Uma resolução da equação de Laplace por Volumes Finitos e diagrama de Voronoi com refinamento adaptativo e de Ruppert

Talles Henrique de Oliveira, Sanderson L. Gonzaga de Oliveira
Departamento de Ciência da Computação
Universidade Federal de Lavras, UFLA
Lavras, Brasil
Email: talles.oliveira2@gmail.com, sanderson@dcc.ufla.br

***Abstract*—Este trabalho apresenta uma solução da equação de Laplace por volumes finitos utilizando o digrama de Voronoi, gerado a partir da triangulação de Delaunay. É utilizada a técnica de refinamento adaptativo de malhas conforme os conceitos do refinamento de Ruppert. Ainda, é utilizado o refinamento de Ruppert para que a malha resultante seja de boa qualidade.**

***Abstract*—This work shows a solution of the Laplace's equation by finite volume discretizations with the Voronoi diagram, generated from the Delaunay triangulation. We apply the adaptative mesh refinement technique together with concepts of the Ruppert refinement. Furthermore, we use the Ruppert refinement to improve the quality of the Delaunay triangulation.**

***Keywords*-Geometria computacional; geração de malhas; triangulação de Delaunay; refinemento de Delaunay; método dos volumes finitos;**

I. Introdução

Neste trabalho, são apresentadas simulações da utilização do diagrama de Voronoi [1], obtido por meio da triangulação de Delaunay [2]. Essas malhas são utilizadas na aplicação do método dos volumes finitos (MVF) na equação de Laplace, com condições de contorno de Dirichlet. O MVF é um método de discretização que vem sendo amplamente utilizado, em que [3], [4], [5] e [6] são exemplos. Este método é baseado em leis de conservação de massa.

Nas simulações descritas neste trabalho, utilizou-se a técnica de refinamento adaptativo de malhas para se obter menor custo computacional em relação a uma malha uniforme. As comparações para redução de custo computacional em relação a uma malha uniforme e com outras técnicas de refinamento adaptativo ainda não foram realizadas. Neste trabalho, apresentam-se as simulações iniciais da implementação. Para obtenção de malhas triangulares de qualidade, utilizou-se o refinamento de Ruppert [7].

Na seção seguinte são apresentados alguns trabalhos relacionados. Na seção III o diagrama de Voronoi e a triangulação de Delaunay são apresentados. Nessa mesma seção, são descritos o algoritmo de Lawson para geração de triangulações de Delaunay e o algoritmo de Ruppert. A implementação é explicada na seção IV. Na seção V são apresentados os resultados e, finalmente, na seção VI são apresentadas conclusões e trabalhos futuros.

II. Trabalhos relacionados

Vários trabalhos utilizando a triangulação de Delaunay, como também o diagrama de Voronoi, na aplicação de métodos numéricos podem ser encontrados na literatura. Como exemplo, em [8], é aplicado o método dos volumes finitos para simulação do espalhamento de um contaminante em um lençol freático com triangulações de Delaunay geradas a partir de um algoritmo de geração de malhas baseado em imagens. Esquemas para aplicação do MVF em malhas de Voronoi são apresentados em [9]. Um pacote para análise bidimensional por elementos finitos é descrito em [10], no qual geram-se automaticamente malhas de boa qualidade para problemas de campo magnético e utiliza um refinamento adaptativo que subdivide apenas elementos cujo erro estimado é considerado grande.

Em geração de malhas, muitos trabalhos foram desenvolvidos com o objetivo de produzir triangulações de Delaunay com a melhor qualidade possível. Entre eles, pode-se citar Shewchuk [11], que apresenta pequenas melhorias para o algoritmo de refinamento de Delaunay proposto por Chew [12], como também para o algoritmo de Ruppert [7]. Além disso, Shewchuk aborda o problema de modelar domínios com ângulos pequenos, apresentando seu próprio algoritmo. O algoritmo de Ruppert será brevemente descrito na seção III-C. Shewchuk [11] desenvolveu um programa que gera triangulações de Delaunay e diagramas de Voronoi, chamado Triangle [13], que utiliza o algoritmo de Ruppert.

III. Geração das malhas

Nesta aplicação do MVF, utilizam-se como volumes de controle os polígonos de Voronoi, limitados a um determinado domínio Ω. Seja $\delta(a, b)$ a distância euclidiana entre os pontos a e b. No plano euclidiano, o diagrama de Voronoi é a decomposição do plano em polígonos. Considere um conjunto $S = \{s_1, s_2, s_3, \cdots, s_n\}$ de pontos geradores do diagrama de Voronoi contidos em Ω. Um ponto qualquer $p \in \mathbb{R}^2$ está no polígono correspondente a um ponto gerador s_i do diagrama de Voronoi se e, somente se, $\delta(p, s_i) \leq \delta(p, s_j)$, $\forall s_j \in S$ com $j \neq i$.

A seguir, descreve-se a triangulação de Delaunay, que é o *dual* geométrico do diagrama de Voronoi. Isso significa que

a partir de uma triangulação de Delaunay é possível se obter um diagrama de Voronoi e vice-versa.

A. Triangulação de Delaunay

Pode-se definir uma triangulação de Delaunay utilizando o conceito de *aresta de Delaunay*. Um círculo c *qualquer* é *vazio* em relação a um conjunto de pontos, se ele não contém nenhum desses pontos em seu interior, mas pontos são permitidos na fronteira do círculo. Uma aresta ligando dois pontos $s_1, s_2 \in S$ é dita ser de Delaunay se e, somente se, existe um círculo vazio c que passa por s_1 e s_2. Uma triangulação em que todas as arestas são de Delaunay é uma triangulação de Delaunay. Uma aresta é *localmente* de Delaunay ao se considerar apenas os vértices dos triângulos que compartilham tal aresta. Se todas as arestas de uma triangulação são localmente de Delaunay, então, todas são de Delaunay.

Uma outra maneira, talvez um pouco mais intuitiva de definir a triangulação de Delaunauy é pelo conceito de *triângulo de Delaunay*. Um triângulo é de Delaunay se seu circuncírculo é vazio. O circuncírculo é o único círculo que passa pelos vértices do triângulo. Uma triangulação em que todos os triângulos são de Delaunay é uma triangulação de Delaunay. Uma triangulação de Delaunay de um conjunto V de vértices é única se não houver quatro vértices de V que sejam cocirculares.

A triangulação de Delaunay tornou-se popular devido principalmente à seguinte propriedade, descoberta por Lawson [14]. Entre todas as triangulações possíveis de um conjunto de vértices, a triangulação de Delaunay maximiza o menor ângulo presente na triangulação. Em métodos numéricos, ângulos próximos a zero grau ou π podem resultar em um erro grande na aproximação.

Na Fig. 1, mostra-se uma triangulação de Delaunay e um diagrama de Voronoi para um conjunto de pontos. Os circuncentros dos triângulos de Delaunay são os vértices do diagrama de Voronoi.

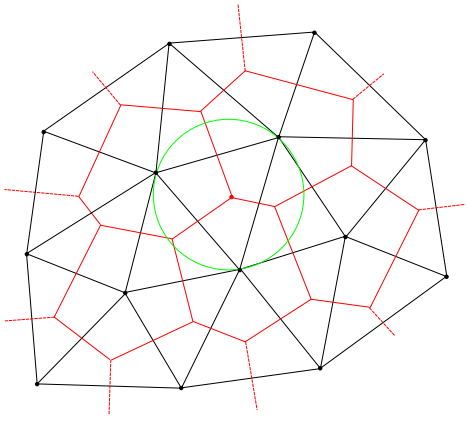

Fig. 1. Uma triangulação de Delaunay e seu respectivo diagrama de Voronoi. Em verde é mostrado o circuncírculo de um triângulo de Delaunay, cujo centro é um vértice do diagrama de Voronoi.

B. Geração da triangulação de Delaunay

Neste trabalho, utilizou-se algoritmo de Lawson [14] para manter a triangulação de Delaunay durante o refinamento adaptativo e de Ruppert. O algoritmo de Lawson foi projetado por uma abordagem incremental, em que os vértices são adicionados um por vez na triangulação e baseia-se no *flip de arestas*. Seja $\overline{ab}$ uma aresta compartilhada pelos triângulos $\triangle abc$ e $\triangle abd$. Se a aresta $\overline{ab}$ não é localmente de Delaunay, então, é possível realizar o *flip* da aresta $\overline{ab}$ no quadrângulo *abcd*, excluindo-se a aresta $\overline{ab}$ e inserindo-se a aresta $\overline{cd}$, que será localmente de Delaunay. Na Fig. 2 é mostrado o *flip* descrito.

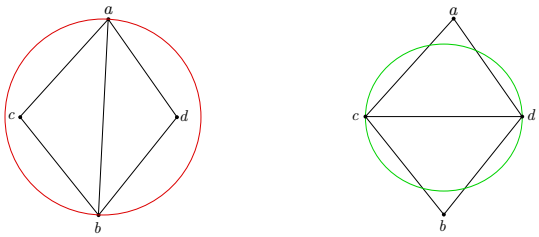

Fig. 2. Exemplificação de uma operação de *flip* de aresta. Na figura da esquerda a aresta *ab* não satisfaz o critério de Delaunay. Na figura da direita é mostrado o resultado após a aplicação do *flip*.

O algoritmo de Lawson funciona da seguinte forma quando um novo vértice *v* é inserido na triangulação:

- se *v* pertence a uma aresta já existente na triangulação, a aresta é excluída e inserem-se quatro arestas ligando *v* aos vértices do quadrângulo em que *v* se encontra;
- se não, o triângulo que contém *v* é encontrado e, então, três novas arestas são criadas ligando-se o novo vértice aos três vértices do triângulo; desse modo, o triângulo original é substituido por três novos triângulos.

Feito isso, um procedimento recursivo verifica se os novos triângulos ou as arestas do triângulo que foi dividido são de Delaunay. Caso alguma aresta (ou triângulo) não atenda à propriedade de Delaunay, é aplicado o *flip* de aresta para remover a aresta não de Delaunay. A cada *flip*, aparecem duas novas arestas que devem ser testadas. O procedimento acaba quando não houver aresta oposta ao novo vértice que não seja localmente de Delaunay.

C. Refinamento de Ruppert

O refinamento de Delaunay é uma técnica para geração de triangulações de Delaunay de qualidade, isto é, oferece alguma garantia quanto à forma dos triângulos. Mais especificamente, em um refinamento de Delaunay, uma triangulação de Delaunay é mantida e um critério é usado para escolher sucessivamente novos pontos para serem adicionados à triangulação.

A entrada para o algoritmo de Ruppert é um grafo planar com arestas não curvas (*planar straight-line graph* - PSLG) $G(V, S)$, em que V é o conjunto de vértices e S é o conjunto de segmentos. O algoritmo de Ruppert gera uma triangulação de Delaunay a partir de $G(V, S)$, a qual contém todos os

vértices de V e todos segmentos de S, sendo que os segmentos estarão representados pela união de uma ou mais arestas. Nesta aplicação, utiliza-se como entrada uma triangulação de Delaunay definida previamente, delimitada por segmentos. Sendo assim, não será descrito a parte da geração da triangulação de Delaunay do algoritmo de Ruppert, focando-se apenas no refinamento da triangulação.

O algoritmo de Ruppert [7] garante que todos os triângulos da triangulação de saída terão ângulos entre α e $\pi - 2\alpha$, sendo que α é um parâmetro que pode ser escolhido entre 0 e 20 graus. O círculo com o segmento $\overline{ab}$ como diâmetro é chamado de *círculo diametral* de $\overline{ab}$. Considere que um terceiro vértice c *invade* o segmento $\overline{ab}$ se ele está no interior do círculo diametral de $\overline{ab}$. O algoritmo de Ruppert possui duas operações principais descritas a seguir. *1)* Divide-se um triângulo adicionando-se um vértice em seu circuncentro. Dado o ângulo mínimo α como entrada, divide-se qualquer triângulo que tenha algum ângulo menor que α, a menos que o circuncentro do triângulo invada algum segmento. Neste último caso, em vez de dividir o triângulo, o algoritmo divide o segmento. *2)* Divide-se um segmento adicionando-se um vértice em seu ponto médio. Ao se dividir um segmento, substitui-se o segmento por dois novos segmentos.

A ideia do algoritmo é eliminar triângulos com ângulo menor que α. Um bom local para se inserir um novo vértice é no circuncentro de um triângulo porque esse vértice é equidistante a todos os vértices do triângulo. Ao se inserir um vértice no circuncentro de um triângulo, este não será mais de Delaunay. Desse modo, mesmo que o circuncentro não se encontre no interior do triângulo com ângulo menor que α, este será eliminado por não ser mais de Delaunay.

O algoritmo de Ruppert poderia falhar se o circuncentro de algum triângulo se encontrasse fora da triangulação. Entretanto isso não acontece. Em uma triangulação de Delaunay, se não houver segmento invadido, não há circuncentro externo à triangulação. Desse modo, o algoritmo de Ruppert não permite circuncentro externo à triangulação (veja uma prova em [15]).

IV. Implementação

A implementação recebe uma triangulação de Delaunay inicial, lida de um arquivo de entrada, delimitada por segmentos. Segmentos incidentes devem estar separados por um ângulo de pelo menos $60°$, para que as garantias do algoritmo de Ruppert sejam válidas. A estrutura de dados que representa a triangulação é composta por dois tipos de elementos: vértices e triângulos. Vértices são representados pela classe *Vertex*, em que se implementou uma lista de adjacências. Triângulos são representados por objetos da classe *Triangle*, a qual contém ponteiros para os três vértices do triângulo, como também ponteiros para triângulos adjacentes.

A malha do MVF são os polígonos de Voronoi, gerados a partir dos circuncentros dos triângulos da triangulação de Delaunay. A equação diferencial parcial (EDP) em questão é a equação de Laplace, $\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$, com condições de contorno de Dirichlet.

Aplicando-se o MVF para a solução da equação de Laplace, tem-se para cada volume de controle (polígono de Voronoi) p a equação linear $\sum_{\forall i \in n} \left(\frac{T_i - T_p}{\Delta S_{ip}}\right) \Delta L_{ip} = 0$, em que n é o conjunto de polígonos adjacentes (ou vizinhos) a p, S é a distância entre p e i e L é o comprimento da interface entre p e i, ou seja, o comprimento da aresta do diagrama de Voronoi compartilhada por p e i.

Para numerar os vértices da malha, foi utilizado o algoritmo *busca em profundidade*. Utilizando tal numeração, vários testes mostraram que a matriz dos coeficientes do sistema linear retornado pelo MVF é simétrica-positiva definida e esparsa. Há fortes indícios que todas as matrizes têm essas características. Por isso, foi aplicado o método do gradiente conjugado [16] nas simulações para a solução dos sistemas lineares resultantes da aplicação do MVF.

Após esses passos, dois refinamentos com objetivos diferentes são aplicados à triangulação de Delaunay. Ambos os refinamentos utilizam o método *RefinaTriângulo*, que é basicamente a parte de inserção do algoritmo de Ruppert, após a escolha do triângulo a ser refinado.

O primeiro algoritmo de refinamento aplicado à triangulação é o refinamento adaptativo. Seu objetivo é obter uma solução melhor que a anterior em determinadas regiões do domínio. O algoritmo RefinamentoAdaptativo é descrito a seguir, o qual recebe como entrada uma triangulação de Delaunay e um limite β para o critério de refinamento. Considere $x.v_i$ o vértice i pertencente à aresta ou triângulo x e $x.v_i.u$ a quantidade de massa corrente em v_i.

Algoritmo RefinaTriângulo
Entrada: triangulação de Delaunay DT;
triângulo $t \in DT$.

1: $S \leftarrow$ Conjunto de segmentos de DT;
2: $c \leftarrow$ Circuncentro de t;
3: **if** (c invade algum segmento $s \in S$) **then**
4: **for all** ($s \in S$ invadido por c) **do**
5: Insira um vértice v no ponto médio de s;
6: Execute o algoritmo de Lawson para v;
7: **end for**
8: **else**
9: Insira um vértice v no circuncentro de t;
10: Execute o algoritmo de Lawson para v;
11: **end if**.

Algoritmo RefinamentoAdaptativo
Entrada: triangulação de Delaunay DT;
limite de refinamento β.

1: $A \leftarrow$ Arestas de DT que não estão na fronteira;
2: **for all** $a \in A$ **do**
3: **if** $mod(a.v_1.u - a.v_2.u) > \beta$ **then**
4: t_1, $t_2 \leftarrow$ Triângulos que compartilham a aresta a;
5: Seja $t_1.v_3$ o vértice de t_1 que não pertence à a;
6: Seja $t_2.v_3$ o vértice de t_2 que não pertence à a;
7: $d_1 \leftarrow mod(a.v_1.u - t_1.v_3.u) + mod(a.v_2.u - t_1.v_3.u)$;
8: $d_2 \leftarrow mod(a.v_1.u - t_2.v_3.u) + mod(a.v_2.u - t_2.v_3.u)$;

```
      if d1 > d2 then
        RefinaTriângulo(DT, t1);
      else
        RefinaTriângulo(DT, t2);
      end if
    end if
  end for
```

No algoritmo RefinaTriângulo, utilizou-se os conceitos do refinamento de Ruppert no triângulo a ser refinado, determinado pelo critério de refinamento. O segundo algoritmo de refinamento aplicado na malha tem por objetivo melhorar sua qualidade. De fato, é o algoritmo de Ruppert sem a parte da criação da triangulação. Mais especificamente, aplica-se o algoritmo RefinaTriângulo a cada triângulo com ângulo menor que α, sendo que α é estabelecido pelo usuário. Apesar de que é garantido que α pode ser entre 0 e 20 graus, na prática, pode-se obter malhas triangulares com ângulos maiores que 20 graus. Tais resultados são apresentados a seguir.

Tendo a malha sido refinada, todo o processo para obtenção de uma solução será repetido (numeração dos vértices, aplicação do MVF, resolução do sistema de equações lineares pelo método do gradiente conjugado) e, novamente será feita a verificação se o refinamento adaptativo é necessário. Quando não houver mais necessidade de refinamento, o programa termina, retornando uma solução para uma determinada malha final.

Com relação a complexidade do algoritmo RefinaTriângulo, pode-se considerar que será limitada pelo número de flips necessários para inserir um vértice pelo algoritmo de Lawson. O número de segmentos invadidos normalmente será um valor muito pequeno em relação ao número de vértices. Segundo [15], em casos degenerados, o algoritmo de Lawson pode chegar a realizar $O(n)$ *flips* para a inserção de um vértice, onde n é o número de vértices da triangulação. Entretando, a média de *flips* em casos comuns é uma constante pequena. Em seu pior caso, o algoritmo RefinamentoAdaptativo terá um limite de $O(|A||V|)$, considerando que $O(|V|)$ é o limite superior para o algoritmo RefinaTriangulo.

V. Resultados Experimentais

Foi obtida uma solução para a equação de Laplace, sendo o domínio um quadrado unitário, com as condições de contorno: $T(a,0) = 0$ para $0 < a < 1$ e $T(a,1) = T(0,a) = T(1,a) = 10$ para $0 \leq a \leq 1$. A entrada da implementação é uma triangulação de Delaunay delimitada por segmentos. A malha inicial é mostrada na Fig. 3, juntamente com os valores encontrados. Note que todos os vértices da fronteira do domínio possuem valores constantes, que são os descritos nas condições de contorno. Na Fig. 4, mostra-se a mesma malha inicial, juntamente com o seu diagrama de Voronoi, em que os polígonos parciais da fronteira não são mostrados.

Vários testes foram executados para o problema descrito, cada um contendo diferentes valores para o limite do refimento adaptativo e ângulo mínimo α. Na Fig. 5 é mostrada uma triangulação com 310 vértices, resultante de uma simulação

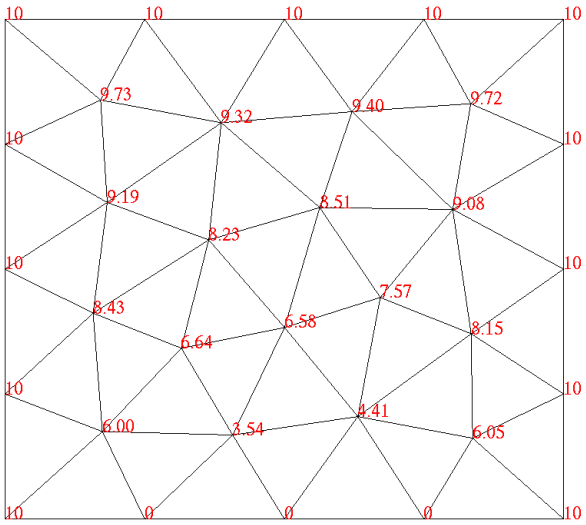

Fig. 3. Malha inicial com uma solução da equação de Laplace com condições de contorno de Dirichlet. Os valores são armazenados nos vértices da triangulação de Delaunauy.

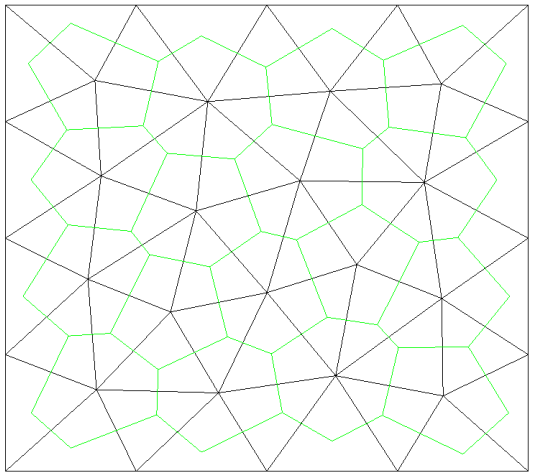

Fig. 4. Triangulação de Delaunay inicial e seu respectivo diagrama de Voronoi (em verde)(arestas infinitas dos polígonos parciais da fronteira não são mostradas).

com ângulo mínimo de 20 graus. Devido ao refinamento adaptativo, há concentração de pontos nos cantos inferiores do domínio, onde a solução apresenta maior variação.

Como comentado, na prática, o algoritmo de Ruppert pode gerar malhas com triângulos com ângulos maiores que 20 graus. Na Fig. 6 é mostrada uma triangulação contendo 2377 vértices. Tal triangulação foi obtida escolhendo-se 1.0 como critério de refinamento e ângulo mínimo de 25 graus. Infelizmente, devido a erros de ponto flutuante, há limitação para o refinamento. Vértices muito próximos podem gerar erros graves na malha. Por causa dessa limitação, em algumas regiões da malha, não é possivel refiná-la até que o critério de parada do refinamento seja satisfeito. Para isso, o usuário

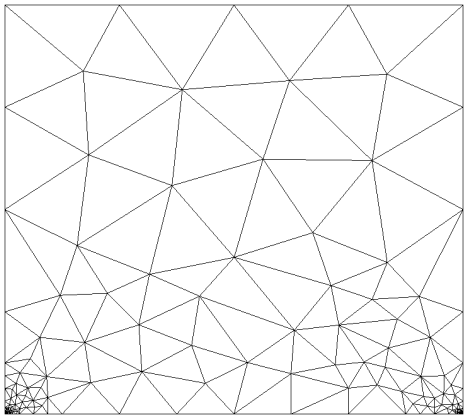

Fig. 5. Malha com 310 vértices em simulação com ângulo mínimo de 20 graus e limite do refinamento adaptativo igual a 2.79.

pode estabelecer um limite máximo de refinamento entre dois vértices (distância mínima entre os vértices), em que, nas simulações, é 10^{-4}. Em [17], são apresentados algoritmos para lidar com problemas de aritmética de ponto flutuante, que devem ser implementados e apresentados em trabalhos futuros. Na Fig. 7 é mostrada uma malha de uma simulação com ângulo mínimo de 32 graus. É mostrado na Fig. 8 o mapa de cores da solução da EDP para essa malha.

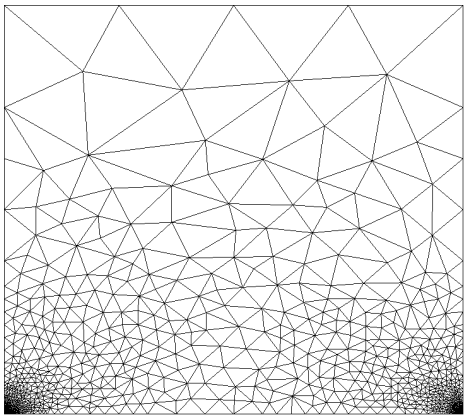

Fig. 6. Malha com 2377 vértices em simulação com ângulo mínimo de 25 graus e limite do refinamento adaptativo igual a 1.

Em se tratando da aplicação de métodos numéricos, o principal objetivo é obter a melhor aproximação possível da solução da EDP em um tempo de execução razoável. Por isso, não é interessante a inserção de uma quantidade grande de vértices na triangulação para se obter ângulos um pouco melhores, se o ganho na qualidade da solução for pequeno e o incremento no tempo de execução for grande. Nas simulações, para ângulos maiores do que 32 graus, houve necessidade de inserção de muitos vértices na triangulação de Delaunay, de modo que o tempo de execução aumentou bastante, sem que houvesse uma grande melhoria na solução. Na Fig. 9, é

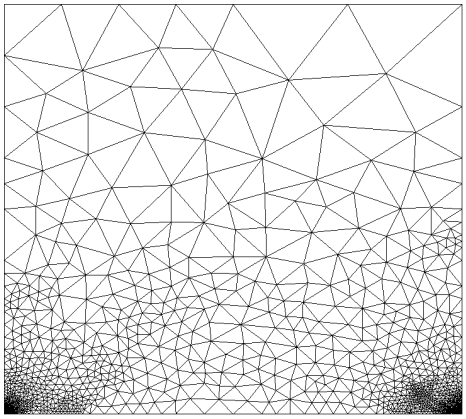

Fig. 7. Malha com 3318 vértices em simulação com ângulo mínimo de 32 graus e limite do refinamento adaptativo igual a 1.

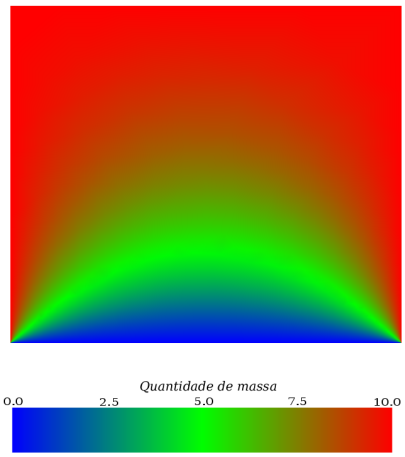

Fig. 8. Mapa de cores de uma simulação com ângulo mínimo de 32 graus e limite do refinamento adaptativo igual a 1.

mostrada uma malha em que o menor ângulo é de 33 graus.

Na Fig. 10, mostram-se as porcentagens de tempo gastas por cada etapa utilizada no processo para obter a solução da EDP, em algumas simulações. Nota-se que a resolução do sistema linear demanda o maior custo computacional. Percebe-se também que o tempo execução do refinamento de Ruppert aumenta de modo significativo quando o ângulo mínimo desejado é maior ou igual a $30°$. Nas últimas linhas da tabela o refinamento adaptativo tem custo computacional grande por causa do critério de refinemento empregado.

VI. Conclusões e trabalhos futuros

Este trabalho apresentou uma aplicação do MVF, utilizando o diagrama de Voronoi para a discretização da equação de Laplace, sendo o diagrama de Voronoi construído por meio da triangulação de Delaunay. O trabalho focou-se na geração de uma triangulação de qualidade, por meio da aplicação do refinamento de Ruppert. Ainda, foi implementado um refinamento adaptativo da malha. O algoritmo de Ruppert

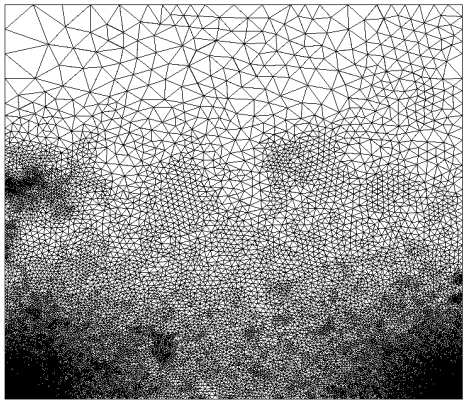

Fig. 9. Malha com 65859 vértices em simulação com ângulo mínimo de 33 graus.

Vértices	*Ang. Mín.*	*Crit. Ref.*	Ruppert	Ref. Adap.	Res. Sist. Linear	Busca prof.	MVF
4565	25	0,70	2,37	14,62	80,22	0,22	2,58
4703	28	0,70	3,23	12,73	81,01	0,38	2,66
5165	30	0,70	6,99	12,18	78,24	0,43	2,16
5258	31	0,70	10,01	10,70	76,59	0,35	2,35
8390	25	0,50	2,19	11,63	84,31	0,37	1,50
8851	28	0,50	3,84	10,78	83,65	0,29	1,43
9329	30	0,50	9,37	11,47	77,52	0,28	1,37
11690	32	0,50	19,39	7,88	71,40	0,21	1,12
12844	25	0,40	2,13	9,31	87,33	0,20	1,02
13140	28	0,40	4,55	9,75	84,44	0,17	1,08
13859	30	0,40	10,07	11,01	77,74	0,16	1,02
16033	32	0,40	14,61	8,76	75,57	0,18	0,88
46485	25	0,20	3,07	20,20	76,21	0,09	0,42
47675	28	0,20	7,30	22,89	69,32	0,09	0,40
49299	30	0,20	11,37	21,05	67,11	0,08	0,38
55163	32	0,20	22,54	19,46	57,59	0,07	0,34
81195	28	0,15	7,85	26,84	64,97	0,06	0,28
84124	30	0,15	12,99	25,61	61,08	0,06	0,26

Fig. 10. Porcentagens de tempo gastas por cada etapa em simulações numéricas para alguns ângulos mínimos e critérios de refinamento. Tempo gasto com operações como em saída padrão foi desconsiderado.

mostrou-se adequado, gerando malhas com ângulos mínimos de até 33 graus.

Em trabalhos futuros, pretende-se implementar outros algoritmos para geração de triangulações de Delaunay de qualidade, entre eles, os algoritmos propostos por Üngör [18], Rivara [19] e Chernikov e Chrisochoides [20]. Outros trabalhos importantes a serem desenvolvidos são a melhoria em relação a erros de ponto flutuante e a utilização de estimadores de erro para controlar o refinamento adaptativo. Comparações do custo computacional em relação a malhas uniformes e com outras técnicas de refinamento adaptativo também devem ser realizadas.

AGRADECIMENTOS

Agradece-se ao apoio financeiro da FAPEMIG, do CNPq e da PRP-UFLA.

REFERÊNCIAS

[1] G. F. Voronoi, "Nouvelles applications des paramètres continus à la théorie des formes quadratiques." *Journal für die Reine und Angewandte Mathematik*, vol. 133, pp. 97 – 178, 1907.

[2] B. N. Delaunay, "Sur la sphère vide," *Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh Nauk*, vol. 7, pp. 793 – 800, 1934.

[3] K. Domelevo and P. Omnes, "A finite volume method for the Laplace equation on almost arbitrary two-dimensional grids," *ESAIM: Mathematical Modelling and Numerical Analysis*, vol. 39, no. 06, pp. 1203–1249, 2005. [Online]. Available: http://dx.doi.org/10.1051/m2an:2005047

[4] S. Noelle, N. Pankratz, G. Puppo, and J. R. Natvig, "Well-balanced finite volume schemes of arbitrary order of accuracy for shallow water flows," *Journal of Computational Physics*, vol. 213, no. 2, pp. 474 – 499, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S002199910500389X

[5] F. Boyer and F. Hubert, "Finite Volume Method for 2D Linear and Nonlinear Elliptic Problems with Discontinuities," *SIAM J. Numer. Anal.*, vol. 46, no. 6, pp. 3032–3070, Sep. 2008. [Online]. Available: http://dx.doi.org/10.1137/060666196

[6] H. Jasak and Z. Tukovi, "Automatic Mesh Motion for the Unstructured Finite Volume Method," 2004.

[7] J. Ruppert, "A Delaunay Refinement Algorithm for Quality 2-Dimensional Mesh Generation," *Journal of Algorithms*, vol. 18, no. 3, pp. 548–585, 1994. [Online]. Available: http://hdl.handle.net/2060/19970014411

[8] J. P. Gois, K. C. Estácio, C. M. Oishi, V. Berttoni, V. A. Botta, A. Nagamine, F. A. Kurokawa, and F. Federson, "Aplicação de volumes finitos na simulação numérica de contaminação em lençóis freáticos," *ICMC/USP - Instituto de Ciências Matemáticas e de Computação - Departamento de Computação e Estatística, São Carlos, SP, Brasil*, 2005.

[9] I. D. Mishev, "Finite Volume Methods on Voronoi Meshes," *Numerical Methods for Partial Differential Equations*, vol. 14, no. 2, pp. 193–212, 1998. [Online]. Available: http://dx.doi.org/10.1002/(SICI)1098-2426(199803)14:2⟨193::AID-NUM4⟩3.0.CO;2-J

[10] Z. Cendes, D. Shenton, and H. Shahnasser, "Magnetic field computation using Delaunay triangulation and complementary finite element methods," *Magnetics, IEEE Transactions on*, vol. 19, no. 6, pp. 2551–2554, nov 1983.

[11] J. R. Shewchuk, "Delaunay refinement algorithms for triangular mesh generation," *Computational Geometry*, vol. 22, no. 1–3, pp. 21–74, 2002, ¡ce:title¿16th ACM Symposium on Computational Geometry¡/ce:title¿. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925772101000475

[12] L. P. Chew, "Guaranteed-quality mesh generation for curved surfaces," in *Proceedings of the ninth annual symposium on Computational geometry*, ser. SCG '93. New York, NY, USA: ACM, 1993, pp. 274–280. [Online]. Available: http://doi.acm.org/10.1145/160985.161150

[13] J. Shewchuk, "Triangle: Engineering a 2d quality mesh generator and Delaunay triangulator," in *Applied Computational Geometry Towards Geometric Engineering*, ser. Lecture Notes in Computer Science, M. Lin and D. Manocha, Eds. Springer Berlin / Heidelberg, 1996, vol. 1148, pp. 203–222, 10.1007/BFb0014497. [Online]. Available: http://dx.doi.org/10.1007/BFb0014497

[14] C. L. Lawson, *Software for C^1 surface interpolation*. Academic Press, 1977, vol. 3, pp. 161–194.

[15] J. R. Shewchuk, "Delaunay Refinement Mesh Generation," Ph.D. dissertation, School of Computer Science, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, 1997.

[16] M. R. Hestenes and E. Stiefel, "Methods of Conjugate Gradients for Solving Linear Systems," *Journal of Research of the National Bureau of Standards*, vol. 49, no. 6, pp. 409–436, Dec. 1952.

[17] D. E. Knuth, *Art of Computer Programming, Volume 2: Seminumerical Algorithms (3rd Edition)*, 3rd ed. Addison-Wesley Professional, Nov. 1997. [Online]. Available: http://www.worldcat.org/isbn/0201896842

[18] A. Üngör, "Off-centers: A new type of Steiner points for computing size-optimal quality-guaranteed Delaunay triangulations," *Computational Geometry*, vol. 42, no. 2, pp. 109 – 118, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S092577210800059X

[19] M.-C. Rivara and C. Calderon, "Lepp terminal centroid method for quality triangulation," *Comput. Aided Des.*, vol. 42, no. 1, pp. 58–66, Jan. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.cad.2008.11.004

[20] A. Chernikov and N. Chrisochoides, "Generalized Delaunay Mesh Refinement: From Scalar to Parallel," in *Proceedings of the 15th International Meshing Roundtable*, P. P. Pébay, Ed. Springer Berlin Heidelberg, 2006, pp. 563–580. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-34958-7_32