Refinamento de Malhas Isotrópicas e Anisotrópicas e Simplificação de Malhas Isotrópicas

Alexandre De Lacassa

Refinamento de Malhas Isotrópicas e Anisotrópicas e Simplificação de Malhas Isotrópicas

Alexandre De Lacassa

Orientador:

Prof. Dr. Antonio Castelo Filho

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos necessários à obtenção do título de Mestre em Ciências de Computação e Matemática Computacional.

USP - São Carlos

Fevereiro de 2007

Dedico este trabalho aos meus pais Miguel e Nilma, com amor.

Agradecimentos

Em primeiro lugar agradeço a Deus, porque é Senhor da minha vida e me ama mais do que eu posso imaginar.

Agradeço aos meus pais por aguentarem a saudade nestes tempos em que fiquei longe, por me apoiarem nas minha decisões, e simplesmente por serem "meus pais".

Agradeço às minhas irmãs Juliana, Michelle, Daniele e Gisele pelo amor que sinto quando volto para casa depois de um tempo longe; aos meu sobrinhos Júlio, Bruno e Luana, vocês sempre me alegram quando estou com vocês. Agradeço também ao meu primo Thiago e minha tia Conceição pelos conselhos e carinho com que sempre me presenteiam.

Agradeço à minha amada Naira, que é uma companheira dedicada e amorosa. Obrigado por suportar a saudade no tempo que ficamos longe um do outro.

Agradeço ao meu orientador, Prof. Dr. Antonio Castelo Filho, pela paciência e orientação neste trabalho.

Agradeço aos meus amigos de laboratório João Paulo, Mário, Igor, Alex, Cláudio, Kémelli, Thiago, pela amizade, paciência e por estarem sempre dispostos a me ajudar. Agradeço à Carol e a Adriana pela amizade, apoio e principalmente pela ajuda quando estudávamos para as provas das disciplinas do mestrado.

E finalmente agradeço à minha família da igreja Comunidade Cristã de São Carlos, pelo carinho e amor com que me receberam fazendo com que a minha vida em São Carlos fosse menos difícil, proporcionando momentos alegres e de comunhão com Deus.

À Fapesp - Fundação de Amparo à Pesquisa do Estado de São Paulo, pelo auxílio financeiro.

Obrigado a todos aqueles que de alguma forma ajudaram na realização deste trabalho, seja me acompanhando nos momentos de lazer, seja me ajudando nos momentos de aflição. OBRIGADO!!!

Sumário

Lista de Figuras

Resumo

Em muitos problemas de simulação de fenômenos físicos ou fenômenos de engenharia, o uso das malhas é um componente muito importante. Uma malha é uma aproximação de uma dada geometria por um conjunto de elementos mais simples, tais como triângulos e quadriláteros (caso bidimensional) ou tetraedros, prismas, pirâmides e hexaedros (caso tridimensional). Nesse texto, as malhas de interesse são as não-estruturadas e compostas por triângulos.

A escolha de uma malha é fortemente influenciada pelo desempenho e precisão dos resultados da simulação. O desempenho depende do número de elementos a serem processados, ou seja, quanto maior for a área coberta por cada elemento da malha, menos elementos são necessários, por conseguinte, mais rápida será a simulação. A precisão nos resultados da simulação está relacionada tanto com o formato quanto com o tamanho dos elementos. Diferente do desempenho, quanto menor forem os elementos, mais precisos serão os resultados. O formato dos elementos também influencia a precisão, em geral, elementos mais próximos dos equiláteros são preferidos. Como é possível observar, desempenho e precisão são requisitos conflitantes e geralmente é necessário fazer uma ponderação entre eles. Para um determinado grupo de aplicações, o melhor compromisso entre desempenho e precisão é conseguido com elementos finos, longos e corretamente alinhados sobre o domínio onde a malha está definida. São as chamadas malhas anisotrópicas. Além disso, um método de refinamento anisotrópico pode melhorar ainda mais a precisão dos resultados.

O principal objetivo desse trabalho é desenvolver métodos de refinamento de malhas anisotrópicas, usando como base, e tendo como ponto de partida, os métodos de refinamento Delaunay isotrópicos, a saber, os métodos de refinamento Delaunay de Jim Ruppert [13] e de Paul Chew [6], e também realizar a simplificação Delaunay proposto por Olivier Devillers [8].

Abstract

The use of polygonal meshes for numerical simulation of physical problems is a well known component. Mesh is an piecewise approximation from a given geometry defined by a set of simpler elements, such as triangles and quadrilaterals (two-dimensional case) or tetrahedra, prisms, pyramid and hexahedra (three-dimensional case). In this work, the interest is unstructured meshes of triangles.

The choice of a mesh is aimed at the performance and the precision of the simulation results. The performance depends of the number of elements that will be processed, i.e., the larger is the covered area for each mesh element, the less element is needed, therefore the simulation is faster performed. The simulation precision is related with the shape and the size of the elements. On the other hand, the smaller the elements are, the more precise are the results.

The shape of the elements also influences on precision, generally, equilateral elements are preferred. It is worth to mention that performance and precision are opposite requirements and it is important to ponder between them. For a group of applications, the best commitment between performance and precision is obtained with thin and long elements correctly aligned on the domain where the mesh is defined. These meshes are named anisotropic meshes. Furthermore, a method of anisotropic refinement can even improve the precision.

We aim at developing anisotropic mesh methods based on isotropic properties from well known Delaunay refinement methods, viz., the Delaynay refinement methods by Jim Ruppert [13] and Paul Chew [6], and performing a Delaunay simplification proposed by Olivier Devillers [8].

Capítulo

1

Introdução

Em uma simulação numérica, encontrar uma discretização apropriada de um domínio contínuo é essencial. Este é o principal problema tratado pela geração de malhas. A geração de malhas é um bom exemplo de atividade interdisciplinar, seu desenvolvimento se dá mediante avanços em geometria computacional e combinatorial, estrutura de dados, análise numérica e aplicações científicas.

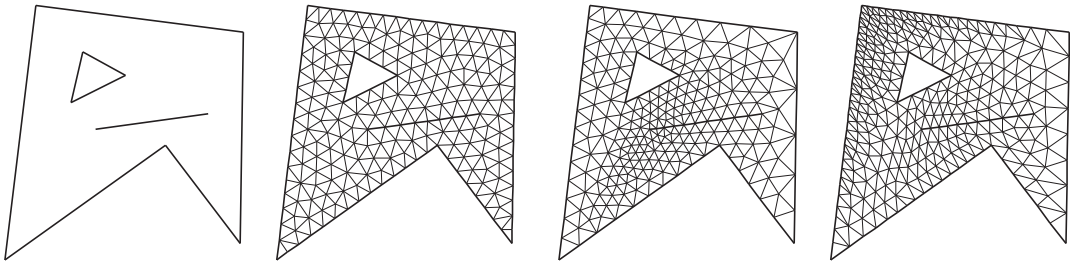
Em simulações computacionais de fenômenos físicos e também de fênomenos de engenharia, a geração de malhas é imprescindível e de grande importância. Estes fenômenos geralmente são modelados por equações diferenciais parciais (EDP). Quando as condições de contorno dessas equações são complicadas, ou são aplicadas em domínios muito irregulares, não é possível resolver o sistema analiticamente, sendo necessário trabalhar com aproximações. Existem vários métodos usados para aproximar EDPs, porém, os métodos mais frequentemente usados são: diferenças finitas, elementos finitos (ou FEM - *finite elements method*) e volumes finitos. Eles são usados para modelar os mais variados fenômenos, tais como: deformação mecânica, transferência de calor, dinâmica dos fluidos, propagação de ondas eletromagnéticas, mecânica quântica, entre outros.

Os métodos numéricos aproximam a solução de uma EDP trocando o sistema contínuo por um número finito de equações algébricas lineares (ou não-lineares). Este processo de discretização espalha um conjunto de pontos, chamados "nós", sobre o domínio e associa uma variável a cada um deles. Um exemplo muito comum é a simulação da transferência de calor sobre uma barra de aço, onde a temperatura é tratada apenas em alguns pontos da superfície e do interior da barra.

Para discretizar um domínio, não é suficiente escolher um conjunto qualquer de pontos para serem os nós, o domínio do problema deve ser particionado em pequenos pedaços de formato simples. No FEM, por exemplo, estes pedaços são chamados elementos e são

triângulos ou quadriláteros (em duas dimensões), ou ainda tetraedros, prismas, pirâmides e hexaedros (em três dimensões). O FEM emprega um nó em cada vértice dos elementos, estes nós são compartilhados por vários elementos. A coleção dos nós e elementos é chamada de malha de elementos finitos. Como os elementos possuem formatos simples, é possível aproximar o comportamento da EDP em cada elemento. Acumulando estes efeitos sobre todos os elementos, deriva-se um sistema de equações cuja solução aproxima um conjunto de quantidades físicas (calor, por exemplo) em cada nó do domínio. Outros métodos numéricos também utilizam malhas, porém com suas próprias características.

A precisão da solução no FEM depende significativamente da qualidade da malha. Uma malha é considerada de boa qualidade se ela for constituída por elementos que possuam tamanho e formato adequados. A figura 1.1 apresenta vários tipos de malhas, onde cada uma se adequa a um tipo de problema específico.



(a) Geometria do domínio (b) Malha uniforme (c) Malha refinada (d) Malha anisotrópica

Figura 1.1: Exemplos de diferentes malhas geradas para um mesmo domínio [3].

As malhas são geralmente categorizadas em estruturadas e não-estruturadas. As malhas estruturadas apresentam uma estrutura topológica uniforme, o que não acontece com malhas não-estruturadas. Nas malhas estruturadas, os índices dos vizinhos de um nó podem ser obtidos por uma simples soma. Nas malhas não-estruturadas, os vizinhos são obtidos por meio de consultas em estruturas de dados mais elaboradas. A figura 1.2 apresenta exemplos de malhas estruturadas e não-estruturadas triangulares. Um tipo particularmente simples de malha estruturada é a malha cartesiana regular, onde os elementos são quadriláteros ou hexaedros idênticos.

As malhas estruturadas são fáceis de serem geradas e manipuladas, requerem estruturas de dados simples e reduzem a dificuldade de programação. No entanto, o uso de malhas regulares estruturadas limita a aplicabilidade dos métodos numéricos a problemas cujos domínios são geometricamente simples e cujas funções solução são suaves. Em problemas envolvendo domínios complicados, onde as soluções variam rapidamente sobre o domínio, é necessário o uso de malhas não-estruturadas. Essas malhas possibilitam variar facilmente a topologia e o espaçamento em diferentes regiões do domínio. Por exemplo, em modelagem de terromotos, é necessário uma discretização mais fina e densa no centro do tremor, tendo o cuidado de não desperdiçar pontos em regiões de pouca atividade. No

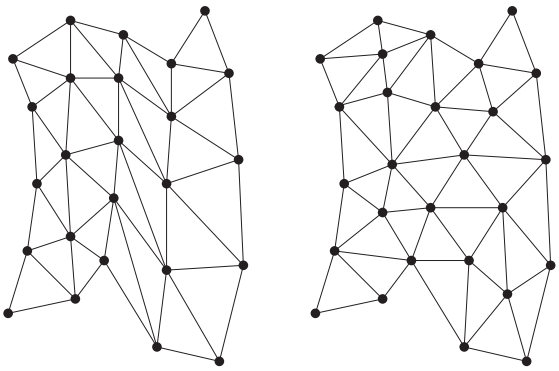


Figura 1.2: Exemplo de malha triangular estruturada e não estruturada.

entanto, para esse tipo de malha a teoria numérica se torna mais difícil e a implementação dos algoritmos mais trabalhosa.

Numa simulação pelo FEM, o tamanho do elemento produz dois efeitos importantes. Uma região com elementos pequenos oferece mais precisão que uma com elementos grandes, porém o tempo necessário para resolver um problema é proporcional ao número de elementos. Portanto, escolher o tamanho dos elementos consiste em ponderar entre desempenho e precisão. Além disso, o tamanho do elemento requerido para se obter uma dada precisão depende do comportamento do fenômeno físico sendo modelado e pode variar entre diferentes regiões do domínio do problema (ver figura 1.3). Por exemplo, em simulação de dinâmica dos fluidos, elementos pequenos devem ser dispostos nas regiões de turbulência e elementos maiores podem ser postos nas áreas de relativa quietude. Em uma malha construída por elementos de tamanho uniforme, tais elementos devem ser pequenos o suficiente para garantir a precisão desejada na parte do domínio mais relevante, podendo, portanto, demandar uma quantidade excessiva de computação.

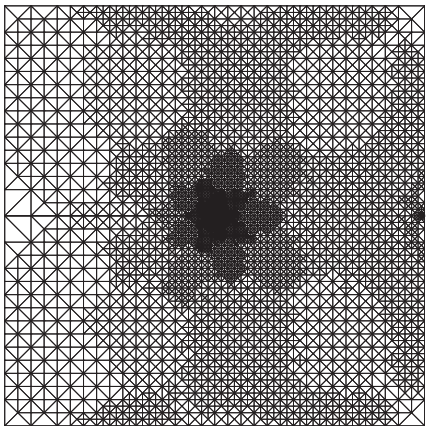


Figura 1.3: Malha não estruturada com densidade de pontos variante sobre o domínio [17]

Dada uma malha grossa (relativamente poucos elementos), não é tão difícil refiná-la para produzir outra com mais elementos, o processo inverso, conhecido como simplificação

de malhas, não é tão fácil.

Outro ponto importante na geração malhas é o formato dos elementos. Para malhas isotrópicas procura-se por elementos tão próximos quanto possível dos eqüiláteros. Elementos com formato inadequado podem degradar a qualidade da solução numérica. Elementos com ângulos próximos de 180° podem causar um grande erro de discretização, portanto, a solução gerada por um método numérico, como o FEM, pode ser bem menos precisa que a solução que o método poderia oferecer.

Sabe-se, portanto, que o tamanho e o formato dos elementos devem ser controlados para garantir uma determinada precisão na solução. Um problema que surge então é determinar quais regiões do domínio devem ser mais densas e quais devem ser menos densas. A solução reside nas condições numéricas do domínio do problema.

O domínio do problema é descrito pela sua geometria (criado em um sistema CAD, por exemplo), e também pelas condições numéricas sobre o domínio. As condições numéricas do problema afetam a precisão da solução, uma vez que estão diretamente relacionadas com os erros de discretização. Essas condições definem a densidade de espaçamento local desejada nas diversas regiões do domínio. A partir de uma análise de erro *a priori* ou *a posteriori*, baseada na simulação numérica inicial, tais condições podem ser obtidas .

Um bom exemplo é o gradiente da solução. Para um problema cuja solução tem uma variação lenta, é possível utilizar uma malha aproximadamente uniforme, ou seja, cujos elementos têm aproximadamente o mesmo tamanho. Para problemas cujas soluções apresentam variações rápidas, pode ser necessário usar malhas bem refinadas em algumas regiões e pouco refinada em outras, podendo até serem simplificadas.

É possível obter uma boa precisão com elementos próximos ao equilátero, que é uma característica das malhas isotrópicas, no entanto a mesma precisão pode ser obtida com menor número de elementos se utilizarmos uma malha anisotrópica. Em uma malha anisotrópica, os elementos são longos, finos e orientados em uma direção específica, tal direção depende das condições numéricas do problema aplicado sobre a malha. Em muitas aplicações o uso de malhas anisotrópicas representa o melhor compromisso entre precisão (menor erro de aproximação) e desempenho (menos elementos para processar). Existem aplicações nas quais a geração ou o uso de uma malha isotrópica fina o bastante para obter uma solução precisa torna-se computacionalmente impraticável, devido ao número de elementos muito grande da malha. No entanto, uma malha anisotrópica pode oferecer uma boa precisão com menos elementos, melhorando assim o desempenho.

O propósito central desse texto é realizar um estudo sobre refinamento e simplificação de malhas isotrópicas, e um estudo sobre anisotropia com o intuito de aplicar os métodos de refinamento isotrópicos em metódos para refinar malhas anisotrópicas. No capítulo 2 são apresentados alguns conceitos e definições que serão usados nos capítulos seguintes. Nos capítulos 3 e 4 são apresentados dois métodos de Refinamento Delaunay e um método de Simplificação Delaunay. O capítulo 5 discute o conceito de anisotropia para malhas

bidimensionais. As implementações realizadas e a descrição da estrutura de dados utilizada estão abordadas no capítulo 6. No capítulo 7 são apresentados os resultados obtidos e por fim o capítulo 8 apresenta a conclusão deste trabalho.

Capítulo

2

Considerações Gerais

Nesse capítulo são apresentados alguns dos principais conceitos envolvidos nas discussões presentes nos capítulos seguintes.

Definição 1 (Espaço Euclidiano) *Chamaremos de* Espaço Euclidiano *de dimensão n como sendo o espaço* $\mathbb{R}^n$ *dotado da métrica euclidiana usual, isto é, dado dois pontos* $p = (p_1, ...p_n)$, $q = (q_1, ...q_n) \in \mathbb{R}^n$, *medimos a distância entre p e q por:*

$$||p - q|| = \sqrt{(p_1 - q_1)^2 + ... + (p_n - q_n)^2} \tag{2.1}$$

Definição 2 (Fecho Convexo) *Dado um conjunto de pontos* $P \subset \mathbb{R}^n$, *o conjunto de todas as combinações lineares convexas, dado pela equação 2.2, exprime a definição de* fecho convexo *(figura 2.1).*

$$Conv(P) = \sum_{i=1}^{n} \lambda_i p_i, \sum_{i=1}^{n} \lambda_i = 1, \lambda_i \geq 0, \lambda_i \in \mathbb{R}, p_i \in P \tag{2.2}$$

O fecho convexo $Conv(P)$ pode ser visto como o menor conjunto convexo que contém os pontos de P. Será notado que o fecho convexo está relacionado com as definições de triangulação (definição 16) e triangulação de Delaunay (definição 19).

Definição 3 (Esfera S^n) *A* esfera S^n *com centro em* $c = (c_0, c_1, ..., c_n)$ *e raio r é o conjunto dos pontos* $\{P \in \mathbb{R}^{n+1} / P = (x_0, x_1, ..., x_n), \sqrt{(x_0 - c_0)^2 + (x_1 - c_1)^2 + ... + (x_n - c_n)^2} = r\}$.

Definição 4 (Subespaço Afim) *Um* subespaço afim *é um subespaço vetorial transladado da origem.*

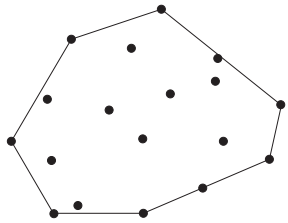


Figura 2.1: Exemplo bidimensional do fecho convexo de um conjunto de pontos.

Definição 5 (Posição Geral) *Dizemos que um conjunto de pontos $X \subset \mathbb{R}^n$ está em* posição geral *se:*

1. *Nenhum subespaço afim contém X;*
2. *Nenhuma esfera S^{n-1} intercepta mais de $n+1$ pontos de X.*

Definição 6 (Simplexo) *Um n-*simplexo *$\sigma \subset \mathbb{R}^n$ é o fecho convexo de $n+1$ pontos em posição geral (*figura 2.2*).*

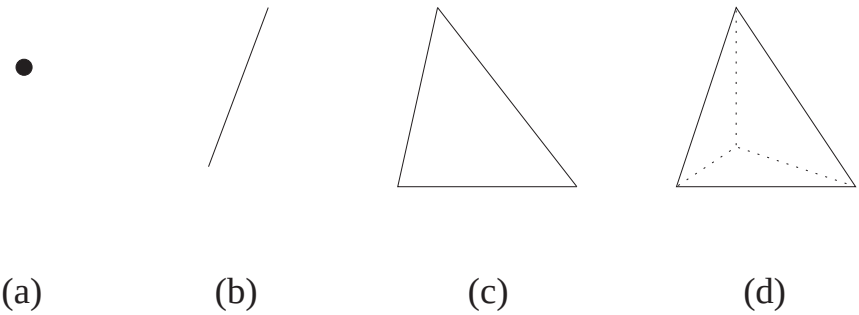


Figura 2.2: a) 0-simplexo, b) 1-simplexo, c) 2-simplexo, d) 3-simplexo.

Definição 7 (Face) *Se σ é um n-simplexo formado pelos pontos $X = \{p_0, ..., p_n\}$, qualquer l-simplexo dado por um subconjunto de $l+1$ elementos de X é* face *de σ.*

Definição 8 (Fronteira de um n-simplexo) *A* fronteira de um n-simplexo *σ, denotada por $\partial\sigma$, consiste de todos os $(n-k)$-simplexos contidos em σ, $0 < k \leq n$.*

Cada simplexo na fronteira de S é uma face de S. Um k-simplexo na fronteira é denominado *k-face*.

Definição 9 (Complexo Simplicial) *Um* complexo simplicial *C é um conjunto finito de simplexos satisfazendo:*

1. *Se σ é um simplexo em C e σ' uma de suas faces, então σ' também pertence a C.*

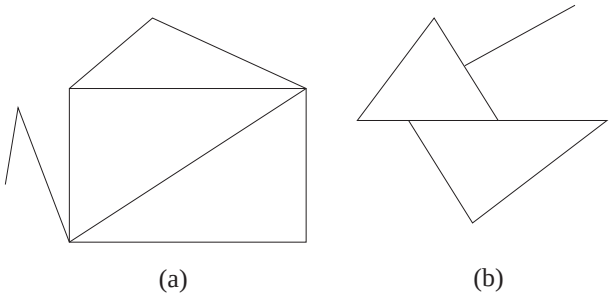


Figura 2.3: a) Exemplo de um complexo simplicial, b) não define um complexo simplicial, note que existem intersecções que não definem uma face.

2. *Se σ_1 e σ_2 são dois simplexos de C, então $\sigma_1 \cap \sigma_2$ ou é face de ambos, ou é vazia (*figura 2.3*).*

Definição 10 (Dimensão de um Complexo Simplicial) *A* dimensão *de um complexo simplicial C é definida pela dimensão máxima de seus simplexos.*

Se σ é um n-simplexo e $n > 0$, então a fronteira $\partial\sigma$ é um complexo simplicial de dimensão $n - 1$.

Definição 11 (Decomposição Simplicial) *Um complexo simplicial C é uma* decomposição simplicial *para um conjunto $B \subset \mathbb{R}^n$ se $B = \bigcup_{\sigma \in C} \sigma$.*

Definição 12 (Estrela) *A* estrela *de um simplexo S, denotada por $Star(S)$, é o conjunto de simplexos C que contém S* (figura 2.4).

Definição 13 (Link) *O* link *de um simplexo S, denotado por $link(S)$, são os simplexos de $\partial(Star(S))$ que não são incidentes a S* (figura 2.4).

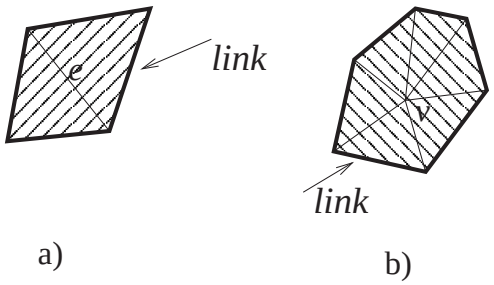


Figura 2.4: As regiões hachuradas definem a) a estrela de uma aresta e e b) a estrela de um vértice v. As arestas mais largas indicam seus respectivos *links* da aresta e e do vértice v.

Definição 14 (Grafo) *Um grafo $G(V, E)$ é definido como um conjunto de vértices $V \neq \emptyset$ e um conjunto finito de arestas E, conectando os vértices de V.*

Definição 15 (Triangulação de um Conjunto Qualquer) *Uma triangulação de dimensão k de um conjunto $B \subset \mathbb{R}^n$ qualquer é uma decomposição simplicial do fecho convexo de B em que qualquer simplexo de dimensão inferior a k está contido em um simplexo de dimensão k.*

Definição 16 (Triangulação de um conjunto de Pontos) *Seja $A = \{p_0, ..., p_k\} \subset \mathbb{R}^n$, um conjunto de pontos em $\mathbb{R}^n$. Uma* triangulação *de dimensão k de A é uma decomposição simplicial de $Conv(A)$ em que os vértices são os pontos de A e qualquer simplexo de dimensão inferior a k está contido em um simplexo de dimensão k.*

Uma triangulação (definição 16) é um exemplo de grafo.

Definição 17 (Diagrama de Voronoi) *Seja $P \subset \mathbb{R}^n$ um conjunto de pontos, no caso do diagrama de Voronoi serão chamados sites. Para cada site $p, q \in P$, seja $B(p, q) = \{x : ||x - q|| = ||x - p||\}$ o bissetor de p e q. Considere o semi-espaço $D(p, q) = \{x : ||x - p|| \leq ||x - q||\}$ contendo p, e $D(q, p)$ é o semi-espaço que contém q. Define-se por $VR(p, P) = \bigcap_{q \in P, q \neq p} D(p, q)$ a* Região de Voronoi[1]*. O* Diagrama de Voronoi *de P é definido por (figura 2.5):*

$$V(P) = \bigcup_{p,q \in P, p \neq q} VR(p, P) \bigcap VR(q, P) \tag{2.3}$$

Na figura 2.5 é apresentado um exemplo de diagrama de Voronoi em $\mathbb{R}^2$.

Definição 18 (Circunsfera) *Define-se circunsfera de um n-simplexo σ como a esfera S^{n-1} que contenha todos os vértices de σ. Em $\mathbb{R}^2$, denomina-se* circuncírculo*. O raio da circunsfera é enominado* circunraio *.*

Definição 19 (Triangulação de Delaunay) *Seja $X = \{p_0, ..., p_k\} \subset \mathbb{R}^n, n \leq k$, um conjunto de pontos em posição geral. Diz-se que uma triangulação de X é e* Delaunay *se a circunsfera de todo n-simplexo não contém nenhum outro ponto de X em seu interior (*figura 2.6*).*

Entre as propriedades da triangulação de Delaunay podemos citar [9, 14]:

- O circuncírculo de cada triângulo não contém nenhum outro vértice da triangulação (impossível generalizar para $\mathbb{R}^n$).

[1]Conhecido também como célula de Voronoi.

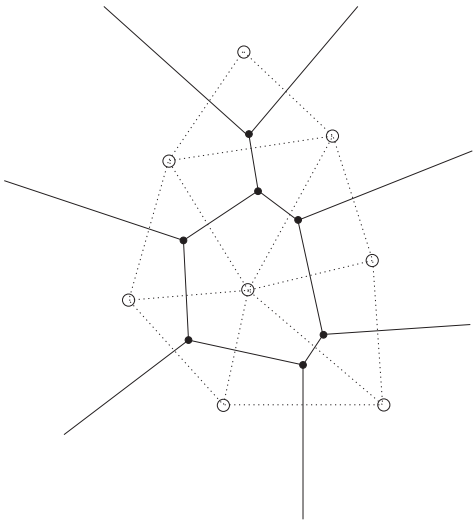


Figura 2.5: Diagrama de Voronoi (dos pontos sem preenchimento) em linhas contínuas. Em linhas pontilhadas, seu dual, a triangulação de Delaunay.

- Maximiza o ângulo mínimo de cada triângulo (válida somente em $\mathbb{R}^2$);
- É o dual do diagrama de Voronoi [1];
- É única para pontos em posição geral;
- Em $\mathbb{R}^2$ é possível obter a triangulação em $O(k \log k)$;
- Para pontos $(x, y) \in \mathbb{R}^2$, a triangulação é a projeção ortogonal da parte inferior do fecho convexo dos pontos $(x, y, x^2 + y^2) \in \mathbb{R}^3$ (impossível generalizar esta relação para $\mathbb{R}^n$) [9].

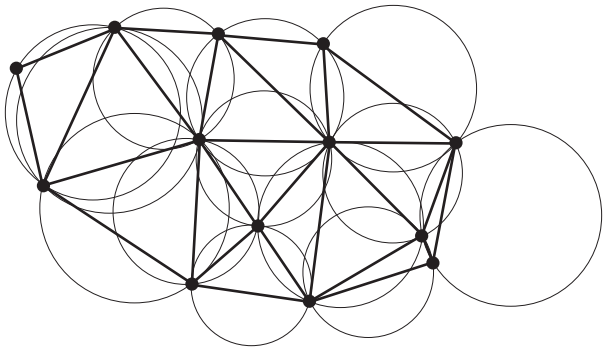


Figura 2.6: Cada triângulo em uma triangulação de Delaunay possui circuncírculo vazio.

Sejam as seguintes definições considerando a triangulação de Delaunay bidimensional.

Definição 20 (Aresta Globalmente Delaunay) *Uma aresta é* globalmente Delaunay *se existir um círculo vazio de pontos passando pelos seus vértices.*

Observe que toda aresta de bordo da triangulação é *globalmente Delaunay.*

Definição 21 (Aresta Localmente Delaunay) *Seja e uma aresta compartilhada por dois triângulos t_1 e t_2. Seja v_1 e v_2 os vértices opostos a e em t_1 e t_2 respectivamente, conforme está mostrado na figura 2.7. A aresta e é* localmente Delaunay *se o circuncírculo de t_1 não inclui v_2 e vice-versa.*

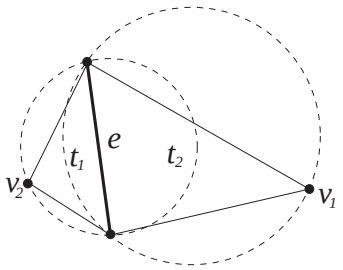


Figura 2.7: Exemplo de uma aresta e localmente Delaunay.

Definição 22 (*Flipping* Bidimensional) *Considere um quadrilátero convexo formado por quatro vértices. Este quadrilátero possui duas diagonais internas chamadas flipping uma da outra (ver figura 2.8).*

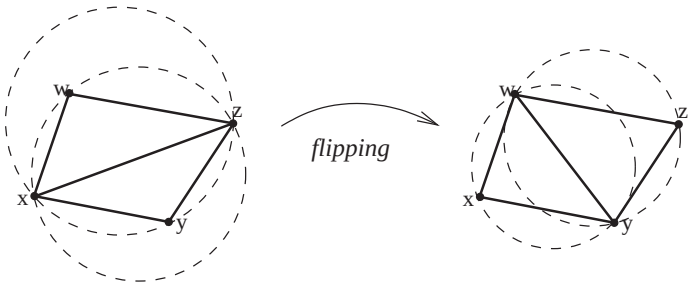


Figura 2.8: *Flipping* entre arestas para garantir o critério Delaunay.

Observe que se o quadrilátero for côncavo então ele não possui *flipping* de diagonais.

Sejam os seguintes lemas relacionados à triangulação de Delaunay, provados em [14].

Lema 1 *Seja T uma triangulação de Delaunay em que todas as arestas são localmente Delaunay. Então T é a triangulação de Delaunay.*

Lema 2 *Seja $\boldsymbol{e}$ uma aresta de uma triangulação T. Então $\boldsymbol{e}$ é localmente Delaunay ou então possui flipping $\boldsymbol{f}$ e $\boldsymbol{f}$ é localmente Delaunay.*

Pelos lemas acima, pode-se observar que o *flipping* é uma ferramenta de considerável importância em algoritmos para construção de triangulações de Delaunay.

Existe um algoritmo chamado de *Flipping* Incremental bidimensional, cuja base do seu funcionamento é dada pelos conceitos acima explicados. Este algoritmo é citado no

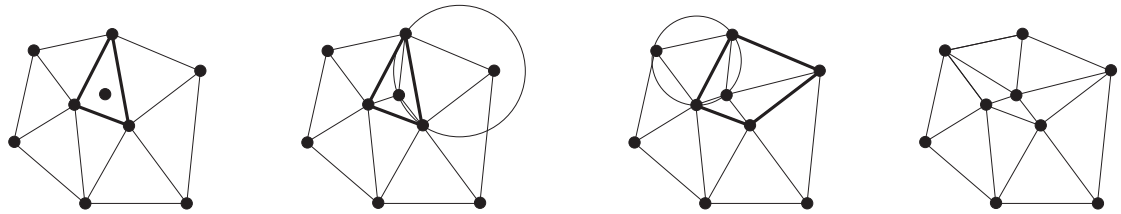


Figura 2.9: Algoritmo de *Flipping* Incremental; os segmentos em negrito indicam as arestas que precisam ser testadas se são localmente Delaunay; (a) o triângulo que contém o ponto é encontrado e suas arestas devem ser testadas; (b) uma das arestas não é localmente Delaunay, ocorre o *flipping*, adicionando duas arestas às que devem ser testadas; (c) *flipping* de outra aresta e (d) final da inserção, onde todas as arestas são localmente Delaunay.

Algoritmo 1 Triangulação de Delaunay por *Flipping* Incremental

Entrada:

V : conjunto inicialmente formado pelos vértices que estarão presentes na triangulação.

início

Seja a triangulação de Delaunay T inicial composta por um triângulo cujos vértices estão no infinito.

enquanto $V \neq \emptyset$ **faça**

Escolha qualquer $v_i \in V$.

Encontre o triângulo $t_{abc} \in T$ que contenha o vértice v_i no seu interior.

Elimine o triângulo t_{abc} de T e cria-se os triângulos t_{iab}, t_{ibc} e t_{ica}.

enquanto existir uma aresta $e \in link(v_i)$ que não seja localmente Delaunay **faça**

Realizar *flipping* de e.

fim_enquanto

Retire v_i de V.

fim_enquanto

Remova os três vértices postos no infinito juntamente com os triângulos incidentes a eles.

fim

próximo capítulo, por isso ele está brevemente descrito no algoritmo 1 e seus principais passos estão ilustrados na figura 2.9.

Os geradores de malhas procuram criar triangulações para um dado domínio bem delimitado. O tipo de entrada mais usual para um gerador de malhas bidimensional é um conjunto de pontos e segmentos de reta, mais conhecido na literatura como PSLG (*Planar Straight Line Graph*), que pode ser definido como:

Definição 23 (PSLG - Planar Straight Line Graph) *Um* PSLG *é um conjunto de vértices e segmentos de reta que satisfaz duas restrições:*

1. *Cada segmento do PSLG deve ter os dois vértices nas suas extremidades também presentes no PSLG.*
2. *Dois segmentos só podem se interceptar nos seus vértices terminais.*

Definição 24 (Orelha) *Considere um polígono* $P = \{p_0, p_1, \ldots, p_k = p_0\}$. *Uma orelha do polígono* P *é um triângulo formado pelos vértices* (p_i, p_{i+1}, p_{i+2}) *desde que o segmento* $p_i p_{i+2}$ *esteja no interior de* P *e que não intercepte sua fronteira.*

Definição 25 (Power) *Seja* C *um círculo de centro em um ponto* $\boldsymbol{p}$ *e raio* r. *O "power" de um ponto* x, *denotado por* $power(x, C)$ *é dado por:*

$$power(x, C) = \|x - \boldsymbol{p}\|^2 - r^2 \tag{2.4}$$

Capítulo

3

Refinamento Delaunay

Os algoritmos de Refinamento Delaunay para geração de malhas trabalham mantendo uma triangulação de Delaunay que é refinada pela inserção de vértices cuidadosamente localizados até que a malha satisfaça certas condições relativas à qualidade de um elemento e seu tamanho.

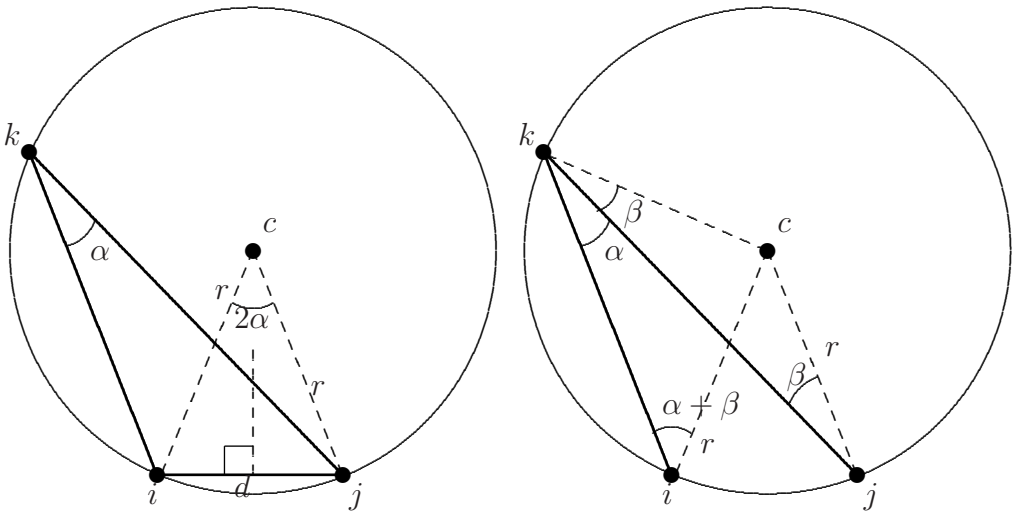
Estes algoritmos são bons porque exploram várias características muito importantes da triangulação de Delaunay. Uma destas características é que a triangulação de Delaunay maximiza o ângulo mínimo de todas as possíveis triangulações de um conjunto de pontos. Outra característica muito importante é que as operações de inserção de vértices são locais, ou seja, o ato de inserir um vértice em uma determinada região de uma malha para melhorar a qualidade dos elementos não interfere em outra região da malha.

Um ponto importante nos algoritmos de refinamento Delaunay é descobrir o local ideal no qual o novo vértice deve ser inserido. Neste capítulo veremos que o local ideal para inserir um vértice é o mais longe possível dos outros vértices, caso contrário resultará em arestas pequenas, o que por sua vez resultará em triângulos muito finos. Como não há vértices no circuncírculo de um triângulo de Delaunay, uma triangulação de Delaunay é a estrutura ideal para se encontrar o ponto mais distante dos outros vértices.

3.1 Uma Medida de Qualidade para Simplexos

Segundo Shewchuk [14] uma medida de qualidade mais natural e elegante para analizar os algoritmos de refinamento Delaunay é a *razão circunraio-menor aresta* de um simplexo, ou seja, o raio da circunsfera de um simplexo dividido pelo tamanho da menor aresta do simplexo. É desejável que o valor desta razão seja o menor possível. No caso bidimensional a razão circunraio-menor aresta de um triângulo é uma função do seu menor ângulo. Seja

$\triangle ijk$ com circuncentro c e circunraio r, como ilustrado na figura 3.1(a). Tome d como o tamanho da aresta ij, e o ângulo oposto a esta aresta como $\alpha = \angle ikj$.



(a) $d = 2rsen(\alpha)$ (b) $\angle icj = 2\angle ikj$

Figura 3.1: Demonstração: razão circunraio-menor aresta

Um fato geométrico bem conhecido é que $\angle icj = 2\alpha$, como mostra a figura 3.1(b). Seja $\beta = \angle jkc$. Como $\triangle kci$ e $\triangle kcj$ são isósceles, $\angle kci = 180° - 2(\alpha + \beta)$ e $\angle kcj = 180° - 2\beta$. Subtraindo $\angle kci$ de $\angle kcj$ temos $\angle icj = 2\alpha$. Observando a figura 3.1(a) pode-se notar que $sen(\alpha) = d/(2r)$. Então, se a menor aresta do triângulo tem tamanho d, α é o seu menor ângulo. Se B é um limitante superior sobre a razão circunraio-menor aresta entre todos os triângulos da malha, então não há ângulo menor que $arcosen\frac{1}{2B}$. Um gerador de malhas deve trabalhar com o menor valor possível para B. Neste trabalho usarei o termo "*qualidade*" de um triângulo ao invés de *razão circunraio-menor aresta*.

Nas proximas seções descreveremos os algoritmos de refinamento Delaunay bidimensional propostos por Paul Chew e Jim Ruppert, utilizando um limitante superior para a razão cincunraio-menor aresta de uma malha triangular. Ambos os algoritmos trabalham com domínios representados por PSLG.

3.2 O algoritmo de Chew

O algoritmo descrito nesta seção foi publicado por Paul Chew e produz triangulações de densidade uniforme [6].

A idéia central do algoritmo proposto por Chew, assim como também de Ruppert, é inserir um vértice no cincuncentro de um triângulo com qualidade ruim. A propriedade de Delaunay é mantida utilizando a mesma estratégia do algoritmo de *Flipping* Incremental para triangulações de Delaunay. Dessa forma o triângulo ruim é eliminado, pois seu circuncentro não está mais vazio. O ato de inserir um vértice no circuncentro de um triângulo é chamado de "*splitting*".

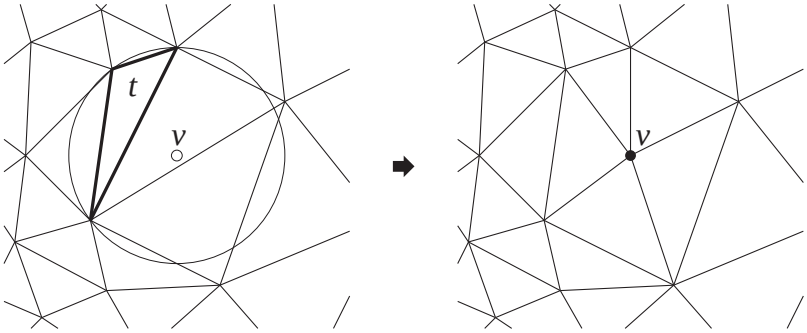


Figura 3.2: Inserção de um vértice no circuncentro de um triângulo com qualidade ruim. A propriedade de Delaunay é mantida e o triângulo ruim é eliminado [14].

A principal vantagem dos algoritmos de refinamento Delaunay é a garantia de que o algoritmo termina se a noção de "qualidade ruim" inclui somente os triângulos com razão circunraio-menor aresta maior do que um limitante superior B. Note que as novas arestas da triangulação de Delaunay criadas com a inserção de um vértice v são arestas incidentes ao vértice v (figura 3.2).

Como v é o cinrcuncentro de algum triângulo t, e não há nenhum vértice no interior do circuncírculo de t antes da inserção de v, nenhuma aresta pode ser menor que o circunraio de t. Como t tem razão circunraio-menor aresta maior que B, cada nova aresta tem tamanho pelo menos B vezes o tamanho da menor aresta de t.

Seja T uma triangulação cujo comprimento da menor aresta é h_{min}. O algoritmo de Chew refina a triangulação introduzindo o circuncentro de todo triângulo cujo circunraio é maior que h_{min}. Dessa forma o algoritmo nunca introduz arestas menores que h_{min}. O reflexo disso é uma triangulação aproximadamente uniforme, ou seja, cujos elementos possuem tamanhos parecidos. O fato de existir um tamanho de aresta mínimo na triangulação final garante que o algoritmo alcança um final, ou seja, não entra em loop infinito, como explicado anteriormente.

As arestas do bordo podem impedir que algum triângulo ruim seja eliminado, pois o circuncentro que deveria ser inserido se posiciona fora do domínio. A figura 3.3 ilustra este problema: o triângulo em negrito não pode ser eliminado pela inserção de um vértice no circuncentro. A região em cinza na figura 3.3 representa a região proibida para inserção de vértice, onde qualquer vértice estaria a uma distância menor que h_{min} de algum outro vértice (observe que a região cobre todo o triângulo problemático).

Para resolver este problema Chew propôs a seguinte estratégia. Supondo que a entrada seja um polígono G cujas arestas definem a região a ser triangulada, considere h (um parâmetro de entrada do algoritmo) tal que os segmentos de G sejam subdivididos em subsegmentos de comprimento l, onde $h \leq l \leq \sqrt{3}h$. O valor de h deve ser pequeno o suficiente para possibilitar a subdivisão dos segmentos de G. Além disso, h não deve ser maior do que a menor distância entre dois pontos provenientes da subdivisão, ou seja, as arestas adjacentes de G devem possuir um ângulo maior que 30°.

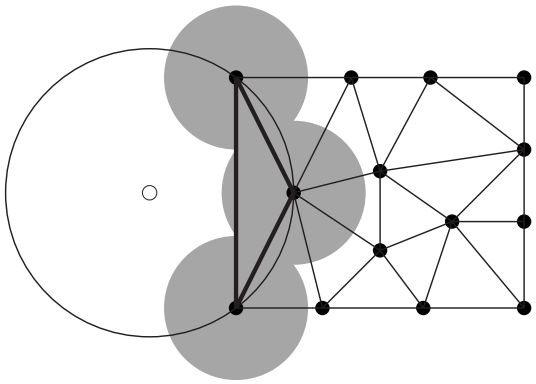


Figura 3.3: Problema do refinamento de Chew em um triângulo com aresta no bordo [14]

A estratégia de subdivisão de G garante que as arestas do polígono de entrada estejam presentes na triangulação final e que os circuncentros sempre estejam inseridos no interior do fecho convexo. O algoritmo 2 esquematiza o refinamento proposto por Chew.

Algoritmo 2 Refinamento Delaunay de Chew

Entrada:

G : PSLG

h : parâmetro de divisão de G

início

Dividir as arestas de G em segmentos de comprimento l tal que $h \leq l \leq \sqrt{3}h$, produzindo G'.

Criar a triangulação de Delaunay T dos vértices de G'.

enquanto existir triângulos $t \in T$ cujos circunraios sejam maiores que h **faça**

Inserir o circuncentro de t em T.

Atualizar T.

fim_enquanto

fim

3.3 O algoritmo de Ruppert

Assim como o algoritmo de Chew, o algoritmo de Ruppert refina a malha inserindo vértices até que todos os triângulos satisfaçam um limitante para a qualidade. A inserção dos vértices segue dois critérios:

- O círculo diametral de um segmento é o menor círculo que contém o segmento. Um segmento é dito *invadido* se um vértice está no interior do seu círculo diametral, ou se o segmento não aparece na triangulação. Quando um segmento é invadido, um vértice é imediatamente inserido no ponto médio deste segmento (figura 3.4). Os dois segmentos resultantes tem círculo diametral menor, e podem ou não estar invadidos.

- Como no algoritmo de Chew, cada triângulo ruim (com razão circunraio-menor aresta maior que um limitante B) é dividido normalmente pela inserção de um vértice no seu circuncentro. A propriedade de Delaunay garante que este triângulo é eliminado. No entanto, se o novo vértice invadir algum segmento, então ele não será inserido, e, em vez disso, todos os segmentos que estariam invadidos serão divididos.

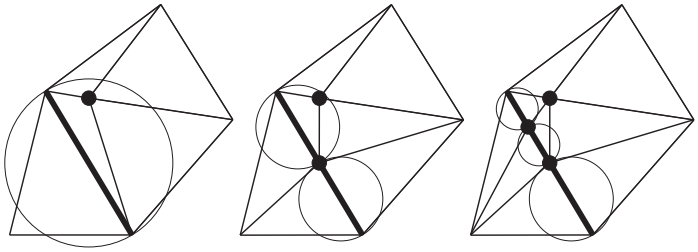


Figura 3.4: O segmento do PSLG (em negrito) é dividido recursivamente até que seus círculos diametrais sejam vazios de pontos [14]

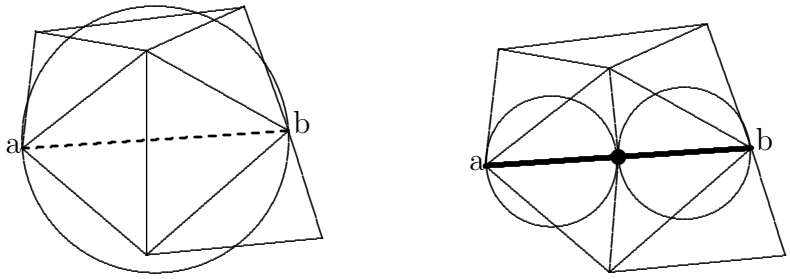
Os segmentos invadidos têm prioridade sobre os triângulos ruins.

Considerando o que foi dito acima, o algoritmo de Ruppert refina a triangulação por meio de dois passos básicos:

1. Recuperar todas as arestas do PSLG que não aparecem na triangulação de Delaunay inicial, gerada a partir dos vértices do PSLG.

2. Eliminar os triângulos ruins por meio de refinamento.

Quanto ao primeiro passo, suponha $\overline{ab}$ um segmento que não aparece na triangulação de Delaunay. Então, o círculo diametral de $\overline{ab}$ contém algum vértice do polígono em seu interior. Nesse caso, o ponto médio de $\overline{ab}$ é inserido na triangulação (figura 3.3).



(a) $\overline{ab}$ não pertence a triangulação, seu círculo diametral está invadido;

(b) Inserir o ponto médio de $\overline{ab}$ faz com que o segmento apareça na triangulação;

Figura 3.5: O segmento $\overline{ab}$ pertence ao PSLG, mas não aparece na triangulação.

Qualquer ponto dentro do círculo diametral de um segmento será chamado de invasor do segmento. Para executar o segundo passo básico do algoritmo, suponha t_{abc} um

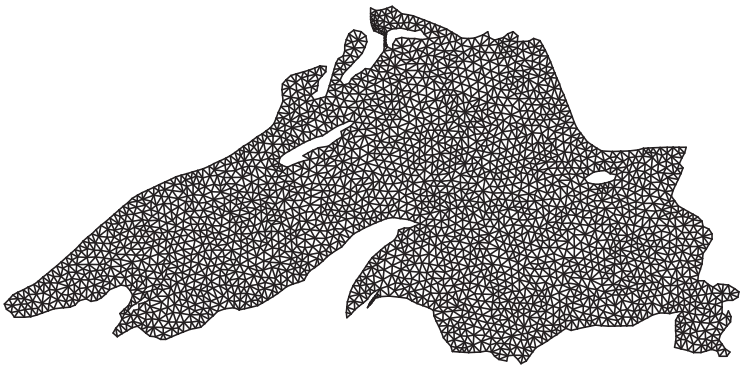
triângulo ruim, isto é, seu ângulo mínimo é menor que um limite inferior. O circuncentro p de t_{abc} será incluído na triangulação se não invadir qualquer subsegmento do PSLG. Se o circuncentro p for um invasor de um segmento s, então p não é inserido e o ponto médio de s é adicionado à triangulação.

Para manter a triangulação Delaunay a cada inserção de vértice, a estratégia utilizada pelo algoritmo de Ruppert é a mesma estratégia utilizada pelo algoritmo de *Flipping* Incremental. O algoritmo 3 esquematiza o refinamento de Ruppert.

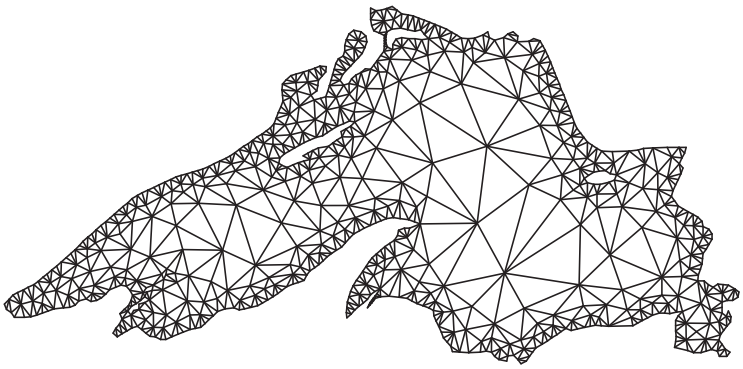
Algoritmo 3 Refinamento Delaunay de Ruppert

Entrada:
G : PSLG
α : qualidade de triângulo mínima aceitável
início
Gerar a triangulação de Delaunay T para os vértices de G.
enquanto existir um segmento $s \in G$ invadido **faça**
Dividir s em dois segmentos inserindo o ponto médio.
Atualizar G e T
fim_enquanto
enquanto existir algum triângulo $t \in T$ cuja qualidade seja inferior a α **faça**
Seja p o circuncentro de t, suponha que p invada os segmentos s_1, s_2, ..., s_k.
se $k \geq 1$ **então**
Dividir em dois os segmentos s_1, s_2, ..., s_k inserindo os k pontos médios.
Atualizar G e T.
senão
Inserir p.
Atualizar T.
fim_se
fim_enquanto
fim

Pode-se observar na figura 3.6 que o número de triângulos produzidos pelo algoritmo de Ruppert é tipicamente menor que o número de triângulos produzidos pelo algoritmo de Chew. Isto acontece porque os algoritmos de Chew e de Ruppert diferem quando se trata de analisar a qualidade de um triângulo para realizar o refinamento. No algoritmo de Chew a qualidade de um triângulo depende de seu formato (se está próximo ao equilátero, ou não) e de seu tamanho (se a área é grande, ou não), o que faz com que a malha apresente triângulos de mesmo tamanho em toda a malha. No algoritmo de Ruppert a qualidade de um triângulo depende apenas de seu formato, sendo que pode haver triângulos grandes (de área grande).



(a) Chew



(b) Ruppert

Figura 3.6: Malhas geradas pelos algoritmos de Chew (a) e Ruppert (b) [14].

Capítulo

4

Simplificação Delaunay

Considerando uma malha inicial que seja uma triangulação de Delaunay para um domínio Ω, a simplificação Delaunay desta malha busca remover da triangulação as arestas que falham em um critério pré-estabelecido. Para remover essas arestas faz-se a remoção do vértice inserido mais recentemente na triangulação, a fim de manter a propriedade de Delaunay da malha. O método a ser utilizado para a remoção de vértices de uma triangulação de Delaunay é discutido no trabalho de Devillers [8]. Algumas idéias que auxiliam no desenvolvimento do algoritmo são baseadas no texto de Xu et al.[18].

Seja T_k uma triangulação que é um subconjunto da triangulação de Delaunay T_n, ou seja, $T_k \subset T_n$, associado ao conjunto de pontos $A_k \subset A_n$. Segundo Xu et al.[18] uma propriedade que deriva diretamente da definição de triangulação de Delaunay é que T_k é uma triangulação de Delaunay.

Desta forma, para uma triangulação de Delaunay T_n associada com um conjunto de pontos A_n, suponha que um vértice, digamos $p_d \in A_n$, deve ser removido. Seja $T_{old} = S(p_d)$ o conjunto de todos os triângulos que contém o vértice p_d. Sejam A_{old} o conjunto de vértices associados a T_{old} e $A_{new} = A_{old} - p_d$. Note que p_d é ligado a todos os vértices de A_{new} via uma aresta dos triângulos em T_{old}, e que não há outro vértice em $A_n - A_{old}$ que seja ligado a p_d desta maneira. Considere ainda que T_{new} é uma triangulação de Delaunay gerada a partir dos vértices de A_{new} (ver figura 4.1).

O processo de remoção de um vértice consiste em primeiro remover T_{old} de T_n e depois adicionar T_{new} a T_n (no "buraco"deixado por T_{old}) formando uma nova triangulação T. A prova de que T é uma triangulação de Delaunay vem da demonstração do próximo teorema, que pode ser encontrada em [18]:

Teorema 1 *A triangulação $T = T_n - T_{old} + T_{new}$ é uma triangulação de Delaunay para o conjunto de vértices $A = A_n - p_d$*

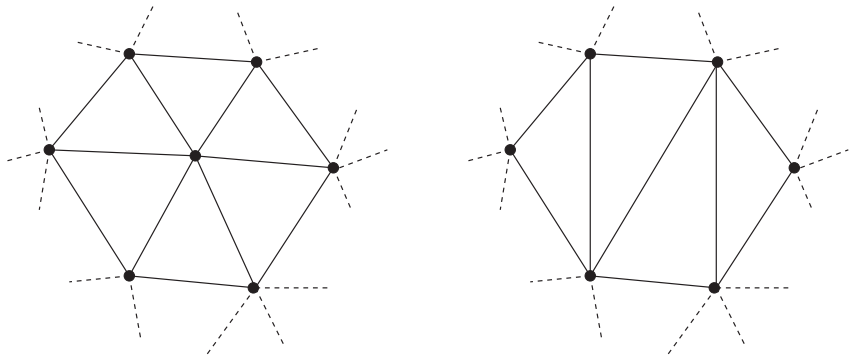


Figura 4.1: Triangulações (a) T_{old} a partir de A_{old} e (b) T_{new} a partir de A_{new}.

O teorema acima é válido, desde que a triangulação T_{new} seja uma triangulação de Delaunay, o que vem a ser o próximo problema a ser resolvido para implementar a remoção de pontos em uma malha Delaunay. Segundo Devillers [8], há vários métodos para garantir tal propriedade, no entanto, o algoritmo desenvolvido pelo mesmo autor se mostrou o mais eficiente. (¿¿¿¿¿¿ pq se motrou mais eficiente???)

O teorema seguinte é a base do algoritmo proposto por Devillers [8] para a triangulação de um buraco em uma malha Delaunay:

Teorema 2 *Considere um polígono* $P = \{p_0, p_1, \ldots, p_k = p_0\}$ *e um ponto* p_d *tal que as arestas* $p_i p_{i+1}$ *pertençam à triangulação de Delaunay de* $\{p_0, p_1, \ldots, p_d\}$. *Considere ainda* C *o circuncírculo de uma orelha* (p_i, p_{i+1}, p_{i+2}) *(definicão 2.24). Se* $|power(p_d, C)|$ *(definição 2.25) é mínimo dentre todas as orelhas de* P, *então* $p_i p_{i+2}$ *é uma aresta da triangulação de Delaunay de* $\{p_0, p_1, \ldots, p_{k-1}\}$.

Dessa forma a triangulação de um buraco em uma malha Delaunay pode ser implementada de maneira fácil, mantendo uma lista de prioridade das orelhas do polígono que definem o buraco, ordenadas pelo seu *power*. O algoritmo 4 descreve os passos para remover um vértice da triangulação triangulando o buraco deixado na malha Delaunay.

Algoritmo 4 Remove Vértice

Entrada:

T : Triangulação de Delaunay

p_d : vértice a ser removido

início

Seja $p_0, p_1, \ldots, p_{k-1}$ os vértices incidentes a p_d em T no sentido anti-horário.

Seja Q uma fila de prioridade.

para $i = 0$ **até** $k - 1$ **faça**

se $ear = (p_i, p_{i+1}, p_{i+2})$ é uma orelha;

então $p \leftarrow |power(p_d, C_{ear})|$;

senão $p \leftarrow \infty$;

Q.insert(p, ear); $//insert(prioridade, chave)$

enquanto o número de elementos em Q for maior que 3 **faça**

$ear \leftarrow Q$.minimo();

Insere a aresta $p_i p_{i+2}$ na triangulação criando assim um novo triângulo em T;

$ear0 \leftarrow ear$.anterior;

$ear1 \leftarrow ear$.posterior;

$ear0$.vertice(2) $\leftarrow ear$.vertice(2); $ear0$.posterior $\leftarrow ear1$;

$ear1$.vertice(0) $\leftarrow ear$.vertice(0); $ear1$.anterior $\leftarrow ear2$;

Q.delete(ear)

Q.modifica-prioridade$(ear0)$;

Q.modifica-prioridade$(ear1)$;

fim_enquanto

As últimas três orelhas restantes formam um mesmo triângulo que deve ser inserido na triangulação;

fim

Capítulo

5

Isotropia e Anisotropia de Malhas

Uma malha é considerada boa se oferecer uma boa aproximação linear por partes g para uma função f que procura discretizar, isto é, se as curvaturas calculadas a partir de f nos vértices da malha forem próximas. Se para qualquer ponto p no domínio da função f as curvaturas tomadas em todas as direções forem próximas, uma malha isotrópica geralmente é usada para aproximar f. Entretanto, em algumas aplicações tais funções tem grandes variações nas curvaturas em diferentes direções. É possível obter uma boa precisão com elementos próximos ao equilátero, mas podemos obter a mesma precisão com poucos elementos se utilizarmos uma malha *anisotrópica*. Em uma malha anisotrópica, os elementos longos e finos são orientados na direção em que a curvatura é pequena. Em muitas aplicações o uso de malhas anisotrópicas representa o melhor compromisso entre precisão (menor erro de aproximação) e desempenho (menos elementos para processar). Existem aplicações nas quais a geração ou o uso de uma malha isotrópica fina o bastante para obter uma solução precisa torna-se computacionalmente impraticável devido ao número de elementos muito grande da malha. No entanto, uma malha anisotrópica pode oferecer uma boa precisão com menos elementos, melhorando assim o desempenho.

O método mais conhecido para julgar se um elemento anisotrópico é adequado, segundo D'Azevedo[7], é construir um mapeamento afim que leva elementos do espaço físico (anisotrópico) para o espaço isotrópico, onde o limitante para a curvatura em um ponto é isotrópico. Um elemento fino no espaço físico que se torna equilátero quando mapeado para o espaço isotrópico é ideal para minimizar o erro de interpolação $\|f - g\|$.

Para julgar ou selecionar elementos anisotrópicos, deve-se caracterizar os limitantes direcionais sobre a curvatura da função $f(p)$ a ser interpolada. Para isto usaremos uma matriz simétrica definida positiva M_t chamada de *tensor de curvatura* para o elemento t.

5.1 O Tensor de Curvatura

A curvatura em um vértice de uma malha definida em um domínio Ω pode ser especificada por uma função f definida em Ω. Dado um ponto $x \in \Omega$, os valores calculados a partir de f em x podem ser utilizados para determinar o tamanho de um elemento nas proximidades de x, bem como seu formato e orientação. A seguir serão apresentadas as condições numéricas sobre f para considerar anisotropia nos elementos.

Em simulações numéricas, por exemplo, as condições numéricas são geralmente obtidas por meio de estimadores de erros *a priori* ou *a posteriori* baseados em uma simulação inicial. Isto define uma função de curvatura para cada ponto x no domínio Ω por meio de um tensor métrico M. Será utilizado M_x para designar o tensor métrico M definido no ponto $x \in \Omega$. O tensor M_x pode ser usado para quantificar o tamanho, formato e orientação de um elemento próximo a um ponto $x \in \Omega$. O tensor métrico provém da geometria riemanniana e define anisotropia determinando como comprimentos e ângulos devem ser medidos.

O tensor métrico M é uma matriz $d \times d$ simétrica definida positiva (com autovalores positivos ξ_i e autovetores $\vec{v}_i$, $i = 1, \ldots, d$, ortogonais entre si). No caso bidimensional, M é definida por:

$$\begin{aligned} M &= V \Xi V^t \\ &= \begin{bmatrix} \vec{v}_1 & \vec{v}_2 \end{bmatrix} \begin{bmatrix} \xi_1 & 0 \\ 0 & \xi_2 \end{bmatrix} \begin{bmatrix} \vec{v}_1 & \vec{v}_2 \end{bmatrix}^t \\ &= \xi_1 \vec{v}_1 \vec{v}_1^t + \xi_2 \vec{v}_2 \vec{v}_2^t \end{aligned} \tag{5.1}$$

onde $\vec{v}_1$ e $\vec{v}_2$ são tomados como unitários.

Na geometria Riemanniana, as operações geométricas básicas são redefinidas. O produto interno e o produto vetorial entre dois vetores $\vec{a}$ e $\vec{b}$ são dados respectivamente por:

$$\vec{a} \odot \vec{b} = \vec{a}^t M \vec{b} \tag{5.2}$$

$$\vec{a} \otimes \vec{b} = \sqrt{|M|}(\vec{a} \times \vec{b}) \tag{5.3}$$

onde $\vec{a} \times \vec{b} = a_x b_y - a_y b_x$ é o produto vetorial bidimensional.

5.2 Medições no Espaço Anisotrópico

5.2.1 Computando Distâncias

Na geometria Riemanniana, o comprimento de uma curva paramétrica $S(t)$ entre pontos x_i e x_j, onde $t \in [0,1]$, $S(0) = x_i$, e $S(1) = x_j$ é definido por:

$$l(S) = \int_0^1 \sqrt{\frac{dS}{dt}^T M_{S(t)} \frac{dS}{dt}} dt \tag{5.4}$$

A distância entre dois pontos é dada pela menor curva paramétrica S, porém medir esta curva em uma métrica arbitrária não é trivial. É possível aproximar o cálculo da distância computando o comprimento de uma curva mais simples. Integrando ao longo do segmento de reta $S(t) = x_i + t \cdot (x_j - x_i)$ parece razoável. Se M varia linearmente entre x_i e x_j então a distância entre dois pontos pode ser definida por [3, 5]:

$$\begin{aligned} d_{i,j} &= \int_0^1 \sqrt{(\vec{r}_{ij})^T (M_i + t(M_j - M_i))(\vec{r}_{ij})} dt \\ &= \int_0^1 \sqrt{(d_i^{ij})^2 + t((d_j^{ij})^2 - (d_i^{ij})^2)} dt \\ &= \frac{2}{3} \frac{(d_i^{ij})^2 + d_i^{ij} d_j^{ij} + (d_j^{ij})^2}{d_i^{ij} + d_j^{ij}} \end{aligned} \tag{5.5}$$

onde

$$d_k^{ij} = \sqrt{(\vec{r}_{ij})^T M_k (\vec{r}_{ij})} \tag{5.6}$$

é a distância de x_i a x_j segundo a métrica M_k definida em um ponto k do domínio; $\vec{r}_{ij}$ é o vetor saindo de x_i para x_j.

5.2.2 Computando Áreas

A área de um domínio Ω é definida por

$$A(\Omega) = \int \int_\Omega \sqrt{det(M_x)} dx_1 dx_2 \tag{5.7}$$

Portanto, seria possível resolver a integral para um triângulo $\triangle_{abc}$, conhecendo a métrica em cada vértice, e interpolando linearmente entre eles. Uma aproximação para o cálculo de áreas é dada por[3]:

$$A(\triangle_{abc}) = \frac{1}{2} \sqrt{det\left(\frac{M_a + M_b + M_c}{3}\right)} \cdot (b - a) \times (c - a) \tag{5.8}$$

5.3 A matriz Hessiana

O tensor métrico M geralmente expressa os efeitos do formato do elemento no erro de interpolação. Portanto, M_p geralmente está relacionada com a matriz hessiana de f no ponto $p \in \Omega$, podendo inclusive ser a própria hessiana.

Seja H_p a matriz hessiana de f no ponto $p \in \Omega$.

A *curvatura* da função f no ponto p ao longo de um vetor $\vec{d}$ qualquer, é a segunda derivada direcional ao longo de $\vec{d}$, $f''_{\vec{d}}(p)$. Especificamente, seja p de coordenadas (x, y), e considere a matriz Hessiana:

$$H_p = \begin{bmatrix} \frac{\partial^2}{\partial x^2} f(p) & \frac{\partial^2}{\partial x \partial y} f(p) \\ \frac{\partial^2}{\partial x \partial y} f(p) & \frac{\partial^2}{\partial y^2} f(p) \end{bmatrix} \tag{5.9}$$

Cada ponto ou vetor p é tratado como um vetor $d \times 1$ e o seu transposto p^t como um vetor $1 \times d$. Para qualquer vetor unitário $\vec{d}$ a curvatura direcional é dada por $f''_{\vec{d}}(p) = \vec{d}^t H_p \vec{d}$, expressa na seguinte multiplicação:

$$f''_{\vec{d}}(p) = \vec{d}^t H_p \vec{d} \tag{5.10}$$

$$= \begin{bmatrix} d_x & d_y \end{bmatrix} \begin{bmatrix} \frac{\partial^2}{\partial x^2} f(p) & \frac{\partial^2}{\partial x \partial y} f(p) \\ \frac{\partial^2}{\partial x \partial y} f(p) & \frac{\partial^2}{\partial y^2} f(p) \end{bmatrix} \begin{bmatrix} d_x \\ d_y \end{bmatrix} \tag{5.11}$$

Se $\vec{d}$ não é um vetor unitário, é fácil mostrar que $f''_{\vec{d}}(p) = \vec{d}^t H_p \vec{d} / |\vec{d}|^2$.

Suponha que f satisfaz, de alguma forma, a seguinte restrição, para qualquer ponto $p \in \Omega$ e qualquer vetor direção $\vec{d}$:

$$|\vec{d}^t H_p \vec{d}| \leq \vec{d}^t M_p \vec{d} \tag{5.12}$$

Esta inequação implica que a curvatura de f no ponto p ao longo da direção $\vec{v}_i$ não excede ξ_i.

Considere um ponto $p \in \Omega$. Podemos construir o tensor métrico M_p a partir de um conjunto de autovalores e autovetores. Logo, estimando os valores da matriz Hessiana H_p, pode-se gerar M_p extraindo os autovalores e autovetores de H_p. Como afirmado anteriormente, diferente da Hessiana, a matriz M_p deve ser positiva definida, portanto, seus autovalores ξ_i devem ser positivos, e servem como limitantes superiores para a magnitude das curvaturas de f no ponto p tomadas ao longo das direções definidas pelos autovetores $\vec{v}_i$. Note que se os autovalores ξ_i forem iguais a 1 para $i = 1, \ldots, d$ então o tensor métrico M_p é igual a matriz identidade, e o espaço é euclidiano.

A anisotropia em um ponto $p \in \Omega$ pode ser visualizada pelas curvas de nível da função $h(x) = x^t M_p x$, $x \in \Omega$ (figura 5.3). Restringindo-se apenas ao elipsóide $x^t M_p x = 1$

com eixos $\vec{v}_i$ e raios $1/\sqrt{\xi_i}$ respectivamente, temos uma simplificação desta equação. Um exemplo bidimensional da representação por elipsóide pode ser visto na figura 5.1.

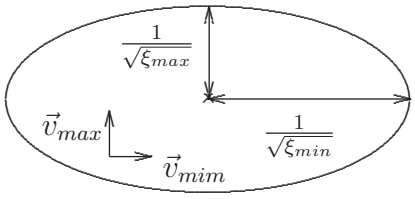


Figura 5.1: Anisotropia representada por uma elipse no caso bidimensional.

Sejam ξ_{min} e ξ_{max} o menor e o maior autovalores de M_p e $\vec{v}_{min}$ e $\vec{v}_{max}$ seus respectivos autovetores unitários. Para qualquer vetor unitário $\vec{d}$, tem-se que:

$$\vec{d}^t M_p \vec{d} \leq \xi_{max} \tag{5.13}$$

assim, a curvatura máxima de f no ponto $p \in \Omega$ é ξ_{max}.

5.4 O tensor de transformação

Seja E o tensor de transformação que faz o mapeamento de um ponto p no espaço físico (o domínio onde a função f está definida) em outro ponto $\widehat{p}$ no espaço isotrópico. O tensor é dado por [15]:

$$E^2 = \frac{1}{\xi_{max}} M \tag{5.14}$$

No caso bidimensional, o tensor de transformação E fica:

$$E = V \begin{bmatrix} \sqrt{\xi_1/\xi_{max}} & 0 \\ 0 & \sqrt{\xi_2/\xi_{max}} \end{bmatrix} V^t \tag{5.15}$$

Portanto $\widehat{p} = E.p$ é a imagem no espaço isotrópico de um ponto p qualquer no espaço físico. Seja $\widehat{t}$ a imagem no espaço isotrópico de um triângulo anisotrópico t.

Seja ψ a constante de condicionamento do tensor métrico M dada por:

$$\psi = \xi_{max}/\xi_{min} \tag{5.16}$$

Seja Ψ o fator de encolhimento de uma aresta quando t é transformado em $\widehat{t}$ pela aplicação de E. O valor de Ψ pode ser de (figura 5.2):

- no máximo $\sqrt{\psi}$, quando a aresta for paralela a $\vec{v}_{min}$;
- no mínimo 1, quando a aresta for paralela a $\vec{v}_{max}$.

Portanto $1 \leq \Psi \leq \sqrt{\psi}$. Observe que Ψ mede o encolhimento, por conseguinte o fator escala é dado por $1/\Psi$.

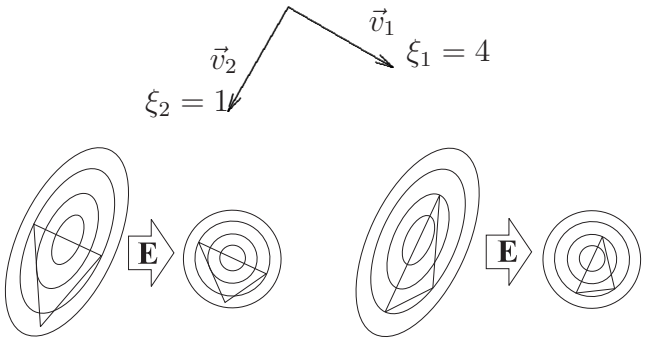


Figura 5.2: Fator de Encolhimento de uma aresta.

O tensor de transformação E pode ser simplificado se não houver a necessidade de preservar a curvatura máxima da função f. No caso bidimensional, o tensor métrico simplificado F é definido por:

$$F = \sqrt{M} \tag{5.17}$$

Assim o tensor de transformação F, é dado por (caso bidimensional):

$$F = V \begin{bmatrix} \sqrt{\xi_1} & 0 \\ 0 & \sqrt{\xi_2} \end{bmatrix} V^t \tag{5.18}$$

Como $\vec{v}_i$ são vetores unitários ortogonais entre si, tem-se que $V^t V = I$, sendo I a matriz identidade, portanto:

$$F^2 = M \tag{5.19}$$

Da mesma forma como M_p representa o tensor métrico M definido no ponto $p \in \Omega$, E_p e F_p designarão tensores de transformação referentes à métrica M definida no ponto $p \in \Omega$.

5.5 O tensor de deformação relativa

Seja um ponto $p \in \Omega$. Se o tensor de transformação F_p for aplicado, diz-se que o espaço é visto segundo a ótica do ponto p. O tensor de deformação relativa $F_q F_p^{-1}$ transforma o espaço a partir da ótica de p para a ótica de q [10], ou seja, faz com que o espaço passe a ser visto segundo a ótica do ponto q. A figura 5.3 ilustra o relacionamento entre os tensores de deformação relativa $F_q F_p^{-1}$ e $F_p F_q^{-1}$ e os tensores de transformação F_q e F_p.

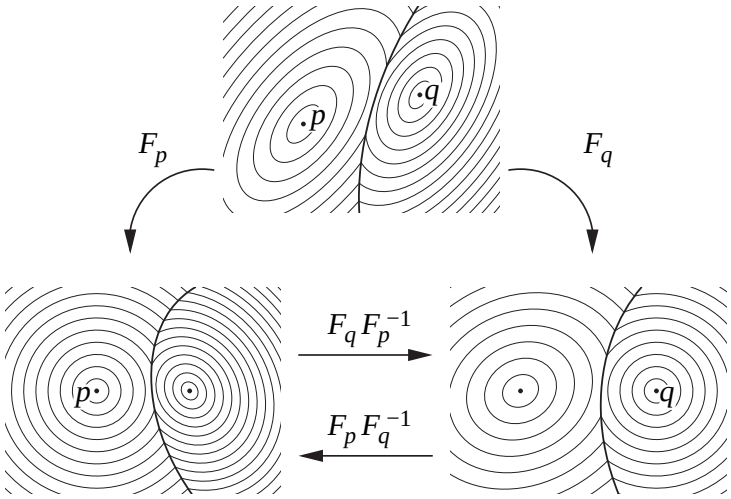


Figura 5.3: Tensores de deformação e tensores de deformação relativa [10].

5.6 A distorção relativa

A distorção relativa $\tau(p,q) = max\|F_q F_p^{-1}\|_2, \|F_p F_q^{-1}\|_2$ fornece um limitante superior de como diferentemente p e q percebem distâncias [10]. Para quaisquer pontos p, q e a, $\tau(p,q) \geq 1$, $\tau(p,q) = \tau(q,p)$ e $\tau(p,q) \leq \tau(p,a)\tau(a,q)$.

Sejam p, q, a e b pontos no domínio Ω, então vale que [10]:

$$\frac{d_p(a,b)}{\|F_p F_q^{-1}\|_2} \leq d_q(a,b) \leq d_p(a,b).\|F_q F_p^{-1}\|_2 \tag{5.20}$$

e

$$\frac{d_p(a,b)}{\tau(p,q)} \leq d_q(a,b) \leq d_p(a,b).\tau(p,q) \tag{5.21}$$

5.7 O critério de Delaunay Modificado

O critério Delaunay original é baseado no teste da circunsfera. Se nenhum vértice estiver dentro da circunsfera de um elemento t, então afirma-se que t é válido quanto ao teste da circunsfera. O critério Delaunay modificado é a versão anisotrópica do teste da circunsfera, para isso, o teste original é aplicado somente após o mapeamento dos vértices para o espaço isotrópico por meio do tensor de transformação (5.4).

Sejam dois triângulos adjacentes t_{xyz} e t_{zwx}. O algoritmo para contrução de uma triangulação Delaunay faz *flipping* de arestas a fim de maximizar o ângulo mínimo, ou seja, se $(\angle zyx + \angle xwz) > 180°$ então ocorre o *flipping* entre arestas xz e yw (ver figura 5.4). Para uma triangulação Delaunay anisotrópica, a medida dos ângulos deve ocorrer no espaço normalizado definido pela métrica. A regra geral para verificar se deve ocorrer

o *flipping* entre as arestas xz e yw é dada por:

$$[(z-y)\times(x-y)]\,(x-w)^t M(z-w)+(z-y)^t M(x-y)\,[(x-w)\times(z-w)]<0 \quad (5.22)$$

onde $M=(M_x+M_y+M_z+M_w)/4$ e x, y, z e w são os quatro vértices do quadrilátero composto pelos triângulos t_{xyz} e t_{zwx} [4].

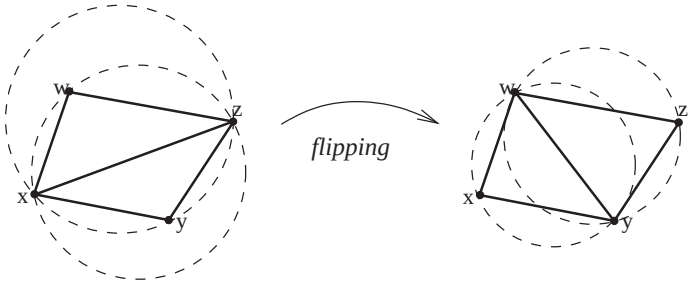


Figura 5.4: *Flipping* entre arestas.

Capítulo

6

Implementações

Este capítulo aborda os detalhes principais dos algoritmos implementados neste trabalho. Para implementar os algoritmos citados foi adotada a linguagem de programação *C++* com orientação a objeto e utilização de *templates*.

A seção 6.1 apresenta a estrutura utilizada no desenvolvimento do trabalho. Na seção 6.2 é apresentado a estratégia utilizada para o Refinamento de Ruppert. A seção 6.3 apresenta o algoritmo utilizado para gerar o refinamento anisotrópico.

6.1 Estrutura de dados

A estrutura de dados utilizada chama-se OF, que quer dizer "**O**pposite **F**ace". Esta estrutura de dados foi inicialmente baseada na estrutura *Corner Table* [12] que é utilizada principalmente para compressão de malhas tridimensionais e também na estrutura *Cgal* [2]. Esta estrutura foi inicialmente desenvolvida no LCAD - Laboratório de Computação de Alto Desempenho do ICMC/USP - São Carlos.

A estrutura OF foi definida com a preocupação de minimizar o consumo de memória necessário para a representação da malha. Outra preocupação foi prover facilidade de utilização e flexibilidade.

6.1.1 A Estrutura de dados "*OF*"

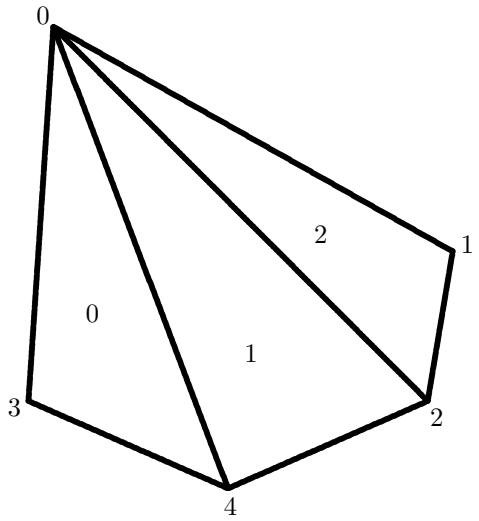
Uma estrutura de dados topológica precisa ser capaz de representar todos os elementos de uma malha e as relações entre eles, porém utilizando-se do menor espaço de memória possível. Como em alguns outros problemas computacionais, a relação de consumo de memória e de necessidade de processamento para a interpretação dos dados podem ser

inversamente proporcionais, portanto a preocupação com o consumo de memória deve considerar o impacto causado no desempenho.

Considerando a restrição de memória, podemos representar um elemento de uma malha de maneira explícita ou implícita. A maneira explícita é a mais comum, visando a representação do elemento por meio de uma variável, estrutura ou objeto. No outro modo de representação, obtemos o elemento desejado indiretamente, deduzindo-o de um elemento explícito como, por exemplo, por meio de métodos. Comparando-se as duas representações, a implícita mostra-se mais vantajosa, pois economiza memória e, em alguns casos, simplifica o modelo na medida em que o número de objetos alocados é reduzido. A estrutura OF faz uso dos dois modelos de representação, ou seja, os vértices, triângulos e tetraedros são representados explicitamente; as arestas e faces são representadas implicitamente.

Os vértices e as células (triângulos ou tetraedros) são identificados por um número – que chamaremos de identificador – inteiro, positivo e único para cada elemento. Com isso, podemos acessar diretamente um determinado elemento se associarmos a posição deste na memória com seu identificador.

Os vértices são formados pelas suas coordenadas geométricas e pelo identificador de uma de suas células incidentes. Por sua vez, nas células temos os identificadores dos vértices que a formam e também os de suas células vizinhas. A relação de vizinhança considera as arestas da célula. Desta forma, no espaço bidimensional, uma célula c_1 é dita ser vizinha da célula c_2 por aresta em relação ao vértice V, se e somente se, $c_1 \cap c_2$ for a aresta que não contém o vértice V. A figura 6.1 ilustra como são representados vértices e células na estrutura OF, bem como as relações de vizinhança de cada célula.



Vertice	Coordenadas	Celula
0	x0,y0	0
1	x1,y1	2
2	x2,y2	1

Celula	Vertice	Celula oposta
0	0	—
	3	1
	4	—
1	0	—
	4	2
	2	0
2	2	—
	1	1
	0	—

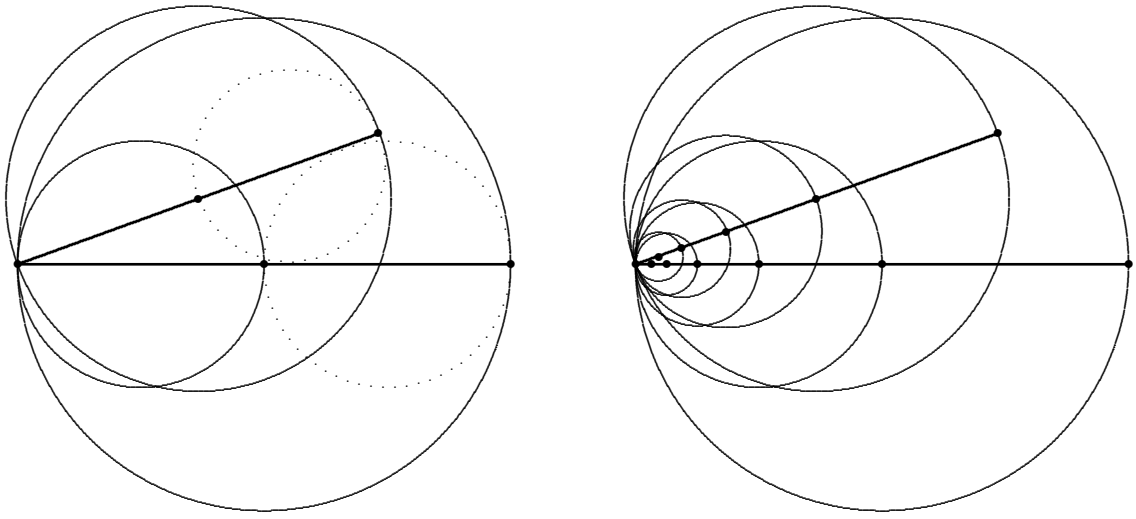
Figura 6.1: Exemplo de vértices, células e relações de vizinhança representados na estrutura OF;

6.2 Estratégia de Refinamento de Ruppert

A estratégia utilizada para implementar o algoritmo de refinamento de Ruppert foi baseado no algoritmo descrito por Jonathan Richard Shewchuk [14] citado no capítulo 3. A única diferença está no tratamento dos ângulos pequenos que pertencem às arestas de bordo da triangulação. O tratamento de ângulos pequenos foi baseado nos estudos de Steven Elliot Pav [11], que determina uma forma diferente e mais eficiente de tratar os ângulos pequenos que podem ocorrer nas arestas que definem o bordo da triangulação.

Se um PSLG contém ângulos pequenos (menor que 45, utilizado por Jonathan Richard Shewchuk [14]), o refinamento de Ruppert aplicado a este PSLG não funcionará. Isto acontece devido à restrição do algoritmo de que não pode haver segmentos do PSLG invadidas por algum vértice pertencente à triangulação, ou seja, não podem haver vértices no interior de um círculo diametral de um segmento que pertence ao PSLG (seção 3.3).

Em casos onde o PSLG contém ângulos pequenos, os vértices dos segmentos que formam um ângulo pequeno podem estar localizados no interior do círculo diametral do segmento adjacente, necessitando assim que o algoritmo insira um vértice no ponto médio do segmento invadido (ver figura 6.2a). Neste caso, tal vértice fará com que aconteça um ciclo ininterrupto de inserções de vértices para tentar eliminar segmentos invadidos (figura 6.2b), fazendo com que o algoritmo de Ruppert entre em um ciclo infinito.



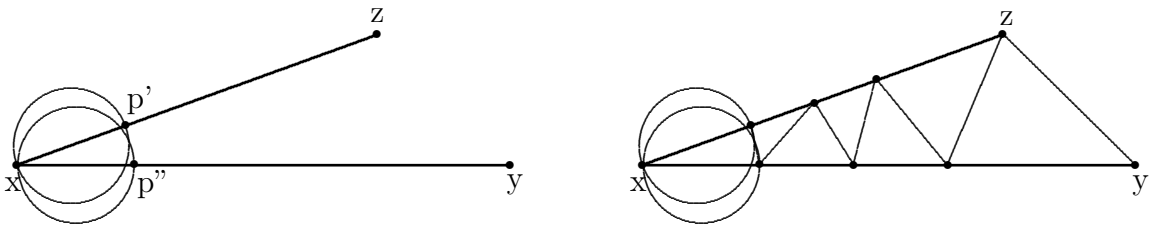
(a) Exemplo de “invasão” de aresta; o círculo pontilhado indica que a aresta não está invadida;

(b) Invasões subsequentes: o processo ocorre infinitamente;

Figura 6.2: Tratamento de ângulos pequenos

Para resolver este problema Pav [11] sugere um pré-processamento realizado sobre o PSLG para tratar possíveis ângulos pequenos. Neste pré-processamento são adicionados vértices ao PSLG localizados sobre os segmentos que formam um ângulo menor que 45.

Este pré-processamento funciona da seguinte forma: suponha que exista um ângulo $\alpha < 45$ no PSLG em que o vértice x é o eixo deste ângulo, e os segmentos $\overline{xy}$ e $\overline{xw}$ são incidentes ao ângulo α. Seja l o comprimento do menor destes segmentos. O algoritmo de pré-processamento insere um vértice p em cada segmento tal que $|x - p| = \frac{2}{1+\sqrt{13}}l$ (ver figura 6.3a). Desta maneira os vértices novos quebram os segmentos existentes formando dois novos segmentos com tamanhos iguais, e assim, isolando o vértice eixo do ângulo. Com estes novos segmentos o algoritmo de refinamento de Ruppert elimina as arestas invadidas sem gerar um ciclo infinito (figura 6.3b). Pav [11] mostra que para a distância $\frac{2}{1+\sqrt{13}}l$, o refinamento gera uma malha com melhores resultados que o algoritmo utilizado por Shewchuk [14].



(a) As inserções de p' e p'' isolam o ângulo $y\widehat{x}z$;

(b) O ciclo de inserções ininterruptas não acontece;

Figura 6.3: Pré-processamento sobre os ângulos menores que 45 do PSLG;

Os novos segmentos que formam o ângulo pequeno são marcados para evitar que sejam quebrados devido à invasão de algum vértice gerado pelo refinamento de Ruppert. Caso o algoritmo de Ruppert tente refinar um triângulo gerando um vértice que invade estes novos segmentos, o algoritmo detecta que é um segmento pertencente a um ângulo pequeno e então ignora este vértice para o refinamento.

6.3 O Refinamento Anisotrópico

No algoritmo implementado para obter um Refinamento Anisotrópico, estamos considerando a anisotropia constante em todo o domínio Ω, ou seja, para qualquer ponto $x \in \Omega$ temos apenas um único tensor métrico M.

Como o tensor de transformação E mapeia um ponto p no espaço anisotrópico para um ponto $\widehat{p}$ no espaço isotrópico, definimos o tensor de transformação inverso E^{-1} que mapeia pontos do espaço isotrópico para o espaço anisotrópico. Com isto podemos aplicar o tensor E para mapear o PSLG para o espaço isotrópico, executar o algoritmo de refinamento isotrópico, e então, aplicando o tensor E^{-1}, retornar a malha gerada pelo refinamento ao espaço anisotrópico. O algoritmo é simples e está descrito logo a seguir.

Algoritmo 5 Refinamento anisotrópico

Entrada:

G : PSLG

B : qualidade de triângulo mínima aceitável

M : tensor métrico

início

Obter G', o mapeamento de G para o espaço isotrópico, aplicando o tensor de transformação E sobre G.

Executar o algoritmo de refinamento isotrópico com entrada G e qualidade B, gerando uma malha T'.

Obter T no espaço anisotrópico aplicando o tensor de transformação inverso E^{-1} sobre T'.

fim

Capítulo

7

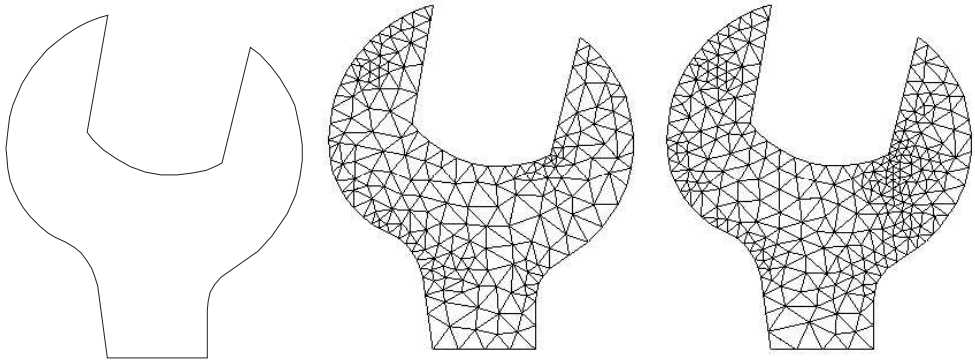
Resultados

Neste capítulo apresentamos uma coleção dos resultados obtidos neste trabalho.

7.1 Resultados

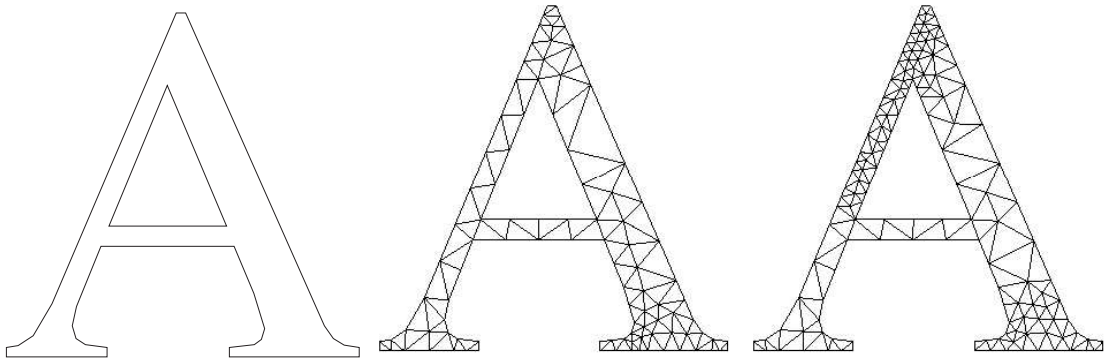
7.1.1 Resultados gerados pelo algoritmo de Refinamento Isotrópico

As figuras abaixo foram geradas pelo algoritmo 3 com diferentes valores de B, utilizando o tratamento de ângulos pequenos descrito na seção 6.2. As figuras 7.4, 7.5 e 7.6 foram geradas a partir de dados extraidos da área da represa Caledônia da empresa FURNAS Centrais Elétricas S.A. situada no estado do Rio de Janeiro.



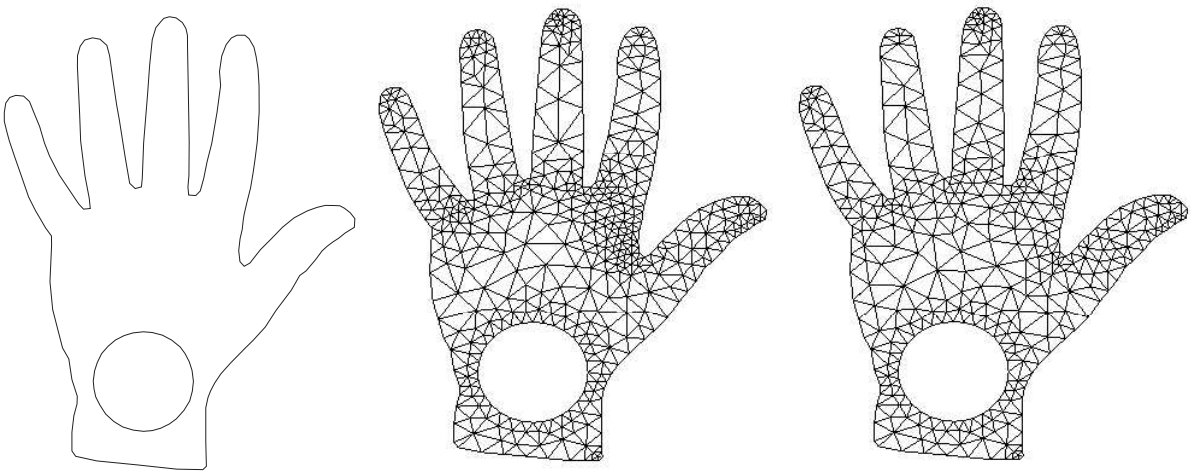
(a) (b) (c)

Figura 7.1: (a) PSLG; (b) $B = 0.97$; (c) $B = 0.93$



(a) (b) (c)

Figura 7.2: (a) PSLG; (b) $B = 0.97$; (c) $B = 0.93$



(a) (b) (c)

Figura 7.3: (a) PSLG; (b) $B = 0.95$; (c) $B = 1.0$

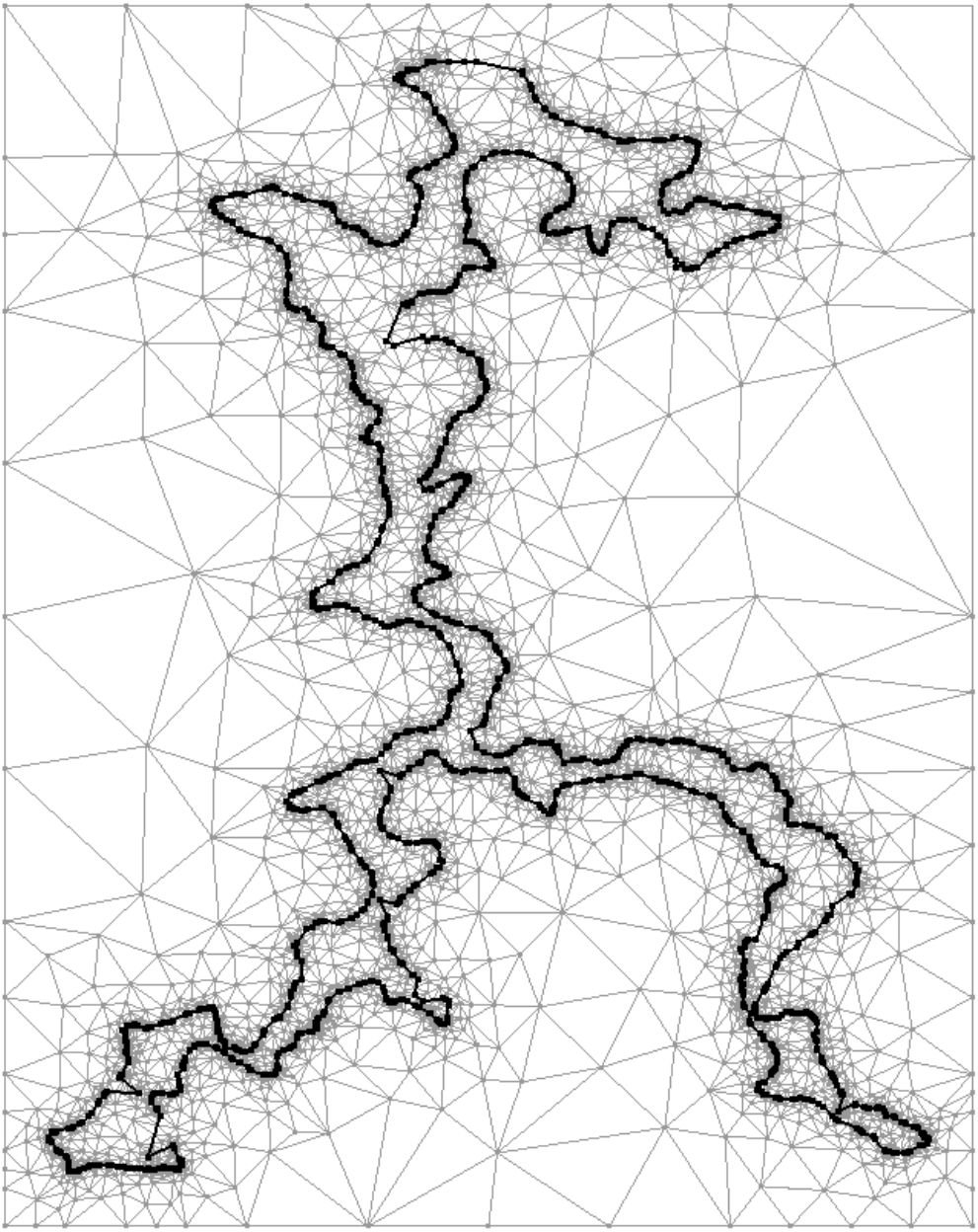


Figura 7.4: Refinamento realizado sobre uma curva de nível (em negrito) de valor 1045 metros; nº de triângulos = 13729; nº de vértices = 6893; $B = \sqrt{2}$.

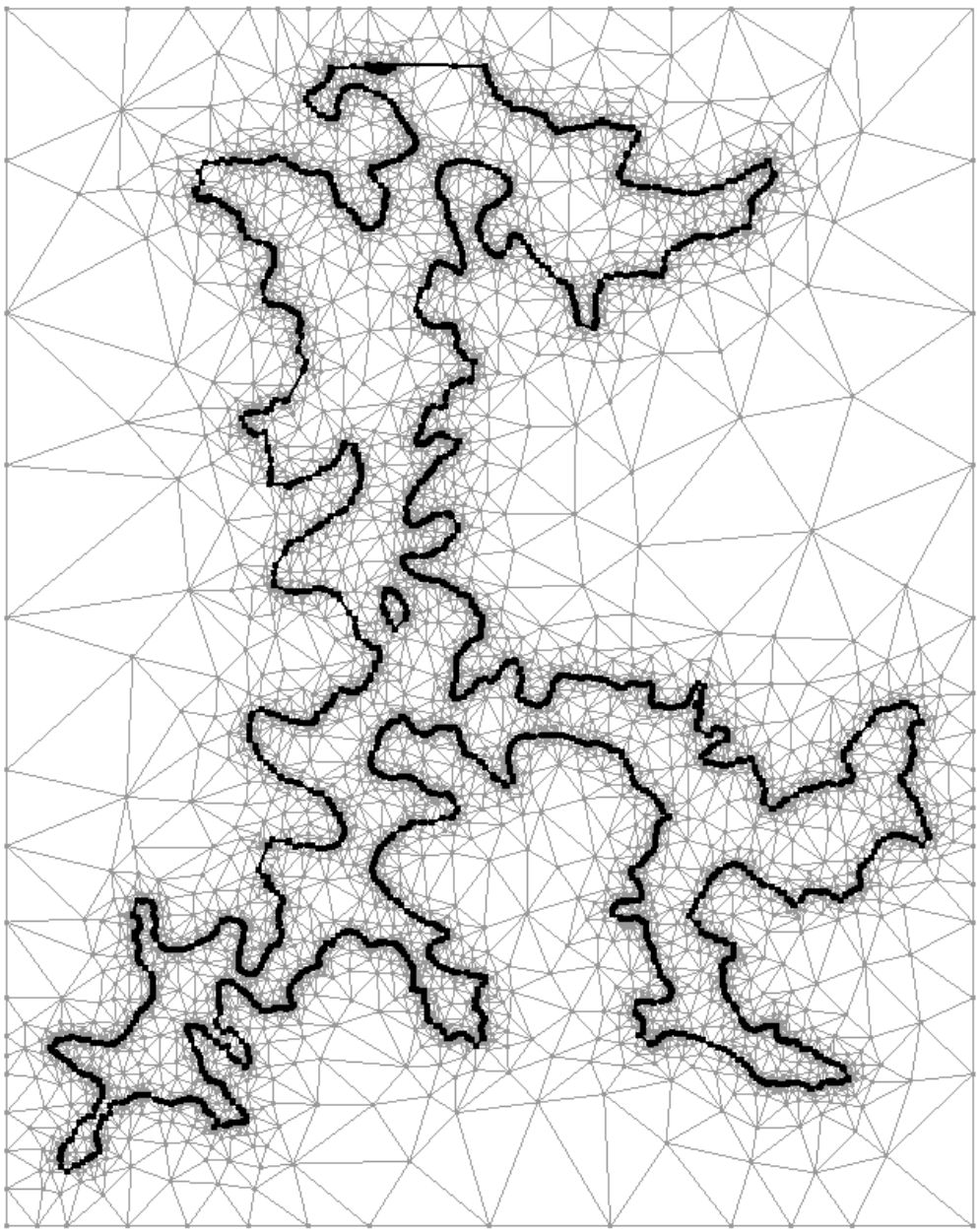


Figura 7.5: Refinamento realizado sobre uma curva de nível (em negrito) de valor 1070 metros; nº de triângulos = 16050; nº de vértices = 8056; $B = \sqrt{2}$.

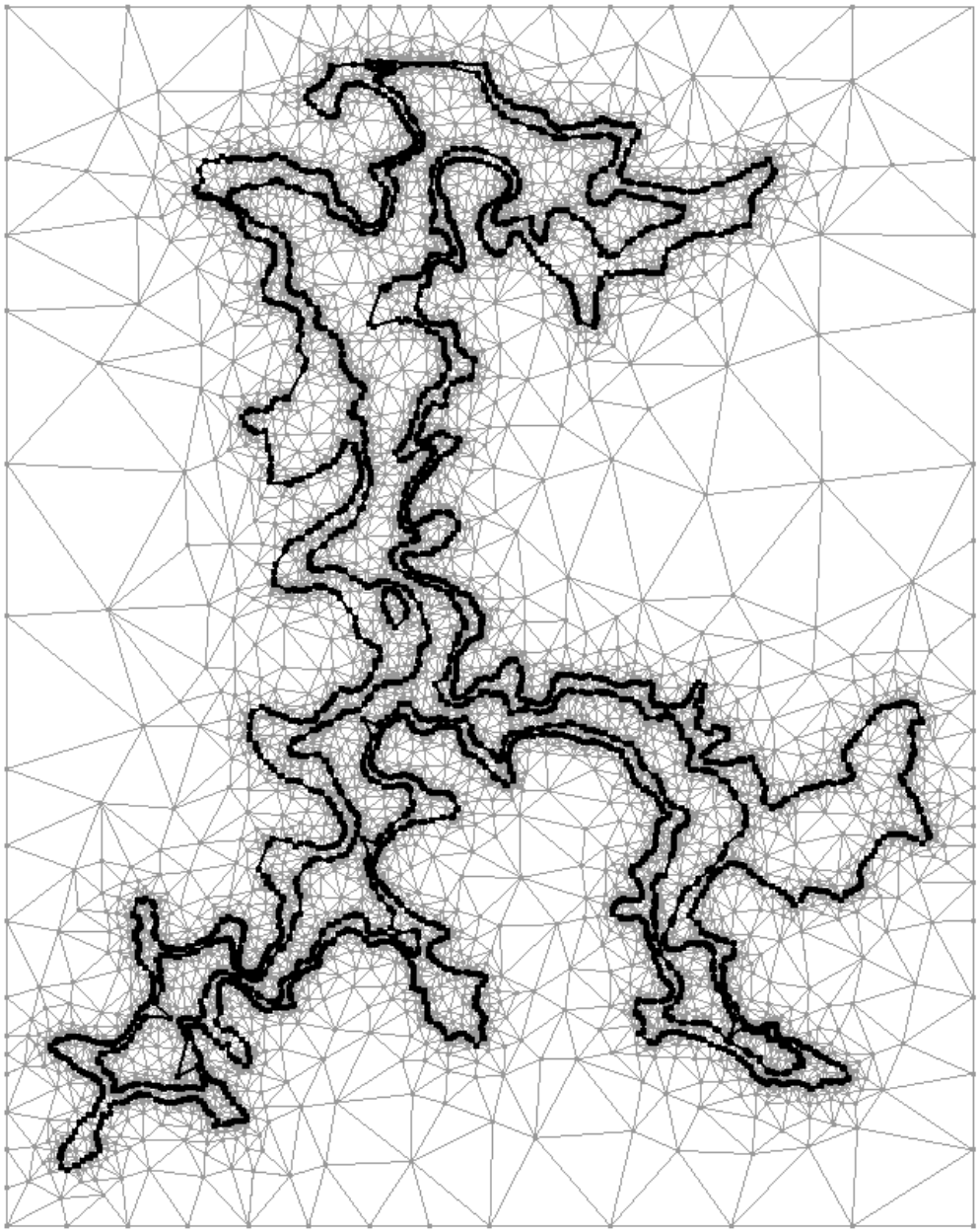


Figura 7.6: Refinamento realizado sobre duas curvas de nível (em negrito) de valores 1045 (interna) e 1070 (externa) metros; nº de triângulos = 25471; nº de vértices = 12767; $B = \sqrt{2}$;

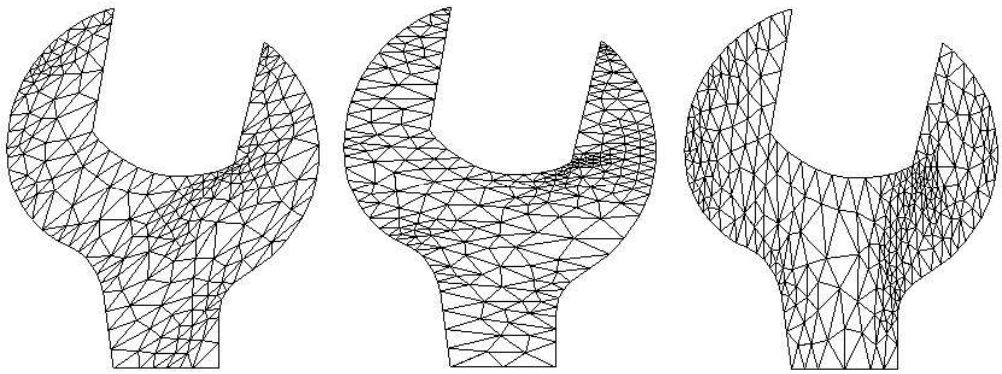
7.1.2 Resultados gerados pelo algoritmo de Refinamento Anisotrópico

As figuras abaixo foram geradas pelo algoritmo 5 com diferentes valores de B, e com um tensor métrico diferente para cada figura. Os tensores métricos são uniformes em todo domínio. Os tensores métricos utilizados para gerar as figuras abaixo são:

$$M_1 = \begin{bmatrix} 1.25 & -0.75 \\ -0.75 & 1.25 \end{bmatrix} \tag{7.1}$$

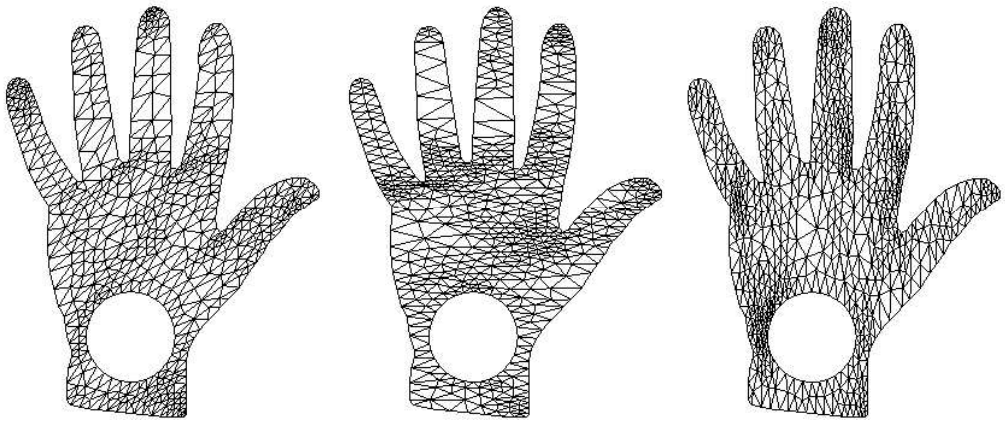
$$M_2 = \begin{bmatrix} 0.5 & 0 \\ 0 & 2.5 \end{bmatrix} \tag{7.2}$$

$$M_3 = \begin{bmatrix} 2.5 & 0 \\ 0 & 0.5 \end{bmatrix} \tag{7.3}$$



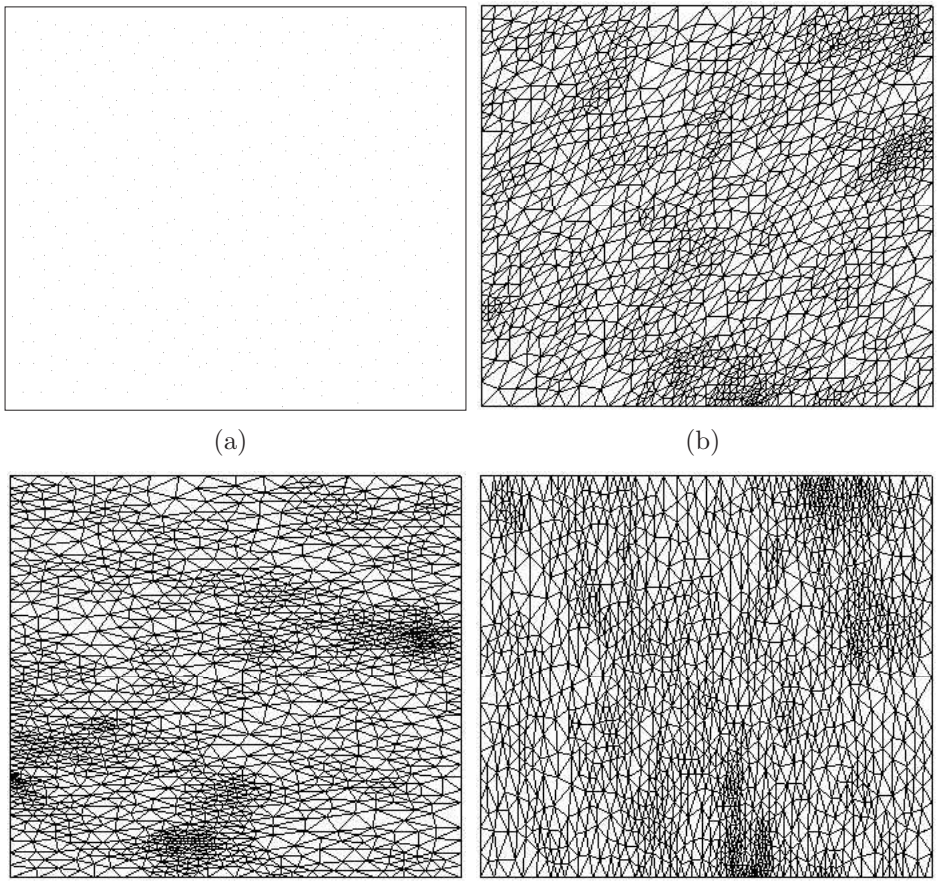
(a) (b) (c)

Figura 7.7: (a) $B = 0.98$ e métrica $= M_1$; (b) $B = 0.93$ e métrica $= M_2$; (c) $B = 0.94$ e métrica $= M_3$



(a) (b) (c)

Figura 7.8: (a) $B = 0.93$ e métrica = M_1; (b) $B = 0.93$ e métrica = M_2; (c) $B = 0.93$ e métrica = M_3



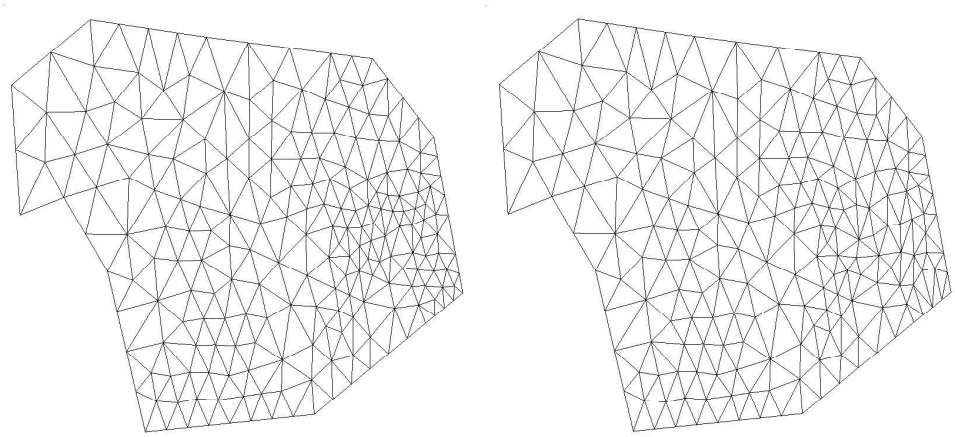
(a) (b)

(c) (d)

Figura 7.9: (a) PSLG; (b) $B = 0.93$ e métrica = M_1; (c) $B = 0.93$ e métrica = M_2; (d) $B = 0.93$ e métrica = M_3

7.1.3 Simplificação de Malhas Delaunay

As figura abaixo foi gerada pelo algoritmo 4 descrito na seção 4. O parâmetro utilizado para simplificar a malha original (figura 7.10a) foi área do triângulo de forma que a figura simplificada (figura 7.10b) tem apenas triângulos com área superior a 0.00046.



(a) Malha original; (b) Malha simplificada;

Figura 7.10: A malha original contém 379 triângulos e 215 vértices; a malha simplificada contém 349 triângulos, 200 vértices e todos os triângulos tem área maior que 0.00046.

Capítulo

8

Conclusão

Os resultados obtidos pelo algoritmo de Refinamento Anisotrópico podem ser utilizados em aplicações que necessitem que os elementos da malha estejam direcionados em um único sentido, como por exemplo, o escoamentos de fluído em tubos. Para aplicações em que é necessário que os elementos da malha estejam direcionados em vários sentidos, é necessário que seja aplicado um tensor métrico M diferente para cada vértice da malha, no entanto, esta abordagem exige um cuidado especial, pois se a diferença das métricas para vértices muito próximos for muito grande podem ocorrer anomalias na malha resultante, como por exemplo, intersecções entre segmentos. Para contornar estes problemas é necessário tomar medidas complexas como pode-se observar em [16].

A simplificação de malhas isotrópicas discutida neste trabalho gerou resultados satisfatórios, porém as aplicações para este algoritmo não são muitas, uma vez que é desejável malhas com triângulos bons, ou seja, triângulos próximos ao equilátero. No entanto se alguma aplicação tiver restrições com relação à área dos elementos da malha, ou necessidade de eliminar vértices, este algoritmo pode ser utilizado.

O algoritmo de Refinamento Isotrópico deste trabalho é um algoritmo de Refinamento Delaunay para malhas bidimensionais. Embora o tema Anisotropia abordado neste trabalho seja relativamente complexo, a obtenção do algoritmo de refinamento foi o que tomou maior tempo. Este algoritmo depende do algoritmo de Triangulação de Delaunay para funcionar, e a Triangulação de Delaunay para a estrutura de dados OF ainda estava sendo testada e melhorada no início deste trabalho. A própria estrutura de dados utilizada ainda estava em fase de construção. Alguns elementos desta estrutura ainda não estavam prontos e precisavam ser testados e melhorados, o que também tomou uma parte do tempo deste trabalho. O principal problema enfrentado na elaboração do algoritmo de refinamento foi o tratamento de ângulos pequenos. Para solucionar este problema foi

utilizada a estratégia proposta em [11], pois é comprovada ser melhor do que a estratégia utilizada em [14]. A não ser pelo tratamento de ângulos pequenos, o restante do algoritmo de refinamento foi elaborado utilizando as idéias propostas em [14] que é baseado no algoritmo de Refinamento de Jim Ruppert [13]. Este algoritmo gera malhas de boa qualidade para várias espécies de entradas desde que estejam de acordo com a definição de PSLG (definição 23).

Como trabalhos futuros pretende-se utilizar o algoritmo de Refinamento Anisotrópico para geração de malhas em simulações númericas de escoamento de fluídos, com a anisotropia dependente do domínio do problema e variável sobre todo o domínio. Também pretende-se realizar um Refinamento Anisotrópico respeitando as curvas de nível de um terreno reservado para a construção de represas.

Referências Bibliográficas

[1] Franz Aurenhammer and Herbert Edelsbrunner. An optimal algorithm for constructing the weighted voronoi diagram. *Pattern Recognition*, 17:251–257, 1984.

[2] Jean-Daniel Boissonnat, Olivier Devillers, Sylvain Pion, Monique Teillaud, and Mariette Yvinec. Triangulations in cgal. *Computational Geometry - Theory and Applications*, 22:5–19, 2002.

[3] Frank J. Bossen. Anisotropic mesh generation with particles. Technical Report CMU-CS-96-134, CS Dept., Carnegie Mellon University, May 1996. http://ltswww.epfl.ch/∼bossen/.

[4] Frank J. Bossen and Paul S. Heckbert. A pliant method for anisotropic mesh generation. In *5th Intl. Meshing Roundtable*, pages 63–74, Oct. 1996. http://www.cs.cmu.edu/∼ph.

[5] M.J. Castro-Díaz, F. Hecht, and B. Mohammadi. New progress in anisotropic grid adaptation for inviscid and viscous flows simulations. Technical Report RR 2671, INRIA, Rocquencourt, France, October 1995. http://www.inria.fr/RRRT/RR-2671.html.

[6] L. Paul Chew. Guaranteed-quality triangular meshes. Technical Report TR-89-983, Department of Computer Science, Cornell University, 1989.

[7] E. F. D'Azevedo. Are bilinear quadrilaterals better than linear triangles? Technical Report ORNL/TM-12388, Computer Science and Mathematics Division - Oak Ridge National Laboratories - Oak Ridge - Tennesse, August 1993. http://citeseer.ist.psu.edu/491362.html.

[8] Olivier Devillers. On deletion in delaunay triangulations. In *SCG '99: Proceedings of the fifteenth annual symposium on Computational geometry*, pages 181–188. ACM Press, 1999.

[9] Herbert Edelsbrunner. *Geometry and Topology for Mesh Generation*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge, 2001.

[10] François Labelle and Jonathan Richard Shewchuk. Anisotropic voronoi diagrams and guaranteed-quality anisotropic mesh generation. In *9th Annual Symposium on Computational Geometry*, pages 191–200, June 2003.

[11] Steven Elliot Pav. *Delaunay Refinement Algorithms*. PhD thesis, Department of Mathematical Science - Carnegie Mellon University, 2003.

[12] J. Rossignac, A. Safanova, and A. Szymczak. 3d compression made simple: Edgebreaker on a corner table. In *Shape Modeling International Conference*, pages 278–283, Genoa, Italy, May 2001.

[13] Jim Ruppert. A delaunay refinement algorithm for quality 2-dimensional mesh generation. *J. Algorithms*, 18(3):548–585, 1995.

[14] Jonathan Richard Shewchuk. *Delaunay Refinement Mesh Generation*. PhD thesis, Scholl of Computer Science - Carnegie Mellon University, 1997.

[15] Jonathan Richard Shewchuk. What is a good linear finite element? - interpolation, conditioning, anisotropy, and quality measures. citeseer.ist.psu.edu/shewchuk02what.html, 2003.

[16] Jonathan Richard Shewchuk. Anisotropic voronoi diagrams and guaranted-quality anisotropic mesh generation. slids presentation, 2004.

[17] S. H. Teng and C. W. Wong. Unstructured mesh generation: Theory, practice, and perspectives. *International Journal of Computer Geometry*, 10(3):227–266, 2000.

[18] X. Xu, C. C. Pain, A. J. H. Goddard, and C. R. E. de Oliveira. An automatic adaptive meshing technique for delaunay triangulations. *Computer Methods in Apllied Mechanics and Engineering*, 161:297–303, 1998.