’s leading CAE modeling system Brochure**, 2001. http://www.mscsoftware.com.

Owen, J.C. and Rockwood, A.P. , 1987. **Intersection of general implicit surfaces. In Geometric Modeling: Algorithms and New Trends**, G. Farin, editor, SIAM, Philadelphia, pp. 335-346.

Owen, S.J., Saigal, S., 2000, “Surface mesh sizing control”, **International Journal in Numerical Methods in Engineering**, vol. 47, pp. 497-511.

Patrikalakis, N. M., 1993. **Surface-to-surface intersections**. IEEE Computer Graphics and Applications, vol. 13, pp. 89-85

Patrikalakis, N. M. and Maekawa, T., 1991. **Intersection Problems**. Tech. report, MIT Sea Grant College Program, MIT.

Pare, E.G. Loring, R.O., Hill, I.L., Pare, R.C., 1996. **Descriptive Geometry**. Prentice Hall, New York.

Peraire, J., Peiro, J., Formaggia, L., Morgan, K., Zienkiewicz, O. C., 1988. “Finite element euler computations in three dimensions”, **International Journal in Numerical Methods in Engineering**, vol. 26, pp. 2135-2159.

Piegl, L. and Tiller, W., 1997. **The Nurbs Book**. Second Edition, Springer-Verlag, Berlin.

Ponthot, J.P., 1995. “**Traitement unifié de la mécanique des milieux continus solides en grandes transformations par la méthode des éléments finis**.” Tese de Doutorado, Université de Liège, Bélgica.

Rebay, S., 1991. “Efficient unstructured mesh generation by means of Delaunay triangulation and Bowyer/Watson algorithm”, In: **Proceedings of 3rd International Conference on numerical grid generation in Comp. Fluid Dyn**., Barcelona.

Riesenfeld, R. F., 1973. **Applications of B-Spline approximation to geometric problems of computer aided design**, PhD Thesis, Syracuse University, Syracuse, New York.

Rodríguez, R. R., 2000. **Desarrollo de un sistema integrado para tratamiento de geometría, generación de malla y datos para el análisis por el método de los elementos finitos**, Tese de doutorado, Universitat Politècnica de Catalunya.

Rogers, D. F., Adams, J. A., 1990. **Mathematical Elements for Computer Graphics**. International Edition, McGraw-Hill, Singapore.

Schaeffer, H. G.,1998. **MSC/NASTRAN PRIMER: Static and Normal Modes Analysis**, Text Binding, 432 p., MSC.

Sederberg, T. W., 1987. “Algebraic Geometry for Surface and Solid Modeling”, In: **Geometric Modeling: Algorithms and Trends**, G. Farin, Editor, SIAM, pp. 29-42.

Shewchuk, J. R., 1997. “Adaptive Precision Floating-Point Arithmetic and Fast Robust Geometric Predicates”, **Discrete & Computational Geometry**, vol.18, pp. 305-363.

Shih, R. H., 2000. **Parametric Modeling with I-DEAS** 8. 250 p. 3rd edition. Schroff Development Corp.

Siltanen, P. and Woodward, C., 1992. “Normal orienatation methods for 3D offset curves, sweep surfaces and skinning”, In: **Proceedings of Eurographics 92**, vol. 11:3, pp. 449-457.

Spiegel, M. R., 1973. **Manual de Fórmulas e Tabelas Matemáticas**. Coleção Schaum, McGraw-Hill, São Paulo, Brasil.

Stoyanov, Tz.E. , 1992. “Marching along surface/surface intersection curves with an adaptative step length”, **Computer Aided Geometric Design**, vol. 9, pp. 485–489.

Straber, W. and Seidel, H-P., 1989. **Theory and Practice of Geometric Modeling**. Springer-Verlag, Berlin.

Swartzfager, G., 1997. **Visual Basic: Programação Orientada a Objetos**. 554p. Ciência Moderna, Rio de Janeiro.

Teixeira, F.G., 2000. **Interface Gráfica para Geração de Modelos de Matrizes para Conformação Mecânica.** Porto Alegre, 2000, 25p, Seminário de Doutorado – Escola de Engenharia, Programa de Pós-Graduação em Engenharia Mecânica, UFRGS.

Tiller, W., 1983. "Rational B-splines for curve and surface representation", *IEEE* **Computer Graphics ans Applications**, vol. 4:9, pp. 61-69.

Tiller, W., 1986. "Geometric modelling using NURBS", mathematical techni-ques, **SIGGRAPH Tutorial Notes**.

Thomas, S. W., 1984. "The Alpha_1 Computer-Aided Geometric Design System in the Unix Environment", In: **Proceedings of Computer Graphics and Unix Workshop** (Katz, L., ed.), USENIX Organization, Dec.

Tristano, J. R., Owen, S. J., Canann, S. A., 1998. "Advancing Front Surface Mesh Generation in Parametric Space Using a RiemannianSurface Definition". In: **7th International Meshing Roundtable**, Sandia National Labs, pp. 429-445.

Trujillo, S., 1997. **Tudo o que você precisa saber sobre Direct3D rápido e fácil**, Ciência Moderna, Rio de Janeiro.

Ueng, W. and Lai, J.,1998. "A sweep-surface fitting algorithm for reverse engineering", **Computers in Industry**, vol. 35, pp. 261-273.

Versprille, K. J., 1975. **Computer-Aided Design Applications of the Rational B-spline Aproximation Form**, Ph.D. Dissertation, Syracuse University

Watt, A., 1993. **3D Computer Graphics**. 2nd Edition. Addison-Wesley, Harlow, England.

Wu, S. T., Andrade, L. N., 1999. "Marching along a regular surface/surface intersection with circular steps", **Computer Aided Geometric Design**, vol. 16: 4, pp. 249-268.

Zienkiewicz, O. C. e Phillips, D. V., 1971. "An automatic mesh geration sheme for plane and curved surfaces by isoparametric coordinates", **International Journal in Numerical Methods in Engineering**, vol. 3, pp. 519-528.

Zienkiewicz, O. C. e Taylor, R. L., 2000. **Finite Element Method**: Volume 1, The Basis, 5th edition Vol 1, Butterworth-Heinemann