Pontifícia Universidade Católica
do Rio de Janeiro

Bruno Nogueira Machado

Desenvolvimento de um gerador de malhas Delaunay em três dimensões

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática, do Departamento de Informática da PUC-Rio.

Orientador : Prof. Waldemar Celes Filho
Coorientador: Dr. Rodrigo Espinha

Rio de Janeiro
Abril de 2021

Pontifícia Universidade Católica
do Rio de Janeiro

Bruno Nogueira Machado

Desenvolvimento de um gerador de malhas Delaunay em três dimensões

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática da PUC-Rio. Aprovada pela Comissão Examinadora abaixo:

Prof. Waldemar Celes Filho
Orientador
Departamento de Informática – PUC-Rio

Dr. Rodrigo Espinha
Coorientador
Instituto Tecgraf – PUC-Rio

Prof. Marcelo Gattass
Departamento de Informática – PUC-Rio

Prof. Luiz Henrique de Figueiredo
IMPA

Prof. Luiz Fernando Campos Ramos Martha
Departamento de Engenharia Civil – PUC-Rio

Rio de Janeiro, 30 de Abril de 2021

Bruno Nogueira Machado

Graduado em Engenharia Naval e Oceânica pela Universidade Federal do Rio de Janeiro (2016). Em 2018 ingressou no programa de mestrado em Informática na Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio) e em 2019 começou o trabalho como pesquisador bolsista no Grupo de Visualização do Instituto Tecgraf.

Ficha Catalográfica

Nogueira Machado, Bruno

Desenvolvimento de um gerador de malhas Delaunay em três dimensões / Bruno Nogueira Machado; orientador: Waldemar Celes Filho; coorientador: Rodrigo Espinha. – 2021.

64 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2021.

Inclui bibliografia

1. Informática – Teses. 2. Geração de malhas. 3. Triangulação de Delaunay. 4. Refinamento de malhas. I. Celes Filho, Waldemar. II. Espinha, Rodrigo. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. IV. Título.

CDD: 004

Agradecimentos

Ao meu orientador Waldemar, e ao meu co-orientador Rodrigo, pela confiança e apoio durante todo o processo.

Aos colegas do grupo de Visualização do Tecgraf, pelo companheirismo durante esse período. Em especial, à Alice, Andrey, Fernanda e Gisele, pelas infinitas discussões sobre geometria.

À minha família, que sempre me proporcionou oportunidades de estudo e apoio incondicional para alcançar meus objetivos.

Aos amigos que estiveram presentes, em especial, à Bianca, Gabrielle, Maurício e Tereza, que me apoiaram e me deram motivação para realizar este trabalho.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

Resumo

Nogueira Machado, Bruno; Celes Filho, Waldemar; Espinha, Rodrigo. **Desenvolvimento de um gerador de malhas Delaunay em três dimensões**. Rio de Janeiro, 2021. 64p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Malhas são amplamente usadas na discretização de domínios geométricos em aplicações na engenharia, como simulações de fluxo, transmissão de calor e deformação mecânica. O problema de geração de malhas é bem conhecido e estudado, mas a geração automática de malhas para um domínio físico com geometrias complexas, criando elementos que obedeçam a forma do objeto, e de tamanho e qualidade adequados, ainda é um desafio. Neste trabalho, foram estudados e implementados métodos para gerar malhas com restrições arbitrárias. O gerador implementado é do tipo de Delaunay, que constrói malhas Delaunay com restrições, e utiliza as propriedades da malha para inserir novos vértices e melhorar a qualidade dos elementos.

Palavras-chave

Geração de malhas; Triangulação de Delaunay; Refinamento de malhas.

Abstract

Nogueira Machado, Bruno; Celes Filho, Waldemar (Advisor); Espinha, Rodrigo (Co-Advisor). **Development of a Delaunay mesh generator in three dimensions**. Rio de Janeiro, 2021. 64p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Meshes are widely used in the discretization of geometric domains for engineering applications such as fluid flow simulator, heat transfer simulations and mechanical deformation. The mesh generation problem is well known and studied, nevertheless the automatic generation of meshes to domains with complex geometry, creating elements that conform to the forms, and of adequate size and quality, is still a challenge. In this work, mesh generation methods capable of generation mesh of arbitrary restrictions were studied and implemented. The implemented generator is a Delaunay generator, which constructs constrained Delaunay meshes, and utilizes the properties of the mesh to insert new vertices and improve the quality of the elements.

Keywords

Mesh generation; Delaunay Triangulation; Mesh refinement.

Sumário

1 Introdução **13**

2 Malhas **15**
2.1 Piecewise Linear Complex 15
2.2 Critérios de qualidade 16
2.3 Malhas de Delaunay 17
2.4 Malhas de Delaunay conformadas 19
2.5 Malhas de Delaunay com restrição 20
2.6 Refinamento de Delaunay 22

3 Trabalhos Relacionados **24**
3.1 Algoritmos de geração de malha 28
3.2 Geradores de malha 28

4 Gerador de malhas **30**
4.1 TopS 30
4.2 Predicados robustos 30
4.3 Gerador de malhas 2D 31
4.3.1 Delaunay 31
4.3.1.1 Grafo de conflito 32
4.3.1.2 *Walking* 32
4.3.2 Delaunay com restrições 35
4.3.2.1 Retriangulação das cavidades 35
4.3.2.2 Bistellar Flip 35
4.3.3 Refinamento 36
4.3.3.1 *Chew's Second* 37
4.3.3.2 *Terminator* e *Terminator Chew* 38
4.4 Gerador de malhas 3D 38
4.4.1 Perturbação numérica 38
4.4.2 Estruturas auxiliares 39
4.4.3 Triangulação de Delaunay 39
4.4.4 Delaunay com restrições 40
4.4.4.1 Recuperação das arestas 40
4.4.4.2 Recuperação das faces 41
4.4.5 Refinamento 41

5 Resultados **42**
5.0.1 Gerador em 2D 42
5.0.1.1 TECVis 42
5.0.1.2 Arestas de restrição com ângulos pequenos 45
5.0.1.3 Meio fraturado 49
5.0.1.4 Análise de desempenho 54
5.0.2 Gerador em 3D 55
5.0.2.1 Bunny 55

5.0.2.2 Meio fraturado 57
5.0.2.3 Modelo de reservatório 59

6 Conclusão 61

Referências bibliográficas 63

Lista de figuras

Figura 1.1 Exemplos de malhas [1] de triângulos em 2D (a) e tetraedros em 3D (b). 13

Figura 2.1 Exemplo de um PLC. [10] 15
Figura 2.2 Exemplo de interseções não permitidas em um PLC. [10] 16
Figura 2.3 Um exemplo de um tetraedro do tipo *sliver*, no qual três vértices estão sobre a linha do hemisfério e um está na superfície da esfera pouco abaixo da linha. 17
Figura 2.4 Exemplo de uma malha de Delaunay em 2D gerada a partir de um conjunto de vértices. 18
Figura 2.5 Exemplo de uma malha Delaunay em 3D gerada a partir de alguns vértices (à esquerda) e uma malha não-Delaunay com qualidade melhor (à direita). 19
Figura 2.6 Exemplo de uma malha Delaunay conformada em 2D gerada a partir de um conjunto de vértices e arestas de restrição. 20
Figura 2.7 Exemplo de uma malha *Constrained Delaunay* gerada a partir de alguns vértices e arestas de restrição. 20
Figura 2.8 O poliedro de Schönhardt é formado rotacionando o topo de um prisma triangular. Não existe triangulação possível para esse poliedro. [6] 21
Figura 2.9 Exemplo de uma malha *Refined Delaunay* gerada a partir de uma CDT. 22
Figura 2.10 CDT de um PLC com ângulos pequenos entre restrições (à esquerda) e uma uma triangulação *Refined Delaunay* do mesmo PLC (à direita). 23

Figura 3.1 Um exemplo de uma malha gerada por um algoritmo do tipo grid. [12] 25
Figura 3.2 Um exemplo de uma malha gerada pelo primeiro algoritmo de Chew. [12] 25
Figura 3.3 Um exemplo de uma malha gerada pelo primeiro algoritmo de Ruppert. [12] 26
Figura 3.4 Área limitada pelas lentes diametrais. 27
Figura 3.5 PLC de entrada (à esquerda) e malha gerada pelo software Triangle (à direita). [8] 29
Figura 3.6 Malha gerada pelo software TetGen. [10] 29

Figura 4.1 Ilustração de um grafo de conflito. 33
Figura 4.2 Tipos de localização *walking* [13] 33
Figura 4.3 Ilustração da distribuição dos vértices entre as rodadas. 34
Figura 4.4 Curvas de Hilbert com graus variados. [15] 34
Figura 4.5 Um exemplo de inserção de um vértice que está fora da triângulação regular (vazado), mas em conflito com um triângulo fantasma. O ponto preto é o vértice fantasma. [6] 35
Figura 4.6 Ilustração das etapas do algoritmo de inserção de arestas. 36

Figura 4.7 Algoritmo *Bistellar Flip* ilustrado em 2D. [6] 37
Figura 4.8 Exemplo de uma malha Delaunay em 3D gerada a partir de 1000 vértices. 40

Figura 5.1 Arestas e vértices do PLC. 43
Figura 5.2 CDT do PLC. 43
Figura 5.3 Malha refinada utilizando o algoritmo *Chew's Second*. (Ângulo mínimo permitido = 20°) 43
Figura 5.4 Malha refinada utilizando o algoritmo *Terminator Chew*. (Ângulo mínimo permitido = 20°) 44
Figura 5.5 Malha refinada utilizando o algoritmo *Terminator*. (Ângulo mínimo permitido = 20°) 44
Figura 5.6 Malha refinada utilizando o algoritmo *Terminator*. (Ângulo mínimo permitido = 20°; Comprimento de aresta máximo = 0.2) 44
Figura 5.7 Arestas e vértices do PLC. 45
Figura 5.8 CDT do PLC. 46
Figura 5.9 Malha refinada utilizando o algoritmo *Chew's Second*. (Ângulo mínimo permitido = 20°) 46
Figura 5.10 Malha refinada utilizando o algoritmo *Terminator Chew*. (Ângulo mínimo permitido = 20°) 47
Figura 5.11 Malha refinada utilizando o algoritmo *Terminator*. (Ângulo mínimo permitido = 20°) 47
Figura 5.12 Malha refinada utilizando o algoritmo *Terminator*. (Ângulo mínimo permitido = 20°; Comprimento de aresta máximo = 1) 48
Figura 5.13 Arestas e vértices do PLC. 49
Figura 5.14 CDT do PLC. 49
Figura 5.15 Malha refinada utilizando o algoritmo *Chew's Second*. (Ângulo mínimo permitido = 20°) 50
Figura 5.16 Malha refinada utilizando o algoritmo *Terminator Chew*. (Ângulo mínimo permitido = 20°) 50
Figura 5.17 Malha refinada utilizando o algoritmo *Terminator*. (Ângulo mínimo permitido = 20°) 50
Figura 5.18 Malha refinada utilizando o algoritmo *Terminator*. (Ângulo mínimo permitido = 20°; Comprimento de aresta máximo = 200) 51
Figura 5.19 Histograma da CDT do PLC. 52
Figura 5.20 Histograma da malha refinada utilizando o algoritmo *Terminator*. (Ângulo mínimo permitido = 20°) 53
Figura 5.21 Histograma da malha refinada utilizando o algoritmo *Terminator*. (Ângulo mínimo permitido = 20°; Comprimento de aresta máximo = 200) 53
Figura 5.22 Análise de desempenho dos algoritmos implementados em comparação com os algoritmos do software Triangle [8].) 54
Figura 5.23 Arestas, vértices e faces do PLC. 55
Figura 5.24 CDT do PLC. 56

Figura 5.25 CDT do PLC. Alguns tetraedros foram removidos utilizando um plano de corte, para permitir a visualização do interior da malha. 56
Figura 5.26 Arestas, vértices e faces do PLC. 57
Figura 5.27 Arestas, vértices e faces do PLC, com a face do topo omitida para permitir a visualização das faces de restrição internas. 57
Figura 5.28 CDT do PLC. 58
Figura 5.29 CDT do PLC. Alguns tetraedros foram removidos utilizando um plano de corte, para permitir a visualização do interior da malha. 58
Figura 5.30 Arestas, vértices e faces do PLC. 59
Figura 5.31 CDT do PLC. 60
Figura 5.32 CDT do PLC. Alguns tetraedros foram removidos utilizando um plano de corte, para permitir a visualização do interior da malha. 60

Lista de Abreviaturas

CDT – *Constrained Delaunay Triangulation*
PLC – *Piecewise Linear Complex*
BRIO – *Biased Randomized Insertion Order*

1
Introdução

Métodos numéricos são extremamente importantes atualmente nos projetos e análises em diversas áreas do conhecimento. Destacam-se os usos nas simulações de escoamento de fluidos, transmissão de calor e simulações de fadiga e deformação. Para que as simulações produzam resultados acurados é necessário que o domínio geométrico seja representado adequadamente, utilizando elementos de boa qualidade e que representem fielmente as restrições e contorno do domínio.

Apesar de ser um problema bem conhecido, a geração automática de malhas ainda é um gargalo nos processos de simulação numérica, principalmente para modelos complexos. A malha gerada deve se adequar às restrições geométricas impostas utilizando elementos de melhor qualidade possível. Além disso, com o crescimento do tamanho dos modelos, é cada vez mais necessária a busca por soluções eficientes e robustas. A Figura 1.1 mostra dois exemplos de malhas, uma em duas dimensões e outra em três dimensões, que foram geradas a partir das restrições geométricos que definem os objetos representados.

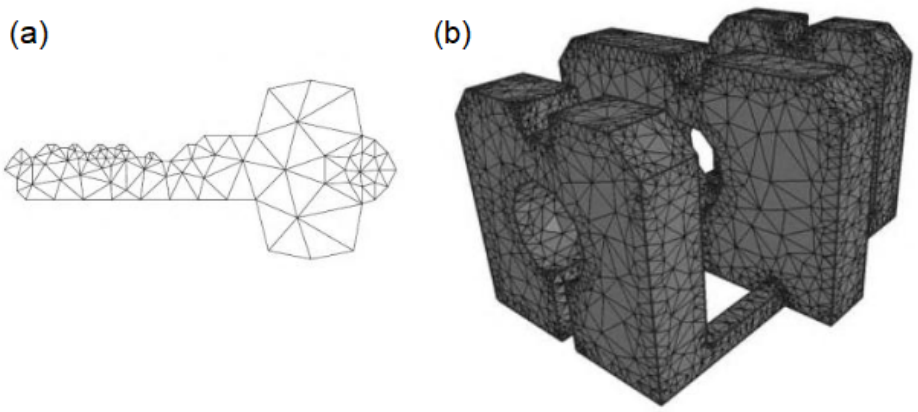

Figura 1.1: Exemplos de malhas [1] de triângulos em 2D (a) e tetraedros em 3D (b).

Este trabalho tem como objetivo implementar um gerador de malhas de tetraedros em três dimensões que seja capaz de satisfazer os requisitos de domínios complexos. Foram estudados os tipos de geradores existentes na literatura, em especial os de malhas de Delaunay, cujas malhas geradas tem propriedades geométricas relevantes para simulação numérica. Foi implemen-

tado um gerador de malhas Delaunay de triângulos em duas dimensões, com dois métodos de localização, e três opções de refinamento. Em três dimensões, foi implementado um gerador de malhas Delaunay de tetraedros, que recebe como entrada um conjunto de faces triangulares e vértices.

Esta dissertação está organizada da seguinte forma: no Capítulo 2 descrevemos alguns aspectos básicos relacionados à geração de malhas de Delaunay. No Capítulo 3 descrevemos alguns dos trabalhos relacionados dentro da área de geração de malhas. No Capítulo 4 detalhamos os algoritmos implementados, seguido pela apresentação dos resultados no Capítulo 5. Por último, no Capítulo 6, são feitas considerações finais e apresentados possíveis trabalhos futuros.

2
Malhas

As malhas podem ser classificadas de acordo com o número de dimensões e os tipos de elementos. Este trabalho considera malhas de politopos simpléticos em duas e três dimensões (triângulos e tetraedros, respectivamente). Foram estudadas malhas não estruturadas, nas quais os vértices podem estar distribuídos de qualquer forma dentro da malha, e podem representar qualquer área ou volume.

A seguir serão apresentados: a definição de um tipo de conjunto que será utilizado como entrada do algoritmo de geração, um critério para avaliar a qualidade dos elementos de uma malha, e as propriedades das malhas de Delaunay.

2.1
Piecewise Linear Complex

Um Complexo de Partes Lineares (*Piecewise Linear Complex* (PLC)) é um conjunto de vértices, arestas e faces sujeitos a algumas restrições, e suas características permitem que seja utilizado para definir as restrições de um domínio. Por esse motivo, esse é o tipo de conjunto utilizado como entrada para o algoritmo de geração de malha. Qualquer poliedro é um PLC, mas um PLC pode representar objetos mais genéricos, como por exemplo o objeto da Figura 2.1.

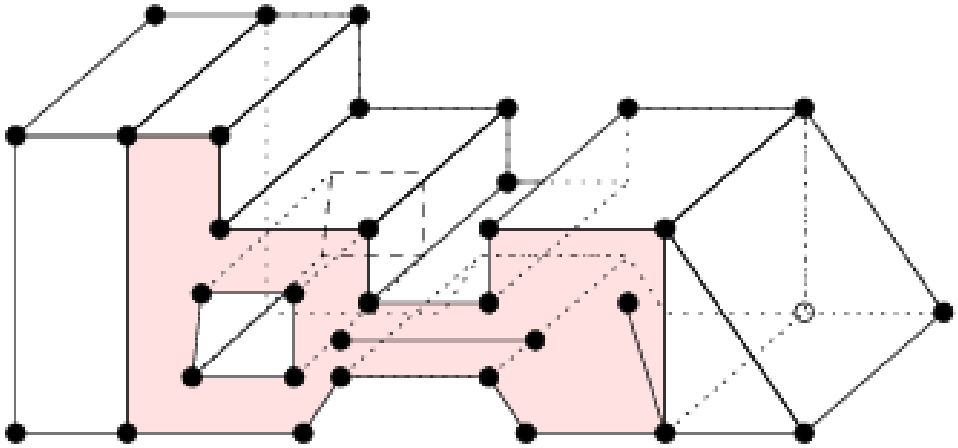

Figura 2.1: Exemplo de um PLC. [10]

Os elementos desse complexo só podem ter interseção sobre outro elemento: duas arestas só podem se interceptar em um ponto do complexo, duas faces só podem se interceptar em uma aresta (ou vértice) do complexo. A Figura 2.2 mostra duas interseções que não são permitidas em um PLC.

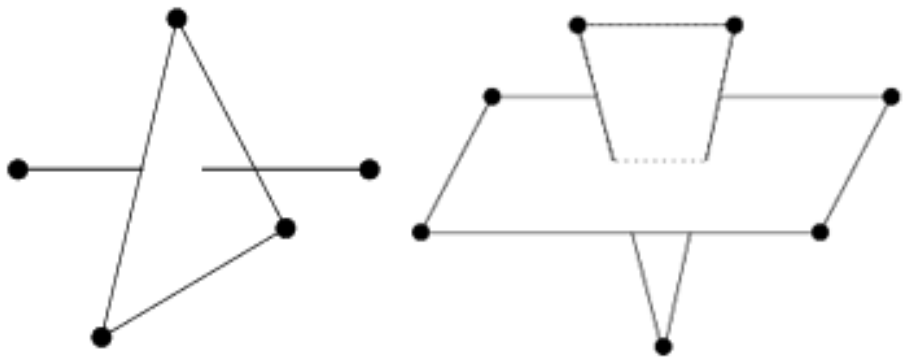

Figura 2.2: Exemplo de interseções não permitidas em um PLC. [10]

2.2 Critérios de qualidade

Para utilização em métodos de elementos finitos, é desejável que a malha tenha uma qualidade adequada, de forma a não introduzir erros de discretização grandes na solução. A avaliação de qualidade pode ser feita a partir de critérios de forma e tamanho dos elementos: ângulos muito grandes (próximos de 180°) podem causar erros de interpolação grandes, enquanto que ângulos muito pequenos (próximos de 0°) podem causar erros nas definições das matrizes de rigidez associadas. Elementos menores aumentam a acurácia do método ao custo de aumentar o esforço computacional.

Um dos critérios comumente utilizados para avaliar a qualidade de um elemento é a razão entre o raio R do circumdisco (2D) ou da circunesfera (3D) e o comprimento da menor aresta l_{min} do elemento [6]. Esse critério, chamado de *radius-edge ratio*, varia de ∞ para elementos degenerados até $1/\sqrt{3}$ (aproximadamente 0.577) para um triângulo equilátero, e tem valor igual a 0.612 para um tetraedro equilátero.

Para malhas de triângulos em duas dimensões, o critério é uma ótima forma de avaliação, já que está diretamente relacionado ao menor ângulo do triângulo θ_{min} pela seguinte fórmula:

$$\frac{R}{l_{min}} = \frac{1}{2\sin\theta_{min}} \tag{2-1}$$

Além disso, o maior ângulo do triângulo vai ser no máximo $180° - 2\theta_{min}$. Portanto definir um limite máximo para o *radius-edge ratio* significa definir

um limite tanto do ângulo mínimo quanto do ângulo máximo do triângulo. Por isso, esse parâmetro tem sido utilizado com sucesso nos algoritmos de refinamento em duas dimensões.

Entretanto, o parâmetro não é tão eficiente em três dimensões. Um refinamento baseado nesse parâmetro consegue remover da malha com sucesso a maioria dos tipos de elementos degenerados menos um: o *sliver*. Esse tipo de elemento é composto por quatro vértices arranjados próximos do hemisfério de uma esfera, praticamente coplanares, como exemplificado na Figura 2.3. O *radius-edge ratio* de um *sliver* pode ser igual a $1/\sqrt{2}$ (aproximadamente 0.707), muito próximo do valor para um tetraedro equilátero, e ainda assim ter ângulos próximos de 180°.

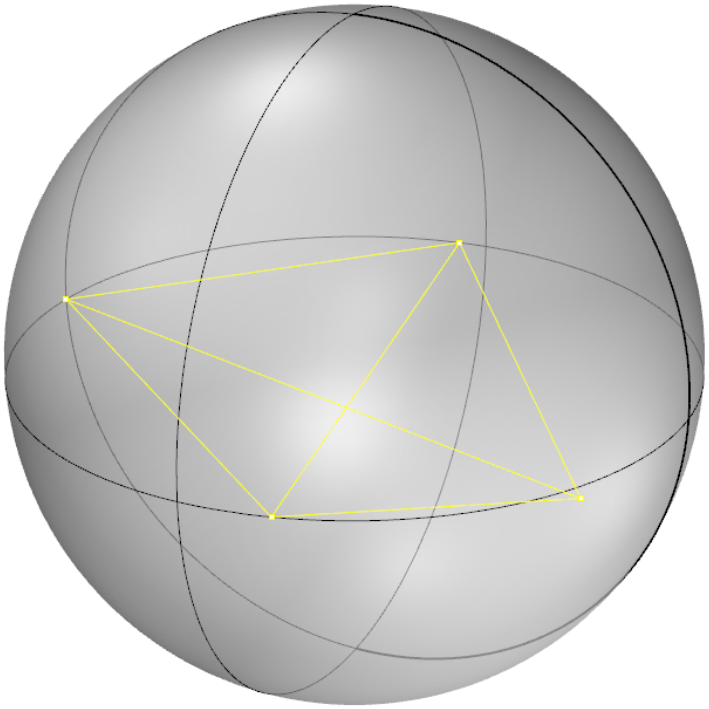

Figura 2.3: Um exemplo de um tetraedro do tipo *sliver*, no qual três vértices estão sobre a linha do hemisfério e um está na superfície da esfera pouco abaixo da linha.

Mesmo com essa limitação, o parâmetro ainda é utilizado no refinamento de malhas em três dimensões, já que remove grande parte dos tetraedros de qualidade ruim.

2.3 Malhas de Delaunay

Triangulações de Delaunay [1] são estruturas geométricas muito utilizadas na geração de malhas por causa de suas propriedades. Em duas dimensões,

a triangulação de Delaunay é caracterizada pela propriedade do circumdisco vazio: um triângulo da malha é Delaunay se nenhum vértice da malha estiver no interior do seu circumdisco aberto, e uma malha é Delaunay se todos os triângulos forem Delaunay. Todo conjunto de vértices tem uma triangulação de Delaunay, e ela é única se os vértices forem *genéricos*, ou seja, se nenhum conjunto de quatro pontos estiver sobre a mesma circunferência. A Figura 2.4 ilustra um conjunto de vértices genéricos e a malha de Delaunay desse conjunto.

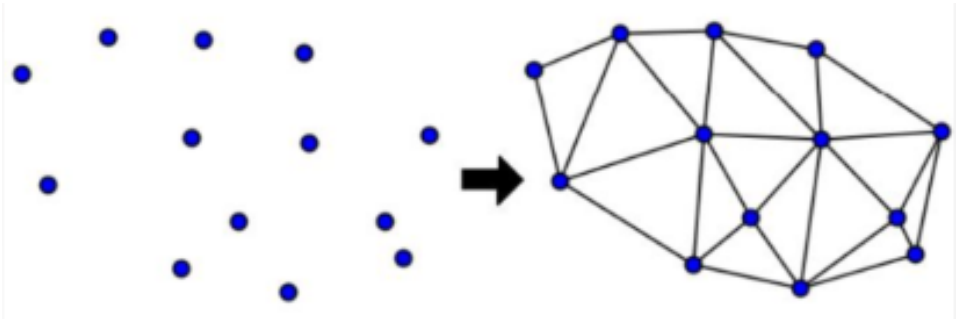

Figura 2.4: Exemplo de uma malha de Delaunay em 2D gerada a partir de um conjunto de vértices.

Um triângulo é *strongly Delaunay* se nenhum vértice da malha estiver no interior do seu circumdisco fechado, com exceção dos vértices do triângulo. Uma aresta é *strongly Delaunay* se existir ao menos um circumdisco fechado que não contenha nenhum vértice da malha, com exceção dos vértices de aresta.

Dentre todas as triangulações possíveis de um conjunto de vértices, a triangulação de Delaunay no plano é a que otimiza os seguintes critérios geométricos:

- Maximiza o ângulo mínimo.
- Minimiza o maior circumdisco.
- Minimiza o maior *min-containment disk*, menor circunferência que contém os vértices do triângulo.

Essas propriedades tornam malhas de Delaunay excelentes opções para utilização em métodos de elementos finitos, por minimizar os erros de interpolação e deixar a simulação estável.

Em três dimensões, a triangulação de Delaunay é caracterizada pela propriedade da circunsfera vazia: nenhum vértice da malha está no interior da circunsfera de qualquer tetraedro da malha. Entretanto, dos três critérios otimizados em duas dimensões, somente um se estende para três dimensões: ela minimiza a maior *min-containment sphere*, menor esfera que contém os vértices

do tetraedro. O ângulo mínimo portanto não é otimizado numa triangulação Delaunay em três dimensões. A Figura 2.5 mostra uma triangulação alternativa que tem ângulo mínimo maior do que a triangulação de Delaunay. Ainda assim, a triangulação de Delaunay continua sendo popular como etapa inicial de algoritmos de geração de malha em três dimensões. A triangulação também otimiza outras propriedades além das apontadas.

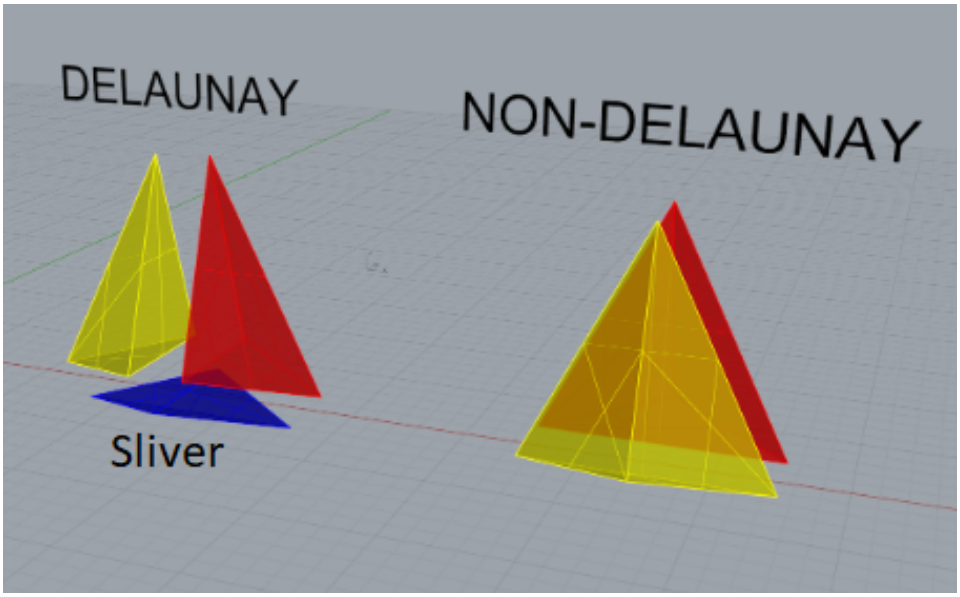

Figura 2.5: Exemplo de uma malha Delaunay em 3D gerada a partir de alguns vértices (à esquerda) e uma malha não-Delaunay com qualidade melhor (à direita).

Apesar de serem muito úteis, as malhas de Delaunay não resolvem os problemas de geração de malha por si só, já que muitas vezes arestas e faces de restrições, como fronteiras com condições de contorno ou arestas internas, podem não estar representadas na malha. Existem duas soluções que podem ser usadas para resolver isso: as *malhas de Delaunay conformadas* e as *malhas de Delaunay com restrições*.

2.4 Malhas de Delaunay conformadas

A triangulação de Delaunay conformada resolve o problema das restrições não representadas inserindo novos vértices que subdividem as restrições até que todas as sub-arestas estejam na triangulação. O resultado é uma triangulação onde todos os elementos são Delaunay e todas as restrições são representadas, embora possivelmente subdivididas.

A vantagem dessa solução é que, como todos os elementos são Delaunay, os circuncentros desses elementos estão necessariamente dentro da malha, e o

Diagrama de Voronoi Dual é bem comportado. A desvantagem é que pode ser necessário inserir novos vértices na triangulação nessa etapa, como mostrado na Figura 2.6, onde dois novos vértices tiveram que ser inseridos para garantir a representação das arestas de restrição. Os novos vértices inseridos, que não faziam parte do conjunto inicial, são chamados de vértices de *Steiner*.

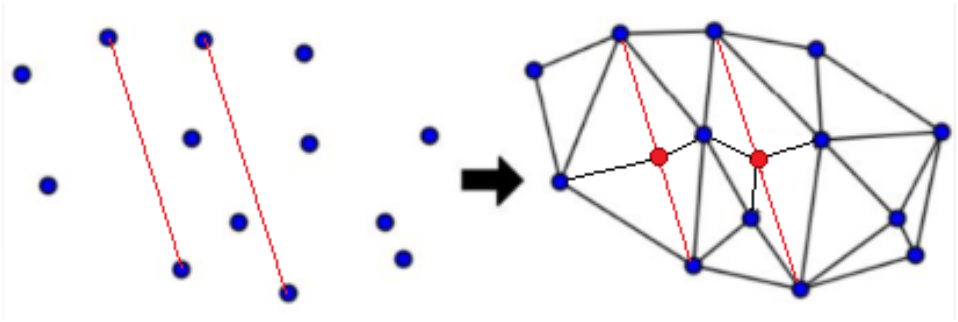

Figura 2.6: Exemplo de uma malha Delaunay conformada em 2D gerada a partir de um conjunto de vértices e arestas de restrição.

2.5 Malhas de Delaunay com restrição

Triangulações de Delaunay com restrição (*Constrained Delaunay Triangulation (CDT)*) são uma variação da triangulação de Delaunay, na qual algumas propriedades de Delaunay são relaxadas localmente para incluir um conjunto de restrições. Apesar de não respeitar o critério de Delaunay para todas as células, dentre todas as triangulações que contêm as dadas restrições, a CDT é a triangulação que otimiza as mesmas propriedades geométricas que uma triangulação Delaunay. Em duas dimensões, qualquer PLC genérico tem uma única triangulação CDT, sem a necessidade de inserir novos vértices na triangulação. Por esses motivos, a CDT é uma solução adequada para geração de malhas em duas dimensões. A Figura 2.7 mostra um PLC e a CDT correspondente.

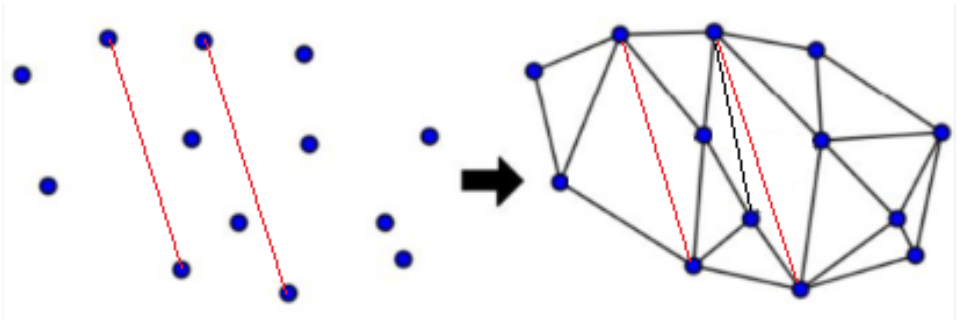

Figura 2.7: Exemplo de uma malha *Constrained Delaunay* gerada a partir de alguns vértices e arestas de restrição.

Em três dimensões as malhas de Delaunay com restrição também são relevantes para geração de malha, porém com algumas dificuldades adicionais. Nem todo PLC genérico em três dimensões tem uma CDT, e mesmo alguns poliedros simples não têm triangulação possível sem que seja necessário incluir novos vértices. Esse é o caso do poliedro de Schönhardt mostrado na Figura 2.8. Além disso, é NP-difícil determinar se um PLC tem uma triangulação ou não.

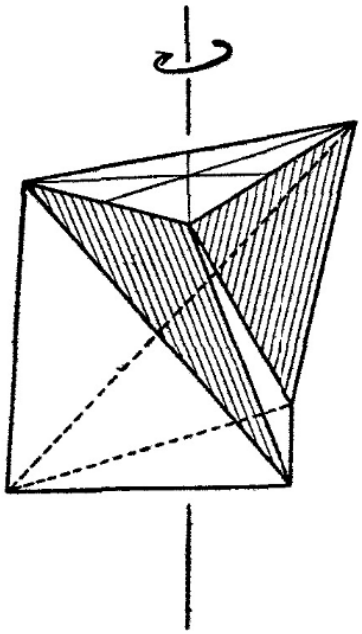

Figura 2.8: O poliedro de Schönhardt é formado rotacionando o topo de um prisma triangular. Não existe triangulação possível para esse poliedro. [6]

Existe uma condição suficiente (mas não necessária) para garantir que um PLC possua uma CDT, chamada *edge-protected* [6]. Um PLC é *edge-protected* se todas as arestas de restrição forem *strongly Delaunay*. Para verificar isso basta testar se elas estão incluídas na triangulação de Delaunay do PLC (assumindo que o PLC é genérico). Se todas as arestas estiverem presentes no Delaunay, o PLC é *edge-protected*, portanto é garantido que existe uma CDT que contém todos os vértices e arestas de restrição. As faces de restrição não são relevantes para o teste da condição *edge-protected*.

A condição *edge-protected* será utilizada no algoritmo para garantir que a construção de uma CDT seja possível, mas para isso pode ser necessário subdividir arestas de restrição inserindo novos vértices, e portanto a malha resultante vai ser uma Steiner CDT do conjunto inicial, uma CDT do conjunto aumentado com os vértices inseridos.

2.6 Refinamento de Delaunay

Com o intuito de melhorar a qualidade das malhas para que possam ser utilizadas em modelos de elementos finitos, é comum realizar uma etapa de refinamento na malha. Essa etapa consiste em inserir novos vértices para subdividir de forma mais regular a malha e obter elementos de maior qualidade.

Um algoritmo genérico de refinamento consiste em gerar uma malha Delaunay conformada ou Delaunay com restrição e, então, testar todos os triângulos da malha, marcando aqueles com qualidade ruim. O critério pode ser baseado na forma, como por exemplo a razão definida na Equação 2-1, ou de tamanho máximo do triângulo (maior aresta, área, raio do circumdisco). Enquanto existirem triângulos abaixo da qualidade desejada, o algoritmo deve inserir novos pontos na malha pra remover esses triângulos. Uma das vantagens dos algoritmos de geração de malha de Delaunay é que a posição dos novos pontos é definida usando a triangulação parcial, permitindo ao algoritmo tomar decisões mais eficientes para atingir a qualidade desejada.

A Figura 2.9 ilustra o processo de refinamento da malha, no qual novos vértices são inseridos na triangulação para melhorar a qualidade da malha. O principal trabalho do algoritmo de refinamento é decidir as posições dos novos vértices.

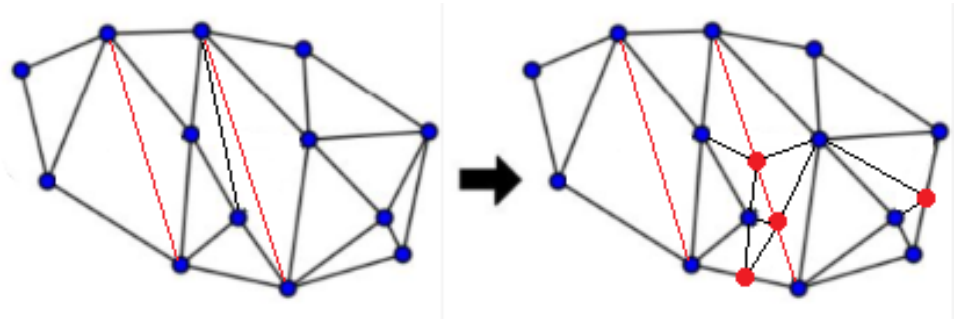

Figura 2.9: Exemplo de uma malha *Refined Delaunay* gerada a partir de uma CDT.

Uma das grandes dificuldades para os algoritmos de refinamentos é tratar PLCs que tenham restrições com ângulos pequenos entre elas. Ângulos pequenos afetam negativamente a qualidade dos elementos próximos a eles, e não podem ser removidos da triangulação por estarem associados às restrições. Os algoritmos devem então aceitar que tais ângulos não podem ser melhorados e tomar decisões durante a inserção de pontos para diminuir o impacto dessas restrições nas qualidades dos elementos próximos. A figura 2.10 mostra o resultado da geração de uma CDT e de uma triangulação *Refined Delaunay*: as

arestas de restrição (linhas brancas) formam ângulos pequenos que não podem ser removidos da malha mesmo após o refinamento.

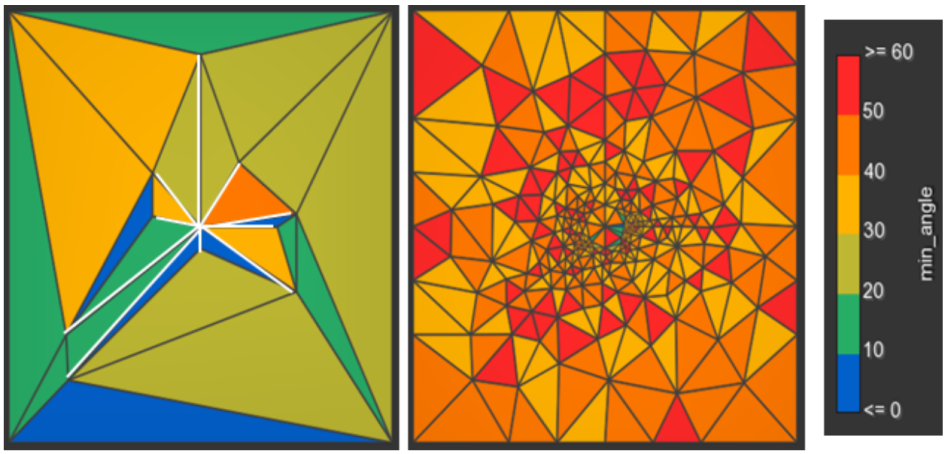

Figura 2.10: CDT de um PLC com ângulos pequenos entre restrições (à esquerda) e uma uma triangulação *Refined Delaunay* do mesmo PLC (à direita).

3
Trabalhos Relacionados

O estudo de geradores de malhas automáticos começou em 1970, quando os primeiros artigos sobre algoritmos de avanço de frente (*advancing front*) e geração de malhas de Delaunay começaram a ser publicados. Uma terceira classe de geradores de malha apareceu depois, por volta de 1983: a classe de Grid, Quadtree e Octree.

O método de avanço de frente consiste em começar a triangular de fora para dentro, começando pelas fronteiras, o que garante uma qualidade boa dos elementos próximos à fronteira. Entretanto, o algoritmo tem dificuldades em construir elementos de boa qualidade no interior, onde acontecem os encontros das frentes, principalmente em três dimensões. É bastante utilizado em problemas de simulação de fluidos, nos quais os fenômenos mais importantes ocorrem próximos das fronteiras [6].

Outra classe de geradores de malha é a classe de Grid, Quadtree e Octree. Nessa classe, a triangulação é construída sobrepondo um grid regular sobre o domínio e alterando os elementos para se adequar às restrições do domínio. Ao contrário do método de frente de avanço, os elementos próximos às restrições têm qualidade muito baixa em comparação aos outros métodos, enquanto que os elementos do interior têm melhor qualidade, como mostrado na Figura 3.1 [12]. Outra desvantagem desse método é que os elementos têm uma tendência a se alinhar com as direções principais do grid, o que pode influenciar na solução de elementos finitos. As vantagens dessa classe são a facilidade de paralelizar a geração das malhas e a robustez.

Já os geradores de malha Delaunay tendem a ter os melhores elementos no interior e de menor qualidade próximos às restrições, com as otimizações garantidas pela condição de Delaunay. Uma das vantagens desse tipo de método é que ele utiliza a triangulação parcial para decidir onde o próximo ponto será inserido na triangulação, e dessa forma é possível projetar o algoritmo com garantias teóricas de qualidade independentes dos dados de entrada. Os algoritmos que possuem essas garantias são chamados de *provably good*, ou comprovadamente bons, e são os algoritmos que serão estudados nesse trabalho.

Em 1989, Chew [2] apresentou o que é considerado o primeiro algoritmo

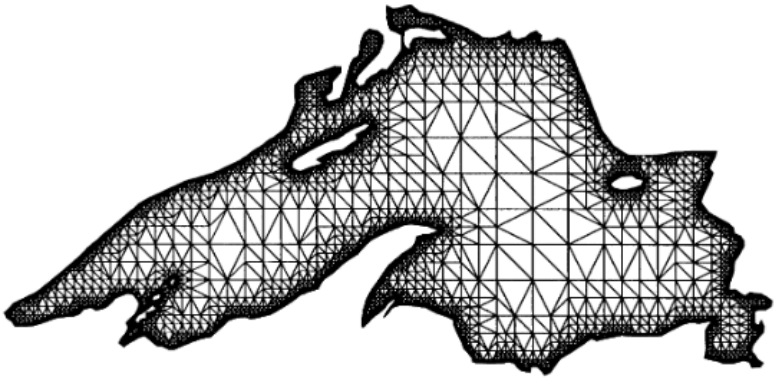

Figura 3.1: Um exemplo de uma malha gerada por um algoritmo do tipo grid. [12]

provably good que usava refinamento de Delaunay para gerar uma malha com garantia de ângulos entre 30°e 120°, o algoritmo de Chew (ou *Chew's First* como é conhecido atualmente). O algoritmo subdivide as arestas de restrição, de forma que todas tenham um tamanho entre h e $\sqrt{3}h$, onde h é a menor distância entre dois vértices de arestas, e então gera uma triangulação de Delaunay com restrições. Em seguida, os triângulos de qualidade ruim são removidos inserindo novos vértices no centro do circumdisco daquele triângulo. A desvantagem desse algoritmo é que a malha resultante é necessariamente uniforme, como mostrado na Figura 3.2, portanto pode ser necessário gerar um número grande de triângulos dependendo do h do PLC de entrada.

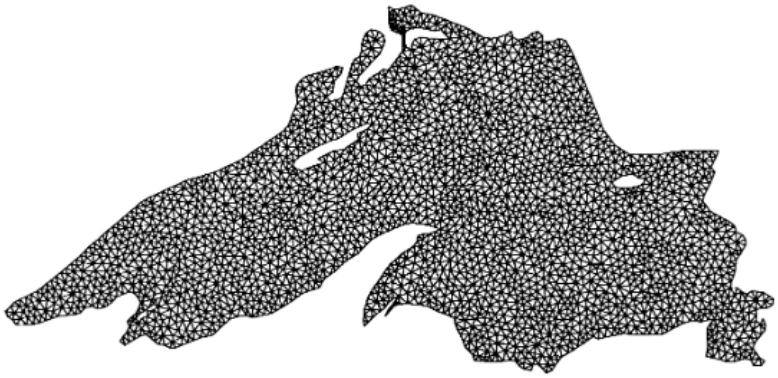

Figura 3.2: Um exemplo de uma malha gerada pelo primeiro algoritmo de Chew. [12]

O algoritmo de Chew [2] foi generalizado para três dimensões por Dey, Bajaj e Sugihara para gerar malhas de tetraedros de domínios convexos [5]. A generalização consegue remover a maioria dos tetraedros com qualidade ruim, com exceção dos *slivers* descritos na Seção 2.2. Entretanto, ainda tem a mesma limitação de gerar uma malha uniforme.

Em 1992, Jim Ruppert [3] publicou um algoritmo que garante uma otimização de um domínio não convexo com restrições internas e faz isso de forma que os triângulos da malha se concentrem nas regiões com mais detalhes, produzindo uma malha com gradações. Em comparação com os algoritmos anteriores, que geravam malhas de triângulos com tamanhos uniformes, isso permitiu uma redução grande do número de triângulos resultantes para a maioria dos PLCs, como apresentado na Figura 3.3. O algoritmo constrói uma triangulação de Delaunay conformada ao subdividir as arestas que não aparecem na triangulação de Delaunay, e utiliza uma caixa envolvente para conter o PLC, gerando os triângulos externos também.

Nesse trabalho, Ruppert [3] faz a definição de arestas *encroached*: uma aresta é considerada *encroached* se existe algum vértice da triangulação dentro do círculo diametral dessa aresta. Durante o refinamento, todas arestas *encroached* devem ser subdivididas antes de começar a remoção dos triângulos de baixa qualidade. Na operação de *split* de um triângulo, ao inserir o novo vértice, é verificado se alguma aresta de restrição vem a ser *encroached*. Caso isso aconteça, a inserção do vértice deve ser revertida, e a aresta que seria *encroached* é subdividida.

Para lidar com ângulos pequenos entre restrições, o algoritmo constrói um círculo protetivo ao redor dos vértices onde as arestas se encontram, criando as chamadas *shield edges* para evitar que os *splits* dessas arestas adjacentes entrem em um laço infinito. Em um trabalho subsequente, Ruppert [4] apresenta uma outra forma de lidar com os ângulos pequenos fazendo subdivisões das arestas em *cocentric shells*, de forma que em certo ponto as arestas vão ter comprimentos iguais e não vão forçar subdivisões das arestas vizinhas.

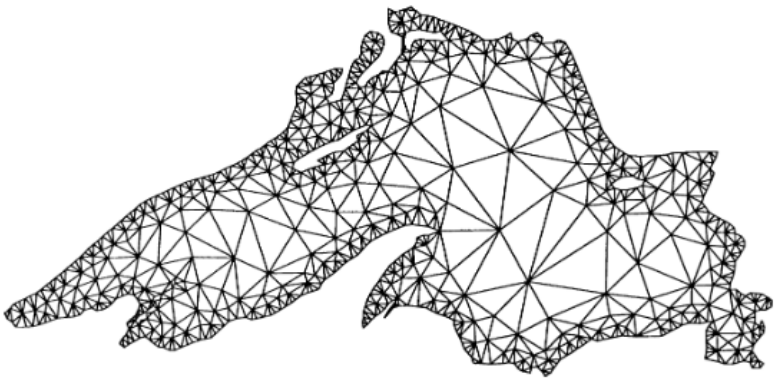

Figura 3.3: Um exemplo de uma malha gerada pelo primeiro algoritmo de Ruppert. [12]

Quatro anos depois de publicar o primeiro algoritmo, Chew [7] apresentou outro algoritmo que ficou conhecido como *Chew's Second*. Nesse algoritmo, a etapa inicial gera uma triangulação Delaunay com restrições, inserindo

todas as arestas na triangulação antes de começar o refinamento. O algoritmo então identifica os triângulos de qualidade ruim, e tenta removê-los inserindo um vértice na posição do seu circuncentro. Mas, diferente dos algoritmos de Delaunay conformados, é possível que o circuncentro do triângulo esteja no lado oposto de uma aresta de restrição ou até fora da malha. Nesse caso, deve-se identificar a primeira aresta de restrição entre o triângulo e o circuncentro e fazer o *split* dessa aresta ao invés de inserir um vértice na posição do seu circuncentro.

Esse algoritmo insere um número menor de pontos em comparação com o de Ruppert, pois não é necessário garantir que todas as arestas não estejam *encroached*. Além disso, utilizar o CDT traz algumas vantagens, como a de ser possível eliminar os triângulos externos (localizados fora do domínio geométrico) antes do refinamento, reduzindo o custo computacional e aumentando a qualidade da malha em alguns casos.

Em 1997, Shewchuk[6] apresentou uma análise matemática mais avançada dos algoritmos de Chew e Ruppert, e além disso fez uma generalização do algoritmo de Ruppert para três dimensões. Ruppert havia provado que seu algoritmo conseguia gerar malhas com nenhum ângulo menor que 20.7°. Utilizando uma análise semelhante, Shewchuk mostra que o segundo algoritmo de Chew tem garantia de gerar malhas otimizadas com ângulos maiores que 26.5°.

De acordo com Shewchuk, o segundo algoritmo de Chew é quase equivalente a uma versão do algoritmo que utiliza o conceito de *diametral lenses*, ou lentes diametrais. Nessa versão, uma aresta é considerada *encroached* quando existe um ponto no interior de uma área definida pela interseção de dois discos, cujos centros se encontram na fronteira do outro e que as fronteiras intersectam os pontos da aresta, como mostrado na Figura 3.4. Essa versão modificada do algoritmo de Ruppert tem a vantagem de inserir menos pontos na triangulação e apresentar um limite garantido de ângulos maiores que 26.5°, melhor que o algoritmo original.

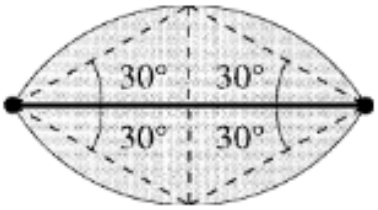

Figura 3.4: Área limitada pelas lentes diametrais.

Shewchuk avalia as dificuldades relativas à presença de ângulos pequenos

entre restrições do PLC. Ele mostra que mesmo utilizando o método das *cocentric shells* é possível que o algoritmo não termine, principalmente nos casos onde existe um ângulo pequeno adjacente a um grande. É apresentado um algoritmo chamado *The Quitter* [6] que define alguns critérios para decidir se o vértice pode ou não ser inserido, baseado na estrutura de heranças da malha.

Ainda nessa publicação, Shewchuk apresenta uma versão do algoritmo de refinamento em 3D. Ele pontua que em 3D a malha Delaunay não é necessariamente adequada para métodos de elementos finitos, graças a presença dos *slivers*. Ainda assim o Delaunay em 3D é muito útil como um ponto inicial para otimização e suavização da malha.

3.1 Algoritmos de geração de malha

Dentre as estratégias utilizadas pelos algoritmos de geração de malhas, três tipos se destacam [6]:

- **Dividir e conquistar**
  Primeiro algoritmo a ser implementado com ordem $O(n \log n)$, é considerado o mais eficiente em 2D, mas não tem bom desempenho em 3D. O algoritmo para construção de CDTs é complexo e raramente implementado.
- **Inserção incremental**
  Algoritmos bem implementados podem ter ordem $O(n \log n)$ em 2D e $O(n^2)$ em 3D. Os algoritmos mais eficientes em 3D atualmente são dessa classe, além de serem muito utilizados para construir CDTs.
- ***Gift wrapping***
  Facilmente implementado para construir CDTs e triangulações em dimensões mais altas. Entretanto o algoritmo básico é $O(n^2)$ em 2D.

3.2 Geradores de malha

Um importante gerador de malhas em duas dimensões é o software Triangle [8], que é capaz de gerar malhas de Delaunay, CDTs, malhas de Delaunay conformadas e malhas triangulares de alta qualidade. Um exemplo de resultado gerado pelo software é mostrado na Figura 3.5. O software tem código aberto, e foi utilizado como base para verificação dos resultados em duas dimensões.

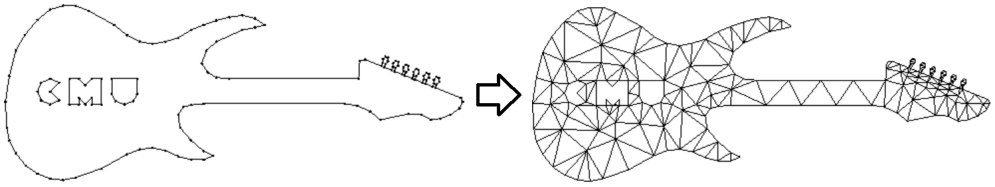

Figura 3.5: PLC de entrada (à esquerda) e malha gerada pelo software Triangle (à direita). [8]

Em três dimensões, um gerador bem conhecido é o software TetGen [10], que tem capacidade de gerar malhas de Delaunay e malhas de Delaunay conformadas, para qualquer PLC. Um exemplo de resultado gerado pelo software é mostrado na Figura 3.6.

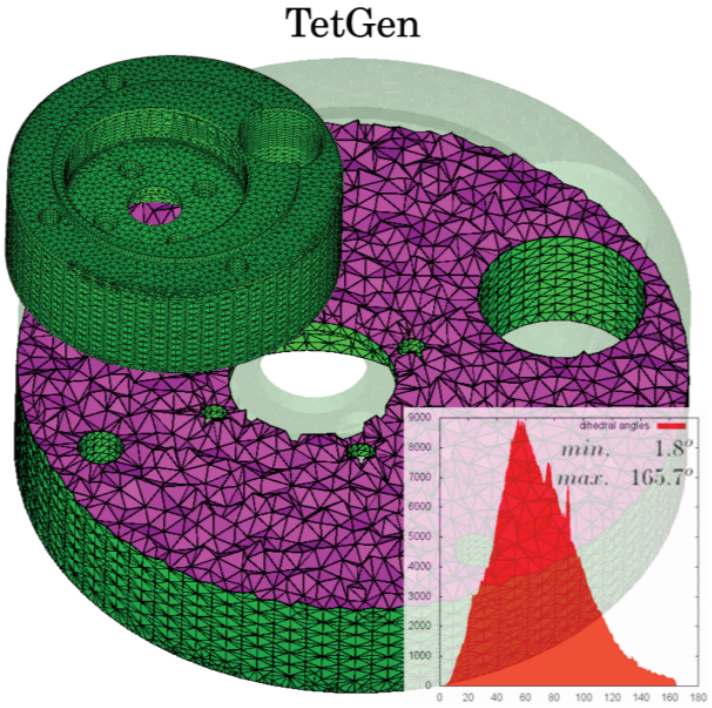

Figura 3.6: Malha gerada pelo software TetGen. [10]

4
Gerador de malhas

Neste capítulo são apresentados os algoritmos de geração implementados como parte do trabalho, em duas e em três dimensões. Primeiramente, serão discutidos brevemente a estrutura topológica utilizada, TopS, e o conceito de predicados robustos usados para os cálculos geométricos.

4.1
TopS

A estrutura de dados topológica TopS [9] foi utilizada como base para a construção de malhas de elementos finitos e interseção de superfícies de fraturas. Essa estrutura oferece uma representação completa das relações de adjacência entre todas as entidades topológicas de uma malha (nós, elementos, vértices, arestas e faces). As relações de adjacência são necessárias para que as operações utilizadas na geração de malhas e otimização da qualidade de elementos possam ser realizadas de maneira eficiente.

4.2
Predicados robustos

Algoritmos de geometria computacional, como cálculos de interseção e geração de malhas, possuem uma estrutura geométrica (ex. coordenadas de vértices) e outra combinatória (ex. topologia da malha). Esses algoritmos utilizam com frequência predicados que determinam o estado da estrutura combinatória a partir de dados geométricos.

No entanto, computadores atuais utilizam representação de ponto flutuante para aproximar números reais. Essa representação possui precisão limitada ao número de bits do tipo de dados correspondente. Dessa forma, predicados geométricos podem fornecer resultados inconsistentes, dependendo da ordenação e da magnitude dos dados de entrada. Isso pode levar a estrutura combinatória a um estado que viola as invariantes do algoritmo, e também invalidar a estrutura geométrica (ex. um polígono com área negativa).

Para facilitar a implementação correta de algoritmos de geometria computacional, evitando-se a necessidade do tratamento explícito de questões relacionadas à precisão numérica, foi utilizado um núcleo de aritmética robusta,

que emprega aritmética de números racionais para o cálculo exato de operações geométricas. Entretanto, como números racionais não são suportados diretamente pelo *hardware*, o custo das operações torna-se significativamente maior que o das operações de ponto flutuante. Assim, para manter o desempenho dos predicados geométricos mais próximos ao desejado, são utilizados "filtros aritméticos", que inicialmente utilizam aritmética de ponto flutuante, considerando margens de erro seguras para garantir a consistência dos resultados. Então, a aritmética de números racionais é aplicada apenas quando necessário. Todos os algoritmos desenvolvidos neste projeto utilizam o núcleo de aritmética robusta como base.

4.3 Gerador de malhas 2D

Devido à complexidade da geração de malha em três dimensões, optou-se por fazer a implementação de um algoritmo de geração de malha em duas dimensões como passo inicial. Todas as decisões com relação ao algoritmo de geração em duas dimensões foram pensadas de forma que a adaptação para três dimensões fosse o mais simples possível.

Nesta seção são descritos todos os passos do gerador de malha em duas dimensões: geração da malha de Delaunay, geração da malha de Delaunay com restrições e refinamento da malha.

Em duas dimensões a entrada do algoritmo é um PLC de vértices e arestas.

4.3.1 Delaunay

O primeiro passo do algoritmo é a geração de uma triangulação de Delaunay utilizando somente os vértices do PLC. O algoritmo escolhido para fazer essa etapa foi o incremental. Quando implementado de forma eficiente, pode ter a mesma ordem do algoritmo dividir e conquistar em duas dimensões, e ainda com a vantagem de poder ser diretamente generalizado para três dimensões.

Os três primeiros vértices são utilizados para criar o primeiro triângulo e iniciar a geração da malha. Em seguida, os próximos vértices são inseridos um a um segundo o algoritmo de *Bowyer-Watson* [6], apresentado no seguinte pseudocódigo.

Algoritmo 1: InserirVertice(v)

1 Localizar um triângulo t cujo circumdisco contém o vertice v;
2 Fazer uma busca local na malha pelos outros triângulos em conflito, a partir de t;
3 Remover da malha os triângulos em conflito;
4 Retriangular a cavidade com os triângulos formados pela união do vértice v com as arestas fronteira da cavidade;

A localização do primeiro triângulo em conflito é a de maior custo, e é o que define a ordem assintótica do algoritmo, já que todos os outros passos têm ordem constante de execução (cada passo é proporcional ao número de triângulos removidos/criados). Como exercício, foram feitas duas implementações diferentes de localização [6]: utilizando um grafo de conflito e o método *walking*.

4.3.1.1 Grafo de conflito

No grafo de conflito é criada uma estrutura auxiliar na forma de um grafo bipartido que para cada vértice armazena o índice do triângulo no qual o vértice está contido, e para cada triângulo armazena os índices dos vértices que estão contidos nele. Um exemplo de grafo de conflito pode ser visualizado na Figura 4.1. Dessa forma, a localização do triângulo t inicial se torna direta. O maior esforço do algoritmo é atualizar o grafo após a retriangulação da cavidade, quando devem ser identificados os vértices que estavam contidos dentro dos triângulos removidos e realocar para um dos novos triângulos, que contém o ponto. Com a atualização implementada, o algoritmo de geração de uma malha Delaunay tem ordem $O(n \log n)$.

4.3.1.2 *Walking*

Para o método *walking* a localização é feita utilizando a malha intermediária. A partir do último vértice inserido, o algoritmo busca o triângulo que contém o próximo vértice "andando" localmente na malha. O tipo de deslocamento pode ser de diferentes maneiras, ilustradas na Figura 4.2: em linha reta até o ponto, nos eixos x e y separadamente e fazendo um teste de visibilidade. O método escolhido foi o último, que tem implementação simples e robustez garantida pelos predicados de orientação, não sendo necessário tratar casos degenerados. Como as malhas intermediárias são Delaunay, é garantido que não vão existir laços onde o algoritmo poderia ficar preso.

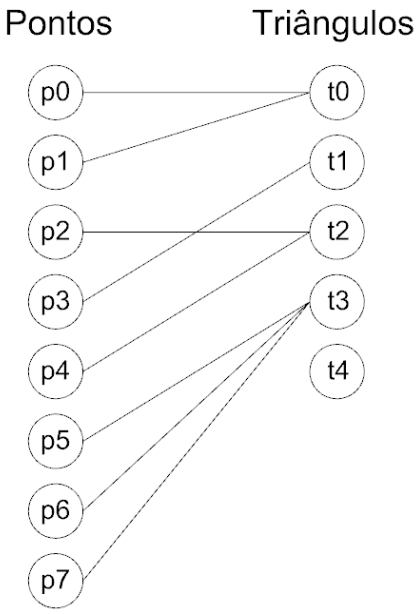

Figura 4.1: Ilustração de um grafo de conflito.

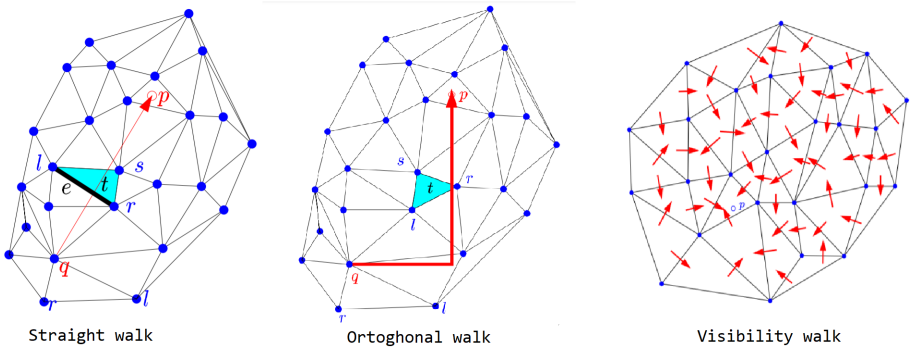

Figura 4.2: Tipos de localização *walking* [13]

Em ambos os métodos de localização é recomendado inserir os pontos em uma ordem aleatória para evitar casos extremos. Entretanto, para o método *walking* em específico, é ideal que seja feita uma ordenação que mantenha a localidade espacial entre os pontos, de forma que a localização leve o mínimo de etapas possíveis. Foi implementada então uma ordenação onde os pontos são inseridos em uma ordem quase aleatória, ou *Biased randomized insertion order* (BRIO) [6].

Nesse tipo de ordenação, os vértices são distribuídos em rodadas seguindo uma distribuição probabilística, ilustrada na Figura 4.3. Cada vértice tem 50% de chance de ser colocado na rodada $\log n$, 25% de chance de ser colocado na rodada $logn - 1$ e assim sucessivamente até a rodada 0, onde n é o número de vértices sendo inseridos. Após a distribuição dos vértices, os vértices são inseridos seguindo as rodadas: primeiro todos os vértices da rodada 0, depois

todos da rodada 1, até a última rodada.

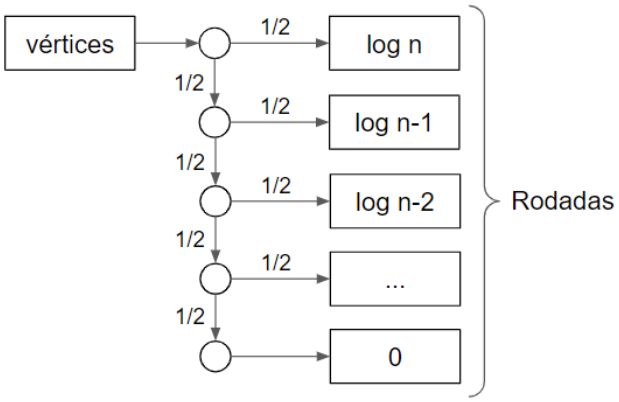

Figura 4.3: Ilustração da distribuição dos vértices entre as rodadas.

A vantagem de utilizar esse método de ordenação é que dentro das rodadas os pontos podem ser inseridos em uma ordem arbitrária, mantendo uma aleatoriedade adequada. Podemos então ordenar os vértices segundo uma curva de preenchimento do espaço para que eles tenham localidade espacial necessária para a eficiência do método *walking*. Para esse fim foi utilizada a curva de Hilbert [15] para transformar as coordenadas (x, y) de cada vértice em um parâmetro d que é a posição do vértice na curva, e então ordená-los de acordo com o valor de d. Alguns exemplos de curvas de Hilbert são mostrados na Figura 4.4.

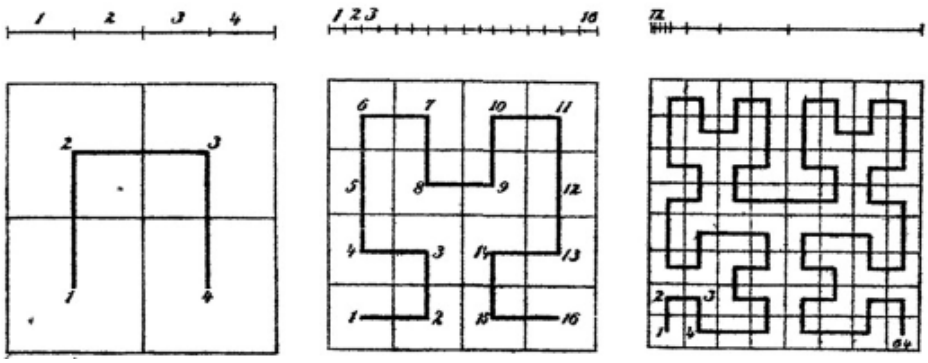

Figura 4.4: Curvas de Hilbert com graus variados. [15]

Para funcionar corretamente, o algoritmo de *Bowyer-Watson* precisa que todos os vértices a serem inseridos estejam dentro de um triângulo da malha. Uma maneira de garantir isso é criar um triângulo envolvente que contenha todos os vértices, porém essa solução pode causar erros numéricos principalmente na região do fecho convexo, dependendo da distribuição dos vértices. Uma solução mais elegante e robusta é a utilização de triângulos *fantasmas*, que são triângulos que ligam as arestas do fecho convexo a um

vértice no infinito: o vértice *fantasma*. Dessa forma qualquer vértice vai estar dentro de algum triângulo, seja ele fantasma ou não, o que garante o funcionamento do algoritmo.

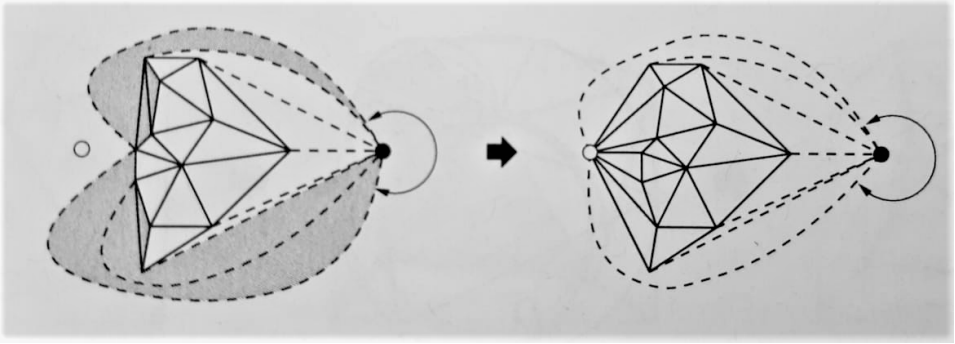

Figura 4.5: Um exemplo de inserção de um vértice que está fora da triângulação regular (vazado), mas em conflito com um triângulo fantasma. O ponto preto é o vértice fantasma. [6]

4.3.2 Delaunay com restrições

Após a inserção de todos os vértices temos uma malha de Delaunay, mas que não necessariamente contém as arestas de restrição definidas. O próximo passo então é inserir todas as arestas que não estiverem na malha. Para essa etapa, também foram feitas duas implementações diferentes: um método de retriangulação das cavidades e outro método baseado em *flips* de arestas.

4.3.2.1 Retriangulação das cavidades

O primeiro algoritmo implementado [6] segue as seguintes etapas, ilustradas na Figura 4.6: a partir do vértice inicial da aresta, identificar o triângulos que são interceptados por ela e os vértices que formam as fronteiras das cavidades esquerda e direita, ordenados. Todos os triângulos interceptados são removidos da malha e em seguida cada cavidade é retriangulada separadamente.

4.3.2.2 Bistellar Flip

O segundo algoritmo implementado foi o *Bistellar Flip* [1], que identifica as operações topológicas (flips de arestas) necessárias para transformar localmente a triangulação, de forma que todas as arestas de restrição passem a existir na malha. O algoritmo *Bistellar Flip* se aproveita do princípio de que,

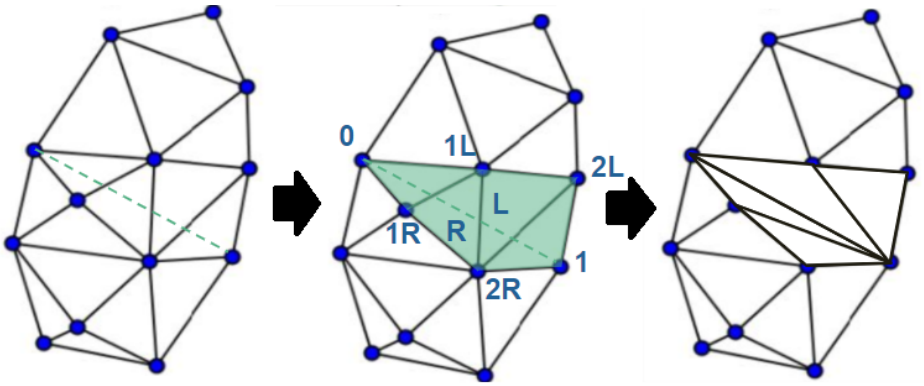

Figura 4.6: Ilustração das etapas do algoritmo de inserção de arestas.

quando a triangulação em duas dimensões é transformada em um mapa parabólico (*parabolic lift map*) em três dimensões, as arestas classificadas como localmente Delaunay na triangulação correspondem a arestas localmente convexas no mapa parabólico.

Para cada aresta de restrição a ser inserida na triangulação, identificam-se as arestas que a intersectam. O conjunto dos triângulos que contém essas arestas forma uma cavidade que deve ser retriangulada de forma que a restrição seja inserida na triangulação. A propriedade de convexidade no mapa parabólico é utilizada para definir a ordem correta dos *flips*, como ilustrado na Figura 4.7. A terceira coordenada dos pontos no mapa parabólico é variada adicionando um componente proporcional à distância do vértice a aresta sendo inserida. Conforme a terceira coordenada aumenta, as relações de convexidade entre faces podem inverter. Nesse momento é feito o *flip* dessas faces para que o mapa parabólico se mantenha convexo. Em um dado momento não haverá mais mudanças nas relações de convexidade, e portanto não existirão mais *flips* possíveis, e consequentemente a aresta de restrição vai ter sido recuperada.

4.3.3 Refinamento

Após a inserção de todas as arestas, temos uma malha CDT que respeita todas as restrições do PLC sem vértices adicionais. Em alguns casos, essa malha é suficiente, mas nos casos onde a malha vai ser utilizada pra cálculos numéricos, pode ser necessário melhorar a qualidade dessa malha. Para cada situação, podemos buscar otimizar algum critério da malha, mas em geral a remoção de triângulos com ângulos pequenos otimiza os parâmetros relevantes para cálculos numéricos. Além do ângulo mínimo dos triângulos resultantes, os refinamentos implementados podem ser controlados definindo-se também o tamanho mínimo e máximo das arestas. Dessa forma, o usuário pode adequar a malha final para o propósito desejado.

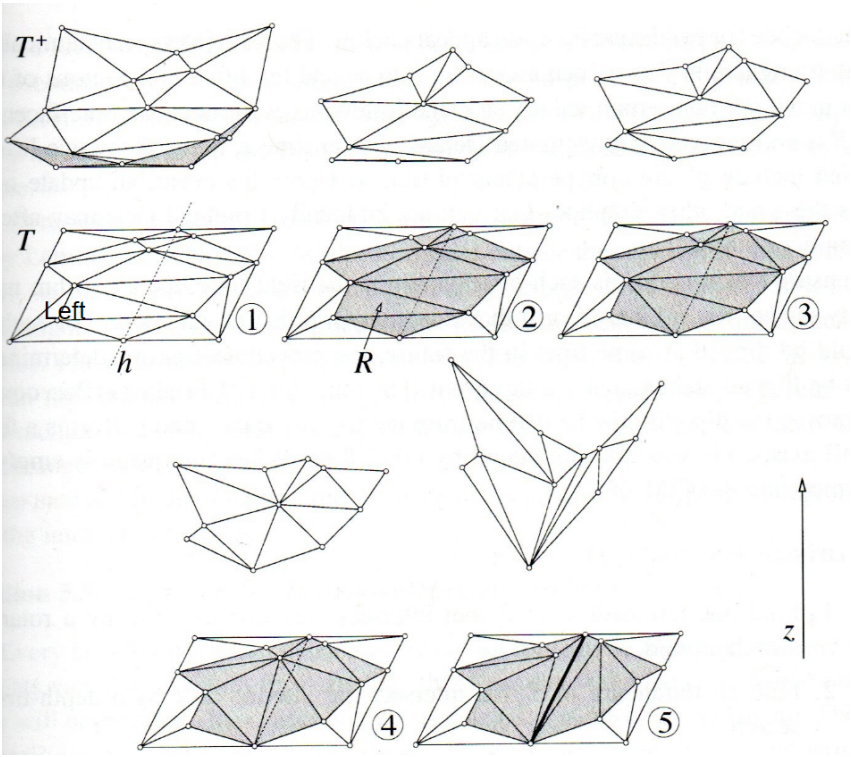

Figura 4.7: Algoritmo *Bistellar Flip* ilustrado em 2D. [6]

Para eliminar triângulos de baixa qualidade, novos vértices são inseridos na triangulação, e a malha é retriangulada ao redor de cada vértice inserido. Durante o refinamento, as arestas de restrição podem precisar ser subdivididas. As decisões que definem onde será inserido o próximo vértice é o que diferencia os algoritmos. Como opções de métodos de refinamento, foram implementados os seguintes algoritmos: *Chew's Second* [7], *Terminator* e *Terminator Chew* [6].

4.3.3.1 *Chew's Second*

O algoritmo *Chew's Second* [7] remove um triângulo de baixa qualidade inserindo um ponto próximo ao circuncentro do mesmo. Caso o ponto candidato à inserção esteja localizado do lado oposto a uma aresta de restrição, esta é subdividida, ao invés de se inserir o ponto. A subdivisão de arestas ocorre apenas quando necessário, nos casos em que não é permitida a inserção direta de um novo ponto, próximo ao circuncentro de um triângulo. O Chew's Second é o algoritmo que costuma gerar uma malha com o menor número de elementos. Porém, é menos resiliente a entradas contendo ângulos pequenos entre arestas de restrição.

4.3.3.2
Terminator e *Terminator Chew*

Os algoritmos *Terminator* e *Terminator Chew* funcionam de forma ligeiramente diferente. Além de manterem uma lista de triângulos de baixa qualidade, como no *Chew's Second*, esses algoritmos também mantêm uma lista de arestas classificadas como *encroached*. Essa é a primeira lista a ser percorrida. As arestas são removidas uma a uma da lista e subdivididas até que a lista esteja vazia. Em seguida, percorre-se a lista de triângulos de baixa qualidade. A definição de *encroached* para o *Terminator* é uma aresta cujo círculo diametral contém algum vértice da triangulação. O *Terminator Chew* define *encroached* como uma aresta cujas lentes diametrais (Figura 3.4) contêm algum vértice da triangulação.

Dentre os algoritmos implementados, o *Terminator* é o único que produz uma malha verdadeiramente Delaunay (atende aos critérios de Delaunay), e onde todos os circuncentros dos triângulos da malha estão localizados dentro do domínio da malha. Isso é desejável para a geração de diagramas de Voronoi a partir de uma triangulação de Delaunay.

O Terminator Chew não garante que a malha atenda aos critérios de Delaunay, porém garante que o refinamento sempre termina, mesmo nos casos de ângulos pequenos entre arestas de restrição. Esse algoritmo geralmente constrói uma malha com menos elementos que o Terminator.

4.4
Gerador de malhas 3D

Nesta seção, são descritos todos os passos do gerador de malha em três dimensões: perturbação numérica dos vértices, criação das estruturas auxiliares, geração da malha de Delaunay, geração da malha de Delaunay com restrições e refinamento da malha.

Em três dimensões, o gerador de malha recebe como entrada um PLC: conjunto de vértices em três dimensões, e um conjunto de restrições geométricas (arestas e faces), que devem ser respeitadas pela malha final.

4.4.1
Perturbação numérica

Para minimizar os problemas relacionados aos casos degenerados de pontos coplanares e coesféricos, foi implementado um algoritmo simples de perturbação numérica. Cada vértice teve suas três coordenadas perturbadas de forma aleatória, com uma distribuição linear entre $-\delta$ e $+\delta$ (para os arquivos testados $\delta = 10^{-8}$).

4.4.2 Estruturas auxiliares

Buscar uma face de restrição na triangulação em três dimensões não é tão direto, pois essa face pode ser composta por múltiplos triângulos na malha, como no caso de uma face poligonal. É necessário então manter uma estrutura auxiliar para guardar a triangulação planar de cada uma das faces de restrição. Essa triangulação deve ser a CDT do polígono pois, se aquela face estiver presente na malha, ela deve necessariamente estar representada por sua triangulação planar CDT. Portanto, para verificar se a face está na triangulação, é necessário buscar todas as faces triangulares do CDT armazenadas na estrutura.

O algoritmo implementado recebe somente faces triangulares, portanto a triangulação auxiliar inicial é composta só pelo próprio triângulo. As arestas de cada face podem ser subdivididas durante a etapa de recuperação das arestas, sendo necessário atualizar a triangulação auxiliar de todas as faces que contém essa aresta subdividida.

4.4.3 Triangulação de Delaunay

A primeira etapa de geração de malhas em 3D é a geração de uma triangulação de Delaunay do conjunto de vértices, utilizando um algoritmo de inserção incremental [6]. Como em duas dimensões, a maior complexidade desse algoritmo é na localização dos tetraedros que estão em conflito com o novo vértice, ou seja, cujas circunsferas contém o novo vértice. O grafo de conflito também seria aplicável em três dimensões, mas com mais dificuldades na etapa de atualização dos conflitos. Foi decidido então utilizar a solução de fazer a localização do tipo *walking*, no qual, a cada inserção, o algoritmo "anda" na malha a partir do vértice que acabou de ser inserido até o próximo. Apesar de não ter a garantia teórica como o grafo de conflito, na prática, o algoritmo *walking* tem ordem próxima de $O(n)$ ($O(1)$ por busca) [6].

Idealmente, os vértices são inseridos seguindo uma ordem que mantém a localidade espacial para minimizar as etapas de busca do *walking*, mas com aleatoriedade suficiente para evitar casos extremos. Entretanto, não foi desenvolvido o BRIO devido à complexidade da implementação de uma curva de preenchimento do espaço em três dimensões, sabendo que isso pode ter um impacto significativo na eficiência do algoritmo.

A triangulação é atualizada após cada inserção de um novo vértice, criando uma triangulação de Delaunay do subconjunto de vértices inseridos até então. O resultado final é uma malha de tetraedros que atende aos critérios

de uma triangulação de Delaunay, e que contém todos os vértices do PLC, mas ainda sem o compromisso de respeitar o conjunto de arestas e faces de restrição. A Figura 4.8 mostra a triangulação de Delaunay de um conjunto de 1000 vértices distribuídos aleatoriamente no interior de uma esfera.

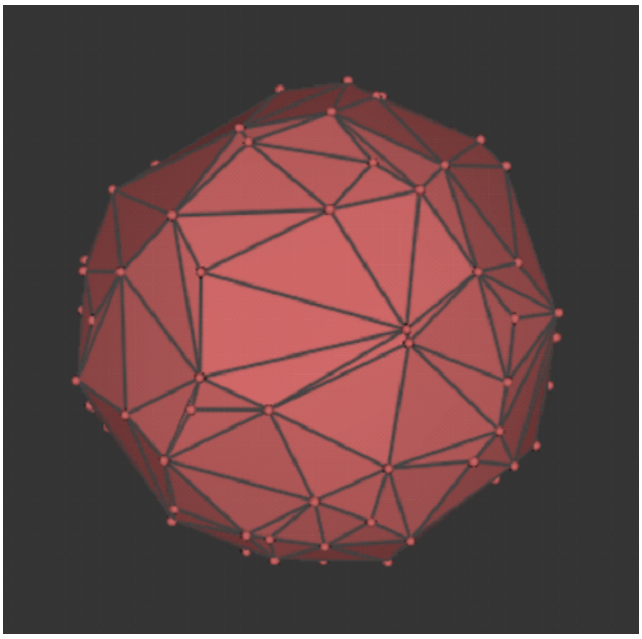

Figura 4.8: Exemplo de uma malha Delaunay em 3D gerada a partir de 1000 vértices.

4.4.4 Delaunay com restrições

A segunda etapa do gerador de malhas verifica se as arestas e faces do PLC (restrições) se encontram representadas na triangulação de Delaunay. Caso negativo, elas são inseridas uma a uma na triangulação.

Como foi apresentado na Seção 2.5, nem todos os PLCs têm uma triangulação possível em três dimensões. O primeiro passo do algoritmo então deve ser recuperar todas as arestas de restrição, dividindo as arestas em sub-arestas quando necessário. Uma vez que todas as arestas e sub-arestas estejam na triangulação, o PLC aumentado é *edge-protected*, e portanto é garantido que existe uma triângulação CDT possível. O próximo passo é recuperar as faces que ainda não estiverem presentes.

4.4.4.1 Recuperação das arestas

A primeira etapa para fazer a recuperação de arestas é verificar se as arestas das restrições estão presentes na triangulação (e por consequência são consideradas *Strongly Delaunay*). Caso não estejam, subdividir as arestas em

duas sub-arestas inserindo um vértice adicional, atualizando a triangulação. As sub-arestas também são testadas para verificar se estão na triangulação e devem ser subdividas caso não estejam.

O algoritmo deve ser feito de forma adequada para evitar laços infinitos na inserção dos novos vértices, principalmente em arestas que fazem ângulos agudos com outras arestas de restrição. Para evitar esse problema, foi utilizado o método descrito em [11].

Para definir onde o novo vértice será adicionado, o algoritmo leva em conta primeiro o tipo da aresta: *Tipo 1* são as arestas que não tem nenhum vértice *agudo* e *Tipo 2* são as arestas que tem um vértice *agudo* (arestas que tem dois vértices *agudos* podem ser subdivididas em duas arestas do *Tipo 2*). Um vértice é considerado *agudo* se existem pelo menos duas arestas de restrição incidentes que formam um ângulo menor de 90°.

Além do tipo, o algoritmo utiliza a propriedade de Delaunay que garante que, se a aresta não está presente na triangulação, existe ao menos um vértice dentro da esfera mínima que contém essa aresta. Para cada vértice é calculado o raio da circunsfera mínima do triângulo definido pelo vértice que bloqueia a aresta e os dois vértices da aresta que está sendo recuperada. O vértice que tem a circunsfera de menor raio vai ser utilizado para calcular a posição do ponto adicional.

4.4.4.2 Recuperação das faces

Para inserir as face de restrição, o algoritmo escolhido foi o *Bistellar Flip* [1], que identifica as operações topológicas (*flips* de faces) necessárias para transformar localmente a triangulação, de forma que cada face passe a existir na malha. O algoritmo em três dimensões se estende diretamente da versão em duas dimensões, com a dificuldade adicional por causa do operador de *flip* de faces, que pode criar e remover diferentes números de tetraedros dependendo da configuração tetraedros adjacentes a face.

4.4.5 Refinamento

O algoritmo de refinamento em três dimensões é análogo ao implementado em duas dimensões, mas além do operador de *split* de arestas e faces, é necessário um operador de *split* de tetraedro. Foi implementado somente o operador de *split* de arestas até o momento.

5
Resultados

Neste capítulo, serão mostrados alguns dos resultados do gerador de malha desenvolvido para diversos PLCs, em duas e três dimensões. Além disso, será apresentada uma análise comparativa de desempenho do gerador de Delaunay implentado em duas dimensões em relação com o Triangle.

5.0.1
Gerador em 2D

5.0.1.1
TECVis

O PLC da Figura 5.1 foi criado pra testar os algoritmos na geração de malhas com regiões separadas pelas restrições e arestas internas. Foram geradas as seguintes malhas: a triangulação CDT (Fig. 5.2), malha refinada com critério de ângulo mínimo permitido igual a 20° com os algoritmos *Chew's Second* (Fig. 5.3), *Terminator Chew* (Fig. 5.4) e *Terminator* (5.5). Além disso, foi gerada uma malha utilizando o algoritmo *Terminator* com o critério de ângulo mínimo permitido igual a 20° e o comprimento máximo de aresta igual a 1 (5.6).

Observando os resultados, notamos que o CDT contém alguns triângulos de baixa qualidade, que são removidos pelos métodos de refinamento. Dos três métodos, o *Chew's Second* gera o menor número de triângulos, seguido pelo *Terminator Chew* e por último o *Terminator*, com o mesmo valor de ângulo mínimo permitido. A última malha gerada, adicionando um critério de tamanho máximo de aresta no refinamento, têm um número consideravelmente mais alto de triângulos em comparação com a gerada sem o critério, com a vantagem que os triângulos têm ângulos mais próximos de 60°.

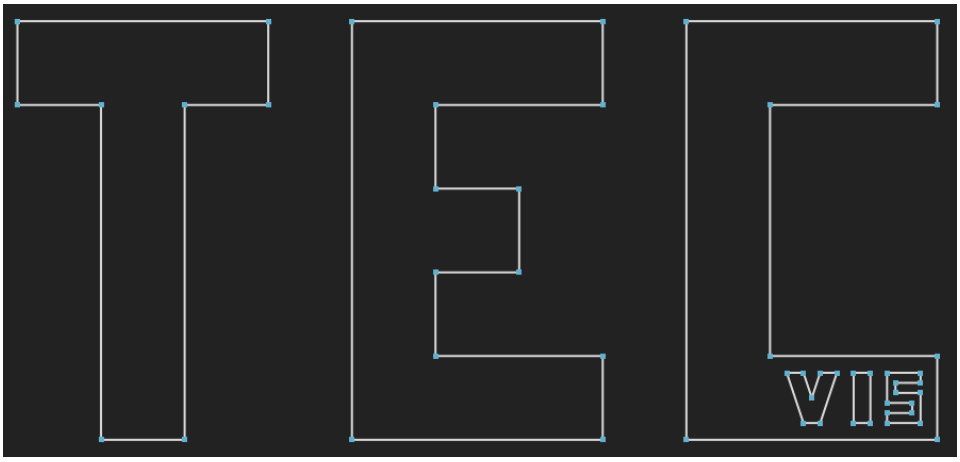

Figura 5.1: Arestas e vértices do PLC.

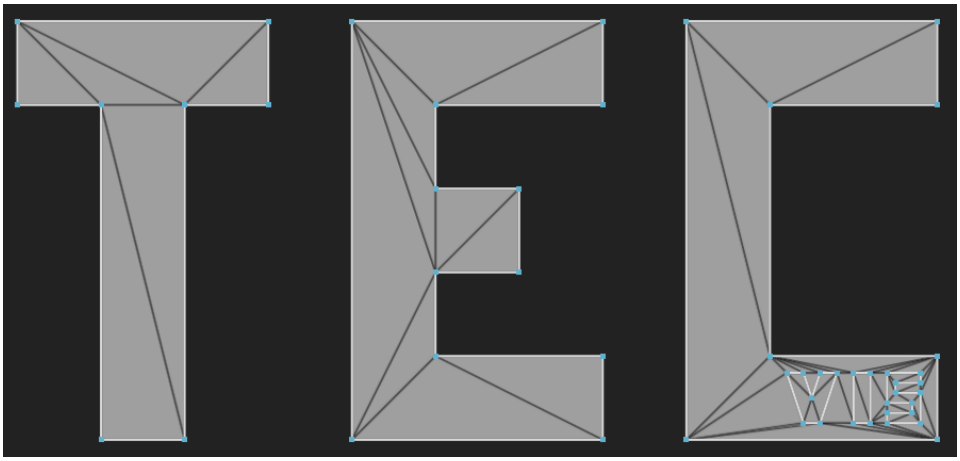

Figura 5.2: CDT do PLC.

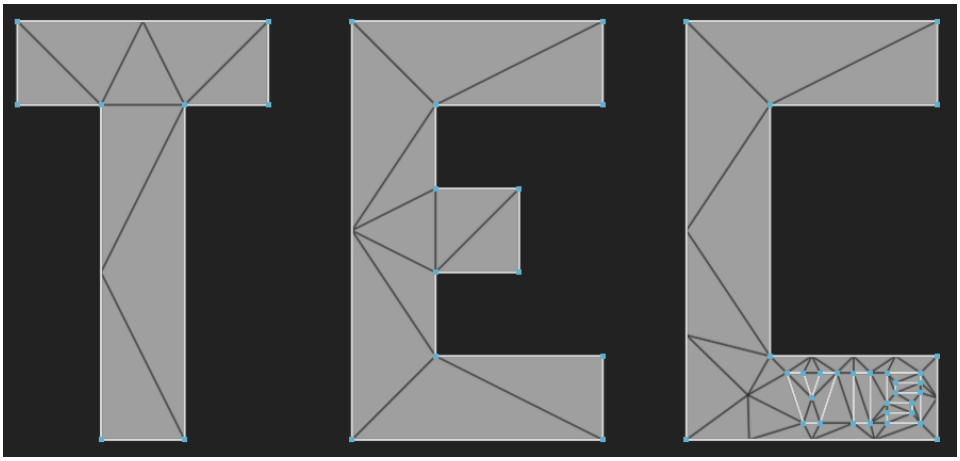

Figura 5.3: Malha refinada utilizando o algoritmo *Chew's Second*. (Ângulo mínimo permitido = 20°)

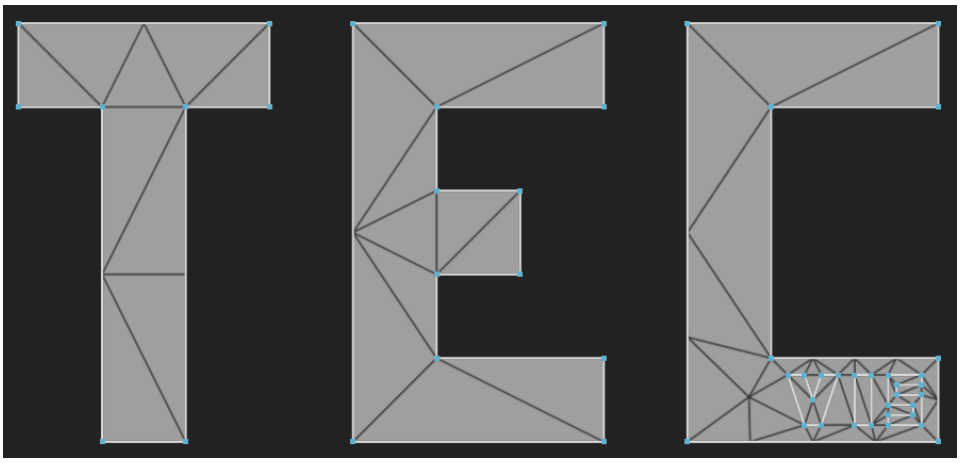

Figura 5.4: Malha refinada utilizando o algoritmo *Terminator Chew*. (Ângulo mínimo permitido = 20°)

Figura 5.5: Malha refinada utilizando o algoritmo *Terminator*. (Ângulo mínimo permitido = 20°)

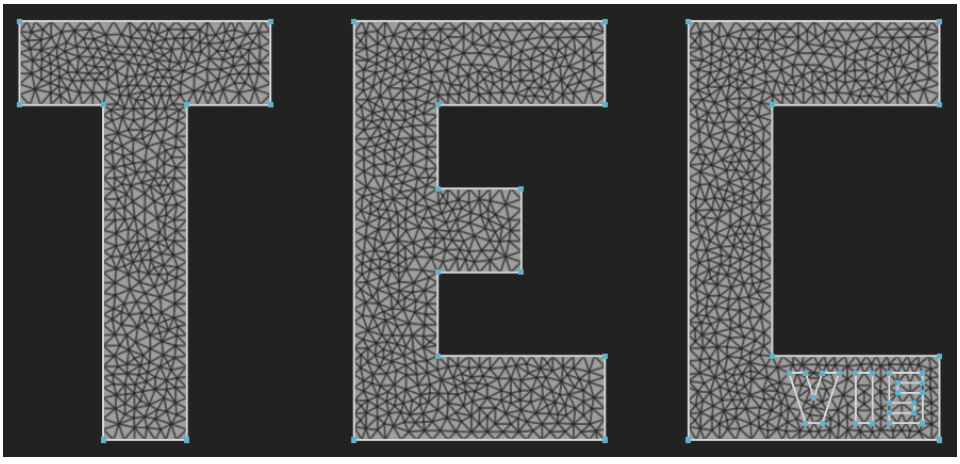

Figura 5.6: Malha refinada utilizando o algoritmo *Terminator*. (Ângulo mínimo permitido = 20°; Comprimento de aresta máximo = 0.2)

5.0.1.2
Arestas de restrição com ângulos pequenos

O PLC da Figura 5.7 foi criado pra testar os algoritmos na geração de malhas com arestas de restrição formando ângulos pequenos. Foram geradas as seguintes malhas: a triangulação CDT (Fig. 5.8), malha refinada com critério de ângulo mínimo permitido igual a 20° com os algoritmos *Chew's Second* (Fig. 5.9), *Terminator Chew* (Fig. 5.10) e *Terminator* (5.11). Além disso, foi gerada uma malha utilizando o algoritmo *Terminator* com o critério de ângulo mínimo permitido igual a 20° e o comprimento máximo de aresta igual a 1 (5.12).

A malha CDT gerada apresenta triângulos de baixa qualidade, o que é esperado, já que as restrições formam ângulos pequenos entre elas. Os métodos de refinamento buscam melhorar a qualidade, entretanto os ângulos definidos pelas restrições não podem ser removidos. É possível notar, principalmente nas Figuras 5.11 e 5.12, como as arestas ao redor do vértice central são subdividas em comprimentos iguais, estratégia utilizada para garantir que o processo de refinamento termine.

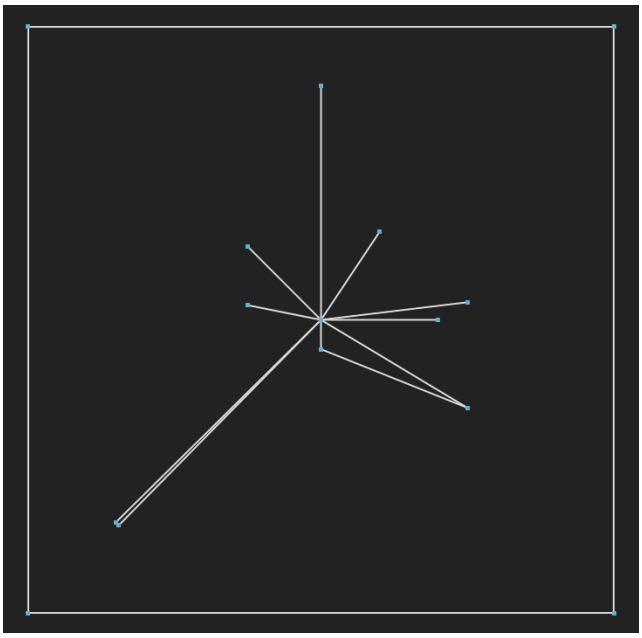

Figura 5.7: Arestas e vértices do PLC.

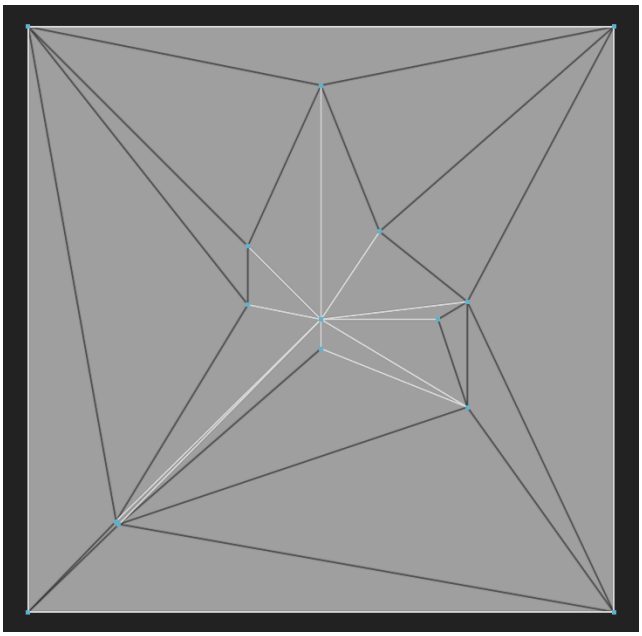

Figura 5.8: CDT do PLC.

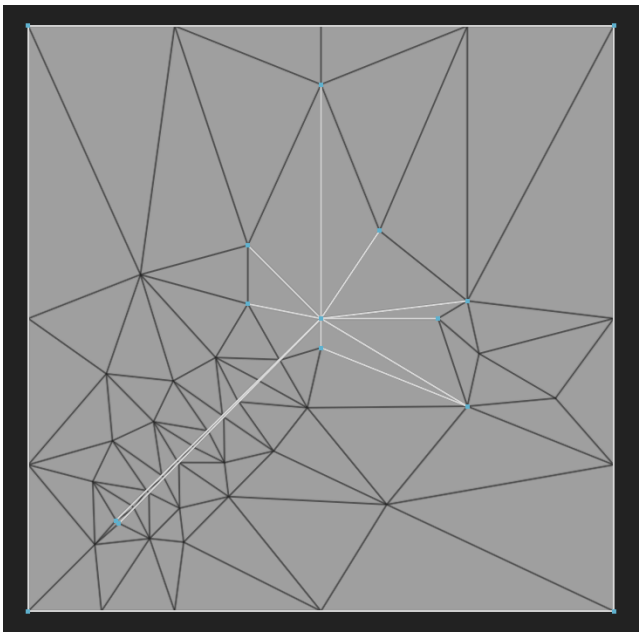

Figura 5.9: Malha refinada utilizando o algoritmo *Chew's Second*. (Ângulo mínimo permitido = 20°)

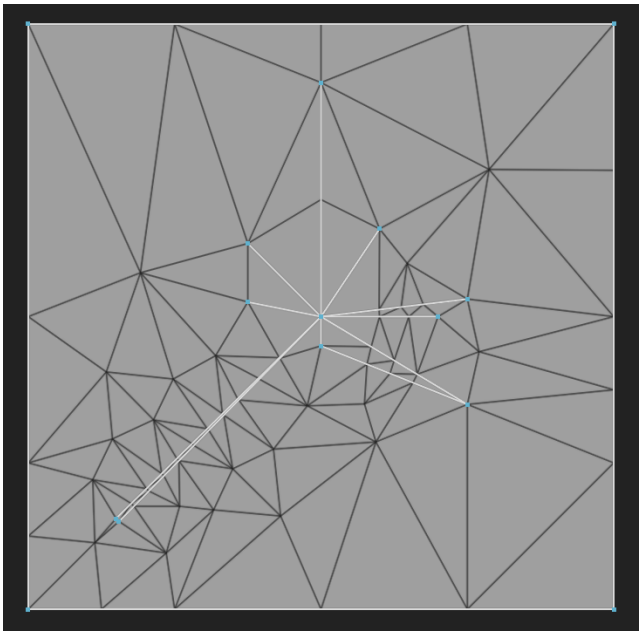

Figura 5.10: Malha refinada utilizando o algoritmo *Terminator Chew*. (Ângulo mínimo permitido = 20°)

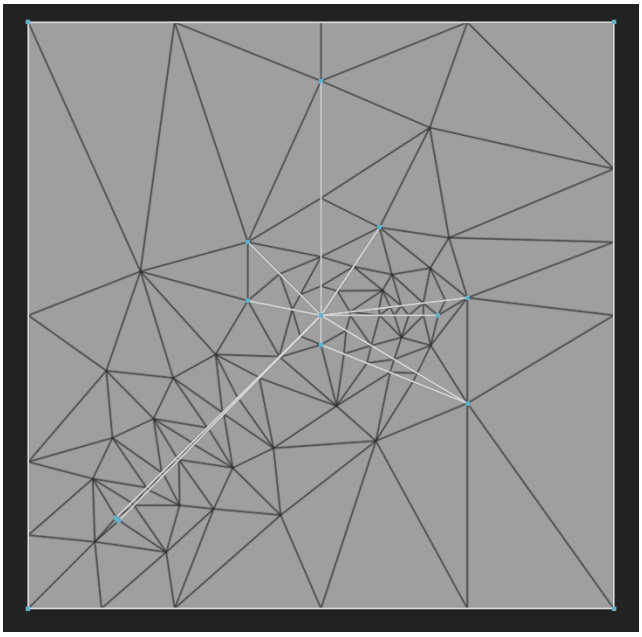

Figura 5.11: Malha refinada utilizando o algoritmo *Terminator*. (Ângulo mínimo permitido = 20°)

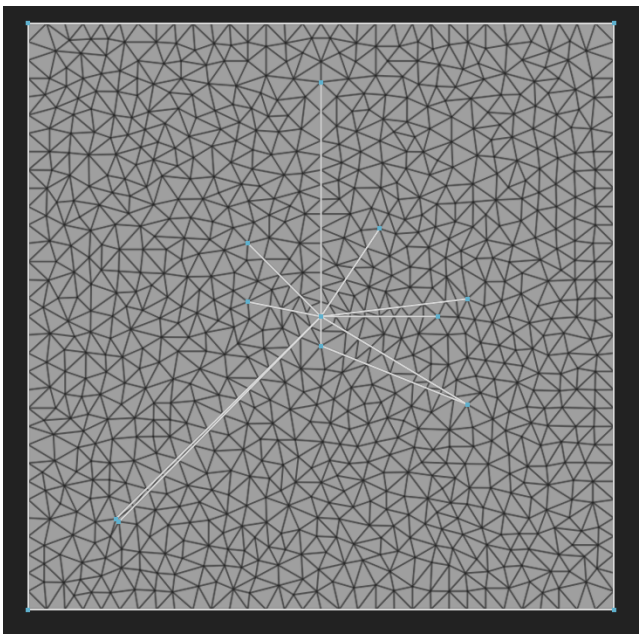

Figura 5.12: Malha refinada utilizando o algoritmo *Terminator*. (Ângulo mínimo permitido = 20°; Comprimento de aresta máximo = 1)

5.0.1.3
Meio fraturado

O PLC da Figura 5.13 representa um modelo de um meio fraturado. Foram geradas as seguintes malhas: a triangulação CDT (Fig. 5.14), malha refinada com critério de ângulo mínimo permitido igual a 20° com os algoritmos *Chew's Second* (Fig. 5.15), *Terminator Chew* (Fig. 5.16) e *Terminator* (5.17). Além disso, foi gerada uma malha utilizando o algoritmo *Terminator* com o critério de ângulo mínimo permitido igual a 20° e o comprimento máximo de aresta igual a 200 (5.18).

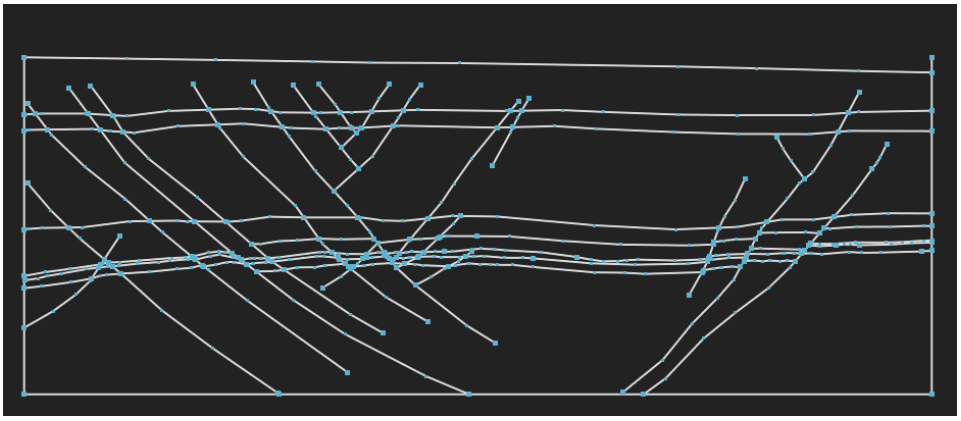

Figura 5.13: Arestas e vértices do PLC.

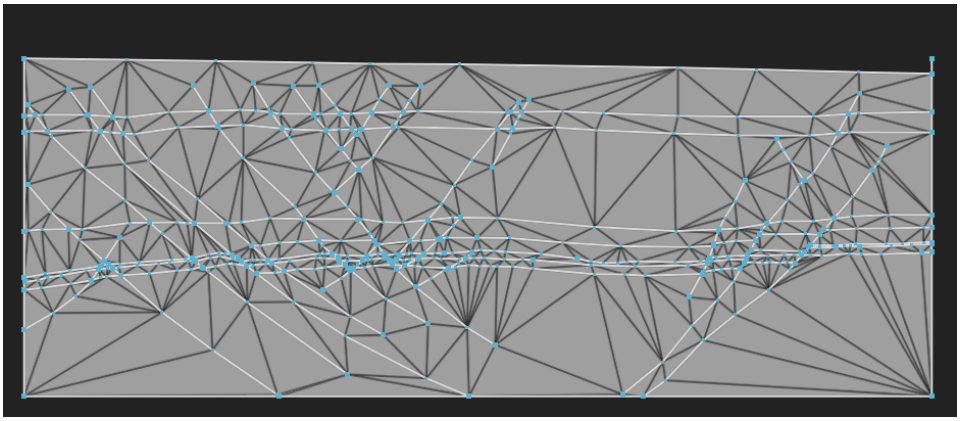

Figura 5.14: CDT do PLC.

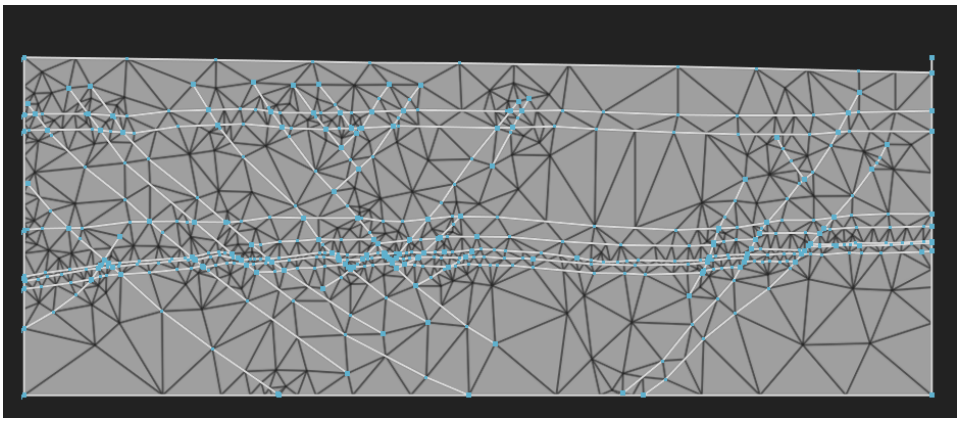

Figura 5.15: Malha refinada utilizando o algoritmo *Chew's Second*. (Ângulo mínimo permitido = 20°)

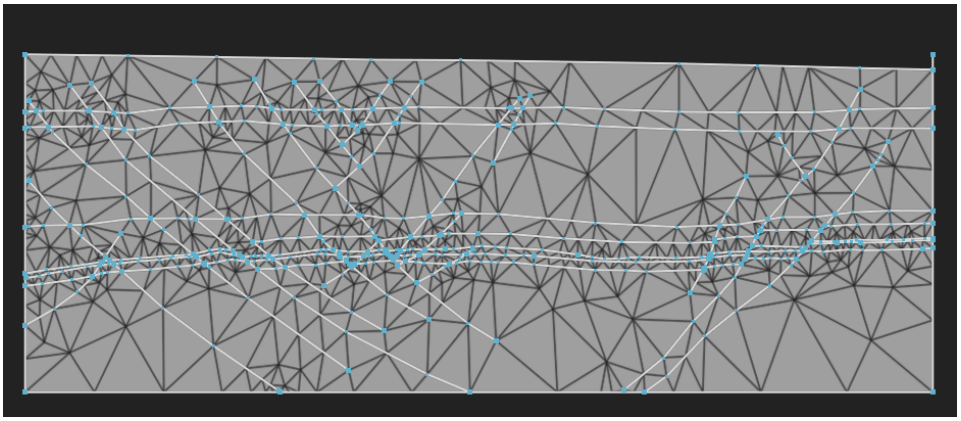

Figura 5.16: Malha refinada utilizando o algoritmo *Terminator Chew*. (Ângulo mínimo permitido = 20°)

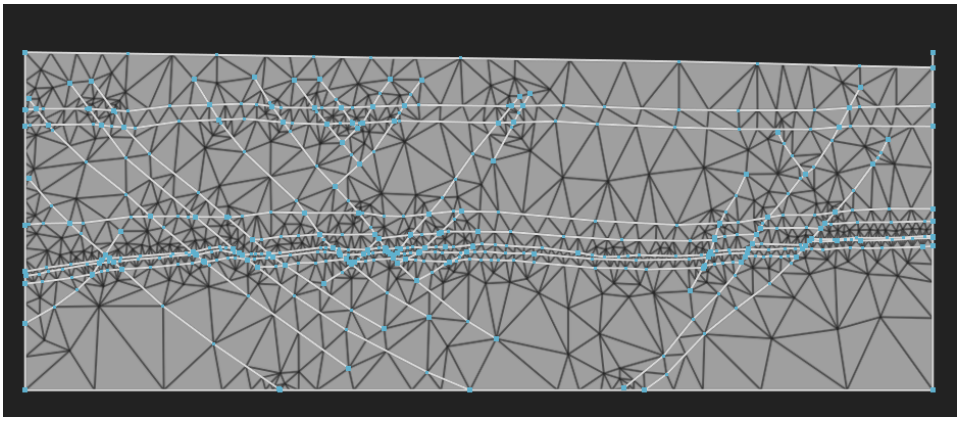

Figura 5.17: Malha refinada utilizando o algoritmo *Terminator*. (Ângulo mínimo permitido = 20°)

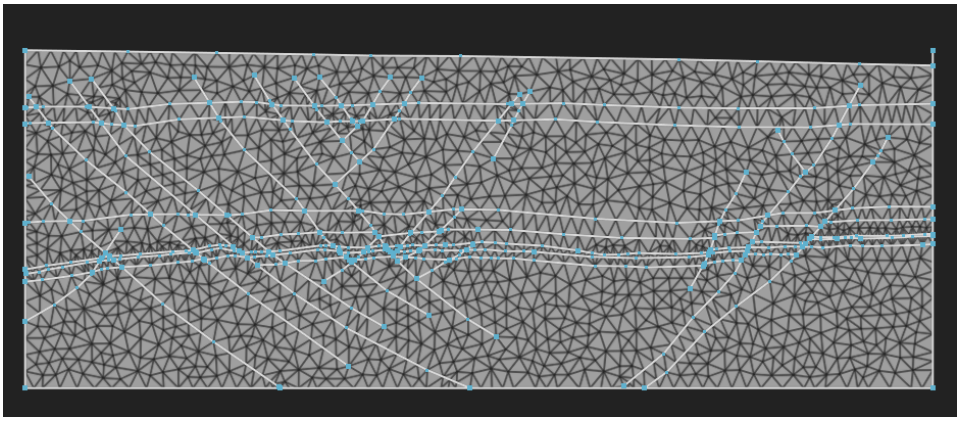

Figura 5.18: Malha refinada utilizando o algoritmo *Terminator*. (Ângulo mínimo permitido = 20°; Comprimento de aresta máximo = 200)

Para esse modelo, a triangulação CDT tem muitos triângulos de baixa qualidade, com ângulos pequenos, como podemos verificar no histograma da Figura 5.19. Utilizando o algoritmo *Terminator*, com o parâmetro de ângulo mínimo permitido igual a 20°, temos uma melhora na qualidade da malha gerada. Observando o histograma da Figura 5.20, podemos ver que os triângulos se concentram mais à direita, com valores mais próximos de 60°. Mantendo o critério de ângulo mínimo, e adicionando ainda um critério de comprimento máximo de aresta igual a 200, a qualidade da malha melhora um pouco mais. O histograma da Figura 5.21 mostra um deslocamento ainda maior da curva de distribuição, significando mais triângulos de qualidade alta.

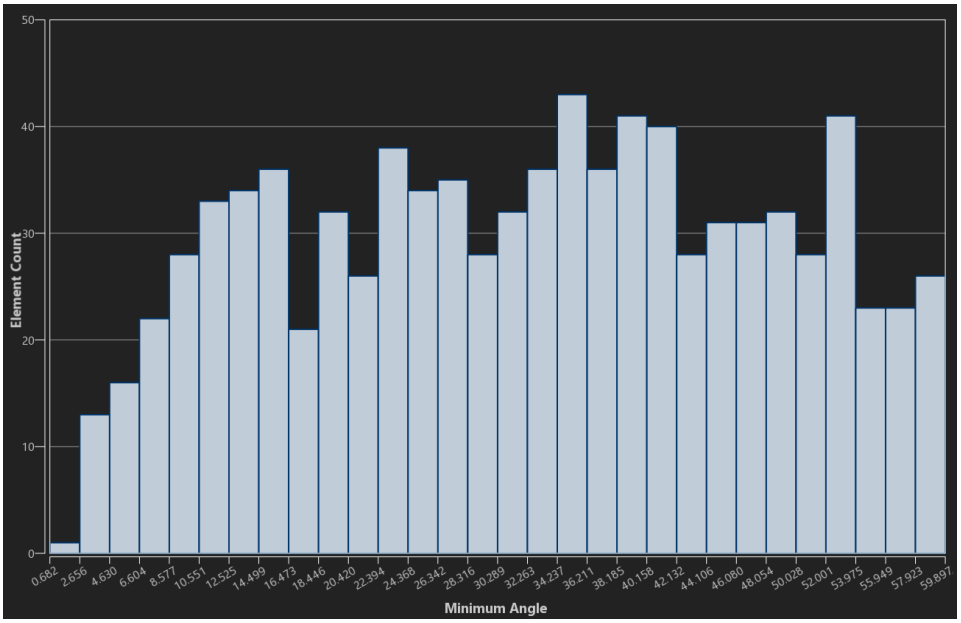

Figura 5.19: Histograma da CDT do PLC.

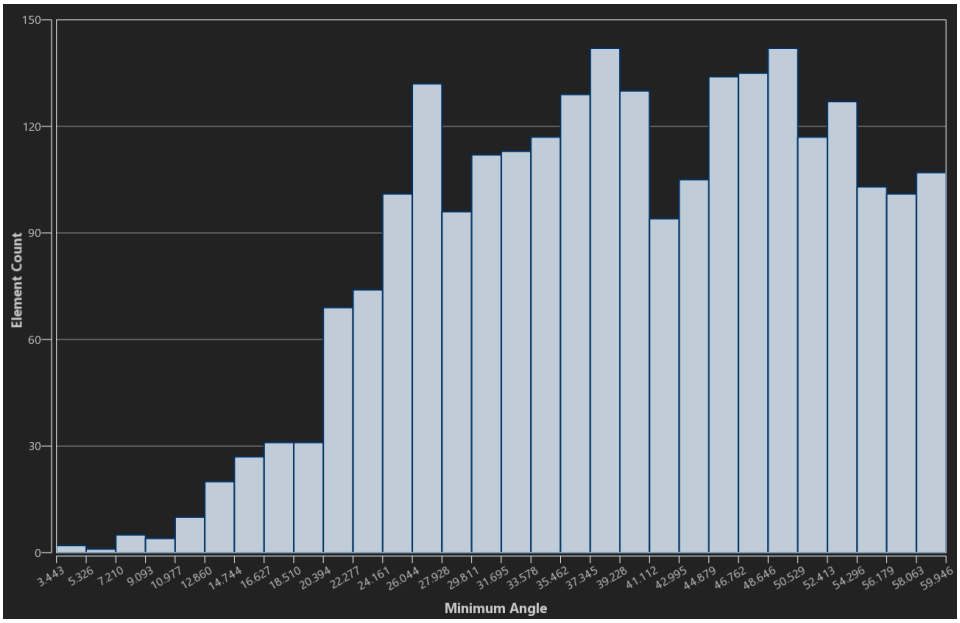

Figura 5.20: Histograma da malha refinada utilizando o algoritmo *Terminator*. (Ângulo mínimo permitido = 20°)

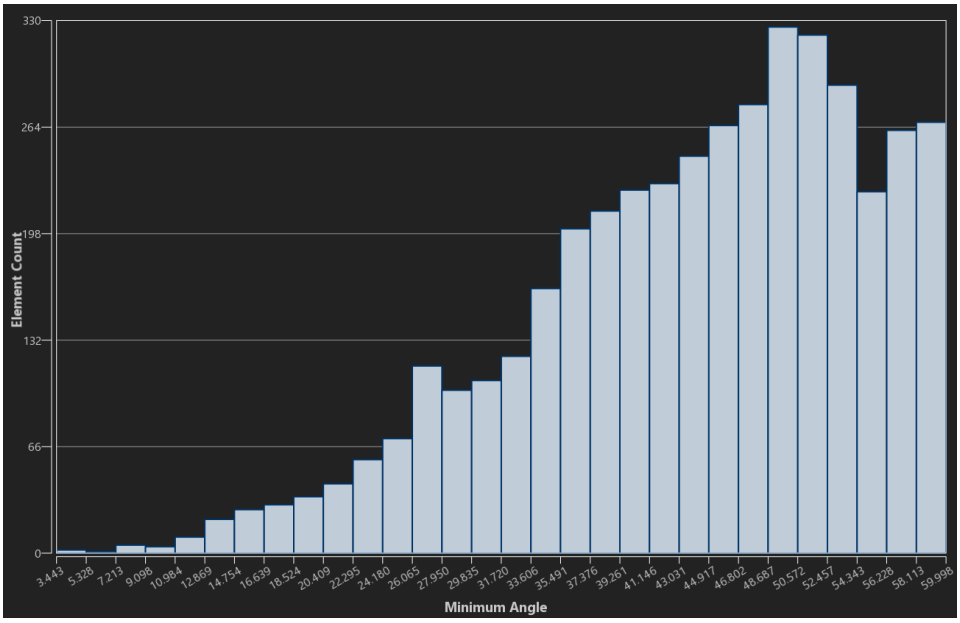

Figura 5.21: Histograma da malha refinada utilizando o algoritmo *Terminator*. (Ângulo mínimo permitido = 20°; Comprimento de aresta máximo = 200)

5.0.1.4 Análise de desempenho

Como forma de verificar o desempenho dos algoritmos implementados, foram feitos testes com conjuntos de vértices distribuídos aleatoriamente, e medidos os tempos de execução de cada algoritmo. Além dos dois métodos de localização de vértices implementados, foi utilizado o software Triangle [8] como comparativo, com os algoritmo dividir e conquistar e inserção incremental. O gráfico da Figura 5.22 apresenta os resultados dos testes, do qual podemos tirar as seguintes conclusões: Todos os algoritmos, com exceção do método inserção incremental do Triangle, têm comportamento linear. O método de inserção incremental implementado tem desempenho próximo do método dividir e conquistar do Triangle, o que mostra que em duas dimensões a inserção incremental é uma boa estratégia.

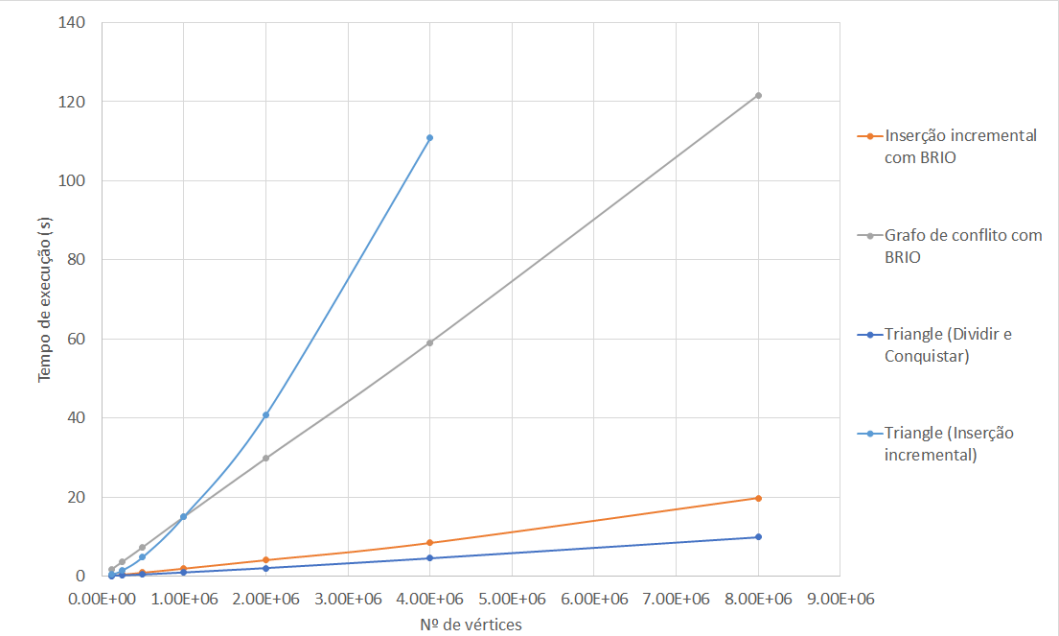

Figura 5.22: Análise de desempenho dos algoritmos implementados em comparação com os algoritmos do software Triangle [8].)

5.0.2 Gerador em 3D

5.0.2.1 Bunny

Esse PLC (Figura 5.23) é um modelo 3D de um coelho, amplamente utilizado na computação gráfica. Foi gerada a triangulação CDT desse PLC (Figs. 5.24 e 5.25). Podemos ver que o gerador foi capaz de gerar um malha de tetraedros que respeita as faces de restrição impostas. Além dos 149 vértices do conjunto inicial, foram inseridos 12 novos vértices na triângulação, como parte do método de recuperação das arestas, necessário para garantir a existência do CDT.

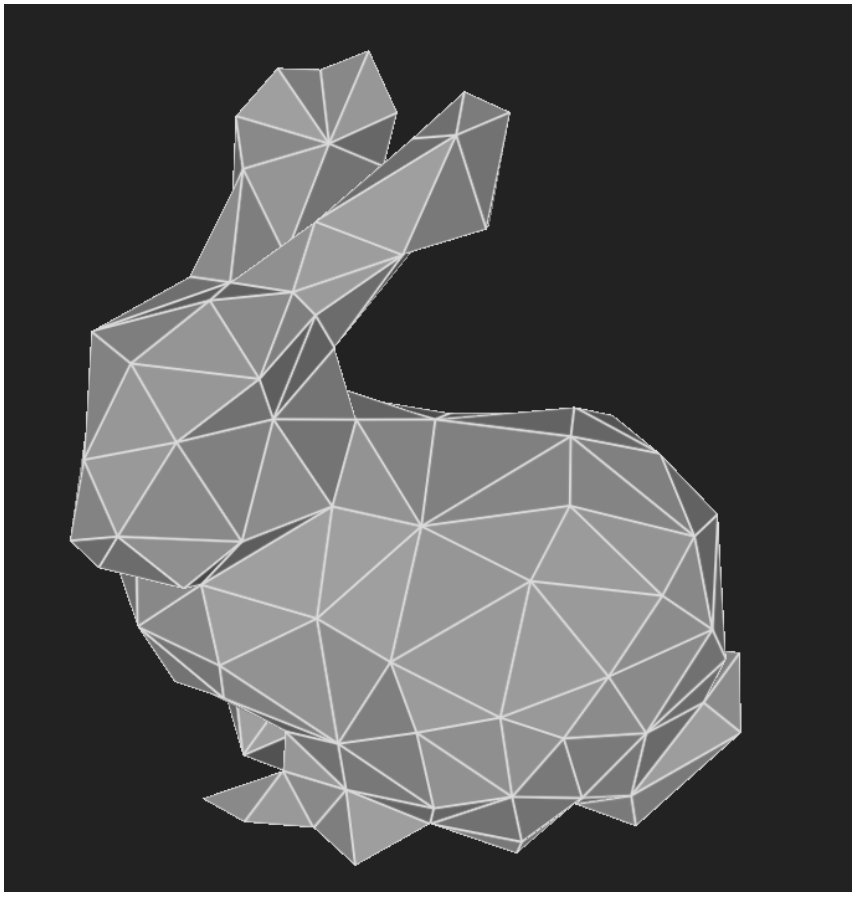

Figura 5.23: Arestas, vértices e faces do PLC.

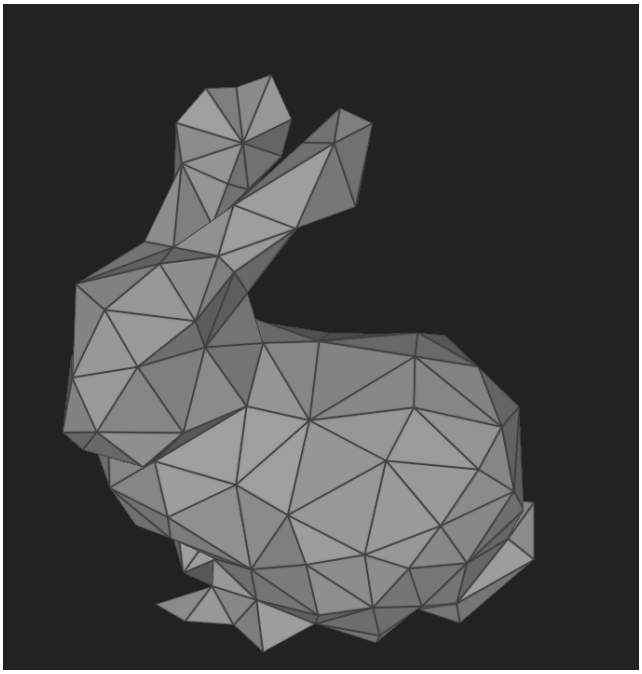

Figura 5.24: CDT do PLC.

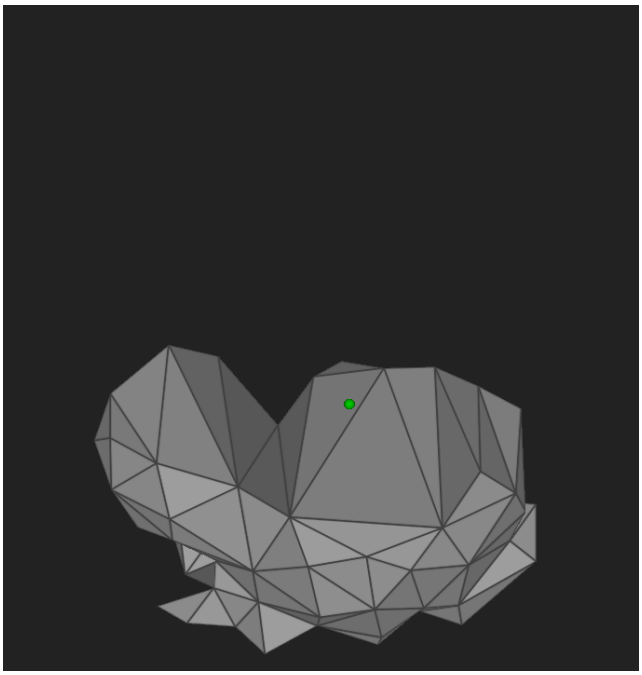

Figura 5.25: CDT do PLC. Alguns tetraedros foram removidos utilizando um plano de corte, para permitir a visualização do interior da malha.

5.0.2.2
Meio fraturado

Esse PLC (Figura 5.26) representa um modelo de um meio fraturado em três dimensões. Na Figura 5.27 podemos ver que o modelo possui faces de restrições internas. Foi gerada a triangulação CDT desse PLC (Figs. 5.28 e 5.29).

A maior dificuldade na geração da malha para esse PLC foi o grande número de vértices coplanares, já que as faces da fratura são planares. O algoritmo de perturbação numérica foi essencial para remover os casos degenerados, garantindo consistência topológica nos resultados das operações.

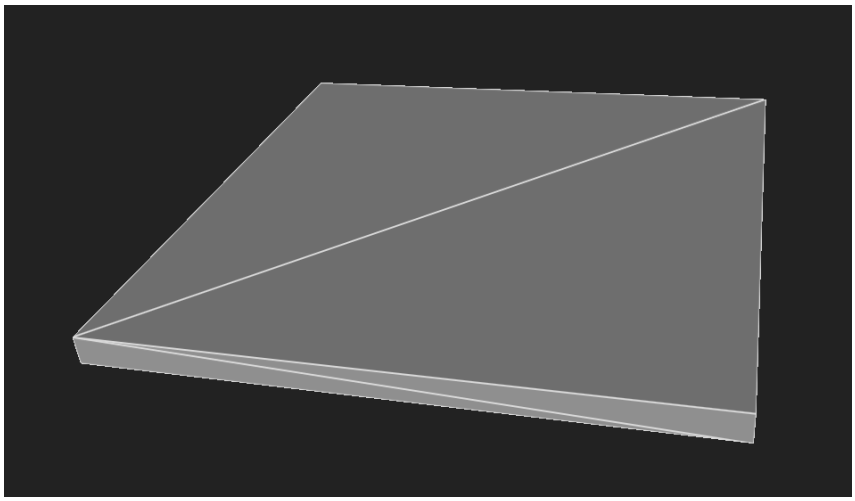

Figura 5.26: Arestas, vértices e faces do PLC.

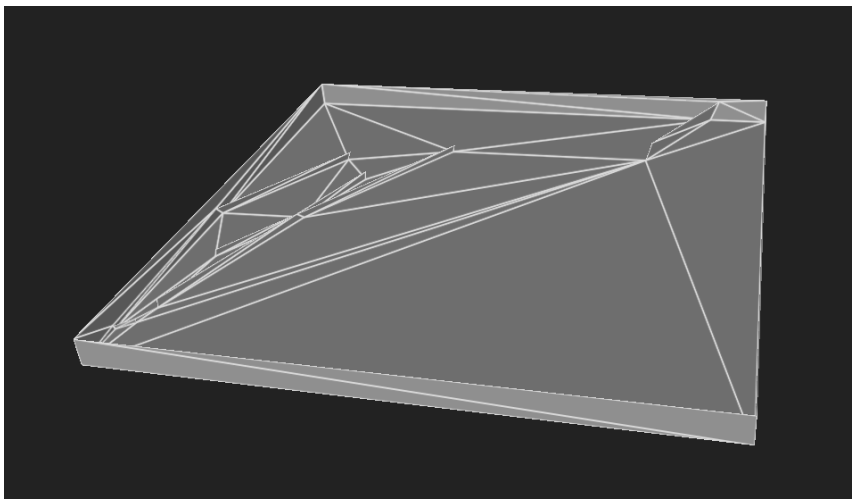

Figura 5.27: Arestas, vértices e faces do PLC, com a face do topo omitida para permitir a visualização das faces de restrição internas.

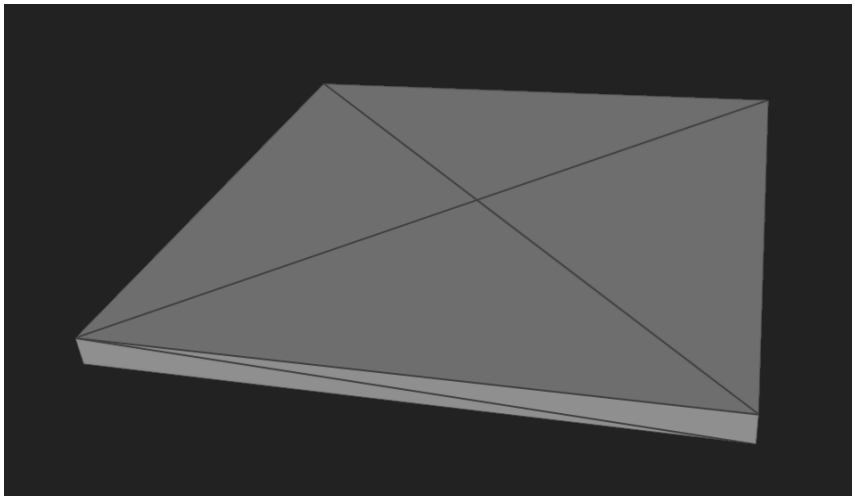

Figura 5.28: CDT do PLC.

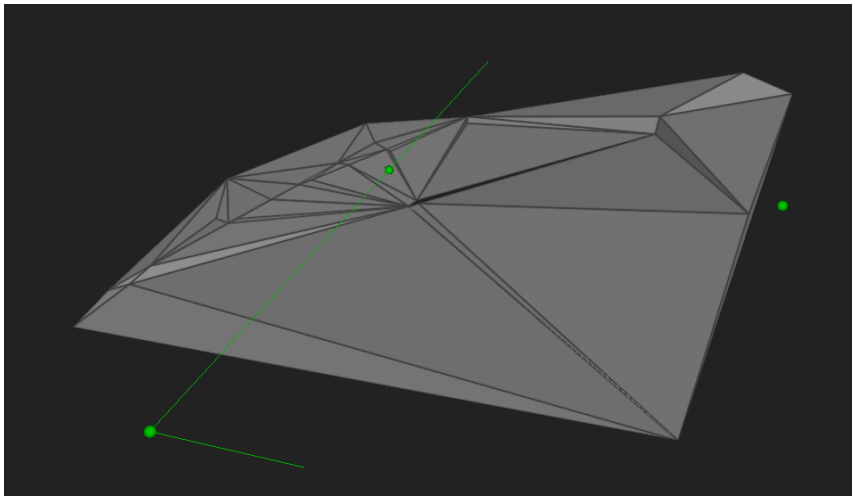

Figura 5.29: CDT do PLC. Alguns tetraedros foram removidos utilizando um plano de corte, para permitir a visualização do interior da malha.

5.0.2.3
Modelo de reservatório

Esse PLC (Figura 5.30) representa uma camada e uma fratura de um modelo de reservatório em três dimensões. Foi gerada a triangulação CDT desse PLC (Figs. 5.31 e 5.32). O modelo é bem complexo, principalmente na região onde as fraturas se interceptam. Ainda assim, o gerador de malhas implementado consegue gerar uma malha de tetraedros consistente topologicamente.

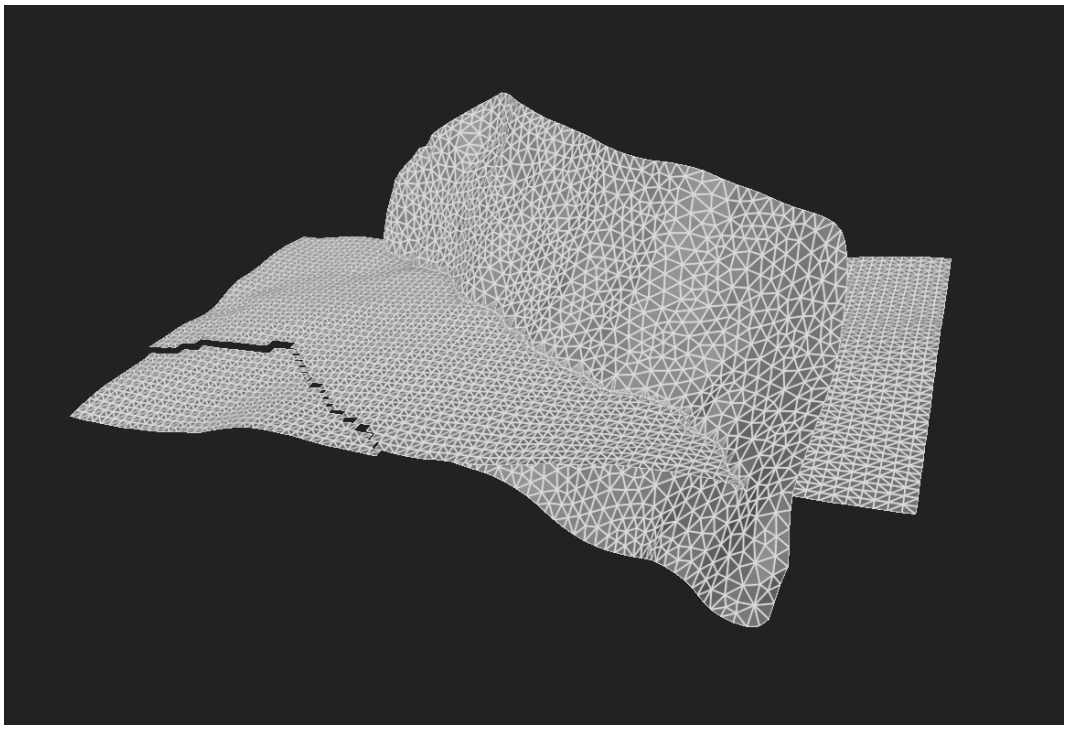

Figura 5.30: Arestas, vértices e faces do PLC.

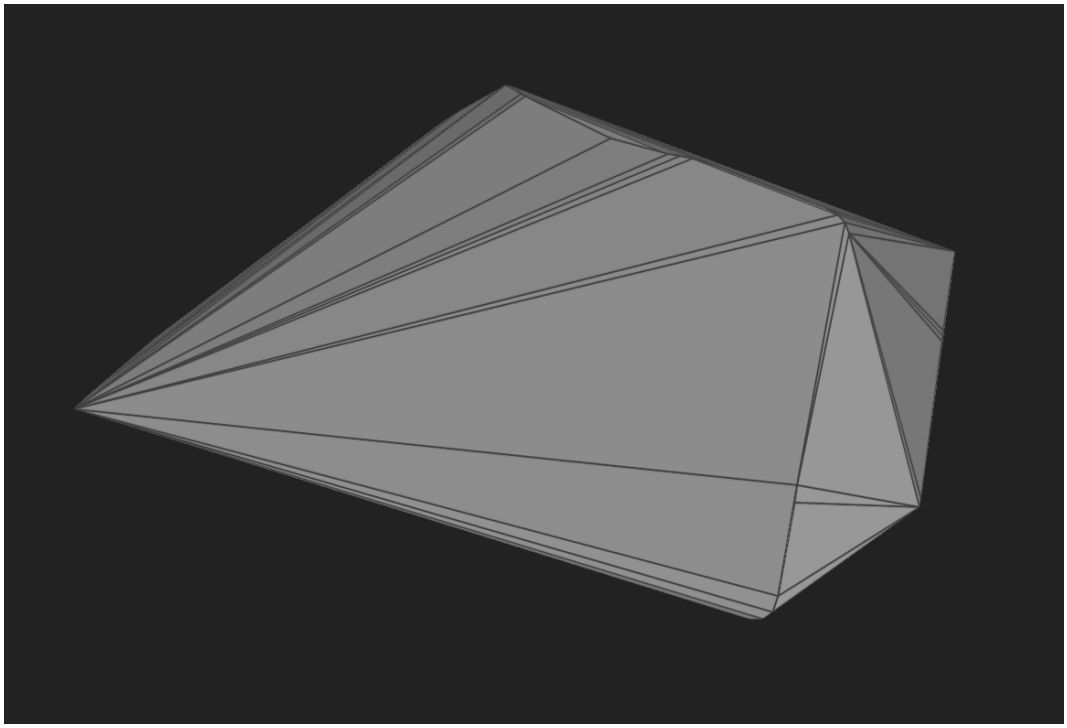

Figura 5.31: CDT do PLC.

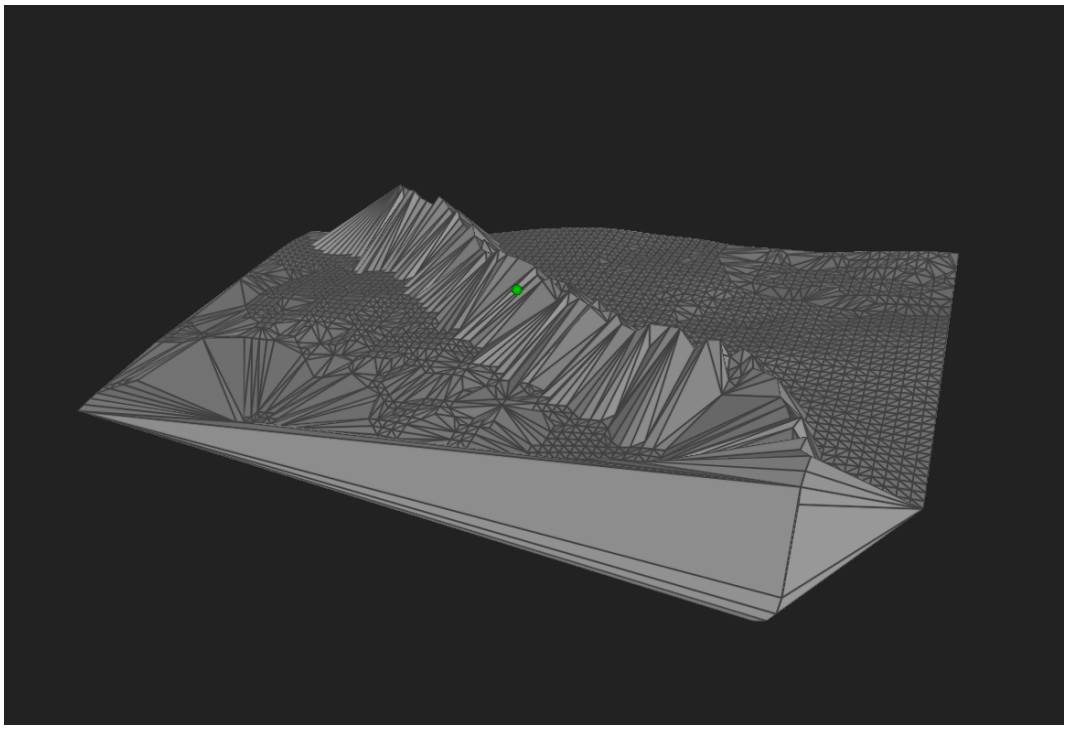

Figura 5.32: CDT do PLC. Alguns tetraedros foram removidos utilizando um plano de corte, para permitir a visualização do interior da malha.

6
Conclusão

Nesta dissertação, foram estudadas as propriedades de malhas de Delaunay e métodos de geração, em duas e em três dimensões. Dos diferentes tipos de geradores, o de inserção incremental foi escolhido para implementação devido à sua facilidade de implementação tanto em duas quanto em três dimensões.

Em duas dimensões o gerador desenvolvido é capaz de gerar uma malha de qualidade adequada para utilização em simulações de elementos finitos, a partir de um PLC de entrada. O gerador possui três opções diferentes de refinamento, e três parâmetros de qualidade, que podem ser variados para obter uma malha com as características desejadas.

Em três dimensões o gerador desenvolvido é capaz de gerar uma malha CDT, a partir de um conjunto de vértices e faces triangulares, mas sem refinamento dos elementos gerados. Durante o desenvolvimento, diversas dificuldades foram encontradas para implementar os algoritmos, principalmente na recuperação de arestas e faces. Além das dificuldades do algoritmo, os problemas de casos degenerados apareceram em modelos complexos que tinham muitos vértices coplanares. Para minimizar esse problema, foi implementado um algoritmo simples de perturbação numérica dos vértices.

Como trabalhos futuros, pretende-se evoluir o gerador 3D para incluir as seguintes funcionalidades:

- Refinamento para remover os tetraedros de baixa qualidade, a partir de parâmetros definidos
- Função de *sliver exudation* para remover os *slivers* que sobreviverem ao refinamento
- Implementar algoritmo de pós-processamento da malha para suavizar e melhorar ainda mais a qualidade
- Adaptar o algoritmo de CDT para receber polígonos como restrições. Para isso é necessário implementar um gerador de malha para polígonos planos em 3D.

Além das funcionalidades mencionadas acima, que são importantes pra gerar malhas melhores, existem diversas maneiras de melhorar a eficiência e robustez do algoritmo, como:

– Implementar uma perturbação simbólica dos vértices, que tenha garantia teórica na remoção dos casos degenerados que prejudicam a execução do algoritmo. Uma abordagem promissora é a implementação de um algoritmo de geração de malhas de Delaunay com pesos, que permitem que a perturbação não altere as coordenadas geométricas dos vértices.

– Implementar a curva de Hilbert em três dimensões, e utilizar como parâmetro de ordenação dos vértices no BRIO. A eficiência do gerador na etapa de Delaunay deve melhorar significativamente, já que o método de localização *walking* depende diretamente da localidade espacial na ordem de inserção.

Referências bibliográficas

[1] CHENG, S.; DEY, T. ; SHEWCHUK, J.. **Delaunay Mesh Generation**. Chapman & Hall/CRC Computer and Information Science Series. CRC Press, 2016.

[2] CHEW, L. P.. **Guaranteed-quality triangular meshes**. Technical Report Technical Report TR-89-983, Department of Computer Science, Cornell University, Ithaca, New York, 1989.

[3] RUPPERT, J.. **A new and simple algorithm for quality 2-dimensional mesh generation**. In: Ramachandran, V., editor, PROCEEDINGS OF THE FOURTH ANNUAL ACM/SIGACT-SIAM SYMPOSIUM ON DISCRETE ALGORITHMS, 25-27 JANUARY 1993, AUSTIN, TEXAS, USA. ACM/SIAM, 1993.

[4] RUPPERT, J.. **A delaunay refinement algorithm for quality 2-dimensional mesh generation**. J. Algorithms, 18:548–585, 1995.

[5] DEY, T.; BAJAJ, C. ; SUGIHARA, K.. **On good triangulations in three dimensions**. Internat. J. Comput. Geom. Appl., 2:431–441, 1991.

[6] SHEWCHUK, J.. **Delaunay refinement mesh generation**. 01 1997.

[7] CHEW, L.. **Guaranteed-quality mesh generation for curved surfaces**. In: SCG '93, 1993.

[8] SHEWCHUK, J. R.. **Triangle: Engineering a 2d quality mesh generator and delaunay triangulator**. In: APPLIED COMPUTATIONAL GEOMETRY TOWARDS GEOMETRIC ENGINEERING, p. 203–222. Springer, 1996.

[9] CELES, W.; PAULINO, G. H. ; ESPINHA, R.. **A compact adjacency-based topological data structure for finite element mesh representation**. 2005.

[10] SI, H.. **Tetgen, a delaunay-based quality tetrahedral mesh generator**. ACM Trans. Math. Softw., 41(2), 2015.

[11] SI, H.; GÄRTNER, K.. **Meshing piecewise linear complexes by constrained delaunay tetrahedralizations**. p. 147–163, 01 2005.

[12] SHEWCHUK, J. R.. **Reprint of: Delaunay refinement algorithms for triangular mesh generation**. Computational Geometry, 47(7):741–778, 2014.

[13] DEVILLERS, O.; PION, S. ; TEILLAUD, M.. **Walking in a triangulation**. In: PROCEEDINGS OF THE SEVENTEENTH ANNUAL SYMPOSIUM ON COMPUTATIONAL GEOMETRY, SCG '01, p. 106–114, New York, NY, USA, 2001. Association for Computing Machinery.

[15] HILBERT, D.. **Ueber die stetige abbildung einer linie auf ein flächenstück**. Mathematische Annalen, 38:459–460, 1891.