FREDERICO SANTOS DE OLIVEIRA

UMA AVALIAÇÃO DE MOVIMENTO DE MALHAS BASEADO NA FORMULAÇÃO LAPLACIANA NA RESOLUÇÃO DA EQUAÇÃO DO CALOR POR DISCRETIZAÇÕES DE VOLUMES FINITOS COM REFINAMENTO DE DELAUNAY

LAVRAS - MG

2014

FREDERICO SANTOS DE OLIVEIRA

UMA AVALIAÇÃO DE MOVIMENTO DE MALHAS BASEADO NA FORMULAÇÃO LAPLACIANA NA RESOLUÇÃO DA EQUAÇÃO DO CALOR POR DISCRETIZAÇÕES DE VOLUMES FINITOS COM REFINAMENTO DE DELAUNAY

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, área de concentração em Inteligência Computacional e Processamento Gráfico, para a obtenção do título de Mestre.

Orientador

Prof. D.Sc. Sanderson L. Gonzaga de Oliveira

LAVRAS - MG

2014

Ficha Catalográfica Elaborada pela Coordenadoria de Produtos e Serviços da Biblioteca Universitária da UFLA

Oliveira, Frederico Santos de.
Uma avaliação de movimento de malhas baseado na formulação laplaciana na resolução da equação do calor por discretizações de volumes finitos com refinamento de Delaunay/ Frederico Santos de Oliveira. – Lavras : UFLA, 2014.
123 p. : il.

Dissertação (mestrado) – Universidade Federal de Lavras, 2014.
Orientador: Sanderson L. Gonzaga de Oliveira.
Bibliografia.

1. Geração de malhas. 2. Refinamento adaptativo. 3. Malhas móveis. I. Universidade Federal de Lavras. II. Título.

CDD – 006.6

FREDERICO SANTOS DE OLIVEIRA

UMA AVALIAÇÃO DE MOVIMENTO DE MALHAS BASEADO NA FORMULAÇÃO LAPLACIANA NA RESOLUÇÃO DA EQUAÇÃO DO CALOR POR DISCRETIZAÇÕES DE VOLUMES FINITOS COM REFINAMENTO DE DELAUNAY

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, área de concentração em Inteligência Computacional e Processamento Gráfico, para a obtenção do título de Mestre.

APROVADA em 28/02/2014

D.Sc. Daniel Furtado Ferreira	UFLA
Ph.D. João Manuel R. S. Tavares	FEUP

Prof. D.Sc. Sanderson L. Gonzaga de Oliveira

Orientador

LAVRAS - MG

2014

AGRADECIMENTOS

À Universidade Federal de Lavras (UFLA) e ao Departamento de Ciência da Computação (DCC), pela oportunidade concedida para realização do mestrado.

À Fundação de Amparo a Pesquisa do Estado de Minas Gerais (Fapemig) pela concessão da bolsa de estudos.

Aos professores do Departamento de Ciência da Computação da UFLA, pelos ensinamentos transmitidos e harmoniosa convivência.

Ao professor Sanderson L. Gonzaga de Oliveira pela orientação, dedicação e ensinamentos que foram de grande relevância para a realização deste trabalho.

Aos meus amigos de mestrado Guilherme, Jéssica, Ramon, Rodrigo.

Aos meus amigos Betão, Diego, Eduardo, Frango, Guto, Letícia, Mateus, Monserrat, Nestor, Paulo Elias, Rafa, Vladimir e a minha namorada Tamis.

À minha família, Carlos, Eva e Tainá.

À Deus.

RESUMO

Neste trabalho, realiza-se a aplicação de um esquema, combinando o refinamento de delaunay e malhas móveis por volumes finitos na solução da equação do calor. Utiliza-se o algoritmo de Green e Sibson (1978) para geração da malha inicial, o algoritmo de Ruppert (1995) para realizar o refinamento adaptativo e o algoritmo de Üngör (2004, 2009) para realizar melhorias na qualidade da malha. Movimenta-se os vértices para regiões de grande variação por meio de uma nova função monitora proposta, baseada na suavização laplaciana, que é responsável por guiar o movimento dos vértices. Compara-se a função monitora proposta com outras quatro funções monitoras, também baseadas na suavização laplaciana, presentes na literatura. Os resultados comparativos mostram que a nova função monitora proposta, mesmo não demandando o menor esforço computacional em relação às demais, apresenta a menor quantidade final de vértices. Todos os procedimentos para realização dos experimentos são detalhados ao longo deste trabalho.

Palavras-chave: Geração de Malhas; Refinamento Adaptativo; Malhas Móveis

ABSTRACT

In this paper we apply a scheme, combining the refinement and delaunay meshing in finite volume solution of the heat equation. We use the algorithm Green and Sibson (1978) to generate the initial mesh, the algorithm Ruppert (1995) to perform adaptive refinement algorithm and Üngör (2004, 2009) to make improvements on mesh quality. We move the vertices to great variability regions using a new proposed monitoring function based on the Laplacian smoothing, which is responsible for guiding the movement of the vertices. We compare the new proposed monitoring function with four other monitor functions, also based on the Laplacian smoothing that we find in the literature. The comparative results show that the new proposed monitoring function, while not requiring the least computational effort compared to the others, has the lowest amount of end vertices. All procedures for the experiments are detailed throughout this work.

Keywords: Meshes Generation; Adaptive Refinement; Moving Meshes

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

AMR	*Adaptive mesh refinement*
CER	*Circunradius-to-shortest edge radio*
CMr	Cuthill-McKee reverso
EDP	Equação diferencial parcial
FEUP	Faculdade de Engenharia Universidade do Porto
GMP	*GNU Multiple-Precision Library*
KHI	Instabilidade de Kelvin-Helmholtz
MGC	Método dos gradientes conjugados
MHD	Magneto-hidrodinâmicos
MPFR	*Multiple-Precision Floating-point computations with correct Rounding*
MVF	Método dos volumes finitos
PE	Princípio de equidistribuição
PSLG	*Planar Straight Line Graph*
SPH	*Smoothed particle hydrodynamics*
SRQ	*Shape Regularity Quality*
UFLA	Universidade Federal de Lavras
VC	Volume de controle
VPH	Partículas de Voronoi hidrodinâmicas

SUMÁRIO

1 INTRODUÇÃO

Técnicas de simulação computacional são utilizadas para descrever diversos fenômenos físicos que são modelados por equações diferenciais parcais (EDPs). Avanços nessas técnicas e o desenvolvimento de computadores de alto desempenho permitiram que fossem aplicadas em diferentes áreas, como o escoamento de fluidos, aerodinâmica, condução térmica, reações químicas, combustão e eletromagnetismo entre outros. Muitas vezes, a complexidade dos modelos que descrevem esses fenômenos torna impraticável o desenvolvimento de soluções analíticas. Desse modo, deve-se recorrer a simulações numéricas, que constituem ferramentas eficientes para solucionar esses problemas. No entanto, para simulação de problemas de grande dimensão, as técnicas numéricas utilizadas devem ser suficientemente eficientes de modo a possibilitarem resultados precisos em tempo viável.

É comum o uso de métodos numéricos, como o método dos elementos finitos e o método dos volumes finitos, que realizam a decomposição do domínio em formas geométricas chamadas, respectivamente, elementos ou volumes de controle. Suas formas variam de tetraedros ou triângulos a quadriláteros, prismas, pirâmides ou hexaedros.

> Segundo Oliveira et al. (2013): A solução aproximada por meio de um método numérico é obtida para um número discreto de pontos, com um determinado erro, proveniente da aproximação e, se o método for convergente, conforme aumenta-se a quantidade de pontos, melhora-se a solução aproximada. A malha pode ser considerada o domínio discretizado geometricamente.

De acordo com Oliveira et al. (2013), EDPs que modelam fenômenos físicos frequentemente possuem grandes variações na solução, ou ainda

envolvem domínios de geometria complexa, ocasionando erros na solução. Uma estratégia utilizada para se diminuir esses erros é realizar o refinamento da malha. Entretanto, ainda segundo Oliveira et al. (2013, p. 11), "ao se utilizar uma malha fina e uniforme ao longo de todo o domínio, tem-se também o aumento do custo computacional, como consequência do aumento do número de pontos." Uma forma de se contornar esse problema é posicionar mais pontos nas regiões de grande variação e menos pontos nas regiões de pequena variação, realizando o refinamento adaptativo, ocorrendo uma economia do custo computacional e uma melhoria da solução gerada.

Dentre os principais aspectos levados em consideração nas técnicas de geração de malhas, destacam-se o tempo de processamento para geração e a qualidade da malha gerada. O tempo de processamento depende da discretização do domínio e, consequentemente, da quantidade de pontos dessa discretização. Já a qualidade, leva em consideração a quantidade, a forma e a distribução dos elementos da malha. A qualidade da malha influencia diretamente nos resultados, tanto nos aspectos visuais quanto no erro da solução. Malhas triangulares tendem a melhor se adaptar ao domínio da solução. Devido a isso, diversas pesquisas de geração e refinamento de malhas vêm sendo realizadas ao longos dos anos. Algumas são brevemente analisadas nos capítulos seguintes deste trabalho.

Também existem várias pesquisas relacionadas às técnicas de adaptatividade de malhas, e algumas delas são citadas neste trabalho, focadas na adaptatividade por refinamento e por movimento dos vértices. Nas técnicas de adaptatividade por refinamento, são inseridos novos pontos nas regiões de interesse. Nas técnicas de adaptatividade por movimento, chamadas de malhas móveis, mantém-se a quantidade original de pontos e realiza-se a realocação destes. Como a inclusão de pontos aumenta o custo computacional,

neste trabalho há uma descrição das técnicas de adaptatividade de malhas com foco em malhas móveis.

1.1 Objetivos

A seguir serão apresentados o objetivo geral e os específicos deste trabalho.

1.1.1 Objetivo geral

Este trabalho tem o objetivo geral de aplicar um esquema que combine o refinamento de Delaunay e malhas móveis por volumes finitos na solução de equações diferenciais parciais de segunda ordem, como a equação de condução do calor.

1.1.2 Objetivos específicos

Os objetivos específicos principais são: revisar os conceitos de geração de malhas para resolução de EDPs utilizando o método dos volumes finitos; revisar técnicas de adaptatividade por inclusão de pontos e por malhas móveis; implementar um algoritmo híbrido de refinamento de Delaunay para a geração da malha inicial; gerar e implementar uma técnica de movimento de malhas que combine simplicidade, eficiência e velocidade; realizar experimentos computacionais para comparar os resultados com soluções encontradas na literatura.

1.2 Organização deste trabalho

Este trabalho está organizado em cinco capítulos. No capítulo (2), elucida-se o referencial teórico para desenvolvimento deste trabalho. Nesse capítulo, apresenta-se a triangulação de Delaunay e o seu dual, o diagrama

de Voronoi; a discretização da equação do calor, utilizando-se o método dos volumes finitos com o diagrama de Voronoi; a construção do método dos gradientes conjugados, com pré-condicionamento e reordenação da matriz de coeficientes; e, por fim, aborda-se a suavização laplaciana e as principais características sobre malhas móveis.

No capítulo (3), tem-se os detalhes da implementação deste trabalho. Descreve-se o procedimento de refinamento adaptativo e o procedimento que realiza o movimento dos vértices por meio de pseudocódigos. No capítulo (4), expõe-se os resultados dos experimentos realizados, conforme demonstrado no capítulo que o antecede.

Finalizando este trabalho, no capítulo (5), apresenta-se a conclusão acerca dos resultados dos experimentos, bem como sugestões de trabalhos futuros.

2 REFERENCIAL TEÓRICO

Neste capítulo é apresentado o referencial teórico para desenvolvimento deste trabalho. Descreve-se a triangulação de Delaunay e o seu dual, o diagrama de Voronoi, na seção (2.1). Também apresenta-se, nessa seção, algoritmos para geração, manutenção e refinamento de malhas de Delaunay.

Na seção (2.2), realiza-se a discretização da equação do calor, utilizando-se o método dos volumes finitos com o diagrama de Voronoi e define-se um algoritmo em forma de pseudocódigo referente a essa discretização.

Na seção (2.3), tem-se a construção do método dos gradientes conjugados, apresentando-se um pseudocódigo. Com o intuito de se diminuir o número de iterações, também expõe-se nessa seção uma forma de pré-condicionamento da matriz de coeficientes e um algoritmo para reordenação dos vértices, o algoritmo de Cuthill-McKee reverso em conjunto com o algoritmo de vértice pseudoperiférico.

Na seção (2.4), descreve-se a suavização laplaciana. Também são apresentadas técnicas de melhoria da qualidade da malha utilizando a suavização laplaciana.

Na seção (2.5), aborda-se as principais características sobre malhas móveis, como adaptatividade de malhas, o princípio de equidistribuição, também apresentando diferentes trabalhos que pretendem solucionar equações diferenciais parciais por diferentes métodos, utilizando os mais variados tipos de malhas. Também apresentam-se técnicas de movimento dos vértices baseadas na formulação laplaciana.

Na seção (2.6), explana-se uma métrica geométrica para verificar a degeneração da malha ao movimentar os vértices.

2.1 A triangulação de Delaunay e o diagrama de Voronoi

Nesta seção, aborda-se a triangulação de Delaunay e o diagrama de Voronoi. Também são apresentados algoritmos de construção, manutenção e refinamento dessa triangulação.

2.1.1 Considerações iniciais

Uma malha pode ser classificada de acordo com a disposição relativa dos diferentes elementos existentes. Uma malha *estruturada* é caracterizada pela sua conectividade *regular*. Isso significa que, a posição dos nós pode ser mapeada de modo a definir quais nós serão vizinhos por meio de cálculo. Isso pode restringir as escolhas dos elementos para quadriláteros em 2D ou hexaedros em 3D. Se os nós não podem ser arranjados de tal forma, a malha é dita *não estruturada* ou *irregular*. Em uma malha irregular, deve-se armazenar a conexão entre os nós que formam a malha. Os elementos podem ser, em sua forma básica, triângulos ou tetraedros em 2D e 3D, respectivamente, bem como hexaedros, ou qualquer outra forma, em malhas mais complexas (THOMPSON; SONI; WEATHERHILL, 1999).

Na busca de se atingir uma determinada qualidade na aproximação, com desempenho computacional aceitável, ao longo das últimas décadas, pesquisas têm sido realizadas em técnicas de geração e refinamento de malhas, como por exemplo em Chen e Xu (2004), Frey e George (2008), Preparata e Shamos (1985), Teng e Wong (2000) e Thompson, Soni e Weatherhill (1999). Ainda em problemas com domínios com geometria complexa, a geração da malha pode ser uma tarefa dispendiosa. Uma forma de se obter uma malha de qualidade a baixo custo computacional é a utilização da triangulação Delaunay.

A triangulação de Delaunay (1939) é um tipo de malha irregular. Com uma triangulação de Delaunay, maximiza-se o menor ângulo da triangulação. Obtém-se triângulos com ângulos não próximos a 0° e a 180°. Em esquemas numéricos, triângulos com ângulos próximos a 0° e 180° são considerados de má qualidade, levando a resultados imprecisos.

A triangulação de Delaunay (1939) possui a característica de unicidade, de forma que, dado um conjunto de vértices, a triangulação é sempre única, exceto em casos degenerados, nos quais quatro ou mais vértices são cocirculares, existindo duas possibilidades de triangulação (local). Cada uma das duas possibilidades de triangulação que divide o quadrilátero em dois triângulos satisfaz a "condição Delaunay", isto é, em que o circuncírculo (único círculo que passa pelos vértices do triângulo) é vazio. Essa noção pode ser estendida a três dimensões.

Um diagrama de Voronoi (1908) é uma decomposição (em partições) de um espaço em torno de um vértice. Cada polígono consiste em uma região de espaço que fica mais próxima de cada vértice do que qualquer outro vértice gerador do diagrama de Voronoi, que é formado pelos polígonos de Voronoi. Diversos algoritmos para a geração do diagrama de Voronoi foram desenvolvidos. São exemplos desses algoritmos, Shamos e Hoey (1975), que desenvolveram um algoritmo $O(n \cdot \lg(n))$ por divisão-e-conquista e Green e Sibson (1978), que produziram um algoritmo $O(n^2)$ por inserção incremental, em que n é a quantidade de vértices da triangulação.

Uma partição do diagrama de Voronoi e a triângulação de Delaunay correspondente pode ser observada na figura (1).

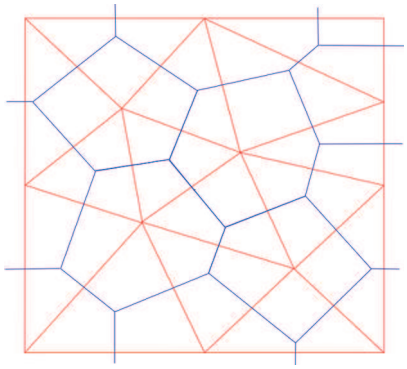

Figura 1 Diagrama de Voronoi
Legenda: Em azul, arestas de uma partição do diagrama de Voronoi e, em vermelho, os triângulos de Delaunay correspondentes.

2.1.2 Manutenção da triangulação de Delaunay - Algoritmo de Lawson

Para construção da triangulação Delaunay, pode-se utilizar o algoritmo de Lawson (1977), um algoritmo de complexidade $O(n^2)$. Esse algoritmo é utilizado para gerar ou para manter uma triangulação de Delaunay. Sua escolha deve-se à sua facilidade de implementação.

A partir de uma triangulação inicial, verifica-se se todas as arestas que compõem a triangulação atendem aos critérios da triangulação de Delaunay. Caso essas arestas não atendam aos critérios da triangulação de Delaunay, realiza-se o *flip*.

O *flip* é um procedimento, definido por Lawson (1977), em que ocorre a inversão da aresta invadida, conforme demonstrado na figura (2). Com o *flip*, uma nova aresta é criada e, em seguida, verifica-se, recursivamente, essa aresta e os dois triângulos que a compartilham. O algoritmo termina quando todas as arestas atendem aos critérios da triangulação de Delaunay, ou seja, são localmente de Delaunay.

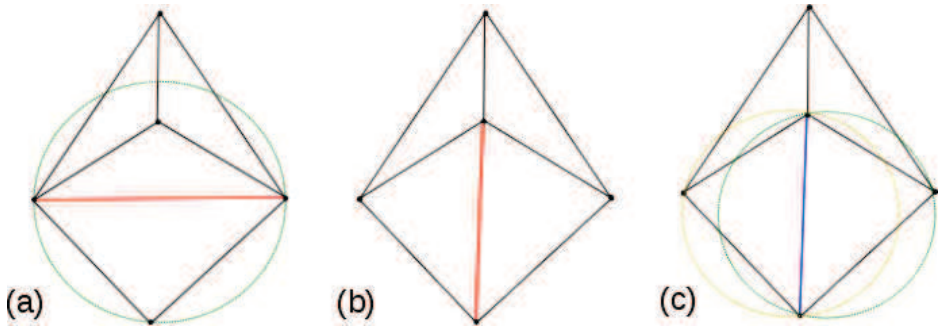

Figura 2 Algoritmo de Lawson
Legenda: Aplicação do algoritmo de Lawson (a) Uma aresta invadida, mostrada em vermelho, que não é de Delaunay, é verificada. (b) O algoritmo do *flip* é aplicado para inverter a aresta invadida, tornando-a localmente de Delaunay. (c) Verifica-se se essa aresta e os dois triangulos que a compartilham são localmente de Delaunay. Como a aresta não é invadida, essa é mostrada em azul.

No algoritmo (1) tem-se um pseudocódigo referente ao algoritmo de Lawson (1977).

Entrada: Triangulação T.
Saída: Triangulação de Delaunay T.
1 **início**
2 **para** *(cada aresta* $\overline{ab} \in T$*)* **faça**
3 `// Efetua teste do circuncírculo.`
4 **se** *(*$\overline{ab}$ *não é localmente de Delaunay)* **então**
5 Sejam Δabc e Δabd os triângulos que compartilham $\overline{ab}$;
6 Trocar $\overline{ab}$ por $\overline{cd}$; `// Realiza o` *flip* `da aresta.`
7 **retorna** T*;*

Algoritmo 1: Algoritmo de Lawson.

2.1.3 Construção da triangulação de Delaunay - Algoritmo de Green-Sibson

O algoritmo de Green-Sibson foi proposto por Green e Sibson (1978) para gerar o diagrama de Voronoi, porém, pode ser facilmente adaptado

para gerar a triangulação de Delaunay. A partir de uma triangulação de Delaunay inicial realiza-se a inserção incremental de vértices. Os pontos inseridos na triangulação de Delaunay são chamados de pontos de *Steiner*. A complexidade desse algoritmo é $O(n^2)$.

Nesse algoritmo, ocorre a inserção de um vértice por vez e, a cada inserção, verifica-se se o novo vértice encontra-se sobre uma aresta ou dentro de um triângulo. Caso o novo vértice e se localize dentro do triângulo Δabc, liga-se e aos três vértices do triângulo que o contém, criando-se três novos triângulos, Δabe, Δace e Δbce. Caso o novo vértice f se localize sobre uma aresta $\overline{de}$, deve-se dividir $\overline{de}$ em duas novas arestas. Considere Δbde e Δcde os triângulos que compartilham $\overline{de}$. Divide-se a aresta $\overline{de}$ em duas novas arestas, $\overline{ef}$ e $\overline{df}$. Em seguinda, liga-se f aos vértices b e c, opostos à aresta $\overline{de}$, pertencentes aos triângulos que compartilham $\overline{de}$, criando-se quatro novos triângulos, Δbef, Δbdf, Δcef e Δcdf. Em ambos os casos realiza-se o teste do circuncírculo após a geração das novas arestas, utilizando o algoritmo de Lawson (1977) para manutenção da malha. Se existirem arestas invadidas, realiza-se o *flip*. O algoritmo termina quando não existirem arestas invadidas. Na figura (3), pode-se observar um exemplo de execução do algoritmo de Green e Sibson (1978).

No algoritmo (2) tem-se um pseudocódigo referente ao algoritmo de Green e Sibson (1978).

2.1.4 Refinamento de Delaunay

Nesta seção, são apresentados dois algoritmos no estado da arte para o refinamento de Delaunay: algoritmo de Ruppert (1995) e *off-centers* (OLIVEIRA, 2012; ÜNGÖR, 2004, 2009).

Algoritmo de Ruppert

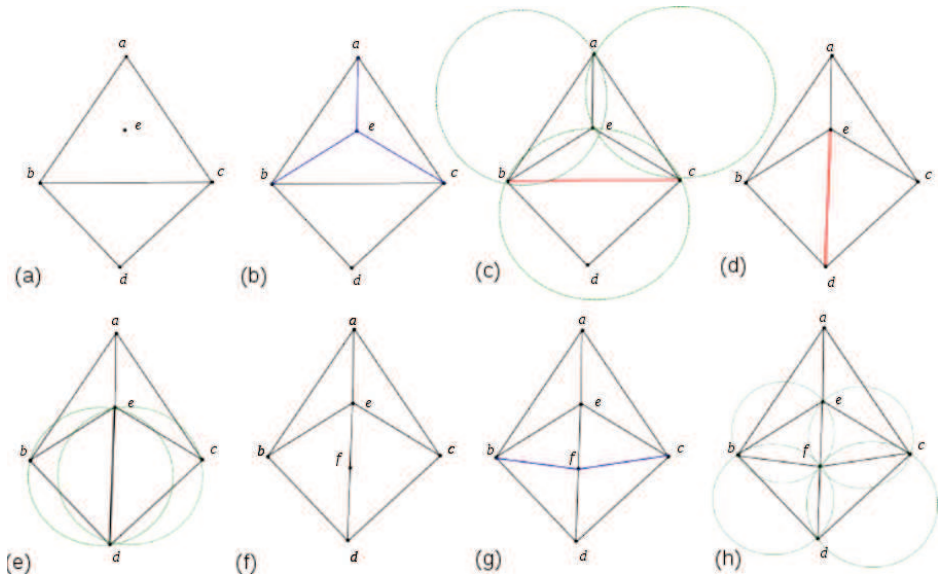

Figura 3 Algoritmo de Green Sibson:
Legenda: Aplicação do algoritmo de Green Sibson (a) Tem-se a malha inicial com a inserção de um novo vértice que localiza-se dentro de um triângulo. (b) Divide-se o triângulo gerando três novos triângulos, formados pelas arestas em azul. (c) Realiza-se o teste dos circuncírculos e encontra-se uma aresta invadida, mostrada em vermelho. (d) Efetua-se o *flip* na aresta invadida. (e) Realiza-se o teste dos circuncírculos. (f) Ocorre a inserção de um novo vértice localizado sobre uma aresta. (g) Ocorre a divisão da aresta que contém o novo vértice, gerando quatro novos triângulos, compostos pelas novas arestas em azul. (h) Realiza-se o teste dos circuncírculos e verifica que não existe aresta invadida.

O algoritmo de Ruppert (1995) garante que todos os triângulos pertencentes à triangulação terão ângulos entre α e $\pi - 2\alpha$, de forma que α pode ser um ângulo máximo de aproximadamente 20,7 graus. Ao se inserir um vértice na triangulação, duas operações são possíveis: dividir um segmento e dividir um triângulo. Ao se dividir um segmento, um vértice é inserido em seu ponto médio. Ao se dividir um triângulo, um vértice é inserido em seu circuncentro. Com isso, uma nova triangulação é realizada (OLIVEIRA, 2012).

Um algoritmo para a geração da triangulação de Delaunay pode utilizar uma medida de qualidade. Uma possível medida de qualidade chama-se

Entrada: Triangulação de Delaunay TD e lista L_V de vértices a serem inseridos em TD.
Saída: TD com os vértices pertencentes à L_V.

```
1  início
2      para (todo vértice v ∈ L_V) faça
3          Inserir v em TD;
4          se (v localiza-se sobre aresta ab ∈ TD) então
5              Sejam Δabc e Δabd os triângulos que compartilham ab;
6              Criar as arestas av, bv, cv, dv e inserir em TD;
7          senão
8              Seja Δabc o triângulo que contém v;
9              Criar as arestas av, bv, cv e inserir em TD;
10     para (cada aresta ab criada) faça
11         AlgoritmoLawson(ab);
12     retorna TD;
```

Algoritmo 2: Algoritmo de Green-Sibson.

Circunradius-to-shortest Edge Radio (CER), que é definida pela razão do raio do circuncírculo r (*circunraio*) do triangulo, pela menor aresta l do triângulo. É especificado um limite superior $\rho = r/l$ para o CER de todos os triângulos da malha. A razão ρ de um triângulo está relacionada com seu menor ângulo α pela fórmula $\rho = 1/[2 \cdot sen(\alpha)]$. Quanto menor for a razão ρ, maior será o ângulo α do triângulo. Esse limite superior ρ garantirá que não há triângulo na malha com ângulo menor que $\arcsin(1/2\rho)$ (PÉBAY; BAKER, 2003).

Esse algoritmo pode receber como parâmetro de entrada um grafo planar de linhas não curvas (*Planar Straight Line Graph* - PSLG). Um PSLG é um conjunto de vértices e seus segmentos. Segmentos são arestas que não podem ser removidos. Claramente, os segmentos não se interceptam. Um segmento é considerado invadido quando um vértice incide dentro de seu círculo diametral. Um pseudocódigo referente ao algoritmo de Ruppert

(1995) é apresentado no algoritmo (3).

Entrada: PSLG G.
Saída: Triangulação de Delaunay de G com todos os $\measuredangle \geq \alpha$.

```
1  início
2      Adicionar um delimitador quadrado D à G:
3      início
4          Calcular os extremos de G: x_min, y_min, x_max, y_max;
5          Seja span(G) = max(x_max − x_min, y_max − y_min);
6          Seja D o quadrado de lado 3 × span(G), centralizado em
           G;
7          Adicionar os quatro segmentos de contorno de D à G;
8      Seja a lista de segmentos L_S = arestas de G;
9      Seja a lista de vértices L_V = vértices de G;
10     Construir a triangulação de Delaunay inicial TD(L_V);
11     repita
12         // Divide todos os segmentos invadidos em seu
13         // ponto médio.
14         enquanto (existe algum segmento s ∈ L_S invadido) faça
15             DividirSegmento(s);
16         // Verifica triângulos com ∠ < α.
17         enquanto (existe algum triângulo δ ∈ TD(L_V) com
           ∠ < α;) faça
18             Seja c o circuncentro de δ;
19             se (c invade algum segmento s ∈ L_S) então
20                 // Divide todos os segmentos invadidos
21                 // em seu ponto médio.
22                 DividirSegmento(s);
23             senão
24                 // Divide triângulos com ∠ < α, em seu
25                 // circuncentro, adicionando c à L_V.
26                 DividirTriangulo(δ);
27     até (até que nenhum segmento esteja invadido e nenhum
       ∠ < α);
28     retorna Triangulação de Delaunay corrente TD(L_V);
```

Algoritmo 3: Refinamento de Ruppert.

Nas situações em que o PSLG possui ângulos menores que 90°, o

algoritmo tende a formar triângulos com ângulos muito agudos. Shewchuk (1997) provou que o algoritmo irá parar para qualquer entrada com ângulos de, no mínimo, 60°.

Off-centers

Üngör (2004, 2009) propôs um refinamento de Delaunay similar ao algoritmo de Ruppert (1995). Em seu trabalho, introduz-se um novo tipo de pontos de *Steiner*, chamados *off-centers*, como uma alternativa aos circuncentros, apresentando um novo algoritmo de refinamento de Delaunay. Caso um triângulo seja considerado de má qualidade, pela medida CER, o algoritmo tenta inserir o *off-center*. Caso o *off-center* invada algum segmento, então, o novo vértice é inserido no ponto médio desse segmento em vez do *off-center*. Um pseudocódigo referente a esse algoritmo pode ser observado no algoritmo (4).

Entrada: PSLG G.
Saída: Triangulação de Delaunay de G com todos os $\measuredangle \geq \alpha$.

```
1  início
2      Seja TD a triangulação de Delaunay dos vértices de G;
3      Calcular:
4        B =candidados à off-centers que invadem algum segmento;
5        C =candidados à off-centers que não invadem segmentos;
6        D =candidados à ponto médio;
7      enquanto (C∪D) ≠ vazio faça
8          Escolher um ponto c_o ∈ (C∪D) e inserir c_o na
           triangulação TD;
9          se (c_o é ponto médio de um segmento s) então
10             DividirSegmento(s);
11         Atualizar TD e recalcular B, C e D;
12     retorna Triangulação de Delaunay corrente TD;
```

Algoritmo 4: Refinamento de Üngor.

Para calcular o *off-center*, considera-se um triângulo δ, de má qualidade, formado pelos vértices p, q e r. A menor aresta de δ é pq e o seu

circuncentro é c. O *off-center* c_o de δ é o seu circuncentro se o CER do triângulo formado pelos vértices p, q e c for menor ou igual a um dado limite ρ_α, em que ρ_α é a medida CER de um ângulo α, conforme mostrado na figura (4.a). Caso contrário, o *off-center* c_o será o ponto na bissetora da aresta, dentro do circuncírculo, que faz com que o CER do triângulo Δpqc_o seja exatamente igual a ρ_α. A bissetora da aresta é a linha que passa pelo ponto médio de uma aresta de um triângulo e pelo seu circuncentro.

O círculo que passa pelos vértices da menor aresta pq, centrado no *off-center*, é chamando *off-circle*. Caso o triângulo δ tivesse duas arestas menores iguais, escolheria-se uma delas, arbitrariamente.

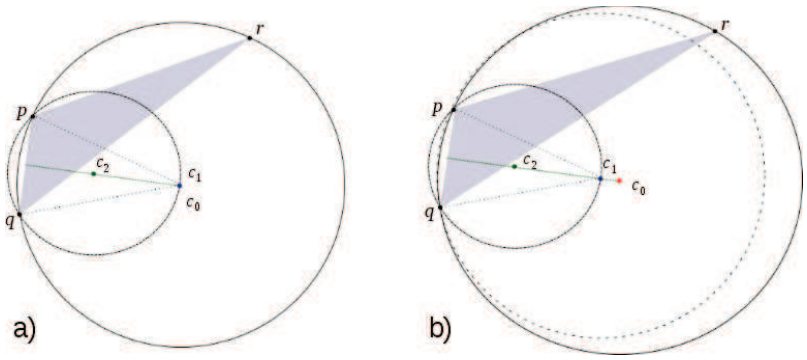

Figura 4 *Off-center*
Legenda: O *off-center* e o circuncentro do triângulo δ, formado pelos vértices p, q e r, são, respectivamente, c_0 e c_1. O circuncentro do triângulo formado pelos vértices p, q e c_0 é c_2. Se $|c_0c_2| \leq \rho_\alpha|pq|$ então $c_0 = c_1$, que é demonstrado em (a). Caso contrário, $c_0 \neq c_1$. Nesse caso, calcula-se c_2 de forma que $|c_0c_2| = \rho_\alpha|pq|$, que é demonstrado em (b). O *off-circle* do triângulo δ é o circuncírculo em (a) e é mostrado em linhas tracejadas em (b). Figura adaptada de Üngör (2004, 2009).

Üngör (2004, 2009) mostrou que seu algoritmo possui as mesmas garantias do algoritmo de Ruppert. Seus experimentos mostraram que seu algoritmo insere aproximadamente 40% menos vértices que outros algoritmos de inserção no circuncentro e malhas 30% menores em relação ao número de elementos.

2.1.5 Resumo

Nesta seção, apresentou-se a triangulação de Delaunay, um tipo de malha irregular que possui a propriedade de maximizar o menor ângulo da triangulação e o seu dual, o diagrama de Voronoi.

Nas subseções (2.1.2), (2.1.3) e (2.1.4) foram explanados um algoritmo para manutenção da triangulação de Delaunay, o algoritmo de Lawson (1977), um algoritmo para construção da triangulação de Delaunay, o algoritmo de Green e Sibson (1978), e dois algoritmos para refinamento de Delaunay: o algoritmo de Ruppert (1995) e Üngör (2004, 2009).

2.2 Discretização da equação do calor

Nesta seção, aborda-se a discretização da equação do calor utilizando-se o método dos volumes finitos com o diagrama de Voronoi. Também mostra-se um pseudocódigo referente a esse método.

2.2.1 Considerações iniciais

No método dos volumes finitos (MVF), o domínio do problema é dividido em um conjunto finito de volumes, adjacentes entre si, chamados de volumes de controle (VC). Nesse método, realiza-se um balanço de conservação da propriedade em questão, para cada volume elementar, de modo a se obter a equação aproximada, respeitando-se a lei de conservação. Há interesse na aplicação do método dos volumes finitos na solução de equações diferenciais parciais (EDPs) devido a essa característica.

Uma forma de se obter as equações aproximadas no MVF é integrar, sobre cada volume elementar, no espaço e no tempo, as equações na forma conservativa, ou divergente. Nessa forma, os termos relacionados aos fluxos

aparecem dentro das derivadas em relação às coordenadas espaciais. Ao realizar a integração para todos os VCs, obtém-se uma equação algébrica para cada volume e, consequentemente, obtém-se um sistema de equações algébricas. Esse sistema de equações é formado pelas variáveis localizadas nos centróides dos VCs. Para as aproximações, utilizam-se funções de interpolação por meio dos valores das variáveis dos VCs vizinhos. Devido a sua generalidade, qualquer tipo de malha pode ser utilizada, regular ou irregular (MALISKA, 2010).

Na seção (2.2.2), apresenta-se o modelo matemático para difusão de fluidos. Na seção (2.2.3), é tratada a discretização da equação do calor com o diagrama de Voronoi e apresenta-se o pseudocódigo dessa discretização.

2.2.2 Equação do calor

A equação do calor é um modelo matemático para a difusão de calor em sólidos, ou seja, descreve o fluxo de calor em um corpo sólido. Esse modelo consiste em uma equação de derivadas parciais que, muitas vezes, é também chamada de equação de difusão. Essa equação, em sua forma generalizada, é definida como

$$\frac{\partial}{\partial t}(\rho\phi) = \nabla \cdot \left(\frac{k}{c_p}\nabla\phi\right), \tag{1}$$

em que ϕ é a temperatura, t é o tempo, ρ é a massa volumétrica do material, c_p é o seu calor específico e k é a condutibilidade térmica.

Segundo Tannehill, Anderson e Pletcher (1997, p. 73), pode-se integrar a equação (1) sobre um volume V qualquer obtendo-se

$$\iiint_V \left(\frac{\partial}{\partial t}(\rho\phi) + \nabla \cdot \mathbf{q}\right) dV = 0,$$

com $\mathbf{q} = -\frac{k}{c_p}\nabla\phi$. Aplicando-se o teorema de Gauss, tem-se

$$\iiint_V \frac{\partial}{\partial t}(\rho\phi)dV + \oiint_S \mathbf{q}\cdot\mathbf{n}dS = 0.$$

O "volume" contém uma unidade de profundidade em problemas bidimensionais. Nesse caso, $\mathbf{n}dS$ pode ser representado por $idy - jdx$ para um caminho de integração na fronteira em sentido horário. Essa equação segue a lei de conservação. O primeiro termo na equação, uma integral sobre o volume, significa a taxa de variação da energia armazenada no volume em função do tempo. O segundo termo é uma integral de linha sobre a superfície do volume e representa a vazão de energia ao longo da superfície do volume na unidade de tempo (TANNEHILL; ANDERSON; PLETCHER, 1997).

2.2.3 Discretização da equação do calor com o diagrama de Voronoi

Nesta seção, é tratada a discretização da equação do calor (1) com o diagrama de Voronoi. Uma partição do diagrama de Voronoi pode ser vista na figura (1), na página 19. Esta discretização baseia-se no capítulo 13 de Maliska (2010) e no capítulo 8 de Versteeg e Malalasekera (2007).

Na figura (5), mostra-se o volume elementar com centróide P, sobre o qual será realizada a integração, e o VC adjacente, com centróide Nb_i. A equação discretizada é obtida integrando-se a equação (1) em cada VC, no intervalo de tempo de t a $t+\Delta t$.

Integrando-se a equação (1) no tempo e no espaço e adotando-se uma formulação implícita, obtém-se $\int_t^{t+\Delta t}\int_V \frac{\partial}{\partial t}(\rho\phi)dVdt = \int_t^{t+\Delta t}\int_V \nabla\cdot\left(\frac{k}{c_p}\nabla\phi\right)dVdt$. O ponto de integração pi localiza-se no ponto médio da interface dos polígonos com centróides P e Nb_i. Aplicando-se o teorema da

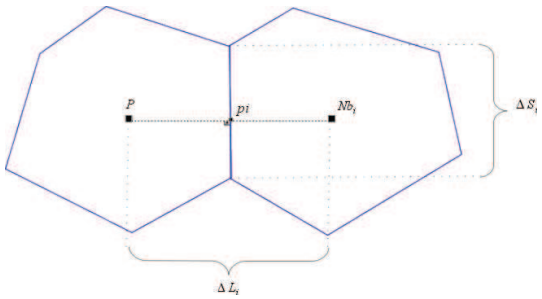

Figura 5 Volume de controle
Legenda: Volumes de controle P e Nb_i para a integração.

divergência e utilizando-se uma aproximação linear, tem-se

$$\frac{M_P^{t+\Delta t}\phi_P^{t+\Delta t} - M_P^t\phi_P^t}{\Delta t} = \sum_{i=1}^{n}\left(\frac{k}{c_P}\left(\phi_{Nb_i}^{t+\Delta t} - \phi_P^{t+\Delta t}\right)\frac{\Delta S_i}{\Delta L_i}\right), \tag{2}$$

em que $M_P = \rho \cdot A$ é a massa dentro do volume de controle e A é a área do polígono com centróide P. ϕ_P é a temperatura do volume de controle com centróide P, ϕ_{Nb_i} é a temperatura no volume de controle com centróide Nb_i e n é o número de VCs adjacentes ao VC com centróide P. ΔS_i é o tamanho da interface entre os polígonos com centróide P e seu polígono adjacente com centróide Nb_i e ΔL_i é a distância entre os pontos P e Nb_i. A equação (2) pode ser escrita como

$$A_P\phi_P^{t+\Delta t} - \sum_{i=1}^{n} A_{Nb_i}\phi_{Nb_i}^{t+\Delta t} = B, \tag{3}$$

em que $A_{Nb_i} = \frac{k}{c_p}\cdot\frac{\Delta S_i}{\Delta L_i}$, $A_P = \sum_{i=1}^{n} A_{Nb_i} + \frac{M_P^{t+\Delta t}}{\Delta t}$ e $B = \frac{M_P^t\phi_P^t}{\Delta t}$.

Caso um ou mais VCs adjacentes ao VC com centróide P esteja localizado na fronteira e considerando-se condições de contorno de Dirichlet, a abordagem é da mesma forma que a empregada na seção anterior. Na aproximação linear da derivada na interface que separa o volume de controle

com centróide P e um volume de controle com centróide F, localizado na fronteira e adjacente ao VC com centróide P, terá o valor da temperatura de F substituído pelo valor prescrito da condição de contorno. Os coeficientes A_{Nb_i}, A_P e B variam de acordo com a quantidade de VCs adjacentes localizados na fronteira. Com um centróide F_i de um VC na fronteira, conforme pode ser visto na figura (6), ΔS_{F_i} é o mesmo que ΔS_i, ou seja, é o tamanho da interface entre os polígonos com centróide P e seu polígono adjacente com centróide F_i e ΔL_{F_i} é o mesmo que ΔL_i, ou seja, é a distância entre os pontos P e F_i.

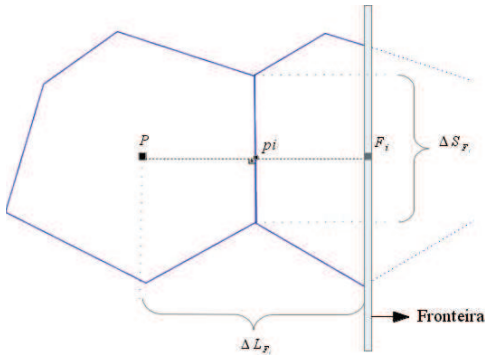

Figura 6 Volume de controle com vizinho na fronteira
Legenda: Volumes de controle P e F_i para a integração com vizinho localizado na fronteira.

A seguir, mostra-se um exemplo para ficar clara a forma com que se aplica o método dos volumes finitos com o diagrama de Voronoi nas ocorrências em que um VC contém VCs adjacentes na fronteira. Considere o volume de controle com centróide P_3 mostrado na figura (7). Os volumes adjacentes ao VC com centróide P_3, internos ao domínio, têm centróides Nb_1, Nb_2 e Nb_3 e os volumes adjacentes ao VC com centróide P_3, localizados na fronteira, contêm valores prescritos e descritos nos pontos F_1, F_2 e F_3, m_1 é o número de volumes internos e adjacentes ao VC com centróide P, m_2

é o número de volumes de fronteira e adjacentes ao VC com centróide P e $n = m_1 + m_2$. Os valores das temperaturas nesses pontos são representados por ϕ_{F_j}. Realizando-se a discretização da equação do calor para um volume localizado na fronteira, a equação (2) torna-se

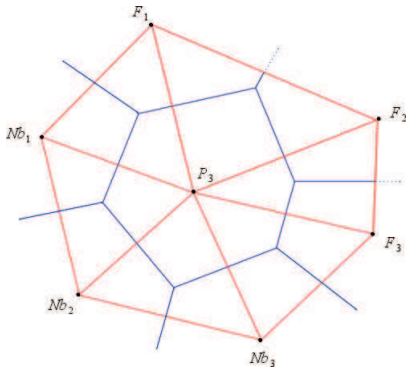

Figura 7 Volume de Voronoi
Legenda: Volume de Voronoi adjacente a VCs com centróides localizados na fronteira. Nb_1, Nb_2 e Nb_3 são os centróides dos VCs internos ao domínio e F_1, F_2 e F_3 contêm valores prescritos em pontos localizados na fronteira.

$$\begin{aligned}\frac{M_P^{t+\Delta t}\phi_P^{t+\Delta t} - M_P^t\phi_P^t}{\Delta t} &= \sum_{i=1}^{m_1}\left(\frac{k}{c_P}\left(\phi_{Nb_i}^{t+\Delta t} - \phi_P^{t+\Delta t}\right)\frac{\Delta S_i}{\Delta L_i}\right) \\ &\quad + \sum_{i=1}^{m_2}\left(\frac{k}{c_P}\left(\phi_{F_i}^{t+\Delta t} - \phi_P^{t+\Delta t}\right)\frac{\Delta S_{F_i}}{\Delta L_{F_i}}\right),\end{aligned}$$

que pode ser reescrita como

$$A_P\phi_P^{t+\Delta t} - \sum_{i=1}^{m_1} A_{Nb_i}\phi_{Nb_i}^{t+\Delta t} = B, \tag{4}$$

em que $A_{Nb_i} = \left(\frac{k}{c_p}\frac{\Delta S_i}{\Delta L_i}\right), A_{F_i} = \left(\frac{k}{c_p}\frac{\Delta S_{F_i}}{\Delta L_{F_i}}\right), A_P = \sum_{i=1}^{m_1} A_{Nb_i} + \sum_{i=1}^{m_2} A_{F_i} + \frac{M_P^{t+\Delta t}}{\Delta t}, B = \frac{M_P^t\phi_P^t}{\Delta t} + \sum_{i=1}^{m_2} A_{F_i}\phi_{F_i}^{t+\Delta t}$. Dessa forma, tem-se a discretização da equação do calor com polígonos de Voronoi.

A seguir, no algoritmo 5, é apresentado um pseudocódigo referente à discretização da equação do calor com polígonos de Voronoi, mostrado na figura (1). Um sistema de equações lineares é gerado, em que a equação de cada VC da malha, interno ao domínio ou localizado na fronteira é, respectivamente, conforme a equação (3) ou conforme a equação (4). No pseudocódigo, os valores dos coeficientes são armazenados em objetos que representam os vértices da triangulação de Delaunay.

Entrada: lista dos vértices da triangulação de Delaunay.
Saída: lista dos vértices da triangulação de Delaunay com os coeficientes A_P, A_{Nb_i} e B preenchidos.

```
1  início
2      para cada ( vértice interno P da triangulação de Delaunay) faça
3          // Area_P é a área do VC com centróide P
4          P.A_P ← (ρ·Area_P)/Δt;
           P.B ← (ρ·Area_P)/Δt · P.valorTemperaturaEtapaAnterior;
5          para cada ( vértice Nb_i adjacente a P) faça
6              P.A_Nb_i ← 0;
7              se (vértice Nb_i localiza-se na fronteira) então
8                  // ΔS_Fi e ΔL_Fi são calculados conforme
9                  // mostrado na fig.(6)
10                 P.A_P ← P.A_P + (k/c_p)(ΔS_Fi/ΔL_Fi);
11                 // valorTemperatura é definida pelas
12                 // condições de contorno
13                 P.B ← P.B + (Nb_i.valorTemperatura · (k/c_p)(ΔS_i/ΔL_i));
14                 // Nb_i localiza-se na fronteira,
15                 // portanto F_i = Nb_i
16             senão
17                 // ΔS_i e ΔL_i são calculados conforme
18                 // mostrado na fig. (5)
19                 P.A_P ← P.A_P + (k/c_p)(ΔS_i/ΔL_i);
20                 P.A_Nb_i ← P.A_Nb_i − (k/c_p)(ΔS_i/ΔL_i);
```

Algoritmo 5: Método dos volumes finitos.

2.2.4 Resumo

Nesta seção, abordou-se a discretização da equação do calor. Primeiramente, na subseção (2.2.2), apresentou-se o modelo matemático que descreve a difusão de calor em sólidos. Na subseção (2.2.3), utilizou-se o MVF com o diagrama de Voronoi. Por fim, apresentou-se um pseudocódigo referente a essa discretização.

2.3 O método dos gradientes conjugados

Nesta seção, apresenta-se o método dos gradientes conjugados (MGC). Em seguida, definem-se os passos do MGC e elabora-se um pseudocódigo. Com o intuito de diminuir o número de iterações, explica-se como realizar o pré-condicionamento e o reordenamento dos vértices.

2.3.1 Considerações iniciais

Um sistema linear, na forma de um produto matricial, pode ser visto como

$$Ax = b, \tag{5}$$

de forma que $A \in \mathbb{R}^{n\times n}$ e os vetores $x, b \in \mathbb{R}^n$ são tomados como vetores coluna. A solução do sistema poderia ser calculada por meio da inversa da matriz A, considerando que A seja inversível, na forma $x = A^{-1}b$. Porém, o cálculo da inversa apresenta imprecisão numérica, além de ser um algoritmo com complexidade $O(n^3)$. Ver Niedu (2012), para detalhes.

O MGC foi, inicialmente, proposto por Hestenes e Stiefel (1952). Foi, originalmente, desenvolvido como um método direto delineado para resolver um sistema linear $n \times n$ positivo definido (BURDEN; FAIRES, 2010).

O MGC é útil na resolução de sistemas esparsos, de tamanho elevado, com n na casa de milhares ou milhões, com entradas não nulas que aparecem em padrões predizíveis. Quando a matriz está pré-condicionada, de forma que os cálculos sejam eficazes, bons resultados são obtidos em, aproximadamente, $\sqrt{n}$ passos, tornando o MGC preferível em relação à eliminação gaussiana e a outros métodos iterativos (BURDEN; FAIRES, 2010).

Na seção (2.3.2), mostra-se que, por meio da minimização de uma função, encontra-se a solução do sistema linear. Na seção (2.3.3), define-se o MGC e um algoritmo em pseudocódigo. Na seção (2.3.4), mostra-se como realizar um pré-condicionamento. Por fim, na seção (2.3.6), tem-se o resumo deste capítulo.

2.3.2 Otimização para resolução de sistemas lineares

A solução de um sistema linear, conforme a equação (5), pode ser encontrada por meio da minimização de uma função $f(x) : \mathbb{R} \to \mathbb{R}$, sujeita a $x \in \mathbb{R}^n$. A demonstração de que a solução do sistema linear é encontrada a partir da minimização de $f(x)$ pode ser encontrada em Burden e Faires (2010).

O vetor x^* é uma solução do sistema linear definido positivo se e, somente se, x^* minimiza

$$f(x) = \frac{x^T Ax}{2} - b^T x. \tag{6}$$

Ao derivar a equação (6) em relação ao vetor x, o resultado é um vetor também pertencente a $\mathbb{R}^n$, que satisfaz $\nabla f(x^*) = Ax^* - b = 0$. O gráfico dessas funções apresenta um comportamento característico, de forma que o centro das curvas de nível é justamente o minimizador de $f(x)$.

2.3.3 Definição do método

O MGC consiste em, a partir de uma estimativa de solução inicial, minimizar o resíduo r, tal que $r = b - Ax$, a cada iteração k, em que $0 \leq k \leq n-1$, ao longo de direções conjugadas. Calcula-se a solução a partir de uma direção de busca d_k e um coeficiente ϑ_k, denominado comprimento do passo, conforme

$$x_{k+1} = x_k + \vartheta_{k+1} d_{k+1}. \tag{7}$$

A direção de busca é definida de forma que

$$d_k = r_{k-1} + \varphi_{k-1} d_{k-1} \tag{8}$$

e o valor do resíduo r_k é dado por

$$r_{k+1} = r_k - \vartheta_{k+1} A d_{k+1}. \tag{9}$$

Os coeficientes ϑ_k e φ_k são calculados conforme

$$\vartheta_k = \frac{r_{k-1}^T r_{k-1}}{d_k^T A d_k} \tag{10}$$

e

$$\varphi_k = \frac{r_k^T r_k}{r_{k-1}^T r_{k-1}}. \tag{11}$$

Dessa forma, o resíduo é minimizado ao longo das direções de busca. A estimativa inicial pode ser $x_0 = 0$, $r_0 = b$ e $d_1 = r_0$. O processo iterativo do MGC, formado pelas equações (7), (9), (8), (10) e (11), pode ser descrito conforme apresentado no algoritmo (6).

Entrada: Matrizes A e b.
Saída: Solução x.

```
1  n ← dimensão do vetor x;
2  // Definir o erro admitido, por exemplo, 10^-3.
3  ε ← 10^-3;
4  // O erro deve ser maior que ε na primeira iteração
5  erro ← ε + 1;
   // Valores para i = 0.  ϑ_0 e φ_0 não são necessários.
6  x_0 ← 0;
7  r_0 ← b − A x_0;
8  d_1 ← r_0;
   // Há n + 1 termos (0 ≤ i ≤ n)
9  i ← 1;
10 enquanto ((i ≤ n) e (erro > ε)) faça
11     ϑ_i ← (r_{i-1}^T r_{i-1}) / (d_i^T A d_i);  // equação (10)
12     x_i ← x_{i-1} + ϑ_i d_i; // equação (7)
13     erro ← ||x_i − x_{i-1}||_∞ / ||x_i||_∞;  // ||x_i||_∞ é a norma infinita de x_i
       // Verifica se a próxima iteração será executada
14     se ((i + 1 ≤ n) e (erro > ε)) então
15         r_i ← r_{i-1} − ϑ_i A d_i; // equação (9)
16         φ_i ← (r_i^T r_i) / (r_{i-1}^T r_{i-1});  // equação (11)
17         d_{i+1} ← r_i + φ_i d_i; // equação (8)
18     i ← i + 1;
```

Algoritmo 6: Método dos gradientes conjugados.

2.3.4 Pré-condicionamento

Segundo Burden e Faires (2010), ao utilizar um pré-condicionamento, o MGC não é aplicado diretamente na matriz A, mas sim em uma outra matriz positiva-definida. O pré-condicionamento consiste na substituição da equação (5) pelo sistema equivalente $C^{-1}Ax = C^{-1}b$, de forma que C^{-1} seja escolhido com o objetivo de se melhorar o condicionamento do sistema original. Considerando-se $\tilde{A} = C^{-1}A(C^{-1})^T$, $\tilde{x} = C^T x$ e $\tilde{b} = C^{-1}b$, obtém-se a equação $\tilde{A}\tilde{x} = \tilde{b}$.

Dado que $\tilde{r}_k = C^{-1}r_k$, $w_k = C^{-1}r_k$ e $\tilde{d}_k = C^T d_k$, as equações que geram o MGC podem ser reescritas, incorporando-se o pré-condicionamento. A equação (7), com o pré-condicionamento, fica

$$\tilde{x}_k = x_{k-1} + \tilde{\vartheta} d_k, \tag{12}$$

em que o parâmetro ϑ_k, conforme a equação (10), fica

$$\tilde{\vartheta}_k = \frac{w_{k-1}^T w_{k-1}}{d_k^T A d_k}. \tag{13}$$

A equação (9) fica

$$\tilde{r}_k = r_{k-1} - \tilde{\vartheta}_k A d_k. \tag{14}$$

A equação (8) fica

$$\tilde{d}_{k+1} = C^{-T} w_k + \tilde{\varphi}_k d_k. \tag{15}$$

O parâmetro φ_k, conforme a equação (11), fica

$$\tilde{\varphi}_k = \frac{w_k^T w_k}{w_{k-1}^T w_{k-1}}. \tag{16}$$

Com as equações (12), (13), (14), (15) e (16), tem-se o MGC pré-condicionado.

2.3.5 Cuthill-McKee reverso

Para se reduzir o custo de execução e armazenamento para se solucionar um problema pelo MGC é realizada a reordenação da matriz de coeficientes. Um algoritmo muito utilizado, por apresentar bons resultados, é o algoritmo de Cuthill e McKee (1969). George (1971) verificou que a ordem inversa do reordenamento realizado pelo algoritmo de Cuthill e McKee (1969) supera o original.

O algoritmo Cuthill-McKee reverso é um método de reordenação, que objetiva reduzir a largura de banda de uma matriz simétrica. A largura de banda de uma matriz é a maior distância do primeiro elemento não-nulo da matriz triangular inferior à diagonal principal. Segundo George, Liu e Ng (1994), dado que a largura de banda da i-ésima linha de uma matriz A, de dimensão n, é $\psi_i(A) = i - min\{(1 \leq j \leq n)|a_{ij} \in A, a_{ij} \neq 0\}$, a largura de banda ψ da matriz A, é dada por

$$\begin{aligned} \psi(A) &= \max\{\psi_i(A)|1 \leq i \leq n\} \\ &= \max\{|i-j|, |a_{i,j} \in A, a_{i,j} \neq 0\}. \end{aligned}$$

Esse algoritmo realiza o reordenamento dos vértices de modo que, na matriz gerada, os elementos não nulos irão localizar-se mais próximos da diagonal. Esse algoritmo não garante que será encontrada a menor largura de banda

possível, no entanto, na prática, bons resultados são obtidos.

Considera-se o *profile* ζ, da matriz A, tal que $\zeta(A) = \sum_{i=1}^{n} \psi_i(A)$, em que n é o número de linhas da matriz A. George (1971) verificou que o algoritmo Cuthill-McKee reverso apresenta resultados melhores em relação ao original, havendo, frequentemente, uma redução do *profile*, mas mantendo a largura de banda da matriz.

No algoritmo (7), tem-se o pseudocódigo do algoritmo Cuthill-McKee reverso, baseado na busca em largura. Esse algoritmo difere-se do algoritmo de busca em largura convencional em relação à ordem de visitação dos vértices, que se dá em ordem crescente de grau dos vértices. A saída do algoritmo é a lista dos vértices ordenados.

Cuthill e McKee (1969) verificaram que a qualidade da ordenação de seu algoritmo depende do vértice inicial. Uma opção proposta para vértice inicial é o vértice de grau mínimo, que apresenta bons resultados, embora sem garantias. Uma outra proposta é apresentada a seguir.

Vértice pseudoperiférico

Um estudo, elaborado por George e Liu (1979), propõe utilizar um vértice inicial a partir de um par de vértices que se encontra o mais distante possível entre si. Alguns conceitos são importantes para entender o funcionamento desse algoritmo.

Considere um grafo $G = (L_V, L_E)$, em que L_V é um conjunto de vértices e L_E o conjunto de arestas do grafo G. A excentricidade de um vértice $v \in L_V$, é dada pela maior distância $d(v,u)$ entre v e um vértice $u \in L_V$, tal que $u \neq v$, dado que a distância entre dois vértices é o tamanho do menor caminho entre eles. Um caminho é um conjunto ordenado de vértices distintos $(v_0, v_1, ..., v_k)$, tal que v_i é adjacente a v_{i+1}, ou seja $v_i \in adj(v_{i+1})$ para $i = 0, 1, ...k-1$. Portanto, a excentricidade τ de $v \in L_V$ é

Entrada: Vértice raiz r
Saída: Lista L de vértices ordenados

```
1  início
2      i ← 1;
3      // Insere vértice raiz r no final da fila.
4      Fila ← r;
5      // Marca vértice raiz r como visitado.
6      Visitou(r) ← Verdadeiro;
7      enquanto (Vazia(Fila) = Falso) faça
8          // Remove o primeiro vértice da fila.
9          v ← Fila;
10         // Insere o vértice v na posição i da lista de
11         // vértices ordenados.
12         L(i) ← v;
13         i ← i + 1;
14         // Ordena vértices adjacentes à v em ordem
15         // crescente de grau.
16         OrdenaAdjacentes(v);
17         para cada (vértice w adjacente à v em ordem crescente
           de grau) faça
18             se (Visitou(w) = Falso) então
19                 // Insere vértice w no final da fila.
20                 Fila ← w;
21                 // Insere o vértice w na posição i da
22                 // lista de vértices ordenados.
23                 L(i) ← w;
24                 i ← i + 1;
25                 // Marca vértice w como visitado.
26                 Visitou(w) ← Verdadeiro;
27     // Inverte a ordem dos vértices na lista L.
28     InverterOrdem(L);
29     retorna L;
```

Algoritmo 7: Cuthill Mckee reverso

dada pela maior distância, de forma que $\tau(v) = \max\{(\forall u \in L_V \wedge u \neq v) d(v,u)\}$. O diâmetro κ de G é a maior excentricidade encontrada em G, ou seja, $\kappa(G) = \max\{(\forall v \in L_V)\tau(v)\} = \max\{(\forall v,u \in L_V) d(v,u)\}$. Um vértice periférico $v \in L_V$ é aquele cuja excentricidade é igual ao diâmetro do grafo, de forma que $\tau(v) = \kappa(G)$. Já um vértice pseudoperiférico possui sua excentricidade próxima ao diâmetro do grafo.

Uma estrutura de nível de um dado nó $v \in L_V$ é o particionamento $\mathscr{L}(v)$ de L_V, satisfazendo $\mathscr{L}(v) = \{L_0(v), L_1(v), ...L_{\tau(v)}(v)\}$ em que $L_0(v) = \{v\}, L_1(v) = adj(L_0(v))$ e $L_i(v) = adj(L_{i-1}(v)) - L_{i-2}(v), i = 2,3,...\tau(v)$. Dessa forma, os vértices com excentricidade τ de v se encontram no conjunto $L_{\tau(v)}(v)$.

Com esses conceitos apresentados, é possível entender o funcionamento do algoritmo (8), que apresenta um pseudocódigo referente ao algoritmo de George e Liu (1979).

2.3.6 Resumo

Nesta seção, apresentou-se a definição do MGC bem como as suas principais características. Também foi descrito um pseudocódigo, que pode ser observado no algoritmo (6), página 37, a fim de resolver um sistema linear do tipo $Ax = b$.

Com o intuito de se otimizar o número de iterações, na subseção (2.3.4), foi apresentado como realizar o pré-condicionamento da matriz de coeficientes. Também foi exposto, na subseção (2.3.5), um algoritmo de reordenação da matriz de coeficientes, o algoritmo Cuthill-McKee reverso, conforme o algoritmo (7), na página 41. Ainda, a fim de melhorar o desempenho do algoritmo de reordenação, apresentou-se um algoritmo que busca um vértice pseudoperiférico, utilizado como vértice inicial no algoritmo de

Entrada: Vértice raiz v
Saída: Vértice pseudoperiférico t

```
1  início
2      // Construir a estrutura de nível de v.
3      L(v) = {L_0(v), L_1(v), ..., L_τ(v)(v)};
4      repita
5          // u é utilizado como controle do laço de
6          // repetição.
7          u ← v;
8          // Busca entre os vértices de maior
9          // excentricidade aquele com grau mínimo.
10         t ← Vértice com grau mínimo ∈ L_τ(v)(v);
11         // Construir a estrutura de nível de t.
12         L(t) = {L_0(t), L_1(t), ..., L_τ(t)(t)};
13         // Verifica qual tem maior excentricidade.
14         se (τ(t) > τ(v)) então
15             v ← t;
16     até ( u = v );
17     t ← v;
18     retorna t;
```

Algoritmo 8: Encontrar vértice pseudoperiférico

reordenação, que pode ser observado no algoritmo (8), página 43.

2.4 Suavização laplaciana

Uma forma de melhorar a qualidade da malha é aplicar um método de suavização, que reduz a distorção dos elementos ajustando a localização dos nós. Um método de suavização bastante conhecido é a suavização laplaciana, que move um vértice para o baricentro do polígono formado pelos seus vértices incidentes. Segundo Freitag (1997), esse método opera heuristicamente e não garante a melhoria na qualidade dos elementos. A forma simplificada da suavização laplaciana, aplicada a um nó P, pode ser escrita na forma

$$S_P^{n+1} = S_P^n + \beta \frac{\sum_{i=1}^{m} \omega_{P,i}(S_i^n - S_P^n)}{\sum_{i=1}^{m} \omega_{P,i}},$$

em que $S_P = (x,y)$ é a posição do vértice P para duas dimensões, S_P^{n+1} é a posição do nó P na etapa $n+1$ de suavização, S_i's são as posições dos nós adjacentes a P, $\omega_{P,i}$ é o peso da aresta que conecta P ao i-ésimo nó vizinho e $0 < \beta < 1$ é um parâmetro de adaptatividade que controla a amplitude do movimento, definido localmente ou globalmente.

Com o intuito de evitar as possíveis distorções de sucessivas iterações, Taubin (1995a, 1995b) propôs combinar duas suavizações sucessivas: $S_P^{n+1} = S_P^n + \lambda \Delta S_P^n$ e $S_P^{n+2} = S_P^{n+1} - \mu \Delta S_P^{n+1}$, em que $\Delta S_P^n = \frac{\sum_{i=1}^{m} \omega_{P,i}(S_i^n - S_P^n)}{\sum_{i=1}^{m} \omega_{P,i}}$ e os parâmetros quantificadores do movimento $0 < \lambda < \mu$. Taubin, Shang e Golub (1996) ainda analisam as propriedades desse método e expõem como minimizar seu tempo de execução. Essa suavização é também chamada de suavização λ/μ. Kobbelt et al. (1998) orientou utilizar $\lambda = \mu = 1$, nomeando esse método de suavização bi-laplaciana.

2.5 Malhas móveis

Nesta seção, aborda-se alguns detalhes sobre malhas móveis, como o princípio de equidistribuição e a computação em malhas móveis. Também são apresentados trabalhos que pretendem solucionar equações diferenciais parciais por diferentes métodos, como o método dos volumes finitos, dos elementos finitos e das diferenças finitas, utilizando os mais variados tipos de malhas, como, por exemplo, o diagrama de Voronoi. Também são apresentadas técnicas de movimento de vértices baseadas em formulação laplaciana e trabalhos que utilizam a suavização laplaciana para melhorar a qualidade da malha.

2.5.1 Considerações iniciais

A utilização de métodos adaptativos de malhas proporciona uma melhora significativa na precisão e eficiência na geração de malhas. Os métodos de malhas móveis possuem vantagem em relação ao refinamento adaptativo por inserção de nós pois, com o aumento da quantidade de nós há o aumento do esforço computacional necessário para solucionar a malha. Os métodos em que ocorrem inserção de nós são robustos em problemas com regiões de rápida variação de acordo com o tempo mas, tendem a se tornar ineficientes devido ao contínuo reajuste, tornando sua execução dispendiosa (HUANG; REN; RUSSELL, 1994).

Muitas pesquisas são focadas em obter soluções com baixo esforço computacional e alta precisão na aproximação, utilizando malhas adaptativas para simular fenômenos físicos. Algumas dessas pesquisas utilizam malhas móveis com discretizações de EDPs por volumes finitos, por exemplo, Dam e Zegeling (2006), Mackenzie (1996), Springel (2009, 2011), Tan

et al. (2004) e Tan, Tang e Zhang (2006). Mesmo com os recentes avanços obtidos nessas e em outras pesquisas, acredita-se que pode-se obter um esquema de malhas móveis mais simples que os apresentados na literatura, com custo computacional bastante baixo.

Na próxima seção, (2.5.2), aborda-se uma classificação dos métodos adaptativos para resolução de equações diferenciais parciais. Explica-se o princípio de equidistribuição na seção (2.5.3) e, na seção (2.5.4), a computação em malhas móveis. Apresenta-se, na seção (2.5.5), uma revisão dos métodos de malhas móveis na resolução de diversos problemas. Na seção (2.5.6), apresentam-se trabalhos que realizam melhorias na qualidade da malha por meio da suavização laplaciana e, na seção (2.5.7), tem-se trabalhos que realizam o movimento dos vértices baseado na formulação laplaciana, com o intuito de obter melhorias na aproximação da solução da EDP. Por fim, na seção (2.5.8), tem-se o resumo desta seção.

2.5.2 Adaptatividade

É comum impor alguma forma espacial na malha e, em seguida, discretizar a solução, utilizando-se elementos finitos, diferenças finitas ou volumes finitos. No entanto, tal estratégia pode não ser eficaz no caso de estruturas que envolvam escalas de pequeno comprimento, ocasionando grandes erros. Nesses casos, pode ser benéfico utilizar alguma forma não uniforme de malha, adaptada para a solução, na qual serão realizados os cálculos. As vantagens dessa estratégia podem ser a redução dos erros, melhor condicionamento do sistema e melhor eficiência computacional. Infelizmente, isso adiciona um nível extra de complexidade ao sistema, que pode conduzir a um custo computacional adicional e também a uma possível instabilidade numérica (BUDD; HUANG; RUSSELL, 2009).

Métodos adaptativos para resolução de equações diferenciais parciais podem ser divididos em três categorias: abordagem h de refinamento, abordagem p de refinamento e abordagem r de refinamento.

> Segundo Oliveira et al. (2013): Na abordagem h de refinamento, inicia-se a simulação com uma malha inicial e essa malha é refinada ou simplificada por meio da inclusão ou da remoção de pontos. Geralmente, a estratégia utilizada na inclusão ou na remoção dos pontos é orientada por uma estimativa a *posteriori* de erro da solução. Isso é chamado, geralmente, de refinamento adaptativo de malhas por usuários do método dos volumes finitos.
>
> Na abordagem p de refinamento, segundo Budd, Huang e Russell (2009 apud OLIVEIRA et al., 2013), utiliza-se uma discretização de EDPs por elementos finitos com polinômios de uma ordem particular. Essa ordem é incrementada ou decrementada de acordo com os erros da solução. Pode-se combinar essa abordagem com o refinamento h para se utilizar uma estimativa a *posteriori* de erro da solução. Com essa combinação, tem-se a subcategoria de refinamento *hp*, cujo objetivo é a obtenção da solução dentro de um erro prescrito limitado pelos procedimentos de refinamento (BUDD; HUANG; RUSSELL, 2009 apud OLIVEIRA et al., 2013).
>
> No refinamento r, o número de pontos da malha é fixo e esses pontos são movidos de forma que sejam concentrados nas regiões de grande variação da solução em função do tempo. Na comunidade de volumes finitos, são chamados, geralmente, de malhas móveis. Segundo Eleftheriou (2011), essa abordagem de refinamento pode ser, em geral, utilizada para problemas transientes por causa da mobilidade da malha, que facilita lidar com integradores de tempo. Entretanto, sua limitação está na dificuldade em definir um intervalo de tempo adequado, uma vez que os nós variam de posição ao longo do tempo, podendo ocorrer o entrelaçamento de arestas. Ainda, a aplicabilidade da adaptatividade r é limitada devido ao número fixo de graus de liberdade e a uma conectividade constante dos polígonos da ma-

> lha. Com isso, a adaptatividade r é, tipicamente, utilizada para acelerar o processo computacional em vez de ser utilizada para se alcançar uma precisão prescrita.

No contexto de malhas móveis, a análise da adaptatividade foca em como otimizar a escolha dos pontos da malha, de forma que o custo computacional seja correlacionado com o número de pontos utilizados (HUANG; RUSSELL, 2011). Os pontos da malha são concentrados em regiões em que há erro ou gradiente grande da solução. Segundo Huang e Russell (2011), a análise da adaptatividade foca em como otimizar a escolha dos pontos da malha, de forma que o custo computacional é correlacionado com o número de pontos utilizados na malha. A adaptatividade r deriva do princípio de realocação, pois a localização dos pontos é dinamicamente realocada durante o curso da computação numérica.

> Segundo Oliveira et al. (2013), como afirmam Huang e Russell (2011 apud OLIVEIRA et al., 2013), os métodos de malhas móveis ainda estão em uma fase relativamente inicial de desenvolvimento. Muitos deles estão em estágio experimental e, quase todos, requerem uma justificativa matemática adicional. Como também explicam Huang e Russell (2011 apud OLIVEIRA et al., 2013), uma análise rigorosa dos métodos de malhas móveis, para resolver EDPs dependentes do tempo, só foi realizada para alguns modelos muito simples de problemas. Ainda, também explicam que muitas formas de se melhorar sua eficiência e robustez serão, sem dúvida, desenvolvidas. Como, por exemplo, ainda são necessários mais estudos numéricos sistemáticos de como se reduzir os custos na resolução de todo um sistema de malha e EDPs, bem como estudos em como equilibrar a adaptação espacial e temporal de uma malha.

2.5.3 Princípio da equidistribuição

> Segundo Oliveira et al. (2013), o princípio de equidistribuição (PE) foi originalmente introduzido por Boor (1973). Com esse princípio, busca-se rearranjar os nós de uma malha de forma que uma determinada medida seja distribuída equitativamente ao longo de cada subintervalo da malha. Essa medida pode ser, por exemplo, uma medida de erro que será comparada com a medida de um elemento desejável, hipoteticamente ótimo. A diferença entre cada elemento da malha e o elemento desejável será, aproximadamente, a mesma para todos os elementos existentes. De acordo com Askes (2000) e Oliveira et al. (2013), algumas restrições topológicas, como cantos não convexos em problemas multidimensionais, podem impedir um movimento de pontos ideal, por isso, não se pode garantir que a equidistribuição seja satisfeita para todos os elementos da malha.

Segundo Oliveira et al. (2013), ao aplicar o PE em um problema unidimensional, redefine-se a posição dos nós, distribuindo-se, equitativamente, um medida denominada função peso, em todo o domínio. Em um problema unidimensional, por exemplo, a posição dos nós x_i, para $i = 1..N$ é realocada, distribuindo-se a função peso $M(x)$ em todo o domínio, conforme $\int_{x_{i-1}}^{x_i} M(x)dx = \int_{x_i}^{x_{i+1}} M(x)dx$, para $1 \leq i \leq N-1$. No formato discreto, essa equação é aproximada por

$$M_{i-1}\Delta x_{i-1} = M_i \Delta x_i, \text{ para } 1 \leq i \leq N-1, \tag{17}$$

em que $\Delta x_{i-1} = x_i - x_{i-1}$ é o tamanho local da malha e M_{i-1} representa a estimativa discreta de $M(x)$ no intervalo $[x_{i-1}, x_i]$. Com a distribuição uniforme ao longo de todo o domínio, tem-se que $\frac{1}{N}\int_{x_0}^{x_N} M(x)dx = \int_{x_i}^{x_{i+1}} M(x)dx = c$, para $0 \leq i \leq N-1$, em que c é uma constante.

Segundo Oliveira et al. (2013), as primeiras aplicações do PE, os tra-

balhos de Dwyer, Raiszadeh e Otey (1981), Dwyer, Sanders e Kee (1979), Gnoffo (1980) e White (1982) resolvem problemas em mecânica dos fluidos e transferência de calor em uma dimensão. White (1982 apud OLIVEIRA et al., 2013) recorreu ao comprimento do arco da solução como função monitora,

$$M(u) = \sqrt{1 + |u'|^2}. \tag{18}$$

Considere-se um exemplo unidimensional, retirado de Oliveira et al. (2013) o qual foi originalmente adaptado de Zhijun (2005), para ilustrar a ideia principal do PE. Seja $f(x) = tanh\left(\frac{1-x}{0.1}\right)$. Considere-se ainda um subconjunto no intervalo $[0,1]$. Suponha-se agora que $x_0 < x_1 < ... < x_n$, em que $x_0 = 0$ e $x_n = 1$. Neste contexto, no gráfico à esquerda da figura (8), a malha é dividida uniformemente, podendo-se observar a malha adaptativa gerada pelo PE no gráfico à direita da figura (8). Nesse caso, a função monitora foi baseada no comprimento do arco, expressa por (18) e os pontos são distribuídos igualmente na curva da solução satisfazendo (17).

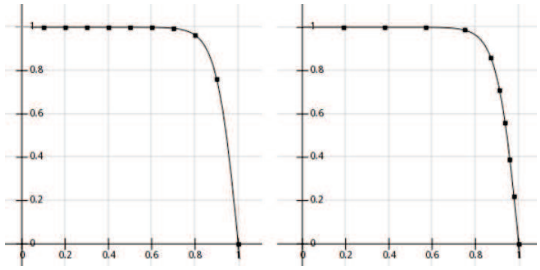

Figura 8 Comparação utilizando o PE e uma malha uniforme
Legenda: Comparação entre o PE utilizando o comprimento do arco (à direita) e uma malha dividida uniformemente (à esquerda) com dez pontos. Exemplo extraído de Oliveira et al. (2013) e originalmente adaptado de Zhijun (2005).

Exemplos similares podem ser encontrados em Li, Tang e Zhang (2001), Zegeling (1996) e Zhijun (2005). Para mais informações sobre o

PE, verifique Huang e Russell (2011), onde se apresentam os princípios básicos da adaptatividade e estratégias de movimentos de malha em casos unidimensionais e multimensionais.

2.5.4 Computação em malhas móveis

Segundo Huang e Russell (2011 apud OLIVEIRA et al., 2013), o problema de se computar soluções de EDPs, utilizando métodos de malhas móveis pode ser separado em três problemas:

- Uma função de densidade da malha, também chamada de função monitora, é necessária para guiar a redistribuição dos pontos da malha na evolução da EDP. Essa função monitora, normalmente, é restrita tanto para equidistribuir essa redistribuição, quanto para se obter um relaxamento da malha na busca de um estado equidistribuído. A escolha da função monitora pode depender do comprimento de arco da solução em problemas unidimensionais, na curvatura da solução e em erros *a posteriori* (HUANG; RUSSELL, 2011 apud OLIVEIRA et al., 2013).
- Determinada a função monitora, deve-se verificar uma malha que se equidistribui de alguma maneira. O problema da equidistribuição em si é um problema algébrico não linear (HUANG; RUSSELL, 2011 apud OLIVEIRA et al., 2013).
- A EDP é, então, discretizada, tanto no domínio computacional da malha quanto no domínio físico original e, geralmente, elementos finitos ou volumes finitos são empregados (OLIVEIRA et al., 2013).

Na prática, qualquer que seja a escolha da função monitora, alguma suavização espacial (e temporal) é empregada. Segundo Huang e Russell (2011 apud OLIVEIRA et al., 2013), a chave para o sucesso dos métodos de malhas móveis reside na escolha adequada dessa função de densidade da malha. Essa função controla a concentração de pontos na malha por meio

> do PE e, tipicamente, mensura-se a dificuldade na aproximação numérica espacial do problema fundamental. Ainda de acordo com Huang e Russell (2011 apud OLIVEIRA et al., 2013), a seleção da função de densidade da malha pode ser baseada na estimativa de erro de interpolação, na invariância de escala ou em uma estimativa de erro *a posteriori*, com o limite ótimo para o erro de interpolação ou o erro da solução, também obtido pela malha equidistribuída correspondente (OLIVEIRA et al., 2013).

Huang e Russell (2011) apresentam diversas equações de malhas para problemas estacionários e dependentes do tempo, abordando-se questões práticas de implementação, incluindo a discretização de equações de malhas e questões sobre adaptatividade de malhas no contexto multidimensional.

2.5.5 Uma revisão de métodos de malhas móveis

Já foi desenvolvida e aplicada uma grande variedade de métodos de malhas móveis, de uma e duas dimensões, na resolução de diversos problemas, conforme pode-se verificar, por exemplo, em Hawken, Gottlieb e Hansen (1991) e Tang (2005), em que há revisão de técnicas em malhas móveis de suas aplicações na dinâmica de fluidos computacional. Ainda, vários outros métodos, especialmente multi-dimensionais, têm sido desenvolvidos e utilizados com sucesso (SPRINGEL, 2009, 2011; TAN et al., 2004; TAN; TANG; ZHANG, 2006).

A seguir, são apresentadas algumas propostas de resolução de problemas utilizando malhas móveis. Também são apresentados alguns dos trabalhos desenvolvidos por Muñoz et al. (2012), Springel (2005, 2009, 2011), Tan (2007), Tan et al. (2004), Tan, Lim e Khoo (2007) e Tan, Tang e Zhang (2006).

Algumas propostas de malhas móveis

De acordo com Marlow (2010), os métodos para movimentar os pontos de uma malha podem ser classificados em duas formas: os métodos com base na localização dos pontos e os métodos com base na velocidade dos pontos. Nos métodos com base na localização, utiliza-se um método para controlar diretamente a posição dos pontos da malha e, nos métodos com base na velocidade, um método é utilizado apenas para proporcionar a velocidade da malha, encontrando-se a posição dos pontos por um esquema a cada intervalo de tempo, como o método *Forward* Euler. Dois trabalhos apresentam uma visão geral desses métodos: Hawken, Gottlieb e Hansen (1991), comentado anteriormente nesta seção, e Budd, Huang e Russell (2009). Em Marlow (2010, 2011), utilizam-se malhas móveis com elementos finitos contínuos para resolver EDPs parabólicas não lineares com fronteiras móveis por meio de um método adaptativo. Marlow (2010, 2011) descrevem os métodos utilizados em detalhes. Também são apresentados exemplos computacionais, com diferentes funções monitoras aplicadas à equação de meios porosos, em uma e duas dimensões espaciais.

Duffell e MacFadyen (2011) generalizam o método para a solução numérica de sistemas de equações hiperbólicas utilizando um diagrama de Voronoi dinâmico. O diagrama de Voronoi foi utilizado para gerar malhas móveis para a solução de sistemas multidimensionais das leis de conservação na forma de volumes finitos. Os pontos da malha são livres para se movimentar com uma velocidade arbitrária, de modo que a escolha da velocidade zero resulta na formulação euleriana. Movimentar os pontos na velocidade local do fluido torna a formulação efetivamente lagrangeana. Um código, TESS, foi escrito para resolver as equações hidrodinâmicas e magneto-hidrodinâmicas compressíveis para fluidos relativistas e não relativistas. Esse código foi modularizado, tornando-o facilmente adaptado para

solucionar sistemas de equações gerais.

Olivier e Alauzet (2011) realizaram simulações, em 3D, envolvendo geometrias móveis sujeitas, potencialmente, a grandes deslocamentos. Apresentaram dois métodos para lidar com deslocamentos grandes nas simulações com malhas móveis: o primeiro método consiste em movimentar a malha, o quanto for possível, mantendo a topologia fixa. Um inconveniente é que, ao movimentar a malha com uma topologia fixa, piora-se a forma dos elementos, o que influencia negativamente na precisão da solução e também retarda a computação, pois o intervalo de tempo é controlado pelo elemento mínimo de malha, isto é, a abordagem é sujeita à condição Courant-Friedrichs-Lewy (CFL) (COURANT; FRIEDRICHS; LEWY, 1928). O segundo método objetiva manter a qualidade da malha o melhor possível, enquanto movimenta os pontos, refazendo a malha utilizando operações locais, como adição de vértices, colapsos de vértices, alterações na conectividade e deslocamento dos vértices. Essa estratégia permite manter a qualidade da malha aceitável, entretanto, isso induz a um número grande de etapas de interpolação. Detalhes desse método podem ser encontrados em Bruchon, Digonnet e Coupez (2009), Compére et al. (2010), Dobrzynski e Frey (2008) e Löhner et al. (1999). Um solucionador HLLC (*Harten-Lax-van Leer-Contact*) (TORO; SPRUCE; SPEARES, 1994) aproximado de Riemann é utilizado em Olivier e Alauzet (2011). Também utiliza-se um método de reconstrução do tipo MUSCL para melhorar a precisão do sistema.

McNally, Lyra e Passy (2012) utilizam uma metodologia rigorosa para comparar resultados de diferentes códigos em duas dimensões para a solução do problema de instabilidade de Kelvin-Helmholtz (KHI). O KHI é uma das mais importantes instabilidades hidrodinâmicas e representa uma regra signifitiva em várias partes da astrofísica. O problema é testado nos

códigos Pencil Code [1], Athena [2], Enzo [3], NDSPMHD [4] e no código Phurbas, desenvolvido por Maron, McNally e MacLow (2012).

Silva et al. (2012) propuseram uma técnica alternativa para atualização de malhas que se submetem a alterações geométricas e topológicas. Explora-se a propriedade *Weighted Delaunay Triangulation*, que pode ser utilizada para, implicitamente, definir a conectividade da malha. Em vez de se manterem as informações de conectividade, simplesmente, mantém-se uma coleção de pesos associados a cada vértice. Essa propriedade da triangulação de Delaunay é o fato de que todos os vértices podem ser emergidos em um poliedro convexo em uma dimensão extra, também conhecida como *lifting property*. Porém, o mesmo não se aplica a malhas que não são de Delaunay. Nesse trabalho também descreve-se um mecanismo para suavização das transições entre as malhas emergidas com diferentes níveis de refinamento.

Proposta de Tan e colaboradores e trabalhos relacionados

Tan et al. (2004) mostraram que, nos métodos de malhas móveis, os intervalos de tempo são proporcionais ao tamanho do menor volume espacial da malha e, como resultado, os volumes são cada vez menores a cada intervalo de tempo. Nesse trabalho, foi desenvolvido um algoritmo de escalonamento local de tempo para métodos de malhas móveis. A ideia principal foi apresentada pela investigação das leis de conservação não lineares hiperbólicas.

Tan, Tang e Zhang (2006) propuseram um método simples de malha móvel para resolver equações de campo de fases. Sua estratégia numérica baseou-se na abordagem proposta em Li, Tang e Zhang (2001) para separar

[1] http://pencil-code.nordita.org/
[2] https://trac.princeton.edu/Athena/
[3] http://enzo-project.org/
[4] http://users.monash.edu.au/~dprice/software/index.html

a malha móvel e a evolução da EDP. As equações de campo de fases foram discretizadas por um método dos volumes finitos e o movimento da malha foi realizado resolvendo as equações euleriana-lagrangeanas com a função monitora baseada em gradiente.

Tan (2007) resolveu problemas magneto-hidrodinâmicos (MHD) por meio de técnicas de malhas móveis adaptativas, com malhas quadrangulares. Tan, Lim e Khoo (2007) resolveram um modelo de campo de fases para o fluxo da mistura de dois fluidos incompressíveis, as equações de Navier-Stokes e Allen-Cahn, com malhas adaptativas quadrangulares. Tan (2007) e Tan, Lim e Khoo (2007) utilizaram uma estratégia baseada na proposta de Li, Tang e Zhang (2001), para separar o movimento da malha e a evolução da EDP a cada intervalo de tempo. Tan (2007) obteve a solução adaptativa da malha pela resolução de um conjunto de EDPs elípticas, não lineares, para o mapa da malha.

Nos trabalhos de Tan (2007) e Tan, Lim e Khoo (2007), a aproximação da solução do sistema gerado foi obtida por meio do método iterativo Gauss-Seidel. As iterações ocorrem até que não existam alterações significativas no cálculo da nova malha, entre uma iteração e sua sucessora. Os autores afirmam que, na prática, algumas iterações foram necessárias a cada intervalo de tempo, mas o custo para gerar a malha não foi tão dispendioso. Em comparação com o método Fourier-spectral, utilizado por Liu e Shen (2003), o trabalho de Tan, Lim e Khoo (2007) necessitou de menos pontos na malha, com economia de custo computacional, para obter o mesmo resultado.

Uma escolha apropriada da função monitora gera malhas com qualidade em termos de suavização, obliquidade e relação de aspecto. Tan (2007) afirma que existem diversas escolhas possíveis da função monitora para mo-

delos aproximativos MHD. Entre as escolhas de funções monitoras, uma escolha convencional, segundo Tan, Lim e Khoo (2007), do tipo comprimento do arco, é $\omega = \sqrt{1+\alpha|\nabla\phi|^2}$. Uma outra, segundo Tan (2007), que depende da magnitude do valor e do gradiente da solução, é $\omega = \sqrt{1+\alpha|u|^2+\beta|\nabla u|^2}$. Nessas funções, α e β são considerados parâmetros de "adaptatividade", que controlam a amplitude da adaptatividade. Para valores altos de α e β, tem-se uma maior adaptatividade. Para α e β iguais a zero, tem-se $\omega = 1$, representando uma malha uniforme. Os valores de α e β são dependentes do problema, não existindo uma regra direta para a escolha desses parâmetros. Segundo Tan, Lim e Khoo (2007), em muitos casos, as funções monitoras envolvem parâmetros definidos pelo usuário que precisam ser obtidos por experimentos iniciais.

Uma função monitora aperfeiçoada envolve um parâmetro *dependente do tempo*, que é *escolhido automaticamente*. Para detalhes, verifique os trabalhos de Beckett et al. (2002), Mackenzie, Robertson e Beckett (2006) e Zegeling (2004, 2005). Huang e Russell (1999) e Huang e Sun (2003) generalizaram uma função monitora com um parâmetro dependente do tempo e com um parâmetro β que controla a proporção dos pontos em regiões críticas. Em Tan, Lim e Khoo (2007), escolheu-se $\beta = 0,5$ e, consequentemente, metade dos pontos localizou-se em regiões críticas.

Propostas de Springel e outros

De acordo com Springel (2009), atualmente, simulações hidrodinâmicas cosmológicas, geralmente, empregam a técnica hidrodinâmica de partículas suavizadas de Lagrange (*smoothed particle hydrodynamics* - SPH) ou a técnica hidrodinâmica de Euler, em uma malha cartesiana com refinamento opcional da malha adaptativa (*adaptive mesh refinement* - AMR). Ambos os métodos têm desvantagens que impactam negativamente na sua

exatidão em determinadas situações, por exemplo, a supressão da instabilidade de fluidos no caso SPH, a falta de invariância de Galileu e a presença de *overmixing* no caso do AMR. Springel (2009) propôs um novo esquema que, em grande parte, eliminou esses pontos fracos. Baseou-se em malhas móveis, irregulares, definidas pelo diagrama de Voronoi de um conjunto de pontos discretos.

Em Springel (2009), a malha é utilizada para resolver as leis de conservação hiperbólicas de hidrodinâmica ideal, por volumes finitos, com base em um esquema de Godunov de segunda ordem, não divisível, com um solucionador exato de Riemann.

De acordo com Springel (2011), é possível obter um comportamento lagrangeano em métodos baseados em malhas, se for permitida que a malha se mova de acordo com o fluxo. No entanto, tal abordagem tem sido, muitas vezes, repleta de problemas substanciais, relacionados ao desenvolvimento de irregularidades na topologia da malha. Springel (2011) descreve um esquema que elimina esses problemas. Esse esquema baseia-se em malhas móveis, irregulares, definidas pelo diagrama de Voronoi de um conjunto de pontos discretos. Em Springel (2011), resolveu-se o mesmo problema de Springel (2009) utilizando-se esse esquema proposto.

Pakmor, Bauer e Springel (2011) discutem a implementação de um problema magneto-hidrodinâmico (MHD) ideal em uma malha móvel, utilizando o código de volumes finitos hidrodinâmicos AREPO, desenvolvido por Springel (2009), que combina algumas das vantagens dos métodos eulerianos e lagrangeanos em uma técnica computacional simples. O código AREPO é um método de volumes finitos, com precisão de segunda ordem, que resolve as equações de Euler baseadas na reconstrução linear por partes e no cálculo do fluxo hidrodinâmico em toda face de célula com um solu-

cionador exato de Riemann. A malha computacional é construída como um diagrama de Voronoi de um conjunto de pontos geradores de malha. O esquema AREPO combina a precisão de malhas baseadas na hidrodinâmica com a adaptatividade natural e a invariância translacional, fornecido, geralmente, por técnicas SPH.

Greif et al. (2011) utilizaram uma série de simulações de alta resolução hidrodinâmica, realizadas com malhas móveis, semilagrangeanas, utilizando o código AREPO, com o intuito de estudar a influência dos parâmetros ambientais no nível de fragmentação. Nesse trabalho, os autores estudaram o colapso de gases em miniauréolas refinadas.

Heß e Springel (2012) realizaram testes, analisando a evolução de galáxias, com os esquemas de partículas suavizadas hidrodinâmicas de Lagrange (SPH) (SPRINGEL, 2005), partículas de Voronoi hidrodinâmicas (VPH) (HEß; SPRINGEL, 2010) e o código AREPO de malhas móveis hidrodinâmicas (SPRINGEL, 2009). Como resultado, o método VPH leva a um *stripping* do gás da galáxia mais rápido do que o método SPH e, em melhor acordo com o código da malha que o método SPH. Mostra-se que, apesar do fato de que o método VPH não é tão preciso quanto o código de malhas móveis, nos casos investigados, a maior precisão das estimativas de inclinação pode tornar o método VPH como uma alternativa atraente ao método SPH.

Muñoz et al. (2012) descreveram uma nova formulação de viscosidade hidrodinâmica contínua, que resolve as equações do movimento sobre uma malha de Voronoi, criada por um conjunto de pontos geradores da malha. Os pontos podem se mover de uma forma arbitrária, mas o movimento mais natural é dado, por si só, pela velocidade do fluido de tal forma que a malha se ajusta dinamicamente em função do fluxo. Essa implementação

considera as equações totais de Navier-Stokes em 2D e em 3D. Muñoz et al. (2012) propuseram uma abordagem para calcular os fluxos viscosos precisos para uma malha de Voronoi dinâmica e para formular um solucionador por volumes finitos das equações de Navier-Stokes.

2.5.6 Trabalhos que utilizam a suavização laplaciana

Nesta subseção, apresentam-se alguns trabalhos que utilizam a suavização laplaciana, a fim de obter melhorias na qualidade da malha.

Ohtake, Belyaev e Bogaevski (2001) apresentam um conjunto de ferramentas de suavização de malhas triangulares baseadas no laplaciano. Realizam comparações gráficas de um método proposto, baseado no fluxo médio da curvatura (*mean curvature flow*), com a suavização laplaciana simplificada, com a suavização de Taubin (1995a, 1995b), com a suavização bi-laplaciana, conforme Kobbelt et al. (1998), que trata-se da anterior com $\lambda = \mu = 1$, e com a suavização baseada no fluxo médio da curvatura (*mean curvature flow*), em que $\Delta S_P^n = H\mathbf{n}(S_P^n)$, tal que H é a versão discreta da curvatura média e $\mathbf{n}$ é o vetor normal unitário. Uma aproximação discreta H da curvatura média, proposta por Desbrun et al. (1999), é tal que $H\mathbf{n}(S_P) = \frac{1}{4A}\sum_{i=1}^{m}(\cot\hat{a}_i + \cot\hat{b}_i)(S_i - S_P)$, em que A é a soma das áreas dos triângulos em volta de P e $\hat{a}_i$ e $\hat{b}_i$ são os ângulos opostos à aresta que liga os vértices com posição S_i e S_P, conforme pode-se observar na figura (9). Na suavização proposta, $\Delta S_P^n = H\mathbf{n}(S_P^n) + C\{\mathscr{U}(S_P^n) - [\mathscr{U}(S_P^n)\cdot\mathbf{n}(S_P^n)]\mathbf{n}(S_P^n)\}$, em que C é uma constante e $\mathscr{U} = \frac{1}{m}\sum_{i=1}^{m} S_i^n - S_P^n$, verificou-se que a malha possui a mesma qualidade utilizando a suavização baseada no fluxo médio da curvatura, mas com uma distribuição uniforme dos vértices.

Em seu trabalho, Belyaev e Ohtake (2003) apresentam uma suavização de malha baseada na difusão linear das retas normais da malha.

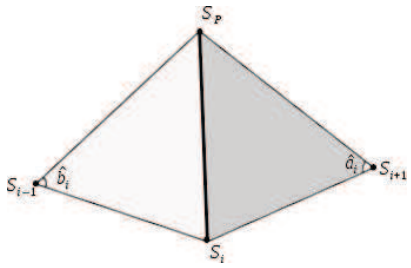

Figura 9 Variáveis da curvatura média
Legenda: Variáveis S_p, S_{i-1}, S_i, S_{i+1} da fórmula da curvatura média.

Apresentam-se resultados comparativos do método proposto, chamado filtragem média iterativa nas retas normais (*iterative mean filtering on mesh normals*), com a suavização laplaciana simplificada, com a suavização baseada no fluxo médio da curvatura, conforme Desbrun et al. (1999), com a suavização de Taubin (1995a, 1995b) e com a suavização bi-laplaciana, conforme Kobbelt et al. (1998). Comparou-se com esses métodos, pois, segundo Belyaev e Ohtake (2003), esses métodos provém ao usuário uma combinação de simplicidade, eficiência e velocidade e, consequentemente, são amplamente utilizados em diversas aplicações de modelagem geométrica. O método proposto foi o mais lento porém, apresentou bons resultados em relação a características nítidas de formas com ruídos (*denoising shapes*).

Freitag (1997) combinou a suavização laplaciana com uma suavização baseada em otimização. Nesse trabalho, definiu-se uma suavização chamada "suavização laplaciana inteligente" (*smart laplacian smoothing*), em que realoca-se os vértices da malha apenas se houve alguma melhora da qualidade local da malha de acordo com alguma métrica de qualidade. Nas técnicas de suavização baseadas em otimização, para encontrar a nova posição do vértice, objetiva-se maximizar a função composta $F(S_P) = \min G_i(S_P)$, para $0 < i < n$, em que n é o número de vértices e a função G_i é uma métrica

de qualidade. Realizam-se experimentos numéricos para comparar métricas de qualidade e custo computacional dos métodos propostos em duas e três dimensões.

Vartziotis, Wipper e Papadrakakis (2013) realizam comparações de um método proposto com a suavização laplaciana em que $\omega_i = A_i$, em que A_i é a área do triângulo i, conforme Jones (1974); com a suavização laplaciana inteligente, conforme Freitag (1997); e com um método de otimização global, conforme Brewer et al. (2003), Diachin et al. (2006) e Zhang, Bajaj e Xu (2005). O novo método é chamado de método de transformações de elemento geométrico (*geometric element transformation method* - GETMe), e baseia-se na técnica de regularização de transformações de elemento, conforme Diachin et al. (2006) e Knupp (2001). Nessa proposta, realiza-se o movimento avaliando duas métricas de qualidade de malha q_{min} e q_{mean}, em que $q_{min} \leftarrow \min q(E_i)$ e $q_{mean} \leftarrow \frac{1}{m}\sum_{i=1}^{m} q(E_i)$, $i = [1,m]$, que representam, respectivamente, o elemento de qualidade mínima e a média da qualidade dos elementos, em que E_i é um elemento da malha,. As novas posições são definidas como médias ponderadas, conforme $S_P^{n+1} = \frac{\sum_{i=1}^{m} \omega_i (S_i^{n+1} - S_P^{n+1})}{\sum_{i=1}^{m} \omega_i}$, de forma que $\omega_i \leftarrow [1 - q(E_i)]^k$, em que $k \geq 0$ é um parâmetro fixo de amplificação. Trata-se de um método de suavização baseado em otimização. Essa técnica é aplicada de forma iterativa até que as melhorias estejam abaixo de um determinado limiar.

Outros trabalhos que utilizam movimento de vértices baseado no laplaciano, alguns apresentando comparações com o bi-laplaciano, conforme Kobbelt et al. (1998), e o método de Taubin (1995a, 1995b), podem ser verificados em Freitag, Jones e Plassmann (1997), Kim e Rossignac (2004), Lange e Polthier (2005), Ohtake, Belyaev e Bogaevski (2000), Schneider, Kobbelt e Seidel (2001), Soni et al. (2000) e Vollmer, Mencl e Müller (1999).

2.5.7 Trabalhos que realizam movimento de vértices baseado na formulação laplaciana

Nesta subseção, apresentam-se alguns trabalhos que realizam o movimento de vértices, por meio da adaptatividade-r, baseado na formulação laplaciana. Nesses trabalhos, movimentam-se os vértices a fim de se obter melhorias na aproximação da solução da EDP.

Dentre as diversas variações desenvolvidas, segundo Thompson, Soni e Weatherhill (1999), existe uma forma típica, conforme a equação $S_P^{n+1} = S_P^n + \beta \frac{\sum_{i=1}^{m} \omega_{P,i}(S_i^n - S_P^n)}{\sum_{i=1}^{m} \omega_{P,i}}$, em que se calcula $\omega_{P,i}$ conforme $\omega_{P,i} = \iota \left| \frac{\phi_i - \phi_P}{\phi_i + \phi_P} \right|$, em que ϕ é o parâmetro adaptativo e $0 < \iota < 1$ é uma constante de intensidade de ω.

Littlefield (2001) descreve um método explícito de elementos finitos de adaptatividade r, para aplicações de alto impacto e penetração. O esquema de adaptatividade r implementado nesse trabalho realiza o movimento baseado na suavização laplaciana e é semelhante à aplicação ALE, de forma que o movimento dos vértices é restrido aos limites físicos do material. A reconstrução da malha é realizada de forma local.

Shontz e Vavasis (2004) realizam um estudo de movimento dos vértices em que se objetiva determinar, a partir da malha inicial, um conjunto de pesos locais para cada nó interior em função de seus vizinhos. Esses pesos são calculados usando uma matriz de rigidez dos elementos finitos. Nesse trabalho, utiliza-se um método chamado suavização laplaciana ponderada linear (*Linear Weighted Laplacian Smoothing - LWLS*).

Rajagopal, Gangadharan e Sivakumar (2006) realizam uma avaliação de algoritmos de adaptatividade r com base no método de forças configuracionais e analogia a molas. A avaliação é feita com base em aspectos qualitativos e quantitativos de estimativas de erro. O método proposto de

adaptação da malha é baseado no método de força de configuracional em conjunto com o movimento baseado na suavização laplaciana ponderada e melhoria da malha através do refinamento h, baseado no erro de discretização. Os estudo numéricos confirmam que a proposta de adaptação rh é mais eficiente do que uma abordagem puramente h e mais flexível do que uma abordagem puramente r, com melhores características de convergência.

Um sistema adaptativo rh, proposto por Rajagopal e Sivakumar (2007), foi formulado para analisar problemas de interface bi-materiais, utilizando o método dos elementos finitos. Trata-se de um combinação da força configuracional de adaptação r, com movimento baseado na suavização laplaciana ponderada e melhoria da malha por refinamento h.

Popiolek e Awruch (2009) apresentam uma estratégia de malha adaptativa para simulação numérica de fluxos transientes incompressíveis com transferência de calor e transporte de massa. Emprega-se o método dos elementos finitos com malhas não estruturadas formadas por tetraedros. A fim de obter resultados precisos, utilizam-se indicadores de erro, um esquema de refinamento e um processo de movimento baseado na suavização laplaciana inteligente (*smart laplacian smoothing*), conforme Freitag (1997).

2.5.8 Resumo

Nesta seção, abordou-se malhas móveis. Na subseção (2.5.2), diferentes tipos de adaptatividade de malhas foram apresentados, com ênfase nos métodos de malhas móveis. Também foi abordado o princípio de equidistribuição, na subseção (2.5.3), com apresentação de um exemplo unidimensional.

Comentou-se sobre o problema de computar soluções de EDPs, na

subseção (2.5.4), utilizando malhas móveis. Ainda, na subseção (2.5.5), foram citados alguns trabalhos sobre malhas móveis desenvolvidos recentemente. Na subseção (2.5.6), apresentaram-se trabalhos que realizam melhorias na qualidade da malha por meio da suavização laplaciana e, na seção (2.5.7), trabalhos que realizam o movimento dos vértices baseado na formulação laplaciana, com o intuito de obter melhorias na aproximação da solução da EDP.

2.6 Métrica geométrica da qualidade da malha

Bank e Smith (1997) propõem um algoritmo de suavização e utilizam uma métrica geométrica. Considere δ o triângulo da figura (10), com vértices v_1, v_2 e v_3 e os vetores $l_1 = \begin{bmatrix} x_3 - x_2 \\ y_3 - y_2 \end{bmatrix}$, $l_2 = \begin{bmatrix} x_1 - x_3 \\ y_1 - y_3 \end{bmatrix}$ e $l_3 = \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \end{bmatrix}$, orientados em sentido anti-horário. A medida geométrica, utilizada por Bank e Smith (1997), chamada *Shape Regularity Quality* (SRQ), denotada por $\upsilon(\delta)$, é dada por $\upsilon(\delta) = \frac{4\sqrt{3}|\delta|}{|l_1|^2+|l_2|^2+|l_3|^2}$, em que $2|\delta| = (x_2 - x_3)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1)$, com os vértices orientados em sentido anti-horário. Caso estejam orientados em sentido horário, o valor de $|\delta|$ será negativo. Trata-se da razão entre a área do triângulo e a soma dos quadrados dos comprimentos das arestas. A constante $4\sqrt{3}$ é utilizada como fator de normalização, para que os valores fiquem entre $0 < \upsilon(\delta) < 1$. Para um triângulo equilátero, $\upsilon(\delta) = 1$, e para triângulos com ângulos muito pequenos será próximo a zero. Para entender o significado geométrico dessa medida, verifique Bank e Smith (1997). Dada uma malha M, a medida SRQ dessa malha é dada pelo menor valor dessa métrica, denominado $\upsilon(\delta)_{min}$.

Segundo Bank e Smith (1997), o único triângulo para o qual $\upsilon(\delta) = 1$

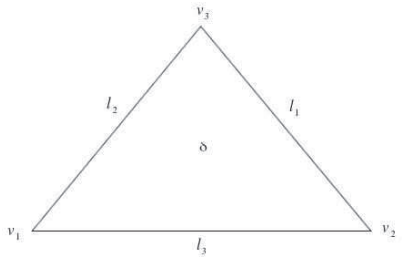

Figura 10 Rótulos e orientação do triângulo
Legenda: Rótulos v_1, v_2, v_3, l_1, l_2 e l_3 e orientação do triângulo $\boldsymbol{\delta}$.

é o triângulo equilátero. Para $\frac{\sqrt{3}}{2} \leq \upsilon(\boldsymbol{\delta}) \leq 1$ tem-se triângulos sem ocorrência de ângulos obtusos. Conforme $\upsilon(\boldsymbol{\delta})$ é reduzido, o triângulo $\boldsymbol{\delta}$ se torna mais degenerado.

3 DESCRIÇÃO DO PROJETO COMPUTACIONAL PARA A SOLUÇÃO DA EQUAÇÃO DE CONDUÇÃO DO CALOR

Neste capítulo, descreve-se o projeto computacional para a solução de equações diferenciais parciais (EDPs) por discretizações por volumes finitos com refinamento de Delaunay e malhas móveis. Na seção (3.1), trata-se de apresentar a biblioteca de aritmética de precisão arbitrária e os motivos que levaram à sua escolha. Na seção (3.2), explica-se como é criada a triangulação inicial. Na seção (3.3), descreve-se o procedimento de refinamento adaptativo, detalhando como é feita a seleção dos triângulos localizados nas regiões de grande variação. Apresenta-se um pseudocódigo desses procedimentos. Na seção (3.4), descreve-se como é realizada a resolução da malha inicial. Na seção (3.5), expõe-se o procedimento que realiza o movimento dos vértices entre as variações temporais da EDP, a função monitora escolhida e o critério de parada. Também apresenta-se um pseudocódigo desse procedimento. Todo o código foi implementado utilizando a linguagem de programação C++.

3.1 Aritmética de precisão arbitrária

Preliminarmente, com o intuito de reduzir erros numéricos, verificou-se que seria necessário utilizar uma biblioteca para aritmética de precisão arbitrária. Dentre as diversas opções disponíveis, há a *GNU Multiple-Precision Library*, também conhecida como GMP, que é uma biblioteca de código aberto para aritmética de precisão arbitrária, trabalhando sobre inteiros, racionais e números de ponto flutuante. Essa biblioteca permite que variáveis tenham um tamanho de bytes variável. Detalhes sobre a GMP podem ser verificados em Granlund (2012).

Outra opção, baseada na GMP, é a *Multiple-Precision Floating-point computations with correct Rounding* (MPFR). Essa biblioteca possui um conjunto maior de funções em comparação com a GMP, em especial funções transcendentais como exponenciais, logaritmos e funções trigonométricas, e valores especiais, como zeros assinados, infinitos e indicando o que não é um número - NaN. Além disso, a MPFR fornece um argumento adicional para as suas funções, em que se define o arredondamento utilizado, conforme especificado pelo padrão IEEE 754-200 [5].

Neste trabalho, o parâmetro utilizado em todas as operações especifica o arredondamento "*round to nearest*", em que, ao realizar o arredondamento, o *bit* menos significativo é definido para zero. Conforme Fousse et al. (2007), utilizando a MPFR é garantido que o resultado de qualquer operação seja o valor de ponto flutuante mais próximo possível do resultado exato. Maiores informações sobre a MPFR podem ser encontradas em Fousse et al. (2007). Ainda, optou-se por utilizar uma interface da MPFR para C++, chamada *mpfr::real* [6], publicada sob os termos da GNU GPL v3 [7], que minimiza as alterações necessárias no código para utilizar a biblioteca para aritmética de precisão arbitrária MPFR.

3.2 Criação da malha inicial

Ao executar o código, inicialmente, lê-se um arquivo de entrada contendo os vértices iniciais da triangulação. O domínio é dividido em dois triângulos, formados pelos vértices situados na fronteira do domínio. A malha inicial, formada pelos vértices do arquivo de entrada, é criada utilizando o algoritmo de Green e Sibson (1978), conforme o algoritmo (2) apresen-

[5] Disponível em http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4610935
[6] Disponível em: http:://chschneider.eu/programming/mpfr_real/
[7] Disponível em: http://www.gnu.org/licenses/

tado na seção (2.1.3), na página 20. Neste trabalho, para geração da malha inicial, utilizou-se o projeto computacional criado para as simulações publicadas em Oliveira e Oliveira (2012b).

A malha inicial gerada conterá triângulos de Delaunay em todo o domínio que envolve o *Planar Straight Line Graph* (PSLG). Este algoritmo recebe como entrada um PSLG, permitindo arestas pendentes e pontos isolados, tendo apenas a restrição de que todos os ângulos presentes no PSLG sejam, de pelo menos, 60°. Dessa forma, tem-se a triangulação inicial, formada pelos vértices do arquivo de entrada. Com a malha inicial gerada, pode-se realizar o refinamento adaptativo.

3.3 Refinamento de Delaunay

Executam-se os algoritmos de Ruppert (1995) e, em seguida, o de Üngör (2004, 2009), cujos códigos-fonte foram criados pelo projeto computacional desenvolvido por Oliveira e Oliveira (2012a). Neste, trabalho realiza-se o refinamento de Delaunay seguindo dois critérios, que são explicados a seguir, até que o número mínimo de vértices, denotado por χ, seja atingido. Apresentou-se o algoritmo de Üngör (2004, 2009) na subseção (2.1.4), representado pelo algoritmo (4), na página 25. O número mínimo de vértices para a fase de refinamento é definido na subseção (4.1), página 83.

O primeiro critério, denotado por σ, definido automaticamente durante a execução, seleciona triângulos localizados em regiões de grande variação. Para isso, utiliza-se o gradiente, chamado aqui de γ. Para calcular γ, considere uma aresta a de um determinado triângulo, formada pelos vértices v_1 e v_2. Calcula-se o módulo da diferença do valor ϕ da EDP nesses vértices, dividido pela distância que os separa. Com isso, tem-se que $\gamma_a = \frac{|\phi_{v_1} - \phi_{v_2}|}{d_{(v_1,v_2)}}$,

em que ϕ_{v_1} é o valor da EDP no vértice v_1, ϕ_{v_2} é o valor da EDP no vértice v_2 e $d_{(v_1,v_2)}$ é a distância que separa os vértices v_1 e v_2.

No passo de refinamento adaptativo, em que se verifica se a quantidade mínima de vértices é alcançada, o valor do critério σ é aumentado, de modo a selecionar triângulos localizados em regiões de maior variação em relação à iteração anterior. Ao se aumentar σ, seleciona-se menos arestas para o refinamento, mas somente arestas nas regiões de maior variação da solução. Para se definir o aumento de σ, calcula-se o maior e menor valor de γ na malha, γ_{max} e γ_{min} respectivamente, e obtém-se a diferença entre esses valores. Essa diferença será multiplicada por uma constante $0 < \theta < 1$ de forma que, quanto menor o valor de θ, mais triângulos serão selecionados por esse critério. Para os testes realizados, define-se o valor de θ na pág. 80, do capítulo (4). Dessa forma, a cada iteração do passo de refinamento adaptativo, atualiza-se σ, com seu valor inicial igual a zero, de forma que $\sigma \leftarrow \sigma + \theta(\gamma_{max} - \gamma_{min})$. Com isso, triângulos que foram refinados em uma iteração não serão selecionados na iteração posterior, pois o valor de γ aumentou. Triângulos que contêm, pelo menos, uma aresta com $\gamma > \sigma$ serão refinados utilizando o algoritmo de Ruppert (1995), cujo código-fonte foi desenvolvido pela implementação em Oliveira e Oliveira (2012b).

O segundo critério refere-se ao ângulo mínimo α, definido pelo usuário. Triângulos com ângulo mínimo menor que α serão refinados utilizando o algoritmo de Üngör (2004, 2009), cujo código-fonte foi desenvolvido pela implementação de Oliveira e Oliveira (2012a). Em vez de se verificar o ângulo mínimo de cada triângulo, considera-se a métrica *Circunradius-to-shortest Edge Radio* (CER), explicada na subseção (2.1.4), na página 21, denotada por ρ. Como ρ é inversamente proporcional ao ângulo, para uma boa qualidade geométrica, ρ deve ser menor que ρ_α, em que ρ_α é o valor da

medida CER do ângulo α. Com isso, refinam-se os triângulos com $\rho > \rho_\alpha$. Consequentemente, o maior valor da medida CER encontrado na malha, denotado por ρ_{max}, será tal que $\rho_{max} < \rho_\alpha$.

Dessa forma, tem-se o refinamento dos triângulos localizados nas regiões de grande variação, com uma qualidade mínima garantida pelo algoritmo de Üngör (2004, 2009).

3.4 Resolução da equação de condução do calor por discretizações de volumes finitos na malha inicial

Após realizar o refinamento adaptativo, executa-se o método dos volumes finitos (MVF), representado pelo algoritmo (5) da subseção (2.2.3), página 33, a fim de gerar o sistema linear. Para solucionar a equação da condução de calor adaptou-se o código-fonte desenvolvido por Oliveira e Oliveira (2012a) a fim de obter a solução da equação de Laplace por discretizações de volumes finitos. Considera-se que o valor da EDP, em todos os vértices internos, na iteração inicial no tempo, é igual a zero.

Realiza-se o reordenamento da lista de vértices utilizando o algoritmo Cuthill-McKee reverso (CMr), conforme o algoritmo (7), introduzido na subseção (2.3.5), página 41, com o vértice pseudoperiférico v como vértice inicial, conforme o algoritmo (8), apresentado na mesma subseção, na página 43, cujo código-fonte foi desenvolvido na implementação de Chagas e Oliveira (2013).

Executa-se o método dos gradientes conjugados (MGC) com precisão numérica ε, cujo código-fonte foi desenvolvido na implementação de Oliveira e Oliveira (2012a), tal qual o algoritmo (6), introduzido na subseção (2.3.3), página 37, a fim de atualizar os valores da EDP em cada vértice, obtendo melhores resultados após o refinamento adaptativo.

O processo para criação da malha inicial e refinamento adaptativo pode ser observado no algoritmo (9). Os parâmetros de entrada do algoritmo (9) são mostrados na tabela (1). O algoritmo 7 é apresentado conforme desenvolvido em Chagas e Oliveira (2013). A próxima etapa trata do movimento dos vértices, na subseção (3.5).

3.5 Movimento dos vértices

O movimento dos vértices ocorre entre cada etapa de variação do tempo, referente à discretização da equação do calor. Entre cada etapa de variação do tempo optou-se por realizar o movimento dos vértices enquanto uma condição de qualidade geométrica da malha for satisfeita. Quando essa condição não for mais satisfeita, executa-se o algoritmo de Üngör (2004, 2009), pela implementação desenvolvida por Oliveira e Oliveira (2012a), com o objetivo de melhorar a qualidade da malha. Essa métrica de qualidade geométrica será explicada mais adiante.

Definiu-se uma nova função monitora, chamada de Λ, baseada no laplaciano. Leva-se em consideração a maior diferença do valor da EDP em toda a malha entre vértices adjacentes. Para um vértice P, a função monitora Λ fica $S'_P \leftarrow S_P + \frac{\beta}{\Delta\phi_{max}} \sum_{i=1}^{n} (S_i - S_P)(|\phi_i - \phi_P|)$, em que S_P é a posição do vértice P, $0 < \beta < 1$ é um parâmentro para controle do movimento, n é o número de vértices adjacentes a P, ϕ_i é o valor da EDP no vértice i, adjacente à P, ϕ_P é o valor da EDP no vértice P, $\Delta\phi_{max}$ é o maior valor de $|\phi_i - \phi_P|$, para toda aresta pertencente à malha e S'_P é o valor da nova posição do vértice P. Essa função realiza movimentos mais rápidos nas regiões de maior variação, onde o $|\phi_i - \phi_P|$ é próximo ao valor de $\Delta\phi_{max}$.

A função monitora Λ é comparada com outras quatro funções, todas conforme $S'_P \leftarrow S_P + \beta \frac{\sum_{i=1}^{n} (S_i - S_P)\omega_{P,i}}{\sum_{i=1}^{n} \omega_{P,i}}$, apresentadas na seção (2.4), página 44.

Entrada: Grafo $G = (L_V, L_E)$, em que L_V é a lista dos vértices e L_A é a lista de arestas da triangulação inicial, α, θ, ε, Δt e χ. // Verifique tabela (1).

Saída: Solução inicial. // Malha com ângulos entre α e // $\pi - 2\alpha$, refinada nas regiões de grande variação.

1 **início**
2 $\sigma \leftarrow 0$;
3 // Gera malha M, a partir de L_V.
4 $M \leftarrow AlgoritmoGreenSibson(L_V)$;
5 // Nas simulações, χ é definido na subseção
6 // (4.1), página 83.
7 **enquanto** $(M.numeroVertices < \chi)$ **faça**
8 // γ_{max} e γ_{min} são o maior e menor valor de γ da
9 // malha, em que γ é o gradiente.
10 $\gamma_{max} \leftarrow calcula\gamma_{max}(M)$;
11 $\gamma_{min} \leftarrow calcula\gamma_{min}(M)$;
12 // σ é o critério de seleção de triângulos
13 // para refinamento.
14 $\sigma \leftarrow \sigma + \theta(\gamma_{max} - \gamma_{min})$;
15 // Refina triangulos com $\gamma > \sigma$.
16 $RefinaPorAlgoritmoRuppert(\sigma, M)$;
17 // Refina triângulos com ângulos menores que
18 // α.
19 $RefinaPorAlgoritmoUngor(\alpha, M)$;
20 // Gera sistema linear pelo MVF.
21 $MetodoVolumesFinitos(M, t \times \Delta t)$;
22 // Executa reordenação dos vértices.
23 $v \leftarrow AlgoritmoVerticePseudoPeriferico(M)$;
24 $AlgoritmoCuthillMcKeeReverso(M, v)$;
25 // Atualiza valores da EDP pelo MGC.
26 $MetodoGradientesConjugados(M, \varepsilon)$;

Algoritmo 9: Método de geração e refinamento da malha.

Tabela 1 Descrição das variáveis

Variável	Algoritmo	Descrição
L	(9)	Lista de vértices e arestas da triangulação inicial de Delaunay.
θ	(9)	Constante quantificadora do critério de seleção para refinamento.
χ	(9)	Número mínimo de vértices na malha.
M	(10)	Malha inicial gerada no algoritmo (9).
β	(10)	Parâmetro que controla a amplitude do movimento dos vértices.
λ,μ	(10)	Parâmetros de relaxamento específicos da função de Taubin (1995a, 1995b).
η	(10)	Valor de tolerância da métrica SRQ para continuar movimentando os vértices.
t_{final}	(10)	Número máximo de iterações temporais.
Δt	(9) e (10)	Variação do tempo utilizado na discretização da equação do calor pelo MVF.
α	(9) e (10)	Ângulo mínimo utilizado para realizar refinamento.
ε	(9) e (10)	Precisão numérica do MGC.

Legenda: Descrição dos parâmetros de entrada dos algoritmos (9) e (10). Todas as variáveis são definidas pelo usuário via arquivo de configuração.

As funções, escolhidas por serem consideradas clássicas na literatura, são as seguintes:

1. Comparação com a função monitora baseada diretamente na formulação laplaciana, em que $\omega_{P,i} = |\phi_i - \phi_P|$. Essa função é aqui chamada de Γ;

2. Comparação com uma função proposta por Taubin (1995a, 1995b), em que $\omega_{P,i} = |\phi_i - \phi_P|$, combinada por dois movimentos sucessivos, conforme tal que $S'_P = S_P + \lambda \Delta S_P$ e $S''_P = S'_P - \mu \Delta S'_P$, em que $\Delta S_P = \frac{\sum_{i=1}^{m} \omega_{P,i}(S_i^n - S_P^n)}{\sum_{i=1}^{m} \omega_{P,i}}$ e os parâmetros de relaxamento $0 < \lambda < \mu$. Essa função é

aqui chamada de Ξ, mas é conhecida como função $\lambda - \mu$. Esse esquema de duas combinações sucessivas foi utilizado no âmbito da suavização laplaciana em Taubin (1995a, 1995b). Na subseção (2.5.6), página 60, apresentam-se alguns trabalhos que utilizam essa função para melhoria da qualidade da malha. Não foram encontrados trabalhos em que se utilizou essa função como função monitora;

3. Comparação com a função monitora proposta por Taubin (1995a, 1995b), entretanto, configurada com $\lambda = \mu$, conforme proposto por Kobbelt et al. (1998), conhecida como bi-laplaciana, chamada aqui de Ψ. Na subseção (2.5.6), página 60, apresentam-se trabalhos que utilizam essa função. Não foram encontrados trabalhos em que se utilizou essa função como função monitora;

4. Comparação com a função definida por Thompson, Soni e Weatherhill (1999), em que o peso $\omega_{P,i}$ é tal que $\omega_{P,i} = \iota \left(\frac{|\phi_i - \phi_P|}{|\phi_i + \phi_P|} \right)$, para $0 < \iota < 1$. Essa função é aqui chamada de Υ.

Os valores de β, λ, μ e ι, utilizados nas funções monitoras, são definidos no capítulo (4).

Escolheram-se essas funções monitoras, pois são funções consideradas eficientes, graças à sua velocidade e simplicidade. São funções já concretizadas na literatura, clássicas, sendo utilizadas em diversos trabalhos comparativos, conforme apresentados nas subseções (2.5.6) e (2.5.7). Além disso, quanto às funções de Kobbelt et al. (1998) e Taubin (1995a, 1995b), chamadas aqui respectivamente de Ξ e Ψ, não foram encontrados trabalhos em que essas funções foram utilizadas a fim de obter melhorias na aproximação da solução da EDP. Optou-se também pela função de Thompson, Soni e Weatherhill (1999), chamada aqui de Υ, por apresentar bons resultados

nos testes realizados e, além disso, segundo o autor, trata-se de uma típica abordagem.

Avaliou-se a possibilidade de movimentar os vértices, repetidamente, até que não houvesse mais movimento, em seguida, executar o algoritmo de Üngör (2004, 2009) e avançar a etapa de tempo. Entretanto, devido à alta precisão da biblioteca MPFR, experimentos mostraram que o movimento, ainda que pequeno, sempre ocorre. Ao se realizar testes em uma malha com 100 vértices, movimentando-a, incondicionalmente, ao atingir 400 iterações de movimento dos vértices, os valores das métricas de qualidade não variavam mais. No entanto, o movimento dos vértices ainda existia.

O procedimento de movimento dos vértices repete-se enquanto a malha tiver uma qualidade geométrica mínima. A métrica *Shape Regularity Quality* (SRQ), descrita na subseção (2.6), na página 65, denotada por υ, é verificada. O valor de υ varia de zero a um e, quanto mais próximo de um, mais se assemelha a um triângulo equilátero. Então, movimenta-se a malha enquanto esta ainda tiver um valor satisfatório de υ. Dessa forma, define-se um valor $0 < \eta < 1$ de forma que, o procedimento de movimento dos vértices é repetido enquanto $\upsilon_{min} > \eta$, em que υ_{min} é o menor valor da medida SRQ encontrado na malha. O valor de tolerância η da métrica SRQ para continuar movimentando os vértices é definido na página 80, do capítulo (4).

Com o término do procedimento de movimento dos vértices, executa-se o algoritmo de Üngör (2004, 2009) e avança-se uma iteração no tempo. O algoritmo de Üngör (2004, 2009) torna a malha uma triangulação de Delaunay, caso essa tenha deixado de ser, exceto se tiver ocorrido o emaranhamento das arestas, apresentando erros caso triângulos com essas arestas sejam selecionados para refinamento. Após avançar a etapa de tempo,

executa-se o MVF, o algoritmo CMr com o vértice pseudoperiférico v como vértice inicial e o MGC, com precisão numérica ε. O procedimento para movimentar os vértices e realizar o refinamento de triângulos com $\measuredangle < \alpha$ pode ser observado no algoritmo (10). Os parâmetros de entrada do algoritmo (10) são mostrados na tabela (1), página 74. Por fim, verifica-se se houve o emaranhamento de arestas, por meio de um método que busca vértices dentro dos triângulos. Um método de força bruta mostrou-se demasiadamente demorado portanto, utilizou-se um método que verifica se os vértices vizinhos e os vizinhos desses encontram-se dentro do triângulo em questão.

Apesar de o algoritmo de Üngör (2004, 2009) utilizar a métrica CER, para se avaliar a qualidade da malha, utiliza-se a métrica SRQ no algoritmo (10). Inicialmente, considerou-se a possibilidade de se utilizar também a métrica CER como critério de parada do movimento de vértices. Entretanto, para utilizar a métrica CER e para existir a possibilidade de movimentar a malha por sucessivas vezes com essa métrica, seria necessário criar uma malha com um ângulo mínimo de 32°, e movimentar a malha enquanto $\alpha_{min} > 30°$, criando uma "folga" de 2°, por exemplo. Caso contrário, cada função seria executada uma só vez. A utilização da SRQ permitiu que o movimento de malhas com uma função monitora pudesse ser executado sucessivas vezes, sem haver emaranhamentos da malha.

3.6 Resumo

Nesse capítulo, apresentou-se a biblioteca de aritmética de precisão arbitrária MPFR e os motivos que levaram à sua escolha. Também elucidou-se como é criada a triangulação de Delaunay inicial, o refinamento adaptativo e o procedimento que realiza o movimento dos vértices.

Definiu-se uma nova função monitora, denotada Λ. Também

Entrada: Malha inicial M, α, β, ε, η, Δt e t_{final}.
// Verifique tabela (1).
Saída: Solução da equação do calor por discretizações de volunes finitos com diagrama de Voronoi.

```
1  início
2      t ← 1; // Para t=0, considera-se o algoritmo (9).
3      enquanto (t ≤ t_final) faça
4          repita
5              // Movimenta a malha e calcula υ_min
6              houveMov ← MovimentaMalha(β, M);
7              υ_min ← ShapeRegularityQuality(M);
8              se (houveMov e υ_min ≥ η) então
9                  // Gera sistema linear pelo MVF.
10                 MetodoVolumesFinitos(M, t × Δt);
11                 // Executa reordenação dos vértices.
12                 v ← AlgoritmoVerticePseudoPeriferico(M);
13                 AlgoritmoCuthillMcKeeReverso(M, v);
14                 // Atualiza valores da EDP pelo MGC.
15                 MetodoGradientesConjugados(M, ε);
16         até (houveMov = falso) ou (υ_min < η);
17         se (υ_min < η) então
18             // Refina triângulos com ∡ < α.
19             RefinaPorAlgoritmoUngor(α, M); // se a malha
20             // deixou de ser uma triangulação de
21             // Delaunay, off-centers fará com que seja.
22         se (houveMov) então
23             // Gera sistema linear pelo MVF.
24             MetodoVolumesFinitos(M, t × Δt);
25             // Executa reordenação dos vértices.
26             v ← AlgoritmoVerticePseudoPeriferico(M);
27             AlgoritmoCuthillMcKeeReverso(M, v);
28             // Atualiza valores da EDP pelo MGC.
29             MetodoGradientesConjugados(M, ε);
30         t ← t + 1; // Avança a etapa de tempo.
31         // Gera sistema linear para etapa t + 1.
32         MetodoVolumesFinitos(M, t × Δt);
33         // Executa reordenação dos vértices.
34         v ← AlgoritmoVerticePseudoPeriferico(M);
35         AlgoritmoCuthillMcKeeReverso(M, v);
36         // Atualiza valores da EDP para etapa t + 1.
37         MetodoGradientesConjugados(M, ε);
```

Algoritmo 10: Método de movimento da malha (malha móvel).

esclareceu-se como é realizado o movimento dos vértices enquanto atender um critério de qualidade mínima, definido por uma métrica de qualidade geométrica. Outras funções monitoras foram apresentadas, que serão comparadas com a função monitora Λ. Também apresentam-se pseudocódigos de todos os procedimentos.

4 TESTES EXPERIMENTAIS E DISCUSSÃO DOS RESULTADOS

Este capítulo tem o objetivo de apresentar os resultados dos experimentos realizados. Os detalhes referentes à implementação foram elucidados no capítulo (3) portanto, os detalhes das variáveis e dos algoritmos citados também se encontram nesse capítulo. Utiliza-se uma precisão de cinco casas decimais na exibição dos resultados e, para valores menores que 10^{-3} ou valores grandes, emprega-se a notação científica, de forma que $1e-4 = 0,0001$.

Os experimentos numéricos foram realizados utilizando o sistema operacional Ubuntu 12.04 LTS 64 bits, processador Intel(R) i3 de 3.10GHz e memória de 16Gb. De modo a se obter soluções em tempo hábil e que atendesse a precisão possível de acordo com a memória disponível, utilizou-se 512 bits de precisão na configuração da biblioteca *Multiple-Precision Floating-point computations with correct Rounding*. Os resultados descritos aqui são uma seleção representativa de um grande número de experimentos.

O domínio discretizado nos experimentos trata-se de um quadrado, de dimensões 50×50, com o valor das condições de fronteira nas faces leste, oeste e sul igual a dez, e na face norte, igual a zero. A estimativa inicial da equação diferencial parcial (EDP), em todos os vértices internos, na iteração inicial no tempo, é igual a zero.

Quanto às variáveis da tabela (1), mostrada na página 74, definiram-se dez iterações no tempo, denotadas por t, tal que $t_{final} = 10$, com uma variação $\Delta t = 0,1$. O ângulo mínimo α, conforme a subseção (2.1.4), página 21, é definido para $\alpha = 30^{\circ}$ com $\rho_{\alpha} = 1,0$. Essa escolha do valor de α se deve por estar próximo dos limites práticos do algoritmo de refinamento. O parâmetro de refinamento adaptativo θ, definido na subseção (3.3), página

69, é tal que $\theta = 1e-10$, tornando menos brusca a diferença na proporção dos triângulos. A precisão numérica ε do métodos dos gradientes conjugados é $\varepsilon = 1e-10$, com o intuito de obter uma precisão aceitável dos resultados, minimizando o número de iterações desse método, conforme observado no algoritmo (6), apresentado na subseção (2.3.3), página 37. O movimento da malha é repetido enquanto $\upsilon_{min} > \eta$ tal que o valor $\eta = 0,6$, em que ambos os valores de υ_{min} e η denotam os valores da métrica *Shape Regularity Quality* (SRQ), apresentados, respectivamente, na subseção (2.6), página 65 e na subseção (3.5), página 72. A escolha desse valor de η é devido ao limite do algoritmo de Üngör (2004, 2009), que consegue gerar malhas com ângulo mínimo próximo à 32°. Os experimentos mostraram que malhas geradas pelo algoritmo (9), configuradas com $\eta = 0,6$ e $\alpha = 30^{\circ}$ possuem $\upsilon_{min} = 0,6$ e $\rho_{max} = 1$

Para todas as funções monitoras, realizam-se comparações, apresentadas nas próximas subseções, para diferentes malhas, utilizando o parâmetro de adaptatividade $\beta = 0,1$, $\beta = 0,2$, $\beta = 0,3$, $\beta = 0,4$, $\beta = 0,5$ e $\beta = 0,6$. Os valores dessas variáveis estão resumidos na tabela (2). Nos experimentos utilizando $\beta = 0,6$ no entanto, ocorreram emaranhamentos na função Λ. Nos resultados apresentados, em que é informado que utilizou-se $\beta = 0,1$, equivale a $\lambda = 0,1$ e $\mu = 0,1001$, na função Ξ, e $\lambda = \mu = 0,1$ na função Ψ. Ao informar que utilizou-se $\beta = 0,2$, equivale a $\lambda = 0,2$ e $\mu = 0,2001$, na função Ξ, e $\lambda = \mu = 0,2$, na função Ψ, e assim suscessivamente.

O grafo $G = (L_V, L_E)$, que representa o *Planar Straight Line Graph*, contém o conjunto de vértices L_V, formado por $v_0 = (0,0), v_1 = (50,0), v_2 = (0,50), v_3 = (50,50), v_4 = (10,10), v_5 = (25,25)$ e $v_6 = (40,40)$, e o conjunto de arestas L_E, formado por $\{(v_0,v_1),(v_0,v_2),(v_3,v_2),(v_3,v_1)\}$. Quanto aos parâmetros exclusivos de algumas funções monitoras, que quantificam o

Tabela 2 Valores das Variáveis

Variável	Algoritmo	Valor
L_V	(9)	$v_0 = (0,0), v_1 = (50,0), v_2 = (0,50), v_3 = (50,50),$ $v_4 = (10,10), v_5 = (25,25), v_6 = (40,40)$
L_A	(9)	$(v_0, v_1), (v_0, v_2), (v_3, v_2), (v_3, v_1)$
θ	(9)	1e-10
χ	(9)	15.000, 30.000, 60.000 e 100.000
β	(10)	0,1, 0,2, 0,3, 0,4, 0,5 e 0,6
λ_Ψ	(10)	0,1, 0,2, 0,3, 0,4, 0,5 e 0,6
λ_Ξ	(10)	0,1, 0,2, 0,3, 0,4, 0,5 e 0,6
μ_Ψ	(10)	0,1, 0,2, 0,3, 0,4, 0,5 e 0,6
μ_Ξ	(10)	0,1001, 0,2001, 0,3001, 0,4001, 0,5001 e 0,6001
ι	(10)	1
η	(10)	0,6
t_{final}	(10)	10
Δt	(9) e (10)	0,1
α	(9) e (10)	30°
ε	(9) e (10)	1e-10

Legenda: Valores das variáveis referentes ao parâmetros de entrada dos algoritmos (9) e (10).

movimento, na função Υ utilizou $\iota = 1$. Na função Ξ, realiza-se experimentos com $\lambda = 0,1$ e $\mu = 0,1001$, $\lambda = 0,2$ e $\mu = 0,2001$, $\lambda = 0,3$ e $\mu = 0,3001$, $\lambda = 0,4$ e $\mu = 0,4001$, $\lambda = 0,5$ e $\mu = 0,5001$ e $\lambda = 0,6$ e $\mu = 0,6001$. Na função, Ψ, utiliza-se $\lambda = \mu = 0,1$, $\lambda = \mu = 0,2$, $\lambda = \mu = 0,3$, $\lambda = \mu = 0,4$, $\lambda = \mu = 0,5$ e $\lambda = \mu = 0,6$ mostrados na subseção (3.5), página 72. Esses valores se encontram resumidos na tabela (2). Para diferenciar os valores de λ e μ das funções Ξ e Ψ, são denotados na tabela (2), respectivamente, por λ_Ξ, λ_Ψ, μ_Ξ e μ_Ψ. Em outros trabalhos, conforme mencionado na subseção (2.4), página 44, utiliza-se $\lambda = \mu = 1$ na função bi-laplaciana entretanto, nos testes realizados com esse valor ocorreu o emaranhamento das arestas, por isso optou-se por utilizar esses valores de λ e μ.

Para que se possa comparar visualmente as diferenças de movimento entre as funções monitoras, criou-se uma malha por refinamento adaptativo, conforme o pseudocódigo (9), com 3.841 vértices e avançou-se dez iterações temporais, movimentando os vértices conforme o pseudocódigo (10), sob as condições de configuração mencionadas nesta subseção. Exemplos de malhas geradas após o movimento, guiado pelas funções monitoras Λ, Γ, Ξ, Ψ e Υ podem ser visualizadas na figura (11).

Nas próximas subseções, apresenta-se os resultados comparativos entre as funções monitoras Λ, Γ, Ξ, Ψ e Υ.

4.1 Comparativo funções monitoras

As malhas, geradas pelo algoritmo (9), que foi configurado com número mínimo de vértices $\chi = 15.000$, $\chi = 30.000$, $\chi = 60.000$ e $\chi = 100.000$, possuem, respectivamente, um total de 15.788, 32.515, 67.963 e 142.253 vértices. Todas as malhas possuem, inicialmente, as métricas de qualidade $\rho_{max} = 1,0$ e $\upsilon_{min} = 0,6$. Na etapa temporal $t = 1$, previamente ao se movimentar os vértices, o maior valor de γ, denotado γ_{max}, e o valor médio de γ, denotado γ_{mean}, das malhas geradas, encontram-se resumidos na tabela (3). O valor de γ_{mean} tende a diminuir ao avançar a etapa temporal. A cada iteração do laço enquanto, linhas 7 a 7, do algoritmo (9) aumenta-se o valor de γ_{mean}, inserindo-se novos vértices nas regiões de maior variação. Realizar o movimento dos vértices em direção à região de maior variação também aumenta o valor de γ_{mean}.

Pode-se verificar nas tabelas (4) a (9) os valores gerais comparativos executando-se o algoritmo (10) para cada função monitora. Utilizaram-se malhas geradas pelo algoritmo (9), com 15.788, 32.515, 67.963 e 142.253 vértices, que são movimentadas pelo algoritmo (10), utilizando-se $\beta = 0,1$,

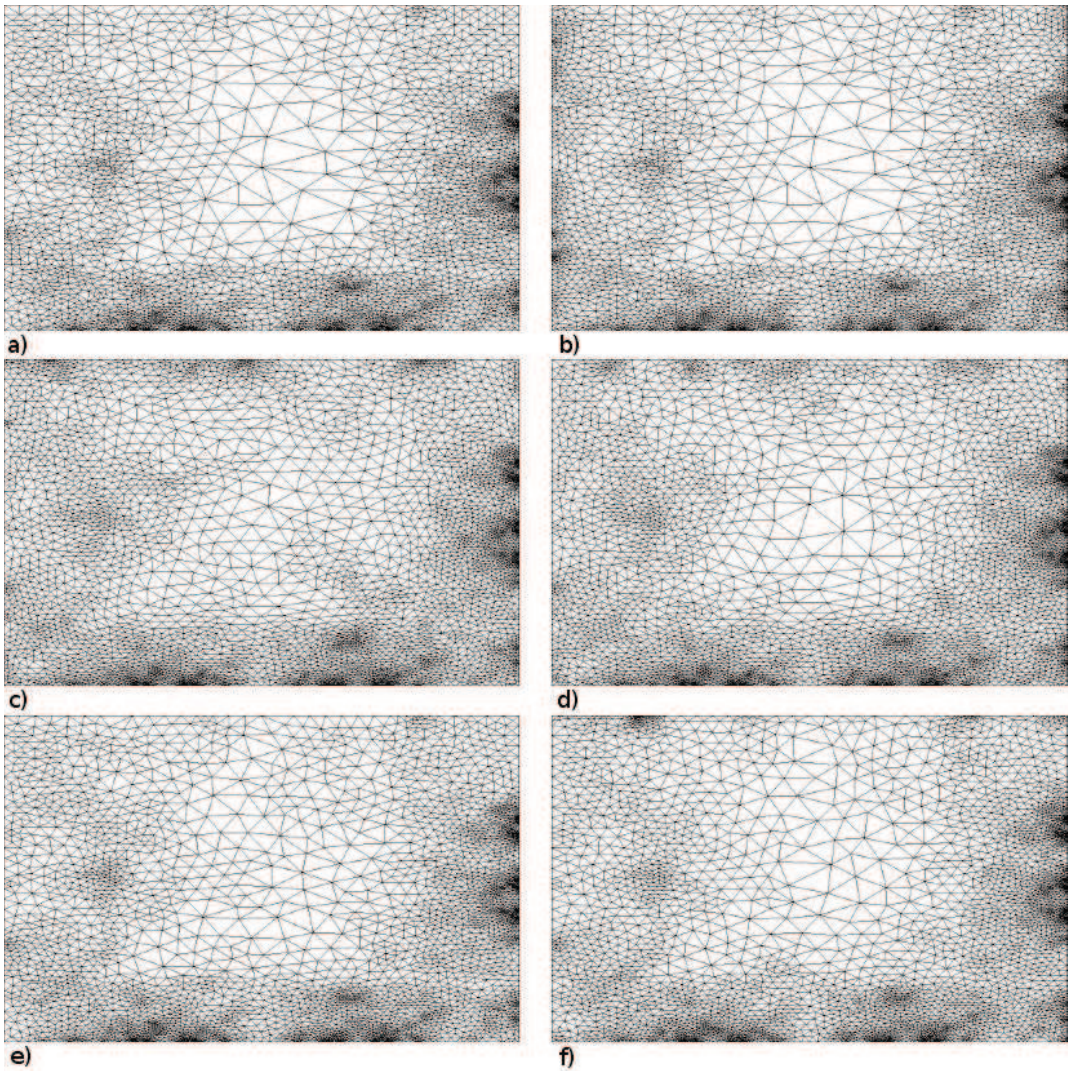

Figura 11 Malhas de exemplo de execução das funções monitoras
Legenda: Em a) malha inicial com 3.841 vértices. Em b), malha final movimentada pela função Λ; em c), pela função Γ; em d), pela função Ξ, em e), pela função Ψ e em f), pela função Υ.

$\beta = 0,2$, $\beta = 0,3$, $\beta = 0,4$, $\beta = 0,5$ e $\beta = 0,6$. Nessas tabelas, nas linhas rotuladas "Total de vértices", tem-se o total de vértices no fim da execução do algoritmo (10), que realiza o movimento dos vértices e a melhoria da

Tabela 3 Valores de γ_{max} e γ_{mean}

χ	15.000	30.000	60.000	100.000
γ_{max}	30,25185	30,91983	45,65052	65,18423
γ_{mean}	3,08635	3,58712	4,08809	4,47127

Legenda: Valores de γ_{max} e γ_{mean}, respectivamente, o maior valor de γ e o valor médio de γ, na etapa temporal $t = 1$.

qualidade da malha. Nas linhas rotuladas "*Clocks* exp." tem-se o total de *clocks* para execução total do experimento. Nas linhas rotuladas "*Clocks* para mov.", tem-se o total de *clocks* para movimentar os vértices, linha 6 do algoritmo (10). Nas linhas rotuladas "Rep. linhas 4-16", apresenta-se o total de repetições das linhas 4 a 16 do algoritmo (10). Nas linhas rotuladas "Valor γ_{max}", tem-se o valor de γ_{max} no fim do experimento. E, nas linhas rotuladas "Valor γ_{mean}", expõe-se o valor de γ_{mean} da malha.

Na penúltima coluna das tabelas (4) a (9), rotulada "S/Mov.1", apresentam-se os resultados obtidos avançando as dez etapas temporais para cada uma das malhas iniciais, geradas pelo algoritmo (9), sem movimentar os vértices, configuradas com os valores de entrada conforme a tabela (2). Na última coluna dessas tabelas, rotulada "S/Mov.2", apresentam-se os resultados obtidos avançando as dez etapas temporais, sem movimentar os vértices, utilizando uma malha com uma maior quantidade de vértices.

Com os dados da penúltima coluna, das tabelas (4) a (9), confirma-se que movimentar os vértices aumenta o valor do gradiente, conforme pode-se verificar os valores de γ_{max} e γ_{mean}. No entanto, em alguns resultados os valores de γ_{max} foram menores que o experimento "S/Mov.1", como por exemplo nas funções Ξ e Ψ com $\chi = 60.000$, $\beta = 0,1$ e $\beta = 0,2$. Acredita-se que a etapa de movimento negativo dessas funções, controlado por μ, afetou os triângulos com maiores valores de γ da malha. Nota-se também que, no

Tabela 4 Comparativo das funções executadas com $\beta = 0,1$

	Função	Λ	Γ	Ξ	Ψ	Υ	S/Mov.1	S/Mov.2
$\chi = 15.000$	Total de vértices	16.255	16.618	16.615	16.481	17.060	15.788	32.515
	Clocks exp.	5,34e10	1,78e10	6,71e10	6,43e10	2,71e10	1,33e09	4,05e09
	Clocks para mov.	6,00e09	1,32e09	9,08e09	8,62e09	2,84e09	-	-
	Rep. linhas 4-16	163	24	344	267	49	-	-
	Valor γ_{max}	128,89	26,39	26,26	26,21	180,42	26,13	36,13
	Valor γ_{mean}	1,02420	0,99315	0,99602	0,99759	0,99436	0,98318	1,03107
$\chi = 30.000$	Total de vértices	33.034	33.790	33.755	33.551	33.908	32.515	67.963
	Clocks exp.	1,03e11	5,12e10	1,33e11	1,23e11	6,25e10	4,05e09	1,53e10
	Clocks para mov.	7,97e09	2,96e09	1,35e10	1,19e10	4,27e09	-	-
	Rep. linhas 4-16	105	26	291	183	37	-	-
	Valor γ_{max}	50,36	51,24	50,21	49,43	111,51	36,13	52,37
	Valor γ_{mean}	1,06513	1,05096	1,05477	1,05560	1,05921	1,03107	1,06202
$\chi = 60.000$	Total de vértices	68.840	70.224	69.926	70.076	70.498	67.963	142.253
	Clocks exp.	2,84e11	1,48e11	3,36e11	3,51e11	1,51e11	1,53e10	5,30e10
	Clocks para mov.	1,69e10	6,49e09	2,43e10	2,60e10	6,81e09	-	-
	Rep. linhas 4-16	109	27	180	215	31	-	-
	Valor γ_{max}	177,60	49,47	52,24	52,35	174,41	52,37	72,37
	Valor γ_{mean}	1,12378	1,09497	1,09474	1,09483	1,09755	1,06202	1,11495
$\chi = 100.000$	Total de vértices	143.107	145.346	145.130	145.163	146.585	142.253	298.574
	Clocks exp.	6,00e11	3,80e11	6,25e11	6,26e11	4,36e11	5,30e10	2,18e11
	Clocks para mov.	2,30e10	9,66e09	2,84e10	2,92e10	1,27e10	-	-
	Rep. linhas 4-16	71	20	92	93	28	-	-
	Valor γ_{max}	104,09	101,95	94,64	94,82	394,68	72,37	104,80
	Valor γ_{mean}	1,12121	1,11986	1,11864	1,11845	1,12224	1,11495	1,13468

Legenda: Comparativo das funções executadas com $\beta = 0,1$, utilizando malhas geradas pelo algoritmo (9), configurado com $\chi = 15.000$, $\chi = 30.000$, $\chi = 60.000$ e $\chi = 100.000$, que possuem, inicialmente e respectivamente, um total de 15.788, 32.515, 67.963 e 142.253 vértices.

Tabela 5 Comparativo das funções executadas com $\beta = 0,2$

	Função	Λ	Γ	Ξ	Ψ	Υ	S/Mov.1	S/Mov.2
$\chi = 15.000$	Total de vértices	16.263	17.077	16.596	16.589	16.785	15.788	32.515
	Clocks exp.	2,82e10	1,55e10	2,73e10	2,63e10	1,61e10	1,33e09	4,05e09
	Clocks para mov.	2,70e09	1,04e09	3,01e09	2,85e09	1,05e09	-	-
	Rep. linhas 4-16	80	19	73	70	20	-	-
	Valor γ_{max}	237,75	48,50	26.22	26.21	218,35	26,13	36,13
	Valor γ_{mean}	1,02373	0,97805	0.99631	0,99664	1.02131	0,98318	1,03107
$\chi = 30.000$	Total de vértices	33.182	34.069	33.775	33.770	34.284	32.515	67.963
	Clocks exp.	6,91e10	4,13e10	6,81e10	6,82e10	4,27e10	4,05e09	1,53e10
	Clocks para mov.	4,63e09	1,91e09	5,92e09	5,57e09	1,94e09	-	-
	Rep. linhas 4-16	61	17	69	69	18	-	-
	Valor γ_{max}	99,92	53,50	50,17	50,18	105,29	36,13	52,37
	Valor γ_{mean}	1,07718	1,04963	1,05470	1,05460	1,06094	1,03107	1,06202
$\chi = 60.000$	Total de vértices	68.789	70.889	70.003	69.997	71.222	67.963	142.253
	Clocks exp.	1,76e11	1,23e11	1,67e11	1,66e11	1,30e11	1,53e10	5,30e10
	Clocks para mov.	8,39e09	3,79e09	8,46e09	8,41e09	4,69e09	-	-
	Rep. linhas 4-16	64	16	45	45	20	-	-
	Valor γ_{max}	221,19	151,52	52,25	52.26	221,87	52,37	72,37
	Valor γ_{mean}	1,10905	1,09070	1.09452	1.09468	1,09799	1,06202	1,11495
$\chi = 100.000$	Total de vértices	143.319	145.768	145.355	145.352	147.731	142.253	298.574
	Clocks exp.	3,95e11	3,08e11	3,72e11	3,73e11	3,59e11	5,30e10	2,18e11
	Clocks para mov.	1,22e10	5,33e09	1,17e10	1,12e10	8,99e09	-	-
	Rep. linhas 4-16	45	11	25	25	19	-	-
	Valor γ_{max}	108,01	102,34	95,79	95,81	135,68	72,37	104,80
	Valor γ_{mean}	1,12327	1,12005	1,11871	1,11870	1,12471	1,11495	1,13468

Legenda: Comparativo das funções executadas com $\beta = 0,2$, utilizando malhas geradas pelo algoritmo (9), configurado com $\chi = 15.000$, $\chi = 30.000$, $\chi = 60.000$ e $\chi = 100.000$. que possuem, inicialmente e respectivamente, um total de 15.788, 32.515, 67.963 e 142.253 vértices.

Tabela 6 Comparativo das funções executadas com $\beta = 0,3$

	Função	Λ	Γ	Ξ	Ψ	Υ	S/Mov.1	S/Mov.2
$\chi = 15.000$	Total de vértices	16.252	16.928	16.911	16.889	17.362	15.788	32.515
	Clocks exp.	1,93e10	1,18e10	2,07e10	2,08e10	1,49e10	1,33e09	4,05e09
	Clocks para mov.	1,47e09	5,72e08	2,04e09	2,08e09	9,19e08	-	-
	Rep. linhas 4-16	35	10	50	50	18	-	-
	Valor γ_{max}	50,62	25,05	26,34	26,35	226,14	26,13	36,13
	Valor γ_{mean}	1,03007	0,98279	0,98485	0,98614	1,00625	0,98318	1,03107
$\chi = 30.000$	Total de vértices	33.094	34.521	34.005	34.007	34.605	32.515	67.963
	Clocks exp.	5,04e10	3,82e10	5,16e10	5,19e10	3,99e10	4,05e09	1,53e10
	Clocks para mov.	2,82e09	1,56e09	3,75e09	3,70e09	1,82e09	-	-
	Rep. linhas 4-16	31	13	40	40	15	-	-
	Valor γ_{max}	103,83	53,56	50,58	50,59	184,96	36,13	52,37
	Valor γ_{mean}	1,06771	1,04496	1,05362	1,05377	1,05276	1,03107	1,06202
$\chi = 60.000$	Total de vértices	68.904	71.040	70.587	70.588	71.576	67.963	142.253
	Clocks exp.	1,48e11	1,07e11	1,35e11	1,35e11	1,11e11	1,53e10	5,30e10
	Clocks para mov.	6,26e09	2,92e09	6,29e09	5,95e09	3,28e09	-	-
	Rep. linhas 4-16	49	12	32	32	14	-	-
	Valor γ_{max}	207,01	75,58	52,55	52,56	221,22	52,37	72,37
	Valor γ_{mean}	1,11019	1,09184	1,09593	1,09592	1,09860	1,06202	1,11495
$\chi = 100.000$	Total de vértices	143.316	147.723	146.457	146.452	147.899	142.253	298.574
	Clocks exp.	3,65e11	3,40e11	3,52e11	3,51e11	3,33e11	5,30e10	2,18e11
	Clocks para mov.	8,88e09	6,97e09	8,82e09	9,52e09	5,74e09	-	-
	Rep. linhas 4-16	30	13	19	19	12	-	-
	Valor γ_{max}	107,99	90,95	98,87	98,88	224,65	72,37	104,80
	Valor γ_{mean}	1,12210	1,11902	1,11867	1,11870	1,12318	1,11495	1,13468

Legenda: Comparativo das funções executadas com $\beta = 0,3$, utilizando malhas geradas pelo algoritmo (9), configurado com $\chi = 15.000$, $\chi = 30.000$, $\chi = 60.000$ e $\chi = 100.000$. que possuem, inicialmente e respectivamente, um total de 15.788, 32.515, 67.963 e 142.253 vértices.

Tabela 7 Comparativo das funções executadas com $\beta = 0,4$

	Função	Λ	Γ	Ξ	Ψ	Υ	S/Mov.1	S/Mov.2
$\chi = 15.000$	Total de vértices	16.701	17.579	16.837	16.821	17.355	15.788	32.515
	Clocks exp.	1,62e10	1,27e10	1,55e10	1,56e10	1,32e10	1,33e09	4,05e09
	Clocks para mov.	1,14e09	6,37e08	1,22e09	1,28e09	7,26e08	-	-
	Rep. linhas 4-16	30	11	26	25	13	-	-
	Valor γ_{max}	388,50	58,95	26,29	26,28	211,07	26,13	36,13
	Valor γ_{mean}	1,14899	0,96986	0,99102	0,99077	1,01422	0,98318	1,03107
$\chi = 30.000$	Total de vértices	33.315	34.980	34.391	34.381	34.746	32.515	67.963
	Clocks exp.	4,27e10	3,54e10	4,47e10	4,50e10	3,69e10	4,05e09	1,53e10
	Clocks para mov.	1,88e09	1,13e09	2,67e09	2,51e09	1,29e09	-	-
	Rep. linhas 4-16	21	10	29	29	11	-	-
	Valor γ_{max}	224,84	65,94	50,86	50,85	215,36	36,13	52,37
	Valor γ_{mean}	1,11159	1,04457	1,04929	1,04969	1,05862	1,03107	1,06202
$\chi = 60.000$	Total de vértices	68.846	71.937	71.170	71.125	72.224	67.963	142.253
	Clocks exp.	1,22e11	1,06e11	1,18e11	1,17e11	1,07e11	1,53e10	5,30e10
	Clocks para mov.	4,39e09	2,50e09	4,39e09	4,41e09	2,63e09	-	-
	Rep. linhas 4-16	24	11	21	20	11	-	-
	Valor γ_{max}	227,53	106,15	52,68	52,64	365,12	52,37	72,37
	Valor γ_{mean}	1,10886	1,09097	1,09448	1,09445	1,10247	1,06202	1,11495
$\chi = 100.000$	Total de vértices	143.306	149.079	147.478	147.508	148.898	142.253	298.574
	Clocks exp.	3,48e11	3,33e11	3,42e11	3,41e11	3,20e11	5,30e10	2,18e11
	Clocks para mov.	7,48e09	5,87e09	7,93e09	8,20e09	5,07e09	-	-
	Rep. linhas 4-16	19	12	16	16	10	-	-
	Valor γ_{max}	298,82	387,05	100,34	100,35	212,82	72,37	104,80
	Valor γ_{mean}	1,12429	1,12151	1,11912	1,11908	1,12472	1,11495	1,13468

Legenda: Comparativo das funções executadas com $\beta = 0,4$, utilizando malhas geradas pelo algoritmo (9), configurado com $\chi = 15.000$, $\chi = 30.000$, $\chi = 60.000$ e $\chi = 100.000$. que possuem, inicialmente e respectivamente, um total de 15.788, 32.515, 67.963 e 142.253 vértices.

Tabela 8 Comparativo das funções executadas com $\beta = 0,5$

	Função	Λ	Γ	Ξ	Ψ	Υ	S/Mov.1	S/Mov.2
$\chi = 15.000$	Total de vértices	16.917	18.196	17.172	17.281	17.933	15.788	32.515
	Clocks exp.	1,50e10	1,29e10	1,44e10	1,46e10	1,30e10	1,33e09	4,05e09
	Clocks para mov.	4,24e08	5,98e08	9,41e08	9,67e08	6,16e08	-	-
	Rep. linhas 4-16	10	10	19	20	11	-	-
	Valor γ_{max}	67,98	50,23	26,38	26,40	447,48	26,13	36,13
	Valor γ_{mean}	1,04517	0,95807	0,98742	0,98068	1,04418	0,98318	1,03107
$\chi = 30.000$	Total de vértices	33.852	35.659	34.639	34.640	35.215	32.515	67.963
	Clocks exp.	4,17e10	3,78e10	4,16e10	4,13e10	3,59e10	4,05e09	1,53e10
	Clocks para mov.	1,60e09	1,23e09	2,20e09	2,04e09	1,09e09	-	-
	Rep. linhas 4-16	16	11	20	20	10	-	-
	Valor γ_{max}	219,21	48,97	51,01	51,02	374,23	36,13	52,37
	Valor γ_{mean}	1,09169	1,03820	1,05415	1,05422	1,05477	1,03107	1,06202
$\chi = 60.000$	Total de vértices	69.059	72.892	71.639	71.622	74.041	67.963	142.253
	Clocks exp.	1,18e11	1,06e11	1,10e11	1,10e11	1,11e11	1,53e10	5,30e10
	Clocks para mov.	4,07e09	2,38e09	3,32e09	3,34e09	3,01e09	-	-
	Rep. linhas 4-16	25	10	14	14	12	-	-
	Valor γ_{max}	417,86	104,08	67,41	67.42	758,03	52,37	72,37
	Valor γ_{mean}	1,12834	1,08956	1,09577	1,09572	1,11319	1,06202	1,11495
$\chi = 100.000$	Total de vértices	143.497	150.672	148.841	148.791	151.400	142.253	298.574
	Clocks exp.	3,36e11	3,25e11	3,28e11	3,29e11	3,30e11	5,30e10	2,18e11
	Clocks para mov.	6,49e09	5,05e09	6,02e09	6,00e09	5,51e09	-	-
	Rep. linhas 4-16	17	10	12	12	10	-	-
	Valor γ_{max}	202,91	99,86	100,82	100,83	723,86	72,37	104,80
	Valor γ_{mean}	1,12883	1,11895	1,11961	1,11971	1,13135	1,11495	1,13468

Legenda: Comparativo das funções executadas com $\beta = 0,5$, utilizando malhas geradas pelo algoritmo (9), configurado com $\chi = 15.000$, $\chi = 30.000$, $\chi = 60.000$ e $\chi = 100.000$, que possuem, inicialmente e respectivamente, um total de 15.788, 32.515, 67.963 e 142.253 vértices.

Tabela 9 Comparativo das funções executadas com $\beta = 0,6$

	Função	Λ	Γ	Ξ	Ψ	Υ	S/Mov.1	S/Mov.2
$\chi = 15.000$	Total de vértices	-	19.997	18.528	18.517	18.614	15.788	32.515
	Clocks exp.	-	1,40e10	1,37e10	1,36e10	1,35e10	1,33e09	4,05e09
	Clocks para mov.	-	6,32e08	6,49e08	6,51e08	5,93e08	-	-
	Rep. linhas 4-16	-	10	11	11	10	-	-
	Valor γ_{max}	-	97,91	26,25	26.26	399,67	26,13	36,13
	Valor γ_{mean}	-	0,91555	0,99277	0,99308	1,03788	0,98318	1,03107
$\chi = 30.000$	Total de vértices	-	38.077	37.604	37.690	37.550	32.515	67.963
	Clocks exp.	-	3,23e10	2,42e10	3,21e10	2,44e10	4,05e09	1,53e10
	Clocks para mov.	-	1,00e09	7,06e08	9,50e08	7,39e08	-	-
	Rep. linhas 4-16	-	10	10	10	10	-	-
	Valor γ_{max}	-	215,22	50,59	50,60	717,38	36,13	52,37
	Valor γ_{mean}	-	1,01722	1,04631	1,04435	1,07486	1,03107	1,06202
$\chi = 60.000$	Total de vértices	-	77.253	77.471	77.536	77.250	67.963	142.253
	Clocks exp.	-	1,15e11	1,14e11	1,16e11	1,17e11	1,53e10	5,30e10
	Clocks para mov.	-	2,64e09	2,67e09	2,44e09	2,51e09	-	-
	Rep. linhas 4-16	-	10	10	10	10	-	-
	Valor γ_{max}	-	371,55	52,58	52,59	1.427,98	52,37	72,37
	Valor γ_{mean}	-	1,07850	1,09331	1.09449	1,11611	1,06202	1,11495
$\chi = 100.000$	Total de vértices	-	157.280	162.631	162.660	157.798	142.253	298.574
	Clocks exp.	-	3,54e11	3,61e11	3,62e11	3,59e11	5,30e10	2,18e11
	Clocks para mov.	-	5,46e09	5,36e09	5,18e09	5,04e09	-	-
	Rep. linhas 4-16	-	10	10	10	10	-	-
	Valor γ_{max}	-	353,98	105,56	105.57	2.236,51	72,37	104,80
	Valor γ_{mean}	-	1,11889	1,11625	1,11611	1,14206	1,11495	1,13468

Legenda: Comparativo das funções executadas com $\beta = 0,6$, utilizando malhas geradas pelo algoritmo (9), configurado com $\chi = 15.000$, $\chi = 30.000$, $\chi = 60.000$ e $\chi = 100.000$, que possuem, inicialmente e respectivamente, um total de 15.788, 32.515, 67.963 e 142.253 vértices. A função Λ apresentou emaranhamentos em todos os experimentos.

caso da função Γ, no experimento $\chi = 60.000$ e $\beta = 0,1$, também obteve-se valor de γ_{max} menor que no experimento "S/Mov.1".

Apresentam-se os experimentos na última coluna, das tabelas (4) a (9), a fim de comparar o custo do refinamento adaptativo com o custo ao se movimentar os vértices. Em todos os experimentos, movimentar os vértices acarretou um custo computacional maior que o experimento "S/Mov.2". Na tabela (10), dividiu-se o tempo de execução do experimento "S/Mov.2" pelo tempo de execução da função com menor custo computacional. A partir dessa tabela, verifica-se a média dessa porcentagem para cada valor de β, com arredondamento para cima. Os melhores valores foram obtidos utilizando-se $\beta = 0,6$.

Ainda em relação ao experimento "S/Mov.2", a maioria dos experimentos, em que movimentou-se os vértices, apresentou valores de γ_{mean} relativamente próximos aos valores do experimento "S/Mov.2" e valores de γ_{max} superiores, com um custo computacional também superior ao custo do experimento "S/Mov.2". Um motivo do valor de γ_{mean} não ser muito superior se deve ao parâmetro β, que é definido globalmente, podendo ocorrer degeneração de triângulos localizados em regiões de baixa variação. Logo, esses triângulos necessitariam ser refinados, gerando dois novos triângulos nas regiões de baixa variação, diminuindo o valor de γ_{mean}. Provavelmente, definir um β local apresentaria melhores resultados.

Em alguns casos, ocorreu redução dos valores de γ_{max} e γ_{mean} ao se utilizar malhas com mais vértices. Como por exemplo, a função Γ, no experimento $\beta = 0,2$ e $\chi = 60.000$, possuía $\gamma_{max} = 151,52$, e no experimento com $\beta = 0,2$ e $\chi = 100.000$ esse valor diminuiu para $\gamma_{max} = 102,34$. Algumas análises mostraram que, ao se movimentar os vértices, tornando-os muito próximos, os valores da EDP desses vértices serão praticamente iguais, de-

Tabela 10 Percentagem do tempo de execução

β	χ	Percentual
	15.000	23 %
	30.000	30 %
0,1	60.000	36 %
	100.000	57 %
	Média	37 %
	15.000	26 %
	30.000	37 %
0,2	60.000	43 %
	100.000	71 %
	Média	44 %
	15.000	34 %
	30.000	40 %
0,3	60.000	50 %
	100.000	65 %
	Média	47 %
	15.000	32 %
	30.000	43 %
0,4	60.000	50 %
	100.000	68 %
	Média	48 %
	15.000	31 %
	30.000	42 %
0,5	60.000	50 %
	100.000	67 %
	Média	48 %
	15.000	30 %
	30.000	63 %
0,6	60.000	46 %
	100.000	61 %
	Média	50 %

Legenda: Percentagem do tempo de execução do experimento "S/Mov.2" em relação ao experimento com menor custo computacional ao se movimentar os vértices. Apresenta-se a média dessa percentagem para cada valor de β.

vido à precisão definida por ε. Com isso, triângulos que contenham esses vértices terão o valor de γ diminuído, afetando os valores de γ_{max} e γ_{mean}.

Analisando-se a velocidade de execução do experimento e a quantidade de vértices nas tabelas (4) a (8), percebe-se que, mesmo a função Λ obtendo a menor quantidade final de vértices em todos os experimentos, demandou mais recursos computacionais que a função Γ e Υ pois, necessitou de um número maior de iterações do laço que movimenta os vértices. Ainda, na função Λ é necessário encontrar o maior valor de $\Delta\phi_{max}$. Entretanto, o

tempo de execução é afetado principalmente pelo método dos gradientes conjugados, procedimento que mais demanda recursos, e é executado dentro do laço que movimenta os vértices. As funções Ξ e Ψ necessitaram de um número maior de *clocks* devido à combinação das duas suavizações, que torna o movimento vagaroso em relação às demais funções.

Pelos resultados das tabelas (4) a (8), verifica-se que a função monitora proposta, Λ, apresentou bons resultados ao se analisar a quantidade final de vértices e os valores de γ_{max} e γ_{mean}. A função Γ foi a que necessitou da menor quantidade de *clocks* de execução em quase todos os experimentos, consequência do menor número de iterações da linhas 4 a 16 do algoritmo (10) e, ainda, resultou em γ_{max} e γ_{mean} finais razoáveis. As funções Ξ e Ψ em todos os experimentos sempre estiveram entre os três piores resultados em relação ao número de *clocks* para movimentar os vértices. A função Υ apresentou, em dois experimentos, o melhor custo computacional, e nos demais, o segundo melhor custo computacional e, ainda, apresentou bons resultados em relação aos valores de γ_{max} e γ_{mean}, sempre estando entre os dois melhores resultados no entanto, quanto à quantidade de vértices, obteve o pior desempenho em quase todos os experimentos.

Na tabela (11), apresenta-se a média das dez iterações temporais da métrica *Circunradius-to-shortest Edge Radio* (CER), denotada ρ_{max}, após movimentar os vértices, para cada experimento realizado. Quanto maior o valor da métrica CER, pior é o resultado, pois menor é o ângulo mínimo. Os valores são agrupados para cada experimento. Nessa tabela, pode-se observar que as funções Ψ e Ξ apresentaram os melhores resultados e Γ os piores, na maioria dos casos.

Na tabela (12), expõe-se a média das dez iterações temporais da métrica SRQ, denotada υ_{min}, após movimentar os vértices, para cada ex-

Tabela 11 Média do valor de ρ_{max}

β	χ	Λ	Γ	Ξ	Ψ	Υ
0,1	15.000	1,15824	1,13563	1,11155	1,08556	1,14298
	30.000	1,08060	1,19005	1,09516	1,08460	1,12088
	60.000	1,12058	1,15848	1,07265	1,08692	1,12528
	100.000	1,07919	1,10902	1,04850	1,04756	1,10878
0,2	15.000	1,17876	1,23320	1,13482	1,13301	1,16649
	30.000	1,12482	1,26227	1,09764	1,10480	1,20191
	60.000	1,10528	1,22472	1,08806	1,09810	1,16519
	100.000	1,12786	1,13104	1,06125	1,06139	1,16649
0,3	15.000	1,20717	1,24393	1,19507	1,20842	1,22321
	30.000	1,16724	1,31336	1,12671	1,11300	1,19550
	60.000	1,14563	1,32805	1,11670	1,11709	1,24962
	100.000	1,15702	1,26131	1,08974	1,08982	1,19858
0,4	15.000	1,44468	1,33633	1,19630	1,19693	1,21874
	30.000	1,39521	1,31444	1,25018	1,25135	1,22399
	60.000	1,26923	1,37858	1,16041	1,15642	1,24856
	100.000	1,14348	1,34365	1,14234	1,14245	1,26469
0,5	15.000	6,17023	1,49158	1,28772	1,28006	1,30849
	30.000	2,95550	1,48424	1,25607	1,25613	1,31918
	60.000	1,33681	1,53118	1,24452	1,24459	1,35976
	100.000	1,42481	1,44644	1,24274	1,24271	1,39356
0,6	15.000	-	1,76168	1,39727	1,38998	1,47413
	30.000	-	1,69881	1,45262	1,46328	1,39493
	60.000	-	1,60797	1,50165	1,49268	1,50090
	100.000	-	1,68472	1,60225	1,60185	1,49906

Legenda: Média do valor de ρ_{max}, das dez iterações temporais, após movimentar os vértices.

perimento realizado. Quanto maior o valor da métrica SRQ, melhor é o resultado, pois mais se torna mais semelhante a um triângulo equilátero. Nessa tabela, pode-se observar que as funções Ξ e Ψ apresentam os melhores resultados e a função Γ os piores, na maioria dos casos.

Na tabela (13), apresenta-se a percentagem média de triângulos com $\rho > \rho_\alpha$ das dez iterações temporais, para cada função monitora. O valor de ρ é obtido após movimentar os vértices. Quanto menor o valor da percentagem

Tabela 12 Média do valor de υ_{min}

β	χ	Λ	Γ	Ξ	Ψ	Υ
$0,1$	15.000	0,59465	0,58347	0,59841	0,59834	0,59466
	30.000	0,59832	0,58450	0,59866	0,59878	0,59375
	60.000	0,59599	0,58650	0,59861	0,59834	0,59236
	100.000	0,59594	0,58931	0,59810	0,59798	0,59292
$0,2$	15.000	0,58912	0,56636	0,59341	0,59332	0,58391
	30.000	0,59181	0,56499	0,59721	0,59748	0,58272
	60.000	0,58949	0,57775	0,59303	0,59298	0,58366
	100.000	0,58635	0,58561	0,59323	0,59322	0,58556
$0,3$	15.000	0,57718	0,54930	0,58115	0,58103	0,58021
	30.000	0,58618	0,54770	0,59056	0,59062	0,57412
	60.000	0,57907	0,53761	0,58731	0,58722	0,55938
	100.000	0,58042	0,54905	0,59029	0,59031	0,57673
$0,4$	15.000	0,53673	0,53757	0,57690	0,57544	0,56354
	30.000	0,50970	0,54284	0,57205	0,57274	0,57245
	60.000	0,56615	0,53543	0,57778	0,57807	0,56777
	100.000	0,57560	0,53598	0,58327	0,58325	0,57559
$0,5$	15.000	0,31796	0,51447	0,55775	0,56101	0,55143
	30.000	0,48989	0,48925	0,56352	0,56358	0,54352
	60.000	0,53837	0,48579	0,55343	0,55345	0,52197
	100.000	0,52618	0,50880	0,54023	0,54019	0,50501
$0,6$	15.000	-	0,44853	0,51762	0,51755	0,52100
	30.000	-	0,42959	0,48023	0,48030	0,52184
	60.000	-	0,46645	0,47547	0,47552	0,51617
	100.000	-	0,44364	0,43217	0,43583	0,50000

Legenda: Média do valor de υ_{min}, das dez iterações temporais, após movimentar os vértices.

média de triângulos com $\rho > \rho_\alpha$, melhor é o resultado. Conforme pode-se observar nessa tabela, a função Λ superou as demais em praticamente todos os experimentos. A função Γ apresentou os piores resultados, com grandes percentagens de triângulos com $\rho > \rho_\alpha$, na maioria dos casos.

Na tabela (14), apresenta-se a percentagem média de triângulos com $\upsilon < \eta$ das dez iterações temporais, para cada experimento realizado. O valor de υ é obtido após movimentar os vértices. Quanto menor o valor da

Tabela 13 Percentagem média de triângulos com $\rho > \rho_\alpha$

β	χ	Λ	Γ	Ξ	Ψ	Υ
0,1	15.000	5,72738e-04%	1,02110e-03%	1,00270e-03%	9,21288e-04%	9,31399e-04%
	30.000	3,62734e-04%	7,69637e-04%	7,95189e-04%	7,03225e-04%	7,20160e-04%
	60.000	2,61211e-04%	5,86783e-04%	6,06642e-04%	6,35853e-04%	5,80936e-04%
	100.000	1,69738e-04%	4,40143e-04%	4,87852e-04%	4,88137e-04%	4,99830e-04%
0,2	15.000	5,72549e-04%	1,56123e-03%	1,00608e-03%	9,99809e-04%	8,76757e-04%
	30.000	4,43828e-04%	9,96196e-04%	8,21628e-04%	8,15540e-04%	7,77340e-04%
	60.000	2,74589e-04%	7,25158e-04%	6,42138e-04%	6,39909e-04%	6,81090e-04%
	100.000	2,00632e-04%	5,06478e-04%	5,18327e-04%	5,17625e-04%	6,19685e-04%
0,3	15.000	5,92929e-04%	1,57200e-03%	1,33465e-03%	1,32853e-03%	1,21410e-03%
	30.000	4,31556e-04%	1,16756e-03%	9,62971e-04%	9,63092e-04%	8,95291e-04%
	60.000	2,88652e-04%	8,32678e-04%	7,83318e-04%	7,84044e-04%	7,59816e-04%
	100.000	2,02775e-04%	7,09407e-04%	6,65755e-04%	6,65742e-04%	6,71386e-04%
0,4	15.000	8,30070e-04%	2,13195e-03%	1,28728e-03%	1,27463e-03%	1,31130e-03%
	30.000	4,91979e-04%	1,40499e-03%	1,21152e-03%	1,21162e-03%	1,01763e-03%
	60.000	2,89390e-04%	1,09661e-03%	9,20326e-04%	9,10266e-04%	9,37984e-04%
	100.000	2,01008e-04%	9,03660e-04%	8,07205e-04%	8,07491e-04%	8,25447e-04%
0,5	15.000	9,52267e-04%	2,85801e-03%	1,72572e-03%	1,74603e-03%	1,60877e-03%
	30.000	6,50956e-04%	1,90112e-03%	1,44341e-03%	1,44340e-03%	1,28081e-03%
	60.000	3,39834e-04%	1,39349e-03%	1,16380e-03%	1,16308e-03%	1,29428e-03%
	100.000	2,28333e-04%	1,15386e-03%	1,06381e-03%	1,06533e-03%	1,10419e-03%
0,6	15.000	-	4,30554e-03%	3,35079e-03%	3,34873e-03%	2,31393e-03%
	30.000	-	3,02524e-03%	3,10910e-03%	3,13179e-03%	2,16016e-03%
	60.000	-	2,28737e-03%	2,72696e-03%	2,72887e-03%	1,93507e-03%
	100.000	-	1,94079e-03%	2,79009e-03%	2,79434e-03%	1,82321e-03%

Legenda: Percentagem média de triângulos com $\rho > \rho_\alpha$ das dez iteraçoes temporais.

percentagem média de triângulos com $\upsilon < \eta$, melhor é o resultado. Com os resultados presentes nessa tabela, pode-se observar que as funções Ξ, Ψ e Λ estão entre os melhores resultados, apresentando baixas porcentagens de triângulos com $\upsilon < \eta$. Γ se destaca como a pior, na maioria dos casos.

Na tabela (15) tem-se a percentagem média de crescimento de γ_{mean} das dez iterações temporais, para cada experimento realizado. O valor de γ_{mean} é obtido antes e após movimentar os vértices. Com esses dados, calcula-se a percentagem média de crescimento de γ_{mean} ao movimentar os vértices. A função Γ apresenta as maiores taxas de crescimento na maioria dos experimentos, enquanto que as demais funções alternam-se, apresentando as menores taxas de crescimento.

Com os resultados das tabelas (11), (12), (13) e (14) demonstram que aumentando a dimensão do movimento, controlado pela variável β, tem-se uma maior degeneração da malha. Os resultados da tabela (15) confirmam

Tabela 14 Percentagem média de triângulos com $\upsilon < \eta$

β	χ	Λ	Γ	Ξ	Ψ	Υ
0,1	15.000	4,21279e-05%	5,79641e-05%	3,52825e-05%	3,53504e-05%	4,49827e-05%
	30.000	1,57026e-05%	2,63429e-05%	1,55642e-05%	1,55685e-05%	1,54909e-05%
	60.000	1,49992e-05%	2,01129e-05%	1,04211e-05%	1,11602e-05%	1,56459e-05%
	100.000	6,06970e-06%	7,43434e-06%	5,68778e-06%	6,03846e-06%	7,42422e-06%
0,2	15.000	4,53466e-05%	7,92577e-05%	5,14722e-05%	5,14711e-05%	4,82341e-05%
	30.000	2,03770e-05%	3,55792e-05%	1,55482e-05%	1,70848e-05%	2,16360e-05%
	60.000	1,79853e-05%	2,97400e-05%	1,71697e-05%	1,71696e-05%	1,92977e-05%
	100.000	7,13577e-06%	9,22842e-06%	8,17855e-06%	8,17858e-06%	9,86586e-06%
0,3	15.000	5,49531e-05%	1,08862e-04%	6,73410e-05%	6,73450e-05%	5,99797e-05%
	30.000	2,35184e-05%	6,90715e-05%	2,16443e-05%	2,47064e-05%	2,61928e-05%
	60.000	1,79742e-05%	3,19193e-05%	2,52998e-05%	2,52997e-05%	2,51032e-05%
	100.000	7,49192e-06%	2,04546e-05%	1,13491e-05%	1,10005e-05%	1,12757e-05%
0,4	15.000	1,29631e-04%	1,78129e-04%	6,75374e-05%	6,75304e-05%	7,28941e-05%
	30.000	4,24071e-05%	8,04860e-05%	3,55417e-05%	3,55440e-05%	2,75320e-05%
	60.000	2,32291e-05%	6,20035e-05%	2,97202e-05%	2,97205e-05%	3,01836e-05%
	100.000	8,92079e-06%	3,42136e-05%	1,73917e-05%	1,73910e-05%	1,72703e-05%
0,5	15.000	3,14392e-04%	3,94424e-04%	1,34591e-04%	1,34540e-04%	9,45161e-05%
	30.000	9,23607e-05%	1,76159e-04%	6,18742e-05%	6,18740e-05%	6,75403e-05%
	60.000	2,77326e-05%	1,34458e-04%	4,91227e-05%	4,91227e-05%	7,64596e-05%
	100.000	1,35465e-05%	7,70591e-05%	3,88710e-05%	3,92226e-05%	5,01487e-05%
0,6	15.000	-	7,39840e-04%	3,34625e-04%	3,28766e-04%	1,92190e-04%
	30.000	-	3,81732e-04%	2,20434e-04%	2,21754e-04%	1,53894e-04%
	60.000	-	2,61074e-04%	2,17662e-04%	2,17608e-04%	1,46646e-04%
	100.000	-	1,96125e-04%	2,23708e-04%	2,24318e-04%	1,48526e-04%

Legenda: Percentagem média de triângulos com $\upsilon < \eta$ das dez iteraçoes temporais.

que o aumento de β provoca um maior crescimento do gradiente da malha.

Em um modelo ideal, os vértices são movidos de forma rápida, até que haja degeneração da malha, dentro do limite η pré-estabelecido pelo usuário. A função que mais se aproximou de um movimento ideal, utilizando-se $\beta < 0,6$, foi a função Γ, que apresentou o menor número de iterações das linhas 4 a 16 do algoritmo (10), sendo que o ideal são dez iterações, uma para cada etapa temporal, conforme pode-se observar nas tabelas (4) a (9). Para $\beta = 0,6$, na maioria dos experimentos houve dez iterações. Utilizando-se valores $\beta < 0,6$, ocorreram movimentos curtos dos vértices até ocorrer a degeneração da malha.

4.2 Organização e discussão dos resultados

Nos gráficos (12) a (21), pontua-se as funções monitoras em função dos resultados apresentados nas tabelas (4) a (15). A função com melhor

Tabela 15 Percentagem média das dez iteraçoes temporais do crescimento de γ_{mean}

β	χ	Λ	Γ	Ξ	Ψ	Υ
0,1	15.000	2,84444e-03%	1,85680e-03%	7,45447e-04%	7,30758e-04%	6,80480e-04%
	30.000	1,37007e-03%	1,42728e-03%	8,05840e-04%	7,37295e-04%	5,23861e-04%
	60.000	7,54116e-04%	1,03740e-03%	4,88688e-04%	5,18231e-04%	6,65238e-04%
	100.000	3,18040e-04%	8,24646e-04%	4,23258e-04%	4,26745e-04%	7,41664e-04%
0,2	15.000	3,01488e-03%	2,42703e-03%	7,76677e-04%	7,71896e-04%	4,02324e-04%
	30.000	1,90341e-03%	1,68887e-03%	8,11502e-04%	8,12612e-04%	8,63099e-04%
	60.000	8,13948e-04%	1,20654e-03%	5,36035e-04%	5,36355e-04%	7,42305e-04%
	100.000	4,36677e-04%	9,07735e-04%	4,71107e-04%	4,71853e-04%	9,34939e-04%
0,3	15.000	2,62197e-03%	2,58248e-03%	9,11691e-04%	9,13738e-04%	2,17064e-04%
	30.000	1,46262e-03%	1,89507e-03%	9,29914e-04%	9,31518e-04%	5,89346e-04%
	60.000	8,50141e-04%	1,26327e-03%	6,36968e-04%	6,38229e-04%	8,74091e-04%
	100.000	4,86621e-04%	1,15187e-03%	5,94349e-04%	5,94528e-04%	8,39555e-04%
0,4	15.000	4,39286e-03%	3,19536e-03%	9,82001e-04%	9,78266e-04%	7,52121e-04%
	30.000	2,08148e-03%	2,03709e-03%	1,03061e-03%	1,03269e-03%	6,08169e-04%
	60.000	9,32076e-04%	1,45870e-03%	7,03445e-04%	6,96073e-04%	8,39647e-04%
	100.000	5,54705e-04%	1,38669e-03%	6,87739e-04%	6,88083e-04%	9,37429e-04%
0,5	15.000	2,63251e-03%	3,71039e-03%	1,11534e-03%	1,13031e-03%	7,85852e-04%
	30.000	2,02566e-03%	2,35898e-03%	1,12605e-03%	1,12430e-03%	5,67102e-04%
	60.000	1,18176e-03%	1,52918e-03%	7,42224e-04%	7,42490e-04%	1,63156e-03%
	100.000	5,93194e-04%	1,42374e-03%	7,39369e-04%	7,39346e-04%	1,28194e-03%
0,6	15.000	-	3,99628e-03%	8,66991e-04%	8,68324e-04%	2,33332e-03%
	30.000	-	2,69575e-03%	7,27805e-04%	7,25817e-04%	2,12933e-03%
	60.000	-	1,62811e-03%	3,31236e-04%	3,32032e-04%	2,11477e-03%
	100.000	-	1,27850e-03%	3,47038e-04%	3,48505e-04%	1,64991e-03%

Legenda: Percentagem média das dez iteraçoes temporais do crescimento de γ_{mean} após movimentar os vértices.

resultado no critério analisado, recebe o pontuação “5”, e a função com pior resultado, recebe pontuação “1”.

Nos gráficos (12) a (21), tem-se a pontuação das funções monitoras para os experimentos configurados com $\chi = 15.000$, $\chi = 30.000$, $\chi = 60.000$, e $\chi = 100.000$, com $\beta = 0,1$, $\beta = 0,2$, $\beta = 0,3$, $\beta = 0,4$, $\beta = 0,5$ e $\beta = 0,6$.

No gráfico (12), tem-se a pontuação das funções monitoras conforme o total de vértices no fim da execução; no gráfico (13), conforme o total de *clocks* para execução total do experimento; no gráfico (14), conforme o total de *clocks* para movimentar os vértices; no gráfico (15), conforme o valor de γ_{max} da malha no fim do experimento; no gráfico (16), conforme o valor de γ_{mean} da malha no fim do experimento; no gráfico (17), conforme o valor médio de ρ_{max} após cada iteração temporal; no gráfico (18), conforme a percentagem média de triângulos com $\rho > \rho_{\alpha}$; no gráfico (19), conforme

o valor médio de υ_{min}; no gráfico (20), conforme a percentagem média de triângulos com $\upsilon > \eta$ e, por fim, no gráfico (21), pontua-se de acordo com o crescimento de γ_{mean}. Supondo-se que todos os itens tivessem o mesmo peso, a função Λ obteria os melhores resultados.

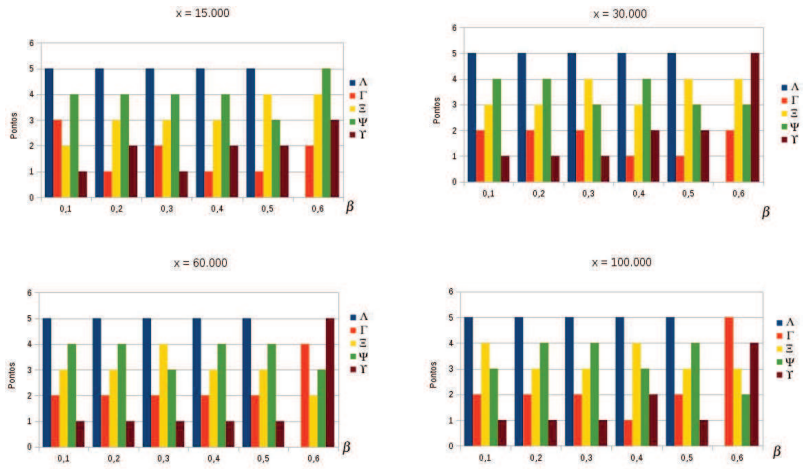

Figura 12 Pontuação total de vértices
Legenda: Pontuação referente ao total de vértices para malhas com $\chi = 15.000$, $\chi = 30.000$, $\chi = 60.000$ e $\chi = 100.000$, com $\beta = 0,1$, $\beta = 0,2$, $\beta = 0,3$, $\beta = 0,4$, $\beta = 0,5$ e $\beta = 0,6$.

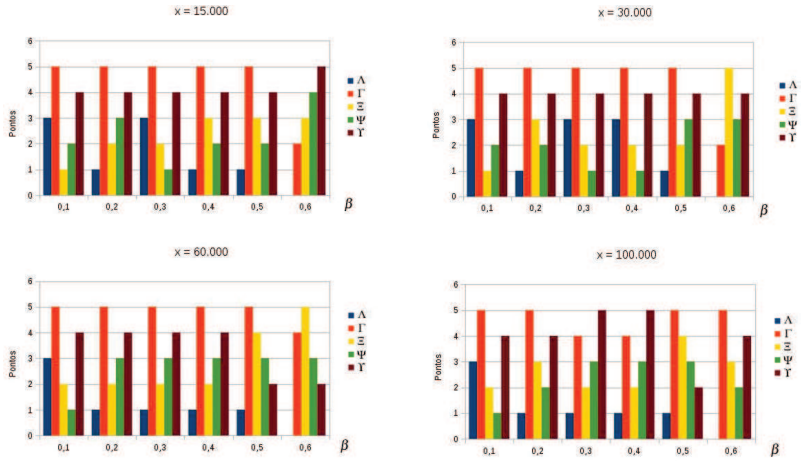

Figura 13 Pontuação total de *clocks*
Legenda: Pontuação referente ao total de *clocks* para malhas com $\chi = 15.000$, $\chi = 30.000$, $\chi = 60.000$ e $\chi = 100.000$, com $\beta = 0,1$, $\beta = 0,2$, $\beta = 0,3$, $\beta = 0,4$, $\beta = 0,5$ e $\beta = 0,6$.

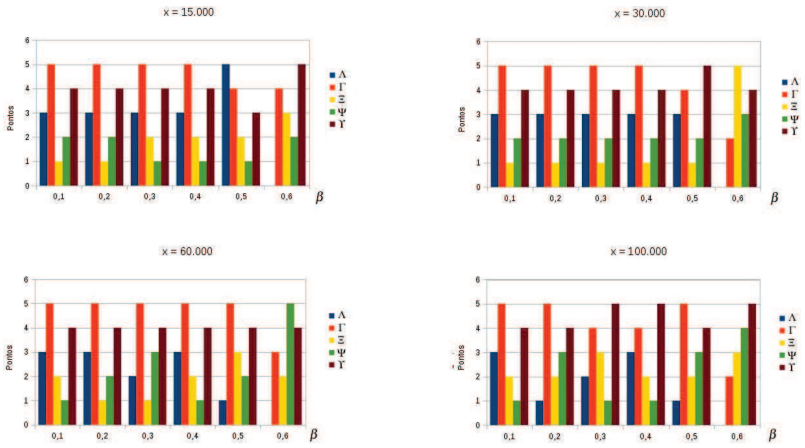

Figura 14 Pontuação total de *clocks* para movimentar
Legenda: Pontuação referente ao total de *clocks* para movimentar os vértices, para malhas com $\chi = 15.000$, $\chi = 30.000$, $\chi = 60.000$ e $\chi = 100.000$, com $\beta = 0,1$, $\beta = 0,2$, $\beta = 0,3$, $\beta = 0,4$, $\beta = 0,5$ e $\beta = 0,6$.

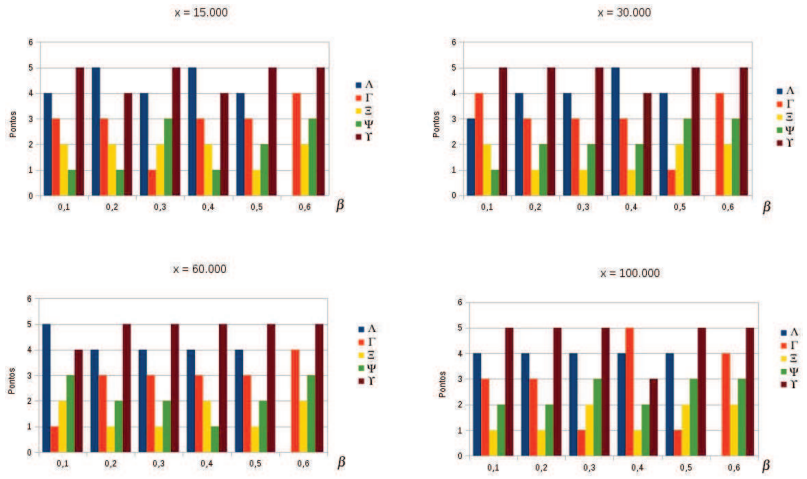

Figura 15 Pontuação γ_{max}
Legenda: Pontuação referente ao valor de γ_{max}, para malhas com $\chi = 15.000$, $\chi = 30.000$, $\chi = 60.000$ e $\chi = 100.000$, com $\beta = 0,1$, $\beta = 0,2$, $\beta = 0,3$, $\beta = 0,4$, $\beta = 0,5$ e $\beta = 0,6$.

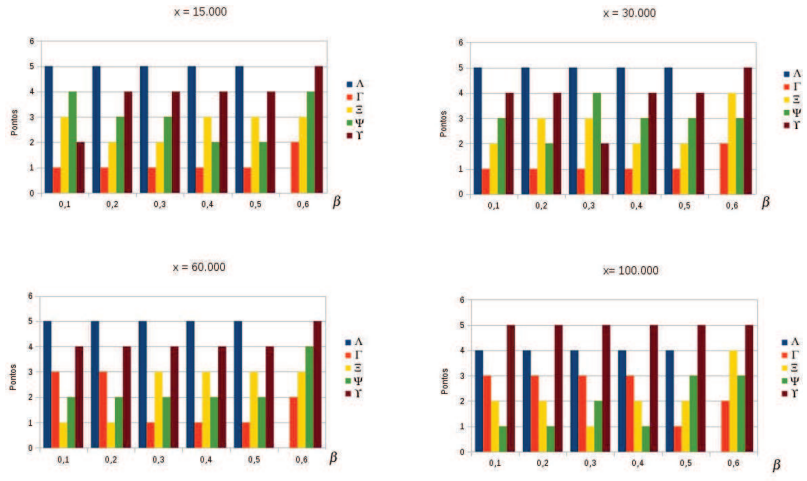

Figura 16 Pontuação γ_{mean}
Legenda: Pontuação referente ao valor de γ_{mean}, para malhas com $\chi = 15.000$, $\chi = 30.000$, $\chi = 60.000$ e $\chi = 100.000$, com $\beta = 0,1$, $\beta = 0,2$, $\beta = 0,3$, $\beta = 0,4$, $\beta = 0,5$ e $\beta = 0,6$.

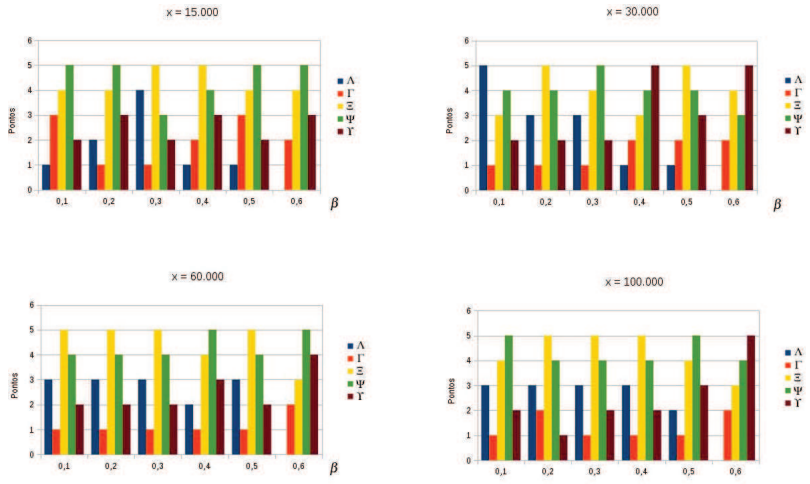

Figura 17 Pontuação ρ_{max}
Legenda: Pontuação referente ao valor de ρ_{max}, para malhas com $\chi = 15.000$, $\chi = 30.000$, $\chi = 60.000$ e $\chi = 100.000$, com $\beta = 0,1$, $\beta = 0,2$, $\beta = 0,3$, $\beta = 0,4$, $\beta = 0,5$ e $\beta = 0,6$.

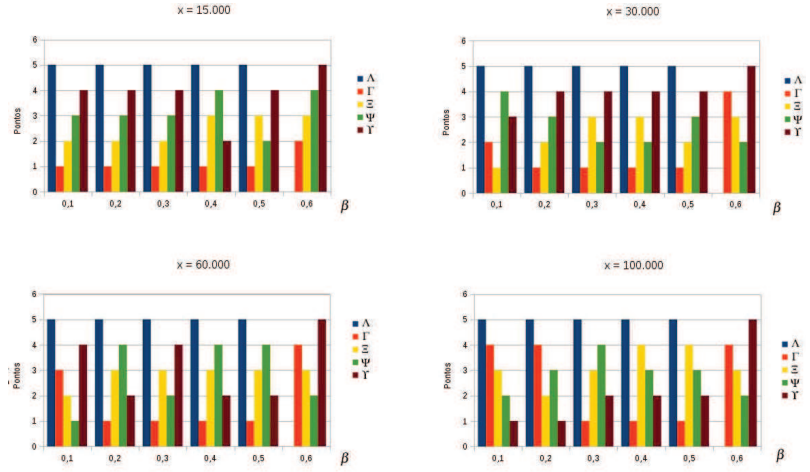

Figura 18 Pontuação triângulos com $\rho > \rho_\alpha$
Legenda: Pontuação referente à porcentagem de triângulos com $\rho > \rho_\alpha$, para malhas com $\chi = 15.000$, $\chi = 30.000$, $\chi = 60.000$ e $\chi = 100.000$, com $\beta = 0,1$, $\beta = 0,2$, $\beta = 0,3$, $\beta = 0,4$, $\beta = 0,5$ e $\beta = 0,6$.

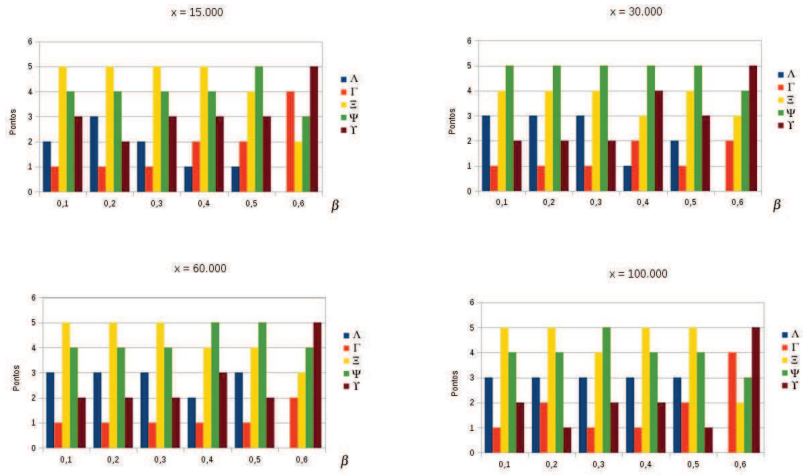

Figura 19 Pontuação υ_{min}
Legenda: Pontuação referente ao valor de υ_{min}, para malhas com $\chi = 15.000$, $\chi = 30.000$, $\chi = 60.000$ e $\chi = 100.000$, com $\beta = 0,1$, $\beta = 0,2$, $\beta = 0,3$, $\beta = 0,4$, $\beta = 0,5$ e $\beta = 0,6$.

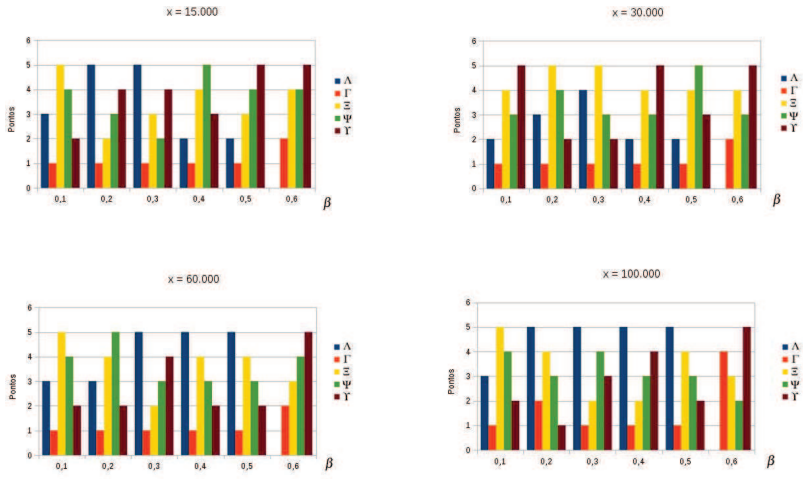

Figura 20 Pontuação triângulos com $\upsilon < \eta$
Legenda: Pontuação referente à porcentagem de triângulos com $\upsilon < \eta$, para malhas com $\chi = 15.000$, $\chi = 30.000$, $\chi = 60.000$ e $\chi = 100.000$, com $\beta = 0,1$, $\beta = 0,2$, $\beta = 0,3$, $\beta = 0,4$, $\beta = 0,5$ e $\beta = 0,6$.

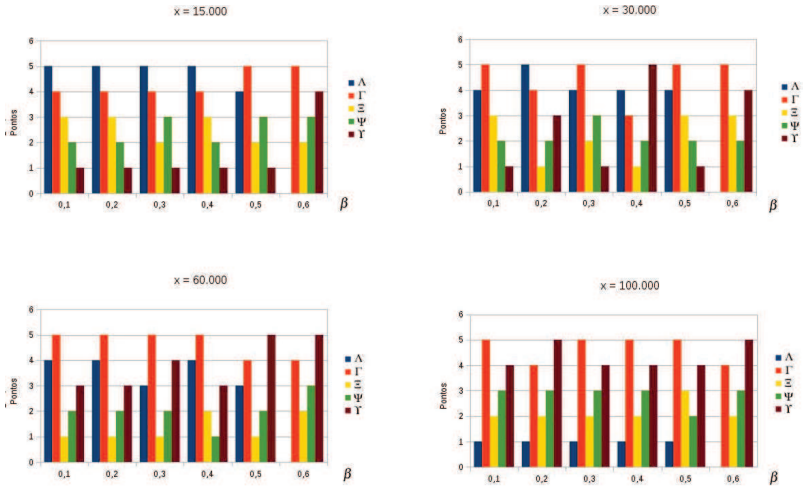

Figura 21 Pontuação crescimento de γ_{mean}

Legenda: Pontuação referente à porcentagem de crescimento de γ_{mean}, para malhas com $\chi = 15.000$, $\chi = 30.000$, $\chi = 60.000$ e $\chi = 100.000$, com $\beta = 0,1$, $\beta = 0,2$, $\beta = 0,3$, $\beta = 0,4$, $\beta = 0,5$ e $\beta = 0,6$.

5 CONCLUSÃO E PROPOSTAS DE TRABALHOS FUTUROS

Neste trabalho, foi proposta uma nova função monitora, chamada de Λ, e comparou-se os resultados obtidos com os resultados de outras funções monitoras encontradas na literatura.

Classificaram-se as funções monitoras de acordo com cada critério analisado. Com os resultados obtidos, pode-se verificar que cada função monitora possui suas qualidades e deficiências. A função Γ mostrou-se eficiente por apresentar movimentos rápidos, com baixo custo computacional. No entanto, também apresentou a maior degeneração da malha, com os piores valores das métricas *Circunradius-to-shortest Edge Radio* (CER) e *Shape Regularity Quality* (SRQ), a maior percentagem de triângulos com as métricas CER e SRQ consideradas ruins, também necessitando de mais vértices para correção da malha. A funçao Λ foi a que apresentou valores razoáveis, em comparação com as demais, do maior valor e do valor médio do gradiente, denotados, respectivamente, γ_{max} e γ_{mean}. Porém, como deficiência, a função Λ demandou maior custo computacional para execução.

Os experimentos com as funções Ξ e Ψ confirmaram que a principal qualidade dessas funções é a melhoria da qualidade da malha. Apresentaram os melhores resultados em relação às métricas CER e SRQ. Também apresentaram movimentos suaves, não ocorrendo grandes variações ao se utilizar diferentes parâmetros de adaptatividade, definido por β. Como deficiência, apresentaram baixos valores de γ_{max} e γ_{mean}, ao se comparar com uma malha sem movimentar os vértices. Até a publicação deste trabalho, não foram encontrados outros trabalhos em que se utilizassem as funções de Kobbelt et al. (1998) e Taubin (1995a, 1995b) a fim de obter melhorias na aproximação da solução de equação diferencial parcial.

A função Υ apresentou valores razoáveis de γ_{max} e γ_{mean}, com custo computacional relativamente baixo. Entretanto, em vários experimentos utilizando essa função, as malhas apresentaram a maior quantidade de vértices em relação às demais.

Para o usuário que deseja movimento dos vértices da malha, com altos valores de γ_{max} e γ_{mean}, sem necessidade de inclusão de vértices para melhoria da malha, a função Λ pode ser a melhor alternativa. Ainda, para o usuário que deseja boa qualidade da malha nas métricas CER e SRQ, as funções Ξ e Ψ podem ser excelentes alternativas.

Para o usuário que deseja custo computacional baixo e gradientes razoáveis na malha, a função Γ é uma boa alternativa em relação às funções Λ e Υ, desde que não haja problemas na inclusão de vértices para melhoria na malha.

A função Υ é interessante para usuários que desejam altos valores de γ_{max}, com taxa de crescimento do gradiente relativamente baixa, sem se importar com a inclusão de vértices para melhoria da qualidade da malha.

É importante salientar que o movimento dos vértices não deve ser utilizado sem análise e planejamento. Em todos os experimentos, ocorreu que movimentar os vértices acarretou um custo computacional maior que realizar somente o refinamento da malha. É possível que o custo de movimentar os vértices possa ser reduzido aumentando-se a intensidade do movimento e, consequentemente, reduzindo o número de iterações das linhas 4-16 do algoritmo (10). Os testes com "S/Mov.2" resultam em malhas com a qualidade mínima estipulada pelo usuário. Isso significa que os testes com "S/Mov.2" resultam em malhas de qualidade. Em todos os testes realizados, "S/Mov.2" resultou em menor custo computacional.

Como melhoria neste trabalho, seria importante realizar pesquisas

sobre estimadores de erros *a posteriori*, equação do calor com termo fonte ou mesmo a aplicação em outra equação para análise e verificação da solução gerada.

Ainda, como proposta de trabalho futuro, seria a adaptação para discretizações de volumes finitos dos esquemas de movimentos de malhas de discretizações de elementos finitos baseados em Huang e Russell (2011). Ainda, os esquemas propostos por Springel (2009) podem ser boas alternativas para a implementação de malhas móveis para se manter a qualidade da malha com baixo custo computacional na solução de equações diferenciais parciais.

Com o intuito de se reduzir o tempo de processamento dos experimentos, bem como obter uma solução com melhor aproximação e em tempo hábil, também necessita-se paralelizar o método dos gradientes conjugados, método que mais demanda recursos de processamento.

Com relação à malha, pode-se realizar experimentos com diferentes valores de fronteira e/ou com um domínio com geometria complexa. Também é possível obter melhorias nos resultados aumentando o parâmetro de adaptatividade β, ou mesmo, utilizando um β local.

REFERÊNCIAS

ASKES, H. **Advanced spatial discretisation strategies for localised failure:** mesh adaptivity and meshless methods. 2000. 168 p. Thesis (Ph.D. in Aerospace Engineering) - Technische Universiteit Delft, Delft, 2000. Disponível em: <http://repository.tudelft.nl/view/ir/uuid3A1f6cd5e6-08ad-4f0f-ba40-ea51ee0f4c16/>. Acesso em: 23 abr. 2013.

BANK, R. E.; SMITH, R. K. Mesh smoothing using a posteriori error estimates. **SIAM Journal on Numerical Analysis**, Philadelphia, v. 34, n. 3, p. 979-997, 1997.

BECKETT, G. et al. Computational solution of two-dimensional unsteady PDEs using moving mesh methods. **Journal of Computational Physics**, Orlando, v. 182, n. 2, p. 478-495, 2002. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0021999102971792>. Acesso em: 23 abr. 2013.

BELYAEV, A.; OHTAKE, Y. A comparison of mesh smoothing methods. In: ISRAEL-KOREA BI-NATIONAL CONFERENCE ON GEOMETRIC MODELING AND COMPUTER GRAPHICS, 1., 2003, Tel-Aviv. **Proceedings…** Tel-Aviv: Tel Aviv University, 2003. p. 83-87.

BOOR, C. de. **Good approximation by splines with variables knots II**. Berlin: Springer, 1973. (Lecture Notes Series, 363). Disponível em: <ftp://ftp.cs.wisc.edu/Approx/goodappr.pdf>. Acesso em: 30 abr. 2013.

BREWER, M. et al. The mesquite mesh quality improvement toolkit. In: INTERNATIONAL MESHING ROUNDTABLE, 12., 2003, Santa Fe. **Proceedings…** Santa Fe: Sandia National Laboratories, 2003. p. 239-250. Disponível em: <http://www.imr.sandia.gov/papers/abstracts/Br280.html>. Acesso em: 30 abr. 2013.

BRUCHON, J.; DIGONNET, H.; COUPEZ, T. Using a signed distance function for the simulation of metal forming processes: formulation of the contact condition and mesh adaptation: from a Lagrangian approach to an Eulerian approach. **International Journal for Numerical Methods in Engineering**, Chichester, v. 78, n. 8, p. 980-1008, May 2009. Disponível em: <http://hal-emse.ccsd.cnrs.fr/emse-00475556>. Acesso em: 30 abr. 2013.

BUDD, C. J.; HUANG, W.; RUSSELL, R. D. Adaptivity with moving grids. **Acta Numerica**, Cambridge, v. 18, n. 18, p. 111-241, Sept. 2009.

BURDEN, R. L.; FAIRES, J. D. **Numerical analysis**. 9th ed. Boston: Thomson Brooks, 2010. 888 p.

CHAGAS, G. O.; OLIVEIRA, S. L. G. de. **Uma verificação de desempenho do método dos gradientes conjugados com redução de banda pelo método Cuthill-Mckee reverso com vértice pseudo-periférico pelo algoritmo de George-Liu**. 2013. 51 p. Monografia (Graduação em Ciência da Computação) - Universidade Federal de Lavras, Lavras, 2013.

CHEN, L.; XU, J. Optimal Delaunay triangulations. **Journal of Computational Mathematics**, Beijing, v. 22, n. 2, p. 299-308, 2004.

COMPÉRE, G. et al. A mesh adaptation framework for dealing with large deforming meshes. **International Journal for Numerical Methods in Engineering**, New York, v. 82, n. 7, p. 843-867, 2010.

COURANT, R.; FRIEDRICHS, K.; LEWY, H. Über die partiellen differenzengleichungen der mathematischen physik. **Mathematische Annalen**, Berlin, v. 100, n. 1, p. 32-74, 1928.

CUTHILL, E.; MCKEE, J. Reducing the bandwidth of sparse symmetric matrices. In: NATIONAL CONFERENCE, 24., 1969, New York. **Proceedings…** New York: ACM, 1969. p. 157-172.

DAM, A. van; ZEGELING, P. A. A robust moving mesh finite volume method applied to 1d hyperbolic conservation laws from magneto hydrodynamics. **Journal of Computational Physics**, San Diego, v. 216, n. 2, p. 526-546, Aug. 2006.

DELAUNAY, B. **Sur la sphère vide, izvestia akademii nauk sssr, otdelenie matematicheskikh i estestvennykh nauk**. Moscow: Zvestiia Akademii Nauk SSSR, 1939. 7 p. (Seriia Biologicheskaia).

DESBRUN, M. et al. Implicit fairing of irregular meshes using difusion and curvature ow. In: ANNUAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, 26., 1999, New York. **Proceedings…** New York: ACM; A. Wesley, 1999. p. 317-324.

DIACHIN, L. F. et al. A comparison of two optimization methods for mesh quality improvement. **Engineering with Computers**, London, v. 22, n. 2, p. 61-74, 2006.

DOBRZYNSKI, C.; FREY, P. Anisotropic Delaunay mesh adaptation for unsteady simulations. In: INTERNATIONAL MESHING ROUNDTABLE, 17., 2008, Berlin. **Proceedings...** Berlin: Springer, 2008. p. 177-194. Disponível em: <http://www.ljll.math.upmc.fr/frey/publications/imrDF2.pdf>. Acesso em: 23 abr. 2013.

DUFFELL, P. C.; MACFADYEN, A. I. TESS: a relativistic hydrodynamics code on a moving voronoi mesh. **Astrophysical Journal Supplement Series**, Chicago, v. 197, p. 15, Dec. 2011. Disponível em: <http://adsabs.harvard.edu/abs/2011ApJS..197...15D>. Acesso em: 23 abr. 2013.

DWYER, H.; RAISZADEH, F.; OTEY, G. A study of reactive difusion problems with stiff integrators and adaptive grids. In: INTERNATIONAL CONFERENCE ON NUMERICAL METHODS IN FLUID DYNAMICS, 17., 1981, Berlin. **Proceedings…** Berlin: Springer, 1981. p. 170-175. (Lecture Notes in Physics, 141).

DWYER, H.; SANDERS, B.; KEE, R. **An adaptive grid method for problems in uid mechanics and heat transfer**. Williamsburg: American Institute of Aeronautics and Astronautics, 1979. 203 p. Disponível em: <http://opac.inria.fr/record=b1070874>. Acesso em: 20 abr. 2013.

ELEFTHERIOU, T. **Moving mesh methods using monitor functions for the porous medium equation**. 2011. 60 p. Thesis (Ph.D. in Matematica) - University of Reading, Reading, 2011.

FOUSSE, L. et al. MPFR: a multiple-precision binary oating-point library with correct rounding. **ACM Transactions on Mathematical Software**, New York, v. 33, n. 2, p. 13:1-13:15, June 2007.

FREITAG, L. A. On combining laplacian and optimization-based mesh smoothing techniques. In: _____. **Trends in unstructured mesh generation**. Washington: AMD Trends in Unstructured Mesh Generation, 1997. p. 37-43. (ASME, 220).

FREITAG, L. A.; JONES, M.; PLASSMANN, P. E. **A parallel algorithm for mesh smoothing**. Disponível em: <http://dblp.uni-trier.de/db/conf/ppsc/ppsc1997.html>. Acesso em: 10 dez. 1997.

FREY, P. J.; GEORGE, P. L. (Ed.). **Mesh generation**. 2nd ed. London: ISTE, 2008. 848 p.

GEORGE, A.; LIU, J.; NG, E. **Computer solution of sparse linear systems**. Illinois: University of Illinois, 1994. Disponível em: <http://www.cs.illinois.edu/n~heath/courses/cs598mh/georgeliu.pdf>. Acesso em: 10 dez. 2013.

GEORGE, A.; LIU, J. W. H. An implementation of a pseudoperipheral node finder. **ACM Transactions on Mathematical Software**, New York, v. 5, n. 3, p. 284-295, Sept. 1979.

GEORGE, J. A. **Computer implementation of the finite element method**. 1971. 222 p. Thesis (Ph.D. in Computer Science) - Stanford Univertity, Stanford, 1971.

GNOFFO, P. Complete supersonic flowfields over blunt bodies in a generalized orthogonal coordinate system. **American Institute of Aeronautics and Astronautics**, Reston, v. 18, n. 6, p. 611-612, 1980.

GRANLUND, T. **TEAM the GMP development:** GNU MP, the GNU multiple precision arithmetic library. Version 5.0.5. New York, 2012. Disponível em: <http://gmplib.org/>. Acesso em: 10 abr. 2013.

GREEN, P. J.; SIBSON, R. Computing dirichlet tessellations in the plane. **The Computer Journal**, London, v. 21, n. 2, p. 168-173, 1978. Disponível em: <http://comjnl.oxfordjournals.org/content/21/2/168.abstract>. Acesso em: 23 abr. 2013.

GREIF, T. H. et al. Simulations on a moving mesh: the clustered formation of population III protostars. **The Astrophysical Journal**, Chicago, v. 737, n. 2, p. 75-80, 2011. Disponível em: <http://stacks.iop.org/0004-637X/737/i=2/a=75>. Acesso em: 23 abr. 2013.

HAWKEN, D.; GOTTLIEB, J.; HANSEN, J. Review of some adaptive node-movement techniques in finite-element and finite-diference solutions of partial diferential equations. **Journal of Computational Physics**, Orlando, v. 95, n. 2, p. 254-302, 1991. Disponível em: <http://www.sciencedirect.com/science/article/pii/002199919190277R>. Acesso em: 23 abr. 2013.

HEB, S.; SPRINGEL, V. Gas stripping and mixing in galaxy clusters: a numerical comparison study. **Monthly Notices of the Royal Astronomical Society**, Oxford, v. 426, p. 3112-3134, Nov. 2012. Disponível em: <http://arxiv.org/abs/1208.0351>. Acesso em: 23 abr. 2013.

HEB, S.; SPRINGEL, V. Particle hydrodynamics with tessellation techniques. **Monthly Notices of the Royal Astronomical Society**, Oxford, v. 406, p. 2289-2311, Aug. 2010. Disponível em: <http://adsabs.harvard.edu/abs/2010MNRAS.406.2289H>. Acesso em: 23 abr. 2013.

HESTENES, M. R.; STIEFEL, E. Methods of conjugate gradients for solving linear systems. **Journal of Research of the National Bureau of Standards**, Washington, v. 49, p. 409-436, Dec. 1952.

HUANG, W.; REN, Y.; RUSSELL, R. D. Moving mesh partial diferential equations (mmpdes) based on the equidistribution principle. **SIAM Journal on Numerical Analysis**, Philadelphia, v. 31, n. 3, p. 709-730, June 1994.

HUANG, W.; RUSSELL, R. **Adaptive moving mesh methods**. New York: Springer, 2011. 432 p. (Applied Mathematical Sciences).

HUANG, W.; RUSSELL, R. D. Moving mesh strategy based on a gradient ow equation for two-dimensional problems. **SIAM Journal on Scientific Computing**, Philadelphia, v. 20, n. 3, p. 998-1015, Jan. 1999.

HUANG, W.; SUN, W. Variational mesh adaptation II: error estimates and monitor functions. **Journal of Computational Physics**, Orlando, v. 184, n. 2, p. 619-648, 2003. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0021999102000402>. Acesso em: 23 abr. 2013.

JONES, R. E. A self-organizing mesh generation program. **Journal of Pressure Vessel Technology**, New York, v. 96, n. 3, p. 193-199, Aug. 1974.

KIM, B.; ROSSIGNAC, J. **Localized bi-laplacian solver on a triangle mesh and its applications**. Atlanta: Georgia Institute of Technology, 2004. 11 p. (GVU Technical Report).

KNUPP, P. M. Algebraic mesh quality metrics. **SIAM Journal of Science Computer**, Philadelphia, v. 23, n. 1, p. 193-218, Jan. 2001.

KOBBELT, L. et al. Interactive multi-resolution modeling on arbitrary meshes. In: ANNUAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, 25., 1998, New York. **Proceedings…** New York: ACM, 1998. p. 105-114.

LANGE, C.; POLTHIER, K. **Anisotropic smoothing of point sets**. Amsterdam: Elsevier Science, 2005. 692 p.

LAWSON, C. L. **Software for C1 surface interpolation**. Washington: National Aeronautics and Space Administration, 1977. 194 p. (Matematical Software, 3).

LI, R.; TANG, T.; ZHANG, P. Moving mesh methods in multiple dimensions based on harmonic maps. **Journal of Computational Physics**, Orlando, v. 170, n. 2, p. 562-588, July 2000. Disponível em: <http://www.math.hkbu.edu.hk/~ttang/moving.html>. Acesso em: 23 abr. 2013.

LITTLEFIELD, D. L. The use of r-adaptivity with local, intermittent remesh for modeling hypervelocity impact and penetration. **International Journal of Impact Engineering**, Oxford, v. 26, n. 1/10, p. 433-442, 2001. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0734743X01000938>. Acesso em: 23 abr. 2013.

LIU, C.; SHEN, J. A phase field model for the mixture of two incompressible uids and its approximation by a fourier-spectral method. **Physica D: Nonlinear Phenomena**, Amsterdam, v. 179, n. 3/4, p. 211-228, 2003. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167278903000307>. Acesso em: 23 abr. 2013.

LÖHNER, R. et al. The numerical simulation of strongly unsteady ow with hundreds of moving bodies. **International Journal for Numerical Methods in Fluids**, New York, v. 31, n. 1, p. 113-120, 1999.

MACKENZIE, J. Moving mesh finite volume methods for one-dimensional evolutionary partial differential equations. In: _____. **Strathclyde mathematics research report**. Glasgow: University of Strathclyde, 1996. p. 26.

MACKENZIE, J.; ROBERTSON, M.; BECKETT, M. An r-adaptive finite element method for the solution of the two-dimensional phase-field equations. **Communications in Computational Physics**, New York, v. 1, n. 5, p. 805-826, 2006. Disponível em: <http://www.global-sci.com/issue/contents/1/issue5.html>. Acesso em: 23 abr. 2013.

MALISKA, C. R. **Transferência de calor e mecânica dos fluidos**. 2. ed. Rio de Janeiro: LTC, 2010. 472 p.

MARLOW, R. **Moving mesh methods for solving parabolic partial differential equations**. 2010. 146 p. Thesis (Ph.D. in School of Computing) - Universit of Leeds, Leeds, 2010.

MARLOW, R. Moving mesh methods for solving parabolic partial diferential equations. **School of Computing**, Leeds, v. 46, n. 1, p. 353-361, July 2011.

MARON, J. L.; MCNALLY, C. P.; MACLOW, M. M. Phurbas: an adaptive, lagrangian, meshless, magnetohydrodynamics code: I., algorithm. **Astrophysical Journal Supplement Series**, Chicago, v. 200, p. 6, May 2012. Disponível em: <http://adsabs.harvard.edu/abs/2012ApJS.200.6M>. Acesso em: 24 abr. 2013.

MCNALLY, C. P.; LYRA, W.; PASSY, J. C. A well-posed kelvin-helmholtz instability test and comparison. **The Astrophysical Journal Supplement Series**, Chicago, v. 201, n. 2, p. 18, Aug. 2012. Disponível em: <http://adsabs.harvard.edu/abs/2012ApJS.201.18M>. Acesso em: 23 abr. 2013.

MUÑOZ, D. et al. **Multi-dimensional, compressible viscous flow on a moving voronoi mesh**. Disponível em: <http://arxiv.org/abs/1203.1037>. Acesso em: 17 dez. 2012.

NIEDU, G. da S. **Um método para solução de sistemas lineares através do gradiente conjugado cooperativo**. 2012. 117 p. Dissertação (Mestrado em Engenharia Elétrica) - Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2012. Disponível em: <http://objdig.ufrj.br/60/teses/coppen m/GuilhermeDaSilvaNiedu.pdf>. Acesso em: 10 dez. 2013.

OHTAKE, Y.; BELYAEV, A. G.; BOGAEVSKI, I. A. Mesh regularization and adaptive smoothing. **Computer Aided Design**, Guildford, v. 33, n. 11, p. 789-800, 2001. Disponível em: <http://dblp.uni-trier.de/db/journals/cad/cad33.html>. Acesso em: 20 abr. 2013.

OHTAKE, Y.; BELYAEV, A. G.; BOGAEVSKI, I. A. Polyhedral surface smoothing with simultaneous mesh regularization. In: GEOMETRIC MODELING AND PROCESSING, 1., 2000, Washington. **Proceedings…** Washington: IEEE Computer Society, 2000. p. 229-237. Disponível em: <http://dblp.unitrier.de/db/conf/gmp/gmp2000.html>. Acesso em: 10 dez. 2013.
OLIVEIRA, F. S. et al. Malhas móveis para solução numérica de equações diferenciais parciais. **Revista de Sistemas de Informação da Faculdade Salesiana Maria Auxiliadora**, Macaé, v. 11, p. 11-16, jun. 2013. Disponível em: <http://www.fsma.edu.br/si/principal.html>. Acesso em: 10 dez. 2013.

OLIVEIRA, S. L. G. de. A review on Delaunay refinement techniques. In: INTERNATIONAL CONFERENCE ON COMPUTATIONAL SCIENCE AND ITS APPLICATIONS, 12., 2012, Berlin. **Proceedings…** Berlin: Springer-Verlag, 2012. v. 1, p. 172-187.

OLIVEIRA, T. H.; OLIVEIRA, S. L. G. de. **Uma aplicação de volumes finitos com refinamento de Ruppert e malhas de qualidade por off-centers**. Lavras: UFLA, 2012. Pesquisa de iniciação científica.

OLIVEIRA, T. H.; OLIVEIRA, S. L. G. de. Uma resolução da equação de Laplace por volumes finitos e diagrama de Voronoi com refinamento adaptativo e de Ruppert. In: INTERNATIONAL CONFERENCE ON GRAPHICS, PATTERNS AND IMAGES - WORKSHOP OF UNDERGRADUATE WORKS, 12., 2012, Ouro Preto. **Proceedings...** Ouro Preto: SBC, 2012. 1 CD-ROM.

OLIVIER, G.; ALAUZET, F. A new changing-topology ale scheme for moving mesh unsteady simulations. In: AIAA AEROSPACE SCIENCES MEETING AND EXHIBIT, 49., 2011, Le Chesnay. **Proceedings...** Le Chesnay: AIAA, 2011. p. 1-25. Disponível em: <https://www.rocq.inria.fr/gamma/gamma/Membres/CIPD/Frederic.Alauzet>. Acesso em: 23 abr. 2013.

PAKMOR, R.; BAUER, A.; SPRINGEL, V. Magnetohydrodynamics on an unstructured moving grid. **Institut für Theoretische Studien**, Heidelberg, v. 418, p. 1392-1401, Dez. 2011. Disponível em: <http://arxiv.org/abs/1108.1792>. Acesso em: 23 abr. 2013.

PÉBAY, P. P.; BAKER, T. J. Analysis of triangle quality measures. **Mathematics of Computation**, Boston, v. 72, n. 244, p. 1817-1839, Oct. 2003.
POPIOLEK, T. L.; AWRUCH, A. M. An adaptive mesh strategy for transient ows simulations. **Computational Modeling**, Los Alamitos, v. 1, n. 3, p. 71-76, Nov. 2009.

PREPARATA, F. P.; SHAMOS, M. I. **Computational geometry:** an introduction. New York: Springer Verlag, 1985. 398 p. (Scientific Computation).

RAJAGOPAL, A.; GANGADHARAN, R.; SIVAKUMAR, S. M. A performance study on configurational force and spring-analogy based mesh optimization schemes. **International Journal for Computational Methods in Engineering Science and Mechanics**, Philadelphia, v. 7, n. 4, p. 241-262, 2006. Disponível em: <http://www.tandfonline.com/doi/abs/10.1080/15502280500388169>. Acesso em: 10 dez. 2013.

RAJAGOPAL, A.; SIVAKUMAR, S. M. A combined r-h adaptive strategy based on material forces and error assessment for plane problems and biomaterial interfaces. **Computational Mechanics**, Berlin, v. 41, n. 1, p. 49-72, Mar. 2007.

RUPPERT, J. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. **Journal of Algorithms**, Duluth, v. 18, n. 3, p. 548-585, May 1995.

SCHNEIDER, R.; KOBBELT, L.; SEIDEL, H. P. Improved bi-laplacian mesh fairing. In: LYCHE, T.; SCHUMAKER, L. L. (Ed.). **Mathematical methods for curves and surfaces:** Oslo 2000. Oslo: Vanderbilt University, 2001. p. 445-454. (Innovations in Applied Mathematics, 1).

SHAMOS, M. I.; HOEY, D. Closest-point problems. In: ANNUAL SYMPOSIUM ON FOUNDATIONS OF COMPUTER SCIENCE, 16., 1975, Washington. **Proceedings…** Washington: IEEE Computer Society, 1975. p. 151-162. Disponível em: <http://dl.acm.org/citation.cfm?id=1398504.1382488>. Acesso em: 23 abr. 2013.

SHEWCHUK, J. R. **Delaunay refinement mesh generation**. 1997. 207 p. Thesis (Ph.D. in Computer Science) - Carnegie Mellon University, Pittsburgh, 1997.

SHONTZ, S. M.; VAVASIS, S. A. **A linear weighted laplacian smoothing framework for warping tetrahedral meshes:** CoRR, cs.NA/0410045. Disponível em: <http://dblp.uni-trier.de/db/journals/corr/corr0410.html csNA-0410045>. Acesso em: 2 dez. 2004.

SILVA, L. F. et al. **Connectivity oblivious merging of triangulations**. Disponível em: <http://dblp.uni-trier.de/db/conf/sibgrapi/sibgrapi2012.html>. Acesso em: 23 dez. 2012.

SONI, B. K. et al. Solution adaptive grid strategies based on point redistribution. **Computer Methods in Applied Mechanics and Engineering**, Amsterdam, v. 189, n. 4, p. 1183-1204, Sept. 2000.

SPRINGEL, V. The cosmological simulation code GADGET-2. **Monthly Notices of the Royal Astronomical Society**, Oxford, v. 364, p. 1105-1134, Dec. 2005. Disponível em: <http://adsabs.harvard.edu/abs/2005MNRAS.364.1105S>. Acesso em: 23 abr. 2013.

SPRINGEL, V. E pur si muove: Galiliean-invariant cosmological hydrodynamical simulations on a moving mesh. **Monthly Notices of the Royal Astronomical Society, Max-Planck-Institute for Astrophysics**, Oxford, v. 401, p. 791-851, Oct. 2009. Disponível em: <http://arxiv.org/abs/0901.4107>. Acesso em: 18 abr. 2013.

SPRINGEL, V. **Hydrodynamic simulations on a moving Voronoi mesh**. Garching: Springer, 2011. 35 p. Disponível em: <http://arxiv.org/abs/1109.2218>. Acesso em: 23 abr. 2013.

TAN, Z. Adaptive moving mesh methods for two-dimensional resistive magneto-hydrodynamic PDE models. **Computers & Fluids**, New York, v. 36, n. 4, p. 758-771, 2007. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0045793006000831>. Acesso em: 23 abr. 2013.

TAN, Z. et al. Moving mesh methods with locally varying time steps. **Journal of Computational Physics**, Orlando, v. 200, n. 1, p. 347-367, 2004. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0021999104001706>. Acesso em: 23 abr. 2013.

TAN, Z.; LIM, K. M.; KHOO, B. C. An adaptive mesh redistribution method for the incompressible mixture ows using phase-field model. **Journal of Computational Physics**, San Diego, v. 225, n. 1, p. 1137-1158, July 2007.

TAN, Z.; TANG, T.; ZHANG, Z. A simple moving mesh method for one-and two-dimensional phase-field equations. **Journal of Computational and Applied Mathematics**, Amsterdam, v. 190, n. 1, p. 252-269, June 2006.

TANG, T. **Moving mesh methods for computational fluid dynamics**. Hong Kong: Hong Kong Baptist University, 2005. 33 p. (Technical Report).

TANNEHILL, J.; ANDERSON, D.; PLETCHER, R. **Computational fluid mechanics and heat transfer**. 2nd ed. Washington: Taylor & Francis, 1997. 792 p. (Series in Computational and Physical Processes in Mechanics and Thermal Sciences).

TAUBIN, G. Curve and surface smoothing without shrinkage. In: INTERNATIONAL CONFERENCE ON COMPUTER VISION, 5., 1995, Washington. **Proceedings…** Washington: IEEE Computer Society, 1995. p. 852-857. Disponível em: <http://dl.acm.org/citation.cfm?id=839277.840029>. Acesso em: 10 dez. 2013.

TAUBIN, G. A signal processing approach to fair surface design. In: ANNUAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, 22., 1995, New York. **Proceedings…** New York: ACM, 1995. p. 351-358.

TAUBIN, G.; SHANG, T.; GOLUB, G. Optimal surface smoothing as filter desing. In: EUROPEAN CONFERENCE COMPUTER VISION, 4., 1996, London. **Proceedings...** London: Springer-Verlag, 1996. p. 283-292.

TENG, S. H.; WONG, C. W. Unstructured mesh generation: theory, practice, and perspectives. **International Journal of Computational Geometry and Applied**, Singapore, v. 10, n. 3, p. 227-266, 2000.

THOMPSON, J. F.; SONI, B. K.; WEATHERHILL, N. P. (Ed.). **Handbook of grid generation**. New York: CRC, 1999. 990 p.

TORO, E. F.; SPRUCE, M.; SPEARES, W. Restoration of the contact surface in the hll-riemann solver. **Shock Waves**, Berlin, v. 4, n. 1, p. 25-34, July 1994.

ÜNGÖR, A. Off-centers: a new type of Steiner points for computing size-optimal quality-guaranteed Delaunay triangulations. **Computational Geometry**, Amsterdam, v. 42, n. 2, p. 109-118, 2009. Disponível em: <http://www.sciencedirect.com/science/article/pii/S092577210800059X>. Acesso em: 23 abr. 2013.

ÜNGÖR, A. Off-centers: a new type of Steiner points for computing size-optimal quality-guaranteed Delaunay triangulations. In: FARACHCOLTON, M. (Ed.). **Theoretical informatics**. Berlin: Springer, 2004. p. 152-161. (Lecture Notes in Computer Science, 2976).

VARTZIOTIS, D.; WIPPER, J.; PAPADRAKAKIS, M. Improving mesh quality and finite element solution accuracy by fGETMeg smoothing in solving the poisson equation. **Finite Elements in Analysis and Design**, Amsterdam, v. 66, n. 1, p. 36-52, 2013. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0168874X12002077>. Acesso em: 10 dez. 2013.

VERSTEEG, H.; MALALASEKERA, W. **An Introduction to computational fluid dynamics:** the finite volume method. New York: Pearson Education, 2007. 503 p.

VOLLMER, J.; MENCL, R.; MÜLLER, H. Improved laplacian smoothing of noisy surface meshes. **Computer Graphics Forum**, Amsterdam, v. 18, n. 3, p. 131-138, 1999. Disponível em: <http://dblp.uni-trier.de/db/journals/cgf/cgf18.html>. Acesso em: 10 dez. 2013.

VORONOI, G. Nouvelles applications des paramètres continus à la théorie des formes quadratiques: deuxième mémoire, recherches sur les paralléllοèdres primitifs. **Journal für die Reine und Angewandte Mathematik**, De Gruyter, v. 1908, n. 134, p. 198-287, Jan. 1908.

WHITE, J. B. A. On the numerical solution of initial/boundary-value problems in one space dimension. **SIAM Journal on Numerical Analysis**, Philadelphia, v. 19, n. 4, p. 683-697, 1982. Disponível em: <http://www.jstor.org/stable/2157026>. Acesso em: 10 dez. 2013.

ZEGELING, P. A. **MFE revisited, part 1:** adaptive grid-generation using the heat equation. Utrecht: Utrecht University, 1996. Disponível em: <http://igitur-archive.library.uu.nl/math/2001-0703-151147/946.pdf>. Acesso em: 12 jun. 2013.

ZEGELING, P. A. Tensor-product adaptive grids based on coordinate transformations. **Journal of Computational and Applied Mathematics**, Amsterdam, v. 166, n. 1, p. 343-360, Apr. 2004.

ZEGELING, P. A. On resistive MHD models with adaptive moving meshes. **Journal of Scientific Computing**, New York, v. 24, n. 2, p. 263-284, Aug. 2005.

ZHANG, Y.; BAJAJ, C.; XU, G. Surface smoothing and quality improvement of quadrilateral/hexahedral meshes with geometric ow. In: INTERNATIONAL MESHING ROUNDTABLE, 14., 2005, Berlin. **Proceedings…** Berlin: Springer, 2005. p. 449-468.

ZHIJUN, T. **Moving mesh finite volume method and its applications**. 2005. 123 p. Thesis (Ph.D. in Matematica) - Hong Kong Baptist University, Hong Kong, 2005.