Método incremental e randômico para geração de malhas não-estruturadas e não-convexas.

Gabriel Felipe de Oliveira Antão

Departamento de Engenharia Mecânica, UFRJ.

Resumo

Este trabalho apresenta a implementação de um algoritmo incremental e randômico para geração de malhas não-estruturadas utilizando-se a Triangulação de Delaunay. O objetivo é gerar uma malha bidimensional a partir de um conjunto de pontos aleatoriamente dispostos no interior de uma circunferência e verificar o tempo necessário para o processo. Adicionalmente, apresenta-se um método para geração de malhas não-convexas a partir do uso de um método de identificação de contorno em imagens rasterizadas monocromáticas. O código foi implementado em Python com auxílio das bibliotecas NumPy, Matplotlib e Scikit-Image.

I. Introdução

Uma das etapas do Método de Elementos Finitos (MEF) é a geração da malha que representa a discretização do domínio físico do problema em questão. Softwares especializados implementam algoritmos para geração automática dessas malhas com base em um modelo previamente criado em um software de CAD. Para domínios bidimensionais, o método mais difundido para geração dessas malhas é a Triangulação de Delaunay (TD).

Em contraste com a malha estruturada, que possuem seus nós regularmente dispostos sobre a região do domínio do problema, a malha não-estruturada possuem uma disposição de nós arbitrária no espaço. Apesar da primeira ser mais fácil de ser gerada, ela limita somente a alguns problemas simples solucionados pelo MEF [2]. Devido a isso, para solução da maioria dos problemas pelo MEF é necessário lançar mão de malhas não-estruturadas para discretização do domínio.

O programa desenvolvido anteriormente[1] gerava apenas malhas estruturadas simples, de modo que os domínios bidimensionais possíveis de serem representados estavam limitados somente a algumas poucas geometrias regulares. A implementação descrita neste trabalho permite a criação de geometrias mais complexas do que era possível anteriormente, desta vez fazendo uso da TD para criação da malha.

Uma limitação da TD é gerar somente malhas de regiões convexas, necessitando de alguns artifícios complementares caso o objetivo seja discretizar um domínio em duas dimensões no qual essa característica não esteja presente. Este trabalho apresenta uma adaptação do algoritmo incremental e randômico demonstrado por Guibas et al. [1] para geração de uma malha bidimensional não-estruturada e não-convexa a partir de uma imagem rasterizada monocromática.

[1] Os códigos deste trabalho e do anterior encontram-se em: https://github.com/gabrielantao/EEK-597-UFRJ

II. Descrição do algoritmo

Geração das malhas

A técnica de TD consiste em, dado um conjunto de pontos **P** no plano, subdividir a região formada pela envoltória convexa desse conjunto em regiões triangulares em que não há nenhum ponto no interior da circunferência circunscrita a qualquer dos triângulos formados. A TD garante o máximo valor para o menor ângulo em todos os triângulos gerados, assim evitando que haja triângulos com baixos valores para os ângulos internos [3].

Há vários algoritmos para realização do TD; muitos deles foram descritos por Bern e Eppstein [2]. O método utilizado aqui consiste em, dado um conjunto de pontos **P**, gerar um triângulo inicial com três pontos – chamados *pontos fictícios* – que envolva todos os pontos do conjunto, sortear um ponto do conjunto, adicionar à região do triângulo inicial e realizar o procedimento de triangulação com esses pontos. Em seguida, repetir o procedimento de sorteio, adição e triangulação para cada um dos pontos ainda não sorteados. O fato de existir um triângulo inicial que envolve todos os pontos do conjunto garante que, para qualquer ponto sorteado, sempre haverá um triângulo sobre o qual esse ponto residirá. O algoritmo é chamado incremental porque apenas um ponto do conjunto é acrescentado à triangulação por vez e randômico porque o próximo ponto a ser adicionado é sorteado do conjunto e esse procedimento leva ao tempo esperado de $\mathcal{O}(N \log N)$ do algoritmo [3].

O procedimento de triangulação baseia-se em, acrescentado um novo ponto, identificar sobre qual triângulo existente esse ponto está, conectar as arestas desse triângulo ao ponto sorteado e proceder a *legalização* das arestas adjacentes ao triângulo original. A legalização das arestas é o procedimento no qual verifica-se, para cada tríade de pontos formados por dois dos vértices do triângulo original e o ponto adicionado, a existência de algum outro ponto no interior da circunferência circunscrita ao triângulo formado por essa tríade de pontos. Caso haja um ponto no interior da circunferência, a aresta é dita *ilegal* e precisa ser invertida. Essa última etapa é justamente o que garante as características da TD citadas na seção anterior. As etapas desse procedimento estão mostradas na Figura 1.

A referência para os triângulos criados é mantida pelo triângulo subdividido, assim os dados são armazenados em uma estrutura de árvore. Como essa estrutura, basta pesquisar a árvore em profundidade para se identificar sobre qual triangulo um novo ponto está e, portanto, a criação das novas arestas e a verificação da necessidade de legalizar arestas adjacentes é bastante agilizada, uma vez que não é necessário iterar sobre todos os triângulos.

Malhas não-convexas

Para a criação de malhas não-convexas é necessária a identificação de regiões com concavidades que são incluídas na TD. A técnica utilizada aqui consiste em gerar uma malha a partir de uma imagem monocromática fornecida ao programa e fazer uso das cores para mascarar os pontos não desejados na triangulação final. O procedimento para realização dessa tarefa se resume a criar uma malha com pontos aleatoriamente distribuídos ao longo da extensão da imagem, filtrar somente os pontos em posição cuja cor na imagem seja o preto, realizar a TD desses pontos e, por fim, eliminar os triângulos com centros sobre região de cor branca na imagem.

Esse procedimento poderia ser realizado desta forma, mas geraria deformações indesejadas nas regiões fronteiriças da imagem, uma

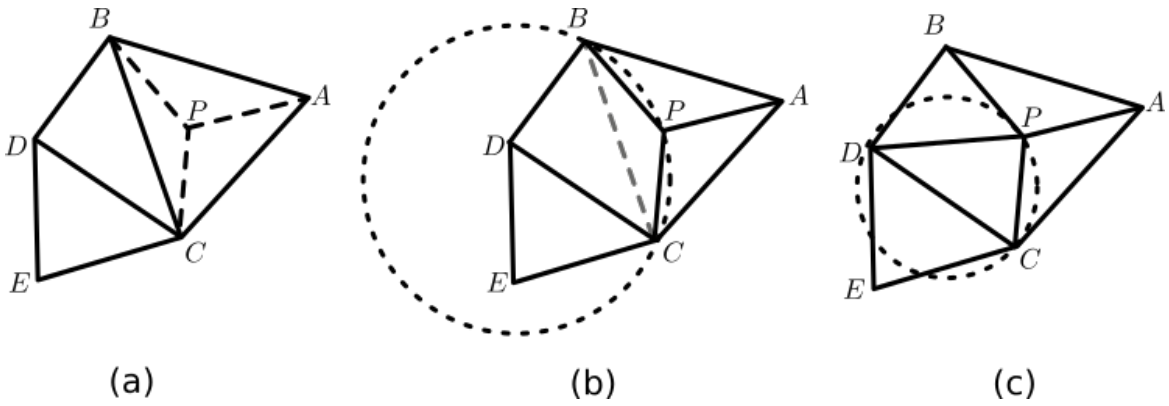

Figura 1: *(a) o ponto P é adicionado e as arestas são criadas (b) a circunferência de BCP contém outros pontos, então BC precisa ser legalizada (c) invertida a aresta BC para DP, a circunferência CDP não contém outros pontos.*

vez que os pontos criados aleatoriamente não coincidiriam necessariamente com as fronteiras da imagem. Sendo assim, acrescenta-se uma fase de identificação dos contornos, ou seja, a obtenção de pontos pertencentes à fronteira entre regiões de cor branca e de cor preta.

III. Resultados

Tempo para geração da malha

A implementação realizada neste trabalho é uma versão não otimizada, de modo que para se definir os limites de utilização do código é desejável se obter medidas de tempo de execução do programa. Foram medidos tempos de execução do código com o timeit para geração de uma malha com pontos no interior de uma região circular. O programa foi executado em um computador pessoal com sistema operacional Linux Mint 17.3, processador Intel Core i5-2310 2.9GHz e placa-mãe Gigabyte H61M-61.

Os tempos da Figura 2 são a média de cinco execuções para cada número de pontos e se verifica um tempo menor que um minuto para construir uma malha com até alguns milhares de pontos. A função nativa da biblioteca SciPy é capaz de criar as mesmas malhas em frações de segundo, portanto chega a ser quase mil vezes mais rápida que a versão implementada neste trabalho. No entanto, a função do SciPy somente realiza a TD em regiões convexas.

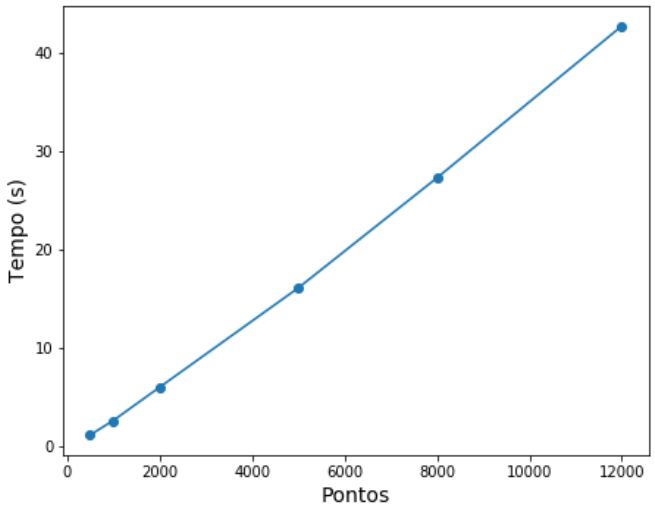

Figura 2: *Tempo de execução do código em função do número de pontos.*

Malha de geometria arbitrária

Utilizou-se a biblioteca Scikit-Image para se obter a TD de uma imagem com regiões de concavidade e com um furo interno. A função

find_contours identifica os contornos da imagem e retorna um conjunto de pontos aqui usados para garantir a existência de pontos ao longo da fronteira entre regiões de cor branca e cor preta evitando assim a deformação no contorno. Obtidos os pontos do contorno, os pontos do interior são acrescentados ao conjunto para então realizar a triangulação e a malha final mostrada na Figura 3.

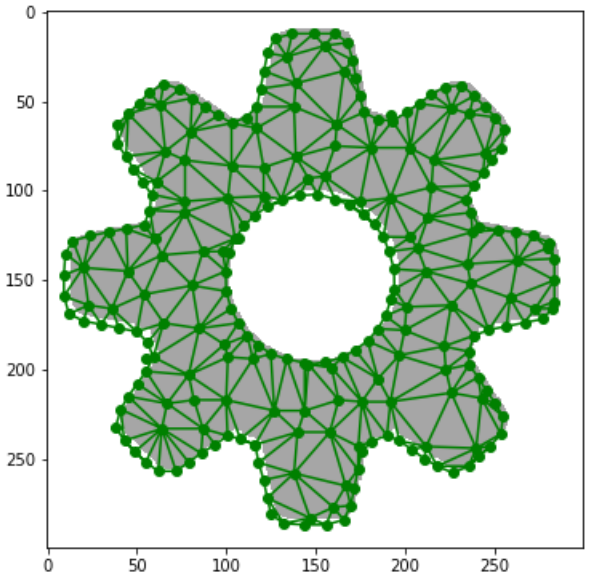

Figura 3: *Malha com 342 elementos de uma imagem rasterizada (em cinza). Dimensões em pixels.*

Apesar de haver uma pequena defasagem entre a imagem original e o contorno da malha, o código é capaz de construir a malha com as regiões de concavidade e com furo central da imagem respeitando as fronteiras da imagem. A colocação interna dos pontos é aleatória, mas outra regra de distribuição desses pontos pode ser utilizada de acordo com a necessidade.

IV. Conclusão

Ao contrário programa descrito no trabalho anterior, o código apresentado aqui gera malhas não-estruturadas de geometria bidimensional qualquer. Adicionalmente, o programa é capaz de gerar malhas de uma nuvem de alguns poucos milhares de nós que representam regiões não-convexas a partir de uma imagem monocromática fornecida.

A implementação apresentada priorizou a legibilidade em detrimento da performance, de modo que este trabalho não pretende de nenhuma forma ser uma alternativa a outras bibliotecas com código já otimizado para esse propósito, mas somente demonstrar o método incremental randômico criado por Guibas et al. Portanto, na linguagem de programação Python, é preferível o uso do NumPy/SciPy para uma grande quantidade de pontos, já que nessa circunstância o tempo necessário de processamento aumenta consideravelmente para o programa aqui apresentado.

Referências

[1] Guibas, L.J., et al. *Randomized Incremental Construction of Delaunay and Voronoi Diagrams* Algorithmica, vol. 7, pp. 381–413, 1992.

[2] Bern, M., Eppstein, D. *Mesh Generation And Optimal Triangulation* Computing in Euclidean Geometry, pp. 47-123, 1995.

[3] Press, William H. et al. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. 3 ed. New York: Cambridge University Press, 2007.

[4] Lischinski, D. *Incremental Delaunay Triangulation.* In: Graphics Gems IV, Heckbert, P.S., (ed.) Cambridge: Academic Press. 1993.