UNIVERSIDADE FEDERAL DE SANTA CATARINA
CURSO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

GERAÇÃO DE MALHAS PARA DOMÍNIOS 2,5 DIMENSIONAIS USANDO TRIANGULAÇÃO DE DELAUNAY RESTRITA

DISSERTAÇÃO SUBMETIDA À UNIVERSIDADE FEDERAL DE SANTA CATARINA
PARA A OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA

CLOVIS RAIMUNDO MALISKA JUNIOR

FLORIANÓPOLIS, FEVEREIRO DE 2001

GERAÇÃO DE MALHAS PARA DOMÍNIOS 2,5 DIMENSIONAIS USANDO TRIANGULAÇÃO DE DELAUNAY RESTRITA

CLOVIS RAIMUNDO MALISKA JUNIOR

ESTA DISSERTAÇÃO FOI JULGADA ADEQUADA PARA A OBTENÇÃO DO TÍTULO DE MESTRE EM ENGENHARIA ESPECIALIDADE ENGENHARIA MECANICA, ÁREA DE CONCENTRAÇÃO ENGENHARIA DE CIÊNCIAS TÉRMICAS, APROVADA EM SUA FORMA FINAL PELO CURSO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

_____

Prof. CLOVIS RAIMUNDO MALISKA, Ph. D.
ORIENTADOR

_____

Prof. JÚLIO CÉSAR PASSOS, Dr. Eng. Mec.
COORDENADOR DO CURSO DE PÓS-GRADUAÇÃO

BANCA EXAMINADORA

_____

Prof. ALTAMIR DIAS, Dr. Eng. Mec.

_____

Prof. ANTONIO CARLOS RIBEIRO, Dr. Eng. Mec.

_____

Prof. ANTONIO FÁBIO CARVALHO DA SILVA, Dr. Eng. Mec.

Dedico este trabalho a meus pais que ao longo de suas vidas nunca pouparam esforços para me mostrar a importância do conhecimento

AGRADECIMENTOS

Ao Prof. Clovis Raimundo Maliska, orientador, pai e exemplo pessoal de pesquisador, pela orientação, motivação, suporte, atenção, dedicação e paciência proporcionados em todos os momentos do trabalho.

A Ana Maria Maliska, mãe, que com muita sabedoria, amor e carinho proporcionou o apoio e a tranqüilidade necessárias para superar as dificuldades existentes neste desafio.

A Karina Maliska pela constante presença e alegria.

Ao amigo Marcus Vinicius F. dos Reis, pelo companheirismo a mim proporcionado nesta jornada, e em especial aos amigos Marcos Cabral Damiani e Rodrigo Machado Lucianetti, pelas longas e valiosas discussões sobre o problema de discretização geométrica e de simulação.

A Juliana dos Santos Faria Lichtemberg pela ajuda na correção ortográfica do material.

A todos os meus familiares e amigos que me apoiaram e de alguma forma se envolveram e contribuíram com este trabalho.

Aos colegas do SINMEC pelo excelente ambiente de trabalho.

Aos professores do curso de pós-graduação do Departamento de Engenharia Mecânica da UFSC.

Ao CNPq pelo financiamento do trabalho.

SUMÁRIO

RESUMO

Gerar uma malha consiste em discretizar um domínio geométrico em pequenos elementos de forma geométrica simplificada, como triângulos e/ou quadriláteros, em duas dimensões, e tetraedros e/ou hexaedros em três dimensões. Malhas são utilizadas em diversas áreas, como geologia, geografia e cartografia, onde elas fornecem uma representação compacta dos dados do terreno; em computação gráfica, a grande maioria dos objetos são mapeados por malhas antes de serem renderizados ou aplicados outros métodos de imagens; e, em matemática aplicada e computação científica, são essenciais na solução numérica de equações diferenciais parciais, resultantes do modelamento de problemas físicos.

Este trabalho concentra-se no desenvolvimento de um gerador de malhas voltadas para esta última aplicação, mas que podem, também, ser empregadas nas outras áreas. Mais especificamente, o interesse está na geração de malhas de triângulos não-estruturadas, através do processo de triangulação de Delaunay, para aplicações na solução de problemas de transferência de calor em superfícies planas tridimensionais. Devido à utilização do método CVFEM (*Control Volume based Finite Element Method*) para a modelagem numérica, um paralelo entre a Triangulação de Delaunay e Diagramas de Voronoi é delineado, apresentando suas propriedades e aplicações.

São estudados os métodos de geração de triangulações de Delaunay para superfícies planas de *inversão de aresta*, *divide-and-conquer* e *incremental*. A estrutura de dados utilizada é a *triangular*, e o método de refino para garantia de qualidade de malha é baseado no algoritmo de Ruppert. Restrições geométricas são tratadas de forma que a malha gerada obedeça as intersecções e conexões entre diversas superfícies.

A contribuição fundamental do presente trabalho está na extensão de métodos de triangulação de Delaunay e de refino de malha bidimensionais para domínios 2,5 dimensionais compostos, isto é múltiplos planos interconectados no espaço tridimensional tratados simultaneamente. Otimização de ângulos internos, tamanho e forma dos elementos através da especificação de parâmetros, conferem ao gerador desenvolvido versatilidade e generalidade.

LISTA DE FIGURAS

1. Introdução

1.1. Preliminares

A discretização de geometrias complexas em entidades de forma geométrica simplificada é um requisito presente em diversas aplicações como geometria computacional, interpolação, computação gráfica, modelagem sólida, descrição topográfica, métodos numéricos, entre outras. A discretização realizada para métodos numéricos para a solução de equações diferenciais parciais (EDP's), também denominada de geração da malha, é a que apresenta maiores dificuldades em função das exigências destes métodos em relação a qualidade da malha. Se conseguirmos obter malhas que satisfaçam este tipo de aplicação, todas as outras aplicações estarão, automaticamente, contempladas.

Dentre os diversos métodos numéricos que usam discretização da superfície ou do volume, os mais estudados e utilizados são os métodos de Volumes Finitos e Elementos Finitos. Estes métodos são utilizados para simular uma classe bastante ampla de fenômenos em transferência de calor, escoamento de fluidos, deformações mecânicas, propagação de ondas eletromagnéticas e mecânica quântica. Para se obter estas soluções, costuma-se efetuar a discretização do domínio de estudo utilizando-se triângulos e/ou quadriláteros, para geometrias 2D, e tetraedros e/ou hexaedros, para geometrias 3D, conforme mostra a Fig. 1. Estes métodos aproximam as EDP's através de funções polinomiais segmentadas, cada parte associada a um elemento da malha. Por elemento da malha, entende-se os elementos geométricos de forma simplificada que a compõe, e que, no presente estudo, são triângulos. Desta forma, um sistema de equações algébricas, lineares ou não-lineares deve ser resolvido para se obter a solução aproximada das EDP's.

Como o tempo de execução para obter as soluções utilizando estes métodos é proporcional ao tamanho da malha, procura-se gerar malhas representativas com o menor número possível de elementos. Como veremos ao longo do texto, o conceito de representatividade da malha para aplicações em simulação, não é apenas quanto à fidelidade de representação da geometria, mas também quanto ao problema físico que está sendo simulado. Além disto, é sabido que a estabilidade numérica e a convergência destes métodos

é diretamente afetada pela forma dos elementos: elementos muito longos e estreitos poderão gerar soluções com grandes erros e de difícil obtenção, conforme mostrado por Freitag e Ollivier-Gooch [45].

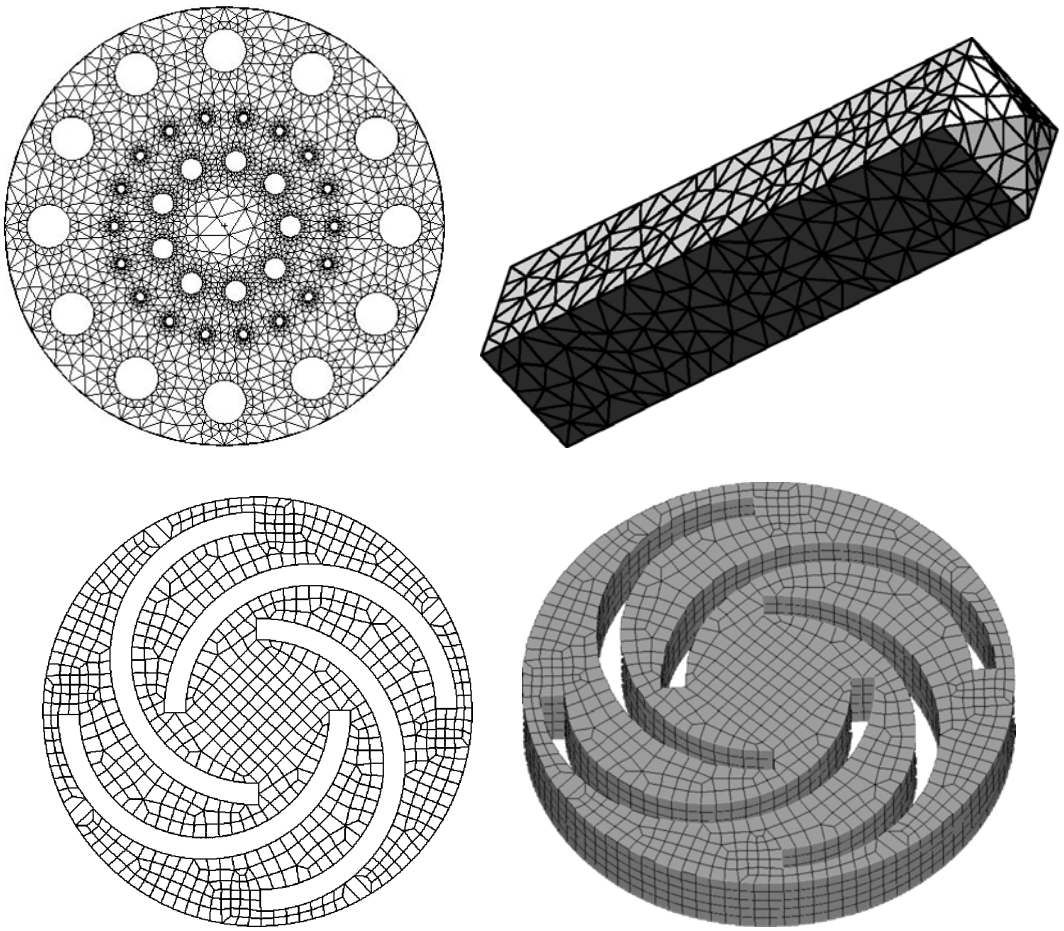


Fig. 1 - Exemplos de malhas bi [1][55] e tridimensionais [99][55]

Em resumo, o problema de geração de malha é bem mais complexo do que o simples particionamento do domínio geométrico com o menor número de elementos possíveis ou do controle da forma destes elementos. A dificuldade maior está em se considerar, simultaneamente, aspectos [99], como:

- número de elementos;
- tamanho e forma dos elementos, e
- conformação das fronteiras da malha à geometria desejada.

É importante também observar que os critérios para o tamanho e a forma dos elementos podem ser muito complexos, envolvendo aspectos como:

- tipo do problema sendo resolvido;
- precisão desejada da solução;
- comportamento da solução, e
- condições de contorno aplicadas.

Um exemplo de como estes critérios podem ser diversos está na solução de problemas de escoamento de fluidos com direção preferencial e em problemas em meios não-isotrópicos. Conforme as condições de contorno aplicadas ou condições do meio, em uma mesma geometria física, a malha que melhor se adapta à solução dos problemas pode ser totalmente diferente. Simuladores com suporte para malhas adaptativas buscam exatamente este tipo de comportamento de forma automática, com a menor interferência possível do analista numérico. Novamente, recai sobre o gerador de malhas também esta tarefa.

Estes comentários mostram que o desenvolvimento de geradores de malhas que apresentem características que atendam aos requisitos das áreas de simulação de fenômenos físicos é um desafio semelhante, ou maior, que o próprio desenvolvimento das metodologias numéricas.

Nos itens seguintes serão apresentados, com objetivo introdutório, aspectos gerais da geração de malha e, mais especificamente, de malhas não-estruturadas compostas de triângulos. A seguir, a revisão bibliográfica específica do assunto tratado nesta dissertação é apresentada, ou seja, aquela relacionada diretamente à triangulação de Delaunay em superfícies 2D (bidimensionais definidas no plano (x,y)) e domínios 2.5D (bidimensionais planas definidas no espaço (x,y,z)). A revisão detalhada da literatura é feita integrada no Cap. 2 e 3 onde os tipos de triangulação e seus métodos de obtê-las são descritos.

1.2.Revisão bibliográfica

Os primeiros trabalhos na área de geometria computacional datam do começo do século. Voronoi [110], em 1907, propôs uma estrutura geométrica, hoje denominada de diagramas de Voronoi, que foi o primeiro passo no sentido de definir-se o tipo de triangulação utilizada neste trabalho. Delaunay [28], em 1934, formalizou os conceitos do que hoje é denominado de triangulação de Delaunay. A presente revisão bibliográfica tem o

objetivo apenas de apresentar os principais trabalhos realizados nesta área, sendo que uma revisão mais detalhada dos métodos de geração será feita no capítulo 3.

Os primeiros trabalhos computacionais utilizando a triangulação de Delaunay datam da década de 70, onde o pioneiro foi Duppe [34], em 1970, com estudos na área de interpolação. Em 1977, LAWSON [63] fez importantes contribuições, tanto nos estudos teóricos de métodos incrementais para a obtenção da triangulação de Delaunay (que veremos ao longo do trabalho, são de vital importância para simuladores adaptativos) quanto na implementação de métodos de interpolação.

Os primeiros trabalhos de aplicação da triangulação de Delaunay para simulação numérica foi de Babuska e Aziz [5], em 1976, onde apresentou-se um estudo empírico do efeito do ângulo interno dos elementos no método de elementos finitos. Os primeiros trabalhos na área de volumes finitos, utilizando malhas não-estruturadas foram de Baliga e Patankar [8], em 1980. Estes, no entanto, não aplicam a triangulação de Delaunay, mas uma triangulação qualquer. Sabe-se hoje, que a conjugação da triangulação de Delaunay com o método de CVFEM, proposto por estes autores, simplifica consideravelmente a realização dos balanços das propriedades nos volumes de controle.

Talvez uma das maiores contribuições para a obtenção de triangulação de Delaunay tenha sido dada por Guibas e Stolfi [53], em 1985, com a formalização dos métodos disponíveis e das estruturas de dados utilizadas em triangulações. Poucos anos depois, em 1987, Fortune [42], com um estudo teórico bastante denso e de grande contribuição na área de diagramas de Voronoi, apresentou um novo método de geração de malha denominado *sweepline*.

Os primeiros trabalhos na área de escoamento externo de fluidos utilizando malhas não estruturadas apareceram no início da década de 90. Um dos pioneiros foi Mavriplis [80], em 1991, utilizando um método de avanço de frente para localização dos pontos e definição dos triângulos. No entanto, aparentemente, nenhum trabalho de escoamento de fluidos, utilizando volumes finitos, tinha considerado até o presente momento a triangulação de Delaunay como base para a discretização geométrica.

Os primeiros estudos na área de otimização de triangulações foram feitos por Bern e Eppstein [13], em 1992. Neste trabalho, Bern e Eppstein apresentaram o método de triangulação ótima *quadtree*, analisando seu comportamento e parâmetros de otimização.

Um dos trabalhos precursores na área de simulação numérica utilizando diagramas de Voronoi foi Palagi [87], com sua tese de doutorado em 1992. Neste trabalho Palagi estuda a influência da ortogonalidade local das malhas obtidas com os diagramas de Voronoi, denominadas de PEBIGrids, na simulação numérica de meios porosos para aplicações de petróleo.

Em 1993 Maliska Jr. [75] propõe um novo método de geração de diagramas de Voronoi, baseado em premissas de robustez. Utilizando este gerador, em 1994, Maliska e Maliska Jr [74], desenvolvem um estudo na simulação de escoamento de traçadores em meios porosos bidimensionais de geometria arbitrária. Neste mesmo ano, Marcondes [78] também utiliza as malhas de Voronoi criadas pelo gerador de Maliska Jr para efetuar estudos de escoamento de fluidos em reservatórios de petróleo.

Em 1995, Ruppert [92] propõe o primeiro método de refino para triangulação de Delaunay com demonstração teórica da garantia de obtenção dos critérios impostos. O trabalho, dada a inovação da sua abordagem, definiu um novo grupo de métodos de refino denominados refinos de Delaunay.

Este histórico mostra que, a maioria esmagadora das aplicações da triangulação de Delaunay até hoje, foram na área de interpolação, e não de simulação. Como as aplicações de interpolação utilizam sempre domínios bi ou tridimensionais, poucos desenvolvimentos na área de triangulação 2,5D, considerando triangulações de Delaunay, foram feitos. O mais importante exemplo de aplicação da triangulação de Delaunay para domínios 2,5D é atribuído à Marcum e Weatherill [79], em 1994. Por incrível que pareça, neste trabalho, primeiramente se cria uma malha de triangulação de Delaunay 3D, para então, utilizar-se os elementos obtidos na fronteiras do domínio (as superfícies 3D) como parâmetro inicial de um processo de geração de malha 3D, baseado na técnica de avanço de frentes. Como não existe ainda uma fundamentação teórica suficiente para desenvolver-se um triangulador de Delaunay baseado em parâmetros ótimos, tal método não garante nenhuma otimização na malha gerada nas superfícies 3D.

O trabalho desenvolvido nesta dissertação é, dentro do conhecimento do autor, um dos primeiros trabalho de triangulação de Delaunay em domínios 2,5 dimensionais, baseado na otimização de parâmetros da malha. Esta otimização é conseguida pela extensão do método de refino de Ruppert para este tipo de domínio geométrico.

1.3. Objetivos e contribuições

O presente trabalho tem como objetivo principal o desenvolvimento de um gerador de malhas para domínios denominados 2,5D, considerando restrições geométricas e qualidade de malha. Como objetivos secundários, tem-se a revisão dos principais problemas e aspectos envolvidos com a geração de malhas não-estruturadas, além de uma revisão sobre os diversos métodos de geração disponíveis na literatura.

A principal contribuição encontra-se na extensão para domínios de dimensão 2,5D de um método de triangulação bidimensional baseado na técnica de solução de problemas *divide-and-conquer* e do método de refino de Ruppert [92]. Este método será aqui denominado de método de triangulação *divide-and-conquer*, e sua extensão para domínios 2,5D é feita através de uma abordagem inovadora baseada em camadas topológicas. O método de camadas topológicas é um método que permite a extensão, para domínios 2,5D, de qualquer algoritmo de triangulação bidimensional que utilize apenas informações topológicas para a definição da conectividade dos elementos. O gerador de malhas desenvolvido neste trabalho pode ser empregado em:

- Métodos de elementos finitos que usam a triangulação como base de discretização;
- Métodos de volumes finitos que usam, tanto o elemento triangular, como o dual da triangulação de Delaunay como volume de realização dos balanços.

Também temos observado que é comum a limitação dos desenvolvimentos numéricos em malhas não estruturadas em instituições acadêmicas pela falta de trianguladores de fácil uso. Isto é particularmente verdade na comunidade dos usuários do método de volumes finitos, cuja tradição é o uso de malhas estruturadas. Espera-se que os desenvolvimentos deste trabalho possam contribuir neste aspecto, disponibilizando-se um gerador versátil e de fácil uso.

O gerador de malhas desenvolvido neste trabalho também está presente no programa SATER100 [40], que é um programa geral de análise térmica de sistemas que resolve problemas condutivo-radiativos em estruturas delgadas definidas no espaço. Neste programa o gerador discretiza as superfícies 2,5D que são empregadas com o método de volumes finitos para a solução das equações da condução e radiação de calor.

1.4. Escopo do trabalho

A grande maioria dos trabalhos sobre geradores de malhas, especialmente triangulações, encontra-se na área de geometria computacional que, naturalmente, possui todo um rigorismo matemático de definição dos entes geométricos. O presente autor tem dedicado grande parte de suas atividades acadêmicas no desenvolvimento de geradores de malhas, visualizadores, computação gráfica, etc., mas não se classifica como um integrante da área de geometria computacional. Ao contrário, seu aperfeiçoamento formal é na área de simulação numérica. Portanto, dosar, neste trabalho, os ingredientes das duas áreas não foi uma tarefa fácil. Por isso, em todo o trabalho o autor preocupou-se em conectar as questões de geração de malha com a área de aplicação de interesse, a simulação numérica. Mesmo assim, os capítulos fundamentais do trabalho consideram a triangulação como uma aplicação geral e não específica para a área numérica. Para os especialistas na área numérica interessados em geração de malha, uma extensa lista de artigos, teses e apostilas podem ser encontradas nas referências bibliográficas.

1.5. Organização do trabalho

Este trabalho está dividido em seis capítulos. O capítulo introdutório, Cap. 1, apresenta a motivação do trabalho, a revisão bibliográfica e uma rápida apresentação dos tipos de malhas empregadas em simulação.

O Cap. 2 apresenta as características de uma triangulação qualquer, os tipos de domínios geométricos aceitos pelo triangulador desenvolvido e as propriedades desejadas para uma malha. Em seguida dedica-se, mais profundamente para a triangulação de Delaunay, característica da triangulação do método implementado.

O Cap. 3 apresenta os métodos disponíveis na literatura que obtém a triangulação de Delaunay. Este capítulo contém toda a discussão teórica empregada no método desenvolvido e, apesar de ser de leitura mais difícil, é imprescindível para, além de identificar o estado da arte no assunto, descrever os métodos existentes e suas conexões com os desenvolvimentos. No final deste capitulo são apresentados também outros métodos de triangulação ótima e uma

breve descrição dos métodos de pós-processamento para melhoramento da qualidade das malhas.

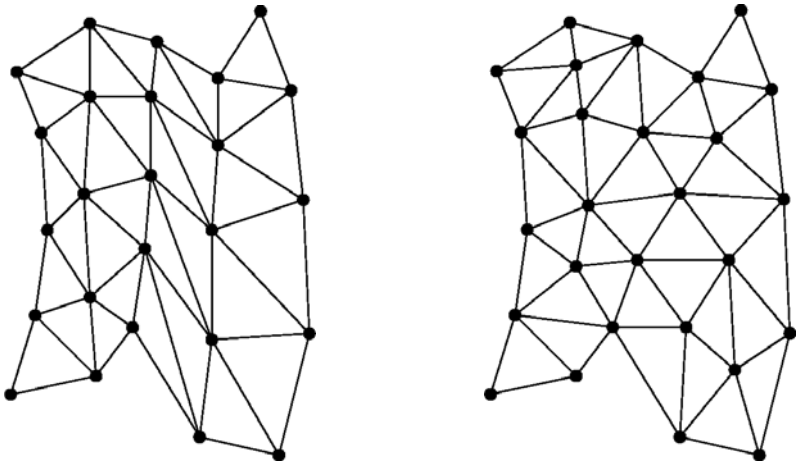
O Cap. 4 descreve em detalhes o algoritmo desenvolvido nesta dissertação, iniciando com a maneira de fornecimento ao triangulador do domínio a ser discretizado, até os detalhes do método usado para o refino da malha. Procurou-se, para cada etapa do algoritmo, fazer a sua descrição geométrica para facilitar àqueles que pretendam reproduzir o algoritmo.

O Cap. 5 apresenta os resultados obtidos com o gerador de triangulações de Delaunay para superfícies tridimensionais com restrições geométricas, utilizando o método de refino de Ruppert. Diversos resultados, desde problemas mais simples concebidos para mostrar as características do método até problemas em superfícies 3D mais complexas, são apresentados para mostrar a generalidade e versatilidade do triangulador.

O Cap. 6 encerra o trabalho com as conclusões e sugestões para estudos e trabalhos futuros em geração de malhas não-estruturadas de triângulos em domínios 2.5 dimensionais.

1.6.Malhas estruturadas e não-estruturadas

Existem dois tipos principais de malhas de grande uso em simulação numérica, as estruturadas e as não-estruturadas. Este trabalho dedica-se apenas ao segundo tipo, mas uma apresentação destes tipos é aqui feita com objetivo de introdução ao assunto.

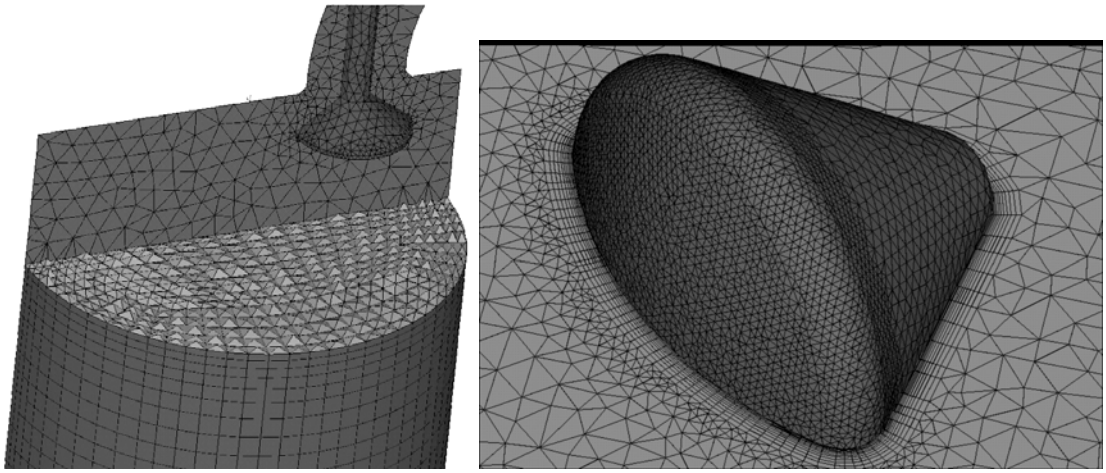


(a) (b)

Fig. 2 - Malha estruturada e não-estruturada de triângulos [99]

Malhas estruturadas são, basicamente, quadriláteros quaisquer em duas dimensões e hexaedros quaisquer em três dimensões. Estes quadriláteros e hexaedros são “encaixados” entre si (quatro vizinhos através de cada aresta no 2D e seis vizinhos através de cada face no 3D) com uma ordem de formação que permite encontrar-se diretamente, através de uma fórmula fechada, os vizinhos para qualquer elemento da malha que esteja sendo visitado. Pode-se, no entanto, construir malhas estruturadas utilizando-se, por exemplo, apenas triângulos. A Fig. 2(a) apresenta um exemplo de uma malha estruturada de triângulos.

Por outro lado, as malhas não-estruturadas são, normalmente, uma composição de quadriláteros e triângulos em duas dimensões e hexaedros e tetraedros em três dimensões. Em domínios 3D, utilizam-se também prismas e pirâmides para se compor encaixes e formações especiais para captação de fenômenos específicos, como camada limite, por exemplo. Nas Fig. 2(b) e Fig. 5(c) podem ser vistos exemplos de malhas não-estruturadas bidimensionais, e na Fig. 3 exemplos de malhas não-estruturadas volumétricas tridimensionais. A malha da Fig. 3(b) mostra a formação de prismas perto da parede para captação dos efeitos de camada limite. Nestas malhas não existe uma ordem de formação para os vizinhos, portanto não é possível descobrir-se os mesmos diretamente. Deve-se, então, armazenar em memória uma matriz com os índices dos elementos vizinhos (conectividades), consumindo-se, assim, maior recurso computacional.

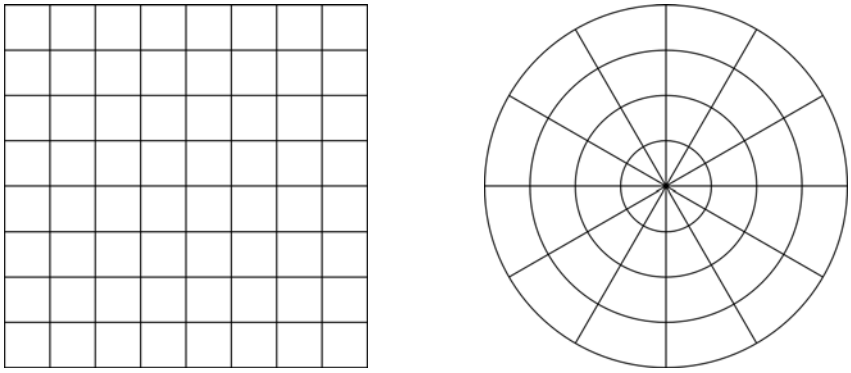


(a) (b)

Fig. 3 – Exemplo de malhas 3D com tetraedros, hexaedros, prismas e pirâmides [55]

Além das características topológicas (conexões entre os elementos), alguns tipos de malhas estruturadas podem fornecer simplificações para descobrir-se também as coordenadas

espaciais de cada vértice (através de fórmulas fechadas). É o caso das malhas cartesianas, polares, cilíndricas e esféricas, sejam elas igualmente ou não-igualmente espaçadas. Na Fig. 4 podem ser vistos exemplos destes tipos de malhas estruturadas. Com isto, é possível otimizar ainda mais a utilização dos recursos computacionais. Além disto, devido às características topológicas destas malhas, o controle sobre o tamanho e forma dos elementos é bastante simplificado, permitindo que o usuário efetue refinos com razoável facilidade. Outra vantagem deste tipo de malha é a facilidade de se efetuar o particionamento do domínio para paralelização do método [93].



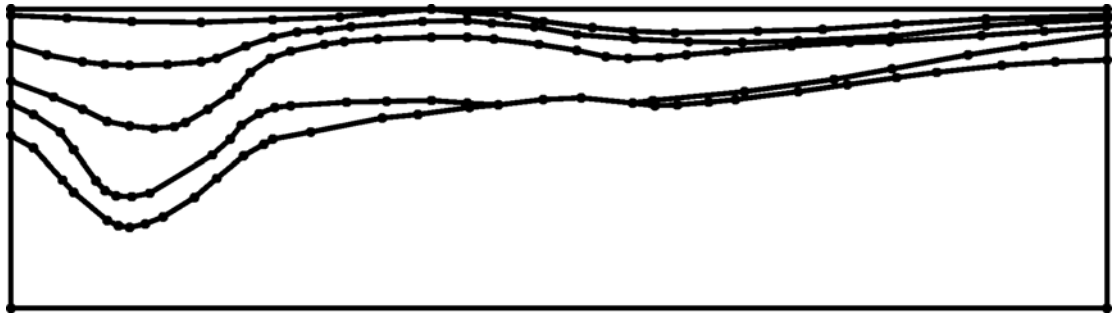
(a) (b)

Fig. 4 – Exemplo de malha cartesiana e polar

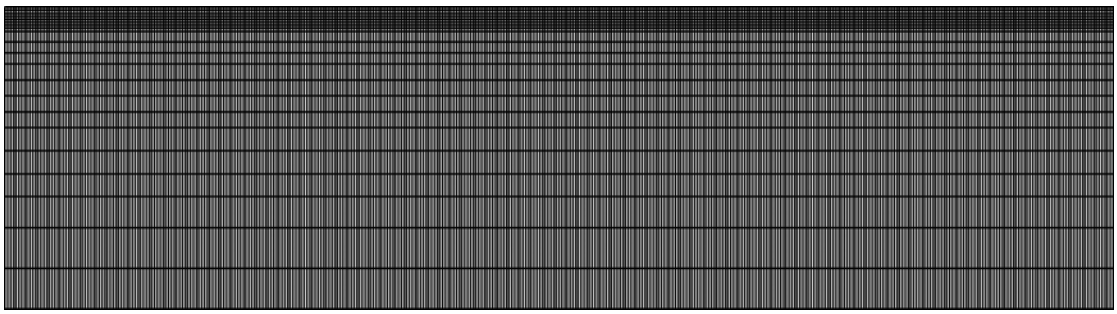
A grande desvantagem do uso de malhas estruturadas, no entanto, está na flexibilidade à conformação com geometrias complexas. Diversas técnicas foram desenvolvidas ao longo dos anos, de forma a se obter malhas estruturadas coincidentes com a geometria: mapeamento conforme [16], malhas generalizadas [100][108][72], blocagem de elementos [45] e outros. Um exemplo é a utilização de blocagem de malhas cartesianas, no qual a malha resultante possui “escadas” na fronteira do domínio, exatamente onde a solução pode ser crítica. A simplificação do domínio pode ser obtida pelo seu particionamento onde seja possível obter-se uma malha estruturada razoável para cada subdomínio [100]. Esta última técnica, chamada de multi-blocos, pode ser considerada uma não-estruturação em nível macro (ou uma estruturação por partes). Mesmo com estas e diversas outras técnicas, é possível encontrar geometrias complexas onde não é possível definir-se uma malha estruturada que se conforme à geometria. Por isso, malhas não-estruturadas são cada vez mais usadas.

Além da facilidade de conformação à geometria, a facilidade de conformação às propriedades do meio é uma característica fundamental presente nas malhas não-estruturadas.

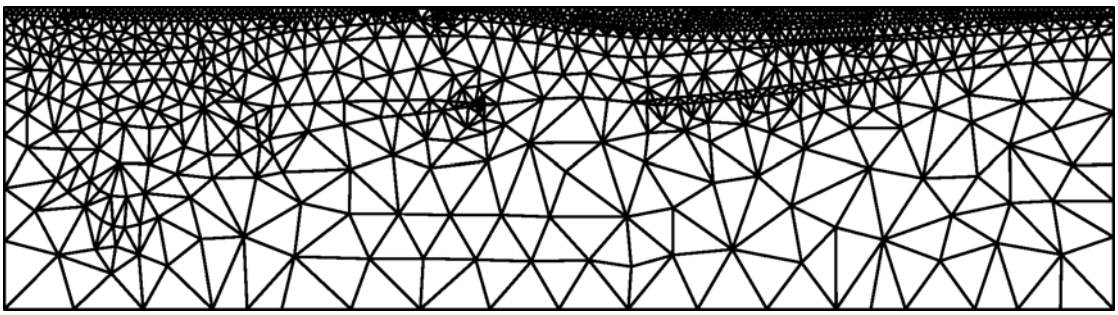
Em aplicações de estudo de reservatórios de petróleo como mapeamento geológico ou engenharia de reservatório, a forma dos elementos da malha constitui uma informação extremamente importante na representação do meio. A Fig. 5(a), por exemplo, mostra uma formação geológica com diferentes propriedades, onde as linhas mostradas identificam estas regiões. A malha estruturada da Fig. 5(b) não poderá representar adequadamente as camadas geológicas, e cálculos que sejam realizados nas regiões de descontinuidades serão imprecisos ou requererão esquemas especiais de interpolação. Para tentar respeitar com maior fidelidade as camadas, a malha requer um refinamento excessivo, como mostra a Fig. 5(b). A malha da Fig. 5(c), por conformar-se às camadas geológicas e permitir refinamento localizado, certamente, produzirá resultados melhores [72].



(a)



(b)



(c)

Fig. 5 – Domínio real e discretização estruturada e não-estruturada [99]

Neste problema, devido a não-flexibilidade de distribuição dos elementos da malha estruturada, a mesma possui cinco vezes mais elementos que sua correspondente não-estruturada. Mesmo considerando a necessidade do armazenamento em memória da matriz dos vizinhos, a malhas não-estruturada, além de melhor representar este problema físico, utiliza muito menos memória que a malha estruturada. Da mesma forma, o problema poderá ser resolvido em um menor tempo de computação, mostrando que, normalmente, a flexibilidade de discretização otimiza o número de elementos e o respectivo uso dos recursos computacionais. Esta otimização costuma ser ainda de maior impacto quando se consideram domínios tridimensionais.

Em função das considerações apresentadas acima, a simulação numérica tem investido grandes esforços no estudo de malhas não-estruturadas, com o objetivo de viabilizar e simplificar seu uso [4][45][106]. Apesar dos resultados até o momento indicarem as malhas não-estruturadas como o esquema de discretização que irá prevalecer no futuro, o uso de malhas estruturadas ainda irá permanecer durante muito tempo nas aplicações industriais, e provavelmente, nunca deixará de ser utilizado na pesquisa devido a sua simplicidade.

O presente trabalho irá mostrar um gerador versátil de malhas triangulares não-estruturada para superfícies 2.5D, que preenche as necessidades de uma grande gama de problemas da engenharia e da academia.

2. Triangulação - Características

Os tipos de triangulação são caracterizados pela observância de critérios de otimização geométricos da malha. Uma triangulação qualquer, por exemplo, relaxa qualquer tipo de exigência na geração e, como conseqüência, pode ser obtida com facilidade. Como conseqüência, a malha resultante pode não ser de boa qualidade. Por isso, quase todos os métodos consideram triangulação ótima, que significa uma triangulação onde pelo menos um critério de otimização é observado.

Este capítulo discute as características de uma triangulação, começando pelos tipos de domínios geométricos aceitos pelo triangulador desenvolvido, passando pela apresentação das propriedades desejadas para uma malha, concentrando-se na triangulação de Delaunay e, apresentando ainda, por completeza, outros tipos de triangulação ótima.

2.1. Tipos de domínios geométricos

Antes da classificação dos tipos de domínios geométricos, é importante fazer algumas definições quanto às primitivas geométricas ilustradas na Fig. 6.

1. *Ponto* - o menor elemento no domínio geométrico;
2. *Vértice* - ponto no domínio geométrico, a partir do qual a triangulação é construída e referenciada;
3. *Segmento* - uma linha que une dois pontos quaisquer no domínio geométrico;
4. *Aresta* - uma linha que une dois vértices quaisquer da triangulação;
5. *Elemento* - qualquer triângulo que une três vértices da triangulação.

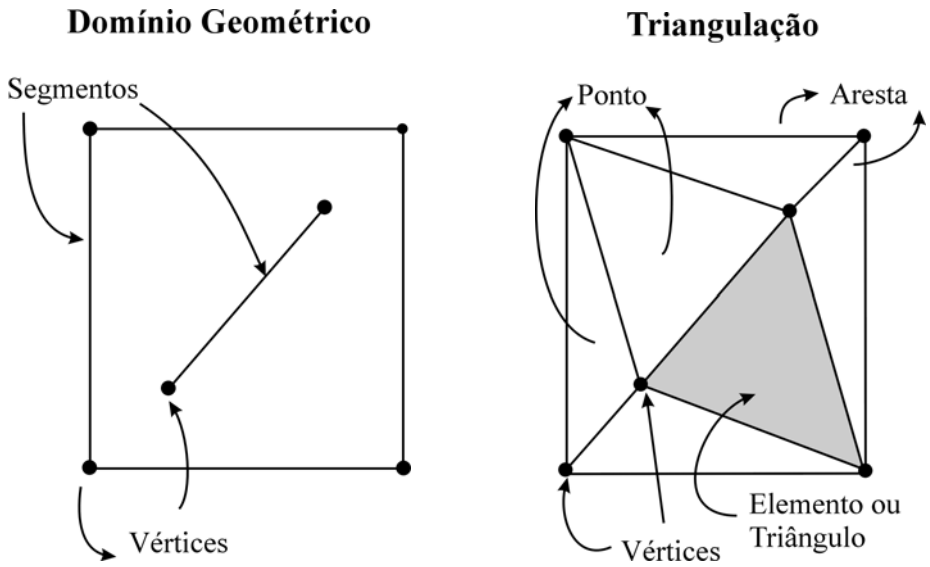


Fig. 6 - Primitivas geométricas utilizadas

Outra definição de importância é a de envelope convexo. O envelope convexo de um conjunto de pontos *p* no plano é o polígono convexo *c*, contido no plano de *p*, que engloba todos os pontos de *p*, e formado apenas por vértices em *p*. A mesma definição é diretamente estendida para n-dimensões. A Fig. 7 apresenta exemplos de envelopes convexos em duas e três dimensões.

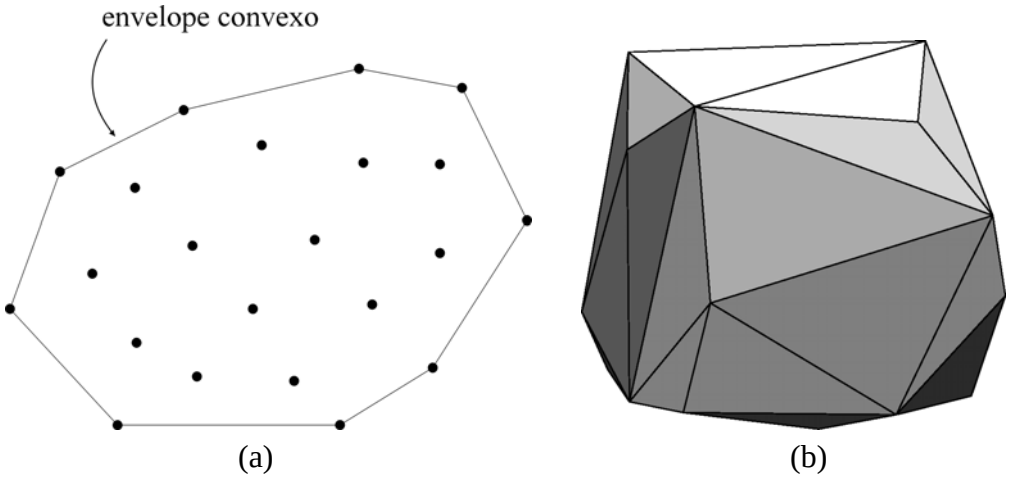


Fig. 7 - Envelope convexo de um conjunto de pontos em duas (a) e três dimensões (b)

Feitas estas definições, podemos classificar os tipos de domínios geométricos. Os domínios bidimensionais são aqui classificados em quatro grupos [11], os quais podem ser vistos, de forma genérica, como uma região poligonal do domínio, mais um conjunto de

restrições pontuais. Isto porque a fronteira do domínio, quando existente, é formada apenas por segmentos de reta (veremos mais sobre esta aproximação nesta mesma seção). O objetivo é obter-se um conjunto de triângulos que mapeiem por completo tal domínio, e que possuam arestas e vértices comuns a este. Tais triângulos, por sua vez, podem ou não obedecer a critérios de otimização definidos pelo usuário. Para todos os quatro tipos de domínios descritos a seguir, o número de vértices é um índice de complexidade do sistema. A Fig. 8 ilustra os quatro tipos de domínios geométricos e suas triangulações com e sem pontos adicionais. Como o nome já explica, pontos adicionais, ou pontos de Steiner, são pontos inseridos, além do conjunto fornecido pelo usuário, para melhorar, ou mesmo permitir, a triangulação.

a) Domínio geométrico definido por um conjunto de pontos

Neste tipo de domínio, o dado fornecido ao triangulador é um conjunto de pontos *p* no plano. Sem a utilização de vértices adicionais na triangulação *t*, os vértices de *t* são exatamente os pontos de *p*, e as fronteiras de *t*, o envelope convexo de *p*. Utilizando-se vértices adicionais em *t*, os vértices de *t* são um super-conjunto de *p*, e as fronteiras de *t* uma região convexa que pode exceder o envelope convexo de *p*.

b) Domínio geométrico definido por uma curva poligonal simples

Nesta classificação, a fronteira do domínio é uma curva poligonal simples fechada *c*, e cada segmento de reta de *c* deve se constituir em uma aresta da triangulação *t*. Caso sejam utilizados vértices adicionais na triangulação, tais vértices podem ser adicionados ao interior e às fronteiras de *t*, fazendo com que cada segmento de reta de *c* seja mapeado por uma ou mais arestas colineares de *t*.

c) Domínio geométrico definido por uma curva poligonal com furos

Esta classificação difere da anterior apenas pelo fato das fronteiras do domínio geométrico serem formadas por duas ou mais curvas poligonais disjuntas de Jordan. Estas curvas, além de definirem as fronteiras externas, definem também os furos no interior do domínio.

d) Domínio geométrico definido por um grafo planar de segmentos de reta (GPSR)

Este é o caso mais geral dos domínios geométricos tratados no presente trabalho. Os dados fornecidos são um conjunto de pontos *p* e segmentos de retas *r*, os quais podem se interseccionar apenas nos seus pontos extremos (para o caso de uma seqüência contínua de segmentos). Na triangulação resultante *t*, *r* é um subconjunto das arestas de *t*, e *p* um subconjunto dos vértices de *t*. Caso seja utilizado vértices adicionais em *t*, cada segmento de *r* pode ser formado por um conjunto de arestas colineares de *t*. Esta classificação tem a habilidade de lidar, simultaneamente, com múltiplos domínios, além de poder representar interfaces internas ao domínio geométrico, entre diferentes materiais presentes na definição do problema físico.

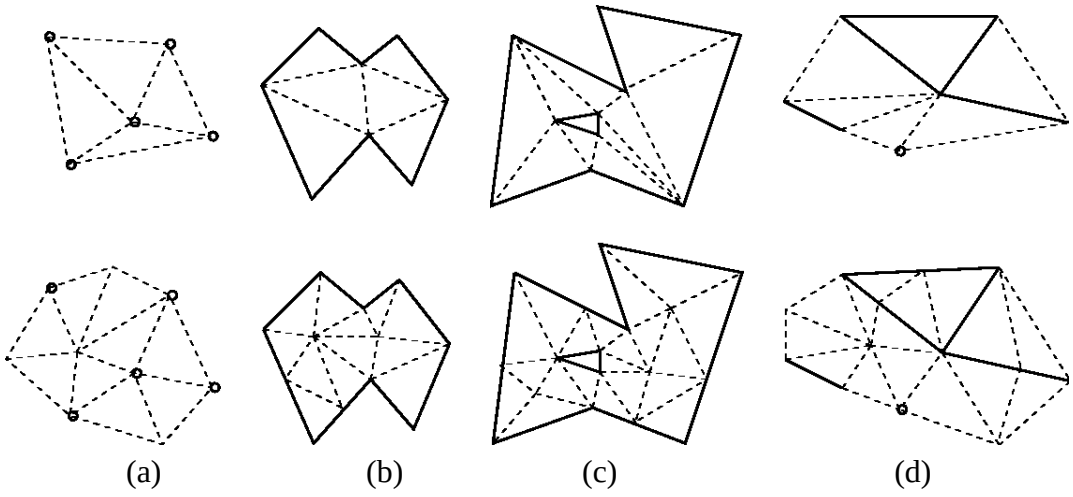


Fig. 8 - Tipos de domínios geométricos definidos anteriormente, triangulados sem pontos adicionais (superiores) e com pontos adicionais (inferiores) [11]

Domínios reais, no entanto, são normalmente compostos por curvas e superfícies não planas. Dentre os diversos modelos de representação geométrica, *NURBS* (*Non-Uniform Rational B-Splines*) é o que vem se destacando e tornando-se padrão na indústria. Este modelo tem a habilidade de representar entidades geométricas analíticas (quadriláteros, círculos, elipses, esferas, torus, etc.) de forma exata e com uma quantidade bastante reduzida de informações.

No entanto, a geração de malhas considerando domínios compostos por curvas é um desafio bastante complexo, e não será tratado neste trabalho. Para podermos manipular com tais domínios, será adotada uma simplificação, onde as curvas são previamente convertidas em conjuntos de segmentos de reta contínuos, passíveis de representação através de um GPSR. Esta é uma consideração razoável, pois resolve grande número de problemas práticos

e contém todos os elementos necessários para o estudo de problemas de triangulação. A Fig. 9 apresenta um exemplo de aproximação de curva por segmentos de retas.

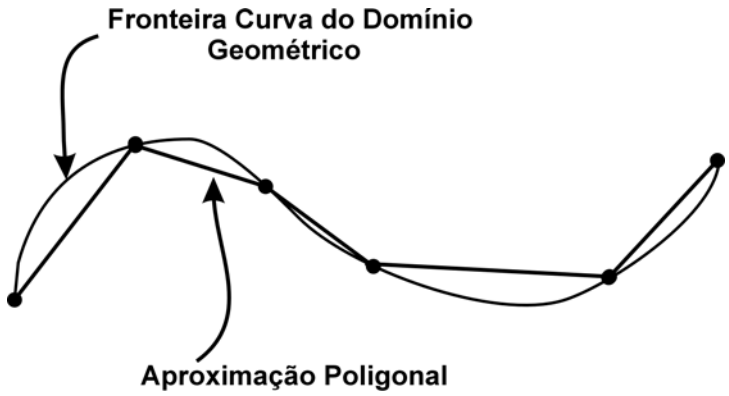


Fig. 9 - Aproximação de curvas por segmentos de retas

2.2. Propriedades desejadas para uma malha

As propriedades desejadas de uma malha são, em geral, definidas pelo número, forma e tamanho dos elementos. O tempo de CPU para obter a malha e a memória utilizada pelo triangulador também são parâmetros importantes, mas são propriedades do triangulador e não da malha resultante. Satisfazer aos critérios da malha e do triangulador são tarefas antagônicas, pois aumentar a qualidade da malha significa, quase sempre, aumentar o esforço computacional. Daí a fertilidade para pesquisas na área de geração de malhas.

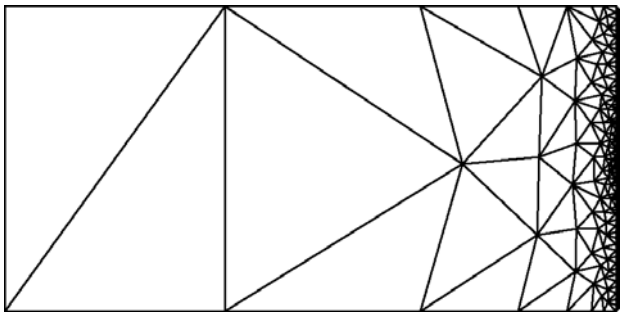


Fig. 10 - Variação do tamanho de elemento [1]

A mais simples das restrições que deve ser observada é a correta representação das fronteiras externas do domínio, tanto em geometrias simples como multiplamente conexas. Crescendo em complexidade, outra restrição importante é o tamanho dos elementos, não apenas como limitação a um valor constante, mas limitando tamanhos em determinadas

regiões com especificação de crescimento por progressões. A Fig. 10 apresenta um exemplo de malha não-estruturada com uma grande variação de tamanho de elemento em uma curta distância, mas com uma variação controlada do tamanho de elementos adjacentes.

Outra fundamental característica que os problemas físicos exigem dos geradores de malhas, e certamente a mais difícil de se conseguir, é o controle sobre a forma dos elementos, o que significa a habilidade de gerar malhas com elementos pouco esbeltos, ou com relações de aspecto próximas a unidade, ou com ângulos internos pré-determinados. Elementos com pequenos ângulos internos normalmente resultam em soluções numéricas com grandes erros e de difícil obtenção. Ângulos internos muito pequenos [16] ou muito próximos de 180° [4], podem levar a soluções inconsistentes. Este comportamento deve-se, normalmente, a problemas de inconsistência da interpolação dentro dos elementos, fato ilustrado na Fig. 11. Utilizando-se uma média aritmética entre os pontos com valores 22 e 80, obtemos um resultado de 51 para o valor no centro da aresta inferior. Se tal interpolação for utilizada para calcular uma derivada na direção transversal, com os valores 48 e 51, o cálculo da derivada terá um erro de truncamento muito diferente do erro de truncamento de uma derivada calculada entre 22 e 80. Isto é, malhas com refinamento completamente diferentes nas duas direções estão sendo usadas no mesmo elemento, gerando coeficientes do sistema linear muito anisotrópicos.

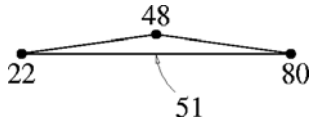


Fig. 11 - Função interpolação no elemento

Outra característica importante do triangulador é a capacidade de gerar malhas com o menor número possível de elementos e, ao mesmo tempo, satisfazer os critérios de otimização nesta condição. Isto é desejável porque a obtenção de uma malha refinada a partir de uma grosseira requer um processo mais simples de particionamento.

A dualidade planar da triangulação [25], também é de muito interesse na área de simulação. O dual de uma triangulação são volumes que podem ser empregados para realização de balanços no método dos volumes finitos. Por exemplo, o método dos volumes finitos baseado no volume de controle (CVFEM) de Baliga e Patankar [8], utilizam o dual de uma triangulação qualquer como volumes de controle. Os trabalhos de Palagi [87],

Marcondes e Maliska [77] e Maliska e Maliska Jr. [74], entre outros, utilizam o dual de uma triangulação de Delaunay, que são os diagramas de Voronoi, como volumes de controle. A Fig. 12 apresenta um exemplo de triangulação e seu dual. Note que nesta figura, no dual, os segmentos de reta que estão conectados à estrutura por apenas um vértice representam a conexão de um elemento da triangulação com o domínio externo.

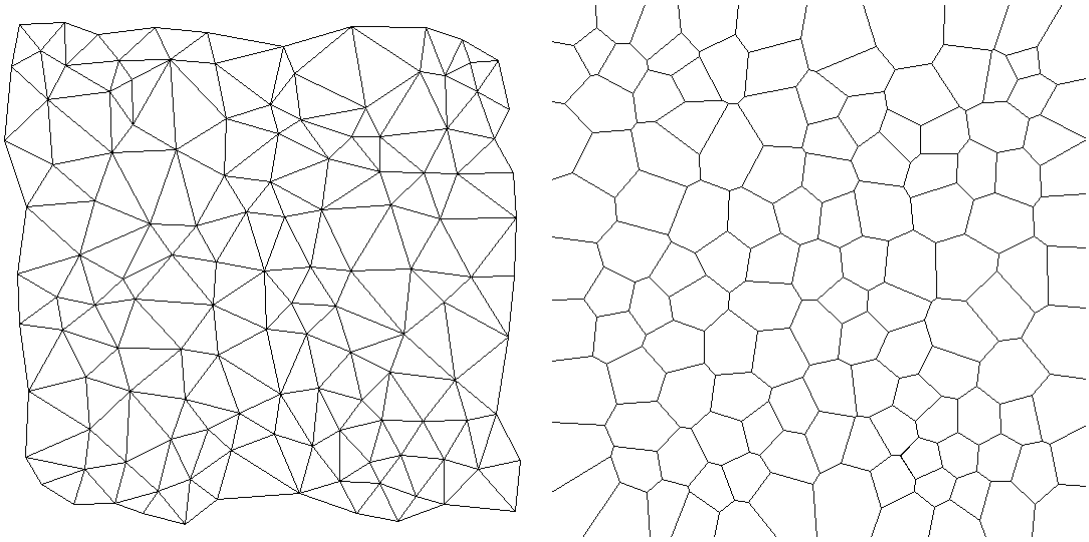


Fig. 12 - Triangulação e seu dual

A Fig. 13 mostra os diagramas de Voronoi obtidos com um outro algoritmo desenvolvido pelo autor [75], onde pode-se constatar a versatilidade da triangulação de Voronoi para gerar volumes de controles hexagonais e polares. Neste método, também incorporado ao algoritmo geral desenvolvido neste trabalho, os diagramas de Voronoi são inicialmente obtidos para depois obter-se a triangulação de Delaunay. Portanto, um procedimento geométrico totalmente diferente do utilizado neste trabalho.

Finalmente, não se pode deixar de comentar sobre mais uma outra importante característica desejável em um gerador de malhas: obter a malha de acordo com o problema físico. Atualmente isto é feito externamente ao gerador, isto é, tendo-se noção da física do problema geramos uma malha que atenda de forma geral ao problema. O desejável seria integrar as condições de contorno e natureza da equação diferencial com o gerador no nível de concepção deste último. Este é, entretanto, um objetivo difícil de ser conseguido. Um paliativo para isso são os geradores acoplados explicitamente com os simuladores, que geram sucessivas malhas enquanto a solução avança, conhecidos como métodos adaptativos. Em função da importância que os métodos adaptativos tem hoje na simulação numérica, neste

trabalho teve-se sempre a preocupação de desenvolver um algoritmo que facilmente possa integrar um método adaptativo.

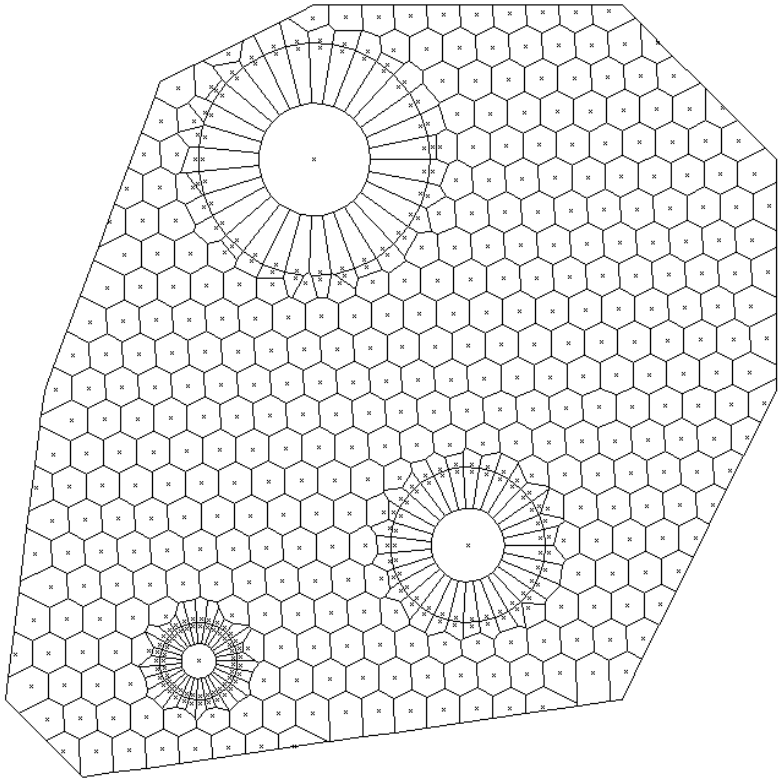


Fig. 13 - Diagrama de Voronoi [73]

2.3. Triangulação de Delaunay-Características

A triangulação de Delaunay foi primeiramente introduzida por Delaunay [23] em 1934, e sua definição para tal diagrama, conforme [13], é apresentada na seção seguinte.

2.3.1. Definição

"*Para definir-se uma triangulação de Delaunay* D *de um conjunto de vértices* V *no plano, define-se primeiramente as seguintes propriedades:*

1. *Qualquer círculo em tal plano é dito* vazio *caso este não contenha nenhum vértice de* V *no seu interior;*
2. *Vértices localizados sobre a circunferência não são considerados no interior do círculo, não invalidando, portanto, o critério anterior;*

3. *Sejam* u *e* v *dois vértices quaisquer de* V. *Um* circuncírculo*, ou círculo circunscrito, da aresta* uv *é qualquer círculo que passe pelos pontos* u *e* v. *Qualquer aresta possui infinitos circuncírculos.*

Com isto, pode-se dizer que a triangulação D *é o grafo que respeita a seguinte regra: a aresta* uv *encontra-se na triangulação* D *se, e somente se, existe um* circuncírculo vazio de uv. *A aresta que satisfaz esta propriedade também é chamada de* aresta de Delaunay.*"*

2.3.2. Propriedades

A triangulação de Delaunay otimiza simultaneamente os seguintes critérios:

1. maxmin ângulo (maximização do mínimo ângulo interno dos triângulo);
2. minmax circuncírculo (minimização do máximo circuncírculo das arestas);
3. minmax min-círculo de contenção (minimização do máximo mínimo-círculo de contenção das arestas), conforme pode ser visto na Fig. 14(c).

O respeito a estes três critérios no processo de geração, dá origem à malhas que são adequadas para a simulação numérica. Esta triangulação, composta de triângulos "bem comportados", pode ser usada em diversas aplicações, sendo largamente empregada em elementos finitos. Conforme já salientado, esta triangulação possui um dual de interesse especial para a discretização de equações diferenciais utilizando o método dos volumes finitos. E a razão é devido a ortogonalidade local entre a triangulação e seu dual, conforme pode ser visto na Fig. 15. Nesta figura o polígono *abcde* é o diagrama de Voronoi, usado como volume de controle para realização dos balanços. As propriedades a serem determinadas pela simulação estão armazenadas nos pontos 1, 2, 3, 4, 5 e 6, centros dos diagramas de Voronoi (volumes de controle). Observe que a linha *1-3* corta *b-c* ortogonalmente, o que deixa o cálculos de derivadas normal à face *c-b* extremamente simples. Além disso, *b-c* corta *1-3* exatamente na metade, o que facilita a avaliação de propriedades na interface do volume de controle. Em resumo, a ortogonalidade local reduz o método numérico em um método muito semelhante ao cartesiano, mas com todas as versatilidade de um método de malhas não-estruturas que se conformam com a fronteira do domínio.

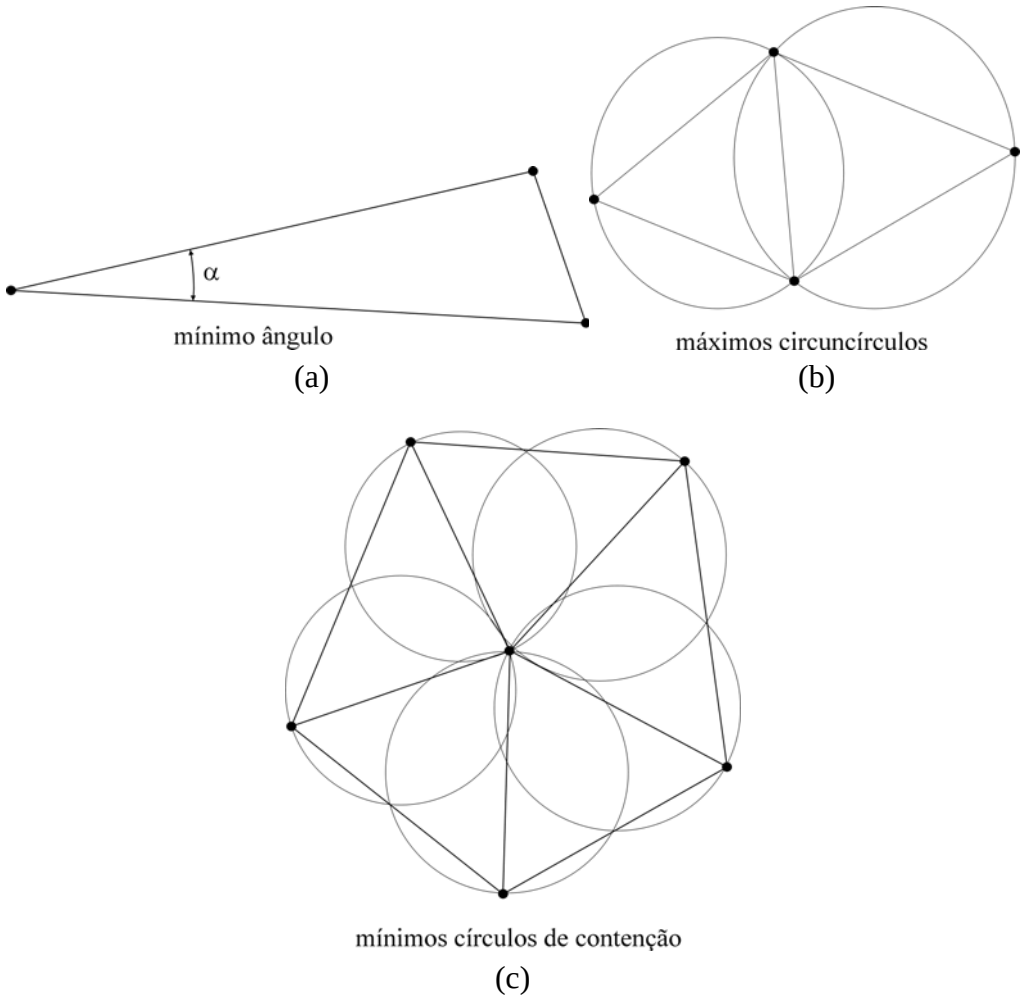


Fig. 14 - Mínimo ângulo (a), máximo circuncírculo (b) e mínimo círculo de contenção (c)

A triangulação de Delaunay conforme definida anteriormente, de um dado conjunto de pontos, é única. Isto porque se obtém uma forma fechada e não-ambígua, para se determinar a presença ou não, de uma aresta de Delaunay entre dois pontos quaisquer do conjunto de pontos fornecido. Uma rápida análise mostra que qualquer aresta pertencente ao envelope convexo de um conjunto de pontos também irá pertencer à triangulação de Delaunay de tais pontos. Para demonstrar, basta iniciarmos com um círculo de tamanho igual à distância entre os dois pontos da aresta e crescermos com tal círculo para fora da triangulação. Quanto mais aumentarmos o raio do círculo, menos curvo (mais próximo de um segmento de reta) torna-se o arco de circunferência formado entre os dois pontos. Como o envelope convexo é a curva poligonal convexa que, necessariamente, engloba todos os pontos, sempre existe um raio no qual nenhum ponto da triangulação reside dentro da circunferência supracitada, mostrando

que tal aresta faz parte da triangulação de Delaunay. A Fig. 16 complementa a explicação anterior.

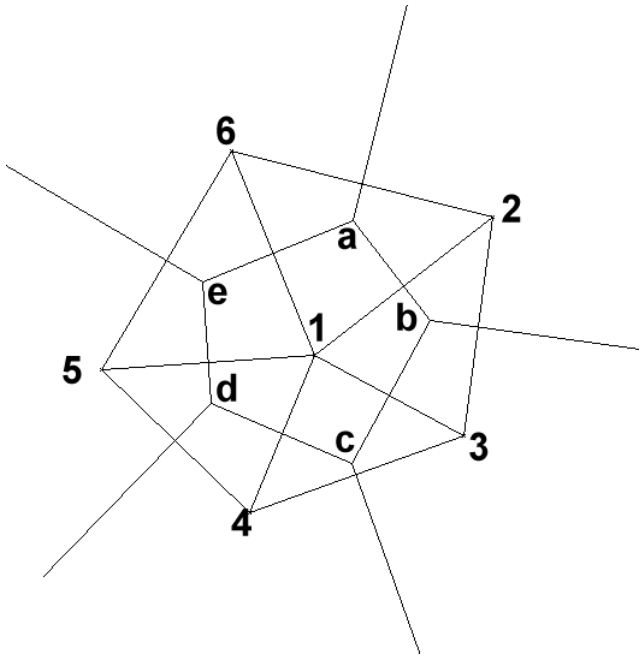


Fig. 15 - Ortogonalidade local entre a triangulação de Delaunay e o diagrama de Voronoi

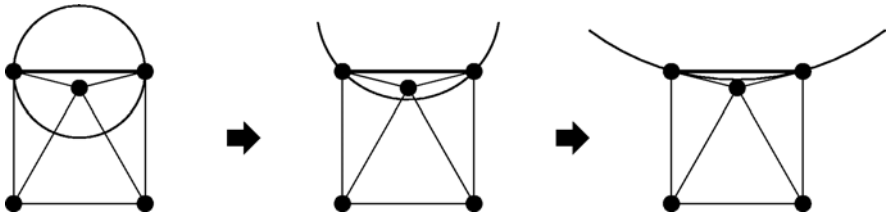


Fig. 16 - Aresta de Delaunay no envelope convexo

Da mesma forma, toda a aresta que conecta os dois vértices mais próximos é uma aresta de Delaunay. Isto porque o círculo diametral da aresta formada pelos dois vértices necessariamente não engloba nenhum outro vértice, pois este círculo tem o diâmetro igual a menor distância entre dois vértices da triangulação, e por sua vez, a aresta é Delaunay.

Uma propriedade não tão simples de ser mostrada, e de fundamental importância, é a de que o conjunto das arestas de Delaunay de um dado conjunto de pontos, forma uma triangulação válida. Para a definição anterior, no entanto, esta propriedade só é verdade, quando os pontos estão em uma configuração denominada *posição geral*. Por *posição geral* entende-se que não existem quaisquer quatro pontos do conjunto de pontos fornecido que

residam sobre a mesma circunferência A demonstração de tal propriedade é algo não intuitivo e de razoável complexidade, fora do escopo deste trabalho, e pode ser encontrada em [11].

Outra propriedade interessante da triangulação de Delaunay é a propriedade do *circuncírculo vazio*. É denominado de circuncírculo de um triângulo o círculo único que passa pelos três vértices de tal triângulo. Esta propriedade define que, assim como uma aresta, um triângulo é dito de Delaunay se, e apenas se, o seu circuncírculo for vazio. Isto significa dizer que não existe um único vértice da triangulação que resida dentro de tal triângulo. A definição de circuncírculo de Delaunay, por sua vez, decorre de forma bastante direta da definição de aresta de Delaunay. A Fig. 17 apresenta graficamente esta propriedade.

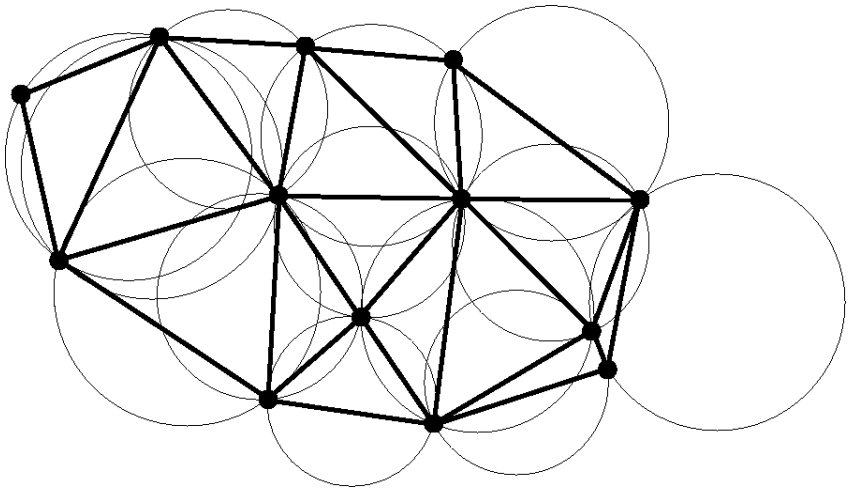


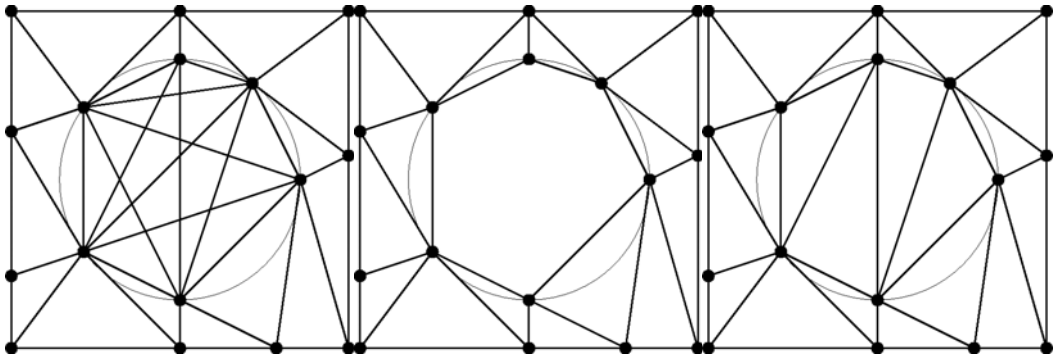
Fig. 17 - Triangulação de Delaunay e seus respectivos circuncírculos [99]

2.3.3. Degeneração/Singularidades

Dos poucos problemas apresentados pela triangulação de Delaunay, provavelmente a degeneração é o principal deles. Este comportamento aparece nesta estrutura geométrica quando existem quatro ou mais pontos cocirculares no conjunto dos pontos de entrada. Nesta situação singular, a triangulação de tais vértices não é única e por isso contém arestas cruzadas, invalidando a mesma. Técnicas relativamente simples podem resolver o problema das arestas cruzadas, e conseqüentemente o problema da invalidade da triangulação [31][35][85]. No entanto, não existe solução para o principal problema acarretado pela degeneração: pequenas perturbações morfológicas nos pontos de entrada acarretam mudanças topológicas na triangulação final. Este comportamento faz com que os métodos falhem com

freqüência na obtenção de triangulações com combinações de pontos deste tipo, e devido a isso, diz-se que tal triangulação sofre uma degeneração.

A Fig. 18 ilustra um exemplo de uma triangulação de Delaunay degenerada, onde vemos 6 pontos co-circulares, originando triangulação com arestas cruzadas (Fig. 18(a)). Para evitar que as arestas de uma triangulação de Delaunay cruzem-se na geração de triangulações com pontos co-circulares, normalmente adota-se uma definição modificada para tal malha. Algumas delas eliminam as arestas cruzadas, resultando em polígonos com mais de três lados dentro da malha (Fig. 18(b)). Estas definições modificadas são interessantes por representar corretamente a topologia do relativo diagrama de Voronoi. No entanto, tal triangulação continua ainda inválida, não sendo interessante para aplicações práticas.



(a) (b) (c)

Fig. 18 - Triangulação de Delaunay degenerada [99]

A modificação na definição original da triangulação de Delaunay a torna robusta e de utilidade prática, por sua vez, consiste na eliminação das arestas cruzadas e posterior triangulação, por um método qualquer, dos polígonos internos remanescentes com mais de três lados, conforme Fig. 18(c). Novamente, neste caso, a triangulação de Delaunay não é única, pois podemos escolher diferentes formas válidas de triangular tais polígonos de alta ordem. Esta liberdade, no entanto, não invalida a definição de Delaunay, pois, qualquer que seja a escolha dos triângulos, esta não implica em nenhuma alteração dos parâmetros ótimos de tal triangulação.

Para que não seja necessário considerar-se, em análises posteriores o caso da degeneração, passar-se-á a utilizar a seguinte definição de triangulação de Delaunay: *uma aresta ou triângulo é dito fortemente Delaunay caso esta possua um circuncírculo tal que*

nenhum vértice resida dentro ou sobre tal circuncírculo, exceto pelos vértices da própria aresta ou triângulo.

Portanto, sempre que não tivermos quatro ou mais vértices co-circulares nos pontos fornecidos, todas as arestas e triângulos de Delaunay serão, necessariamente, fortemente Delaunay.

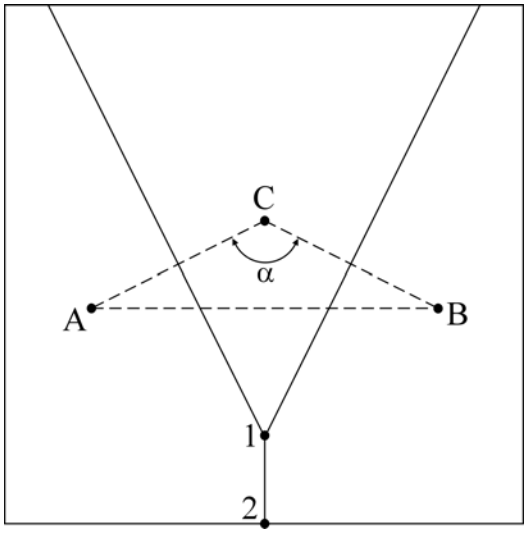


Fig. 19 - Degeneração do diagrama de Voronoi

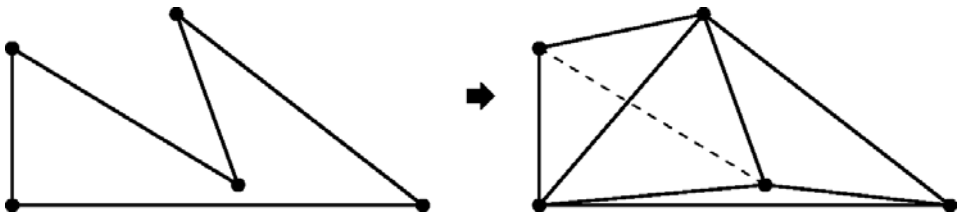
Apesar de, formalmente, não ser considerada uma degeneração da triangulação de Delaunay, no jargão da área numérica chama-se de degeneração o diagrama de Voronoi obtido com triângulos que possuam ângulos obtusos. A Fig. 19 apresenta este caso, onde a triangulação de Delaunay aparece em linhas tracejadas, e o ângulo obtuso é indicado por α. No diagrama de Voronoi, em linhas cheias, podemos observar que a linha que une os pontos nodais *A* e *B* passa externamente à linha *1-2*, é a área de passagem dos fluxos da propriedade em consideração. Com esta configuração, a avaliação de derivadas e de valores da função nas interfaces do volume de controle estarão fora da área que representam. No gerador desenvolvido neste trabalho, isto é evitado utilizando-se um método de refino, que elimina os ângulos obtusos, conforme descrito no Cap. 4.

2.3.4. Triangulação de Delaunay Restrita

Como vimos anteriormente, dado um conjunto de pontos, a triangulação de Delaunay, em duas dimensões, maximiza o mínimo ângulo interno dos triângulos. Ou seja, não existe nenhuma outra triangulação possível para o conjunto de pontos fornecido que contenha

ângulos maiores que a triangulação de Delaunay. Por que, então, o problema de geração de malha não está totalmente resolvido, já que encontramos uma malha ótima para tal critério?

Existem dois motivos, e são ilustrados na Fig. 20. O primeiro motivo é que não basta a triangulação conformar-se aos pontos do domínio fornecido. É necessário levar-se também em consideração os segmentos. Estes segmentos, por sua vez, podem não estar presentes na triangulação, conforme pode ser visto na Fig. 20, onde a triangulação obtida (b) não contém todas as arestas do domínio fornecido (a). Com isto, adaptações à triangulação devem ser efetuadas de forma a respeitar esta restrição. Resumindo, queremos encontrar um algoritmo que obtenha triangulações de Delaunay em um domínio do tipo GPSR (composto por pontos e segmentos de reta). O segundo motivo pode ser observado no triângulo achatado da base da Fig. 20(b), o qual obedece a regra de Delaunay, mas não possui uma boa forma geométrica para a simulação, pois possui dois ângulos muito pequenos, e um ângulo próximo de 180^o..



(a) (b)

Fig. 20 - Problemas adicionais da triangulação de Delaunay

Para cada um destes problemas, existe uma solução. Para o caso da triangulação respeitar as arestas do domínio fornecido, definiremos, em seguida, uma variante da triangulação de Delaunay: a triangulação de Delaunay Restrita (TDR) que utiliza-se de um novo conceito, denominado visibilidade para redefinir a regra de Delaunay para o circuncírculo. Para o problema da qualidade dos elementos da triangulação podem ser utilizados pontos de Steiner na triangulação.

Para a definição de uma TDR, consideremos um GPSR qualquer X. Dois pontos no interior deste GPSR são visíveis, caso o segmento de reta formado por estes dois pontos não interseccione nenhuma aresta de X, além de nos seus pontos extremos. Por sua vez, uma aresta ou triângulo é denominado de *Delaunay restrito* caso ele satisfaça as duas seguintes condições: primeiro, o seu interior não intersecciona nenhuma aresta do domínio fornecido (a não ser que ela seja a própria aresta do domínio); segundo, ele possui um circuncírculo que não engloba nenhum vértice de X que seja visível do interior da aresta ou triângulo. Uma

TDR é uma triangulação onde todas as arestas e triângulos são *Delaunay restritos*. Por sua vez, uma TDR nada mais é que uma TD com esta nova definição de aresta ou triângulo de Delaunay.

A Fig. 21 apresenta dois exemplos onde a aresta *e* e o triângulo *t* não respeitam a regra de Delaunay original, mas respeitam a regra de Delaunay restrita. Os segmentos do domínio fornecido são apresentados em negrito. Apesar de *e* não possuir um circuncírculo vazio, o circuncírculo apresentado engloba apenas vértices que não são visíveis por *e*. Existem dois vértices dentro do circuncírculo, mas os dois são escondidos pelas arestas do domínio fornecido. O mesmo ocorre para o triângulo *t*, onde os dois vértices, englobados pelo circuncírculo apresentado na figura, não são visíveis do interior de *t*. Tais triangulação são, então, ditas triangulações de Delaunay restritas.

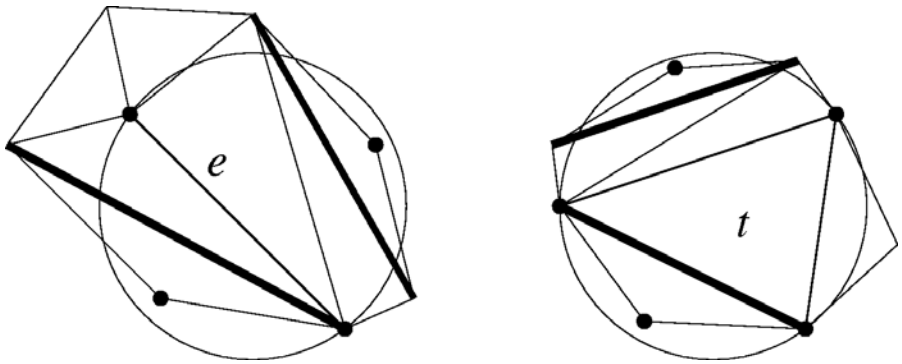


Fig. 21 - Aresta (a) e triângulo (b) de Delaunay restritos

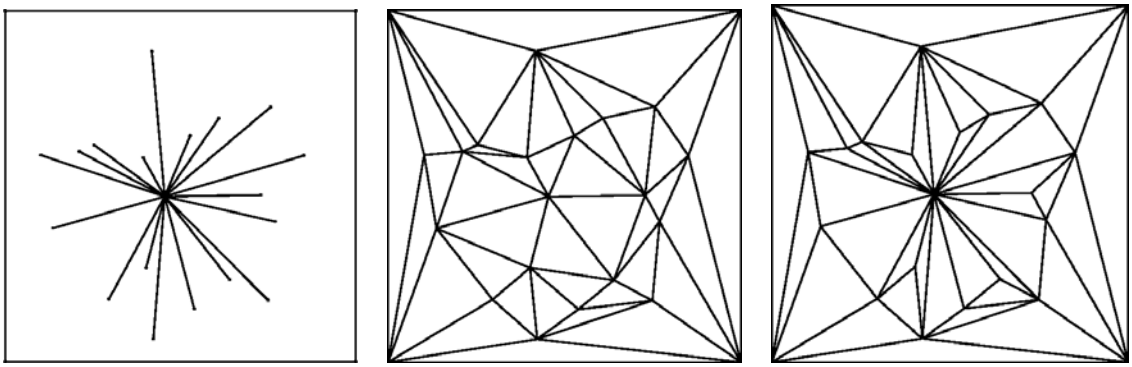
Com esta nova definição, a TDR, podemos facilmente obter triangulações que respeitem os segmentos do domínio, segmentos estes que, não necessariamente, estão presentes na triangulação de Delaunay. Para obtermos a TDR é bastante simples; basta se construir a TD por qualquer um dos métodos existentes descritos no próximo capítulo e utilizar o algoritmo de inserção de aresta (seção 3.1.1.2), onde a triangulação dos dois polígonos resultantes da remoção das arestas seccionantes da aresta inserida deve respeitar a regra de Delaunay restrita. Para a triangulação destes polígonos simplesmente convexos existem algoritmos extremamente eficientes.

Outra forma de obtermos a TDR é com uma versão modificada do algoritmo de inversão de aresta. Como o algoritmo de inversão de aresta, por sua vez, necessita de uma triangulação inicial, a triangulação fornecida ao método deve conter todas as arestas do

domínio fornecido. Assim, podemos começar o processo de triangulação através dos pontos do domínio aplicando o algoritmo de triangulação qualquer. Após este algoritmo, inserimos os segmentos do domínio, preenchendo os polígonos resultantes da inserção dos segmentos também com uma triangulação qualquer. Finalmente, aplica-se o algoritmo de inversão de aresta modificado para obtermos a TDR (seção 3.1.1.1). O algoritmo de inversão de aresta modificado difere do original, apenas pelo fato de que as arestas do domínio fornecido nunca são adicionadas à lista de possíveis arestas invertidas. Com isto, estas arestas nunca são removidas, e a GPSR é preservada na triangulação.

Assim como a TD é o dual do diagrama de Voronoi, a TDR é o dual do *diagrama de Voronoi delimitado* [67][96][111]. O diagrama de Voronoi delimitado é uma construção geometria que divide o plano em diversas células, uma para cada vértice fornecido, tal que cada célula é delimitada pela região do plano para qual *a* é o ponto visível mais próximo. Veja que esta definição é exatamente igual a do diagrama de Voronoi, exceto pela utilização do conceito de visibilidade para definição do ponto mais próximo.

Um último exemplo é mostrado na Fig. 22. Primeiramente, o GPSR fornecido na Fig. 22(a), a triangulação de Delaunay de seus vértices Fig. 22(b), e finalmente, a triangulação de Delaunay restrita Fig. 22(c). Pode-se observar que alguns dos vértices da TDR são Delaunay restritos, mas não são Delaunay. Com isto, é importante não se deixar confundir pela denominação, e sempre lembrar que uma triangulação de Delaunay restrita não é necessariamente uma triangulação de Delaunay.



(a) (b) (c)

Fig. 22 - Etapas de obtenção de uma triangulação de Delaunay restrita

2.4. Outras triangulações ótimas

Esta seção apresenta, de forma muito sucinta, tipos de triangulações denominadas ótimas, ou seja, que respeitam algum critério de otimização da malha. Estes métodos são aqui listados, principalmente para fornecimento da bibliografia sobre triangulação.

2.4.1. Minmax comprimento de aresta

Dos diversos trabalhos no campo de otimização de critérios de triangulações, apenas alguns poucos estudos consideram o problema topológico, ou seja, otimização do grau dos vértices [58] ou da conectividade [29].

A grande maioria das triangulações ótimas concentra-se na otimização de parâmetros morfológicos, como por exemplo, o comprimento de aresta e o mínimo ângulo interno. O comprimento de aresta é um parâmetro bastante interessante, pois este afeta diretamente os erros de truncamento da aproximação numérica [100].

Um dos primeiros trabalhos no campo de otimização foi de Edelsbrunner e Tan [35], que consideraram a minimização do máximo comprimento de aresta. Eles demonstraram que esta triangulação, assim como a triangulação de Delaunay, contém as arestas da árvore mínima de separação das arestas. A árvore mínima de separação das arestas é a árvore que conecta os vértices através da menor distância possível. Em razão disso, é possível encontrar-se a triangulação em um tempo de ordem $O(n^3)$. Com algumas operações não triviais este tempo pode ser reduzido para $O(n^2)$.

2.4.2. Greedy

Esta triangulação minimiza o vetor ordenado de comprimentos de aresta, o que não necessariamente implica em minimizar o número de arestas da triangulação [48][59]. Nesta triangulação, os métodos de geração são incrementais através da inserção de arestas, e não da inserção de pontos, como usualmente é feito. Inicia-se com o conjunto de pontos que se deseja triangular, e escolhe-se, seqüencialmente, a aresta de menor distância disponível, sem que esta cruze as arestas já definidas.. Sabe-se que sem a utilização de pontos de Steiner para um dado conjunto de pontos de entrada, independente do critério de otimização utilizado, o

número de elementos obtidos na triangulação é sempre o mesmo. Isto pode ser facilmente mostrado através de uma análise do grau dos vértices de uma triangulação qualquer.

Para um conjunto de pontos qualquer, as operações necessárias para obter uma triangulação de Greedy podem ser computadas em tempos da $O(n^2)$ através da manutenção continua de uma triangulação de Delaunay, para auxiliar a descobrir as arestas da triangulação de Greedy. Para domínios poligonais convexos [59] ou conjunto de pontos randômicos [30], é possível reduzir a ordem do tempo de computação para $O(n)$.

2.4.3. Ponto mais distante

Outra triangulação de aplicação prática, primeiramente estudada por Eppstein [38], minimiza o mínimo ângulo interno dos triângulos. O particionamento resultante é denominado de *triangulação de Delaunay do ponto mais distante*, e suas operações, novamente, consomem um tempo de ordem $O(n)$ [1]. Esta triangulação dualiza o diagrama de Voronoi do ponto mais distante (oposto ao tradicional, que poderia ser chamado de *diagrama de Voronoi do ponto mais próximo*). Esta geométrica é utilizada para se descobrir o ponto mais distante de um ponto escolhido arbitrariamente no espaço. Apesar de poder ser definida para um conjunto qualquer de pontos, normalmente esta partição não é uma triangulação válida, já que suas arestas conectam vértices apenas no envelope convexo e que, provavelmente, se interseccionam.

2.4.4. Maxmin da altura do triângulo

Consideremos agora o problema de obtenção de uma triangulação maximizando a altura mínima dos triângulos. Este problema aparece no algoritmo de triangulação tridimensional de Mitchell e Vavasis [80], e está diretamente relacionado com a qualidade da aproximação de curvas por segmentos de retas [50].

2.4.5. Minimização do peso dos triângulos (MPT)

O problema que ainda encontra-se sem solução até hoje em geometria computacional é a triangulação de um conjunto arbitrário de pontos minimizando o peso dos triângulos [34]. Minimizar o peso dos triângulos significa minimizar o comprimento total das arestas. Por isso, os primeiros autores a trabalharem com geometria computacional costumavam

referenciar-se a este problema como sendo o “problema da triangulação ótima”, já que trabalha conjuntamente com as três arestas do triângulo.

Garey e Johnson **[48]** incluíram o problema de MPT à sua famosa lista de problemas sem solução ou com solução em tempo polinomial. Se generalizarmos o problema de MPT, de forma que o peso de cada triângulo não seja uma função diretamente relacionada ao comprimento total das arestas, o problema de MPT passa a ser solucionável (no entanto, não necessariamente em tempo polinomial) **[66]**. Para esta nova definição de MPT, chamamos MPT aproximado, ou MPTA, à qual os estudos de otimização dedicaram consideráveis esforços.

3. Triangulação - Métodos

No capítulo anterior foi apresentada uma breve análise sobre os diversos tipos de triangulações, suas propriedades e métodos de geração. No presente capítulo, focalizaremos as atenções em um tipo específico de triangulação, denominada *triangulação de Delaunay*. Os fundamentos desta estrutura geométrica, métodos de geração da triangulação e variantes da mesma para aplicações práticas são apresentados, com ênfase em aspectos voltados para geração de malha, considerando a qualidade dos elementos e conservação das fronteiras do domínio geométrico fornecido. Este capítulo também apresenta uma discussão sobre métodos de melhoramento da malha, incluindo os métodos de suavização e de refino.

3.1. Métodos de triangulação de Delaunay

Dos diversos componentes que fornecem funcionalidades a um gerador de malhas, como filtros de entrada/saída, manipulador geométrico, algoritmo de discretização, interface gráfica, visualização, entre outros, o principal deles, e no qual reside o desempenho do gerador, é o algoritmo de discretização e sua respectiva estrutura de dados.

Os diversos algoritmos de triangulação de Delaunay disponíveis na literatura podem ser divididos em dois grandes grupos: diretos e incrementais. Os algoritmos diretos têm como característica fundamental a necessidade de saber de antemão o conjunto completo de vértices envolvidos na geração, enquanto os algoritmos incrementais precisam saber apenas sobre a triangulação atual e o novo vértice que será adicionado. Para se adicionar um único vértice utilizando um algoritmo direto, é necessário recalcular-se toda a triangulação. Desta forma, os algoritmos diretos são interessantes para se obter malhas a partir apenas da definição da geometria e de pontos previamente definidos, ou seja, no processo de gerar a triangulação inicial e não na etapa de refino. Nesta função, os algoritmos diretos costumam ser bem mais rápidos que os algoritmos incrementais. É importante lembrar que, quase sempre, a triangulação inicial está distante de ser uma malha adequada para simulação numérica. Os métodos de refino é que transformam a triangulação inicial em uma malha com boas características para simulação. Os algoritmos incrementais, por sua vez, são interessantes para

serem utilizados em processos de refino de malha ou em simuladores adaptativos. Isto porque a adição e remoção de pontos é uma operação meramente local, não envolvendo toda a malha. Modificações locais à triangulação são efetuadas de forma que esta contenha o novo vértice e mantenha as propriedades de Delaunay.

Os algoritmos mais conhecidos de geração de triangulação de Delaunay disponíveis na literatura são o *incremental* de Lawson [69], o *divide-and-conquer* de Lee e Schachter [64] e o *plane-sweep* de Fortune [42]. Os tempos de geração entre estes três algoritmos são bastante similares, no entanto, o *divide-and-conquer*, segundo Shewchuk, [101], é levemente mais rápido, vindo o *plane-sweep* em segundo e o *incremental* por último.

O algoritmo *incremental*, apesar de ser o candidato natural a ocupar o primeiro posto de performance, consome a maioria do tempo com a localização do triângulo em que o novo ponto vai ser inserido. Apesar desta operação poder ser otimizada com diversas técnicas, como árvores de busca e *caching* de triângulos, o método incremental ainda fica em último lugar quando a geração de triangulações com um grande número de vértices for de interesse. Além disto, as técnicas de otimização da operação de localização de ponto podem ser facilmente derrubadas através de determinados arranjos de pontos, os quais, infelizmente, acontecem com freqüência na prática (estas técnicas supõem uma distribuição aleatória de pontos, o que normalmente não ocorre).

Apesar de não serem tão eficientes, os algoritmos incrementais podem também, se desejado, ser utilizados para a geração de malhas a partir de um conjunto pré-definido de pontos, ou seja, não na etapa de refino. Para isto, basta criarmos uma triangulação vazia e adicionarmos os vértices, um a um. Esta abordagem é interessante, pois existe um conjunto de algoritmos incrementais que podem ser generalizados para n-dimensões.

Shewchuk [101] apresenta maiores informações de resultados comparativos entre os métodos citados anteriormente, com a utilização ou não de aritmética exata para o aumento da robustez. Algoritmos utilizando aritmética exata, apesar de serem mais lentos que os algoritmos utilizando aritmética de ponto flutuante, falham apenas com um número muito grande de pontos. Maiores detalhes sobre o uso de aritmética exata aplicada a geração de triangulações de Delaunay podem ser encontrados em [93][67][25][44].

Em seguida, são apresentados alguns dos algoritmos de geração de triangulação de Delaunay disponíveis na literatura, classificados como diretos e em seguida os incrementais.

3.1.1. Métodos Diretos

3.1.1.1. Inversão de aresta

O algoritmo de inversão de aresta, assim como outros aqui apresentados, é um algoritmo apenas de otimização, e não de geração. Ou seja, parte-se de uma triangulação inicial já existente, modificando-a de forma a otimizar os critérios desejados. A triangulação inicial, por sua vez, pode ser obtida utilizando qualquer método de geração, como por exemplo, o método de triangulação qualquer apresentado adiante em 3.2.1.

O método de inversão de aresta é baseado em uma otimização local, onde se utiliza a definição de uma aresta localmente Delaunay, para decidir quando se deve executar uma inversão ou não de uma aresta escolhida. O algoritmo funciona da seguinte forma: cria-se uma lista de arestas inicialmente vazia. Visita-se, então, todas as arestas da triangulação realizando o seguinte teste para cada aresta *e*: considere o quadrilátero *q* formado pelas arestas externas da fusão dos dois triângulos adjacentes a aresta *e* (ver Fig. 23(a)). A aresta *e* é inválida, caso ela não respeite a regra da aresta de Delaunay no que diz respeito à triangulação local (ver Fig. 23(b)). Isto é equivalente a dizer que a soma dos ângulos internos dos vértices opostos a aresta *e* é maior que 180°, ou que o menor ângulo dos triângulos utilizando a aresta *e* é menor que os triângulos utilizando a aresta inversa, ou finalmente, que a aresta *e* é maior que a aresta inversa. Caso esta aresta seja inválida, ela é adicionada à lista de arestas.

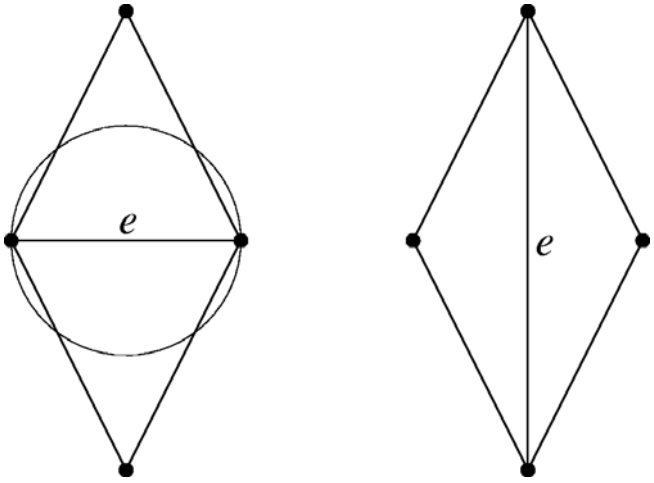


Fig. 23 - Triangulação local do método de inversão de aresta

Após percorrida toda a triangulação, a lista inicial informa quais arestas não respeitam a regra de Delaunay. Para cada entrada desta lista, opera-se da seguinte forma: verifica-se se a aresta respeita a regra de Delaunay ou não. Caso positivo, remove-se a aresta da triangulação. Caso negativo, reverte-se a aresta da mesma forma que o algoritmo de relaxação de malha, e adiciona-se as quatro arestas vizinhas à lista de possíveis arestas inválidas. Veja que as arestas vizinhas devem ser adicionadas à lista de arestas possivelmente inválidas pois ocorreu uma mudança local na topologia da triangulação. Esta operação é efetuada repetidamente, até que não exista mais nenhuma aresta na lista, resultando em uma triangulação apenas com arestas de Delaunay, que nos leva a uma triangulação de Delaunay.

Um aspecto importante a ser observado é que a lista criada representa o conjunto das arestas *possivelmente* inválidas, e não *certamente* inválidas. Isto se deve ao fato de que algumas inversões podem validar não apenas a aresta em questão, mas algumas arestas vizinhas também, deixando uma aresta válida em tal lista. A única afirmação que pode ser feita é que caso a aresta não esteja em tal lista, certamente ela é uma aresta válida.

O algoritmo de inversão de aresta é um exemplo de um algoritmo que otimiza um critério global baseado em uma otimização local (são considerados apenas os dois triângulos adjacentes à aresta quando é verificado se tal aresta respeita a regra de Delaunay). Conforme apresentado anteriormente, uma otimização local não necessariamente leva a uma otimização global. Neste caso, no entanto, isto sempre ocorre porque a triangulação de Delaunay em um espaço de ordem mais elevada pode ser encarada como uma função monotônica, que não apresenta mínimos locais, conforme discutido por Bern e Eppstein em [13]. O algoritmo de inversão de aresta é parecido com o método de relaxação de malha. Este, no entanto, baseia-se na checagem de parâmetros morfológicos, enquanto que o método de relaxação de malha baseia-se na checagem de parâmetros topológicos.

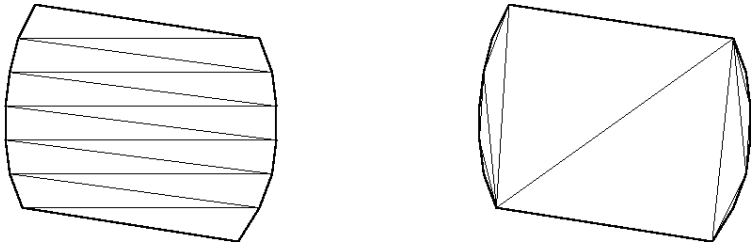
A extensão do algoritmo de inversão de aresta para obter uma triangulação de Delaunay restrita (ver seção 2.3.4), mais especificamente, uma triangulação de Delaunay de um GPSR, basta nunca incluirmos as arestas da triangulação, que formam os segmentos do GPSR, à lista de arestas possivelmente inválidas. Com isto, nunca ocorre a inversão de tais arestas, e as mesmas são preservadas, resultando em uma triangulação de Delaunay restrita.

Através de uma análise mais elaborada, pode-se mostrar que o algoritmo de inversão de aresta utiliza da ordem de $O(n^2)$ inversões para executar uma otimização de Delaunay. Ou

seja, o tempo de execução é da ordem de $O(n^2)$. Para triangulação de geometrias mais complexas ou de grande quantidade de pontos, este tempo, por sua vez, pode ser demasiadamente grande, e tornar a geração inviável. Com isto, Lee e Schachter [64] criaram um algoritmo mais eficiente, denominado *divide-and-conquer*, que é apresentado na próxima secção.

Dado o sucesso do algoritmo de inversão de aresta, diversos outros critérios, além do critério da aresta de Delaunay, foram utilizados com o método, como regra para a inversão ou não da aresta (diagonal do quadrilátero – ver Fig. 23). Como exemplo de outros critérios, tem-se o grau do vértice (que é, na verdade, o método de relaxação de malha), máximo ângulo interno dos triângulos, comprimento total de aresta e razão das áreas do círculo inscrito e do triângulo. Nestes casos, o algoritmo de inversão de aresta obtém uma otimização local mas, no entanto, não pode garantir uma otimização global.

O problema do algoritmo de inversão de aresta para a otimização destes outros parâmetros é que o mesmo pode ficar preso em um ótimo local, no qual a operação de inversão não otimiza a triangulação globalmente (um ótimo local, por sua vez, pode estar bastante distante de um ótimo global). Um exemplo prático deste problema pode ser visto na Fig. 24, onde o critério de otimização é o comprimento total de aresta. Pode-se ver facilmente que o comprimento total das arestas da triangulação de (a) é maior que de (b). A operação de inversão de aresta, por sua vez, não consegue sair de (a) e chegar em (b), falhando para a otimização de tal caso.



(a) (b)

Fig. 24 - Inversão de aresta sem melhora do critério de otimização [11]

A possibilidade do algoritmo de inversão de aresta ficar preso em um ótimo local deve-se ao fato de que a operação de inversão tem uma abrangência espacial muito restrita, onde apenas os quatro pontos mais próximo são afetados (os quatro pontos do quadrilátero, sendo dois deles vértices da aresta original, antes da inversão, e os outros dois vértices da nova

aresta, após a inversão). Para a triangulação de Delaunay, o algoritmo de inversão de aresta sempre obtém uma otimização global, pois a derivada da função de otimização nunca troca de sinal.

3.1.1.2. Inserção de aresta

O problema do método de inversão de aresta ficar preso a um ótimo local, quando da otimização de mais parâmetros, pode ser resolvido com o método de inserção de aresta.

A inserção de aresta consiste na seguinte operação: incluir uma aresta *e* desejada na triangulação (Fig. 25(a)). Em seguida, elimina-se todas as arestas seccionadas pela aresta *e*, formando dois polígonos simples de cada lado da aresta (Fig. 25(b)). Finalmente, este dois polígonos simples devem ser triangulados, o que pode ser feito por um algoritmo que forneça diretamente a triangulação conforme os critérios desejados, ou por uma triangulação qualquer, onde serão efetuadas otimizações posteriores na tentativa de se obter uma otimização global (Fig. 25(c)).

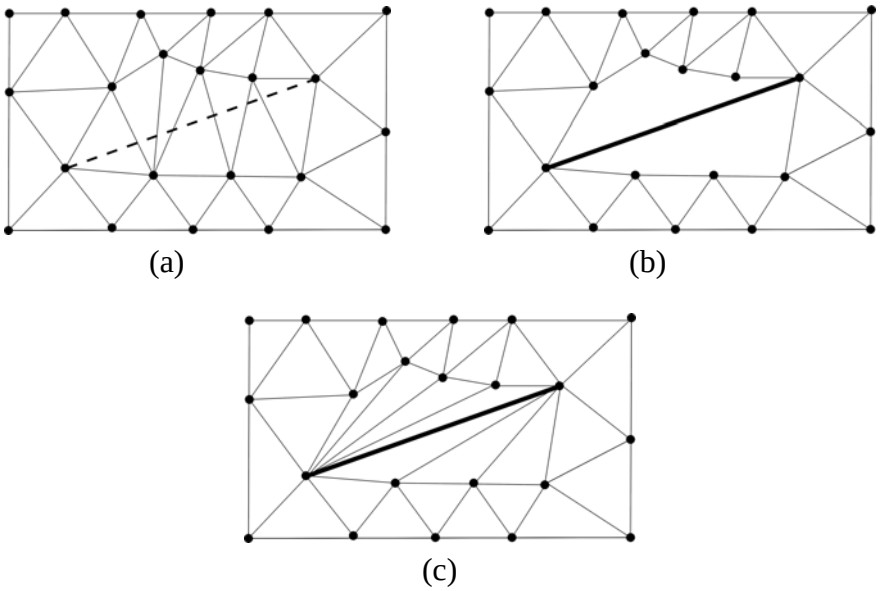


Fig. 25 - Processo de inserção de uma aresta em uma triangulação

Observe que dado uma aresta qualquer *e*, a inversão da aresta *e* é equivalente a inserção da aresta inversa a *e*, pois *e* será a única aresta seccionada pela aresta inversa (e, assim, removida), e os dois polígonos resultantes não precisam ser triangulados, pois estes já são os próprios triângulos.

Com isso, em termos de algoritmo de otimização, podemos encarar a inserção de aresta como uma generalização do algoritmo de inversão de aresta, onde é possível, em um único passo, atingir-se pontos mais distantes na função de otimização. A inserção de aresta, então, pode permitir que se escape de um mínimo local, através de um grande salto na função de otimização. A aplicação da inserção de aresta, no entanto, não é trivial, já que existem diversas opções de arestas a serem utilizadas (quaisquer dois vértices da triangulação formam uma aresta que pode ser inserida). Existem alguns métodos disponíveis na literatura para se escolher tais arestas, sendo que estes, normalmente, utilizam-se vértices dos piores triângulos, segundo o critério que deseja-se otimizar, para montar tais arestas.

Um fato importante a ser observado é que nem sempre a inserção de aresta leva a um ponto melhor na função de otimização. Esta apenas leva à um ponto diferente, distante em tal função. Caso a função de otimização avaliada após a inserção da aresta seja pior que antes de tal inserção, rejeita-se tal aresta, voltando para a triangulação antiga e testando-se uma nova aresta a ser inserida, até que uma otimização dos critérios desejados ocorra.

O algoritmo de inserção de aresta, como um algoritmo de otimização e uma generalização do algoritmo de inversão de aresta, foi primeiramente apresentado por Edelsbrunner, Tan e Waupotitsch [34]. Estes, por sua vez, mostraram que a inserção de aresta obtém uma triangulação ótima que o algoritmo de inversão de aresta não é capaz: uma triangulação minimizando o máximo ângulo. Tal demonstração é razoavelmente extensa e envolve conceitos que não são de interesse do presente trabalho, e pode ser encontrada, juntamente com maiores informações sobre este método, em Bern, Edelsbrunner, Eppstein, Mitchell e Tan [17].

3.1.1.3. Divide-and-conquer

O algoritmo *divide-and-conquer*, proposto por Lee e Schachter [64], diferentemente do algoritmo de inversão de aresta, é um algoritmo de geração da triangulação a partir da geometria fornecida, e não apenas um algoritmo de otimização. Este, por sua vez, baseia-se no conceito de subdividir recursivamente o domínio e aplicar a geração em cada parte, para então reuni-las, obtendo assim a triangulação final.

Para obtermos o algoritmo *divide-and-conquer*, a operação fundamental a ser implementada é a união de duas triangulações. Esta operação, no entanto, depende do tipo de

triangulação envolvida, já que os novos triângulos que serão gerados na interface devem respeitar o tipo de triangulação em questão.

Desta forma, devemos focar a atenção na operação de união de duas triangulações de Delaunay disjuntas em uma única triangulação resultante, também de Delaunay. Obtido tal algoritmo, basta operarmos, primeiramente dividindo o domínio em duas partes, através de uma linha vertical, de forma que cada uma das duas partes fique com *n/2* pontos. Depois, executando a mesma operação sucessivamente sobre cada parte, até que cada sub-parte tenha apenas três pontos. A triangulação destas partes é, então, trivial (há apenas uma combinação de triângulos, e esta é a de Delaunay), e com isto basta unir as diversas partes até obter a triangulação completa. A Fig. 26 apresenta uma ilustração com uma etapa intermediária do processo de união de duas triangulação de Delaunay disjuntas. Os triângulos tracejados das bordas são os triângulos que formam as fronteiras externas do domínio, enquanto que os tracejados internos são os triângulos criados em função da união. Os triângulos destacados são os triângulos que estão envolvidos na atual etapa de união. O processo de união de duas triangulações do método divide-and-conquer será explicado em detalhes no Cap. 4.

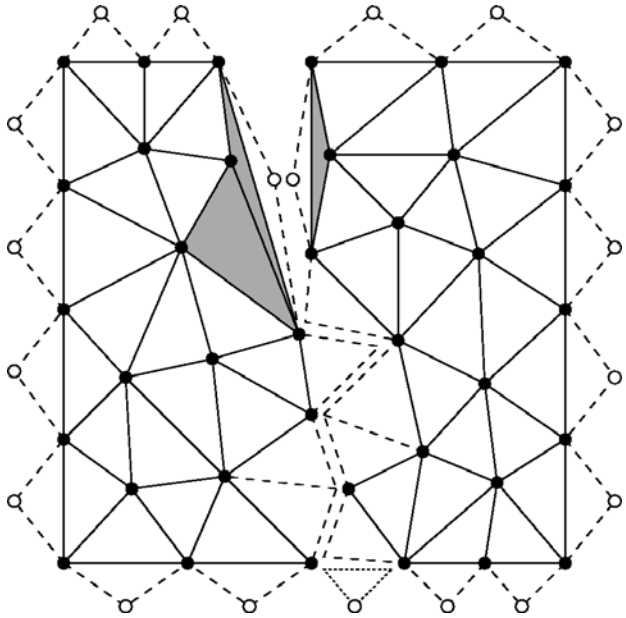


Fig. 26 - Etapa intermediária da junção de duas triangulações disjuntas

Efetuando uma análise do tempo gasto para se unir duas triangulações disjuntas, chegamos a ordem de *O(n)*, onde *n* é o número de arestas na interface entre as duas triangulações. Isto é fácil de ser observado, já que a união das duas partes irá influenciar, necessariamente, apenas a primeira camada de triângulos da interface das duas triangulações.

Para obtermos as arestas corretas na conexão, basta aplicarmos a regra de Delaunay para a aresta, que para este caso, consiste em encontrar os pares de pontos mais próximos, cada ponto em uma das triangulações [97]. Como o número de divisões executadas no algoritmo *divide-and-conquer* é da ordem de *O(logn)*, onde *n* é o número de pontos da triangulação, obtemos um tempo total para geração de uma triangulação de Delaunay com o algoritmo de *divide-and-conquer* da ordem de *O(nlogn)*.

Podemos aumentar a robustez deste algoritmo através da utilização de cortes verticais e horizontais alternados. Utilizando apenas cortes verticais, as sub-partes passam a ser cada vez mais esbeltas, contendo, muitas vezes, pontos bastante próximos na direção x, mas ao mesmo tempo bastante distantes na direção y, resultando em uma distância euclidiana grande. Como sabemos que a triangulação de Delaunay contém as menores arestas possíveis na triangulação, muito provavelmente os triângulos iniciais de cada sub-parte não serão os triângulos finais da triangulação. A utilização de cortes alternados faz com que a distância euclidiana dos pontos em cada sub-parte diminua, reduzindo os erros de truncamento efetuados pelas operações aritméticas de ponto flutuante e, com isto, aumentando a robustez do algoritmo. Além disto, aumenta a probabilidade dos triângulos de cada sub-parte serem os triângulos finais da triangulação. Com tal combinação, uma triangulação com mais de um milhão de pontos pode ser facilmente computada em menos de um minuto em uma estação de trabalho de médio porte (SUN SparcStation Ultra 60 300Mhz). A utilização de cortes alternados é uma modificação do método *divide and conquer*, implementada neste trabalho.

A extensão do método *divide-and-conquer* para domínios do tipo GPSR é bastante simples. Para isto, basta considerarmos dois aspectos: primeiramente, os segmentos de retas definidos no domínio fornecido devem estar sempre presentes em qualquer triangulação das sub-partes; ao mesmo tempo, precisamos considerar a regra da visibilidade (ver seção 2.3.4) no momento de calcularmos os pares de pontos mais próximos, obtendo as arestas de Delaunay responsáveis pela união das duas triangulações.

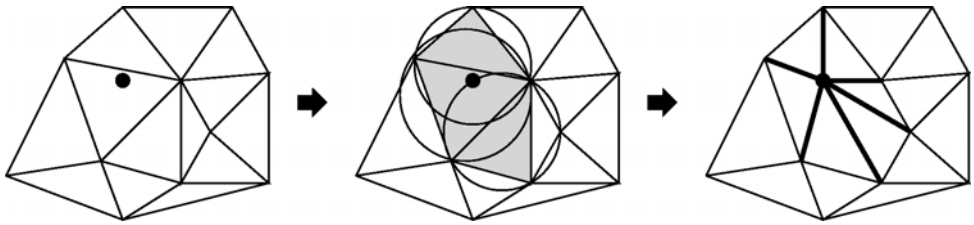
3.1.2. Métodos Incrementais

O grupo dos algoritmos incrementais pode ser dividido em dois sub-grupos: baseados e não-baseados em inversão de aresta. Os algoritmos não-baseados em inversão de aresta têm como aspecto positivo, em relação aos outros, a possibilidade de extensão para dimensões

arbitrárias. O primeiro método não-baseado em inversão de aresta, foi introduzido por Bowyer [18] e Watson [111], enquanto que o baseado em inversão de aresta foi introduzido por Lawson [69]. Nas seções seguintes, tais algoritmos serão apresentados com maiores detalhes.

3.1.2.1. Algoritmo não-baseado em inversão de aresta

O algoritmo de Bowyer [18] e Watson [111] é extremamente simples. Quando um novo vértice é adicionado (Fig. 27(a)), os triângulos nos quais seu circuncírculo contém tal vértice, perdem a propriedade de Delaunay, e com isto, devem ser corrigidos (Fig. 27(b)). Esta correção consiste em eliminar tais triângulos e conectar o novo vértice com os vértices do polígono remanescente através de novas arestas (Fig. 27(c)). Este polígono, formado pela união de todos os polígonos removidos, é denominado de polígono de inserção.



(a) (b) (c)

Fig. 27 - Etapas do algoritmo de Bowyer/Watson

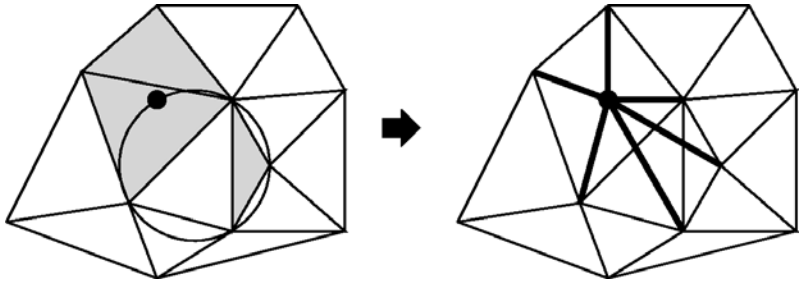
No entanto, diversos detalhes devem ser observados: depois de criada a lista, não podemos eliminar os triângulos através da eliminação de suas três arestas, pois estas arestas também são parte de triângulos vizinhos, e estaríamos, então, eliminando tais triângulos também. A eliminação dos triângulos, na verdade, consiste em um processo no qual se elabora uma lista dos triângulos, cujos circuncírculos contêm tal vértice, e depois eliminam-se todas as arestas comuns de dois triângulos que estejam nesta lista. Com isto, a eliminação destas arestas transforma os diversos triângulos no denominado polígono de inserção.

Uma primeira abordagem, natural, mas extremamente demorada, para a montagem da lista de triângulos a serem removidos, é visitar todos os triângulos da triangulação, checando se o novo vértice é contido ou não pelo seu circuncírculo. Uma outra abordagem, baseada em uma busca topológica recursiva através da triangulação, e que reduz o processo de ordem *O(n)* para ordem *O(1)*, é a montagem desta lista de forma recursiva, iniciando-se pelo triângulo que contém o novo vértice. O triângulo que contém o novo vértice, por sua vez,

necessariamente faz parte da lista, pois seu circuncírculo contém tal vértice. Encontrado tal triângulo, este é incluído na lista (é o primeiro), e seus três vizinhos são também testados. Caso um dos três vizinhos testados seja incluído na lista, os outros dois vizinhos (apenas dois, e não três, pois o terceiro vizinho necessariamente já foi inserido na lista). Isto cria um processo recursivo em árvore, onde, para cada triângulo inserido na lista, testam-se outros dois vizinhos. Quando nenhum dos vizinhos testados é inserido, tal ramo da árvore encerra sua recursividade.

A abordagem anterior transforma o processo de caráter global em local, onde o tempo de computação para se inserir um novo vértice na triangulação independe do número de vértices da triangulação, a não ser pelo processo de localização do triângulo que contém o novo vértice. Este, por sinal, é o gargalo que torna os algoritmos incrementais pouco atrativos para aplicações de geração completa de malha.

O atrativo principal do algoritmo de Bowyer/Watson, por sua vez, é a extensão direta para n-dimensões. Todos os processos anteriormente descritos para duas dimensões podem ser facilmente reescritos para n-dimensões: localização de tetraedro, remoção de tetraedro, obtenção do poliedro de inserção, conexão do novo vértice com os vértices e arestas do poliedro de inserção através de arestas, faces e etc. Onde anteriormente considerava-se um círculo, passa-se à considerar seus semelhantes em maiores dimensões, como a esfera em 3D, ou uma hiper-esfera em 4D. A seqüência de operações definidas pelo algoritmo não muda, independentemente da dimensão de geração da malha de Delaunay.



(a) (b)

Fig. 28 - Sensibilidade do algoritmo de Bowyer/Watson

Por outro lado, o algoritmo de Bowyer/Watson, em sua forma primitiva, utilizando aritmética de ponto flutuante, é bastante sensível a erros de arredondamento. A obtenção de triangulações com conjunto de pontos cocirculares (Fig. 28(a)) pode levar a malhas

inconsistentes, impossibilitando a continuidade do processo (Fig. 28(b)). Este problema, por sua vez, pode ser bastante reduzido em duas dimensões, através da utilização do algoritmo de Lawson [69], apresentado em seguida.

3.1.2.2. Algoritmo baseado em inversão de aresta

O método de Lawson [69] é bastante parecido com o método de Bowyer/Watson, no entanto, difere deste pelo fato de que nenhuma lista recursiva de triângulos de Delaunay inválidos é montada. Um processo recursivo, por sua vez, baseado em inversão de aresta, é executado de forma a serem mantidas as propriedades de Delaunay na nova triangulação.

O processo inicia-se com a localização do triângulo *t* (em destaque na Fig. 29(a)) que contém o novo vértice *v*. Após isto, *v* é adicionado à triangulação, juntamente com outras três arestas que ligam *v* com os três vertices do triângulo *t* (Fig. 29(b)). Com isto, a triangulação remanescente não garante a manutenção das propriedades de Delaunay. Para nos assegurarmos disto, utiliza-se o método de Lawson de inversão de aresta.

O processo de garantia da propriedade de Delaunay é iniciado através da verificação das três arestas do triângulo que originalmente continha o novo vértice. Para cada aresta testada e invertida, outras duas arestas devem ser também testadas, criando uma árvore recursiva bastante parecida com a árvore de triângulos inválidos do método de Bowyer/Watson (Fig. 29(c) e Fig. 29(d)). Quando mais nenhuma aresta testada é revertida, a propriedade de Delaunay é assegurada para a nova triangulação (Fig. 29(e)).

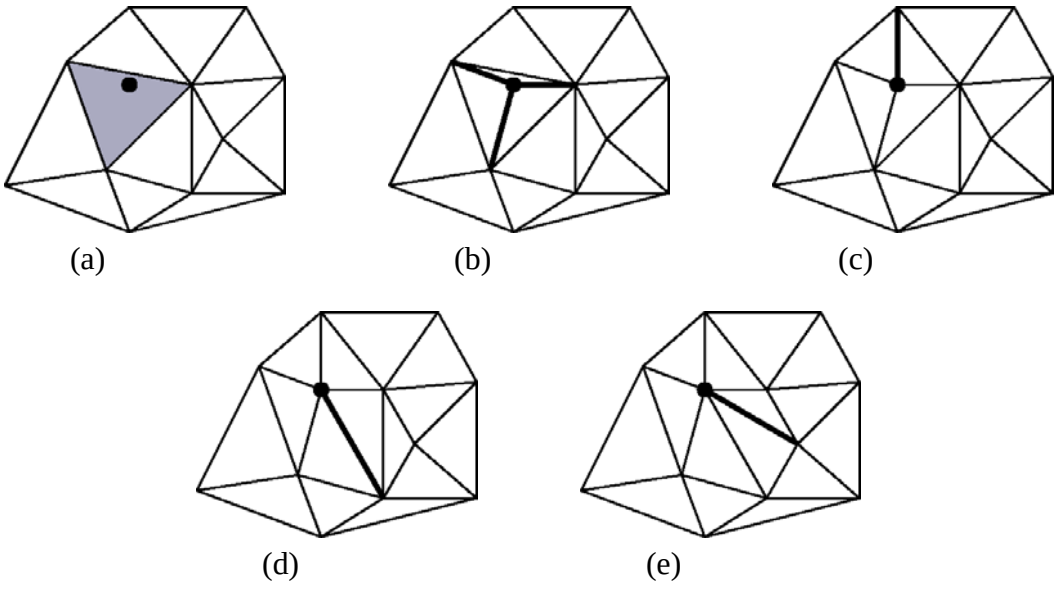


Fig. 29 - Etapas do algoritmo de Lawson [69]

Desconsiderando-se os erros de arredondamento, o método de Lawson obtém exatamente o mesmo resultado que o método de Bowyer/Watson, assim como o esperado de qualquer outro método de inserção de vértice em uma triangulação de Delaunay. No entanto, devido à manutenção da consistência topológica da triangulação ao longo de todo o processo de inserção de vértice de Lawson, tal método é bem mais robusto, quando utilizando aritmética de ponto flutuante, à geração de triangulações de Delaunay degeneradas ou com vértices cocirculares. Outra característica interessante, que deve ser levada em consideração, é que o algoritmo de Lawson é razoavelmente mais fácil de ser implementado que o algoritmo de Bowyer/Watson.

Um aspecto negativo, já mencionado anteriormente, do método de Lawson é a sua não extensibilidade direta para n-dimensões. Isto se deve ao fato de que o algoritmo de inversão de aresta em dimensões maiores que dois é bastante complexo.

3.1.3. O Problema da adição de vértices fora do envelope convexo

Nos dois algoritmos analisados anteriormente, está implícita a hipótese de que para o vértice que está sendo inserido sempre existe um triângulo, pertencente à triangulação, que contém tal vértice. No entanto, quando se utilizam algoritmos incrementais para a geração completa de malha, isto pode não ocorrer, ou seja, o vértice adicionado pode ser localizar fora do envelope convexo da atual triangulação.

Uma saída para tal problema é a criação de arestas que conectem tal vértice a todos os vértices visíveis do envelope convexo, com a posterior verificação da propriedade de Delaunay através do procedimento de inversão de aresta. Esta solução, apesar de ser a solução mais correta, é a mais difícil de ser implementada. Uma outra solução para tal problema, é a utilização de um triângulo (em 2D) ou tetraedro (em 3D) como uma fronteira envolvente para todos os vértices da triangulação (Fig. 30). Com isso, o processo de triangulação inicia com tal fronteira e, após a adição de todos os vértices, a mesma é removida. Com isto, não é necessário manipular casos específicos.

Esta abordagem, no entanto, não é interessante, e costuma não ser recomendada, devido à necessidade de escolher uma fronteira que seja grande o suficiente para englobar todos os pontos, e que não perca nenhum triângulo ao longo do processo de adição de vértices. Outro aspecto negativo na utilização desta fronteira são os problemas de arredondamento e de

aritmética de ponto flutuante que surgem na manipulação com os números de magnitude elevada de tal fronteira. Estes erros podem se propagar para dentro da triangulação e resultar em comportamentos catastróficos.

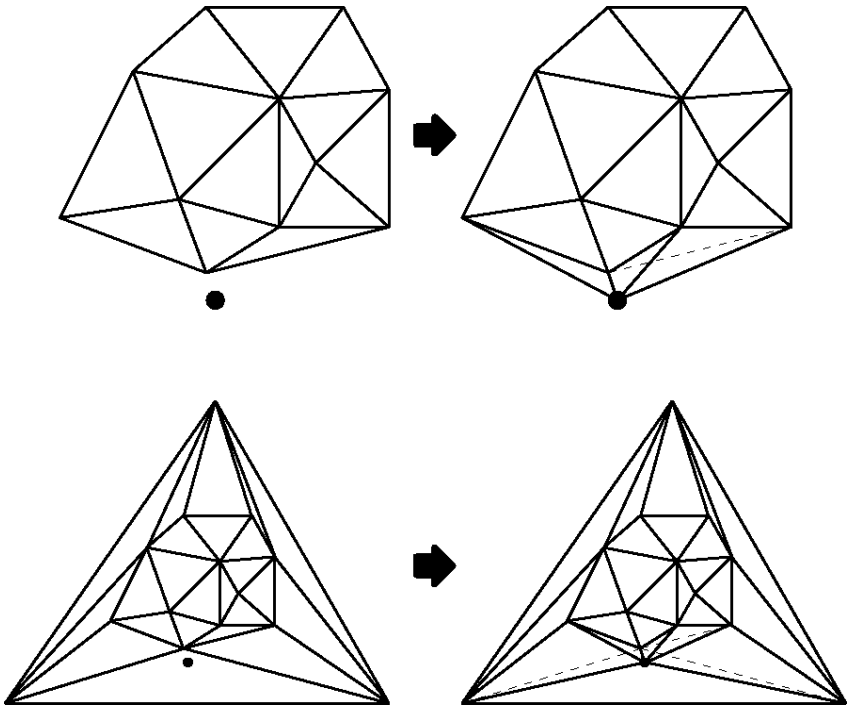


Fig. 30 - Adição de vértices fora do envelope convexo

3.1.4. Localização de pontos

Nos métodos incrementais apresentados anteriormente, a operação de inversão de aresta não é a única a consumir recursos computacionais. Na verdade, a operação de encontrar-se o triângulo que contém um determinado ponto (também denominado de localização de pontos) dentro de uma triangulação é a maior responsável pelo tempo de CPU. Felizmente, muitos dos métodos de geração de triangulações de Delaunay inserem pontos em locais onde já ocorreram manipulações geométricas ou que foram previamente identificados como precisando de refino, tornando a localização do ponto praticamente conhecida. Mesmo assim, em um método de triangulação de Delaunay de propósito geral, a localização de pontos pode representar uma parcela considerável dos recursos computacionais consumidos.

Um método de localização de pontos baseado na manutenção de um *grafo de conflito* é o de Clarkson e Shor [26]que, quando aplicado para duas dimensões, reduz o tempo de localização de um ponto dentro de uma triangulação de ordem *O(n)* para ordem *O(logn)*.

Apesar de tal ordem só ser atingida se todos os pontos envolvidos no processo forem conhecidos desde o início, Guibas, Knuth e Sharir [52] apresentam uma adaptação (com um desempenho razoável) de tal método para aplicações, onde os pontos são definidos apenas ao longo do processo.

Para uma triangulação com *n* vértices, os algoritmos incrementais, onde a inserção de um ponto sem considerar o procedimento de localização do mesmo é executada em um tempo de ordem *O(1)*, que é o caso dos algoritmos apresentados anteriormente, o presente método de localização de ponto permite que tais algoritmos sejam executados em tempo de ordem *O(nlogn)*. Sem tal método de localização de ponto, tais algoritmos consomem um tempo da ordem de $O(n^2)$.

Para apresentar o método, vamos, primeiramente, definir *grafo de conflito*. Consideremos um estágio intermediário de geração de uma triangulação de Delaunay utilizando um algoritmo incremental, onde o número total de vértices da triangulação é *n* e o número atual de vértices inseridos é *p*. O número de vértices não inseridos, por sua vez, é *n-p*. Seja *t* um triângulo da atual triangulação D_p, e *w* um dos *n-p* vértices ainda não inseridos. Caso w esteja dentro do circuncírculo de *t*, então *t* e *w* estão em conflito. Devido a isto, é claro que *t* não pode ser um triângulo da triangulação de Delaunay completa D_n.

Um grafo de conflito, então, é um grafo bipartido, onde cada nó representa um triângulo ou vértice não inserido na triangulação, e cada aresta representa um conflito entre um vértice e um triângulo (Fig. 31). Cada vértice inserido altera o grafo de conflito. Quando um vértice é inserido, os triângulos criados, referentes a tal vértice, podem conflitar com vértices não inseridos, adicionando novos conflitos ao grafo. Por outro lado, a inserção de um novo vértice na triangulação causa a remoção dos conflitos associados a tal vértice.

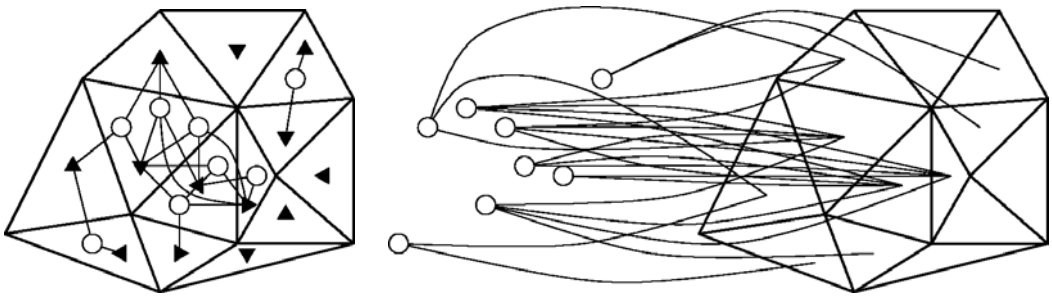


Fig. 31 - Esquemas do grafo de conflito [26]

A implementação da manipulação com a estrutura de dados do grafo de conflito é algo não trivial, além de que a quantidade de informações armazenadas pode se tornar bastante grande, dependendo do número de vértices e da forma dos elementos. O número esperado de conflitos de um ponto com triângulos é quatro, mesmo assim, para arranjos não aleatórios de pontos, este número pode crescer consideravelmente.

Devido a este fato, Shewchuk [99] propôs uma variante do grafo de conflito: o *grafo de conflito simplificado*. Neste grafo simplificado é mantido apenas um conflito por vértice não inserido na triangulação. Os demais conflitos, apesar de ainda serem considerados conflitos, pois tal conceito não foi modificado, são omitidos. Por sua vez, o conflito de um vértice, mantido no grafo, é aquele que relaciona tal vértice com o triângulo, da atual triangulação, que o contém. Este grafo, por sua vez, poderia também ser chamado de grafo de contenção. A Fig. 32 apresenta uma ilustração do grafo de conflito simplificado.

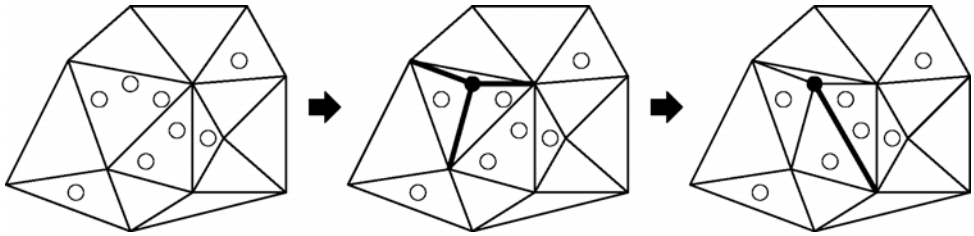


Fig. 32 - Esquema do grafo de conflito simplificado [99]

Como exercício, consideremos um exemplo simples hipotético, onde iniciamos a triangulação com um único triângulo que envolve todos os vértices da triangulação. No inicio do processo, todos os vértices possuem conflitos indicando tal triângulo. Conforme os vértices vão sendo adicionados, novos triângulos vão sendo criados, e os conflitos vão sendo ajustados de forma a manter o grafo correto. Se um vértice recai sobre uma aresta da triangulação, um dos dois triângulos é escolhido arbitrariamente. Para cada vértice não inserido, a estrutura de dados mantém um ponteiro para o triângulo que contém tal vértice. Da mesma forma, para cada triângulo da triangulação, a estrutura de dados mantém uma lista indicando os vértices não inseridos que são contidos por tal triângulo. Com isto, a localização do triângulo que contém qualquer vértice não inserido na triangulação é uma operação direta, consumindo um tempo de computação da ordem de *O(1)*.

O resultado acima, no entanto, conflita com informações fornecidas no começo deste tópico, onde foi afirmado que o grafo de conflito reduz o tempo de computação para ordem

de *O(logn)*. Isto porque na análise acima, não foi considerado o tempo de computação para se efetuar a atualização da lista de conflito. Quando um vértice *v* é inserido na triangulação, os conflitos dos pontos associados ao triângulo *t*, que contém *v*, tem que ser redistribuídos entre os quatro novos triângulos formados pela subdivisão de *t*. Ao mesmo tempo, quando uma operação de inversão de aresta é executada, o grafo de conflito deve ser atualizado. Ou seja, para cada operação de alteração da triangulação, uma atualização no grafo de conflito, refletindo tal modificação, deve ser efetuada, de forma a manter o grafo sempre correto. Com isto, o custo fundamental na utilização de grafo de conflito, para localização de vértices, está na manutenção deste grafo. A demonstração de que o custo de manutenção desta lista leva à um tempo médio de CPU, por ponto, da ordem de *O(logn)*, é bastante complicada, e baseia-se em diversos teoremas e em análise reversa. Análise reversa é uma ferramenta extremamente poderosa para se efetuar estimativas dos recursos computacionais consumidos por algoritmos, mas que não será abordado no presente texto.

Conforme mencionado anteriormente, para métodos de triangulação incrementais, onde não se conhece todos os vértices de antemão, Guibas, Knuth e Sharir [51], sugeriram uma abordagem bastante interessante denominada *history dag* (*Direct Acyclic Graph*). Esta abordagem, utilizada em conjunto com o grafo de conflito, consiste no armazenamento de todas as operações de modificação geométrica efetuadas sobre a triangulação. Com isto, dado um vértice anteriormente não conhecido, é possível rastrear-se, desde o princípio do processo de geração, o triângulo que contém tal vértice. A operação de rastreamento, por sua vez, confirmando a análise do tempo de CPU gasto pelo grafo de conflito, consome um tempo de CPU da ordem de *O(logn)*. O uso da *history dag*, no entanto, pode tornar-se inviável para diagramas com um número elevado de vértices, devido à memória utilizada pelo mesmo para o armazenamento do histórico completo das operações geométricas executadas.

3.2. Outros métodos de triangulação ótima

Esta secção apresenta, de forma também breve, diversos métodos de geração de malhas bidimensionais não-estruturadas de triângulos. O texto não pretende ser uma descrição detalhada dos métodos, mas apenas apresentar os mesmos como uma revisão bibliográfica relacionada a triangulação de Delaunay, foco principal deste trabalho.

3.2.1. Triangulação qualquer

De todos os métodos apresentados nesta seção, este é o único que não considera otimização. Para este desenvolvimento, primeiramente será definido o conceito de diagonal: uma diagonal de um polígono simples é um segmento de reta que liga quaisquer dois vértices deste polígono, que se encontra no interior deste, e que, necessariamente, intersecciona com o mesmo apenas nos pontos extremos (ver Fig. 33).

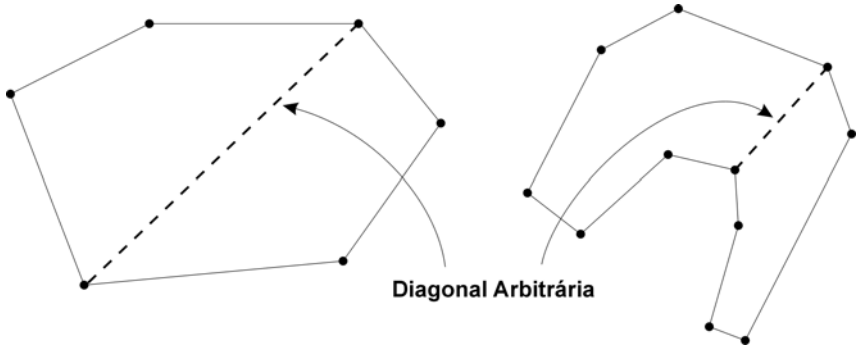


Fig. 33 - Diagonal de um polígono qualquer

De acordo com Bern e Eppstein [11], todo polígono com mais de três lados possui ao menos uma diagonal. Agora, considerando-se um polígono qualquer, como é sempre possível obter-se uma diagonal para o mesmo, esta divide tal polígono em outros dois polígonos quaisquer. Aplicando sucessivamente este processo, até que cada polígono resultante tenha apenas 3 lados, a triangulação é obtida, provando que sempre é possível obter a triangulação de um polígono simples.

Como o tempo de computação para se encontrar a diagonal de um polígono é de ordem *O(n)*, o algoritmo descrito anteriormente triangula um polígono qualquer com um tempo da ordem de $O(n^2)$. Analisando-se outros tipos de domínios, veremos que o algoritmo anterior também vale para um conjunto qualquer de pontos, para polígonos não-simples (multiplamente-conexos e finalmente para GPSR's.

3.2.2. Decomposição de polígono

O método de decomposição de polígono é bastante parecido com o de triangulação qualquer, sendo considerado uma extensão do mesmo. A diferença principal está na utilização de pontos de Steiner, objetivando a otimização de critérios de qualidade escolhidos pelo

usuário. Outra diferença, também, é quanto à escolha das linhas que dividem inicialmente o domínio (normalmente um GPSR), que procuram levar em consideração características intrínsecas do domínio e não apenas linhas quaisquer (como na triangulação qualquer).

O algoritmo de decomposição de polígono foi inicialmente proposto por Joe e Simpson [59]. Primeiramente, o domínio (Fig. 34(a)) é dividido em polígonos simplesmente conexos, também denominados de polígonos principais, definindo regiões no domínio (Fig. 34(b)). Em seguida, são efetuadas duas operações buscando a otimização de malha (Fig. 34(c)): a primeira consiste na divisão destes polígonos principais em outros polígonos, utilizando como critério a similaridade do tamanho de aresta dentro de cada novo polígono. A segunda consiste na secção deste polígono através de segmentos de retas, buscando eliminar pequenos ângulos. Finalmente, os polígonos principais são discretizados utilizando-se triângulos de tamanhos aproximadamente iguais (Fig. 34(d)), sempre lembrando que os pontos de Steiner devem obedecer os segmentos de reta definidos na etapa de otimização.

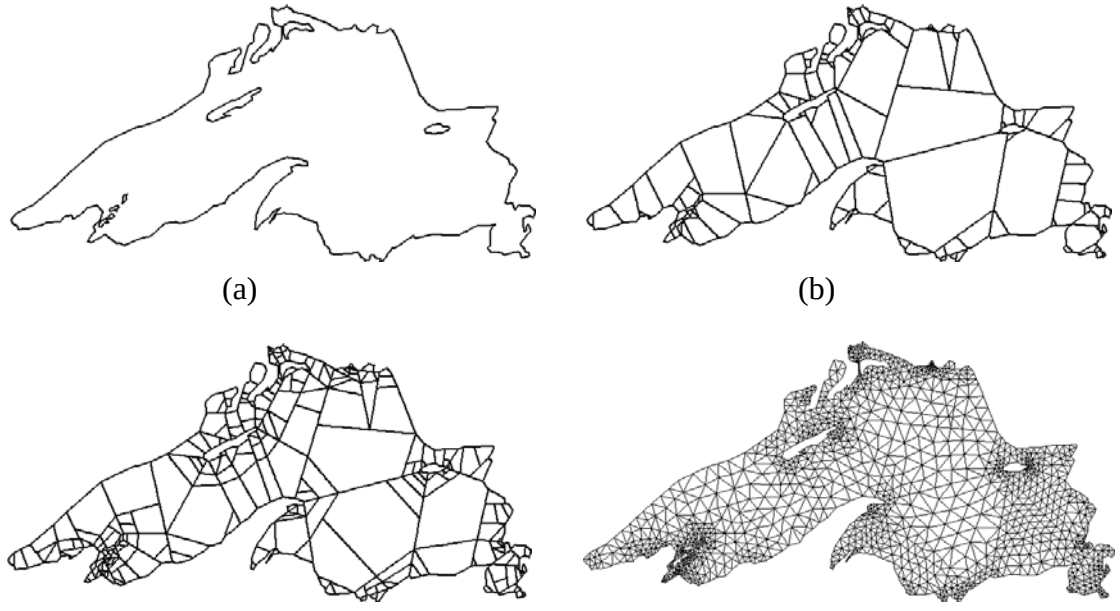


Fig. 34 - Triangulação gerada por decomposição de polígono [11]

As operações de otimização de malha apenas definem restrições geométricas para a adição dos pontos de Steiner e os tamanhos característicos dos triângulos para cada região a ser discretizada. Neste método, é disponibilizado dois parâmetros para o controle da malha: um número de triângulos para a discretização (normalmente excedidos por uma pequena

quantidade) e um coeficiente de suavização, que controla quantas vezes, maior ou menor, os triângulos podem ser, relativo a seus vizinhos.

3.2.3. Avanço de frentes

Dos diversos métodos apresentados até agora, um bastante utilizado em problemas de escoamento de fluidos [55] é o de avanço de frentes [45][66][69][80]. Neste método, a primeira etapa a ser executada é a divisão do domínio em partes simplesmente conexas. No avanço de frentes, no entanto, diferentemente de outros métodos, esta divisão é feita pelo próprio usuário, baseada no problema físico, de forma que as fronteiras dos subdomínios fiquem mais alinhadas o possível com o escoamento e/ou que estas coincidam com as fronteiras sólidas.

Definidos os subdomínios, camadas de pontos de Steiner são adicionadas, uma a uma, a partir das fronteiras em direção ao centro. A triangulação destes pontos pode ser feita simultaneamente com a adição destes ou em uma etapa posterior, de forma a obtermos o particionamento do domínio. Quando a triangulação é efetuada em etapas posteriores, qualquer método de geração de malha pode ser utilizado, como por exemplo, a triangulação de Delaunay. Com esta abordagem, o método de avanço de frente gera elementos alinhados com o escoamento, facilitando a captação de diversos fenômenos e, usualmente, levando a melhores soluções numéricas. Outros problemas físicos podem também influenciar a geração de malhas pelo método de avanço de frentes.

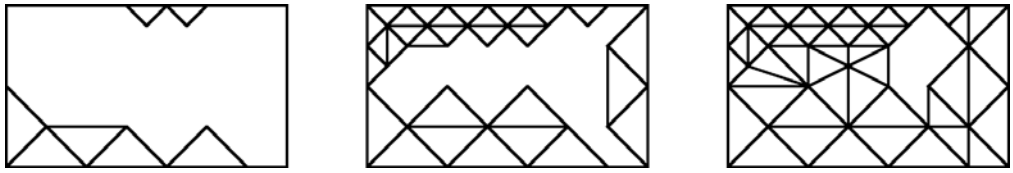


Fig. 35 - Encontro de frentes de avanço

Este método, apesar de gerar elementos de excelente qualidade próximo as fronteiras, enfrenta sérios problemas quando no encontro destas. Caso duas frentes com grande diferença de tamanhos de elementos se encontrem, muito dificilmente se conseguirá gerar elementos de qualidade aceitável em tal região (ver Fig. 35) Por isso, este método é muito sensível à escolha das frentes e de seus respectivos tamanhos de elementos. Diversos autores

consideram este problema suficientemente limitante para que seja possível desenvolver algoritmos de avanço de frente eficientes para geração de malhas para problemas práticos.

A extensão do método de avanço de frente para domínios tridimensionais é considerada uma tarefa complexa devido ao fato de que a simples geração da malha superficial já representa um desafio razoável.

Variantes do método de avanço de frentes, que não precisam necessariamente conformar-se às fronteiras do domínio, foram sugeridas por diversos autores. Uma delas, por exemplo, é atribuída a Lo [70], que desenvolveu um gerador de malhas triangulares que posiciona pontos ao longo das linhas de fluxo do escoamento. Uma solução inicial do escoamento é necessária para podermos fazer uma pré-localização dos pontos. Devido a este fato, este método tem um perfil teórico voltado mais para a adaptatividade de malhas do que necessariamente para um esquema de geração.

Outra variante, sugerida por Mavriplis [80], para a geração de malhas triangulares para problemas de escoamento de fluidos com alto número de Reynolds, também baseia-se no conceito de localizar pontos sobre linhas de fluxo. Este método identifica linhas de maior cisalhamento do escoamento, e insere pontos, primeiramente, nestes locais. Posteriormente, malhas estruturadas são utilizadas para gerar os elementos nas regiões entre estas linhas, sendo que uma triangulação de Delaunay é utilizada para gerar a malha no encaixe entre as diversas malhas estruturadas, obtendo localmente elementos longos e finos, direcionados com o escoamento.

Além do uso de malhas estruturadas, para a localização de pontos de Steiner nas regiões de preenchimento, em conjunto com o avanço de frente, Mavriplis também utilizou a triangulação de Delaunay. A triangulação de Delaunay é uma estrutura interessante, pois a mesma fornece, de forma rápida e precisa, a distância existente entre as diversas possíveis frentes. De posse destas informações é possível interromper o processo de avanço das frentes antes destas se encontrarem, e combinar um método alternativo para o preenchimento do espaço resultante. Um exemplo de uma malha gerada com o método de avanço de frente é apresentado na Fig. 36, obtida com o gerador de Barth e Jesperson [11]. O problema de colisão das frentes, em geometrias deste tipo, é minimizado devido à geração da malha ser feita para um problema de escoamento externo.

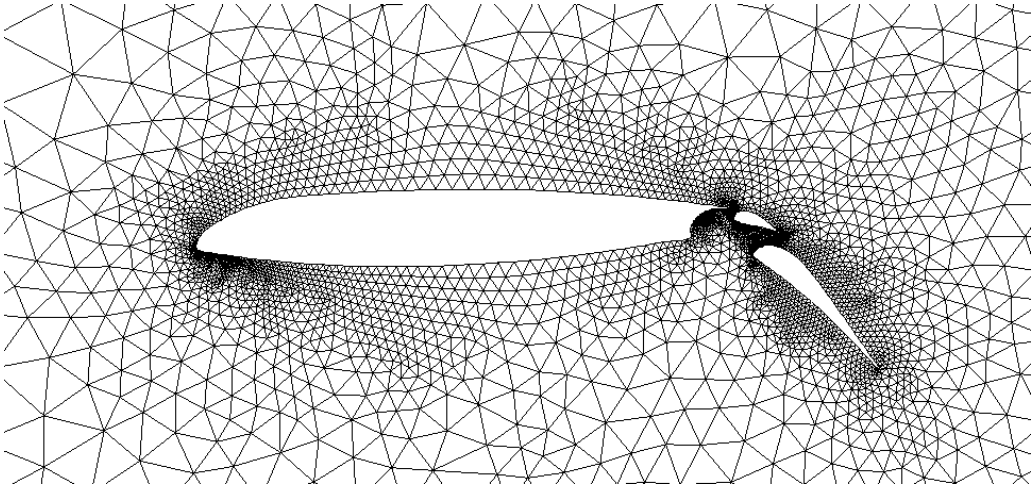


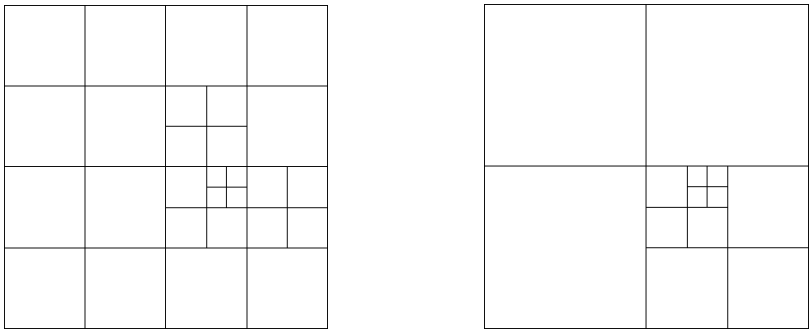
Fig. 36 - Malha gerada por frente de avanço [11]

3.2.4. *Quadtree*

Dentre os diversos métodos de triangulação, o primeiro método prático desenvolvido para a triangulação de domínios bidimensionais, foi o método *quadtree*, apresentado por Bern, Eppstein e Gilbert [14].

Um *quadtree* [41][93][95] consiste em uma partição recursiva de um domínio bidimensional em quadriláteros alinhados com os eixos ortogonais principais. O primeiro quadrilátero, também chamado de quadrilátero raiz, contém o domínio por completo. Um quadrilátero pode ser subdividido em quatro quadriláteros filhos, seccionando-se o mesmo através dos eixos principais alinhados em seu centro. A coleção de quadriláteros forma uma árvore, com quadriláteros cada vez menores em níveis mais baixos da árvore, denominada de quadtree (árvore de quadriláteros). Um *octree* é a generalização de um *quadtree* para três dimensões, onde cada cubo pode ser dividido em oito cubos menores. Na Fig. 37(a) pode ser visto um exemplo de *quadtree*.

À definição primitiva de *quadtree*, pode-se adicionar um critério de balanço, onde cada quadrilátero deve ser, no máximo, duas vezes maior que seus respectivos vizinhos. Isto é equivalente a dizer que a aresta de um quadrilátero nunca se sobrepõe a mais de duas arestas de um quadrilátero vizinho. Um *quadtree* considerando o critério de balanço pode ser visto na Fig. 37(a).



(a) (b)

Fig. 37 - Estrutura de uma *quadtree* com (a) e sem (b) o critério de balanço

Para se obter o método de triangulação *quadtree*, basta utilizar a definição de *quadtree* para um conjunto de pontos, aplicando as sucessivas subdivisões até que cada ponto fornecido esteja devidamente separado de seus pontos vizinhos. Tradicionalmente, isto significa dizer que cada ponto está localizado no quadrilátero central de um arranjo de 5x5 quadriláteros, todos de mesmo tamanho, e não contendo mais nenhum ponto, a não ser o próprio ponto central. Diversos outros critérios, como por exemplo, tamanho máximo de elemento, podem ser utilizados como regra para a subdivisão de quadriláteros.

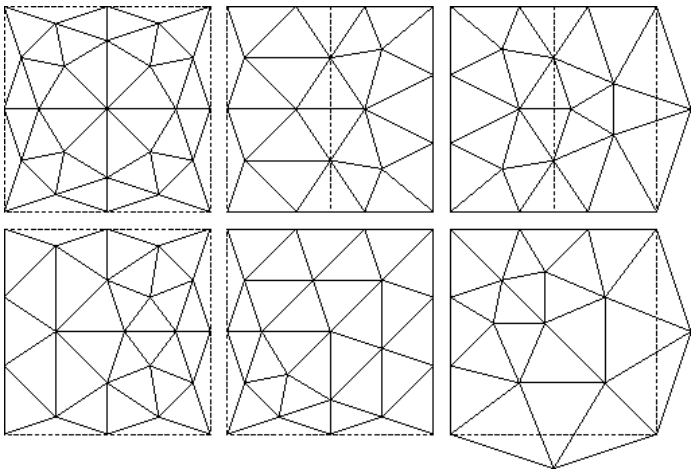


Fig. 38 - Padrões para o preenchimento de uma *quadtree* [14]

A etapa seguinte consiste no preenchimento dos quadriláteros com triângulos. Diversos métodos utilizam padrões de pequenas malhas triangulares para preencher os quadriláteros (ver Fig. 38). No entanto, tais métodos não conservam por completo a geometria, distorcendo sua fronteira. Métodos mais avançados executam uma etapa intermediária de conformação dos vértices da *quadtree* aos pontos da triangulação. Nesta etapa, desloca-se o vértice mais próximo da *quadtree* para o ponto da triangulação. Um vértice da *quadtree* não pode ser

escolhido duas vezes devido às subdivisões feitas anteriormente. Finalmente, cada quadrilátero é dividido em dois triângulos e obtém-se a triangulação final. Um exemplo de uma triangulação quadtree que conserva as fronteiras do domínio fornecido por ser vista na Fig. 39.

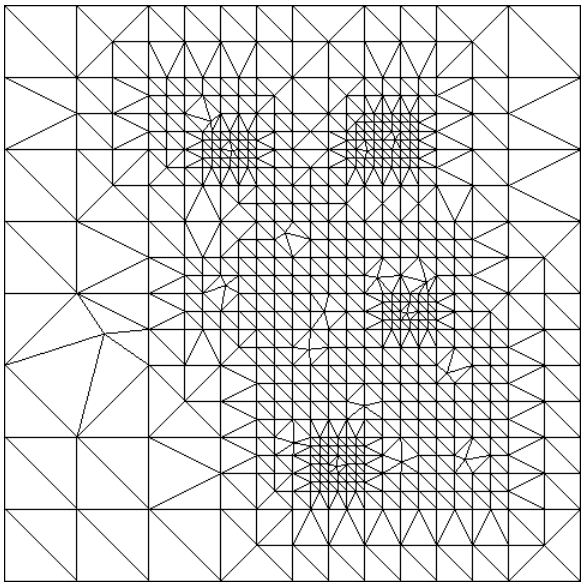


Fig. 39 - Malha gerada pelo método *quadtree* [11]

É fácil demonstrar que, devido ao critério de balanço, para quadriláteros onde não foi efetuado nenhum deslocamento de vértices, sempre teremos ângulos internos maiores que *arctan(1/2)*, o que equivale a aproximadamente 26.5°. Para quadriláteros onde um dos vértices foi deslocado para se conformar com os pontos de entrada, é possível demonstrar-se que, com a adição de uma diagonal, sempre teremos ângulos maiores que 20°. Nunca ocorre, em uma *quadtree*, o deslocamento de dois ou mais vértices de um mesmo quadrilátero. Juntamente com este limite mínimo dos ângulos internos, Bern et al. [14] demonstraram que o número de pontos de Steiner utilizados no método de *quadtree* é considerado pequeno.

Pode-se dizer, então, que o método de *quadtree* é um algoritmo de triangulação que respeita o critério de ângulos pequenos, com um limitador de 20°. Como um triângulo possui três ângulos internos, e todos estes respeitam tal limitador mínimo, temos também um limitador para o máximo ângulo interno, de magnitude 140°. Outras propriedades podem ser obtidas através de análises do método *quadtree*, como a ordem do número de quadriláteros e triângulos utilizados para mapear o domínio.

O algoritmo de *quadtree* apresentado anteriormente aplica-se apenas para a triangulação de um conjunto de pontos. Este, no entanto, pode ser estendido para domínios do tipo GPSR [80]. A idéia fundamental é duplicar-se os quadriláteros que contenham mais de um segmento do domínio fornecido, de forma que seja possível conformar-se a aresta de cada quadrilátero ao segmento do GPSR o qual ele contém. Outra característica interessante é que tal método, devido à característica de multi-escala (cada sub-divisão pode ser considerada como uma escala de discretização) é facilmente adaptado para o uso de aritmética inteira, eliminando os problemas decorrentes do uso de aritmética de ponto flutuante. A Fig. 40 apresenta um exemplo de triangulação de um domínio do tipo GPSR através do método *quadtree*.

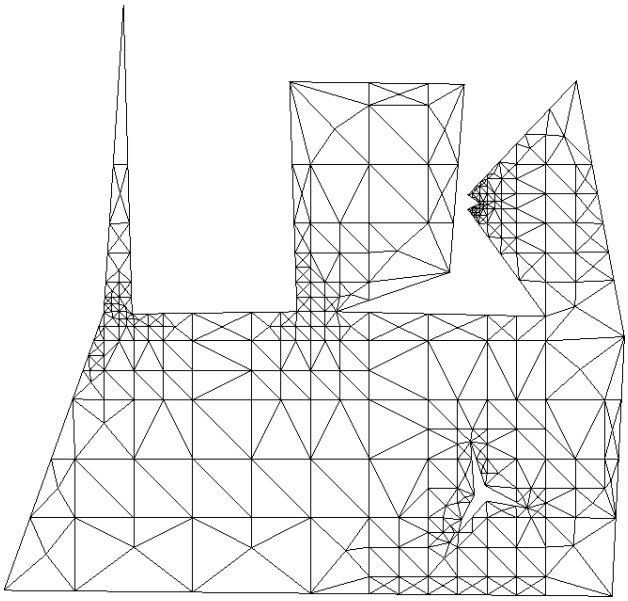


Fig. 40 - Triangulação de um GPSR obtida pelo método *quadtree* [80]

Um dos maiores problemas do método de *quadtree* é a criação de malhas com elementos que contém direções preferenciais [100] (no caso, as direção dos eixos principais). Esta preferência direcional, no entanto, pode ser resolvida com a utilização de métodos de introdução de deslocamentos aleatórios de vértices (com o objetivo de eliminar a direcionalidade das arestas) acoplado à um método de suavização Laplaciana, para tentar garantir uma boa qualidade de malha. Outro problema, que pode ser considerado bastante grave, é a dificuldade de extensão de tal método para domínios tridimensionais. Esta generalização é bastante complexa, e os resultados obtidos através de desenvolvimentos

teóricos não são suficientes para garantir uma boa qualidade de malha para a grande maioria dos casos práticos. O algoritmo *quadtree* é dito um algoritmo baseado em grades.

3.3.Melhoramento de malha

3.3.1. Generalidades

Métodos de melhoramento de malha são processos aplicados sobre uma malha completa, em etapa posterior, ou em conjunto com o processo de geração, e baseiam-se, em essência, no deslocamento dos vértices e troca de arestas, com o objetivo de melhorar a forma e o tamanho dos elementos. A literatura não é unânime ao definir que tipo de operação em uma malha é definida como melhoramento. Na área numérica é mais comum interpretar-se o melhoramento como operações feitas em uma malha sem alterar o número de elementos. Em geometria computacional, como a malha obtida pelo gerador, como resultado dos pontos fornecidos como dados de entrada, é, em geral, grosseira, o refino é um processo interpretado como melhoramento. Na área numérica o refino é mais associado ao próprio processo de geração da malha. Neste trabalho, o refino apenas pela subdivisão dos elementos, o refino com base em parâmetros geométricos, o refino com base na solução(refino adaptativo) e a suavização da malha sem alterar o número de elementos, são todos considerados processos de melhoramento da malha. Estas definições melhor se classificam quando inseridas nos métodos dos grupos morfológicos, topológicos e mistos.

Como já discutido, o tamanho, a forma, a orientação e outras propriedades dos elementos de uma triangulação são de fundamental importância para a geração de uma malha de qualidade para simulação numérica. Diversos tipos de triangulações foram aqui definidas, cada uma levando em consideração diferentes critérios. A triangulação de Delaunay, uma estrutura geométrica que otimiza, simultaneamente, diversos parâmetros, e de grande interesse para a engenharia, recebeu atenção especial por ser o foco do trabalho. No entanto, todos os métodos de triangulação de Delaunay apresentados até agora, obtêm a triangulação utilizando apenas os pontos fornecidos, ou seja, nenhum ponto adicional da triangulação é utilizado. Com isto, caso o número de pontos fornecidos seja pequeno ou o posicionamento destes inadequado, o triangulador estará impossibilitado de garantir determinados critérios de qualidade.

O refino de malha neste contexto significa sempre o uso de pontos adicionais na triangulação, de forma a permitir que o triangulador garanta a obtenção dos critérios de qualidade desejados. Estes pontos adicionais são denominados de pontos de Steiner.

Devido aos desdobramentos e ao impacto que os métodos de refino de malha exercem sobre os trianguladores, um número bastante grande de pesquisadores considera o refino como o passo principal na geração deste tipo de malha [10][85][45].

Um dos primeiros métodos, heurístico, de refino de malha é atribuído à Frey [45]. Neste método, pontos de Steiner são adicionados à fronteira de acordo com uma função espacial que aproxima o tamanho característico local. Após este passo, as arestas da triangulação de Delaunay, para a grande maioria das geometrias, mapeiam por completo as fronteias do domínio. Em seguida, pontos de Steiner são adicionados ao interior do domínio da seguinte forma: procura-se por um triângulo *t* que contenha o seu circuncentro; localiza-se um possível ponto de Steiner *a* entre o baricentro e o circuncentro de *t*; caso *a* não esteja muito perto dos vértices de *t* (onde o critério de estar perto ou não é definido pelo tamanho característico local) adiciona-se *a* à triangulação e reconstrói-se a mesma.

Posteriormente, Shaw [97] desenvolveu um algoritmo de triangulação baseado no refino triangular, onde o usuário fornece uma triangulação inicial bastante grosseira (Fig. 41(a)), e um tamanho característico local associado a cada vértice desta malha. O algoritmo procede da seguinte forma: triângulos maiores que o menor tamanho característico local de seus vértices são divididos em quatro triângulos menores semelhantes, através da adição de um ponto de Steiner no seu baricentro e no ponto médio de suas arestas. Conseqüentemente, os três triângulos vizinhos são divididos em dois triângulos menores, de forma a manter a consistência da triangulação (igual a [6] e [10], com o método de melhoramento da divisão quadrática). Com isto, o tamanho característico local dos vértices adicionados nas arestas passa a ser a média dos tamanhos característicos locais dos vértices de tal aresta, enquanto que o tamanho característico local do vértice central passa a ser a media dos tamanhos característicos locais dos três vértices daquele triângulo. O processo de divisão de triângulos é seguido de ciclos de inversão de aresta de Delaunay e de suavizações Laplacianas ponderadas, para então iniciar-se uma nova divisão. Conforme os triângulos das fronteiras vão sendo refinados, os seus pontos são ajustados à fronteira real da geometria, que pode conter curvas, por exemplo. Os resultados obtidos por Shaw são bastante satisfatórios. A Fig.

41(b) apresenta um exemplo de uma malha gerada com tal triangulador. O aspecto dos elementos finais da triangulação, praticamente, independe da malha inicial fornecida.

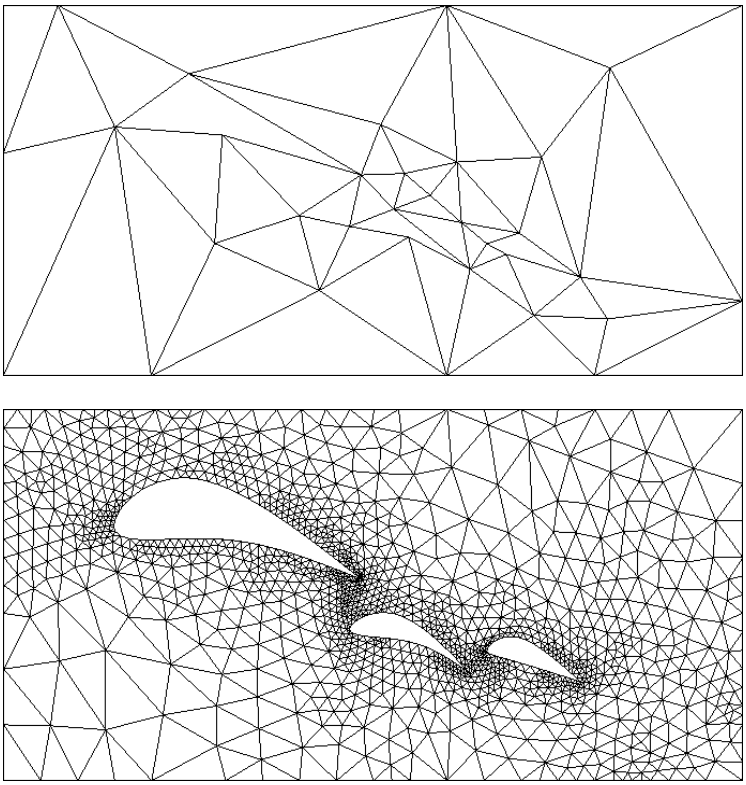


Fig. 41 - Malha gerada com o algoritmo de refino de Shaw [97]

O primeiro algoritmo a garantir o controle do critério de forma dos elementos foi apresentado por Baker, Grosse e Rafferty [7]. Este algoritmo garante que todos os triângulos da malha não terão ângulos internos obtusos (todos os três ângulos serão menores que 90°), ao mesmo tempo que o menor ângulo interno terá ao menos 13° (isto se nenhum ângulo interno da geometria fornecida for menor que 13°, é claro). Estes dois limitadores, em conjunto, garantem que nenhum elemento da malha terá uma relação de aspecto maior que 4.6. A idéia básica é sobrepor uma malha cartesiana regular sobre o polígono, com o espaçamento determinado pelo tamanho característico local. Como o tamanho característico local pode ser pequeno, e este determina a densidade da malha ao longo de todo o domínio, o número de pontos de Steiner utilizados pode ser consideravelmente grande.

O primeiro algoritmo a garantir dois critérios distintos simultaneamente, a forma e o tamanho dos elementos, foi apresentado por Bern, Eppstein e Gilbert [14]. Este método é baseado no método *quadtree* apresentado no Cap. 3, no entanto, os triângulos gerados

respeitam a regra de Delaunay, ou seja, o método de *quadtree* é apenas a base para a localização dos pontos de Steiner da triangulação de Delaunay.

Todas as técnicas citadas são técnicas baseadas em malhas. Uma outra técnica, conceitualmente diferente, para o refino de malha, denominada de refinamento de Delaunay, foi proposta por Chew [23]. Esta técnica é denominada de refinamento de Delaunay pois os critérios da triangulação de Delaunay são mantidos ao longo de todo o processo, e estes são utilizados como guias para a localização de novos pontos. O algoritmo de Chew, além de garantir os ângulos internos dos elementos entre 30° e 120°, produz malhas uniformes, o que significa que os triângulos têm, fundamentalmente, o mesmo tamanho. A triangulação gerada, por sua vez, é considerada ótima quanto ao número de triângulos, apesar de existirem casos onde o número de elementos é maior que o necessário.

3.3.2. Suavização Laplaciana

Uma das técnicas mais tradicionais de melhoramento de malha é a suavização. É uma técnica puramente morfológica, que se baseia na movimentação de vértices para alterar a forma e tamanho dos elementos vizinhos.

Uma das mais conhecidas foi desenvolvida em meados dos anos 60, denominada de suavização Laplaciana [114]., pois sua fórmula de reposicionamento dos vértices pode ser derivada da uma aproximação numérica da equação de Laplace [51].

Na suavização Laplaciana, um vértice no interior da uma malha é movido para o centróide (centro de massa) de seus vizinhos (ver Fig. 42). Caso o centróide encontre-se fora do polígono formado pelos vizinhos, este não deve ser reposicionado. Vértices localizados na fronteira do domínio, obviamente não podem ser deslocados, sendo que a nova localização deve consistir de uma projeção sobre tal fronteira. Alguns pontos específicos não podem ser reposicionados, como, por exemplo, pontos de cantos de fronteiras. Como o reposicionamento dos pontos é feito de forma explícita, para se conseguir resultados razoáveis com este método, deve-se executar uma série de iterações, normalmente cinco ou seis. Uma variante do método de suavização Laplaciana é considerar pesos diferentes para os diversos vizinhos, baseando-se nas áreas dos respectivos elementos, o que é equivalente a aproximação numérica da equação de Poisson com termo fonte variável no espaço.

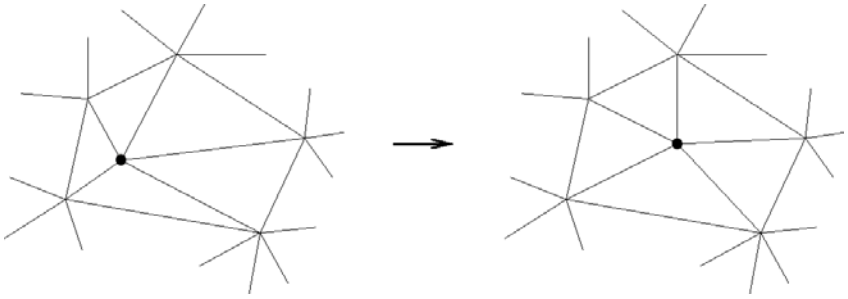


Fig. 42 - Deslocamento de vértice do método de suavização Laplaciana

O reposicionamento pela suavização Laplaciana normalmente melhora o tamanho e a forma dos elementos em duas dimensões. Já, para três dimensões, os resultados obtidos deixam bastante a desejar. A Fig. 43 apresenta um exemplo da aplicação da suavização Laplaciana, onde podemos ver que a malha apresenta as linhas que unem os vértices com características mais suaves e triângulos mais homogêneos.

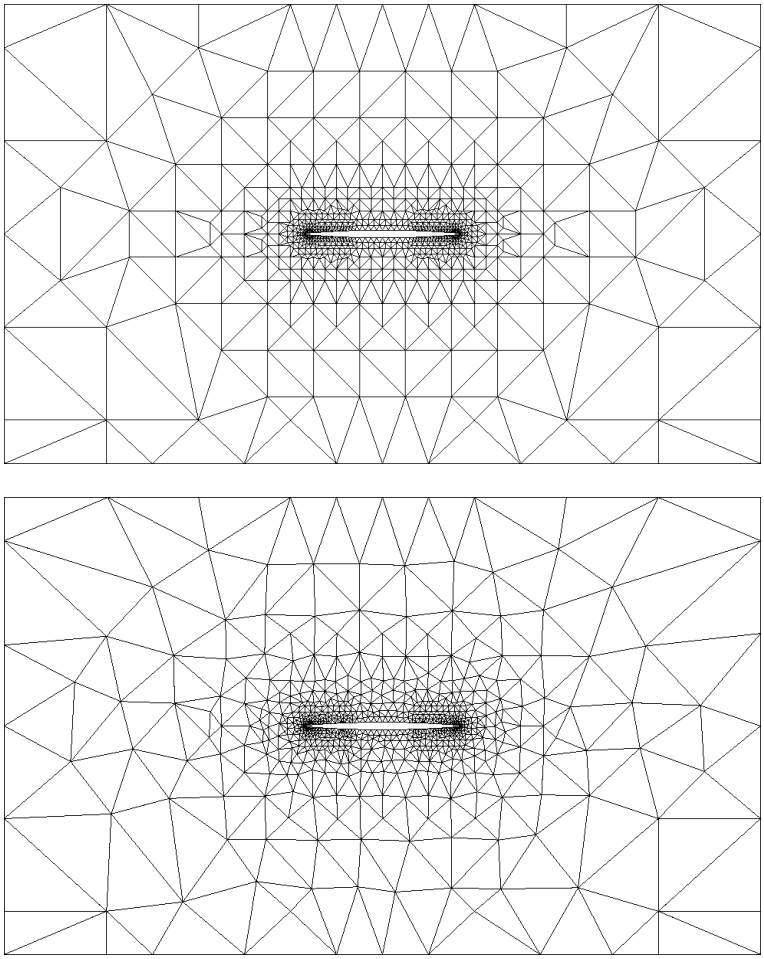


Fig. 43 - Malha de triângulos obtida pelo método quadtree, antes (superior) e após (inferior) a aplicação do método de suavização Laplaciana [55]

Como a suavização Laplaciana não é de fácil implementação, e os tempos de execução costumam ser bastante elevados, este método é, atualmente, considerado obsoleto. Diversos outros métodos de suavização bem mais eficientes foram desenvolvidos, normalmente baseados em técnicas de otimização com restrições. Na verdade, a comunidade de geometria computacional considera como atual estado da arte, os métodos de otimização não baseados em suavização, ou seja, baseados em parâmetros calculados durante a própria geração da malha, o que significa uma triangulação com restrições de suavização. O presente trabalho adota métodos deste último tipo para garantir a qualidade da malha.

3.3.3. Relaxação de malha

A relaxação de malha, desenvolvida por Frey e Field [48], é baseada em modificações topológicas e varre as arestas da triangulação verificando o grau dos vértices adjacentes[1]. Se considerarmos o quadrilátero resultante da união dos dois triângulos que compartilham a mesma aresta, este método inverte a diagonal daquele quadrilátero, ou seja, elimina a aresta em questão, criando uma nova aresta que tem grande chance de estar posicionada quase ortogonalmente a anterior (ver Fig. 44). Esta operação, no entanto, só é executada caso a soma do grau dos vértices da diagonal atual exceder em duas unidades a soma dos graus dos vértices da diagonal inversa.

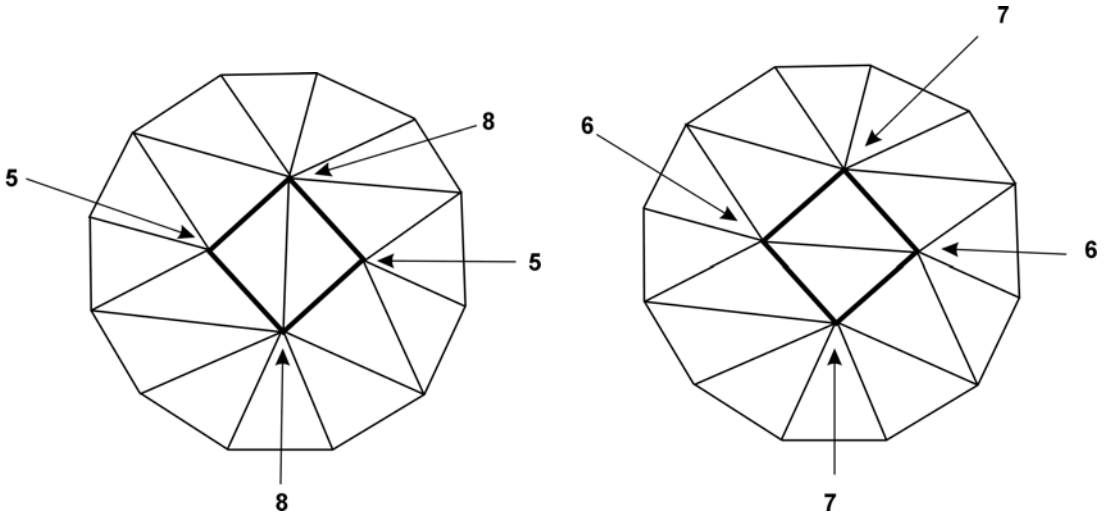


Fig. 44 - Inversão de aresta do método de relaxação de malha

[1] Grau de um vértice é o número de arestas que chegam até ele.

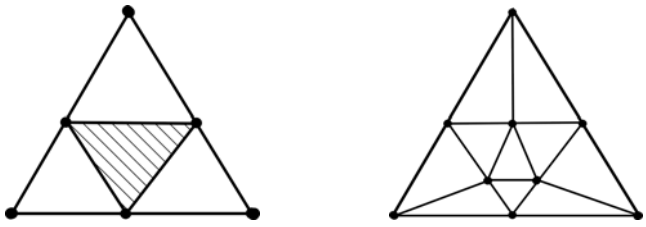
Como resultado deste método, temos a regularização, ou normalização do grau dos vértices, o que aumenta a eficiência de um posterior método de suavização Laplaciana que venha a ser aplicado.

Outra técnica que utiliza troca de diagonais de quadriláteros e, portanto, semelhante a aqui descrita, é o algoritmo de inversão de aresta. Esta técnica, como veremos logo mais, é utilizada para obtermos triangulações de Delaunay.

3.3.4. Refinamento por divisão quadrática

As metodologias apresentadas anteriormente executam operações de melhoramento de malha baseadas em parâmetros mensuráveis na própria malha (forma do elemento, grau dos vértices e etc.). No entanto, existem casos onde é importante o melhoramento de aspectos da malha baseados em parâmetros externos a mesma. Um exemplo prático disto é o refino de malha para a obtenção de soluções mais precisas em problemas de simulação numérica, como capturar fenômenos específicos que ocorram ao longo do domínio como choques, por exemplo. [61]. Uma solução inicial de um problema de simulação, utilizando uma malha grosseira, fornece informações a respeito de regiões de grandes gradientes, indicando um local adequado para a execução de um refino. Da mesma forma, em regiões com gradientes desprezíveis, as malhas podem ser engrossadas para economia de recursos computacionais. É vasta a literatura neste tipo de métodos, conhecidos como adaptativos.

O presente método, proposto por Bank [6][10], refina um triângulo qualquer da malha, onde a variação da função solução é demasiadamente elevada, através da divisão deste em quatro outros triângulos menores semelhantes (ver Fig. 45). Da mesma forma, os três triângulos vizinhos são divididos em dois, afim de manter a integridade da triangulação. A simplicidade deste algoritmo torna o engrossamento da malha uma operação trivial.



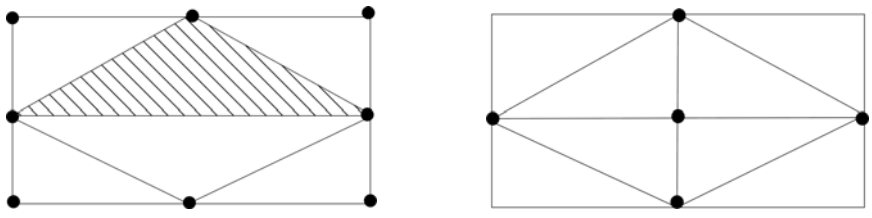
(a) (b)

Fig. 45 - Esquema do método de divisão quadrática

Este algoritmo, aparentemente funciona bem, mas apresenta o problema de “caminhar” em sentido contrário ao objetivo inicial do método, que é melhorar a malha de forma a obtermos uma solução numérica mais precisa. Durante o processo, quando divide-se o triângulo principal em quatro triângulos menores, os ângulos internos deste elementos mantêm-se inalterados, pois todos os triângulos menores são semelhantes ao maior. Para os triângulos adjacentes, no entanto, a divisão do mesmo através de uma diagonal saindo do vértice oposto até o ponto médio da aresta do triângulo central, faz o ângulo no vértice oposto cair para metade. Se efetuarmos sucessivos refinos do triângulo central, certamente o aspecto dos elementos vizinhos ficará bastante prejudicado, da mesma forma que a qualidade local da malha. Para amenizar este problema é possível utilizar, posteriormente, algoritmos de suavização. Isto, no entanto, levaria a malhas que não representariam mais um refino da malha original, trazendo à tona problemas de transporte e interpolação de propriedades, bem como outros aspectos importantes do processo de adaptatividade.

3.3.5. Refinamento de Rivara

No sentido de obter métodos de refino de malha sem prejudicar a qualidade dos elementos, Rivara [85][89] sugeriu o seguinte algoritmo recursivo: dividir o triângulo onde se deseja efetuar o refino através de uma diagonal que sai do vértice oposto ao ponto médio da maior aresta. Para manter a integridade da triangulação, divide-se o triângulo de aresta comum da mesma forma. A divisão dos elementos provavelmente irá se propagar ao longo da triangulação, no entanto o algoritmo sempre termina, já que a divisão é efetuada na aresta maior.



(a) (b)

Fig. 46 - Esquema do refino de Rivara [85][89]

Esta metodologia é denominada de refinamento de Rivara, o qual demonstrou que repetições arbitrárias de tal algoritmo nunca produzem ângulos menores que a metade do menor ângulo presente na triangulação original. Felizmente, na prática, devido à forma de se

dividir o elemento, o método de refino de Rivara costuma, além do aspecto de refino da malha, melhorar também os ângulos internos dos elementos.

3.3.6. Transformações topológica de Canann

Em duas dimensões, presume-se que o grau ideal dos vértices de uma triangulação seja seis (na tentativa de reproduzir uma topologia uniforme presente em uma malha de eqüiláteros). Com base nisto, Canann [19] efetuou diversos estudos e sugeriu mais uma metodologia baseada na topologia da malha, bastante similar à relaxação de malha de Frey e Field [48].

A metodologia de Canann, no entanto, utiliza um número consideravelmente maior de transformações, as quais são capazes de trocar, inserir e remover vértices, triângulos e arestas (não apenas a troca de diagonal, como efetuada pelo algoritmo de relaxação) de forma a tornar o grau dos vértices envolvidos na transformação igual ao valor ideal de seis. Algumas das transformações possíveis no algoritmo de Canann são exemplificadas na Fig. 47, onde se pode observar, na primeira transformação, três vértices com valores diferentes de seis serem transformados em 4 vértices de grau seis.

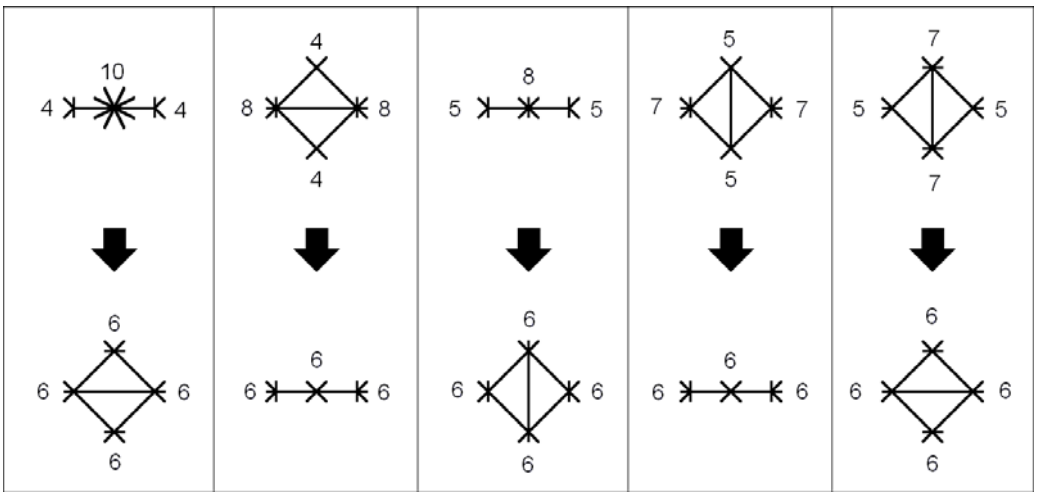


Fig. 47 - Transformações topológicas de Canann [19]

Um dos argumentos utilizado por Canann para justificar a eficiência e performance do seu método (assim como dos métodos baseados em topologia) é a não utilização de nenhum tipo de cálculo geométrico. Para completar o ciclo de melhoramento de malha, Canann sugere a utilização de uma suavização após o processo de transformação topológica, com o objetivo

de remover qualquer tipo de irregularidades morfológica. O estudo de Canann impressiona pela grande quantidade de transformações levadas em consideração.

3.3.7. Transformações de suavização/topológicas mistas

Outra possível abordagem para o melhoramento de malha é a ação combinada de métodos morfológicos e topológicos. A execução de transformações topológicas locais seguidas de suavização dos vértices envolvidos pode resolver simultaneamente os problemas de não uniformidade do grau dos vértices e disparidades na forma e tamanho dos elementos.

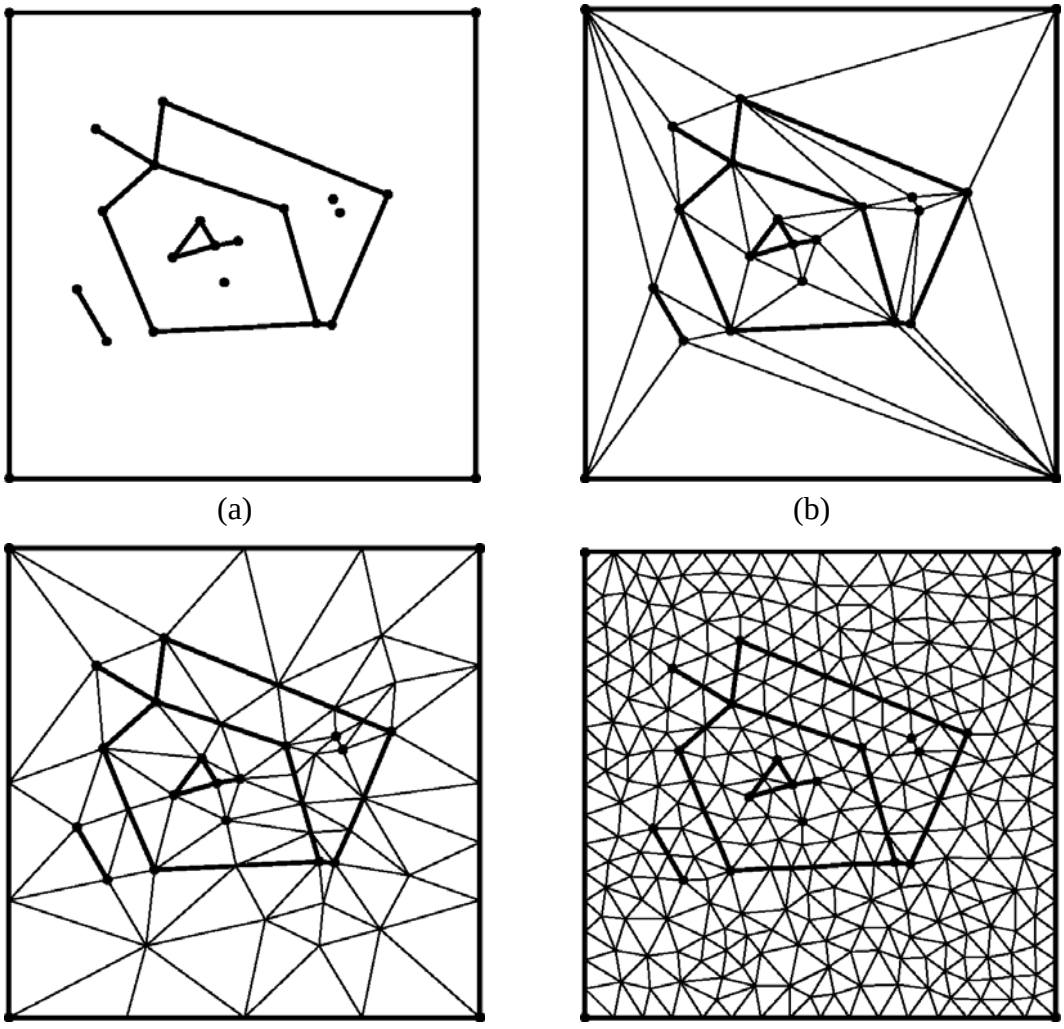
Golias e Tsiboukis [51] obtiveram bons resultados com passadas alternadas de suavização Laplaciana e relaxação de malha. Os métodos combinados, no entanto, costumam utilizar apenas transformações topológicas simples, como a inversão de diagonal. Freitag e Olliver-Gooch [45][46] foram os únicos a apresentar um trabalho que utilizasse, além da inversão da diagonal, transformações topológicas do tipo “troca de arestas”, conjugadas com a suavização Laplaciana. Da mesma forma, eles reportaram bons resultados com este método.

3.3.8. Refino de Ruppert

O algoritmo de refino de Ruppert [45] é o primeiro algoritmo de refino para triangulações de Delaunay a garantir a otimização, simultânea, de critérios de forma e tamanho. Com isto, todos os triângulos da malha têm uma razão de aspecto limitada, e o número de pontos utilizados está distante do número mínimo necessário de pontos (número ótimo) por um fator constante. Outra característica adicional do método de Ruppert é a capacidade de discretizar domínios geométricos do tipo GPSR. Na Fig. 48 é apresentada uma ilustração, onde podemos ver o GPSR em (a), a triangulação de tal GPSR sem nenhum refino em (b), com refino para respeitar a restrição de ângulo mínimo interno dos elementos de 20^o em (c), e com refino para respeitar a restrição de ângulo mínimo interno de 30^o em (d).

O método utilizado neste trabalho é uma extensão do método de Chew [23], e consiste na divisão de triângulos de má qualidade (triângulos com ângulos internos menores ou com área maior que um limite especificado pelo usuário) através da adição de novos vértices no circuncentro de tais triângulos, e na divisão de segmentos que são invadidos por outros vértices da triangulação através da adição de novos vértices no seu ponto médio. Tal método permite que os triângulos variem de tamanho ao longo do domínio, e a sua explicação

detalhada, bem como de sua implementação, é apresentada no Cap.4. Esta seção pretende apenas apresentar as diferenças do método de Refino de Ruppert e outros métodos, assim como suas características principais.



(a) (b)

(c) (d)

Fig. 48 - Triangulações de Delaunay de um GPSR [45]

O método de Ruppert, por sua vez, garante que todos os triângulos resultantes terão ângulos internos entre α e π-2α, onde α é um parâmetro controlado pelo usuário, e que pode ser escolhido entre 0° e 20°. Ângulos maiores que 20° (até cerca de 30°) podem ser também utilizados, no entanto, para estes ângulos, não é garantido que tal critério seja contemplado.

Em teoria, o método de refino de Ruppert é similar aos métodos para GPSR, apresentados em [14] e [82], com as modificações sugeridas em [11]. No entanto, na prática, ele se distingue por fatores, como:

1. A abordagem do método de refino de Delaunay é conceitualmente diferente do método *quadtree* e a conceituação é muito mais simples. Em diversos casos a sua implementação, inclusive, é mais fácil;
2. Geralmente produz um número menor de elementos;
3. É parametrizado: o usuário tem controle sobre a qualidade da malha que deseja atingir. Neste caso, o algoritmo permite diversos níveis de qualidade, trabalhando com o equilíbrio entre forma/tamanho de malha;
4. Os elementos da malha gerada não têm direção preferencial. Em contraste, o método *quadtree* obtém elementos alinhados com os eixos principais, o que pode afetar os cálculos utilizando tal triangulação;
5. A malha gerada não depende da orientação do domínio geométrico fornecido. Independente do sistema coordenado utilizado, a mesma geometria resulta na mesma triangulação.

Diversos autores fizeram adaptações ao método de refino de Ruppert, como por exemplo Shewchuk [1], para se obter melhor performance em aplicações específicas. No entanto, o presente trabalho utilizará o algoritmo original de Ruppert devido a sua generalidade.

Algoritmos de refinamento de Delaunay para a geração de malhas de triângulos, que é o caso do algoritmo aqui abordado, operam através da manutenção contínua de uma triangulação de Delaunay (ou da sua variante, a triangulação de Delaunay restrita). Esta triangulação, por sua vez, é refinada através da adição de vértices estrategicamente localizados dentro da malha, até que esta obedeça as restrições impostas quanto à forma e tamanho dos seus elementos.

Estes algoritmos obtém sucesso nesta função, devido ao fato de explorarem diversas características importantes da triangulação de Delaunay. Uma destas características, já mencionada anteriormente, obtida por Lawson [69], mostra que tal triangulação maximiza o mínimo ângulo interno dos triângulos entre todas as possíveis triangulações de um conjunto de pontos. Outra característica importante é que a adição de um vértice em tal triangulação é uma operação estritamente local, fazendo com que tal processo independa do número de elementos da malha. Da mesma forma, a adição de um ponto para melhorar um elemento de

má qualidade em uma determinada região da malha, não afetará outra região da malha que contenha apenas elementos de boa qualidade, sugerindo uma condição de estabilidade ao método.

A questão fundamental que surge é, onde localizar os pontos que serão adicionados à triangulação, depois de identificados os elementos de má qualidade. Diversos trabalhos desenvolvidos mostraram que o melhor local para se adicionar um novo ponto a uma triangulação de Delaunay é o mais distante possível de qualquer outro ponto já presente na triangulação. Caso um vértice seja adicionado próximo à um outro, os elementos resultantes certamente terão uma razão de aspecto ruim. Por sua vez, a triangulação de Delaunay, não por coincidência, é a estrutura geométrica ideal para se efetuar este tipo de procura, uma vez que o circuncentro de cada triângulo é o local, dentro de um triângulo de Delaunay, mais distante dos três vértices simultaneamente (o circuncentro de cada triângulo de Delaunay é um vértice do diagrama de Voronoi).

3.3.8.1. Variações do método

O método de refino de Ruppert, dado a sua filosofia operacional, é extremamente simples de ser modificado. Variações de tal método podem ser facilmente obtidas através da mudança dos critérios de divisão dos triângulos e arestas. Este critério pode ser avaliado tanto para parâmetros geométricos da malha (qualidade de malha) como para parâmetros externos, como, por exemplo, propriedades físicas de um simulador adaptativo. Neste caso, a malha é refinada conforme a solução física buscada.

Outra variação possível do método é a mudança da forma de localização dos pontos adicionados. Outros pontos, diferentes do circuncentro do triângulo e do ponto médio da aresta, podem ser adotados. Por exemplo: quando localizamos um novo vértice no circuncentro do triângulo, os ângulos de tal triangulo são multiplicados pelo fator dois. Isto, no entanto, pode criar ângulos demasiadamente grandes (quando um ângulo já for grande o suficiente antes da divisão do triângulo). Por outro lado, se localizarmos o novo ponto mais próximo ao vértice que contém o ângulo pequeno (e por sua vez, mais distante dos vértices que contém os ângulos grandes), faremos com que o ângulo pequeno cresça mais, enquanto que os ângulos grandes cresçam menos, obtendo triângulos com os ângulos internos mais equalizados. Esta abordagem pode levar à obtenção de malhas com um menor número de vértices, que obedeçam aos mesmos critérios de otimização.

4. Descrição do método implementado

Este capítulo descreve o método de triangulação implementado nesta dissertação. O método discretiza, utilizando uma triangulação de Delaunay, superfícies planas interconectadas no domínio tridimensional, conforme mostrado na Fig. 49. A implementação está dividida em quatro partes:

1. Tratamento do domínio 2,5D
    a. Fornecimento do domínio geométrico 2,5D;
    b. Separação das superfícies planas 3D em domínios bidimensionais.
2. Triangulação dos domínios bidimensionais
    a. Obtenção da triangulação dos vértices;
    b. Inserção na triangulação das arestas do domínio fornecido;
    c. Eliminação das arestas externas ao domínio e no interior de furos.
3. Refino da malha para obtenção dos critérios de qualidade
4. União dos domínios bidimensionais em uma única malha tridimensional

A primeira parte é uma contribuição original deste trabalho, onde foi definido um modelo geométrico hierárquico que fornece automaticamente as conectividades entre as diversas superfícies planas. Isto permite que a operação de separação das superfícies 3D em diversos domínios bidimensionais seja uma operação totalmente automatizada.

A segunda parte é uma adaptação do método de *divide-and-conquer* de Lee e Schachter [64] para a geração de uma triangulação de Delaunay restrita, que permite a representação completa da geometria fornecida na triangulação.

A terceira parte é uma adaptação do método de refino de Ruppert [92] para múltiplos domínios bidimensionais com arestas com múltiplas conexões.

A quarta e última, mais uma inovação deste trabalho, é feita a união das diversas triangulações dos domínios bidimensionais em uma única triangulação tridimensional.

Das quatro etapas apresentadas, a primeira e última são implementada especificamente para problemas 2,5D, enquanto que a segunda é específica do problema bidimensional e a terceira pode ser aplicada tanto em domínios bidimensionais como em domínios 2,5D, já que o domínio bidimensional é um caso específico do 2,5D.

4.1. Tratamento do domínio 2,5 dimensional

A primeira parte do procedimento implementado consiste no fornecimento do domínio 2,5D, por parte do usuário, ao programa de triangulação, com a sua posterior separação em diversos domínios bidimensionais interconectados. Por domínio 2,5 dimensional entende-se um conjunto de superfícies planas interconectadas no espaço tridimensional.

4.1.1. Fornecimento do domínio 2,5 dimensional- Hierarquia Geométrica

O domínio geométrico fornecido para o triangulador é um conjunto de superfícies planas interconectadas dispostas no espaço. Estas informações são fornecidas para o triangulador através da definição de uma estrutura denominada geometria hierárquica.

A geometria hierárquica é uma estrutura formada por três tipos de entidades geométricas: vértices, segmentos de reta e polígonos. Os vértices podem ser de três tipos: tri, bi ou unidimensionais, e podem ser criados, respectivamente, em qualquer espaço tri, bi ou unidimensional. Os segmentos de retas e os polígonos, podem ser criados apenas utilizando os vértices disponíveis na geometria, ou seja, os vértices servem como “âncoras” para criarmos estes elementos geométricos, que definem, respectivamente, um novo espaço uni e bidimensional local.

Desta forma, a geometria desejada é construída da seguinte forma: a raiz da hierarquia da geometria é sempre um espaço tridimensional, onde podem ser inseridos apenas vértices do tipo tridimensional. Sobre estes vértices, podem ser criados outros elementos, como um segmento de reta ou um polígono. O segmento de reta, por sua vez, define um novo espaço unidimensional, onde podem ser inseridos apenas pontos unidimensionais, enquanto os polígonos definem um novo espaço bidimensional, onde podem ser inseridos apenas pontos bidimensionais. Estes novos pontos uni e bidimensionais, podem servir de âncoras para a

criação de novas entidades geométricas, utilizando, simultaneamente, pontos uni, bi e tridimensionais, se necessário.

Devido a este encadeamento de espaços e sub-espaços, definidos pelas novas entidades geométricas criadas, é que a geometria é denominada hierárquica. Este encadeamento, por sua vez, fornece, automaticamente, a conectividade entre os elementos, não exigindo mais do usuário esta informação, que será de fundamental importância no processo final de refino. Do ponto de vista numérico, a automatização das conectividades é de auxílio extremamente valioso na programação do método, pois as conectividades são intensamente utilizadas na formulação do método.

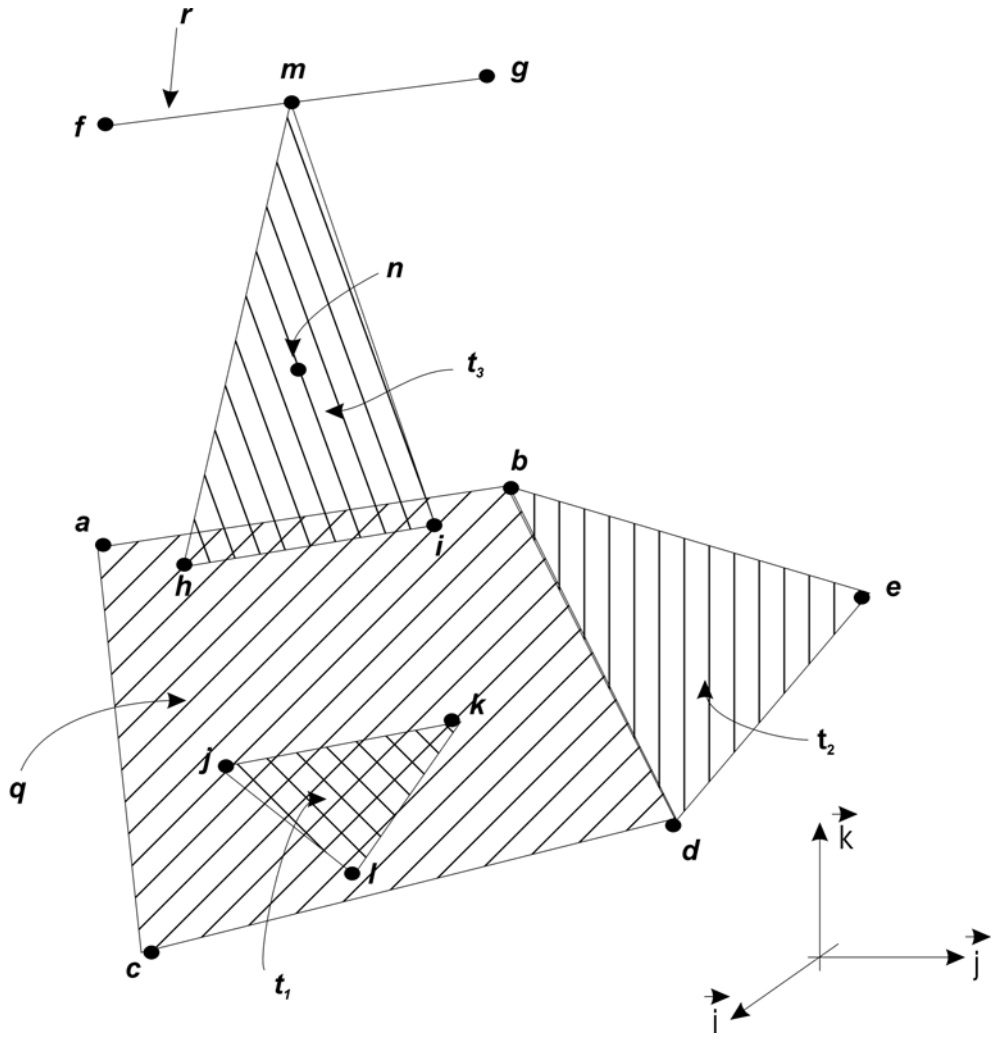


Fig. 49 - Esquema de definição da geometria 2,5 dimensional

No exemplo da Fig. 49, podemos observar as seguintes propriedades:

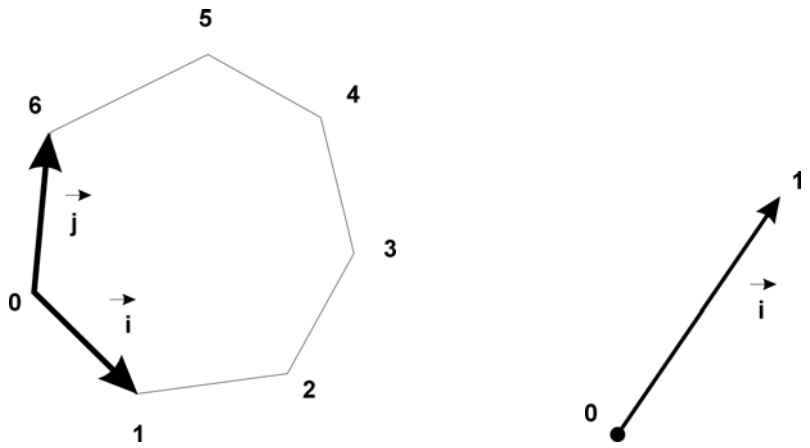
- Os vértices de *a* até *g* são vértices 3D;

- O polígono *q* (que é um quadrilátero) está ancorado nos vértices de *a-b-c-d*, e define um sub-espaço bidimensional, que contém os vértices bidimensionais de *h* a *l*;
- O segmento de reta *r*, está ancorado nos pontos tridimensionais *f* e *g*, e define um sub-espaço unidimensional, que contém o ponto unidimensional *m*.
- O polígono t_1 (que é um triângulo) está ancorado nos vértices bidimensionais *j-k-l*;
- O polígono t_2, que é um triângulo, está ancorado nos vértices *b-d-e*;
- O polígono t_3 (que é um triângulo) está ancorado nos vértices bidimensionais *h* e *i*, e no ponto unidimensional *m*.

Quanto às conexões, podem ser observadas as seguintes propriedades:

- O polígono t_1 é filho do polígono *q* por contenção;
- O polígono t_2 é irmão do polígono *q* através da aresta *bd*
- O polígono t_3 é filho do polígono *q*, através da aresta *hi*, e filho do segmento de reta *r*, através do vértice *m*.

Dentro desta triangulação, a entidade geométrica de interesse para o presente trabalho são os polígonos e segmentos de reta. Todas as outras informações utilizadas (vértices 3D, 2D e 1D) servem apenas para a obtenção automática das conectividades. Como estes polígonos e segmentos de reta definem sub-espaços para a criação interna de pontos, uma base local deve ser definida para cada um. A Fig. 50 apresenta estes vetores de base. Para o polígono (Fig. 50(a)), os vetores de base são formados pelo primeiro e o último segmento fornecido, enquanto que para o segmento de reta (Fig. 50(b)) o vetor de base é o próprio segmento.



(a) (b)

Fig. 50 - Base local para as entidade geométricas polígono (a) e segmento de reta (b)

Com isto, as informações para a criação de um vértice são o local na hierarquia em que este vai se localizar e as suas coordenadas locais. De posse destas informações, é possível calcular-se as coordenadas de qualquer ponto da triangulação em relação à qualquer um dos sub-espaços da geometria (caso este seja um pai do espaço do ponto na hierarquia da geometria), ou do espaço 3D global.

Dado um polígono p, escrito na base B_g, a passagem da coordenada de um ponto bidimensional, filho deste polígono, da base local do polígono B_l, para a base B_g, pode ser obtida por

$$p_g = T_p \times p_l$$

onde p_g é o ponto descrito na base B_g, p_l é o ponto escrito na B_l, e T_p é a matriz de transformação da base do polígono *p*, que é dada por

$$T_p = \begin{bmatrix} | & | \\ \vec{i} & \vec{j} \\ | & | \end{bmatrix}_{mx2}$$

onde $\vec{i}$ e $\vec{j}$ são os vetores de base do polígono *p*, conforme apresentado na Fig. 50. Conforme pode ser visto na equação anterior, a matriz T_p é uma matriz *mx2*, onde *m* é a dimensão do espaço formado pela base B_g.

Prossegue-se da mesma forma para o segmento de reta, com a única modificação que a matriz passa a ter a dimensão *mx1*, já que o segmento de reta define um espaço unidimensional.

A implementação da hierarquia geométrica foi toda feita em C++, utilizando orientação objeto. Esta implementação está inserida na COI-lib 2.0 (*Classes & Objetcs for Interfacing*) [39], disponibilizando um suporte para o desenvolvimento de posteriores softwares que necessitem do uso de geometrias 2,5 dimensionais. Nesta biblioteca estão disponíveis também objetos para o armazenamento e leitura em disco, assim com a visualização hierárquica da geometria, fornecendo um suporte direto para a visualização 3D.

Ao final desta primeira etapa, tem-se a geometria completa a ser discretizada, informada ao triangulador.

4.1.2. Separação das superfícies planas 3D em domínios bidimensionais

A segunda etapa do tratamento dos domínios 2,5 dimensionais é a passagem das superfícies planas 3D para domínios bidimensionais. São consideradas as superfícies planas 3D, os polígonos existentes na geometria 2,5D.

A primeira operação a ser executada, nesta etapa, é a montagem de uma lista com todos os polígonos da geometria. Para cada polígono, deve-se obter uma base local ortogonal, que defina um sub-espaço bidimensional para tal polígono. A obtenção desta base é feita usando a base local do polígono (conforme Fig. 50), e sua ortogonalização é obtida através do processo de Gram-Schmidt. A ortogonalização de Gram-Schmidt, para a base B_l, de um polígono é dada por

$$\vec{j}_o = \vec{j} - \vec{j}.\left(\frac{\vec{i}}{|\vec{i}|}\right)$$

onde $\vec{j}_o$ é o vetor de base $\vec{j}$ ortogonalizado em relação à $\vec{i}$. A base ortogonal final é formada pelo vetor $\vec{i}$ original e o vetor $\vec{j}_o$. A Fig. 51 ilustra o processo de ortogonalização de Grand-Schmidt aplicado para esta base bidimensional.

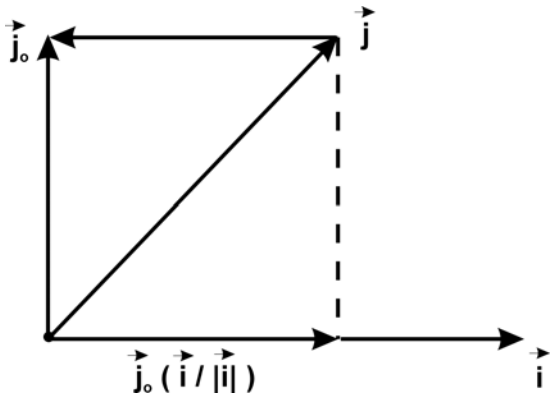


Fig. 51 - Ortogonalização do vetor $\vec{j}_o$ da base local do polígono

Uma observação a ser feita é que a triangulação final obtida deve ser escrita na base tridimensional raiz da geometria hierárquica. Como os polígonos podem estar ancorados em vértices de altos níveis da hierarquia, como por exemplo o vértice *m* da geometria da Fig. 49, deve ser efetuada a transferência das coordenadas destes vértices através de todos os espaços pais, até chegar aos espaço 3D raiz. Com isto, obtém-se para cada polígono, uma base

ortogonal centrada na origem do respectivo polígono, escrita com relação ao domínio raiz da geometria. Cada uma dessas bases formará um domínio bidimensional a ser discretizado.

De posse destas bases bidimensionais, a próxima operação é a transferência das diversas entidades filhas de cada polígono, bem como do próprio polígono, para a sua respectiva base bidimensional. Isto é efetuado através da transferência dos diversos vértices escritos nesta base, da seguinte forma, para cada uma das bases bidimensionais obtidas (ilustração na Fig. 52):

1. Para a base bidimensional em questão B_b, formada pelos vetores $\vec{i}$ e $\vec{j}$, efetua-se o produto vetorial entre $\vec{i}$ e $\vec{j}$, e obtém-se o vetor $\vec{k}$, criando-se, então, uma nova base *3x3*, denominada B_t, composta por $\vec{i}$, $\vec{j}$ e $\vec{k}$.

2. Transfere-se, do espaço do polígono para o espaço raiz da geometria, todos os seus elementos filhos. Isto significa transferir todos os vértices. Para o exemplo da Fig. 52, consideremos o vértice V;

3. Com as entidades da etapa 2 descritas no espaço raiz da geometria, transferem-se estas para o espaço formado pela base B_t. Isto é efetuado através do produto do vetor tridimensional, que representa a coordenada da entidade geométrica na base raiz, pela inversa da matriz B_t, obtida na etapa 1. Isto resultará em um ponto também tridimensional, onde as coordenadas *x,y* são a projeção do ponto no plano da base B_b, e a coordenada *z* é a distância entre este ponto e tal plano. Veja que a coordenada *z* pode não ser zero, dado que os vértices que formam um polígono não precisam, necessariamente, estar no mesmo plano. Com isto, a forma como é definida a base B_t, e a maneira como são executadas as transformações lineares de mudança de base, faz com que este desvio esteja representado unicamente na coordenada *z*.

4. Ignora-se a coordenada *z* do ponto obtido na etapa 3, sendo a coordenada final do ponto escrito na base B_b, o par ordenado *(x,y)*.

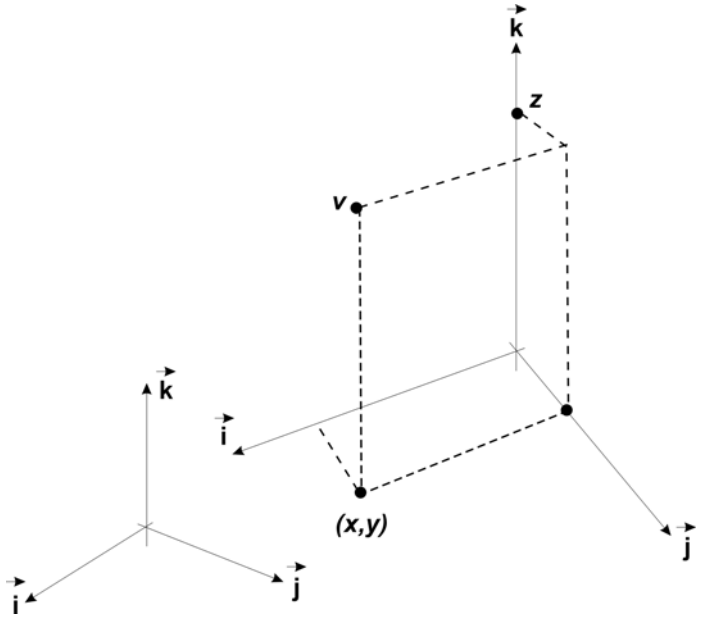


Fig. 52 - Transferência das coordenadas para os espaços bidimensionais

Neste momento, dispõe-se, de diversas bases bidimensionais, uma para cada polígono, onde esta bases estão escritas no espaço raiz da geometria. Cada uma destas bases representa um dos domínios bidimensionais do processo, e para cada um destes domínios, tem-se os vértices pertinentes escritos na sua base. Ou seja, toda a etapa morfológica do problema está pronta. Precisa-se agora, resolver a parte topológica, que diz respeito a conectividade entre estes domínios.

A conectividade entre os domínios é efetuada através do armazenamento e verificação de índices. Cada entidade da geometria possui um índice único associado. Ou seja, existe um índice para cada vértice, aresta e polígono da geometria, Como existe um domínio bidimensional para cada geometria, o índice do polígono indica diretamente o domínio.

Para cada conexão estabelecida na geometria, o objeto implementado cria, automaticamente, uma informação relativa a esta conexão. Esta informação contém:

1. Um índice de identificação da conexão;
2. O número de domínios envolvidos na conexão; e
3. Para cada domínio, o seu índice e o índice da aresta ou vértice, através do qual a conexão é estabelecida.

Desta forma, quando são montados os diversos domínios bidimensionais, basta se associar a cada ponto ou aresta deste domínio, o índice da conexão do qual este faz parte. Será visto no tópico de estrutura de dados que existe um espaço reservado na estrutura da triangulação para o armazenamento da informação de múltiplos índices. Este espaço é utilizado, aqui, para o armazenamento das conexões.

Estabelecido a forma de armazenamento das conexões, apresenta-se os tipos de conexões possíveis na geometria. As conexões podem ser dadas através de qualquer uma das entidades geométricas: vértice, aresta ou polígono completo. Os exemplos citados abaixo são pertinentes à Fig. 49.

- Conexão através de vértice: para as conexões através de vértices, a única conexão possível é vértice filho de polígono. No exemplo, o vértice *m* é filho do polígono t_3. Da mesma forma, o vértice *k* é filho do polígono t_1. Este no entanto, faz parte de uma conexão por aresta, e por isto não é considerado uma conexão por vértice. Para a conexão através de vértice, adiciona-se tal vértice como restrição geométrica do polígono e referencia-se o índice da conexão no vértice adicionado;

- Conexões através de arestas: para a conexão através de aresta, temos os seguintes tipos: polígono irmão de polígono (t_2 irmão de *q*), segmento de reta irmão de polígono e segmento de reta filho de polígono (aresta *h*-*i* do polígono t_3 é filha do polígono *q*). Quando a conexão é do tipo polígono irmão de polígono ou polígono irmão de segmento de reta, basta referenciar-se, em tal aresta, o índice da conexão, sem nenhuma modificação na estrutura geométrica. Quando a conexão é do tipo segmento de reta filho de polígono, tal segmento de reta deve ser inserido como restrição geométrica no polígono filho, e o índice da conexão referenciado na aresta adicionada.

- Conexões do tipo polígono: para as conexões do tipo polígono temos uma única possível: polígono filho de polígono. No exemplo, o polígono t_1 é filho do polígono *q*. Nesta conexão, o espaço que o polígono filho ocupa no polígono pai deve ser removido, caso contrário, irá gerar-se duas malhas no mesmo local do espaço. Esta conexão, então, forma um furo no polígono pai, e o polígono pai passa a estabelecer conexões com o polígono filho através das arestas deste. Com isto, deve-se adicionar as arestas do polígono filho no polígono pai, e a informação que no

interior de tal polígono não existe malha. Esta informação é dada pelo furo, que é um ponto no domínio que indica uma região sem malha. Nas arestas adicionadas, deve-se referenciar o índice da conexão;

Efetuada esta etapa, dispõe-se das informações geométricas dos diversos domínios bidimensionais, e as informações de conexões estabelecidas entre eles. As informações de conexão, por sua vez, estão armazenadas como índices, nas arestas e vértices, que indicam as estruturas que armazenam as informações de quais entidades participam de tal conexão.

4.2. Triangulação dos domínios bidimensionais

A segunda parte do processo de triangulação implementado no presente trabalho consiste na triangulação, independente, de cada um dos domínios bidimensionais obtidos na primeira parte. Esta parte é um processo unicamente bidimensional.

4.2.1. Estruturas de dados utilizadas na triangulação bidimensional

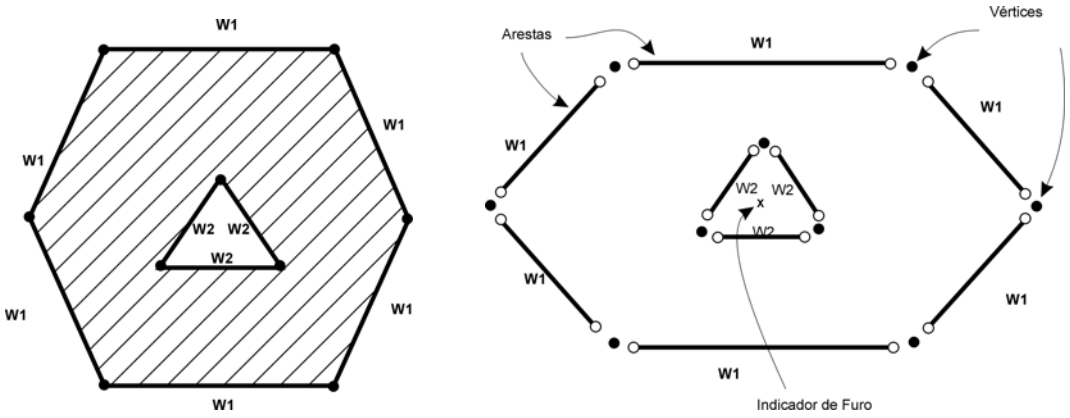
Antes de prosseguirmos para a primeira etapa do processo de triangulação bidimensional, é fundamental conhecermos as estruturas de dados que serão manipuladas neste processo.

Existem duas estruturas de dados utilizadas no processo de triangulação bidimensional: a estrutura de dados da geometria bidimensional, e a estrutura de dados da triangulação bidimensional.

A estrutura de dados da geometria bidimensional é composta por 3 entidades:

1. *Vértice*: ponto localizado no plano, com um índice associado à este;
2. *Aresta*: par ordenado de índice dos vértices que a compõe;
3. *Indicador de furo*: ponto localizados no plano, que define uma região fechada, delimitada por arestas, que não contém elementos da triangulação.

Na Fig. 53 pode-se ver um exemplo de uma estrutura geométrica que é entregue ao triangulador bidimensional. Tal estrutura é formada por nove vértices e nove arestas, com um indicador de furo no interior do triângulo central.



(a) (b)

Fig. 53 - Exemplo de geometria bidimensional (a) e os objetos de sua estrutura de dados (b)

Além das informações geométricas apresentadas acima, os vértices e arestas podem armazenar uma lista de índices definidos pelo usuário. Estes índices que podem armazenar informações pertinentes à condições de contorno, por exemplo, que serão aplicadas em tais arestas ou vértices. No presente caso, estes índices definidos pelo usuário são utilizados para indicar referências a conexão estabelecidas em tal aresta ou vértice.

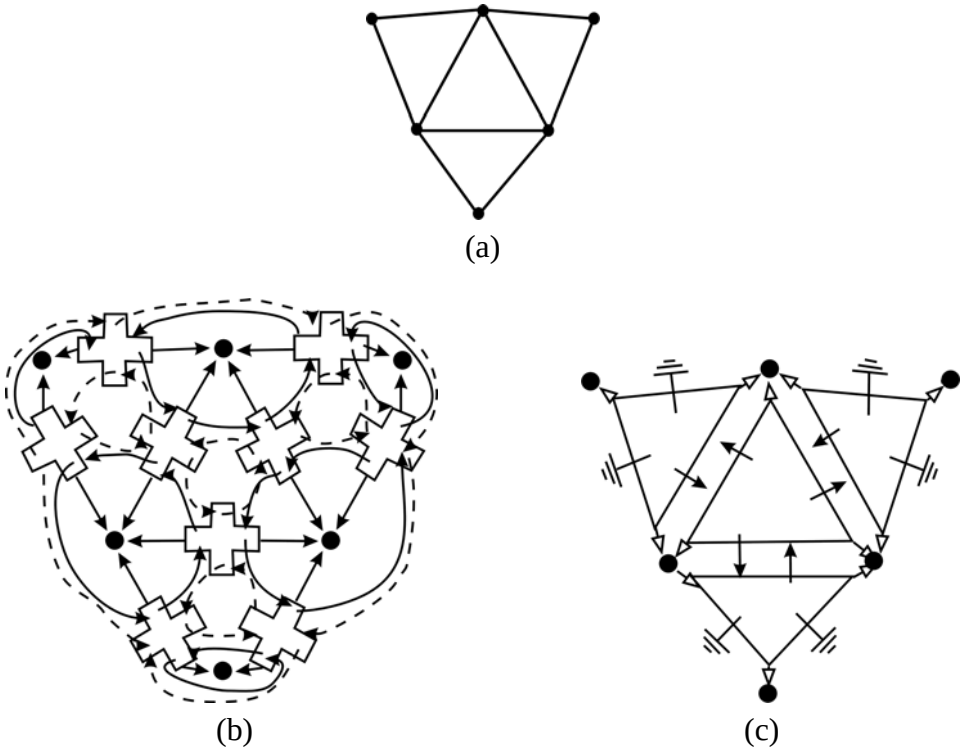
A segunda estrutura de dados utilizada no triangulador bidimensional é a estrutura de dados da triangulação. Existem duas estruturas de dados tradicionalmente utilizadas no desenvolvimento de trianguladores: *quadedge* e *triangular*.

A estrutura quadedge, proposta por Guibas e Stolfi [53], armazena as arestas da triangulação e referencias para os vértices e arestas vizinhas, sendo, sem sombra de dúvidas, a estrutura mais popular utilizada no desenvolvimento de trianguladores. Esta popularidade deve-se a abordagem elegante fornecida pela *quadedge*, que representa, ao mesmo tempo, o grafo da triangulação e seu dual (e.g., triangulação de Delaunay e o diagrama de Voronoi). Além disto, Guibas e Stolfi disponibilizaram na literatura uma extensa e detalhada quantidade de rotinas que fornecem pseudo-códigos para as operações normalmente utilizadas na triangulação.

A estrutura *triangular*, por sua vez, armazena os triângulos e referências para os vértices e triângulos vizinhos. Apesar de menos popular, os resultados descritos por Shewchuk [101], mostram que as implementações utilizando a estrutura *triangular* chegam a ser duas vezes mais rápidas que as implementações utilizando a estrutura *quadedge*. Por outro

lado, a sua não popularidade deve-se ao fato dos códigos implementados utilizando a estrutura *triangular* serem mais extensos que utilizando a estrutura *quadedge*.

A Fig. 54 apresenta um esquema das estruturas *quadedge* (Fig. 54(b)) e *triangular* (Fig. 54(c)), para um conjunto de quatro triângulos vizinhos, conforme em Fig. 54(a).



(a)

(b) (c)

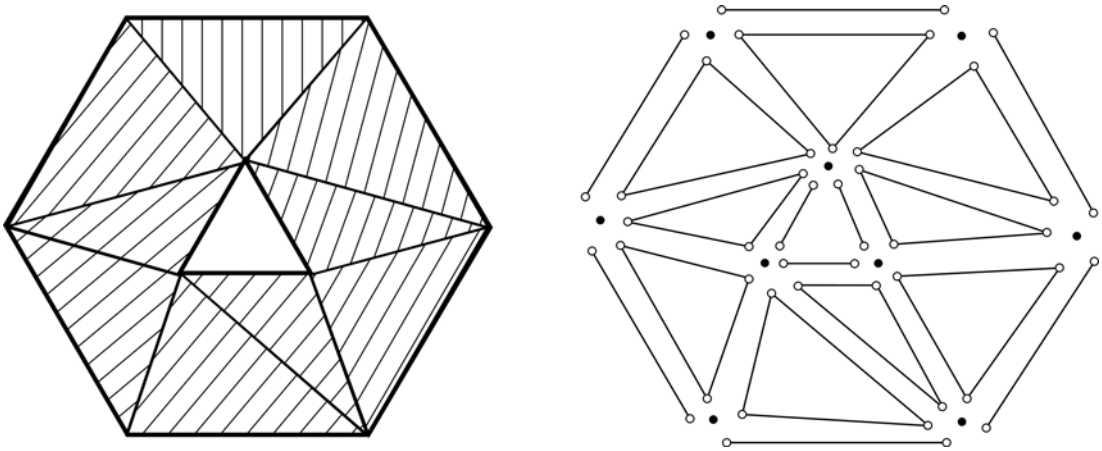
Fig. 54 - Esquema das estruturas de dados quadedge e triangular

A estrutura adotada no presente trabalho, por sua vez, é uma adaptação da estrutura triangular. Esta adaptação consiste na inclusão da entidade segmento, com o objetivo de representar, dentro da estrutura da triangulação, as arestas do domínio geométrico fornecido. A estrutura triangular modificada proposta neste trabalho é então composta pelas seguintes entidades:

1. *Vértice*: ponto no plano da triangulação, com um índice associado a este;
2. *Segmento*: par ordenado de índice dos vértices que a compõe. Além dos índices para os vértices, possui dois índices para os triângulos adjacentes;
3. *Triângulo*: conjunto de três índices dos vértices que a compõe. Além dos índices para os vértices, possui três índices para os triângulos adjacentes, e mais três índices para possíveis segmentos.

Da mesma forma que a estrutura de dados da geometria, a estrutura de dados da triangulação é capaz de armazenar um conjunto de índices definidos pelo usuário. Estes índices, então, representarão os mesmos índices informados na estrutura da geometria. Este processo é denominado de discretização dos parâmetros informados pelo usuário. A conexão, por exemplo, que foi informada através de índices nas arestas da geometria, estará presente nos diversos segmentos que serão discretizados na triangulação.

A Fig. 55 apresenta a triangulação e a estrutura de dados referente a geometria da Fig. 53. Observe que as arestas informadas na geometria estão presentes na triangulação através dos objetos segmentos, e que no interior do triângulo central não existe elementos da malha. Ou seja, não existe um triângulo em tal local.



(a) (b)

Fig. 55 - Exemplo de triangulação bidimensional (a) e os objetos de sua estrutura (b)

De posse das informações das estruturas de dados utilizadas na triangulação, pode-se iniciar a obtenção da mesma.

4.2.2. Obtenção da triangulação dos vértices

O método de discretização utilizado no presente trabalho é o *divide-and-conquer*, de Lee e Schachter [64], com adaptação para cortes horizontais e verticais, simultaneamente. O uso de cortes verticais e horizontais tem o objetivo de aumentar a robustez do método.

O método *divide-and-conquer* é um método direto, onde todos os vértices precisam ser conhecidos à priori, e o seu produto é uma triangulação de Delaunay. Nesta etapa, apenas as informações dos vértices são utilizadas no processo, e todos os vértices, e somente estes,

estarão presentes na triangulação. Desta forma, a estrutura de dados da triangulação, ao final desta etapa, possui apenas vértices e triângulos. Nenhuma informação de segmento está presente.

As etapas do método *divide-and-conquer* são as seguintes:

1. Ordenação dos vértices em ordem crescente, onde a ordenada principal de comparação é *x*, e a secundária é *y*;
2. Eliminação dos vértices repetidos;
3. Divisão do domínio em duas partes. Esta divisão pode ser através de um corte vertical, quando os limites da coordenada *x* dos vértices é maior que a coordenada *y*, e vice-versa. Caso cada parte do domínio tenha mais de 3 pontos, divide-se novamente cada parte. Isto gera um processo recursivo, onde cria-se uma árvore binária de divisões dos pontos;
4. Triangulação de cada parte. Como cada parte não possui mais de 3 pontos, a triangulação é direta;
5. Retorno da recursividade, com a união das diversas partes, até que se obtenha a triangulação completa.

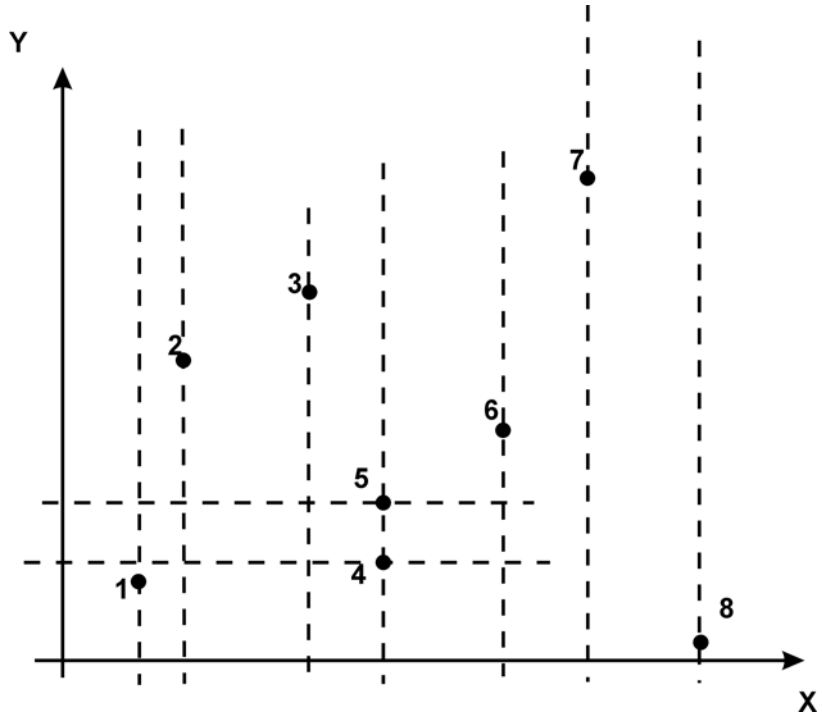


Fig. 56 - Ordenação dos pontos no método divide-and-conquer

Na implementação da primeira etapa é utilizado o método de ordenação *quick-sort*. A ordenada principal de comparação significa que um vértice é menor que o outro se a sua coordenada *x* é menor. A coordenada *y* é utilizada apenas quando a coordenada *x* dos dois vértices é exatamente igual. Na Fig. 56 podemos ver um exemplo de ordenação dos vértices. Apesar do vértice 8 ter a menor coordenada *y* que todos os outros vértices, ele é colocado em último na lista, já que sua coordenada *x* é a maior de todas. Já o vértice 4 é ordenado antes do vértice 5 pois, apesar dos dois terem a mesma coordenada *x*, o 4 tem coordenada *y* menor que o 5.

A segunda etapa consiste na eliminação dos pontos repetidos. Esta é uma operação extremamente simples, já que os pontos estão ordenados. Para isto, basta visitar todas as posições do vetor de pontos, verificando se o ponto atual é igual ao ponto seguinte. Caso verdadeiro, elimina-se o ponto seguinte, e desloca-se todo o restante do vetor uma posição abaixo.

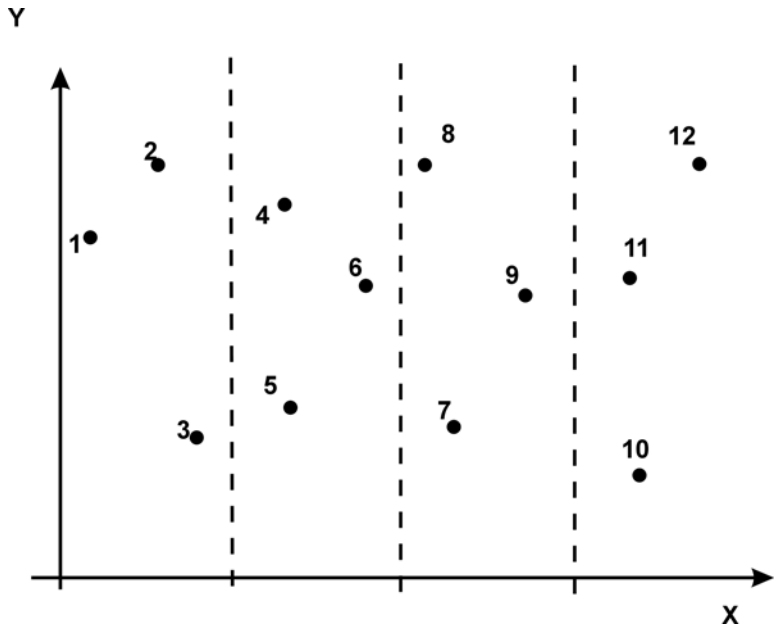


Fig. 57 - Divisão do domínio em uma árvore binária

A terceira etapa, consiste na divisão do domínio em uma árvore binária. Conforme o exemplo da Fig. 57, onde o domínio é composto por 12 vértices, o primeiro nível de divisão cria duas partes, cada uma com 6 vértices. Como cada parte possui mais de 3 vértices, divide-se novamente cada parte em duas, criando 4 partes com 3 vértices cada. O processo de subdivisão de cada parte é independente. No exemplo apresentado, apesar do processo

efetuado ser igual para cada uma das partes, poderíamos ter a subdivisão de apenas uma das partes.

A quarta etapa da triangulação é a mais simples de todas. De posse das diversas partes do domínio, cada uma com no máximo 3 pontos, criam-se os triângulos iniciais, em cada parte.

A quinta e última etapa é o coração do processo. Nesta etapa serão unidas as diversas triangulações criadas na etapa quatro. A união, por sua vez, é feita no sentido contrário à divisão do domínio. Ou seja, percorre-se ao contrário a árvore binária criada na etapa três, de forma a se chegar na raiz desta com a triangulação final completa.

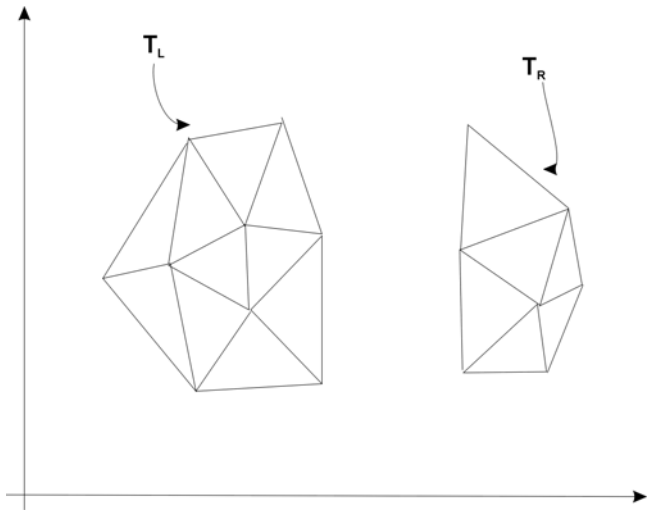


Fig. 58 - Duas triangulações disjuntas

A operação fundamental, então, a ser implementada para esta etapa é a união de duas triangulações disjuntas. Um exemplo de duas triangulação disjuntas, T_L e T_R, é apresentado na Fig. 58.

A primeira propriedade necessária para o correto funcionamento do processo de união é que as duas triangulações devem ser convexas. Isto sempre ocorre neste método de triangulação, pois a união de duas triangulações convexas é uma triangulação convexa. Como sempre inicia-se o processo com um único triângulo, que pode ser considerada uma triangulação convexa, sempre teremos triangulações convexas ao longo do processo. Isto demonstra, também, que a triangulação final obtida neste processo é convexa, conforme é a triangulação de Delaunay de um conjunto de pontos.

O processo de união das duas partes, então é dividido nas seguintes etapas:

1. Encontram-se os segmentos de reta inferior e superior, tangentes às duas triangulações. Um exemplo é apresentado na Fig. 59. Os dois segmentos de reta tangentes sempre conectam dois vértices, um em cada triangulação, indicados por V_{UL}, V_{LL}, V_{UR} e V_{LR} na Fig. 59, e estes serão, necessariamente, arestas da triangulação ao final do processo de união;

2. Em seguida, partindo-se da aresta inferior (L_0R_0, conforme Fig. 60), cria-se uma aresta auxiliar R_0L_1, e verifica-se a propriedade de Delaunay para o quadrilátero criado. Caso $\alpha + \beta \leq \pi$, dentro do quadrilátero $L_0R_0L_1L_2$, seja verdadeiro, a aresta L_0L_1 permanece. Caso contrário, a aresta L_0L_1 é removida, e o ponto L_2 passa a ser L_1, e o processo de verificação reinicia-se. Esta etapa encerra apenas quando o teste $\alpha + \beta \leq \pi$ for verdadeiro;

3. Efetua-se o mesmo processo para o quadrilátero $L_0R_0R_2R_1$, para a triangulação do lado direito;

4. Quando as arestas inválidas são todas removidas, a diagonal correta do quadrilátero $L_0R_0R_1L_1$ é adicionada, conforme pode ser visto na Fig. 61. Esta diagonal, passa então, a ser a nova aresta de base, e retorna-se para a etapa 2, executando-se o mesmo processo até que a aresta tangente superior seja atingida. Quando isto ocorre, a união das duas triangulações foi efetuada, conforme pode ser observado na Fig. 62.

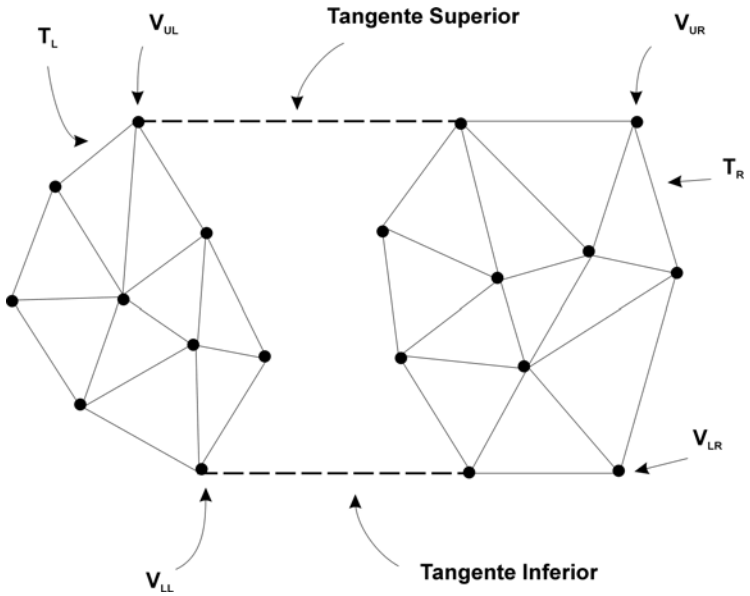


Fig. 59 - Tangentes iniciais do método *divive-and-conquer*

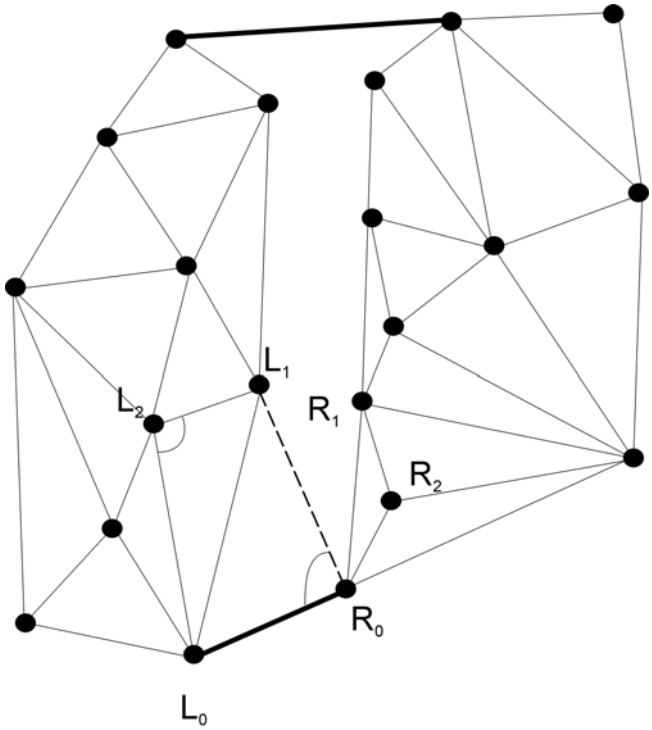


Fig. 60 - Primeira etapa da união das triangulações disjuntas

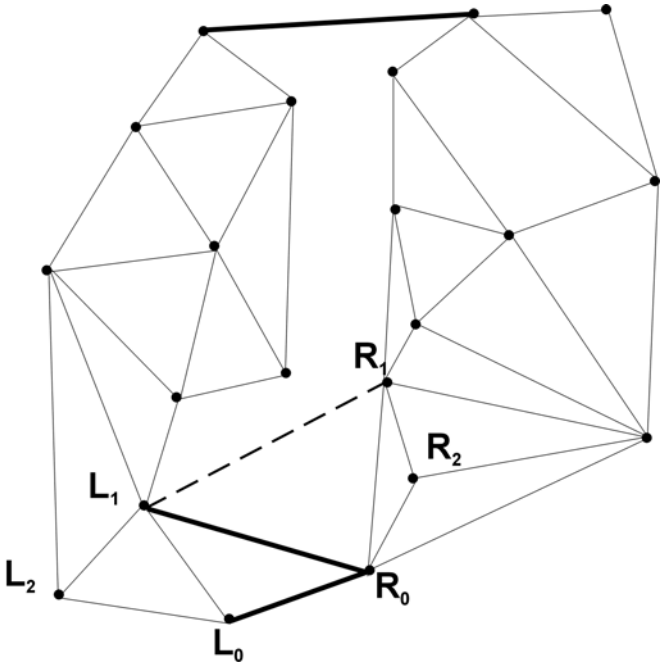


Fig. 61 - Segunda etapa da união das triangulações disjuntas

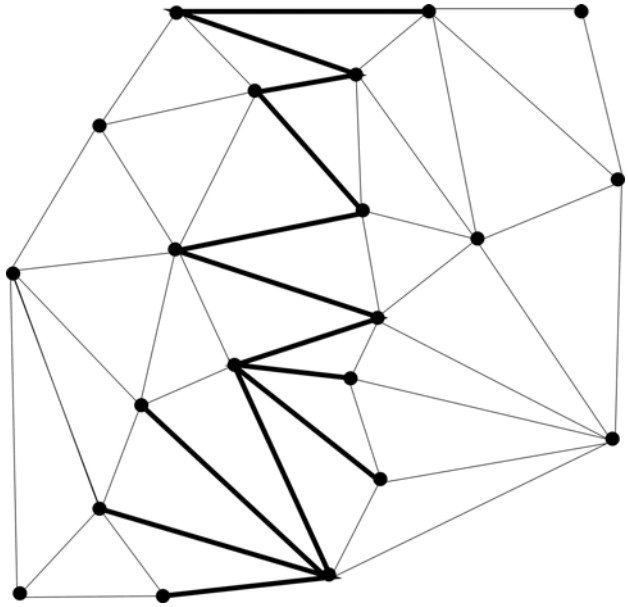


Fig. 62 - Terceira etapa da união das triangulações disjuntas

Obtida a triangulação dos pontos, são transferidos aos vértices, os índices definidos pelo usuários nos vértices do domínio geométrico bidimensional.

Ao final desta etapa, temos em mãos a triangulação do conjunto de pontos dos diversos domínios bidimensionais.

4.2.3. Inserção na triangulação das arestas do domínio fornecido

Como na etapa anterior a triangulação de Delaunay foi obtida considerando-se apenas os vértices do domínio, é possível que arestas presentes no domínio bidimensional não estejam presentes na triangulação. Esta etapa, então, tem dois objetivos:

1. Inserir tais arestas na triangulação, de forma a garantir que a geometria do domínio bidimensional esteja totalmente presente na triangulação;
2. Informar à triangulação, através da inserção dos elementos geométricos segmentos, quais os limites e regiões do domínio geométrico fornecido. As entidades segmentos, por sua vez, serão de fundamental importância para a etapa seguinte do processo de triangulação dos domínios bidimensionais;

O processamento desta etapa resume-se, então, a passar por todas as arestas do domínio geométrico, inserindo as mesma na triangulação. O processo de inserção de uma aresta na triangulação segue as seguintes etapas:

1. Localização dos dois vértices da triangulação que referem-se aos vértices do domínio geométrico que compõe a aresta a ser inserida;
2. Eliminação de todos as arestas da triangulação que interseccionam a aresta sendo inserida. Este processo cria dois polígonos convexos, um de cada lado da aresta, denominados polígonos de inserção;
3. Triangulação, respeitando as propriedades de Delaunay, dos dois polígonos de inserção;
4. Adição do segmento, com as informações de índices definidos pelo usuário.

A primeira etapa é bastante simples, se utilizarmos os índices que foram marcados nos vértices da triangulação durante a etapa de obtenção da triangulação dos vértices. Na Fig. 63(a) a aresta a ser inserida na triangulação aparece em tracejado, com a denominação de A_i.

Para a segunda etapa é preciso processar-se a interseção das arestas do domínio com A_i. Se for preciso passar por todas as arestas do domínio efetuando tal verificação, será um custo excessivo. Por isso, utiliza-se as informações de conectividade existentes na triangulação atual para realizar-se uma busca recursiva em arestas de triângulos vizinhos. A operação de busca recursiva tem a seguinte seqüência:

1. Escolhe-se um dos vértices de A_i, por exemplo V_1, e verifica-se a intersecção entre A_i e a aresta oposta de todos os triângulos do qual V_1 faz parte. Necessariamente, apenas uma aresta será interseccionada. Elimina-se, então, tal aresta;
2. Segue-se para o triângulo que possui a aresta eliminada em comum, e verifica-se a intersecção entre A_i e as outras duas arestas de tal triângulo. Novamente, apenas uma das arestas será interseccionada. Elimina-se, então, tal aresta;
3. Repete-se o processo da etapa 2, até que seja atingido um triângulo no qual V_2 seja um dos seus vértices;

Ao final deste processo, eliminamos todas as arestas que interseccionam a aresta sendo inserida. Adiciona-se, então, à triangulação, o segmento dos vértices V_1 a V_2, obtendo-se dois polígonos de inserção, conforme a Fig. 63(b).

O próximo passo é triangular tais polígonos respeitando as regras de Delaunay. Para isto, basta criar-se arestas ligando os vértices dos polígonos de inserção aos vértices V_1 ou V_2.

Para um dado vértice *V* no polígono de inserção, conforme Fig. 63(c), escolhe-se V_1 ao invés de V_2, caso a distância entre V_1 e *V* seja menor que V_2 e *V*, e vice-versa.

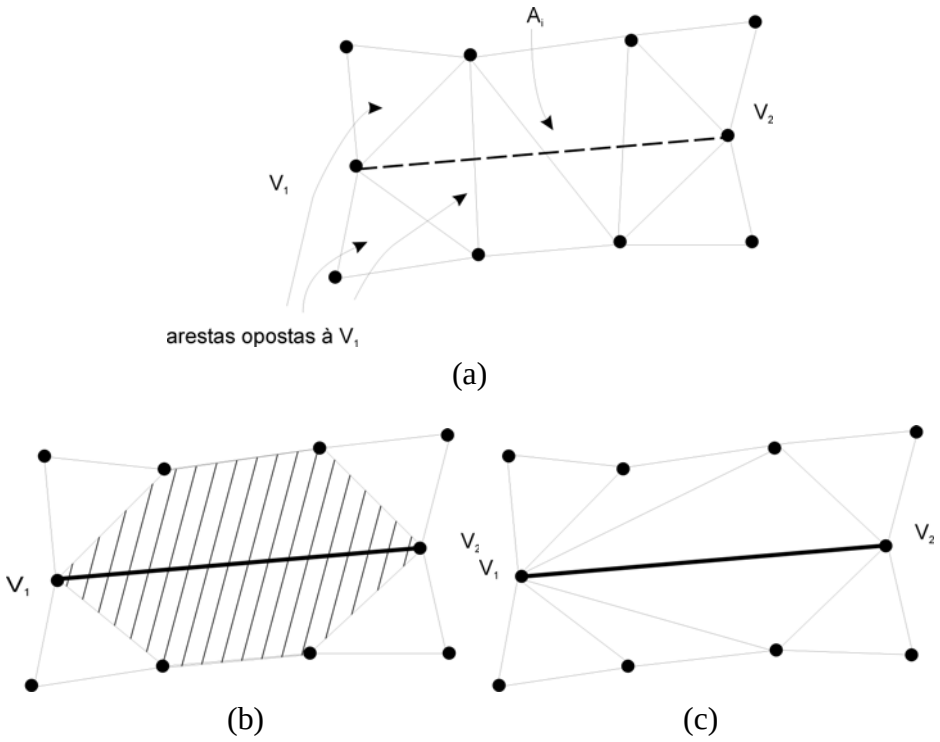


Fig. 63 - Etapas da inserção de aresta

A triangulação criada, então, não é mais uma triangulação de Delaunay, mas uma triangulação de Delaunay restrita, conforme as definições da seção [2.3.4].

O último passo do processo consiste na transferência das informações de índices criados pelo usuário para os segmentos inseridos na triangulação. Isto é trivial porque estas informações podem ser repassadas à medida que os segmentos vão sendo inseridos.

Ao final desta etapa, temos a triangulação dos diversos domínios bidimensionais, respeitando os vértices e arestas e incluindo as informações de índices definidos pelo usuário, tanto nos vértices quando nos segmentos inseridos.

4.2.4. Eliminação das arestas externas ao domínio e dos furos

Como a triangulação de Delaunay inicial foi obtida a partir de um conjunto de pontos, e as arestas do domínio geométrico foram inseridas posteriormente, a triangulação gerada

necessariamente é convexa, conforme exposto na etapa de obtenção da triangulação dos vértices. Esta triangulação, por sua vez, pode conter triângulos externos ao contorno do domínio geométrico, já que o domínio geométrico fornecido por ser não convexo. Da mesma forma, a triangulação pode conter triângulos no interior de furos definidos pelo usuário. Um exemplo é apresentado na Fig. 64.

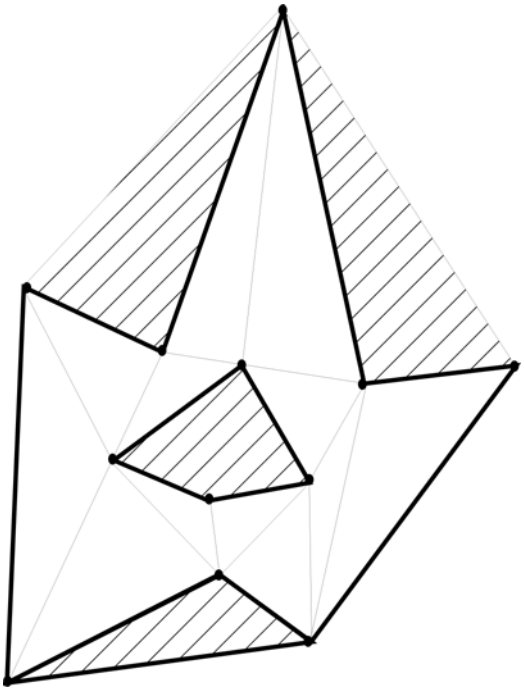


Fig. 64 - Triangulação de Delaunay e triângulos indesejados

Esta última etapa do processo de triangulação dos domínios bidimensionais, então, consiste na eliminação dos triângulos fora do domínio geométrico fornecido e no interior de furos. Somente então, a triangulação irá mapear apenas o domínio geométrico definido pelo usuário.

A eliminação destes triângulos está dividida em duas etapas:

1. Eliminação dos triângulo externos ao domínio geométrico;
2. Eliminação dos triângulo no interior de furos.

Para a primeira etapa, varre-se as arestas externas da triangulação, e caso em uma determinada aresta não esteja presente um segmento, inicia-se um processo recursivo de eliminação de triângulo a partir do triângulo que contém tal aresta. O processo recursivo funciona da seguinte forma: para um dado triângulo, caso uma de suas três arestas não contenha um segmento, elimina-se o triângulo comum com tal aresta, e utiliza-se o triângulo

eliminado como base para uma nova verificação de vizinhança. Quando mais nenhum triângulo tiver arestas sem a estrutura segmento, encerra-se o processo para tal região. Este processo irá eliminar todo um conjunto de triângulo delimitados por uma poligonal fechada de segmentos.

A primeira etapa, no entanto, só estará encerrada quando forem verificadas todas as arestas externas da triangulação. Assim, ao final da primeira etapa, todas as arestas externas da triangulação contém segmentos, fazendo com que a poligonal externa da triangulação seja exatamente a poligonal externa do domínio geométrico fornecido pelo usuário.

Para a segunda etapa, a eliminação dos triângulos no interior de furos, o processo é bastante similar. Para cada um dos pontos fornecidos pelo usuário, que definem os furos, encontra-se o triângulo que contém o ponto, e inicia-se no triângulo o processo de eliminação recursiva de triângulos. Feito isto para todos os pontos, encerra-se a presente etapa.

Ao final desta etapa, temos os domínios bidimensionais triangulados, respeitando os vértices, arestas, não convexidades do domínio geométrico e furos interiores, além das informações de índices definidos pelo usuário para os vértices e segmentos.

4.3.Refino da malha para obtenção dos critérios de qualidade

A terceira parte do processo, consiste no refino da malha para a obtenção dos critérios de qualidade. A presente implementação permite a utilização de restrições mínimas de tamanhos de elemento (tanto a área quanto o comprimento de aresta) e restrições mínimas de mínimo ângulo interno dos triângulos, simultaneamente.

O método de refino utilizado é uma adaptação para múltiplos domínios bidimensionais com arestas com múltiplas conexões do método de refino de Ruppert [92]. Este por sua vez, pode ser também aplicado a domínios bidimensionais.

A idéia básica deste método é a manutenção contínua, ao longo de todo o processo, de uma triangulação de Delaunay válida. Tal triangulação é, então, utilizada para decidir-se onde novos vértices serão inseridos de forma a remover-se triângulos de má qualidade. Por triângulos de má qualidade entende-se triângulos com área maior que a mínima área definida

pelo usuário, ou com algum ângulo interno menor que o definido. Cada adição de vértice representa uma retriangulação, onde métodos incrementais têm aplicação direta.

Conforme Ruppert, um local adequado para a adição de um novo vértice é o circuncentro de um triângulo de má qualidade. Isto pode ser observado pela seguinte propriedade geométrica elementar: *se um triângulo t = abc tem o ângulo bca = θ, e o ponto p é o circuncentro de t, então o ângulo bpa = 2θ.* A Fig. 65 ilustra esta propriedade.

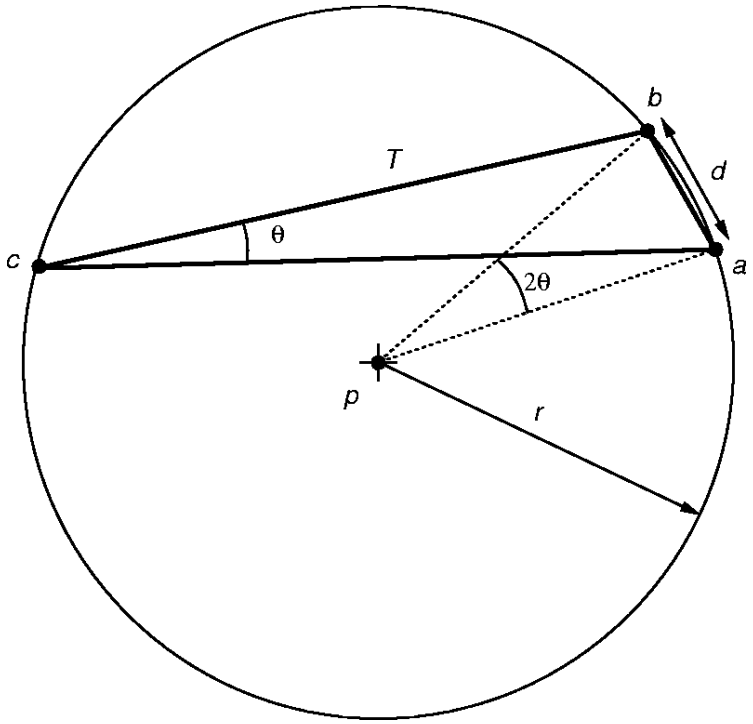


Fig. 65 - Modificação do ângulo com a adição do novo vértice no circuncentro

As operações básicas do método de refino são duas:

1. Divisão de uma aresta através da adição de um vértice no seu ponto médio;
2. Divisão de um triângulo através da adição de um vértice no seu circuncentro.

Um conceito a ser definido antes de apresentar-se o algoritmo de refino, é o conceito de *invasão de segmento*. Primeiramente, vamos definir o que é um circulo diametral: dado um segmento *s*, o círculo com diâmetro *s* é denominado *círculo diametral*. Quando um vértice *v* encontra-se dentro do círculo diametral de *s*, diz-se que tal vértice *invade o segmento s*. A Fig. 66 ilustra este fato: o vértice *a* invade os segmentos s_1 e s_2 (apenas o círculo diametral de s_1 é apresentado).

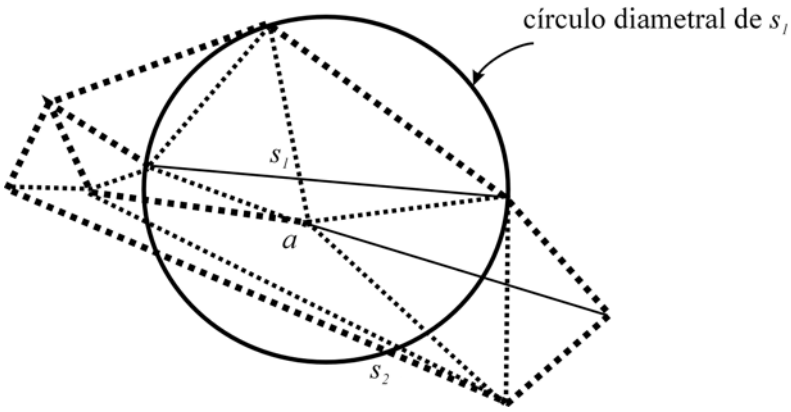


Fig. 66 - Inserção de vértice

Na sua essência, o algoritmo de refino diz que triângulos de má qualidade (estreitos ou grandes) devem ser divididos, a menos que o circuncírculo destes invada algum outro segmento da triangulação (segmento são apenas aqueles definidos no domínio geométrico, e não qualquer aresta). Neste caso, este segmento deve ser dividido também. A divisão destas entidades que compõe a triangulação é efetuada conforme exposto no começo desta seção.

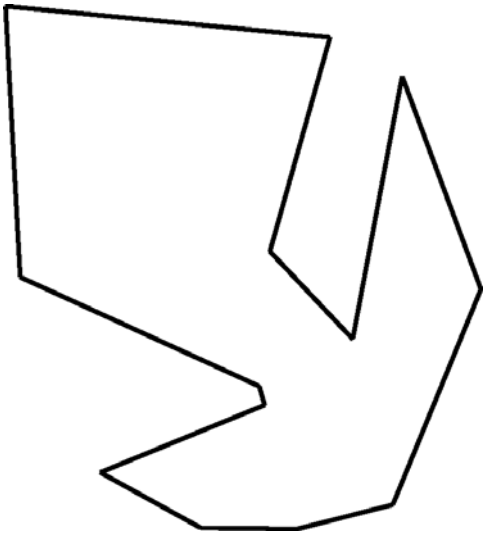
Segue, em linguagem natural, o algoritmo de refino detalhado. Primeiramente são apresentadas as duas rotinas básicas de divisão de segmento de reta e triângulo, e então o algoritmo principal, que utiliza tais rotinas.

```
routine SplitTriangle( Triangle t )
     Add circumcenter of t to V, updating DT(V)

routine SplitSegment( Segment s )
     Add midpoint of s to V, updating DT(V)
     Remove s from S, add its two halves s1 and s2 to S

algorithm DelaunayRefine
INPUT:    planar straightline graph X;
          desired minimum angle bound α
OUTPUT:   triangulation of X, with all angles ≥ α
Initialize
     add a bounding square B to X:
          compute extremes of X: xmin, ymin, xmax, ymax
          let span(X) = Max(xmax-xmin, ymax-ymin)
          let B be the square of side 3 X span(X), centered on X
          add the four boundary segments of B to X
     let segment list S = edges of X
     let vertex list V = vertices of X
     compute initial Delaunay triangulation DT(V)
```

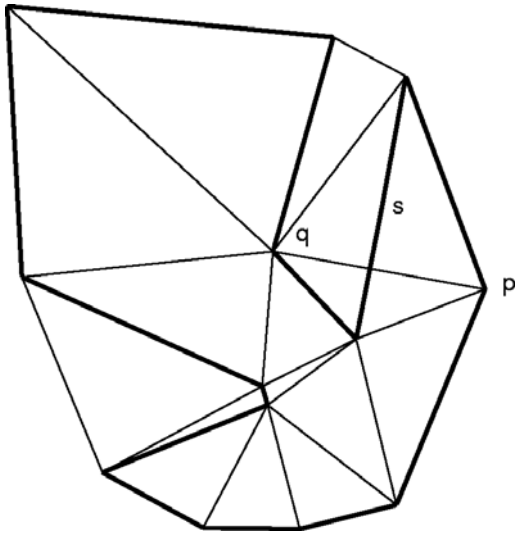
```
repeat
     while any segment s is encroached upon:
          SplitSegment( s )
     let t be (any) skinny triangle (min angle < α)
     let p be t circumcenter
     if p encroaches upon any segment s1,...,sk then
          for i=1 to k:
               SplitSegment( si )
     else
          SplitTriangle( t ) (* adds p to V *)
     endif
until no segments encroached up, and no angles < α
output current Delaunay triangulation DT(V)
```

A Fig. 67 apresenta alguns dos passos da execução do algoritmo de refino de Ruppert sobre um domínio poligonal como exemplo. Para tornar a visualização mais clara, não é apresentada a fronteira envolvente. Em cada figura, o domínio fornecido é mostrado com linhas em negrito, enquanto que a triangulação de Delaunay é sobreposta com linhas normais.

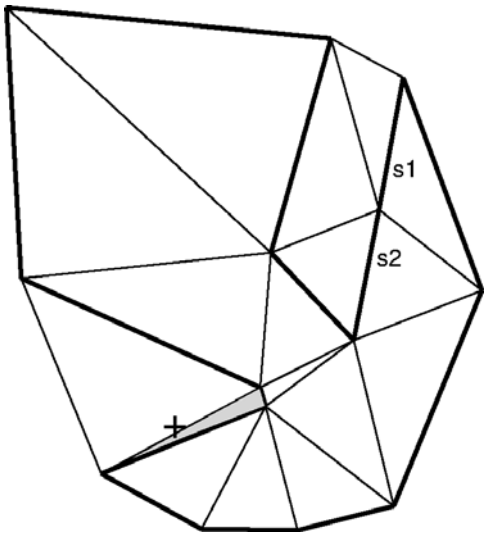
Podemos ver na Fig. 67(a), a geometria poligonal bidimensional fornecida pelo usuário. Na Fig. 67(b), a triangulação de Delaunay do conjunto de pontos do domínio fornecido. Observe que *s* não é uma aresta de Delaunay, pois esta corta o segmento *pq* do domínio geométrico. Na Fig. 67(c), o segmento *s* é dividido no seu ponto médio em duas partes: s_1 e s_2. Ao mesmo tempo, o triângulo destacado possui o menor ângulo interno dos elementos da triangulação (5.9^o). A cruz indica o seu circuncentro, onde será adicionado mais um vértice. Na Fig. 67(d), o vértice *p* já foi adicionado, no entanto este invade os segmentos s_3 e s_4. Na Fig. 67(e), os segmentos são então divididos, nos pontos *q* e *r*. Novamente, o triângulo com o menor ângulo interno é destacado (9.8^o), e a cruz representa seu circuncentro, onde um novo vértice será adicionado. Na Fig. 67(f), o vértice já foi adicionado, e o triângulo com o menor ângulo interno novamente é destacado. Processando sucessivamente, até que não exista nenhum triangulo com angulo interno menor que 25^o, obtemos a triangulação da Fig. 67(g). Esta no entanto, possui triângulos fora da poligonal que define o domínio. Na Fig. 67(h) tais triângulos já foram removidos, e a triangulação final é apresentada.



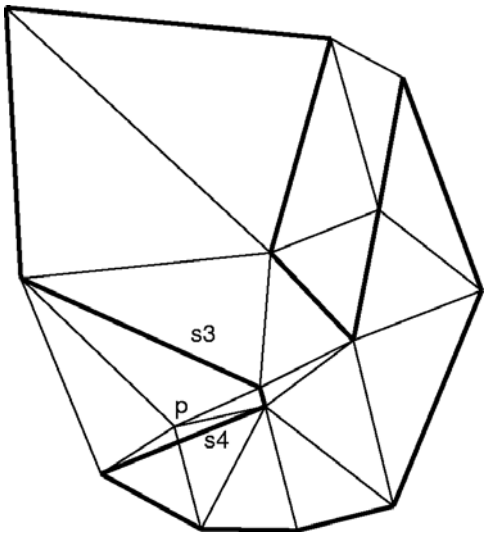
(a)



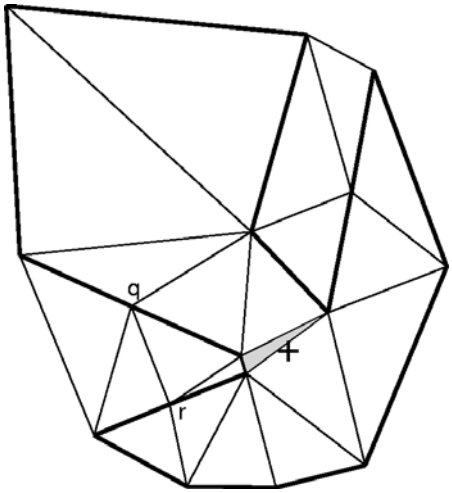
(b)



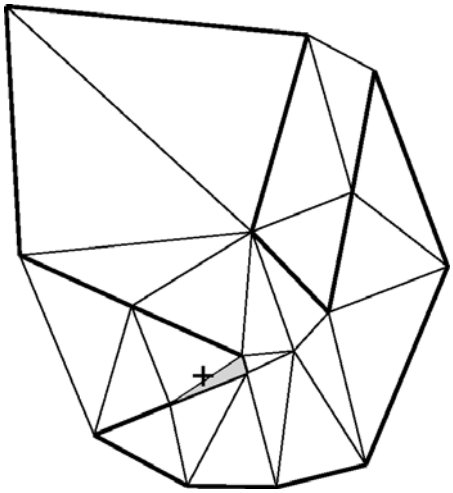
(c)



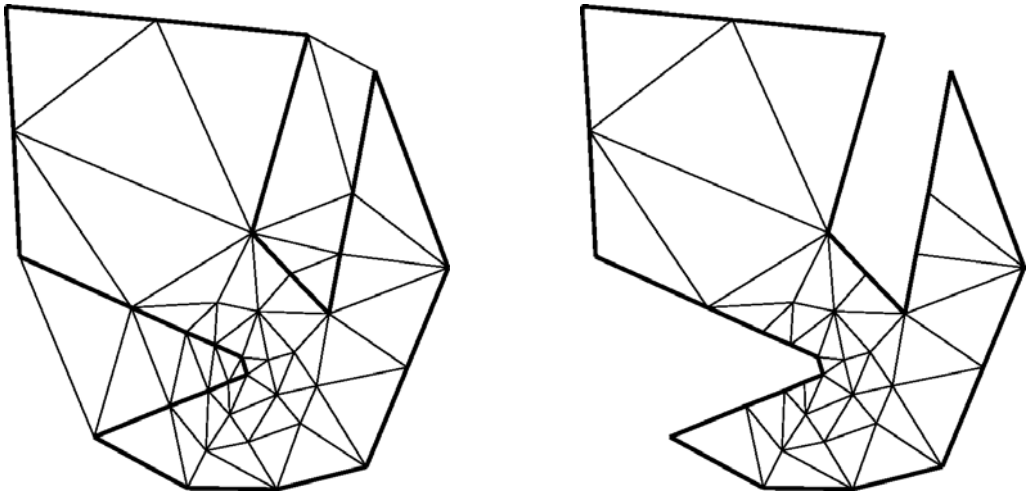
(d)



(e)



(f)



(g) (h)

Fig. 67 - Exemplo de refino

É possível mostrar que o método de refino implementado sempre encerra com sucesso para ângulo mínimos menores que 25^o. Na prática, no entanto, consegue-se normalmente gerar malhas com ângulos internos maiores que 30^o.

A extensão do método de refino tradicional para múltiplos domínios com múltiplas conexões está exatamente no momento de divisão de um segmento. Quando um segmento, que possui uma informação de conexão, é dividido em dois, tal divisão deve ser refletida em todos os outros domínios que participam da conexão através de tal aresta. Isto faz com que exista sempre uma concordância das arestas nos segmentos que conectam os domínios.

Pela forma que o método é concebido, este não leva em consideração nenhuma informação do ângulo entre os domínios. Por este motivo, é como se todos os domínios estivessem no mesmo plano, conectados por tais segmentos, mas totalmente independente com relação à todas as informações morfológicas. Estes domínios bidimensionais, então, podem ser vistos como um conjunto de “layers” conectados entre si, e por isto, o nome do método é “camadas topológicas”.

4.4. União dos domínios bidimensionais

A quarta parte, da mesma forma que a primeira, é implementada especificamente para o problema 2,5D. Nesta parte, diversas triangulações bidimensionais, independentes, interligadas apenas pelas informações de conexão obtidas da geometria na primeira parte, são convertidas em uma única triangulação tridimensional.

A primeira operação que deve ser feita, então, é a renumeração de todos os vértices e arestas e índices que referenciam-se à tais entidades. Em seguida, as coordenadas de todos os vértices devem ser transferidas dos domínios bidimensionais, de volta para o domínio tridimensional original. Isto é feito através da multiplicação das coordenadas de cada vértice, pela matriz, B_b, obtida na seção 1.1.2, conforme abaixo.

$$v_t = B_b \times v_b$$

onde v_b é a coordenada do vértice em seu respectivo domínio bidimensional, e v_t a coordenada final do vértice da triangulação no domínio tridimensional.

Feitas estas transformações lineares, a malha tridimensional 2,5D final foi obtida.

Um aspecto importante a ser observado é que os vértices que ocupam a mesma posição espacial, mas que estavam em domínios bidimensionais diferentes (vértices de segmentos com conexões) não são “fundidos” em um único vértice. Na verdade, além de serem mantidos tais vértices, é adicionado um novo vértice, no mesmo local do espaço que estes outros, que representa a conexão em sí. Um esquema de uma conexão com três placas pode ser vista na Fig. 68.

Esta abordagem é extremamente interessante para lidar-se com resistências de contato em contados com múltiplas placas, um tópico de grande interesse nas aplicações que utilizam a malha aqui obtida. No exemplo da Fig. 68, os vértices V_1, V_2, V_3 e V_c estão todos na mesma coordenada espacial. No entanto, os vértices V_1, V_2 e V_3, representam tal coordenada em cada uma de suas respectivas placas, enquanto que o vértice V_c representa tal coordenada exatamente no contato. Se estivermos utilizando um método numérico que armazena temperaturas em tais vértices, é possível ter-se uma temperatura para cada placa, e uma temperatura representando o contato.

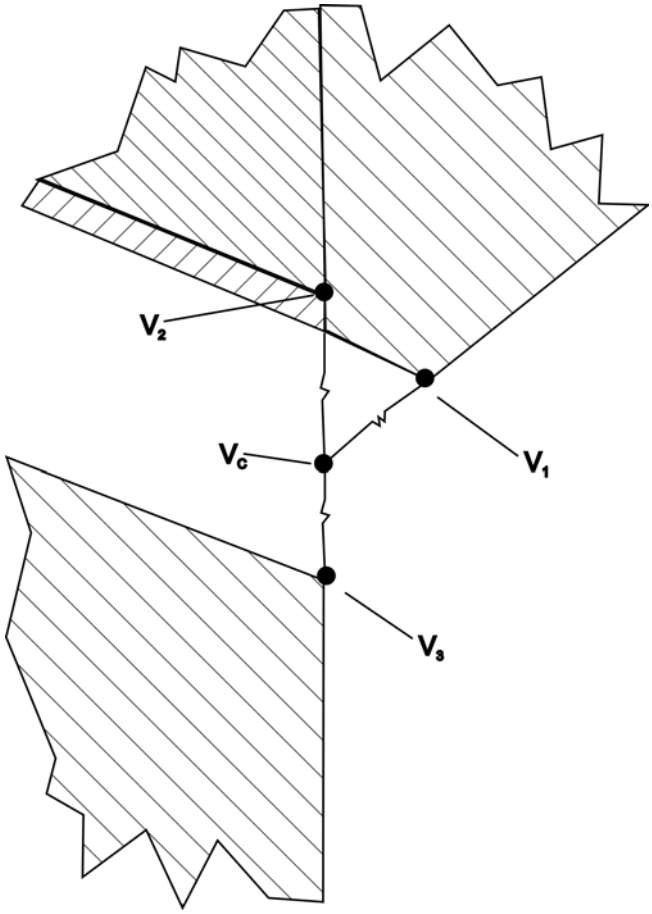


Fig. 68 - Esquema explodido das conexões entre domínios

Ao final desta etapa, temos a triangulação das superfícies planas 3D, considerando-se restrições geométricas dadas pelas conexões entre estas, e com critérios de qualidade de malha.

4.5. Características computacionais do código implementado

O código descrito anteriormente foi totalmente implementado em C++, com orientação objeto. Tal código está disponível no módulo científico da biblioteca COI-lib 2.0, e foi aplicado no do programa SATER 100, provendo a discretização para a solução do problema conjugado de condução e radiação resolvido por tal software.

As triangulações geradas por tal trabalho também foram utilizadas no trabalho de dissertação de mestrado entitulado “Simulação Numérica da Transferência de Calor em Problemas Radiativos Condutivos”, do aluno Marcus V. F. dos Reis.

5. Resultados

O presente capítulo apresenta um conjunto de malhas obtidas com o triangulador descrito no capítulo 4. Os resultados estão divididos em 2 grupos: bidimensionais e 2,5 dimensionais, com os resultados bidimensionais sendo utilizados para analisar o comportamento e a flexibilidade do triangulador para adaptar-se à geometrias diversas com diferentes restrições geométricas e critérios de qualidade de malha. Estes resultados fornecem uma indicação da capacidade do método desempenhar adequadamente para geometrias 2,5 dimensionais. Os resultados 2,5 dimensionais são o objetivo final deste trabalho.

Os resultados bidimensionais estão divididos em 16 casos, e os resultados 2,5 dimensionais em 3 casos. Para cada caso, são apresentados os seguintes dados:

1. Fornecidos pelo usuário:
    - Número de vértices;
    - Número de arestas;
    - Número de furos.

1. Saída do triangulador:
    - Tempo total de CPU;
    - Número de vértices;
    - Número de triângulos;
    - Número de arestas;
    - Número de arestas presentes no contorno geométrico.

Para os casos onde efetua-se uma análise paramétrica, além dos dados citados acima, apresenta-se os valores utilizados no parâmetro estudado, assim como imagens das malhas geradas e uma análise dos resultados obtidos no caso em consideração.

Para todos os gráficos apresentam-se, simultaneamente, a quantidade dos diversos tipos de entidades geométricas utilizadas na triangulação na abscissa esquerda, e o tempo total de CPU, na abscissa direita.

Todos os resultados foram obtidos utilizando-se um microcomputador Intel Pentium II 300 Mhz, com 256 Megabytes de memória RAM e sistema operacional Windows NT 4.0.

5.1.Resultados bidimensionais

5.1.1. Caso 1: Quadrado com restrição de área máxima de elementos

O presente caso tem o objetivo de analisar o comportamento do gerador quando imposta uma restrição de área máxima para os triângulos. A geometria é um quadrado com lado de tamanho unitário, e foram impostas cinco condições diferentes de tamanho de malha: área livre e área máxima de 10^{-1}, 10^{-2}, 10^{-3} e 10^{-4}. O número de vértices fornecidos é 4, com 4 segmentos.

Caso 1					
Inverso da área	0	10	100	1000	10000
Tempo total	44	60	95	250	971
Vértices	4	13	81	545	8321
Triângulos	2	16	128	1024	16384
Arestas	5	28	208	1568	24704
Arestas de contorno	4	8	32	64	256

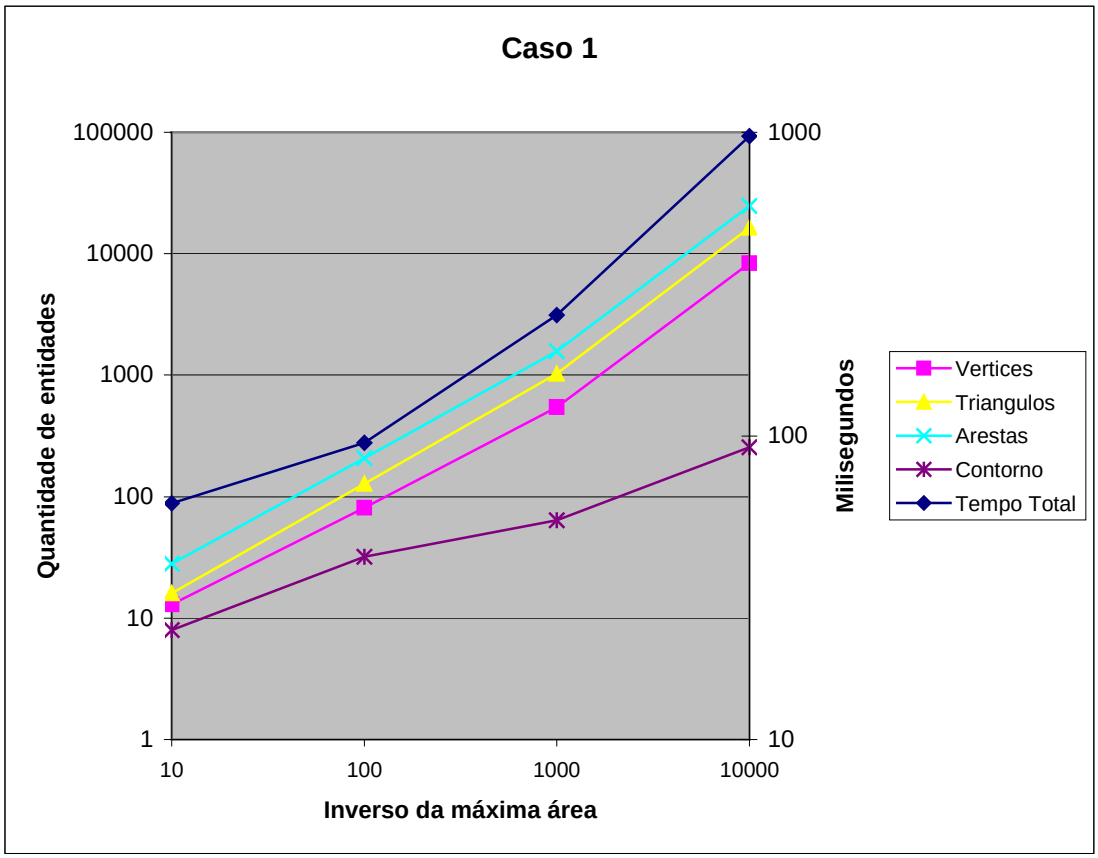


Fig. 69 – Caso 1: Gráfico dos resultados obtidos

Conforme pode ser observado na Fig. 69, o número de entidades geométricas utilizadas pelo triangulador é linear com relação ao inverso da restrição de área máxima imposta. O tempo de CPU também apresentou-se linear quanto à quantidade de elementos utilizadas.

Observando a Fig. 70, vemos que o método cria malhas uniformes, refinando em escala os elementos até que todos os triângulos satisfaçam o critério imposto. Como todos os triângulos são triângulos retângulos isósceles, esta triangulação respeita o critério de mínimo ângulo interno com valor 45^{o}. É importante lembrar, no entanto, que tal critério foi obtido secundariamente, e não era o alvo da otimização.

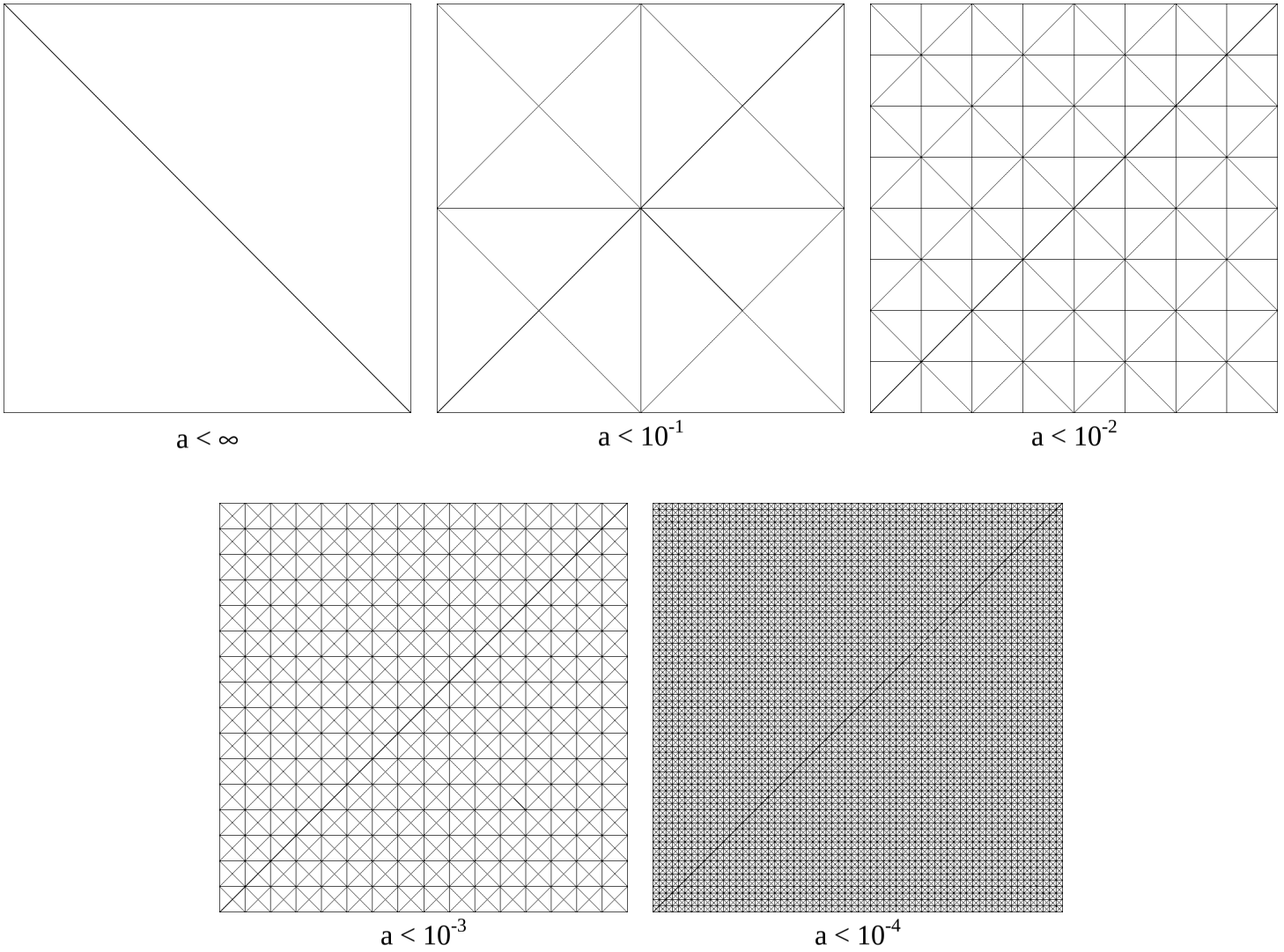


Fig. 70 - Caso 1: visualização das triangulações obtidas

5.1.2. Caso 2: Quadrado com refino na aresta inferior e restrição de mínimo ângulo interno de elementos

O caso 2 tem o objetivo de analisar o comportamento do triangulador quanto à restrição de mínimo ângulo interno dos triângulos. A geometria é um quadrado com lado de tamanho 1 e com uma restrição na aresta inferior de comprimento máximo de aresta dos triângulos de $2x10^{-2}$. São utilizados 9 valores diferentes de ângulo interno mínimo: 0^o, 5^o, 10^o, 15^o, 20^o, 25^o, 30^o, 32^o e 34^o. O número de vértices fornecidos é 4, e de segmentos 4.

Caso 2									
Ângulo mínimo	0	5	10	15	20	25	30	32	34
Tempo total	50	51	52	56	58	60	66	142	347
Vértices	53	69	84	97	107	124	202	301	679
Triângulos	51	76	104	130	148	182	326	521	1249
Arestas	103	144	187	226	254	305	527	821	1927
Arestas de contorno	53	60	62	62	64	64	76	79	107

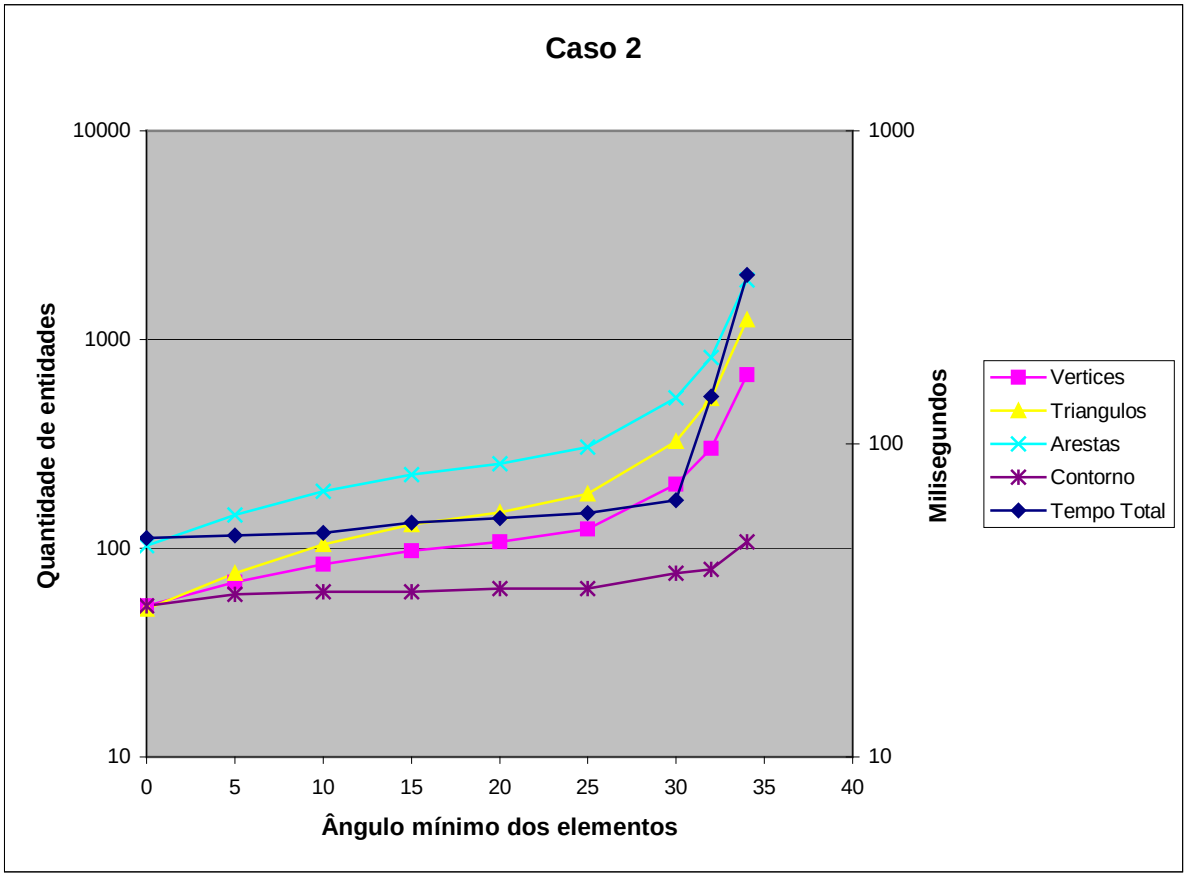


Fig. 71 - Caso 2: Gráfico dos resultados obtidos

Conforme o gráfico de resultados obtidos, podemos observar um crescimento exponencial no número de elementos utilizados e tempo de computação com relação ao mínimo ângulo interno. Este comportamento, no entanto, só ocorre até cerca de 30^o, onde então, o número de elementos utilizado passa a crescer muito rapidamente.

A explicação para este comportamento pode ser encontrada nas visualizações das malhas obtidas na Fig. 72. Mas malhas com até 25^o de ângulo mínimo interno, os elementos com os piores ângulos internos encontram-se exclusivamente na fronteira inferior. Quando a restrição de ângulo chega à cerca de 30^o, os elementos nesta fronteira são todos triângulo muito próximos de eqüilátero, que é a forma ideal de ser encontrada em uma triangulação. A partir deste momento, não interessa mais ao triangulador refinar elementos nesta região, onde então o processo passa a propagar-se para dentro do domínio, conforme pode ser visto na figura com ângulo mínimo de 34^o.

Outra informação que pode ser obtida observando-se a Fig. 72, é que o método consegue uma grande variação de tamanho dos elementos em uma curta distância. Isto permite, para este caso, que a grande quantidade de elementos adicionadas em função da restrição de mínimo ângulo interno fique concentrada próxima à aresta inferior. Para os casos 2,5D, este comportamento é interessante, pois minimiza a influência em superfícies planas adjacentes de restrições impostas em uma superfície plana específica.

O triangulador não conseguiu obter, independentemente do número de elementos utilizados, uma malha com restrição de ângulo interno de 35^o.

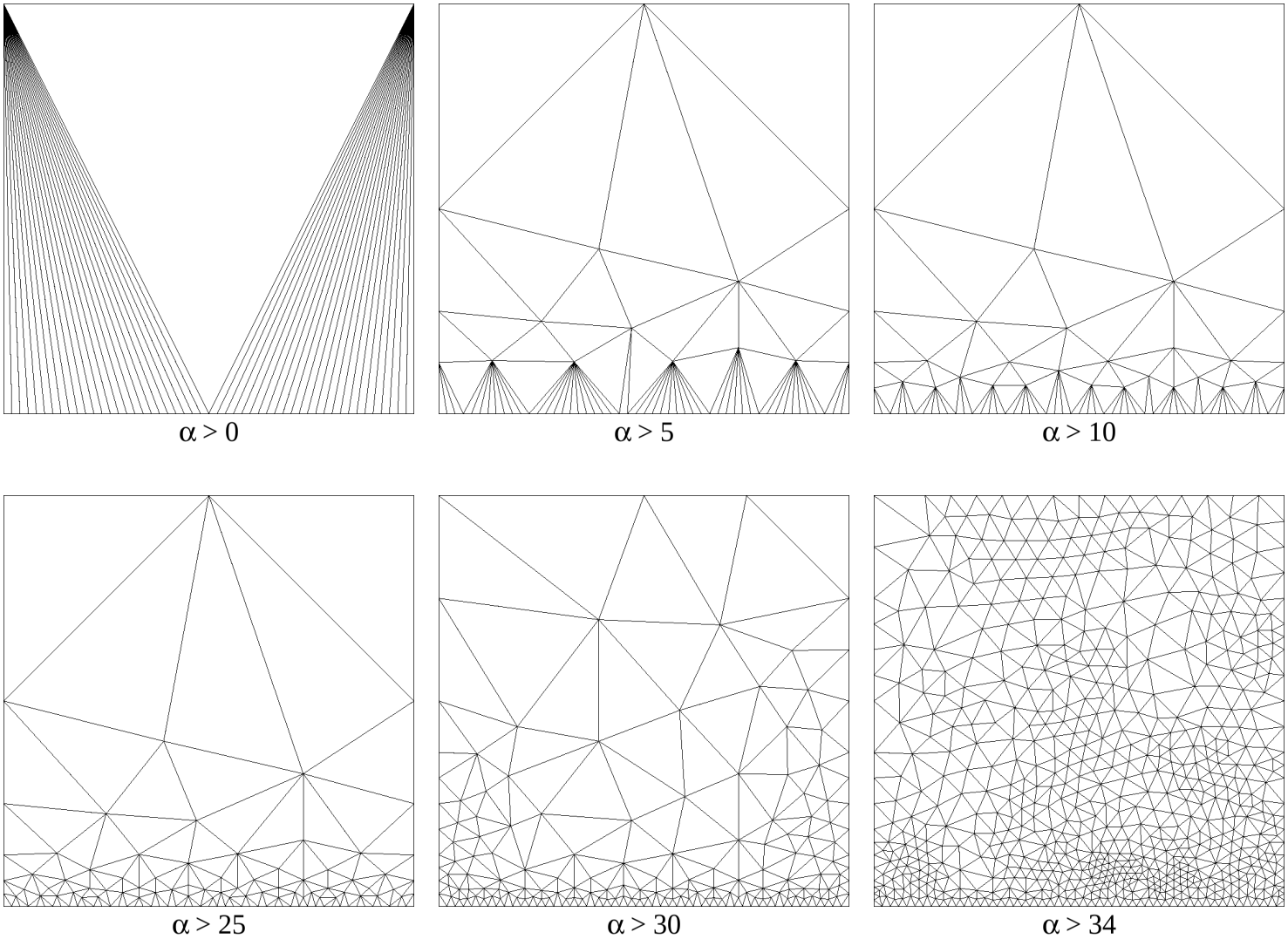


Fig. 72 - Caso 2: Visualização das triangulações obtidas

5.1.3. Caso 3: Quadrado com refino na aresta inferior e restrição de mínimo ângulo interno de elementos

O caso 3 tem o objetivo de verificar se os comportamentos obtidos no caso 2 mantém-se para uma restrição de tamanho de aresta de triângulo 10 vezes menor. A geometria é a mesma do caso 2, mas a restrição de tamanho máximo de aresta de triângulo é de $2x10^{-3}$. Os ângulos mínimos intermos admissíveis utilizados foram 0^o, 10^o, 20^o, 30^o, 32^o e 33^o.

Caso 3						
Ângulo mínimo	0	10	20	30	32	33
Tempo total	121	210	240	351	500	621
Vértices	503	784	1022	2005	3411	4391
Triângulos	501	1048	1522	3473	6265	8199
Arestas	1003	1831	2543	5477	9675	12589
Arestas de contorno	503	518	520	535	555	581

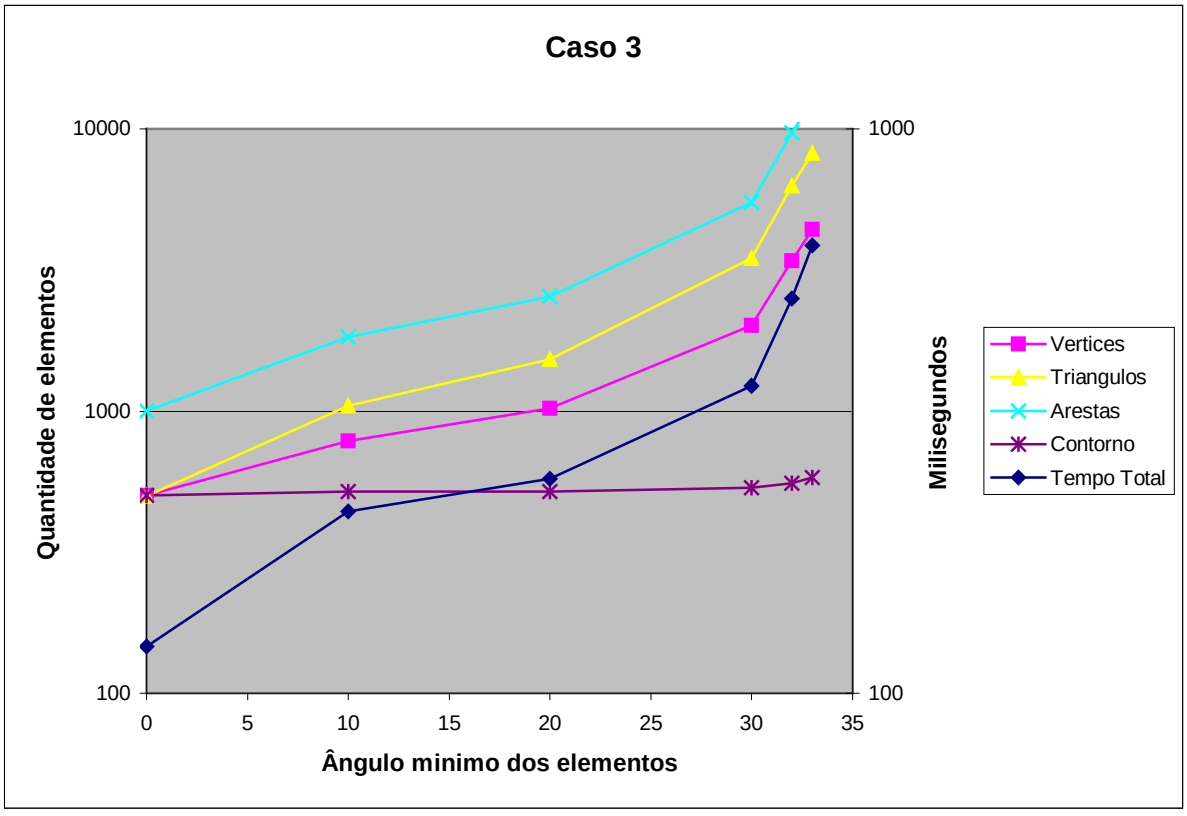


Fig. 73 - Caso 3: Gráfico dos resultados obtidos

Os resultados observados no gráfico da Fig. 73 e nas visualizações da Fig. 74 confirmam um comportamento do triangulador similar ao caso 2, com destaque para o número de arestas na fronteira superior, que manteve-se praticamente igual ao caso 2 (até cerca de 30^o).

Na primeira visualização da Fig. 74 não é possível identificar-se os elementos em função da excessiva concentração.

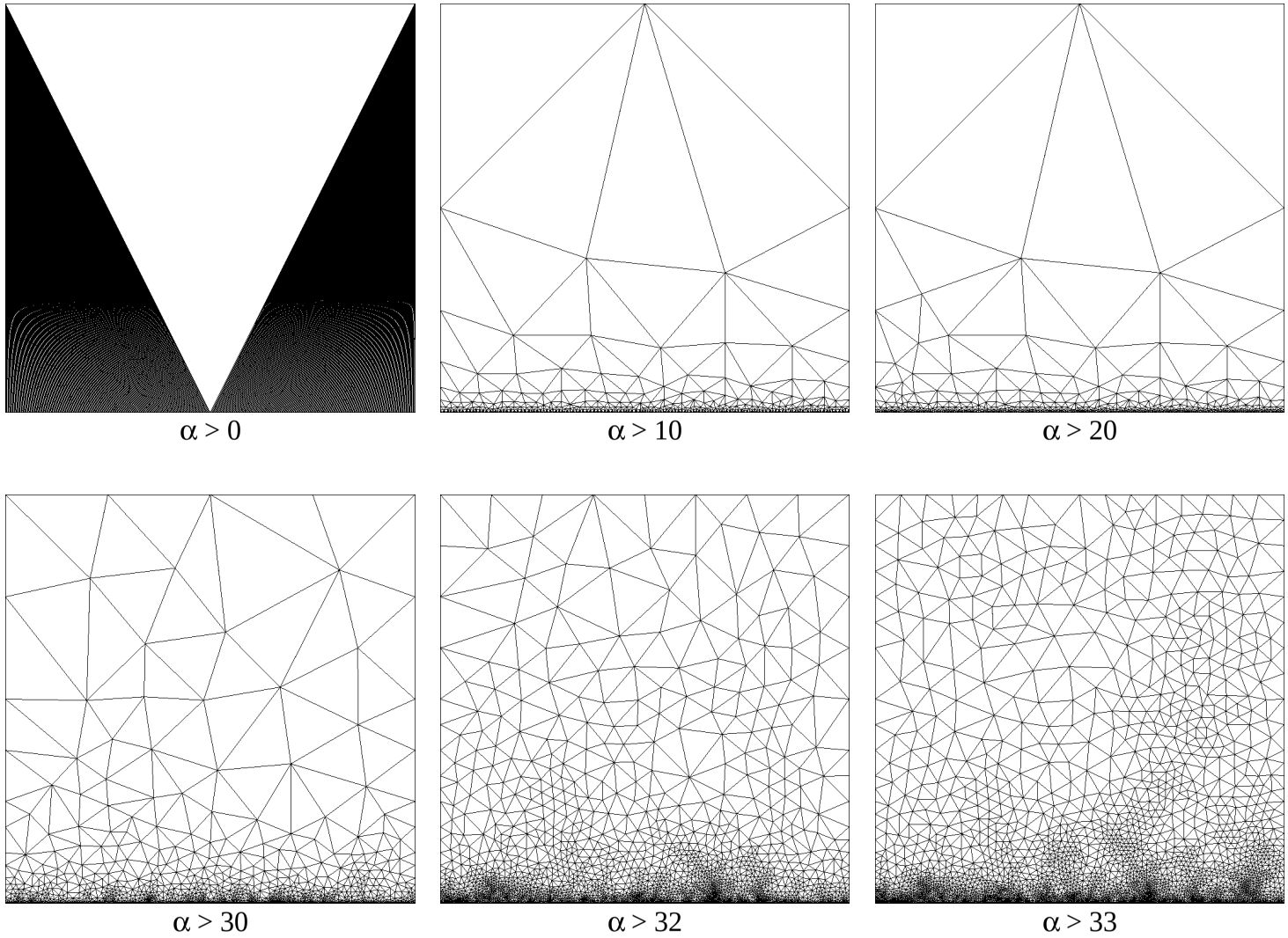


Fig. 74 - Caso 3: visualização das triangulações obtidas

5.1.4. Caso 4: Quadrado com refino na aresta inferior e esquerda e restrição de mínimo ângulo interno de elementos

O caso 4 tem o objetivo de verificar o comportamento observado nos casos 2 e 3 com a imposição de uma restrição de comprimento máximo de aresta dos triângulos na aresta inferior e esquerda da geometria. O comprimento máximo imposto é igual nas duas arestas e de valor $2x10^{-2}$, e os ângulos mínimos internos utilizados para a análise paramétrica foram 0^o, 10^o, 20^o, 30^o, 32^o e 34^o. O número de vértices fornecidos na geometria é 4, e de segmentos 4.

Caso 4						
Ângulo mínimo	0	10	20	30	32	34
Tempo total	52	58	66	108	121	230
Vértices	102	148	191	377	483	980
Triângulos	100	184	268	637	842	1810
Arestas	201	331	458	1013	1324	2789
Arestas de contorno	102	110	112	115	122	148

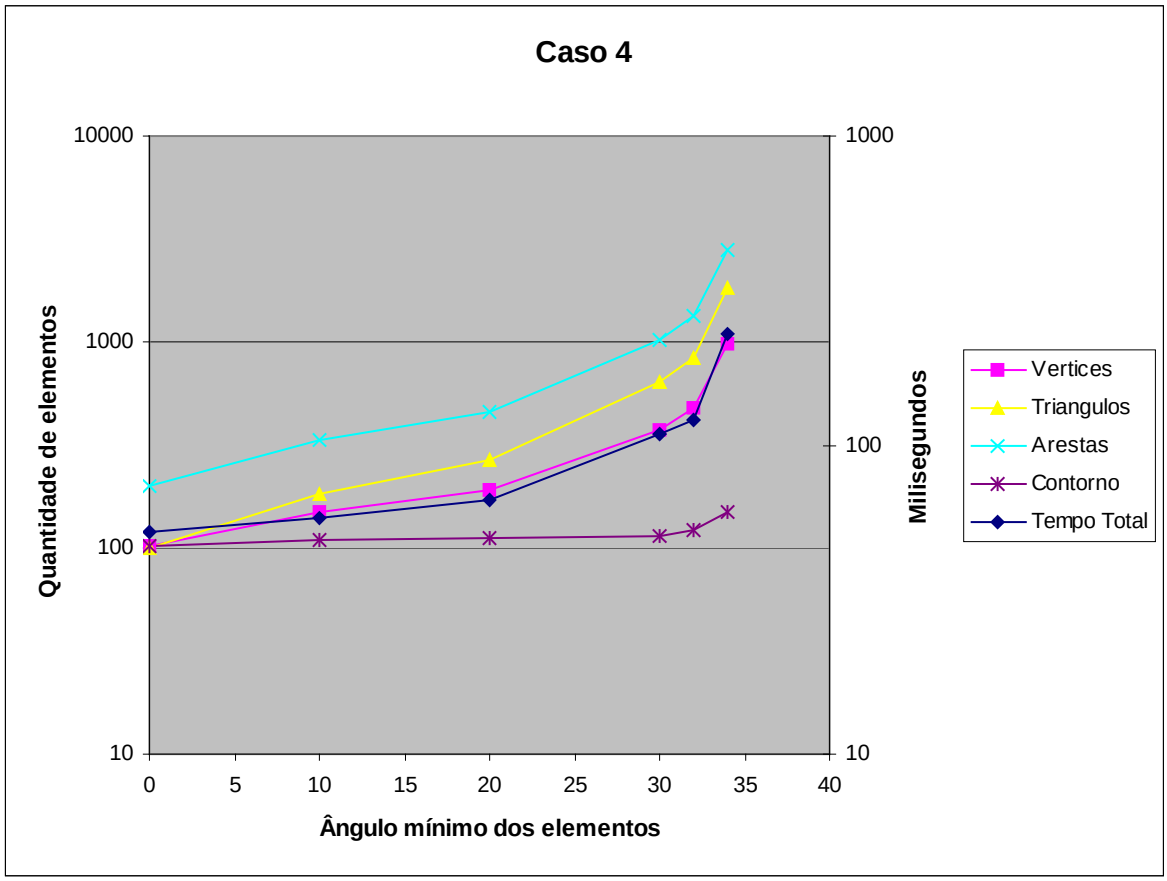


Fig. 75 - Caso 4: Gráfico dos resultados obtidos

Os resultados observados no gráfico da Fig. 75 e nas visualizações da Fig. 76 confirmam um comportamento do triangulador similar aos casos 2 e 3.

Como agora as restrições de comprimento máximo de aresta dos triângulos foi feita para as arestas lateral esquerda e inferior, ambas passam a ser refinadas.

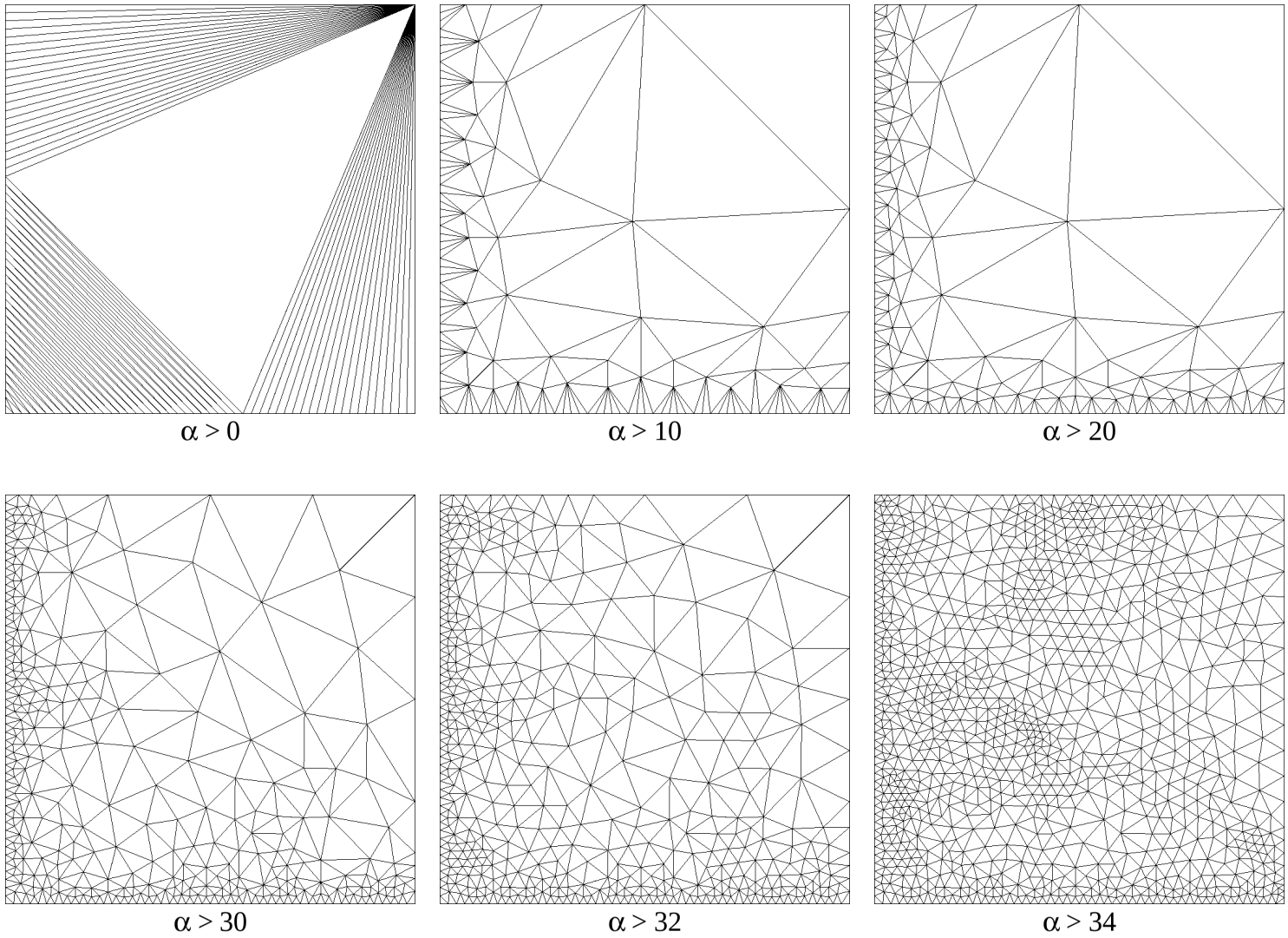


Fig. 76 - Caso 4: visualização das triangulações obtidas

5.1.5. Caso 5: Quadrado com restrição geométrica interna

O caso 5 tem o objetivo de analisar o comportamento do gerador quando imposta uma restrição geométrica interna no domínio (linha grossa nas visualizações da Fig. 78). A geometria é um quadrado com lado de tamanho 1 e com uma aresta de tamanho 0,5 posicionada de forma inclinada no interior deste quadrado. Foi efetuada uma análise paramétrica com quatro condições diferentes de mínimo ângulo interno admitido: 0^o, 10^o, 20^o e 30^o. Além destas, foi gerada uma triangulação na mesma geometria com uma área máxima de elemento admitida de 10^{-2}. O número de vértices fornecido é 6, e de segmentos 5.

Caso5						
Angulo mínimo	0	10	20	30	Área máxima	0,01
Tempo total	77	80	84	90	Tempo total	120
Vértices	6	9	10	18	Vértices	91
Triângulos	6	9	11	22	Triângulos	148
Arestas	11	17	20	39	Arestas	238
Contorno	4	7	7	14	Contorno	32

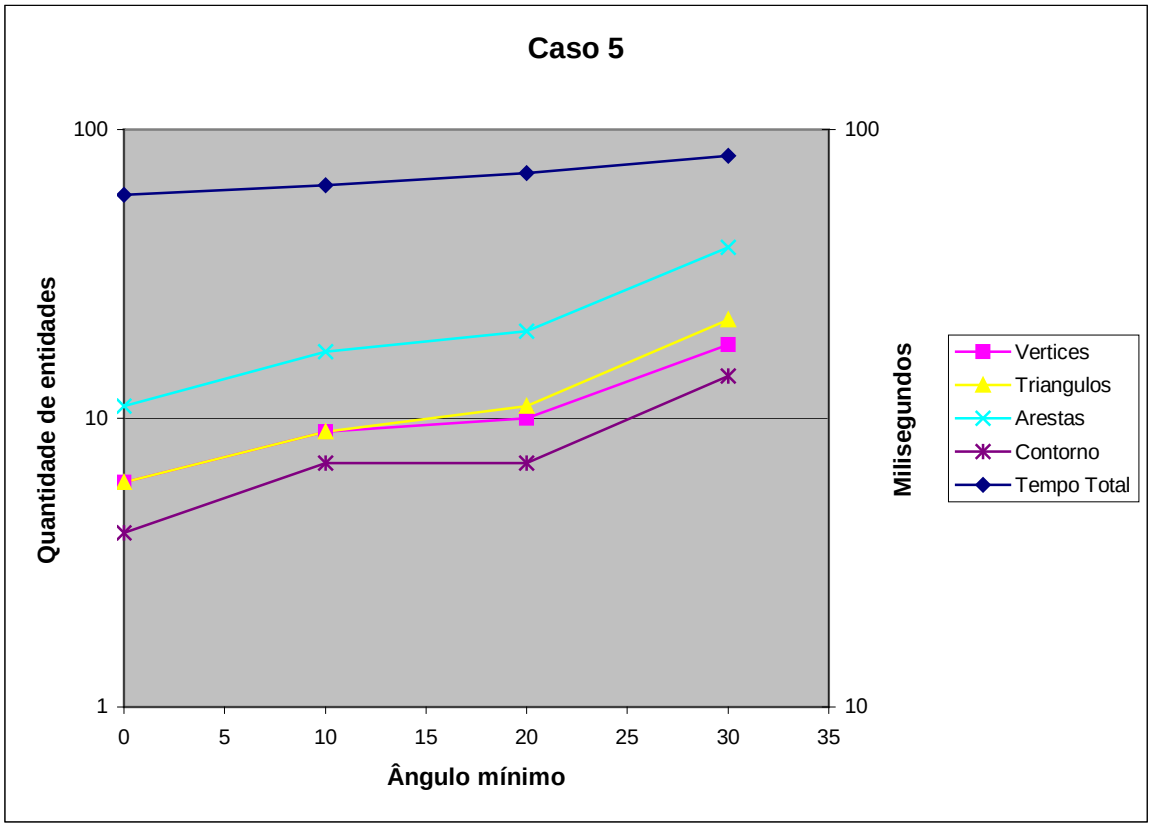


Fig. 77 - Caso 5: Gráfico dos resultados obtidos

Os resultados do caso 5 apresentaram a capacidade do triangulador respeitar ângulos internos mínimos de valores elevados (acima de 20^o) com uma quantidade muito pequena de elementos (22 elementos para o caso de 30^o). Isto indica uma dependência do número de triângulos para com o mínimo ângulo interno apenas quando os existem tamanhos de aresta na geometria muito diferentes. Como o caso apresentado tem tamanhos de arestas entre 0,5 e 1,0, o número de triângulo utilizado foi sempre pequeno.

A triangulação obtida quando imposta a restrição de área máxima de elemento mostra a flexibilidade do triangulador de ajustar-se simultaneamente à este critério e as restrições geométricas. Apesar do número de elementos utilizador ser consideravelmente maior que o esperado (foram utilizador 148 triângulos quando o esperado seria exatamente 100), o triangulador mostrou sua eficiência não precisando executar nenhum tipo de refino local em torno da restrição geométrica para conseguir adequar-se a ela.

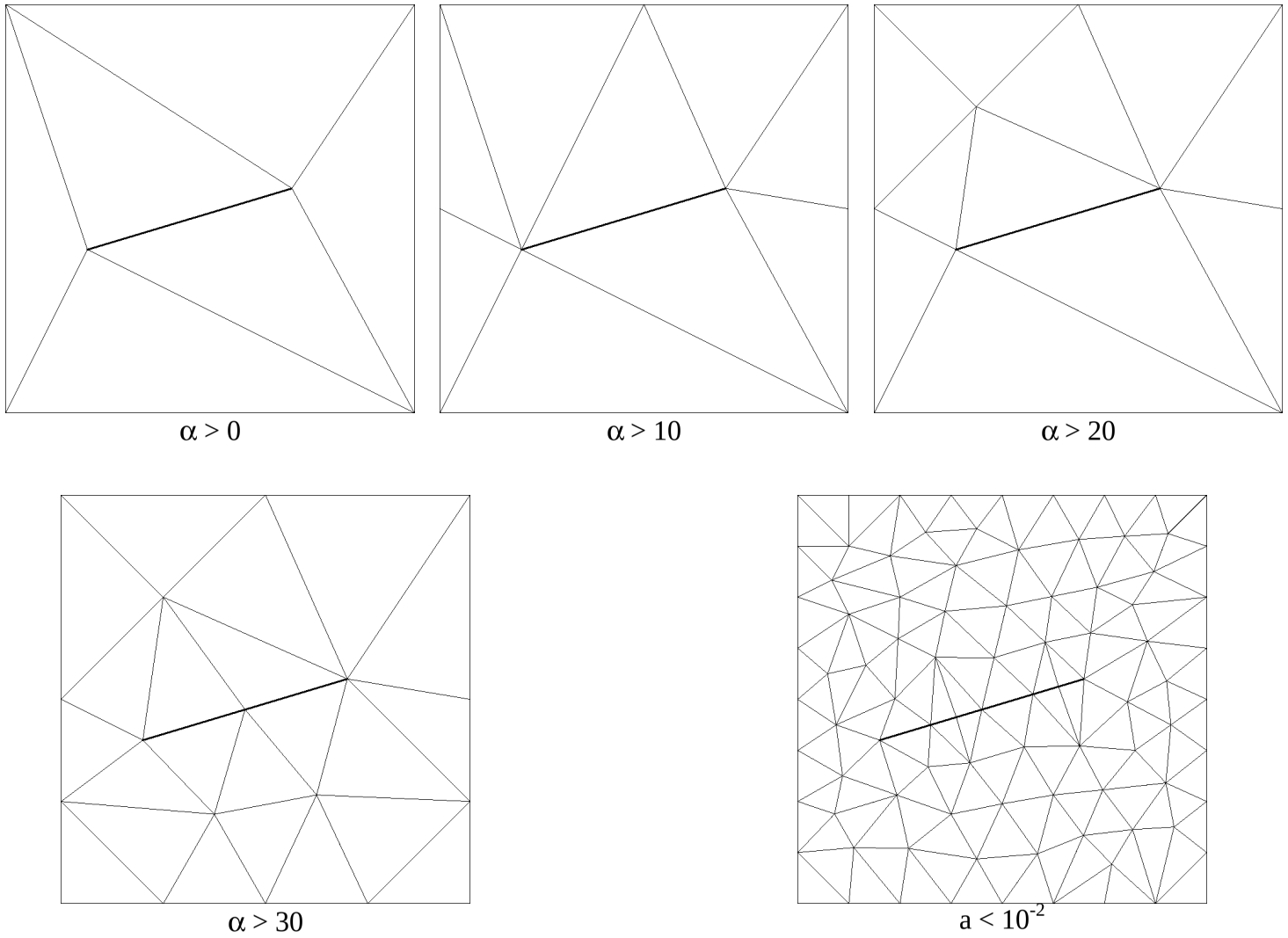


Fig. 78 - Caso 5: visualização das triangulações obtidas

5.1.6. Caso 6: Quadrado com aresta interna e refino em na aresta

O caso 6 tem o objetivo de analisar o comportamento do gerador apresentado nos casos 2 e 3, quando utilizada uma geometria similar a do caso 5. Enquanto que no caso 2 e 3 era imposto um refinamento na fronteira, no presente caso temos um refinamento imposto no interior do domínio. A geometria utilizada é exatamente igual ao caso 5, mas com uma restrição de comprimento máximo de aresta da triangulação de $2x10^{-2}$. Foram impostas cinco condições de mínimo ângulo interno admitido (0^o, 10^o, 20^o, 30^o, 32^o e 34^o). O número de vértices e segmentos fornecidos são 6 e 5, respectivamente.

Caso 6						
Ângulo mínimo	0	10	20	30	32	34
Tempo total	44	56	68	114	159	854
Vértices	55	118	180	383	628	3966
Triângulos	104	220	343	738	1211	7784
Arestas	158	337	522	1120	1838	11749
Arestas de contorno	4	14	15	26	43	146

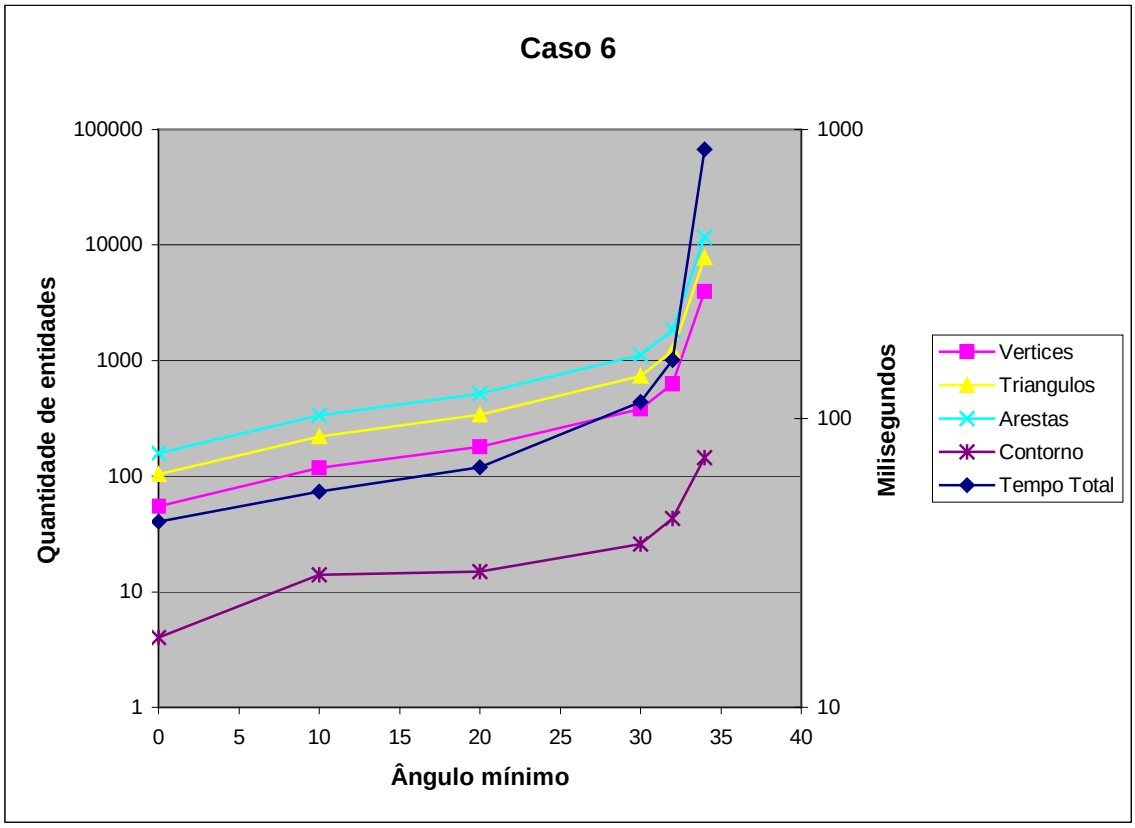


Fig. 79 - Caso 6: Gráfico dos resultados obtidos

Os resultados obtidos foram similares aos casos 2 e 3, onde:

- Os elementos adicionados pelo refino para respeitar o critério de mínimo ângulo interno localizaram-se muito próximos a aresta com restrição de comprimento máximo;
- O número de elementos cresce exponencialmente com o ângulo, até ângulos próximos a 30^o, quando então o refino de malha propaga-se para o interior do domínio, adicionando elementos longe dos pontos onde foram impostas as restrições;
- O triangulador não conseguiu gerar uma malha com ângulo interno mínimo de 35^o;
- A influência do refino nas arestas de fronteira é bastante pequena até cerca de 30^o. Isto é interessante para o método de geração 2,5D pois minimiza a influência em domínios vizinhos de restrições impostas em um determinado domínio;

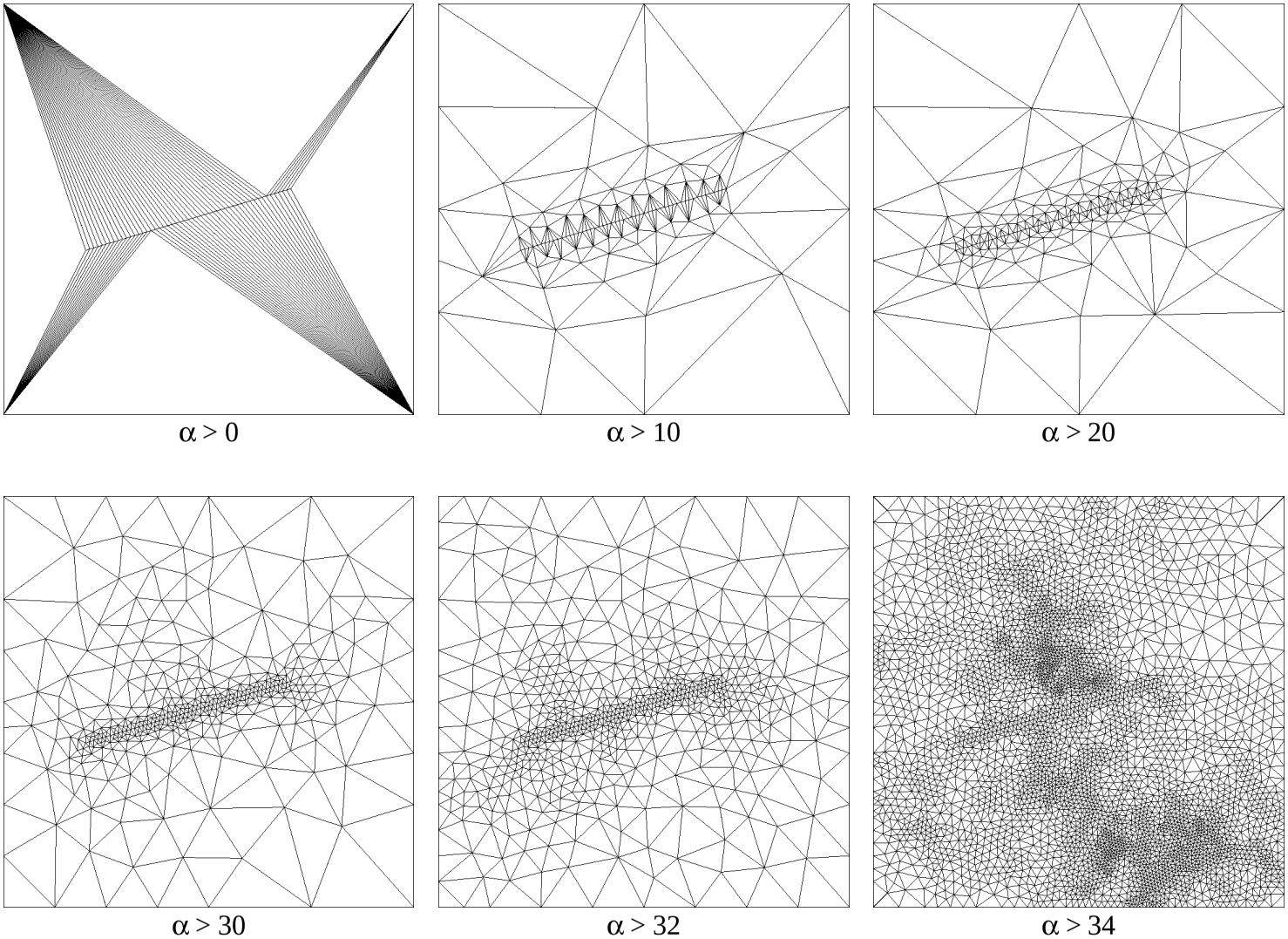


Fig. 80 - Caso 6: visualização das triangulações obtidas

5.1.7. Caso 7: Quadrado com aresta interna e refino em na aresta (caso 6 refinado)

O caso 7 tem o objetivo de verificar o comportamento observado no caso 6, mas com uma restrição de comprimento máximo de aresta dos triângulos 10 vezes maior. O comprimento máximo de aresta admitido é $2x10^{-3}$, e foram utilizados 4 valores de ângulo interno mínimo dos triângulos na análise paramétrica: 0^o, 20^o, 30^o e 32^o. O número de vértices na geometria fornecida é 6 e de segmentos 5.

Caso 7				
Ângulo mínimo	0	20	30	32
Tempo total	146	473	948	1152
Vértices	505	1559	3961	6849
Triângulos	1004	3102	7882	13635
Arestas	1508	4660	11842	20483
Arestas de contorno	4	14	38	61

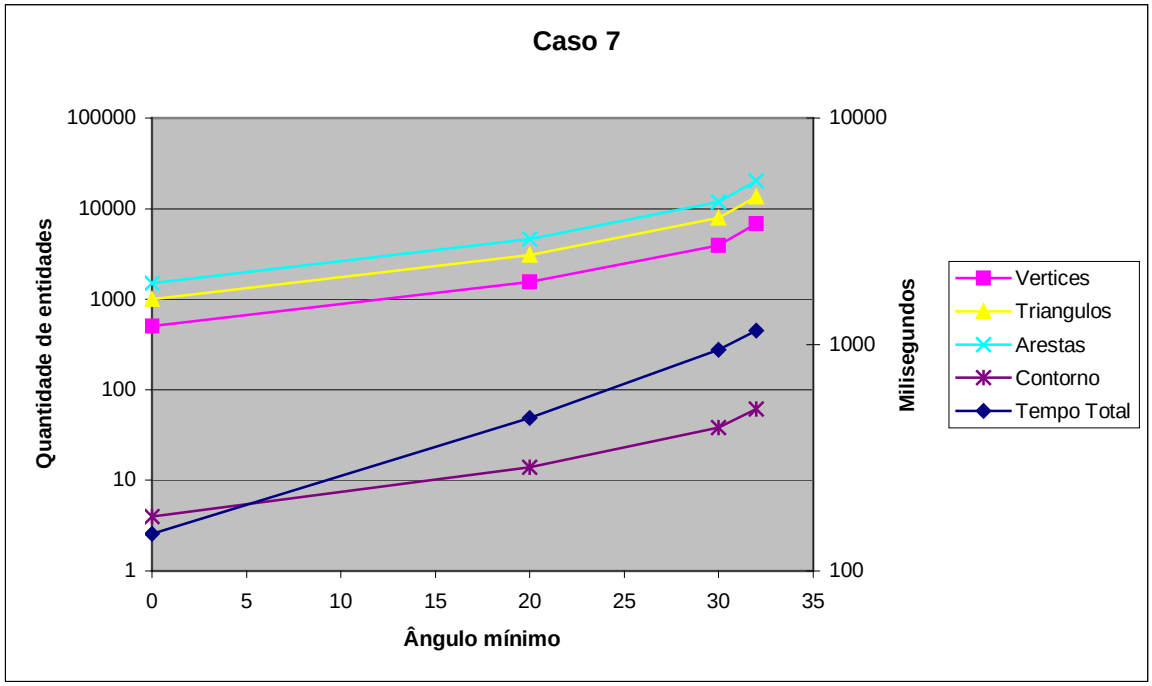
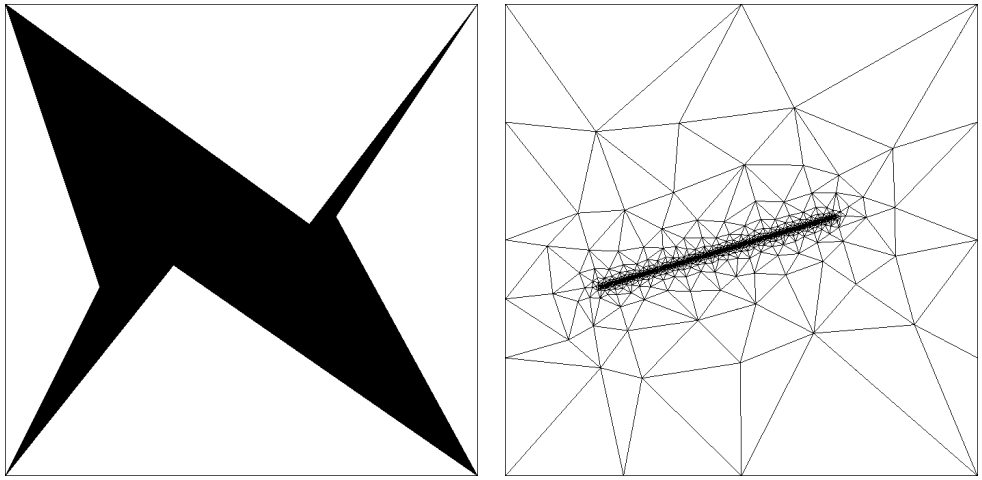
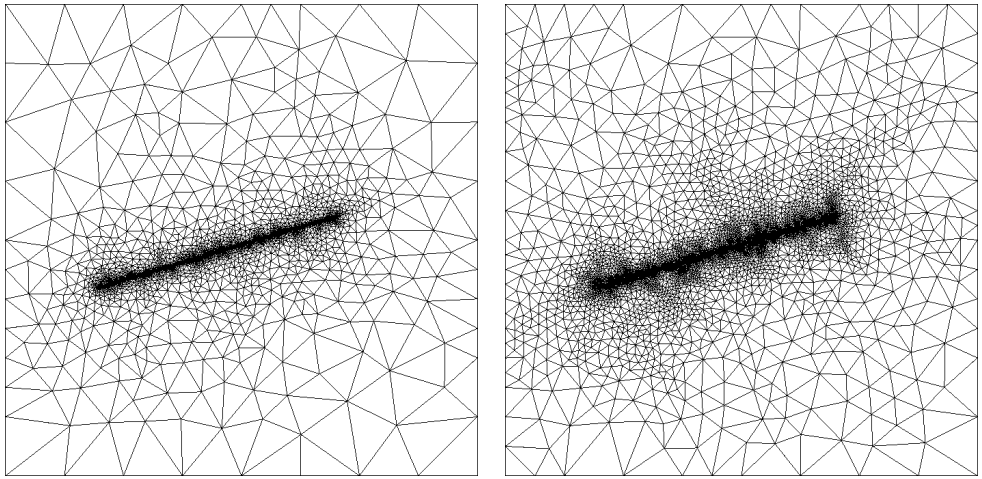


Fig. 81 - Caso 7: Gráfico dos resultados obtidos

Os resultados observados no gráfico da Fig. 81 e nas visualizações da Fig. 82 confirmam um comportamento do triangulador similar ao caso 7, com destaque para o número de arestas na fronteira superior, que manteve-se praticamente igual ao caso 2 (até cerca de 30^o).



α > 0 α > 20



α > 30 α > 32

Fig. 82 - Caso 7: visualização das triangulações obtidas

5.1.8. Caso 8: Geometria qualquer com restrição geométrica interna

O caso 8 tem o objetivo de demonstrar a capacidade do triangulador de conformar-se à geometrias diversas, com múltiplas arestas interna e furos no domínio. A geometria fornecida contém 23 vértices, 22 segmentos e 1 furo, e foram impostas restrições simultâneas de máxima área dos triângulos de $5x10^{-2}$, e mínimo ângulo interno de 30 graus.

Caso 8	
Tempo total	110
Vértices	603
Triângulos	1107
Arestas	1710
Arestas de contorno	135

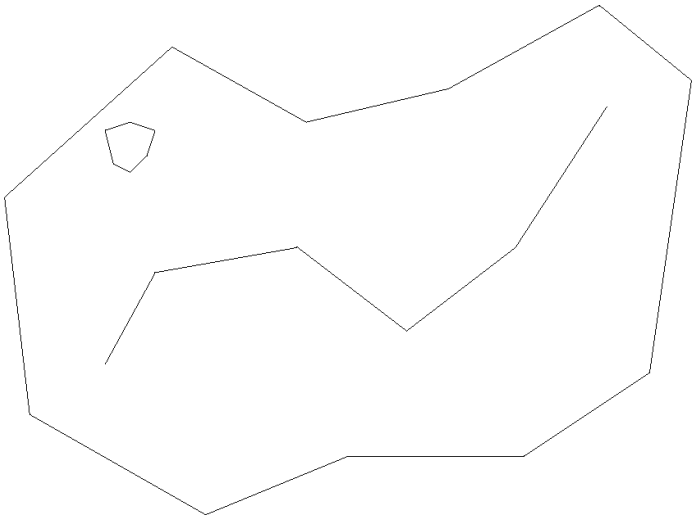


Fig. 83 - Caso 8: geometria do domínio fornecido

O resultado deste caso, mostrado na Fig. 84, apresenta uma triangulação com um tamanho uniforme de elementos e sem a utilização de nenhum tipo de refino local para respeitar-se as restrições geométricas impostas. O tempo de geração pode ser considerado bastante baixo (pouco mais de um décimo de segundo), e a quantidade de elementos utilizada excede por pouco o esperado (1107 contra um ótimo de 900 elementos, dado que a área da figura é aproximadamente 45 unidades de área).

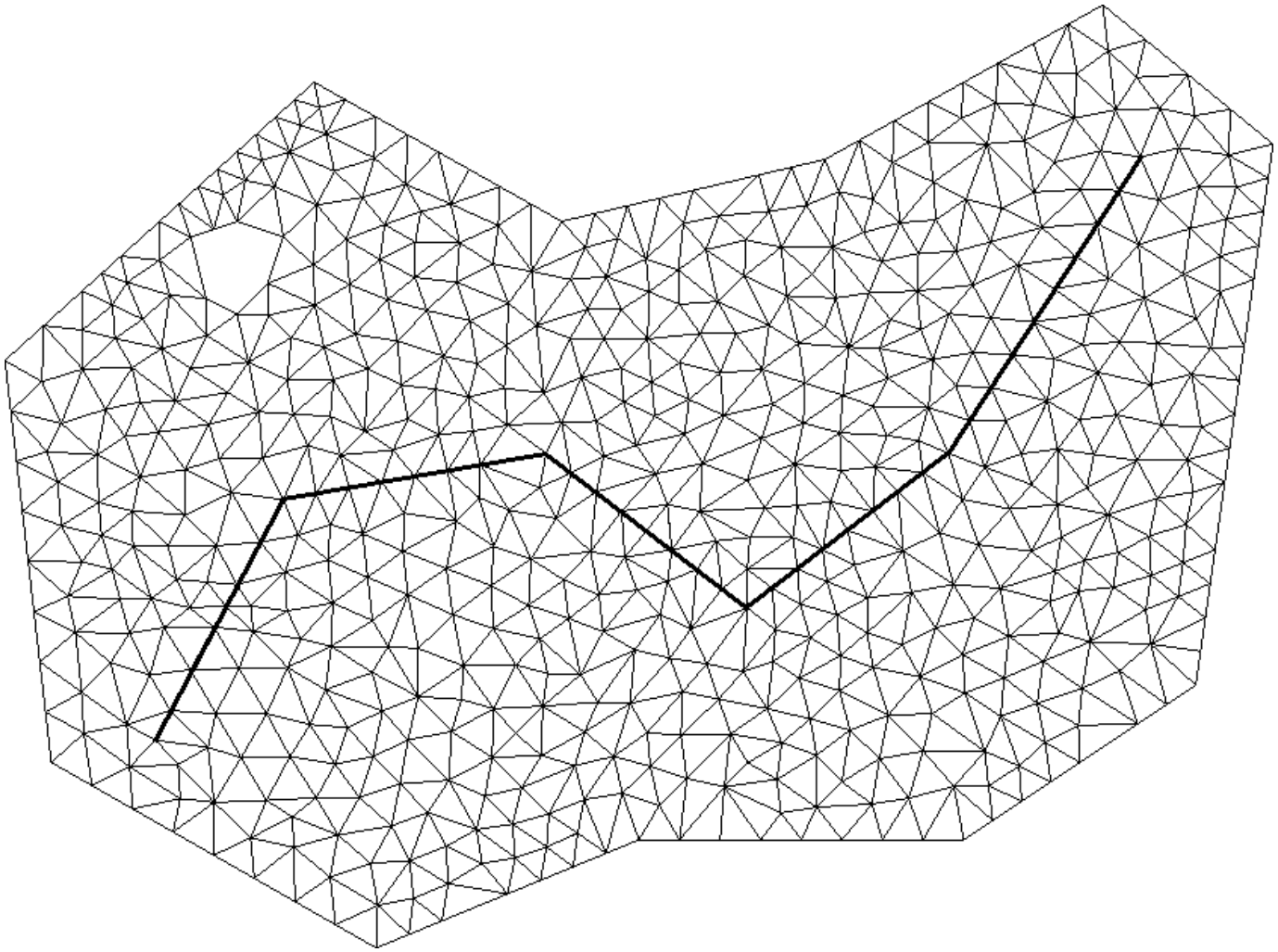


Fig. 84 - Caso 8: visualização da triangulação obtida

5.1.9. Caso 9: Quadrado com restrição pontual de comprimento máximo de triângulos na fronteira e mínimo ângulo interno dos elementos

O caso 9 tem o objetivo de analisar o comportamento do triangulador quando varia-se o comprimento de uma restrição geométrica na fronteira e mantém-se fixo o mínimo ângulo interno admissível dos elementos. Estes tipo de restrição geométrica serve para a aplicação de condições de contorno em comprimentos muito pequenos nas fronteiras do domínios. Um exemplo prático da aplicação deste tipo de condição de contorno é a injeção de um fluido em um domínio através de um pequeno orifício ou a geração localizada de calor. A geometria do presente caso é um quadrado de lado 1, com a restrição geométrica centrada na aresta esquerda. Os comprimentos utilizados na análise paramétrica foram $2x10^{-1}$, $2x10^{-2}$, $2x10^{-3}$, $2x10^{-4}$ e $2x10^{-5}$, e a restrição de mínimo ângulo interno dos elementos foi fixada em 32 graus. O número de vértices da geometria fornecida é 6, e o de segmentos 6.

Caso 9					
Inverso da aresta	5	50	500	5000	50000
Tempo total	19	39	80	169	180
Vértices	17	54	231	487	649
Triângulos	21	87	411	897	1207
Arestas	37	140	641	1383	1855
Arestas de contorno	11	19	49	75	89

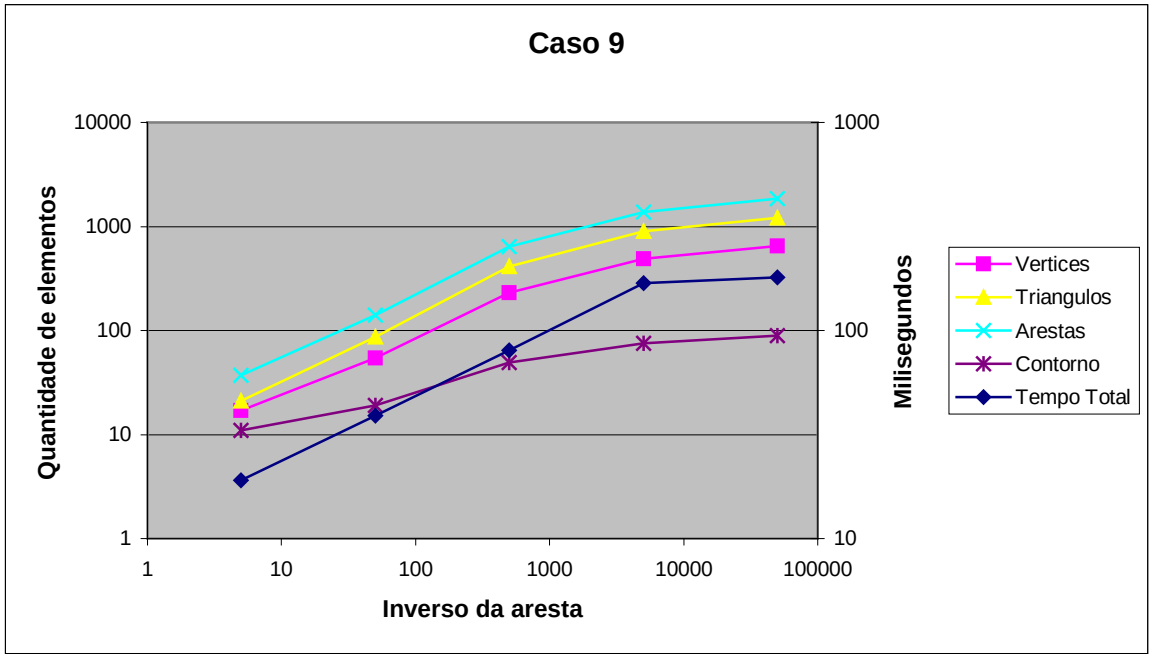
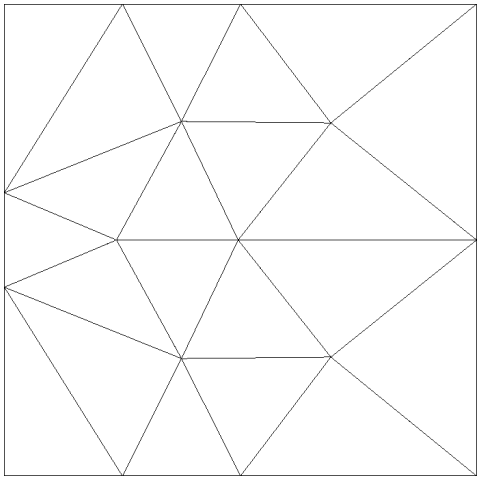
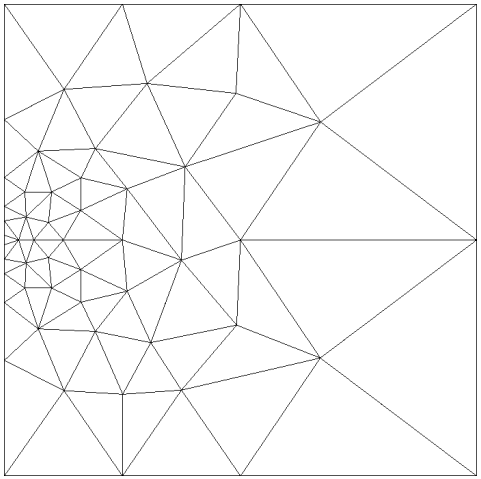


Fig. 85 - Caso 9: Gráfico dos resultados obtidos

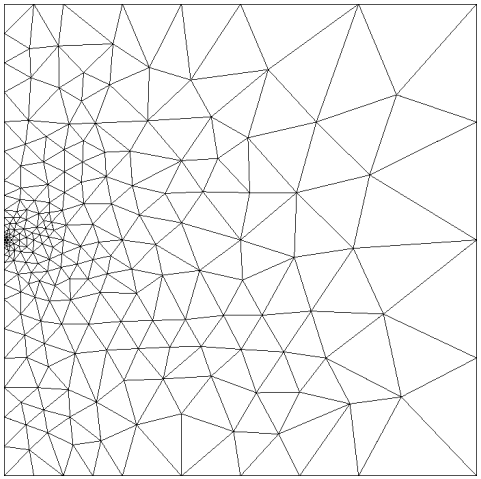
Os resultados obtidos apresentaram uma variação dos parâmetros de saída do triangulador linear com relação ao inverso do comprimento da aresta. Novamente, de forma positiva, a influência do refino concentrou-se praticamente apenas na região da restrição. É importante lembrar que todos os resultados consideraram um ângulo mínimo interno dos elementos de 32^{o}, o que é um valor quase crítico conforme os resultados anteriores.



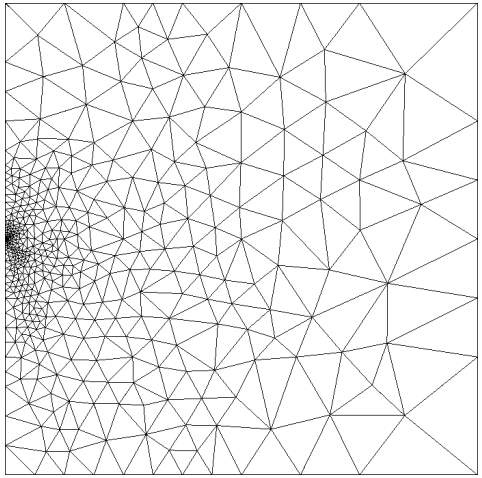
$d = 2x10^{-1}$



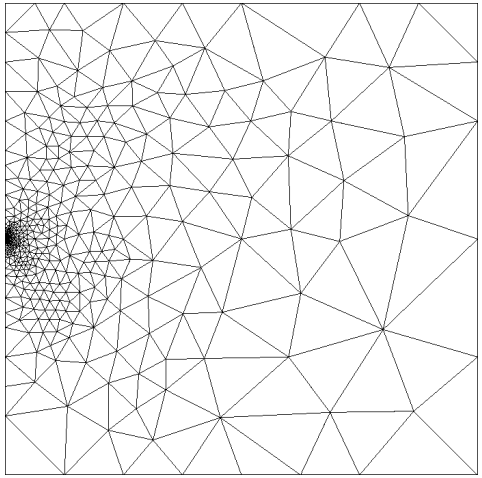
$d = 2x10^{-2}$



$d = 2x10^{-3}$



$d = 2x10^{-4}$



$d = 2x10^{-1=5}$

Fig. 86 - Caso 9: visualização das triangulações obtidas

5.1.10. Caso 10: Quadrado com restrição pontual de comprimento máximo de triângulos no interior do domínio e mínimo ângulo interno dos elementos

O caso 10 tem o objetivo de verificar se o comportamento apresentado no caso 9, onde a restrição geométrica é na fronteira, permanece quando deslocamos tal restrição para o interior do domínio. A geometria utilizada é um quadrado de lado 1, com uma restrição geométrica inclinada localizada no interior de tal quadrado. Foram utilizados comprimentos de aresta de $2x10^{-1}$, $2x10^{-2}$, $2x10^{-3}$, $2x10^{-4}$ e $2x10^{-5}$ na análise paramétrica, e restrição de mínimo ângulo interno dos elementos fixa de 32 graus. A geometria fornecida possui 6 vértices e 5 segmentos.

Caso 10					
Inverso da aresta	5	50	500	5000	50000
Tempo total	17	52	166	325	355
Vértices	12	111	615	1293	1524
Triângulos	14	195	1184	2533	3004
Arestas	25	305	1798	3825	4527
Arestas de contorno	8	25	44	51	42

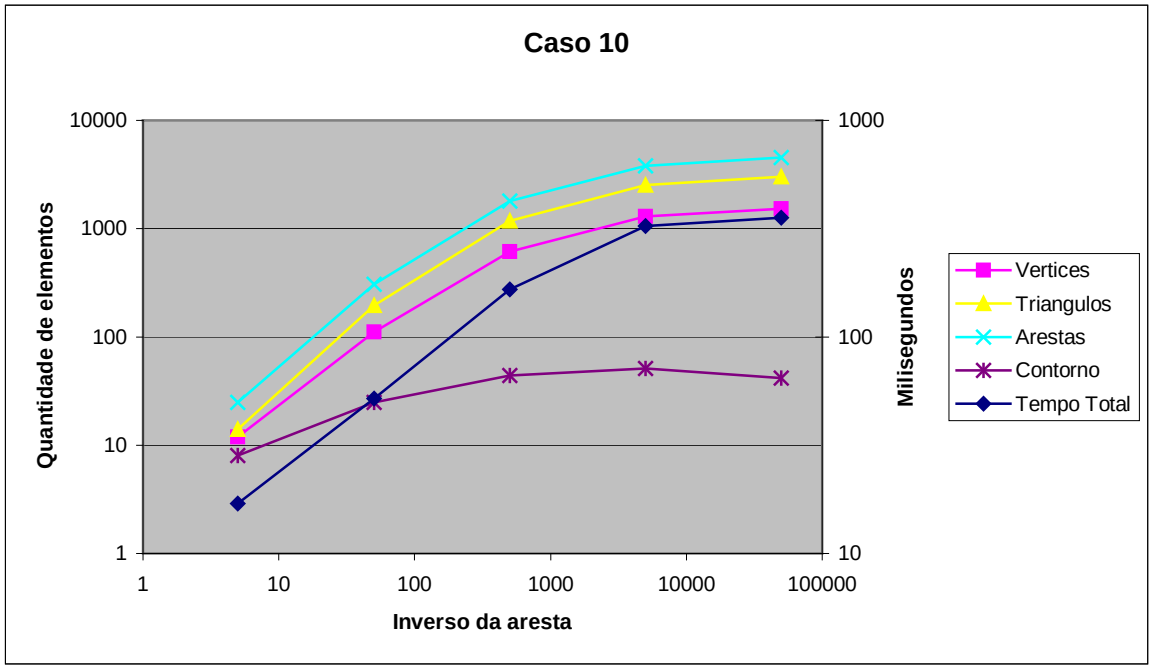
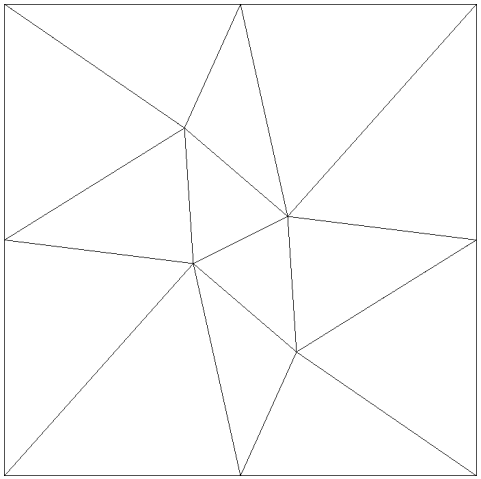
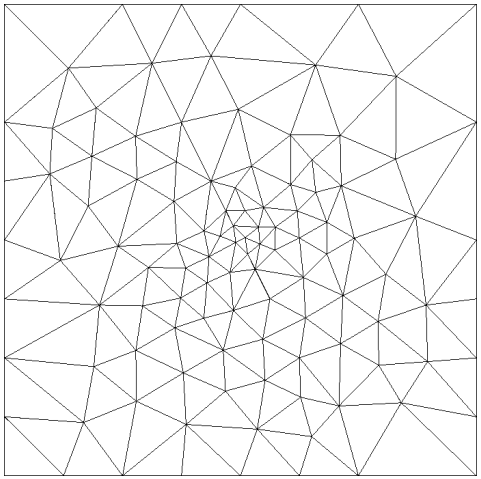


Fig. 87 - Caso 10: Gráfico dos resultados obtidos

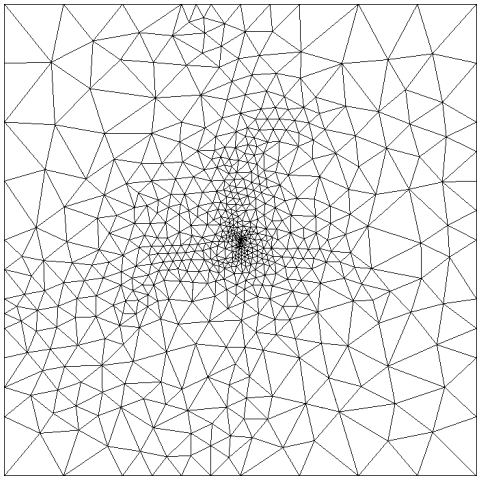
Os resultados observados no gráfico da Fig. 87 e nas visualizações da Fig. 88 confirmam um comportamento do triangulador similar ao caso 9.



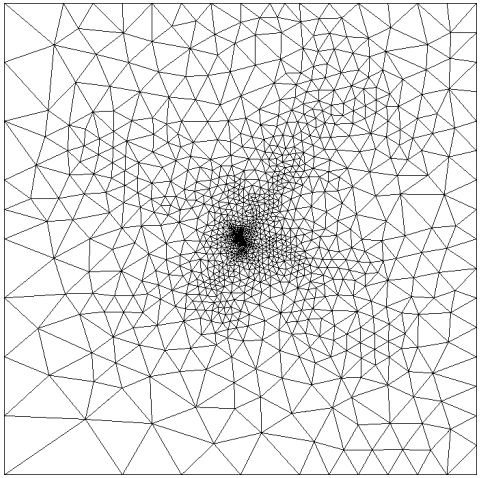
$d = 2x10^{-1}$



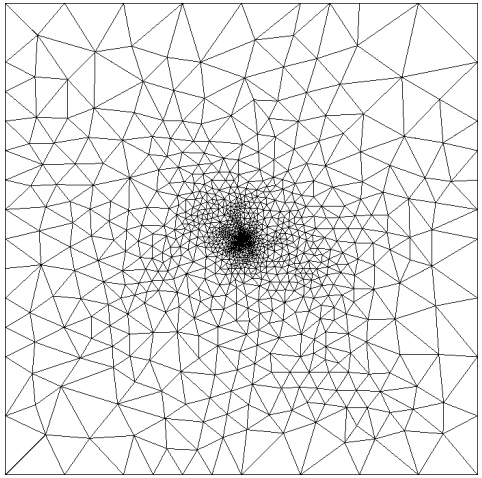
$d = 2x10^{-2}$



$d = 2x10^{-3}$



$d = 2x10^{-4}$



$d = 2x10^{-5}$

Fig. 88 - Caso 10: visualização das triangulações obtidas

5.1.11. Caso 11: Aproximação de fronteira curva por segmentos de reta

O caso 11 tem o objetivo de analisar o comportamento do gerador para diferentes níveis de aproximação de uma fronteira curva do domínio por um conjunto de segmentos de reta. A geometria é um quadrado com lado de tamanho 1 e com o canto superior esquerdo formado por um arco de circunferência de raio 0,3. Foram considerados cinco diferentes níveis de aproximação da curva na análise paramétrica, utilizando 5, 10, 50, 100 e 500 segmentos de reta, e imposta uma restrição de mínimo ângulo interno dos elementos de 20^{o}. O número de vértices do domínio geométrico fornecido é 4 mais a quantidade de vértices utilizadas na aproximação do arco de circunferência.

Caso 11					
Numero de arestas	5	10	50	100	500
Tempo total	26	32	52	67	317
Vértices	19	31	114	214	1037
Triângulos	23	40	164	310	1552
Arestas	41	70	277	523	2588
Arestas de contorno	13	20	62	116	520

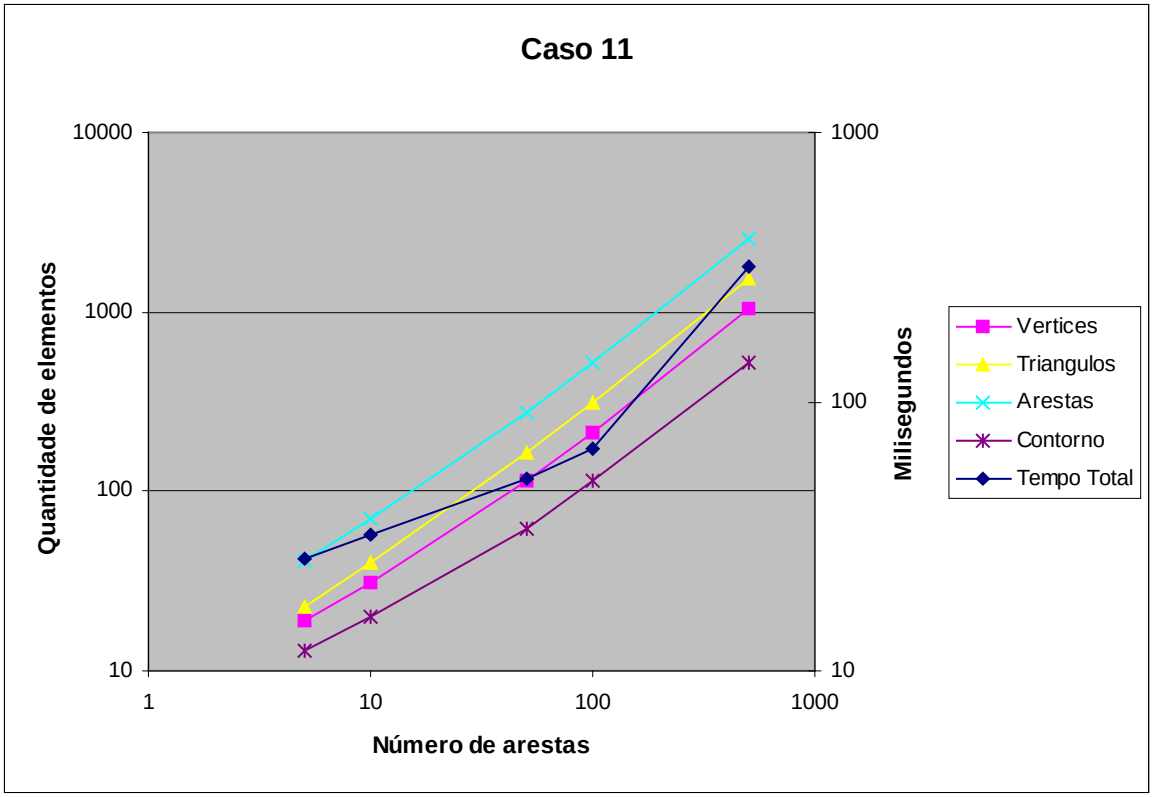


Fig. 89 - Caso 11: Gráfico dos resultados obtidos

Os resultados obtidos para este caso são compatíveis com os resultados apresentados nos casos anteriores. A quantidade de elementos utilizada na triangulação apresentou-se linear com relação ao número de segmentos de reta utilizados na aproximação, e o refino de malha para respeitar o critério de mínimo ângulo interno dos triângulo localizou pontos basicamente próximos à fronteira curva.

Um aspecto interessante de ser observado, no entanto, foi com relação ao tempo de CPU consumido. Este apresentou um comportamento não-linear, diferentemente do caso 2, por exemplo. Esta resposta deve-se ao posicionamento dos pontos de fronteira de forma cocircular. Este pontos definem uma triangulação inicial degenerada, a qual o triangulador consome mais tempo para efetuar os cálculos. Passada a etapa de obtenção da triangulação inicial, o método de refino adiciona novos pontos para respeitar o critério de mínimo ângulo interno, e com isso eliminar tal degeneração.

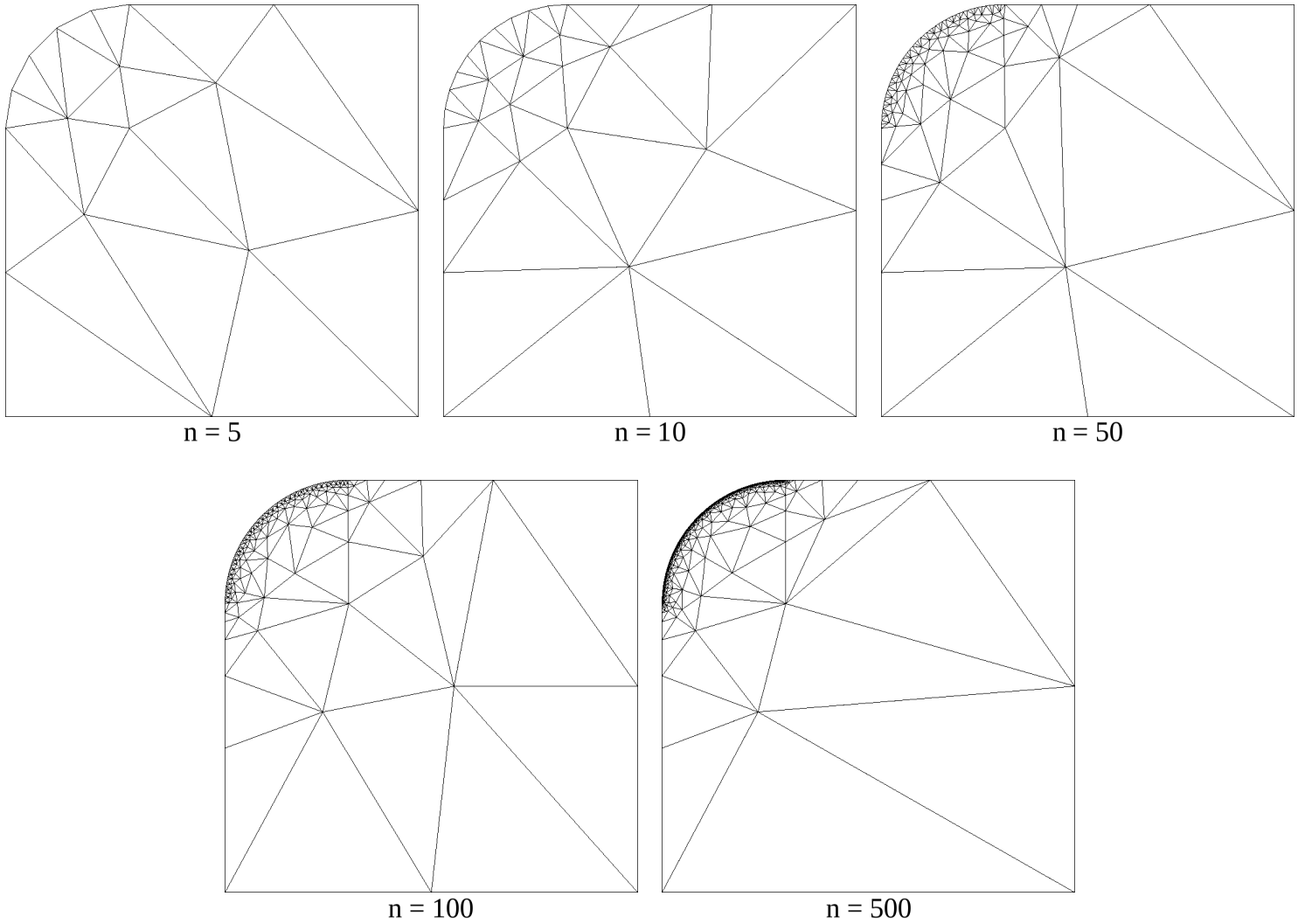


Fig. 90 - Caso 11: visualização das triangulações obtidas

5.1.12. Caso 12: Quadrado com furo central

O caso 12 tem o objetivo de verificar o comportamento observado nos casos 2, 3 e 11, mas com a aproximação de uma fronteira curva no interior do domínio. A geometria é um quadrado com lado de tamanho 1 e um furo central aproximado por 20 segmentos de reta. Para a análise paramétrica foram impostas restrições de mínimo ângulo interno com valores variando em 0^o, 10^o, 20^o, 30^o, 32^o e 34^o. O número de vértices na geometria fornecida é 24, e de segmentos 24.

Caso 12						
Numero de arestas	0	10	20	30	32	34
Tempo total	76	84	90	97	102	108
Vértices	24	40	60	94	98	107
Triângulos	24	44	84	138	146	164
Arestas	48	84	144	232	244	271
Arestas de contorno	24	36	36	50	50	50

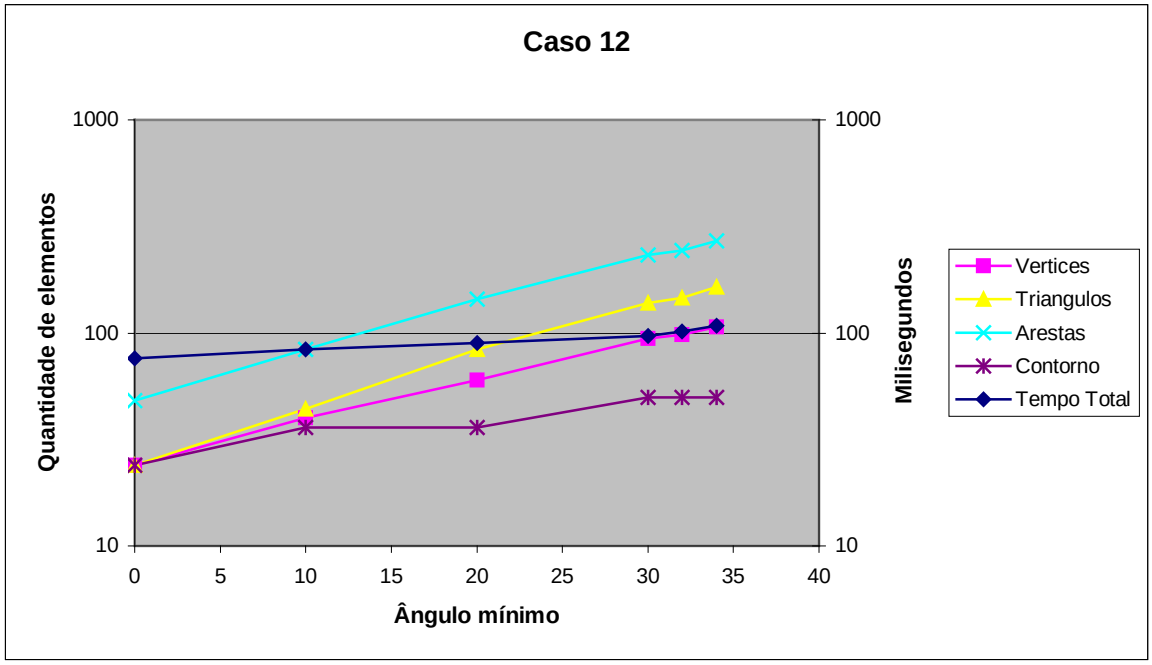


Fig. 91 - Caso 12: Gráfico dos resultados obtidos

Os resultados apresentados no gráfico da Fig. 91 e nas visualizações da Fig. 92 confirmam o comportamento do caso 12 similar aos casos 2, 3 e 11, com destaque para a pequena quantidade de elementos utilizada, devido à similaridade no comprimento das arestas da geometria fornecida.

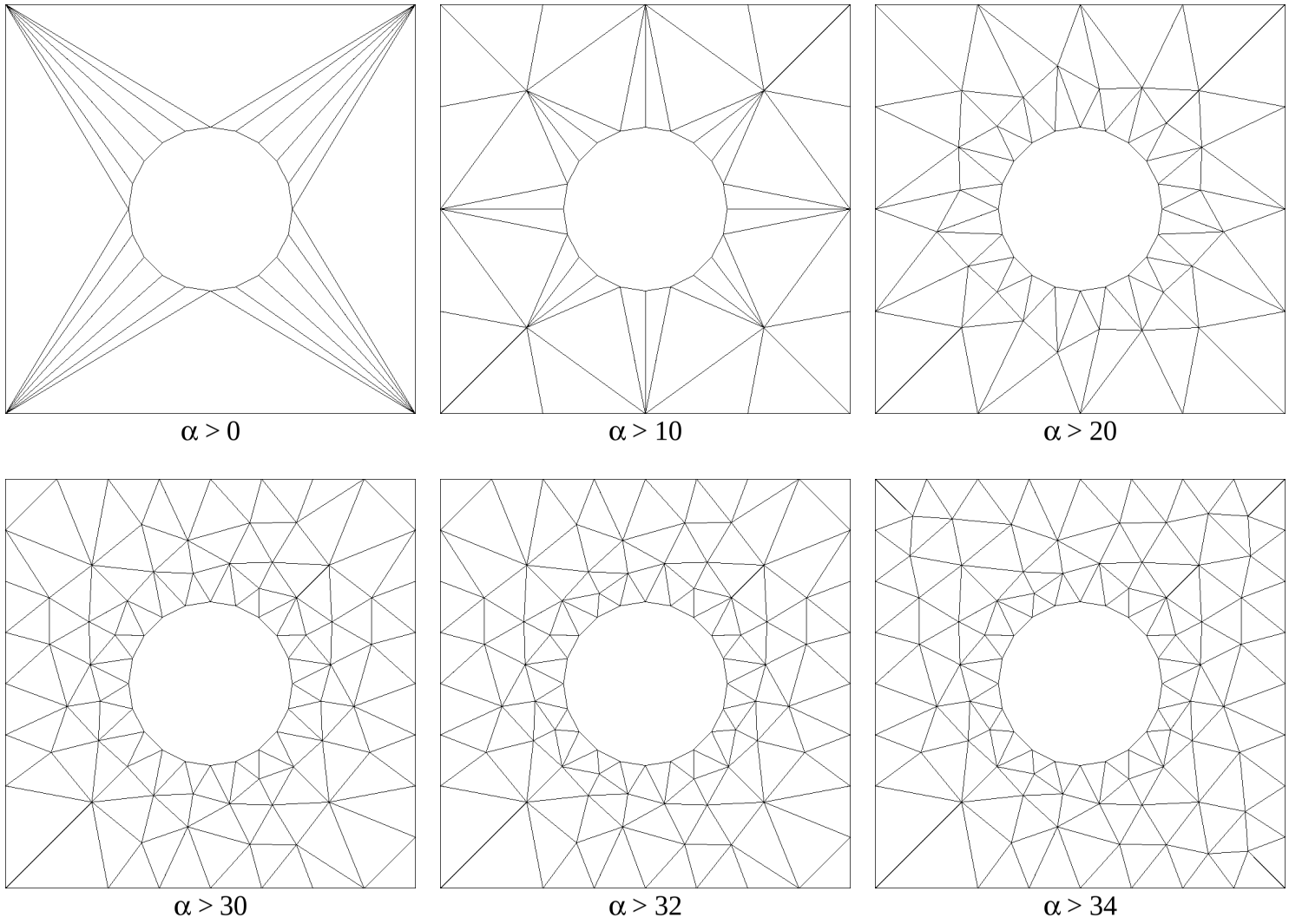


Fig. 92 - Caso 12: visualização das triangulações obtidas

5.1.13. Caso 13: Retângulo com furos internos e refino vertical

O presente tem o objetivo analisar o comportamento do gerador quando variando o número de furos internos na geometria. A geometria utilizada é um retângulo de altura 1, e comprimento 1, 1.7 e 2.4 para um, dois e três furos, respectivamente. A aproximação dos furos internos foram feitas utilizando 100 arestas em cada, e no refino vertical foram impostos uma condição mínima de 30 elementos. O mínimo ângulo interno admitido é de 30^{o}. A geometria fornecida possui, respectivamente, 104, 204 e 304 vértices, e 104, 204 e 304 segmentos.

Caso 13			
Número de furos	1	2	3
Tempo total	160	250	331
Vértices	913	1754	2398
Triângulos	1669	3226	4382
Arestas	2582	4981	6781
Arestas de contorno	157	284	418

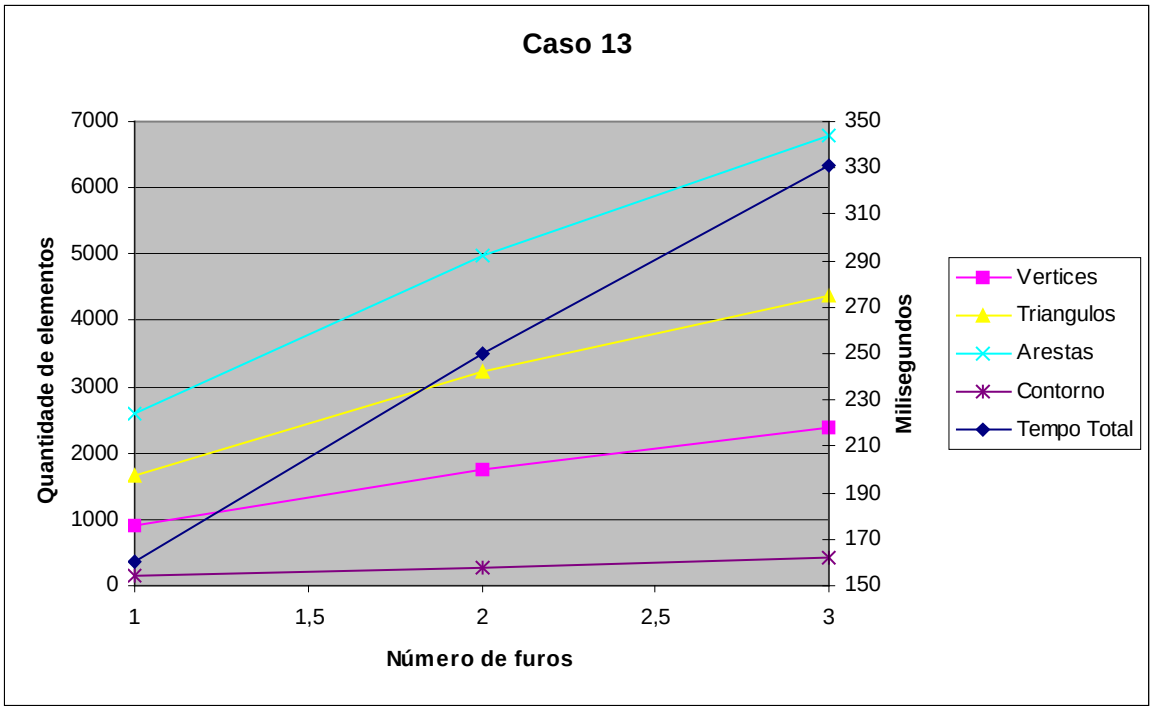
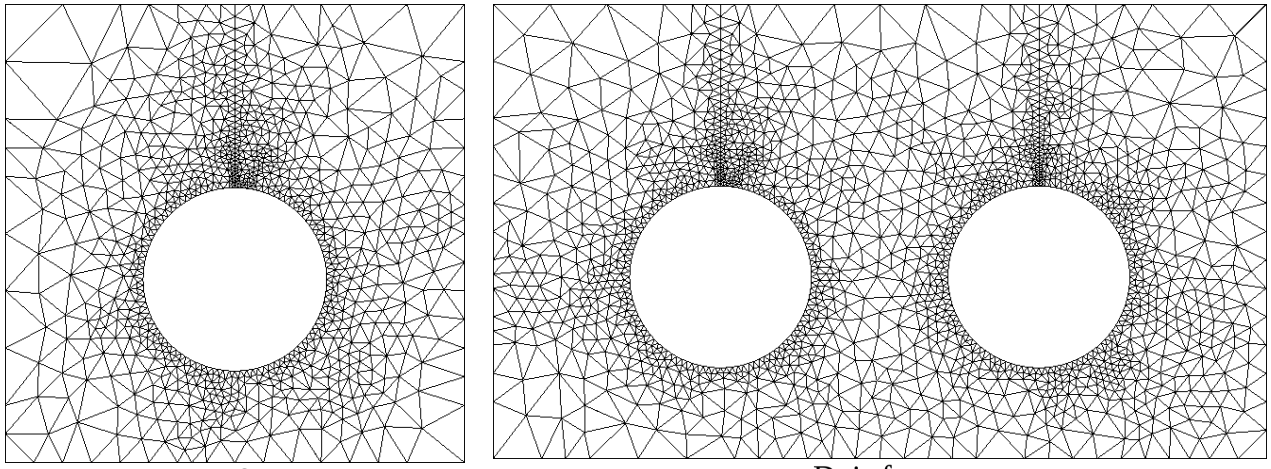
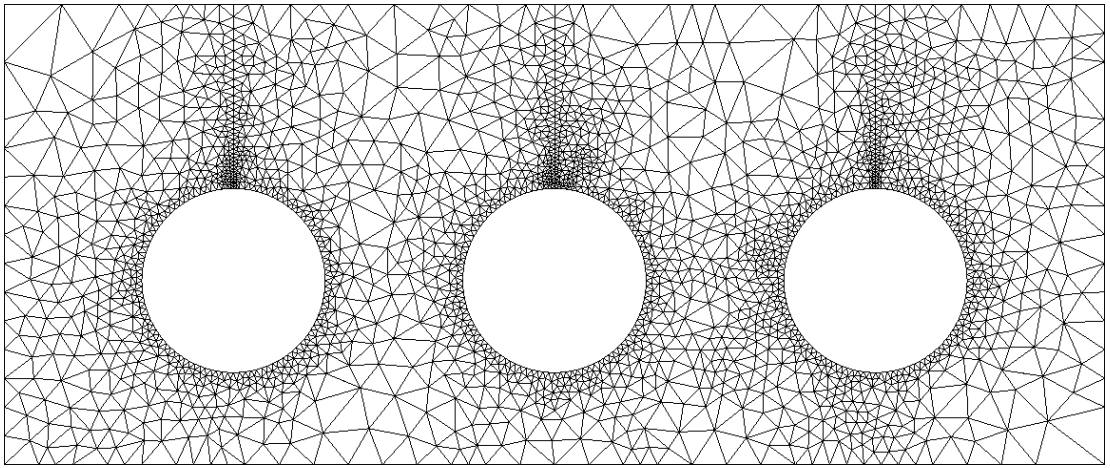


Fig. 93 - Caso 13: Gráfico dos resultados obtidos

O triangulador, de forma bastante positiva, mostrou-se insensível à conexidade da geometria, com um comportamento praticamente linear nos parâmetros de saída.



Um furo Dois furos



Três furos

Fig. 94 - Caso 13: visualização das triangulações obtidas

5.1.14. Caso 14: Geometria do VLS com restrição de mínimo ângulo interno

O caso 14, juntamente com o caso 15 e 16, representam os testes com o triangulador em geometrias reais. O caso 14 consiste na análise paramétrica do comportamento do triangulador com relação a variação da restrição de mínimo ângulo interno dos triângulos em uma geometria externa da parte frontal do VLS (Veículo Lançador de Satélites Brasileiro). A geometria fornecida ao gerador é o contorno externo apresentado na figura Fig. 96, onde são utilizados 10 segmentos de reta para aproximar o arco de circunferência do bico do foguete, e 20 segmentos de reta no arco de oval do contorno externo. Os ângulos utilizados na análise foram 0^o, 10^o, 20^o, 30^o e 32^o, e a geometria fornecida contém 36 vértices e 36 segmentos.

Caso 14					
Ângulo mínimo	0	10	20	30	32
Tempo total	41	50	60	80	95
Vértices	36	53	70	147	220
Triângulos	34	60	89	237	376
Arestas	69	112	158	383	595
Arestas de contorno	36	44	49	55	62

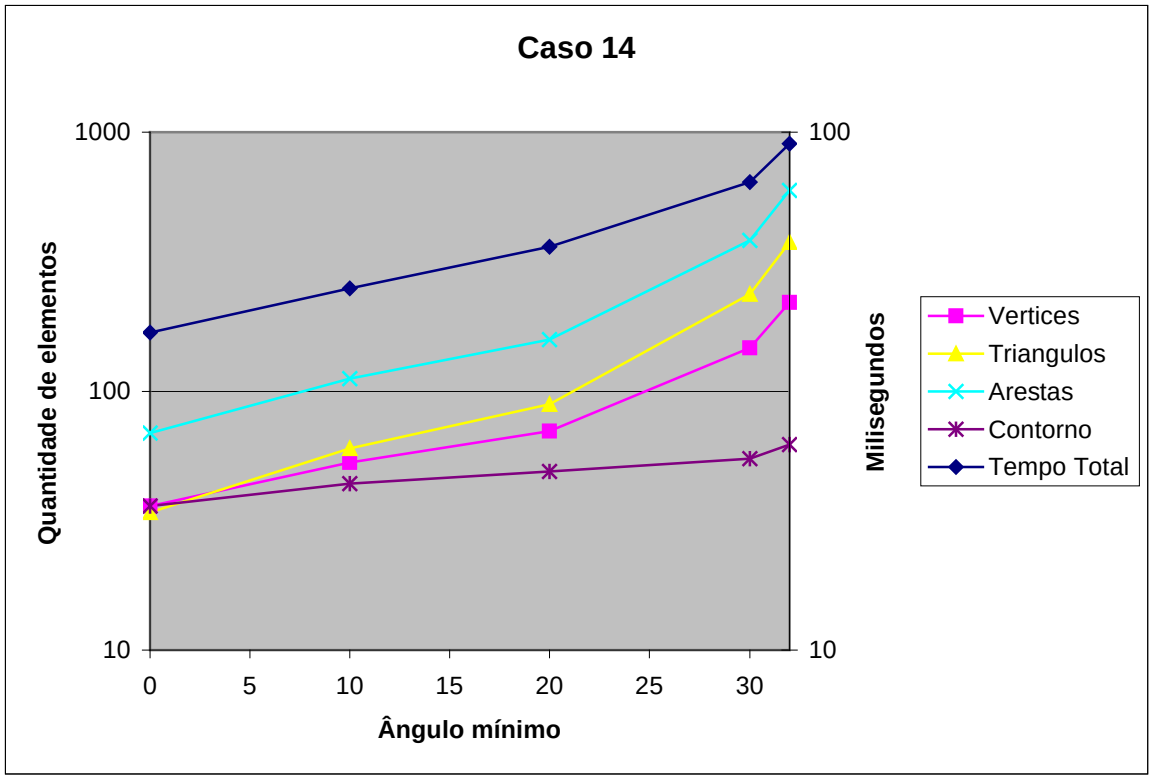


Fig. 95 - Caso 14: Gráfico dos resultados obtidos

Conforme pode ser observado nos resultados, o número de elementos utilizado foi baixo, demonstrando, de forma positiva, uma insensibilidade do método ao tipo de geometria utilizada. Com isto, podemos concluir que as geometrias que estavam sendo testadas até aqui, que por praticidade e facilidade de análise eram geometrias quadradas, não invalidam as análises efetuadas.

A maior parte dos elementos concentrou-se na região do bico do foguete, e o número de arestas de contorno foi praticamente igual ao valor de entrada, sendo que as arestas de contorno adicionadas estão todas nos segmentos retos com comprimento maior e próximas ao bico também.

Os resultados confirmam mais uma vez que a dependência do número de elementos com o ângulo interno mínimo existe praticamente apenas quando são fornecidas arestas de contorno de comprimentos bastante diferentes.

Da mesma forma que em todos os outros casos, a malha não apresenta direções preferenciais, o que pode ser um fator positivo ou negativo, dependendo do contexto da aplicação.

Um último aspecto interessante de ser observado é que para ângulos de até cerca de 30^{o}, o tamanho médio dos triângulos obtidos é bastante similar ao tamanho das aresta de contorno próximas a este.

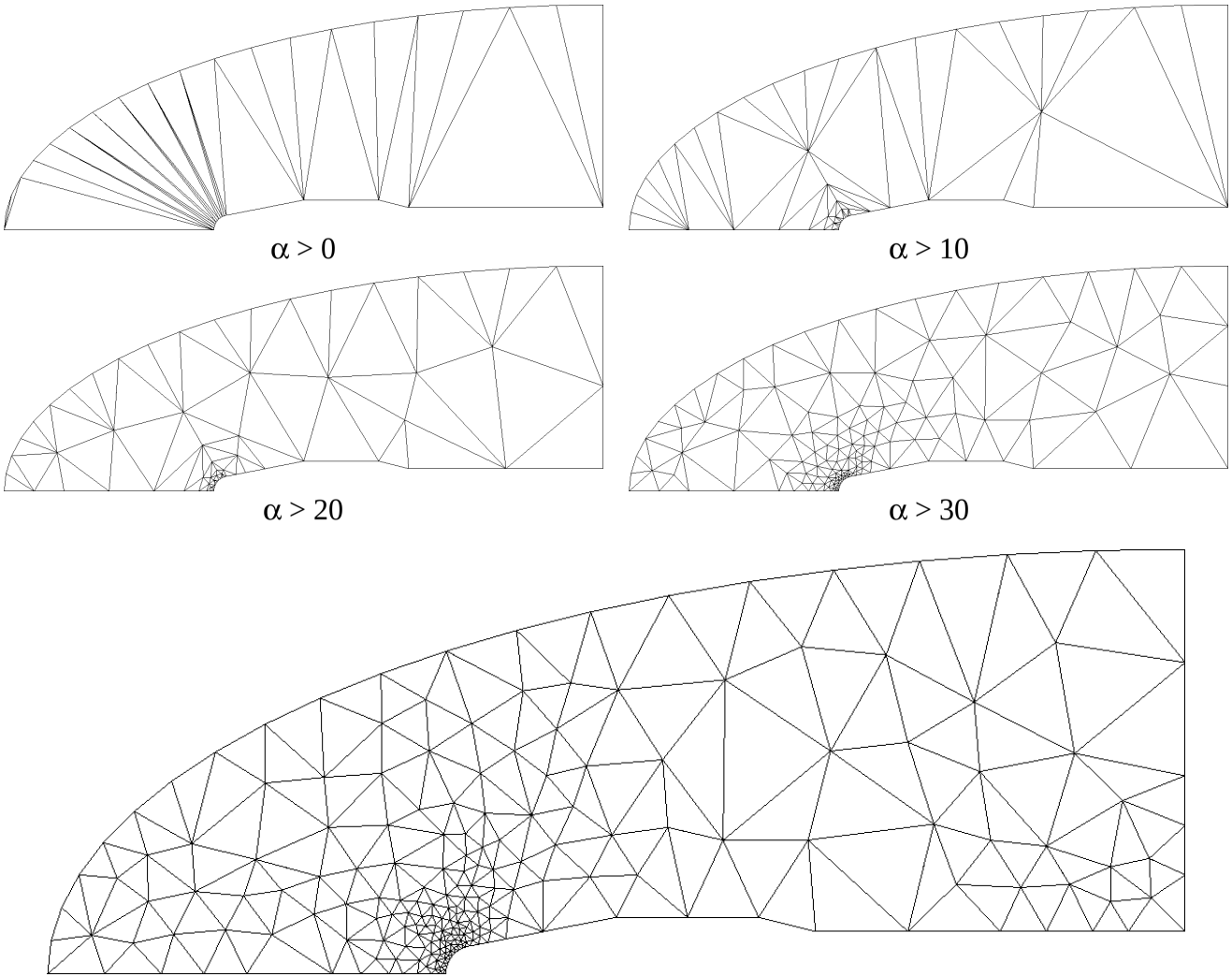


Fig. 96 - Caso 14: visualização das triangulações obtidas

5.1.15. Caso 15: Geometria do VLS com restrição de máxima área de elementos

O caso 15 tem um objetivo similar ao caso 14, mas utilizando como restrição o tamanho máximo dos elementos. A geometria é exatamente a mesma do caso 14.

Caso 15						
Inverso área	0	0,1	1	10	100	1000
Tempo Total	41	52	140	786	7152	71321
Vértices	36	68	392	3501	33714	335472
Triângulos	34	88	680	6726	66536	668176
Arestas	69	155	1071	10226	100249	1003647
Arestas de contorno	36	46	102	274	890	2766

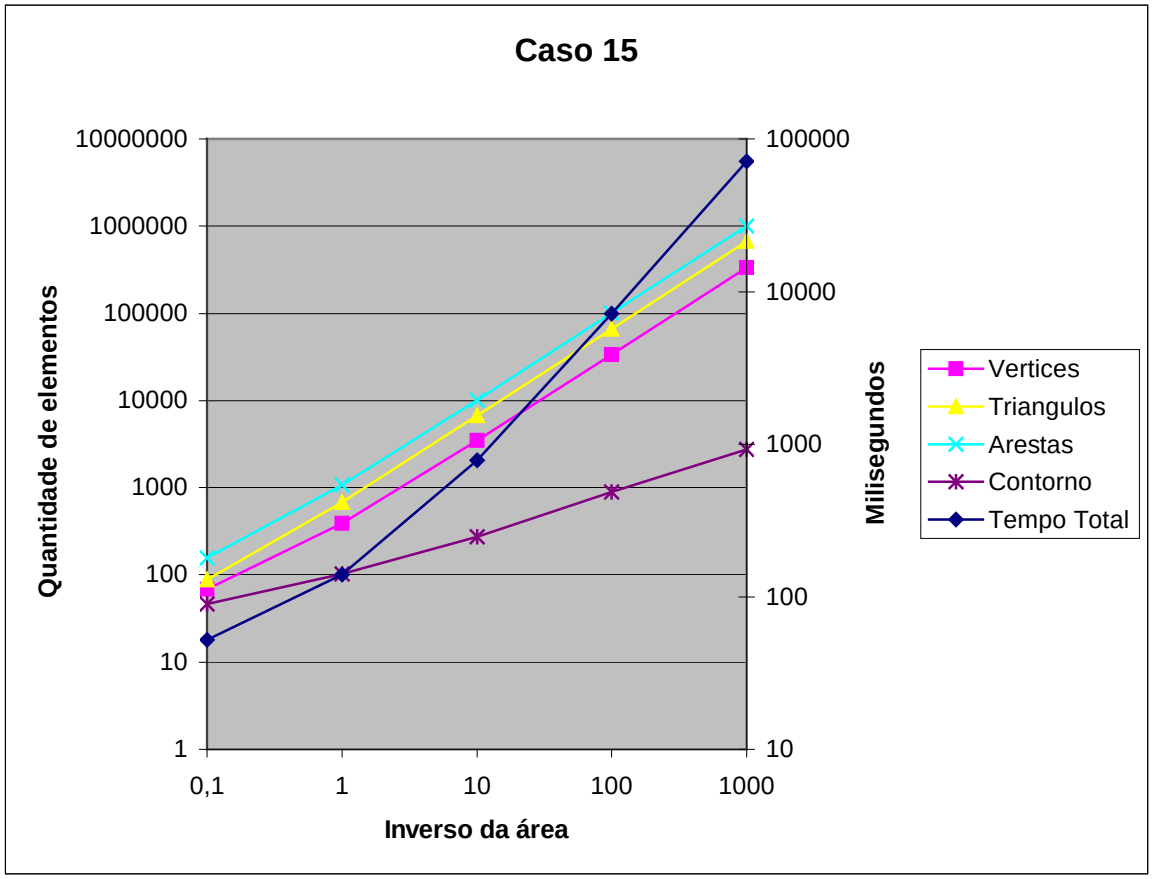


Fig. 97 - Caso 15: Gráfico dos resultados obtidos

Os resultados apresentados confirmam uma independência do triangulador quanto a geometria também quando utilizando-se restrição de área máxima dos triângulos.

Um detalhe interessante é que os elementos da região próxima ao bico do foguete (segmentos fornecidos de menor comprimento) tem ângulos internos ruins até restrições de áreas de 10^{-0}. Apenas quando a restrição de área de 10^{-1} é aplicada, quando então os elementos internos passam a ter arestas de comprimento similares às arestas fornecidas no bico, é que a malha passa a ter elementos de boa qualidade em todo o domínio.

Novamente, a malha apresentou elementos de tamanho uniforme sem nenhuma direção preferencial.

As malhas para os casos de restrição de área máximo de triângulo de 10^{-2} e 10^{-3} servem apenas para ilustrar a capacidade do triangulador de gerar malhas com uma grande quantidade de elementos. Mesmo nestes dois níveis de refino (acima de 100mil triângulos), o triangulador apresentou um comportamento praticamente linear do tempo de computação em relação ao número de elementos.

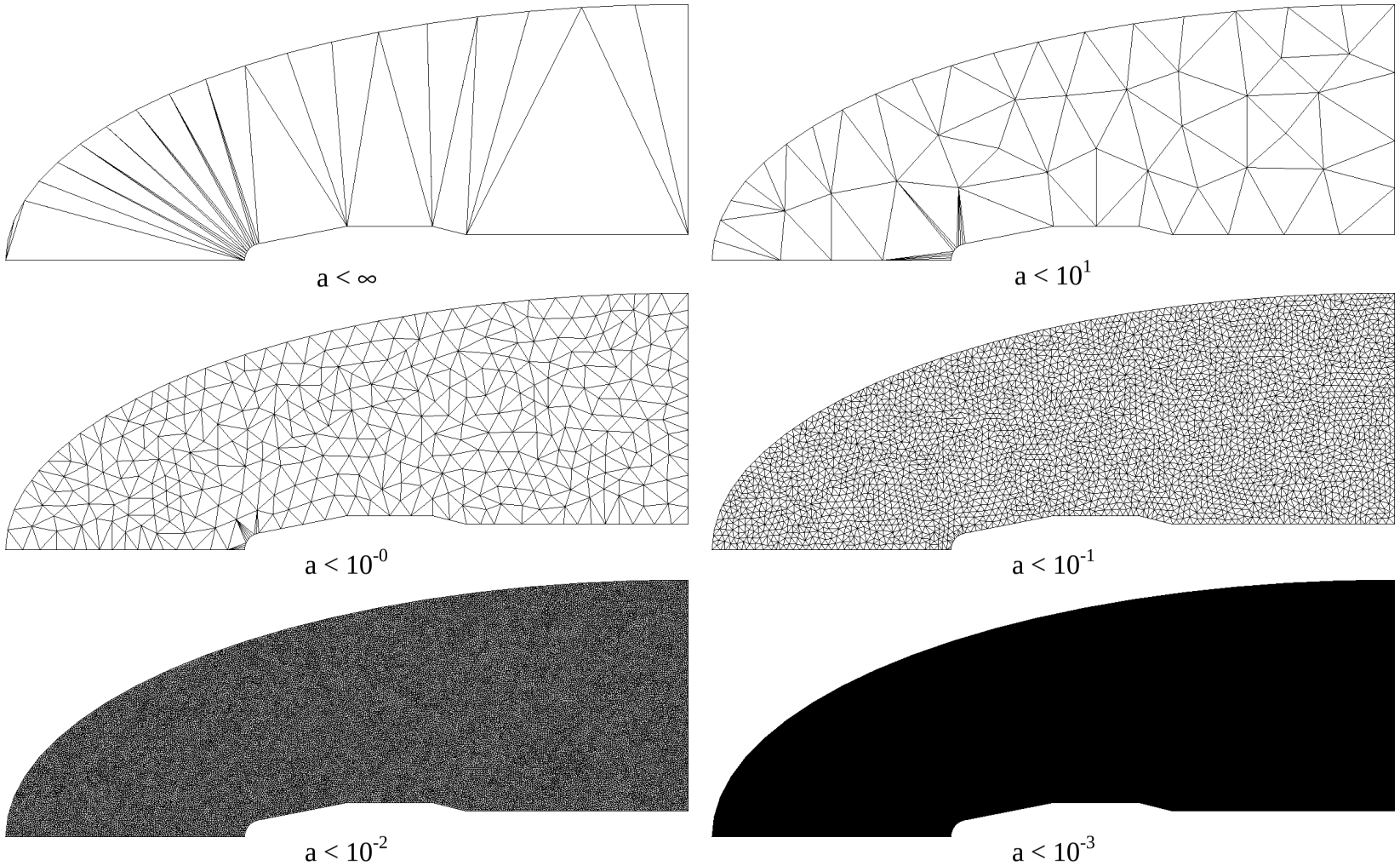


Fig. 98 - Caso 15: visualização das triangulações obtidas

5.1.16. Caso 16: Geometria do VLS com refino na região de choque

O ultimo caso bidimensional apresentado neste trabalho tem o objetivo de analisar o comportamento do triangulador em uma geometria real, com restrições simultâneas de mínimo ângulo interno dos elementos e refino na suposta região de choque. Esta região de choque pode ir sendo captada adaptativamente durante a solução. A geometria utilizada é similar aos casos 14 e 15, mas com 50 arestas na aproximação do bico do foguete. Para todos os casos, o mínimo ângulo interno admitido dos elementos é 30^{o}, e a análise paramétrica foi feita sobre o tamanho de elemento admitido no choque, onde foram utilizadas restrições de comprimento de $1x10^{0}$, $5x10^{-1}$, $2{,}5x10^{-1}$, $1x10^{-2}$ e $5x10^{-2}$, sendo o perímetro total do oval 50. Os resultados são analisados contra o número total de arestas utilizados no refino.

Caso 16					
Arestas no refino	50	100	200	500	1000
Tempo Total	121	219	366	836	1732
Vértices	402	732	1439	3466	7308
Triângulos	717	1367	2772	6811	14485
Arestas	1118	2098	4210	10276	21792
Arestas de contorno	85	95	104	119	129

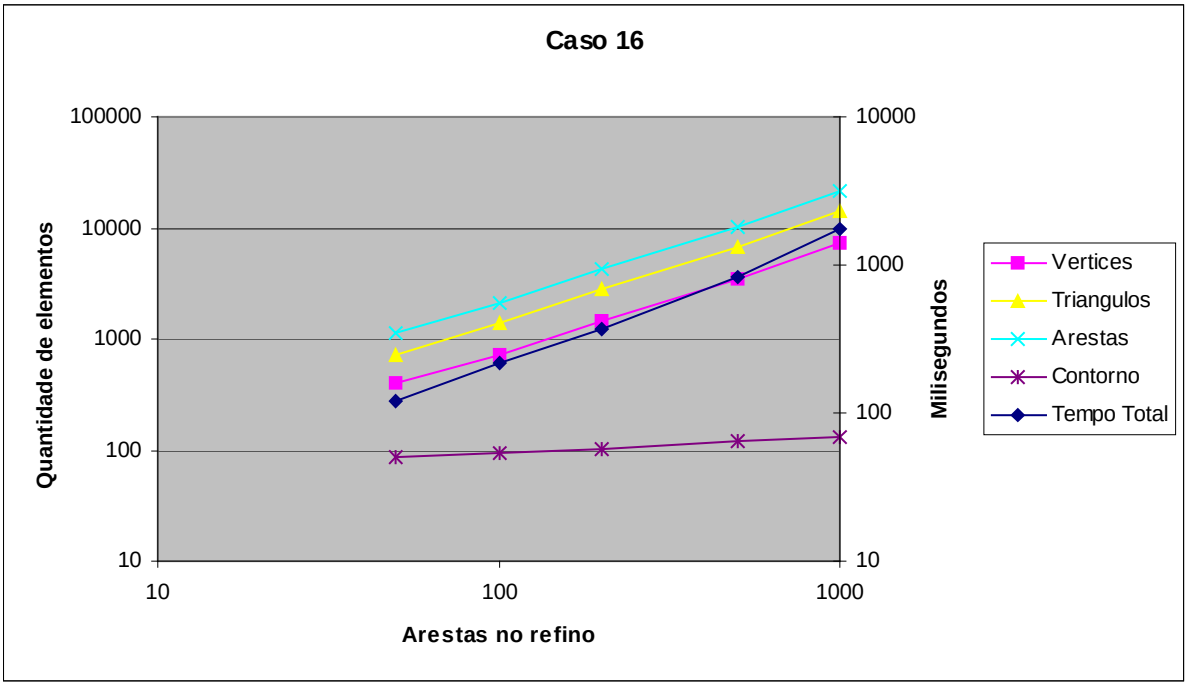


Fig. 99 - Caso 16: Gráfico dos resultados obtidos

Conforme pode ser observado nos resultados apresentados no gráfico da Fig. 99, o triangulador apresentou uma característica linear quanto ao número de arestas na região do refino.

Devido à restrição de mínimo ângulo interno de 30^o para todas as malhas, desde a primeira malha a região do bico do foguete possui um refino considerável de elementos.

A região de influência do refino manteve-se praticamente inalterada, independente do número de arestas impostas em tal região. Isto é extremamente interessante pois pode-se impor tamanhos de elementos extremamente pequenos sem aumentar ou influenciar a malha em regiões que não são de interesse. Ou seja, o refino não se propaga exageradamente.

Fora a região do refino, onde foi imposta a condição de que os elementos fossem tangentes à curva de refino, a malha não apresentou direção preferencial.

De uma forma geral, o triangulador apresentou um comportamento extremamente interessante para a sua utilização em simuladores adaptativos, lembrando que a restrição de mínimo ângulo interno foi, para todas as malhas, fixa em 30^o, o que é uma condição excelente de qualidade de malha para simulação.

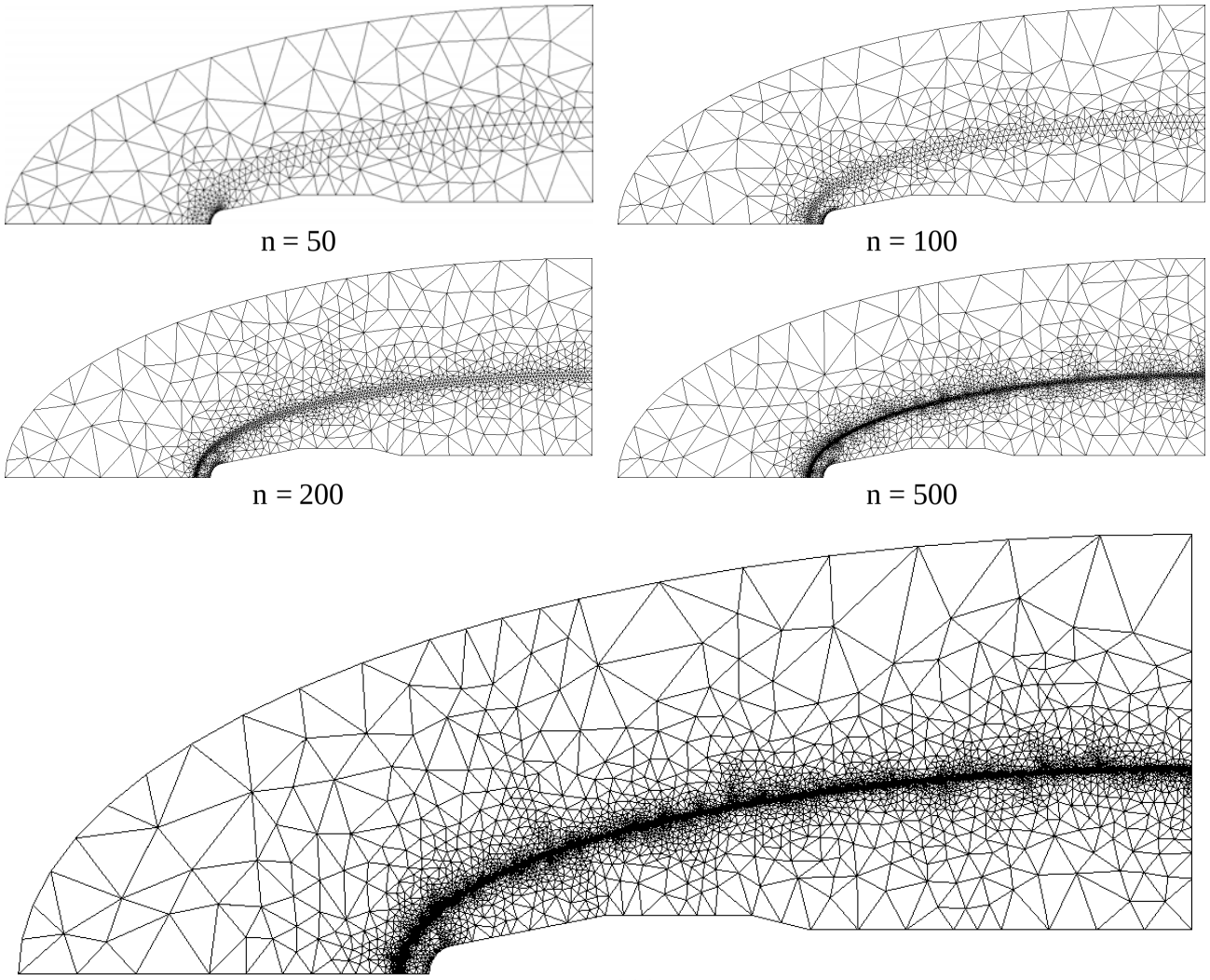


Fig. 100 - Caso 16: visualização das triangulações obtidas

5.2. Resultados 2,5 dimensionais

5.2.1. Caso 17: Junção de quadriláteros

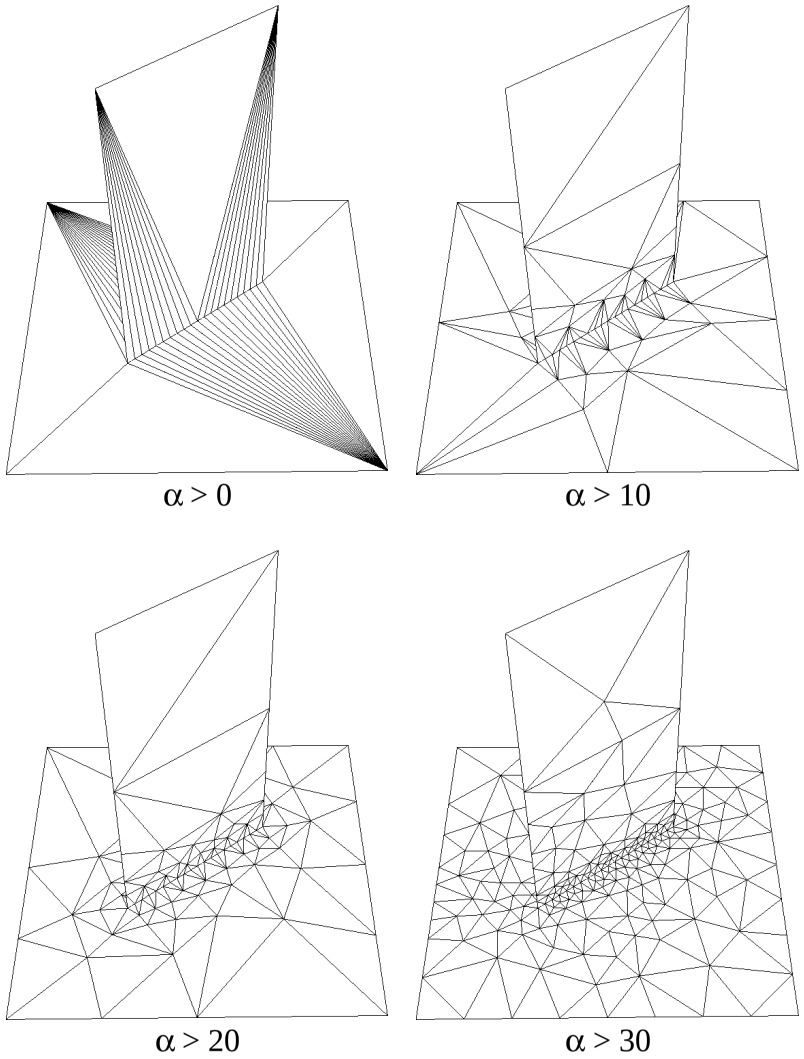
O primeiro caso para dimensão 2,5D procura englobar os comportamentos apresentados nos casos de 1 a 10, afim de verificar se tais comportamentos mantém-se na adaptação do gerador para estes domínios.

A geometria utilizada são duas placas, uma horizontal e uma vertical. A placa horizontal é um quadrado de lado 1, e a vertical um retângulo de 0,5 por 1. A placa vertical encosta na placa horizontal de forma inclinada no seu interior, formando uma restrição geométrica. Em tal restrição geométrica, restrições de máximo comprimento de aresta dos triângulos e mínimo ângulo interno dos triângulos foram impostas.

A análise paramétrica foi efetuada sobre dois parâmetros. Utilizou-se três valores diferentes de comprimento máximo admitido de aresta: $5x10^{-2}$, $1x10^{-2}$ e $5x10^{-3}$. Para cada comprimento máximo de aresta, foi variado o mínimo ângulo interno admitido, utilizando-se valores de 0^o, 10^o, 20^o e 30^o. O número de vértices na geometria fornecida é 8, e de segmentos 8.

Os resultados mostrados em Fig. 101, Fig. 102 e Fig. 103 apresentaram um comportamento bastante similar aos casos de 1 a 10:

- O triangulador tem uma resposta linear dos seus parâmetros de saída quanto ao inverso do comprimento máximo de aresta;
- O triangulador tem uma resposta exponencial dos seus parâmetros de saída quanto ao mínimo ângulo de aresta até ângulos de cerca de 30^o. Para ângulos maiores que 30^o, o número entidades utilizadas cresce demasiadamente. Para ângulos maiores que 34^o, o gerador não consegue obter uma triangulação que respeite tais critérios;
- O refino localizou-se fundamentalmente próximo a aresta com restrição de comprimento máximo de elementos.



Caso 17 com 20 arestas na conexão				
Ângulo mínimo	0	10	20	30
Tempo Total	80	120	170	210
Vértices	48	86	127	284
Triângulos	65	128	206	498
Arestas	111	212	331	780

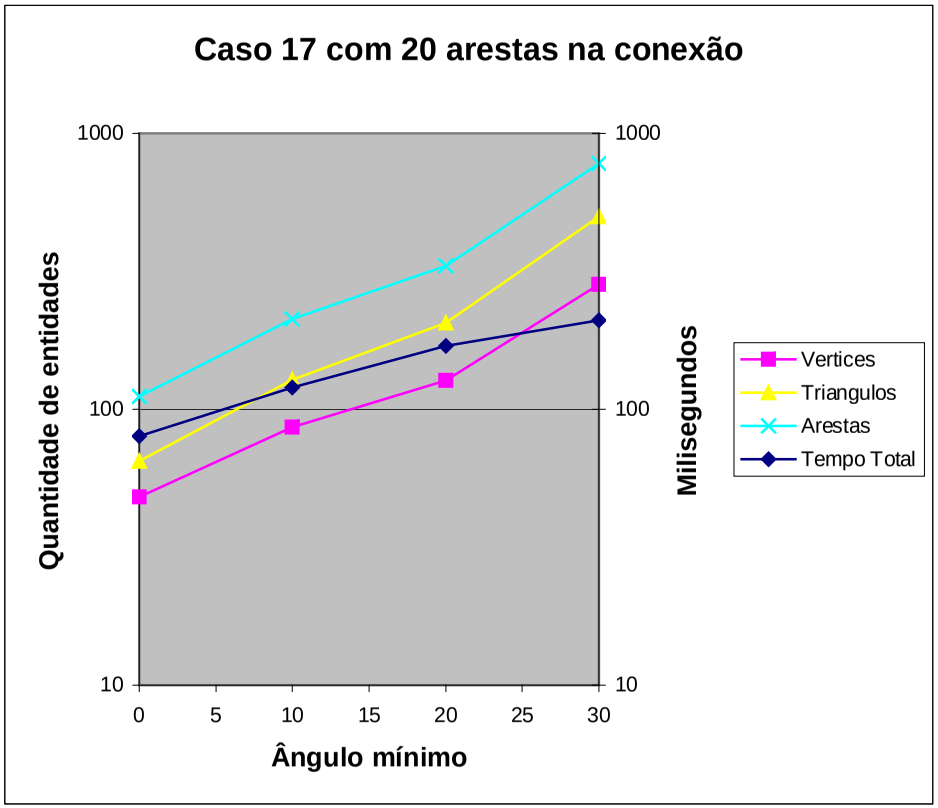
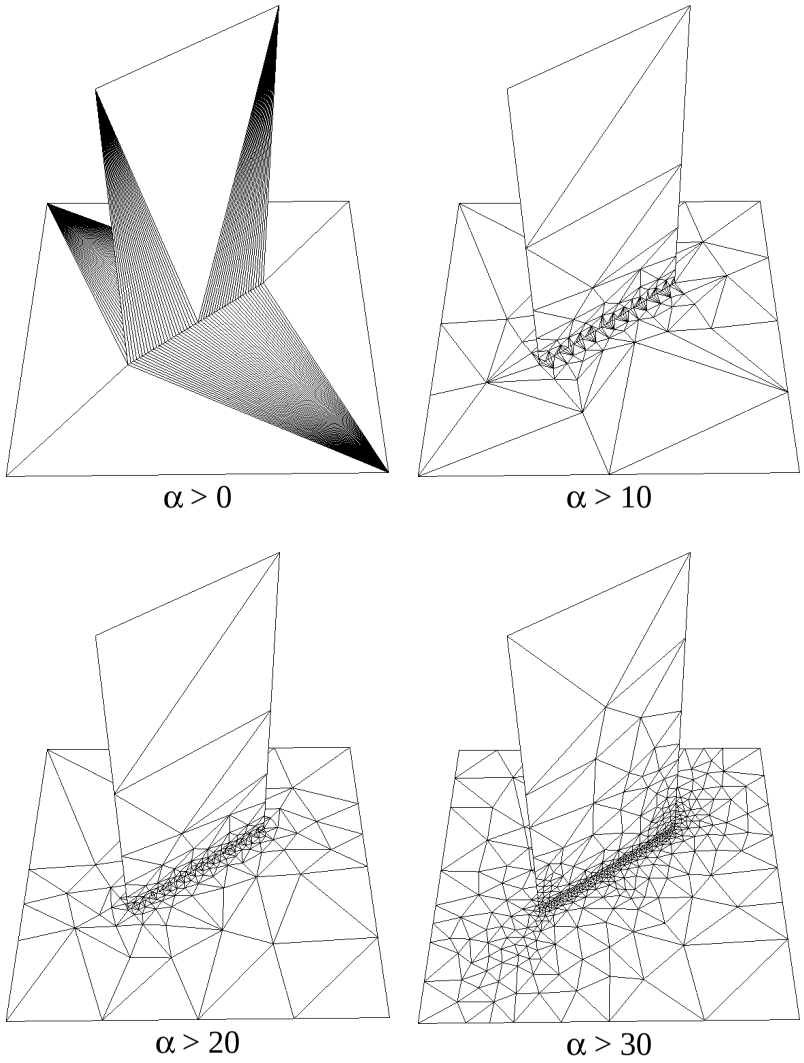


Fig. 101 - Caso 17: Junção de quadriláteros com 20 arestas na conexão



Caso 17 com 50 arestas na conexão				
Ângulo mínimo	0	10	20	30
Tempo Total	120	150	175	200
Vértices	108	201	291	658
Triângulos	155	329	497	1203
Arestas	261	531	786	1859

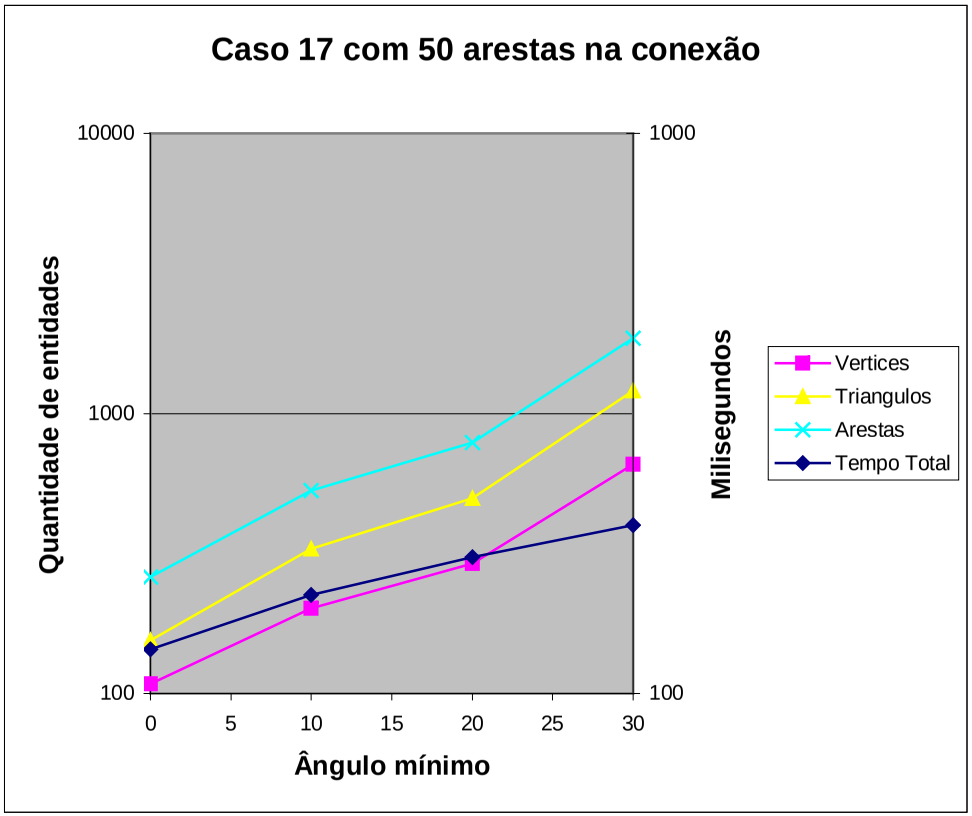
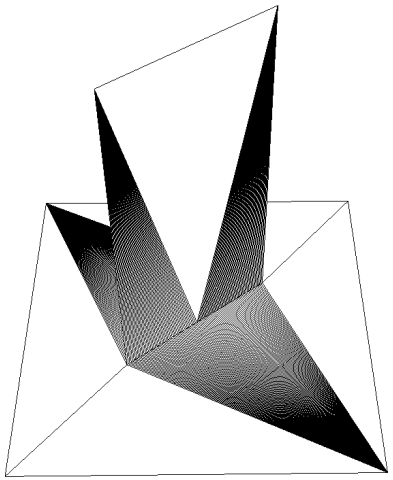
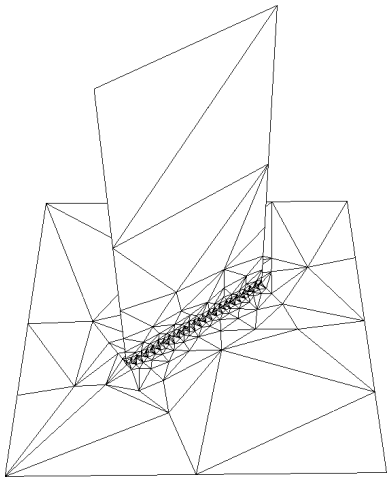


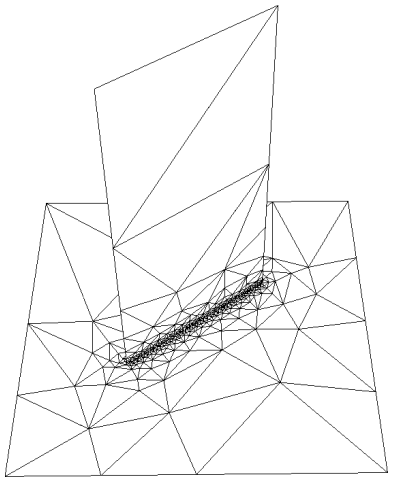
Fig. 102 - Caso 17: Junção de quadriláteros com 50 arestas na conexão



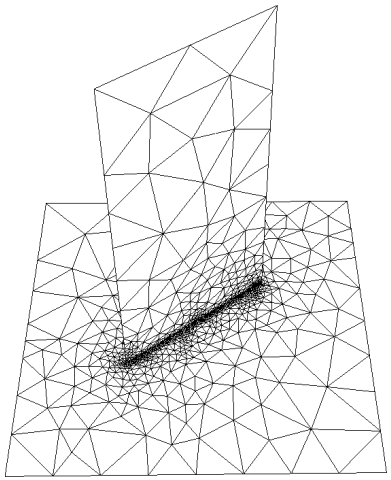
α > 0



α > 10



α > 20



α > 30

Caso 17 com 100 arestas na conexão				
Ângulo mínimo	0	10	20	30
Tempo Total	120	160	180	240
Vértices	208	389	548	1287
Triângulos	305	647	962	2407
Arestas	511	1034	1508	3693

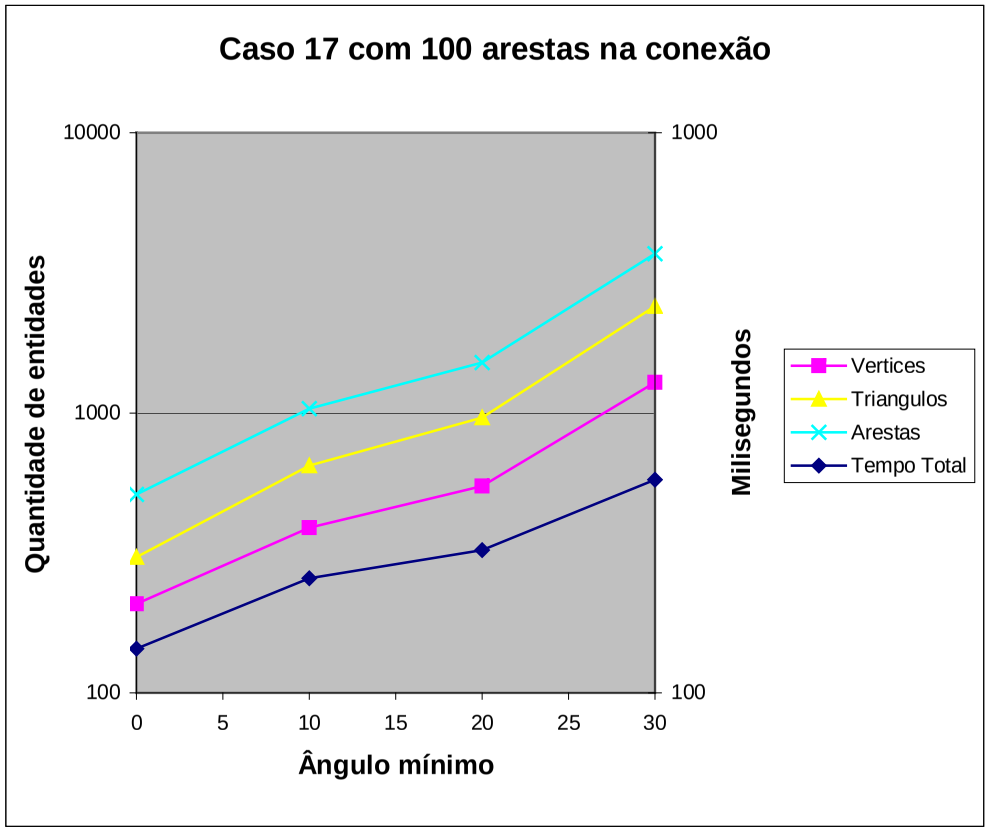


Fig. 103 - Caso 17: Junção de quadriláteros com 100 arestas na conexão

5.2.2. Caso 18: Junção de elementos planos curvos

O segundo caso para dimensão 2,5D tem o objetivo de verificar o comportamento do gerador quando utilizando-se geometrias interconectados com fronteiras curvas aproximadas por segmentos de reta. A geometria é composta por um disco com um furo quadrado, encaixado perpendicularmente com um retângulo com um furo semi-circular. A circunferência da base é aproximada por 200 segmentos de reta, e a semi-circunferência do plano vertical é aproximada por 100 segmentos de reta. A análise paramétrica é efetuada sobre o mínimo ângulo interno admitido, utilizando-se valores de 0^o, 10^o, 20^o e 30^o. A geometria fornecida contem 308 vértices, 308 segmentos de reta e 1 furo.

Caso 18				
Angulo mínimo	0	10	20	30
Tempo total	110	170	190	210
Vértices	311	482	668	1296
Triângulos	309	605	957	2175
Arestas	619	1082	1620	3466

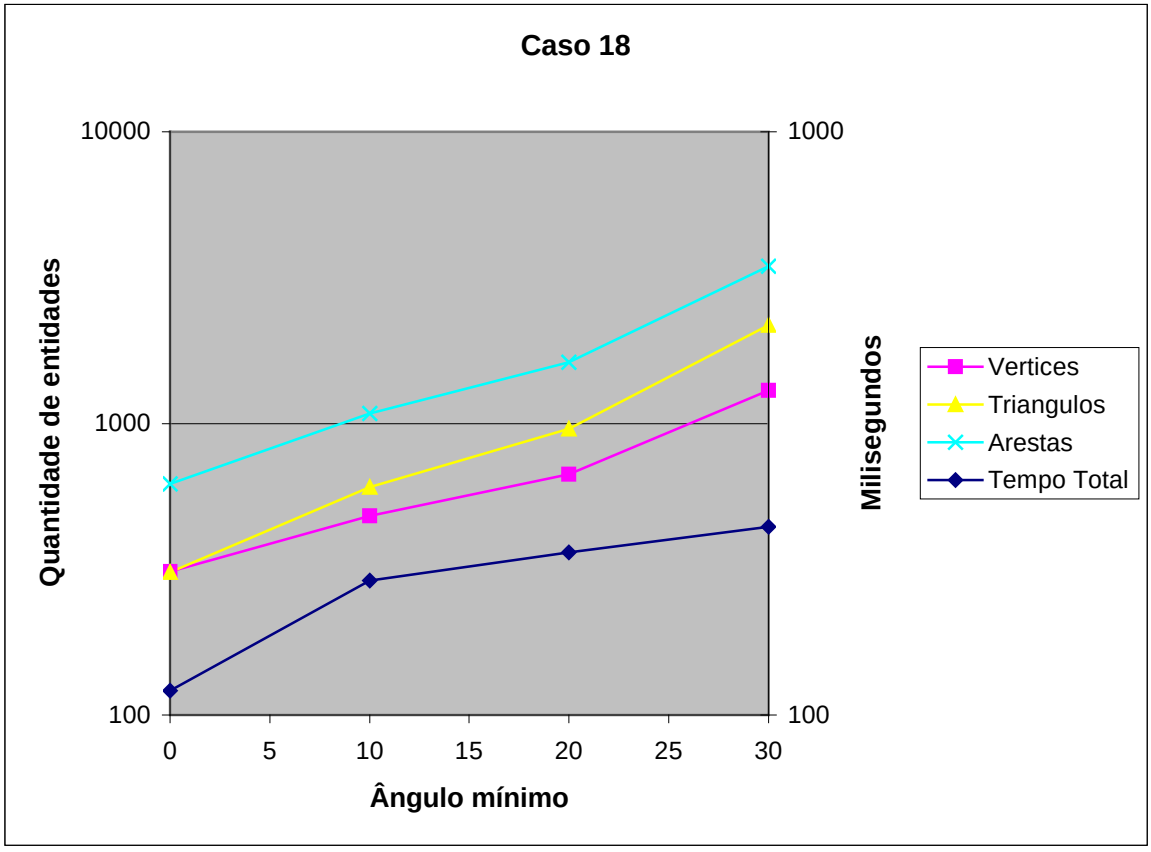


Fig. 104 - Caso 18: Gráfico dos resultados obtidos

Os resultados obtidos com tal caso foram similares aos obtidos anteriormente, com destaque para um efeito que pode ser observado na Fig. 106: o refino dos elementos na placa vertical próximo à aresta curva, necessário para manter-se a condição de mínimo ângulo interno, propaga-se para a placa horizontal, onde não seria necessário um refino para manter-se tal critério. Este fenômeno mostra a interconectividade entre os domínios, onde os elementos à serem refinados são escolhidos pelo método nos diversos domínios planos simultaneamente.

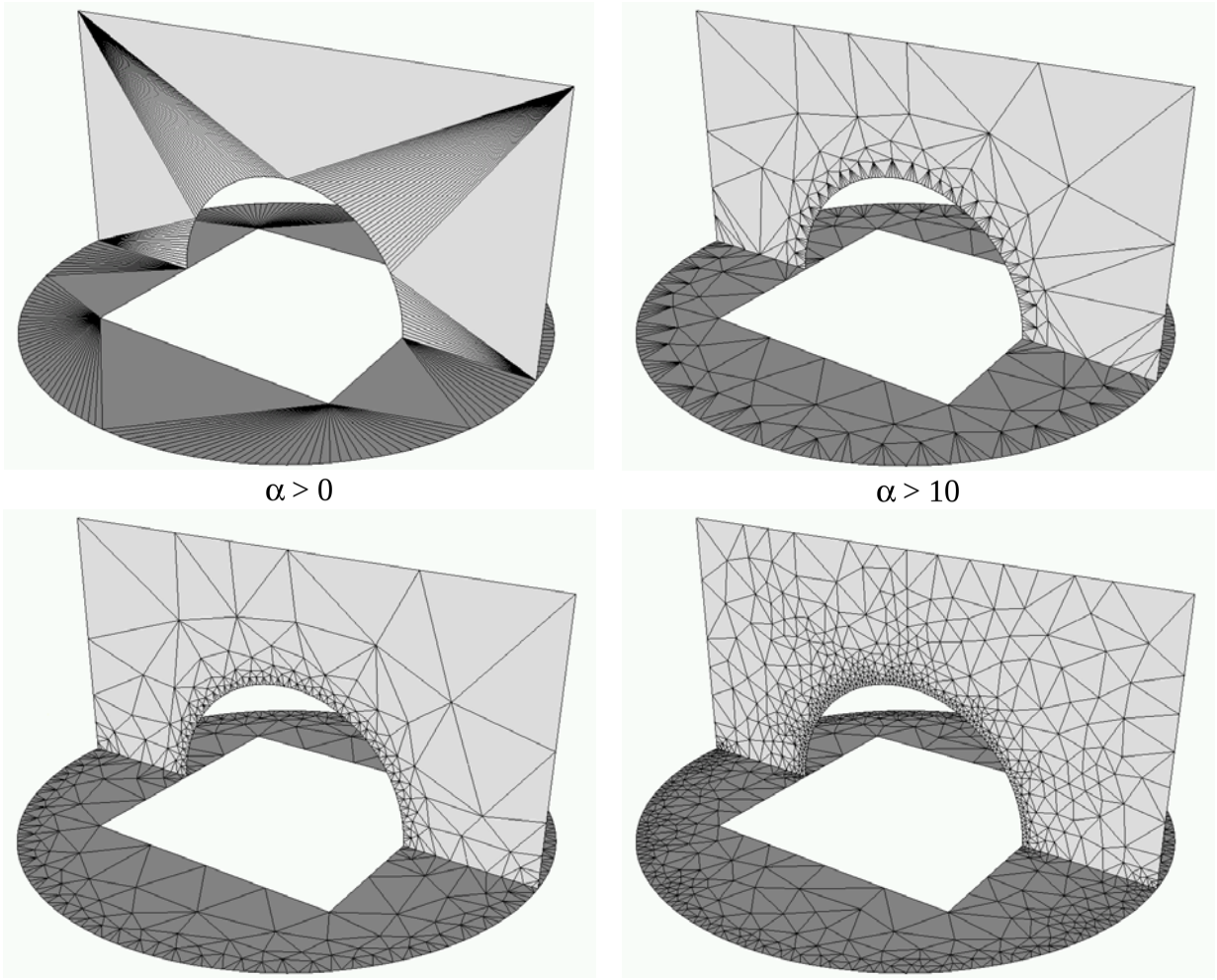
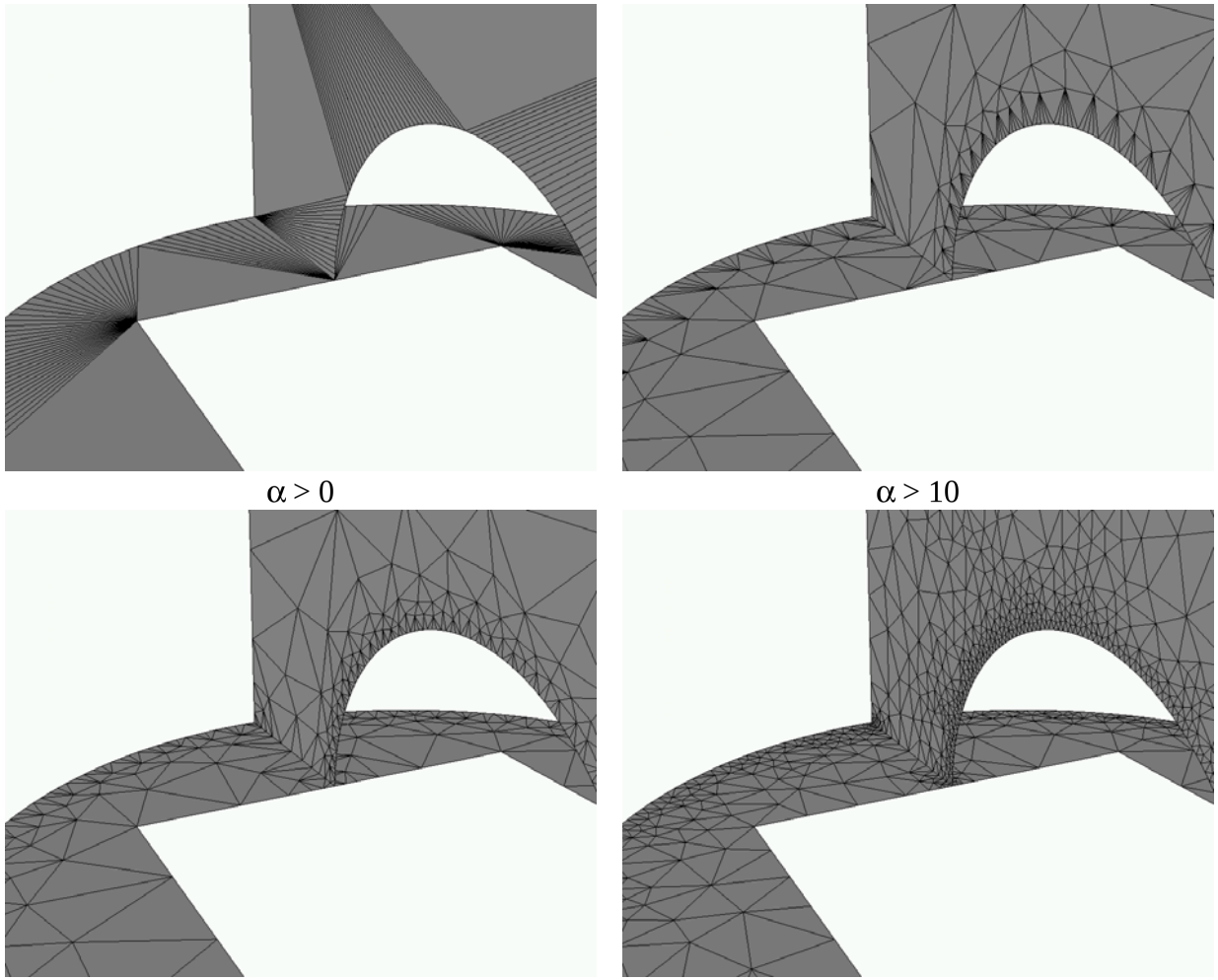


Fig. 105 - Caso 18: visualização das triangulações obtidas



α > 0 α > 10

α > 20 α > 30

Fig. 106 - Caso 18: visualização ampliada das triangulações obtidas

5.2.3. Caso 19: Geometria final

O ultimo caso do presente trabalho é uma ilustração de todos os aspectos envolvidos nos casos anteriores em uma única geometria 2,5D. A geometria contém 192 vértices, 196 arestas e 2 furos, e todos os elementos possuem ângulos internos de no mínimo 30^{o}.

Caso 19	
Tempo total	611
Vértices	3056
Triângulos	5335
Arestas	8386

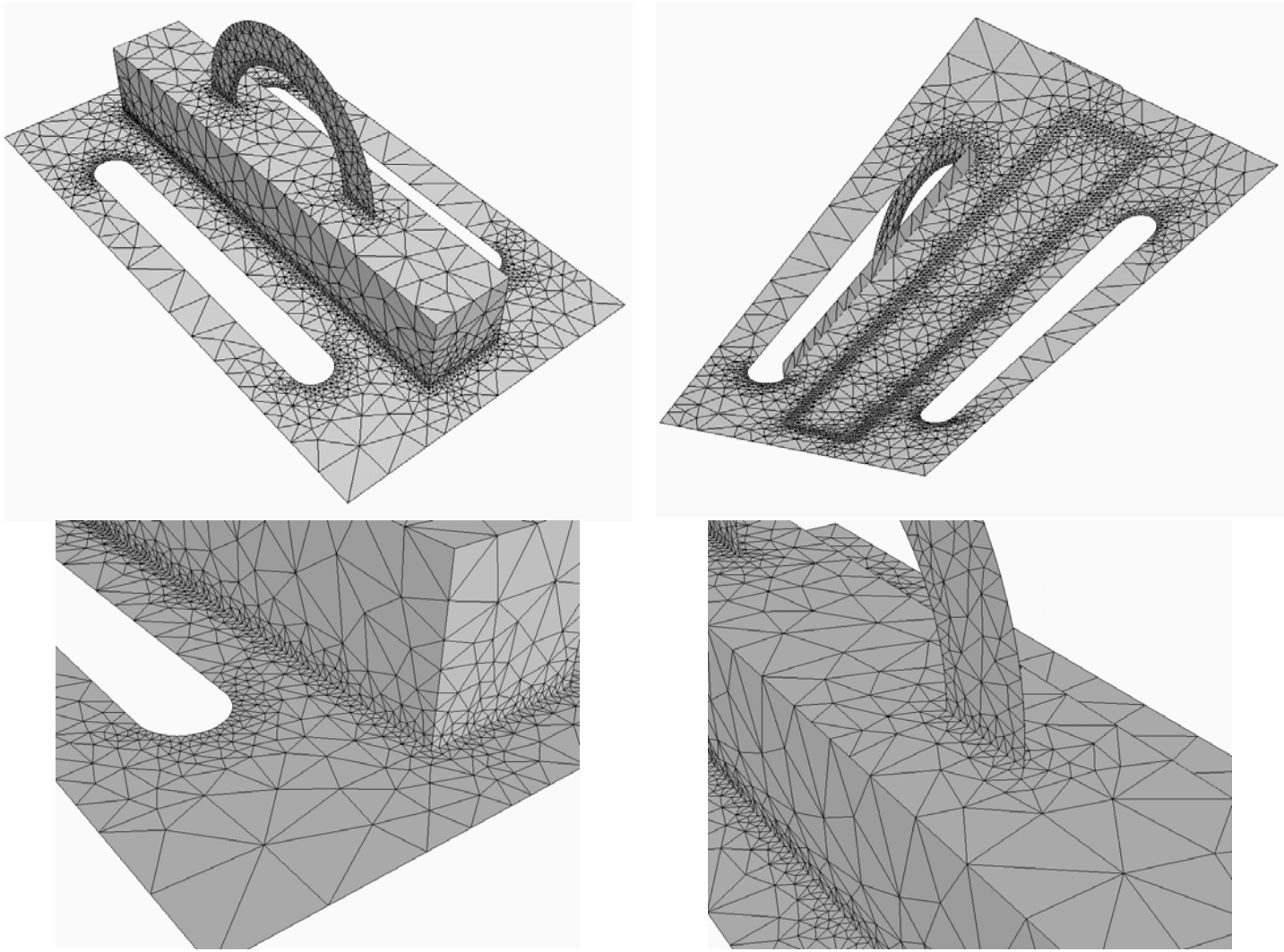


Fig. 107 - Caso 19: visualização das triangulações obtidas

Referências Bibliográficas

[1] AEA Technology. “Build 4.3 – users manual”, 1999.

[2] AGGARWAL, A., GUIBAS, L. J., SAXE, J. e SHOR, P. W. “A linear-time algorithm for computing the Voronoi diagram of convex polygon”. Discretization and Computational Geometry, vol. 4, p. 591-604, 1989.

[3] AURENHAMMER, F. “Voronoi diagrams – a survey of a fundamental geometric data structure”. Anais ACM Computing Surveys, vol. 23, p. 345-405, 1991.

[4] AVNAIM, F., BOISSONNAT, J-D, DEV-ILLERS, O., PREPARATA, F. P. e YVINEC, M. “Evaluating signs of determinants using single-precision arithmetic”, 1995.

[5] BABUSKA, I. e AZIZ, A. K. “On the angle condition in the finite element method”. SIAM Journal on Numerical Analysis, vol. 13, nº 2, p. 214-226, 1976.

[6] BAKER, T. J. ”Automatic mesh generation for complex three-dimensional regions using a constrained Delaunay triangulation”. Engineering with Computers, nº 5, p. 161-175, 1989.

[7] BAKER, B., GROSSE, E. e RAFFERTY, C. S. “Non-obtuse triangulation of polygons”. Discrete and Computational Geometry, vol. 3, p. 147-168, 1988.

[8] BALIGA, B. R. and PATANKAR, S. V. “A new finite element formulation for convection-diffusion problems”, Numerical Heat Transfer, vol. 3, p. 939-409, 1980.

[9] BANK, R. E., “PLTMG User’s Guide”. SIAM, 1990.

[10] BANK, R. E., SHERMANN, A. H. e WEISER, A. “Refinement algorithms and data structures for irregular local mesh refinement”. R. Stepleman et al., Scientific Computing. IMACS/North-Holland, p. 3-17, 1983.

[11] BARTH, T. J. e JESPERSEN, D. C. “The design and application of upwind schemes on unstructured meshes”. Anais AIAA 27º Aerospace Sciences Meeting, Reno, 1989.

[12] BERN, M., EDELSBRUNNER, H., EPPSTEIN, D., MITCHELL, S. e TAN, T. S. “Edge-insertion for optimal triangulations”. Discretization and Computational Geometry, vol. 10, p. 47-65, 1993.

[13] BERN, M. e EPPSTEIN, D. “Mesh generation and optimal triangulation”. Lecture Notes Series on Computing, vol. 1, p. 23-90, World Scientific, Singapore, 1992.

[14] BERN, M. e EPPSTEIN, D. “Polynomial-size non-obtuse triangulation of polygons”. International Journal of Computational Geometry and Applications, vol. 2, p. 241-255, 1992.

[15] BERN, M., EPPSTEIN, D. e GILBERT, J. R. “Provably good mesh generation”. Anais 31º IEEE Symposium on Foundations of Computer Science, p. 231-241, 1990.

[16] BERN, M., MITCHELL, S. e RUPPERT, J. “Linear-size non-obtuse triangulation of polygons”. Anais 10º ACM Symposium on Computational Geometry, p. 221-230, 1994.

[17] BERN, M. e PLASSMANN, P. “Mesh generation”. NEC Research Index Website, http://citeseer.nj.nec.com/cs, 2000.

[18] BOWYER, A. “Computing Dirichlet tessellations”. Computer Journal, vol. 24, nº 2, p. 162-166, 1981.

[19] CANANN, S. A., MUTHUKRISHNAN, S. N. e PHILLIPS, R. K. “Topological refinement procedures for triangular finite element meshes”. Engineering with Computers, vol. 12, nº 3 e 4, p. 243-255, 1996.

[20] CAREY, G. F. e ODEN, J. T. “Finite elements: computational aspects”. Prentice-Hall, Englewood Cliffs, New Jersey, 1984.

[21] CASTILLO, J. E. ”Mathematical aspects of numerical grid generation”. SIAM, 1991.

[22] CAVENDISH, J. C., FIELD, D. A. e FREY, W. H. “An approach to automatic three-dimensional finite element mesh generation”. International Journal for Numerical Methods in Engineering, vol. 21, nº 2, p. 329-347, 1985.

[23] CHEW, L. P. “Guaranteed-quality triangular meshes”. Technical report TR-89-983, Computer Science Department, Cornell University, 1989.

[24] CHEW, L. P. “Guaranteed-quality mesh generation for curved surfaces”. Anais 9º ACM Symposium on Computational Geometry, p. 274-280, 1993.

[25] CLARKSON, K. L. “Safe and effective determinant evaluation”. Anais 33º IEEE Symposium on Foundations of Computer Science, p. 387-395, 1992.

[26] CLARKSON, K. L. e SHOR, P. W. “Applications of random sampling in computational geometry, II”. Discrete & Computational Geometry, vol. 4, nº 1, p. 387-421, 1989.

[27] D’AZEVEDO, E. F. e SIMPSON, R. B. “On optimal interpolation triangle incidences”. SIAM Journal on Scientific and Statistical Computing, vol. 10, p. 1063-1075, 1989.

[28] DELAUNAY, B. N., “Sur la sphère vide”. Izvestia Akademia Nauk SSSR, VII Seria, Otdelenei Matematicheskii i Estestvennyka Nauk, vol. 7, p. 793-800, 1934.

[29] DEY, T., DILLENCOURT, M. B. e GHOSH, S. K. “Triangulating with high connectivity”. Technical Report 94-24, Department of Informatics and Computer Science, UC-Irvine, 1994.

[30] DICKERSON, M. T., DRYSDALE, R. L. S., McELFRESH, S. A., WELZL, E. “Fast greedy triangulation algorithm”. Anais 10º ACM Symposium on Computational Geometry, p. 211-220, 1994.

[31] DILLENCOURT, M. B. e SMITH, W. D. “A simple method for resolving degeneracies in Delaunay triangulations”. Anais 20º International Coloque – Automata, Languages and Programming. Springer-Verlag LNCS 700, p. 177-188, 1993.

[32] DJIDJEV, H. e LINGAS, A. “On computing the Voronoi diagram for restricted planar-figures”. Anais 2 º Workshop on Algorithms and Data Structures, Springer-Verlag LNCS 519, p. 54-64, 1991.

[33] DOBKIN, D. P. “Computational geometry and computer graphics”, Technical Report, Department of Computer Science, Princeton University, 1998.

[34] DÜPPE, R. D. e GOTTSCHALK, H. J. “Automatische interpolation von isolinien bei willkürlichen stützpunkten”. Allgemeine Vermessungsnachrichten, vol. 77, p. 423-426, 1970.

[35] EDELSBRUNNER, H. e MÜCKE, E. P. “Simulation of simplicity, a technique to cope with degenerate cases in geometric computations”. ACM Trans. Graphics, vol. 9, p. 66-104, 1990.

[36] EDELSBRUNNER, H. e TAN, T. S. “A quadratic time algorithm for the minmax length triangulation”. Anais 32º IEEE Symposium on Foundations of Computer Science, p. 414-423, 1991.

[37] EDELSBRUNNER, H., TAN, T. S. e WAUPOTITSH, R. “A polynomial time algorithm for the minmax angle triangulation”. SIAM Journal on Scientific and Statistical Computing, vol. 13, p. 994-1008, 1992.

[38] EPPSTEIN, D. “The farthest point Delaunay triangulation minimizes angles”. Computational Geometry Theory and Applications, vol. 1, p. 143-148, 1992.

[39] ESSS – Engineering Simulation and Scientific Software. “COI-lib 2.0 beta 9 - Users Manual”, 2000.

[40] ESSS – Engineering Simulation and Scientific Software. “SATER 100 - Users Manual”, 2000.

[41] FINKEL, R. A. e BENTLEY, J. L. “Quadtrees: a data structure for retrieval on composite keys”. Acta Informatic, vol. 4, p. 1-9, 1974.

[42] FORTUNE, S. “A sweepline algorithm for Voronoi diagrams”. Algorithmica, vol. 2, nº 2, p. 153-174, 1987.

[43] FORTUNE, S. “Voronoi diagrams and Delaunay triangulations”. Lecture Notes Series on Computing, vol. 1, p. 193-233, World Scientific, 1992.

[44] FORTUNE, S. e VAN WYK, C. J. “Efficient exact arithmetic for computational geometry”, Anais 9º ACM Symposium on Computational Geometry, p. 163-192, 1993.

[45] FREITAG, L. A. e OLLIVIER-GOOCH, C. “A comparison of tetrahedral mesh improvement techniques”. Anais 5º International Meshing Roundtable, Pittsburgh, PE, p. 87-100. Sandia National Laboratories, 1996.

[46] FREITAG, L. A. e OLLIVIER-GOOCH, C. “Tetrahedral mesh improvement using swapping and smoothing”. International Journal for Numerical Methods in Engineering, 1997.

[47] FREY, W. H. “Selective refinement: a new strategy for automatic node placement in graded triangular meshes”. International Journal for Numerical Methods in Engineering, vol. 24, nº 11, p. 2183-2200, 1987.

[48] FREY, W. H. e FIELD, D. A. “Mesh relaxation: a new technique for improving triangulations”. International Journal for Numerical Methods in Engineering, vol. 31, p. 1121-1133, 1991.

[49] GAREY, M. R. e JOHNSON, D. S. “Computers and intractability: a guide to the theory of NP-completeness”. W. H. Freeman, 1979.

[50] GOLD, C., CHARTERS, T. e RAMSDEN, J. “Automated contour mapping using triangular element data structure and an interpolant over each irregular triangular domain”. Anais SIGGRAPH, p. 170-175, 1977.

[51] GOLIAS, N. A. e TSIBOUKUS, T. D. “An approach to refining three-dimensional tetrahedral meshes based on Delaunay transformations”. International Journal for Numerical Methods in Engineering, vol. 37, p. 793-812, 1994.

[52] GUIBAS, L. J., KNUTH, D. E. e SHARIR, M. “Randomized Incremental construction of Delaunay and Voronoi diagrams”. Algorithmica, vol. 7, nº 4, p. 381-413, 1992.

[53] GUIBAS, L. J. e STOLFI, J. “Primitives for the manipulation of general subdivision and computation of Voronoi diagrams”. ACM Transactions on Graphics, vol. 4, nº 2, p. 74-123, 1985.

[54] HERMANN, L. R. “Laplacian-isoparametric grid generation scheme”. Journal of the Engineering Mechanics Division of the American Society of Civil Engineers, vol. 102, p. 749-756, 1976.

[55] ICEM CFD. “ICEM CFD 4.1 – users manual”, 2000.

[56] JAMESON, A., BAKER, T. J. e WEATHERILL, N. P. “Calculation of invicid transonic flow over a complete aircraft”. Anais 24º AIAA Aerospace Sciences Meeting, Reno, 1986.

[57] JAMET, P. “Estimations d’erreur pour dês elements finis droits presque degeneres”. Technical Report CRM-447, Centre d’Etudes de Limeil.

[58] JANSEN, K. “One strike against the minmax degree triangulation problem”. Computational Geometry Theory and Applications, vol. 3, p. 107-120, 1993.

[59] JOE, B. “Delaunay triangular meshes in convex polygons”. SIAM Journal on Scientific and Statistical Computing, vol. 7, p. 514-539, 1986.

[60] JOE, B. e SIMPSON, R. B. “Triangular meshes for regions of complicated shape”. International Journal for Numerical Methods in Engineering, vol. 23, p. 751-778, 1986.

[61] JONES, M. T. e PLASSMANN, P. E. “Parallel algorithms for adaptive mesh refinement”. Technical Report MCS-P421-0394, Mathematics and Computer Science Division, Argonne National Laboratory, 1994.

[62] KLEIN, R. e LINGAS, A. “A linear-time randomized algorithm for the bounded Voronoi diagram of a simple polygon”. Anais 9º ACM Symposium on Computational Geometry, p. 124-132, 1993.

[63] LAWSON, C. L. “Software for C^1 surfaces interpolation”. J. Rice, Academic Press, Mathematical Software III, p. 161-194, 1977.

[64] LEE, D-T e SCHACHTER, B. H. “Algorithms for constructing a Delaunay triangulation”. International Journal of Computer and Information Sciences, vol. 9, nº 3, p. 219-242, 1980.

[65] LEVCOPOULOS, C. e LINGAS, A. “Fast algorithms for greedy triangulation”. Anais 2º Scandinavian Workshop on Algorithm Theory, Springer-Verlag LNCS, vol. 447, p. 238-250, 1990.

[66] LINDHOLM, D. A. “Automatic triangular mesh generation on surfaces of polyhedra”. Anais IEEE Trans. Magnetics MAG-19, p. 2539-2542, 1983.

[67] LINGAS, A. “Voronoi diagrams with barriers and their applications”. Information Processing Lett., vol. 32, p. 191-198, 1989.

[68] LLOYD, E. L. “On triangulation of a set of points in the plane”. Anais 18º IEEE Symposium on Foundation of Computer Science, p. 228-240, 1977.

[69] LO, S. H. “A new mesh generation scheme for arbitrary planar domains”. International Journal for Numerical Methods in Engineering, vol. 21, p. 1403-1426, 1985.

[70] LO, S. H. “Automatic mesh generation and adaptation by using contours”. International Journal for Numerical Methods in Engineering, vol. 24, p. 1741-1756, 1987.

[71] LÖHNER, R. “Generation of three-dimensional unstructured grids by the advancing front method”. Anais AIAA 26º Aerospace Sciences Meeting, Reno, 1988.

[72] MALISKA, C. R. “Transferência de calor e mecânica dos fluidos computacional”. Rio de Janeiro, R. J., LTC – Livros Técnicos e Científicos Editora, 1995.

[73] MALISKA, C. R., CZESNAT, A. F. C., LUCIANETTI, R. M. e MALISKA JR., C. R. “Thee-dimensional multiphase flow simulation in petroleum reservoir using the mass fractions as dependent variables”, Anais 5º Latin American and Caribbean Petroleum Engineering Conference and Exhibition, 1997.

[74] MALISKA, C. R. e MALISKA JR., C. R. “A finite method using Voronoi grids for the solution of miscible displacement in porous media”, Revista Brasileira de Ciências Mecânicas – RBCM, vol. 16, nº 4, p. 415-422, 1994.

[75] MALISKA JR., C. R. “Um robusto gerador de diagramas de Voronoi para discretização de domínios irregulares”, Anais 13º CILAMCE – Congresso Ibero Latino-Americano sobre Métodos Computationais para a Engenharia, S. P., 1993.

[76] MALISKA JR., C. R., e BEZERRA, L. H., "Resolução numérica de problemas de transferência de calor em diagramas de Voronoi", Anais 17º Congresso Nacional de Matemática Aplicada e Computacional - CNMAC, pp. 449 - 450, 1994.

[77] MARCONDES, F. e MALISKA, C. R. “Convecção natural elíptica em canais de forma arbitrária”, Anais 11º Congresso Brasileiro de Engenharia Mecanica - COBEM, p 05-08, 1991.

[78] MARCONDES, F., ZAMBALDI, M. C. e MALISKA, C. R. “Simulação numérica de reservatórios de petróleo utilizando malhas de Voronoi”, Anais 5º Encontro Nacional de Ciências Térmicas, p. 335-338, 1994.

[79] MARCUM, D. L. e WEATHERILL, N. P. “Unstructured grid generation using iterative point insertion and local reconnection”, Anais 12º AIAA Applied Aerodynamics Conference , Colorado Springs, Colorado, nº AIAA 94-1926, 1994.

[80] MAVRIPLIS, D. J. “An Advancing front Delaunay triangulation algorithm designed for robustness”. Technical Report 92-49, ICASE, 1992.

[81] MAVRIPLIS, D. J. “Unstructured and adaptive mesh generation for high Reynolds number viscous flows”. ICASE Report 91-25, NASA Langley Research Center, 1991.

[82] MELISSARATOS, E. e SOUVAINE, D. “Coping with inconsistencies: A new approach to produce quality triangulations of polygonal domains with holes”. Anais 8º ACM Symposium on Computational Geometry, p. 202-211, 1992.

[83] MITCHELL, S. A. “Refining a triangulation of a planar straight-line graph to eliminate large angles”. Anais 34º IEEE Symposium on Foundations of Computer Science, p. 583-591, 1993.

[84] MITCHELL, S. A. e VAVASIS, S. “Quality mesh generation in three dimensions”. Anais 8º ACM Symposium on Computation Geometry”, p. 212-221, 1992.

[85] MOUNT, D. M. e SAALFELD, A. “Globally-equilangular triangulations of cocircular points in O(nlogn) time”. Anais 4º ACM Symposium on Computational Geometry, p. 143-152, 1988.

[86] NACKMAN, L. R. e SRINIVASAN, V. “Point placement for Delaunay triangulation of polygonal domains”. Anais 3º Canadian Conference on Computational Geometry, p. 37-40, 1991.

[87] PALAGI, C. “Generation and application of Voronoi grid to model flow in heterogeneous reservoir”, Ph.D. Thesis, Stanford University, California, EUA, 1992.

[88] RIVARA, M. C. “Algorithms for refining triangular grids suitable for adaptive and multigrid techniques”. International Journal on Numerical Methods for Engineering, vol. 20, p. 745-756, 1984.

[89] RIPPA, S. “Minimal roughness property of the Delaunay triangulation”. CAGD, vol. 7, p. 489-497, 1990.

[90] RIPPA, S. e SCHIFF, B. “Minimun energy triangulations for elliptic problems”. Computer Mething in Applied Mechanics and Engineering, vol. 84, p. 257-274, 1990.

[91] RIVARA, M. C., “A discussion on the triangulation refinement problem”. Anais 5º Canadian Conference on Computational Geometry, p. 42-47, 1993.

[92] RUPPERT, J. “A Delaunay refinement algorithm for quality 2-dimensional mesh generation”. Journal of Algorithms, vol. 18, nº 3, p. 548-585, 1995.

[93] SAALFELD, A. “Delaunay edge refinements”. Anais 3º Canadian Conference on Computational Geometry, p. 33-36, 1991.

[94] SAMET, H. “The quadtree and related hierarchical data structures”. Computing Surveys, vol. 16, p. 188-260, 1984.

[95] SAMET, H. “The Design and analysis of spatial data structures”. Addison-Wesley, 1990.

[96] SEIDEL, R. “Constrained Delaunay triangulations and Voronoi diagrams with obstacles”. 1987-1988 Ten Years IIG, p. 178-191, 1988.

[97] SHAMOS, M. I. e HOEY, D. “Closest-point problems”. Anais 16º IEEE Symposium on Foundations of Computer Science, p. 151-162, 1975.

[98] SHAW, J. G. Xerox Webster Center, Webster, New York.

[99] SHEWCHUK, J. R. “Lecture notes on Delaunay mesh generation”, Technical Report, Department of Electrical Engineering and Computer Science, University of California at Berkley, 1999.

[100] SHEWCHUK, J. R. “Robust adaptive floating-point geometric predicates”. Anais 12º ACM Symposium on Computational Geometry, 1996.

[101] SHEWCHUK, J. R. “Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator”, 1º ACM Workshop on Applied Computational Geometry, 1996.

[102] SMITH, B. F., BJORSTAD, P. E. e GROPP, W. D. “Domain decomposition – parallel multilevel methods for elliptic partial differential equations”, Cambridge University Press, N.Y., 1996.

[103] SRINIVASAN, V., NACKMAN, L. R., TANG, J-M. e MESHKAT, S. N. “Automatic mesh generation using the symmetric axis transformation of polygonal domains”. Technical Report RC 16132, Computer Science, IBM Research Division, Yorktown Heights, NY, 1990.

[104] STRANG, G. e FIX, G. J. “An Analysis of the Finite Element Method”. Prentice-Hall, 1973.

[105] TAN, T. S. “An optimal bound for conforming quality triangulations”. Anais 10º ACM Symposium on Computational Geometry, p. 240-249, 1994.

[106] THACKER, W. C. “A brief review of techniques for generating irregular computational grids”. International Journal for Numerical Methods in Engineering, nº 15, p. 1335-1341, 1980.

[107] THOMPSON, J. F. “Numerical grid generation”. North-Holland, 1982.

[108] THOMPSON, J. F. e WARSI, Z. U. A. “Numerical grid generation: foundations and applications”. North-Holland, 1985.

[109] T-SURF. “gOcad – users manual”, 2000.

[110] VORONOI, G. “Nouvelles applications des param`etres continus `a las th'eorie des formes quadratiques”. J. Reine Angew. Math., vol. 133, p. 97-178, 1907.

[111] WANG, C. e SCHUBERT, L. “An optimal algorithm for constructing the Delaunay triangulation of a set of line segments”. Anais 3º ACM Symposium on Computational Geometry, p. 223-232, 1987.

[112] WATSON, D. F. “Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes”. Computer Journal, vol. 24, nº 2, p. 167-172, 1981.

[113] WILLIAMS, R. D. “Adaptive parallel meshes with complex geometry”. Technical Report CRPC-91-2, Center for Research on Parallel Computation, California Institute of Technology, 1991.

[114] WINSLOW, A. M. “An irregular triangle mesh generator”. Report UCXRL-7880, National Technical Information Service, Springfield, VA, 1964.