Aline Regina Becher

Malhas poligonais exteriores no plano

Dissertação apresentada ao Curso de Pós-Graduação em Matemática Pura e Aplicada do Departamento de Matemática, do Centro de Ciências Físicas e Matemáticas da Universidade Federal de Santa Catarina para obtenção de grau de Mestre em Matemática Pura e Aplicada, com Área de Concentração em Matemática Aplicada.

Orientador:
Prof. Dr. Leonardo Koller Sacht

UNIVERSIDADE FEDERAL DE SANTA CATARINA

Florianópolis

2018

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Becher, Aline Regina
Malhas poligonais exteriores no plano / Aline Regina Becher ; orientador, Leonardo Koller Sacht, 2018.
90 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro de Ciências Físicas e Matemáticas, Programa de Pós-Graduação em Matemática Pura e Aplicada, Florianópolis, 2018.

Inclui referências.

1. Matemática Pura e Aplicada. 2. Malhas poligonais. 3. Geração de malhas no plano. I. Sacht, Leonardo Koller . II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Matemática Pura e Aplicada. III. Título.

Aline Regina Becher

Malhas poligonais exteriores no plano

Florianópolis

2018

Esta dissertação foi julgada para a obtenção do Título de Mestre, e aprovada em sua forma final pelo Curso de Pós-Graduação em Matemática Pura e Aplicada.

Prof. Dr. Ruy Coimbra Charão
Coordenador de Curso

Banca Examinadora:

Prof. Dr. Leonardo Koller Sacht
Orientador

Prof. Dr. Luiz Carlos Pacheco Rodrigues Velho (IMPA)

Profa. Dra. Melissa Weber Mendonça (UFSC)

Prof. Dr. Vinícius Viana Luiz Albani(UFSC)

A meu irmão, Leonardo Joaquim Becher,
cuja presença é sentida em todos os momentos.

“Tenho a impressão de ter sido uma criança brincando à beira-mar, divertindo-me em descobrir uma pedrinha mais lisa ou uma concha mais bonita que as outras, enquanto o imenso oceano da verdade continua misterioso diante de meus olhos.”

— Isaac Newton

Agradecimentos

O caminho que me trouxe até aqui foi longo e árduo. Muitos foram os momentos difíceis, e a vontade de desistir foi grande. Mas estou aqui, e não cheguei a este ponto sozinha. Foram muitas as pessoas envolvidas nesta caminhada, a quem agradeço com profundo respeito e admiração.

A meus pais, José e Eliane, que estiveram ao meu lado em todos os momentos e sempre acreditaram e confiaram em mim. Com eles aprendi sobre respeito, caráter, humildade e persistência.

A meu irmão, Tiago, e à minha cunhada, Luana, que me deram o maior presente que já recebi, meu afilhado Henrique. Obrigada pela preocupação e incentivo.

A meu irmão, Leonardo, que me ensinou sobre amar incondicionalmente.

A meu irmão, Matheus, que me ensinou sobre o valor de um sorriso.

A meu afilhado, Henrique, o amor da minha vida.

À minha vó, Doraci, pela preocupação e cuidado.

À minha vó, Lica, pelas orações.

A minha prima, Larissa, pelas risadas infinitas e por acreditar em mim mais do que eu mesma.

A todos os familiares, em geral, pelas palavras de apoio e incentivo.

Aos amigos que a Matemática me presenteou, e que vou levar sempre comigo.

À Daniella Losso, minha melhor amiga, que apesar das divergências de opinião mantém-se ao meu lado sempre.

A Eduardo Pandini, pela companhia para o almoço, fins de semana de estudos, festas e muitos doces.

A Leonardo Biz, pela amizade desde o início da graduação.

A Natã Machado, pela grande amizade e por toda a ajuda com a Matemática.

À Marina Geremia, pela companhia para os doces, pelas melhores conversas e por muitas risadas.

À Ruana Schneider, cuja distância não diminuiu o valor da nossa amizade.

Meus mais sinceros agradecimentos, respeito e admiração ao professor Leonardo Koller Sacht, meu orientador. Por aceitar orientar este trabalho e por conduzi-lo com maestria. Obrigada pelos sábios conselhos durante o mestrado, por todo o apoio durante o processo de seleção para a USP, e principalmente, pela paciência.

Aos professores que tive durante o mestrado, Douglas Gonçalves, Fábio Botelho, Danilo Royer e Giuliano Boava, cujos exemplos e ensinamentos levarei para sempre.

A todos os professores que tive durante a graduação, em especial, à professora Melissa Weber Mendonça, meu maior exemplo de profissional. Obrigada pela orientação na graduação e pela ajuda com projeto da USP.

Aos professores Luiz Carlos Pacheco Rodrigues Velho, Melissa Weber Mendonça e Vinícius Viana Luiz Albani, por aceitarem fazer parte da Banca examinadora e pelas valorosas contribuições a este trabalho.

À CAPES, pelo suporte financeiro.

Resumo

Neste trabalho, apresentamos um método para geração de malhas poligonais exteriores em duas dimensões baseado no artigo *Nested Cages* para malhas triangulares em três dimensões. Este método consiste de duas etapas: fluxo e re-inflação. Detalhamos todos os conceitos necessários para o fluxo, como quadratura numérica, índice de um ponto e projeção, bem como detecção e resposta a colisões e minimização de um problema restrito para a re-inflação. Apresentamos alguns resultados numéricos e limitação do método. Por fim, implementamos e disponibilizamos todos os códigos necessários para este problema em duas dimensões.

PALAVRAS-CHAVE: Malhas poligonais. Geração de malhas. Fluxo. Re-inflação. Minimização de distâncias.

Abstract

We present in this work a method to generate exterior polygonal meshes in two dimensions based on the paper *Nested Cages*. We provide details of all necessary concepts, such as flow, numeric quadrature, winding number, projection, collision and detection response and the constrained problem for reinflation. We also present some numerical results and limitation of the method. Finally, we implement and make available all the source code for this problem in two dimensions.

KEY-WORDS: Polygonal meshes. Generate meshes. Flow. Reinflate. Distance minimization.

Lista de Figuras

Sumário

1 Introdução

1.1 Motivação

A complexidade e o tamanho dos objetos computacionais estão em constante crescimento. Por isso, faz-se cada vez mais necessário o uso de algoritmos de aceleração. Nossa técnica consiste em, dada uma malha poligonal de alta resolução, usar uma malha poligonal de menor resolução exterior a esta.

A construção de malhas poligonais externas tem diversas aplicações, entre elas deformações em animação, detecção conservativa de colisões em simulações físicas e resolução de EDP's e método multigrid.

Para ilustrar o uso de malhas poligonais externas em deformação em animação, temos como referência o artigo *Harmonic Coordinates*, proposto por DeRose *et al* [12]. Este trabalho consiste em, dado um objeto com muitos vértices, em duas dimensões, construir uma malha exterior com menos vértices para mover os pontos do objeto de acordo com a deformação da malha externa. No momento da animação, em vez de mover os vértices do objeto, que são muitos, move-se os vértices da malha externa. Veja figura 1.1. Para cada vértice da malha externa que movemos, são movidos todos os pontos do objeto que são interpoladas pelas arestas incidentes àquele vértice.

Na figura 1.2, temos o exemplo de um objeto com muitos vértices e uma malha exterior de menor resolução. Podemos deformar o objeto apenas mo-

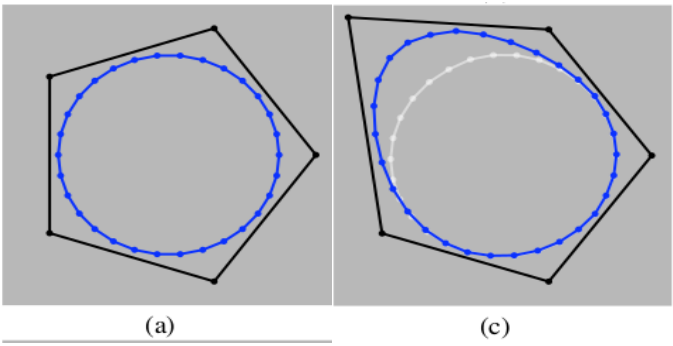

Figura 1.1: Em (a) temos um objeto (em azul) com muitos vétices e uma malha externa de menor resolução (em preto).Quando movemos um vértice da malha exterior ((c)), movemos todos os pontos do objeto que são interpolados pelas arestas incidentes ao vértice em questão. Imagem oriunda de [12].

vendo vértices da malha externa e movendo os pontos da malha fina com o método Harmonic Coordinates. Em consequencia, ao mover esses vértices, todos os pontos interpolados pelas arestas incidentes a esses vértices também se movem.

Entre os métodos que calculam malhas exteriores, podemos citar *Progressive Hulls*, proposto por Sander *et al* [14] e *OBBS*, proposto por Xian *et al* [15].

Neste trabalho vamos detalhar e implementar uma versão em duas dimensões de *Nested Cages*, proposto em 2015 por Sacht *et al* [1]. Este método mostrou-se mas eficiente que os métodos citados anteriormente uma vez que permite a otimização de uma função objetivo que mede a qualidade da malha final.

1.2 Contribuições e Objetivos

Este trabalho teve como inspiração o artigo *Nested Cages* proposto por Sacht *al* [1] em 2015, que consiste na geração de malhas triangulares exteri-

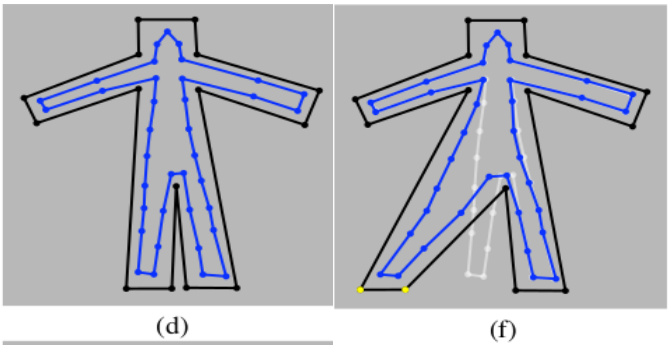

Figura 1.2: Em (d) temos um objeto com pontos interpolados por uma malha exterior com menos vértices. Movendo apenas os vértices da malha externa, é possível mover todos os pontos do objeto interpolados pelas arestas incidentes aos vértices ((f)).Imagem oriunda de [12].

ores em 3 dimensões através de etapas de fluxo e re-inflação.

Nosso objetivo neste trabalho é detalhar vários conceitos abordados em *Nested Cages*, como quadratura numérica, Winding Number e projeção, necessários para o fluxo, bem como detecção e resposta a colisão, e restrições de igualdade e desigualdade para resolver o problema de minimização de distâncias com sinal.

Além disso, implementar e disponibilizar todos os códigos necessários para a geração e malhas poligonais exteriores em duas dimensões. O trabalho *Nested Cages* só resolve este problema para 3 dimensões.

1.3 Colocação do problema

Definimos uma malha poligonal em duas dimensões como uma coleção de vértices ligados por arestas que definem um objeto no plano.

Uma malha M com n vértices é descrita por uma matriz $M \in \mathbb{R}^{n\times 2}$ tal que

$$M = \begin{bmatrix} v_{1x} & v_{1y} \\ v_{2x} & v_{2y} \\ \vdots & \vdots \\ v_{nx} & v_{ny} \end{bmatrix}.$$

em que $[v_{ix}, v_{iy}]$, com $i = 1, 2, \ldots, n$ representa a posição do vértice v_i no plano. Neste trabalho, dada uma malha poligonal com n vértices, a qual chamamos de malha fina, queremos construir uma malha com m vértices, com $m < n$, a qual chamamos de malha grosseira, de forma que esta seja totalmente exterior a primeira. Como entrada do método, temos uma malha fina e uma malha grosseira que se intersectam. Como saída, queremos a malha grosseira totalmente exterior à malha fina. Veja Figura 1.3.

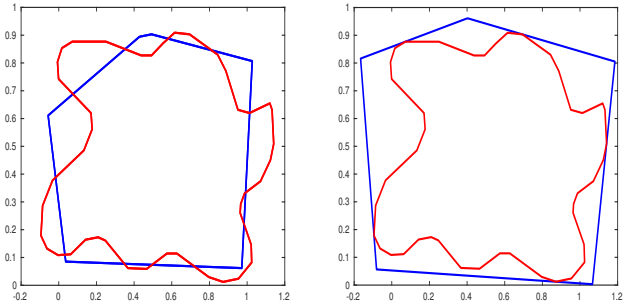

Figura 1.3: Esquerda: Malhas fina e grosseira, com intersecções.Saída: Malha grosseira totalmente no exterior da malha fina, após aplicação do método descrito neste trabalho.

Para isso, fazemos o fluxo da malha fina para o interior da malha grosseira e em seguida revertemos o fluxo fazendo a re-inflação da malha fina de volta para sua posição original empurrando a malha grosseira.

1.4 Descrição da solução

A primeira etapa consiste em mover os vértices da malha fina para o interior da malha grosseira ao longo de um fluxo que minimiza a distância com sinal entre as duas malhas. Para evoluir o fluxo da malha fina para o interior da malha grosseira tomamos passos na direção oposta ao gradiente da uma função distância com sinal. Uma vez que esta função é difícil de calcular numericamente, fazemos uma aproximação do gradiente com o auxílio de pontos de quadratura e pesos correspondentes. Iniciamos determinando os pontos de quadratura sobre a malha fina. Em seguida, atribuímos um sinal para cada ponto (geralmente -1 se ela está no interior da malha grosseira e 1 se está no exterior) e calculamos a distância de cada ponto para a malha grosseira. Feito isso, podemos aproximar o gradiente da função distância como uma soma de termos. Andamos na direção oposta ao gradiente até que a malha fina seja totalmente interior a malha grosseira.

A re-inflação consiste em reverter o fluxo levando a malha fina de volta para sua posição original, empurrando a malha grosseira. Em cada passo de fluxo, iniciamos detectando todas as colisões que acontecem entre vértices da malha fina com arestas da malha grosseira (ou arestas da malha fina com vértices da malha grosseira). Respondemos às colisões montando uma matriz de restrições que garante que a malha grosseira se mantenha exterior em cada passo. Além disso, adicionamos uma restrição de igualdade para garantir que a malha fina volte para a sua posição em cada posso do fluxo. Ao final, minimizamos uma função energia satisfazendo estas restrições.

1.5 Organização do trabalho

Este trabalho está dividido em 4 capítulos.
No Capítulo 2 tratamos do processo de fluxo. Iniciamos detalhando os conceitos de quadratura numérica, em particupar a quadratura gaussiana, além dos conceitos de projeção ortogonal e Winding Number, necessários para a aproximação do gradiente da função distância com sinal responsável por fazer o fluxo da malha fina para interior da malha grosseira.

No Capítulo 3, tratamos da re-inflação, em que revertemos o fluxo de forma que a malha fina volte para sua posição inicial empurrando a malha grosseira e detectando e respondendo colisões em cada passo. Resolvemos esse problema minimizando uma função energia com restrições impostas pela detecção e resposta a colisões.

Finalmente, no Capítulo 4, apresentamos alguns resultados numéricos, bem como a implementação dos códigos.

2 Fluxo

O Fluxo consiste em levar a malha fina para o interior da malha mais grosseira de forma a obter um estado final sem intersecções. Um exemplo com alguns passos de fluxo é dado na Figura 2.1. A Re-inflação consiste em levar a malha fina de volta para sua posição inicial empurrando a malha grosseira e respondendo colisões em cada passo de tempo. Esta etapa do método será discutida com detalhes no próximo capítulo. Vamos detalhar aqui o processo de Fluxo.

Definição 2.1. *Dadas duas malhas poligonais $\bar{F}$ e $\hat{G}$, dizemos que $\bar{F}$ é mais fina que $\hat{G}$ se possui mais arestas.*

Uma vez que queremos que a malha fina esteja totalmente no interior a malha grosseira, devemos minimizar a distância com sinal entre as duas malhas. A distância com sinal consiste em estipular um sinal para a região interna da malha grosseira, geralmente negativo. Se um ponto da malha fina está no interior da malha grosseira, multiplica-se sua distância por -1, negativando-a. Se este ponto está no exterior da malha grosseira, atribui-se a ele o sinal positivo. Minimizar a distância com sinal é necessário para que possamos garantir que a malha fina esteja totalmente interior a malha grosseira. A Figura 2.2 nos mostra um campo de distâncias com sinal no processo de Fluxo. A etapa de fluxo consiste em mover todos os vértices da malha fina

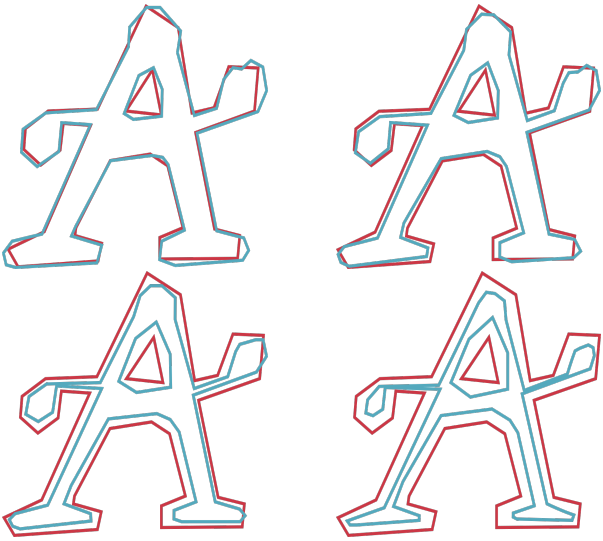

Figura 2.1: Durante o fluxo minimizamos a distância com a sinal entre as duas malhas obtendo um estado final com a malha fina totalmente no interior da malha grosseira. Imagem oriunda de [1].

$\bar{F}$ ao longo de um fluxo que minimiza a distância com sinal para uma malha mais grosseira $\hat{G}$ integrando sobre todos os pontos $\bar{p} \in \bar{F}$. Em outras palavras, queremos minimizar

$$\Phi(\bar{F}) = \int_{\bar{F}} s(\bar{p}) d(\bar{p}) dC, \tag{2.1}$$

em que $s(\bar{p})$ é o sinal de $\bar{p}$ em relação a $\hat{G}$, isto é, -1 se está no interior e 1 caso contrário e $d(\bar{p})$ é a distancia de $\bar{p}$ para $\hat{G}$.

Uma vez que, localmente, a direção do gradiente de uma função é a de maior crescimento da mesma, a direção oposta ao gradiente é a direção de máximo declive da função. Então, para cada vértice $\bar{f} \in \bar{F}(t)$, minimizamos Φ tomando pequenos passos de tempo t na direção oposta ao gradiente até

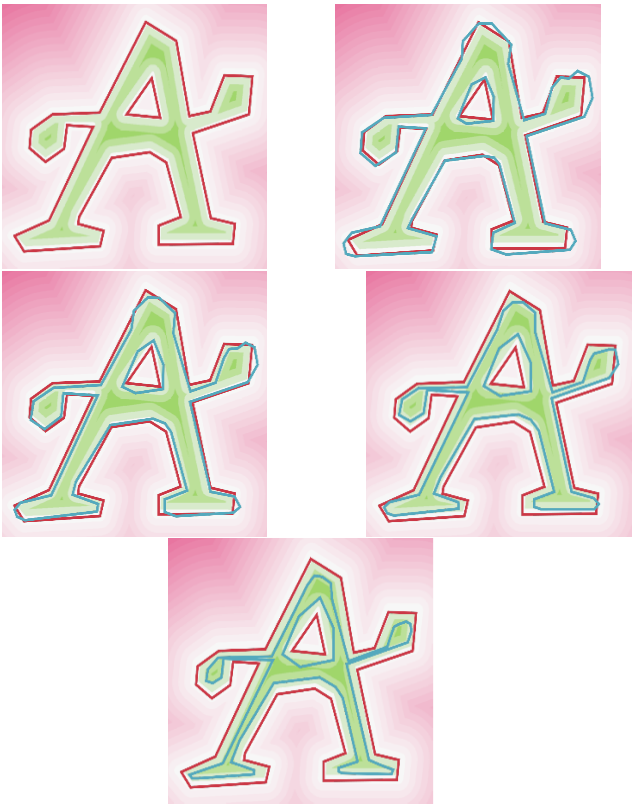

Figura 2.2: Campo de distâncias: Sinal positivo em rosa no exterior da malha final e negativo em verde no interior da malha fina. Imagem oriunda de [1].

que todos os pontos de $\bar{F}$ estejam no interior de $\hat{G}$.

$$\frac{\partial \bar{f}}{\partial t} = -\nabla_{\bar{f}} \Phi(F) \tag{2.2}$$

Mais detalhes sobre o Metodo de Máximo Declive, ou Método do Gradiente, podem ser encontrados em [5].

Sendo Φ é uma função não linear e intratável de calcular exatamente, fazemos uma aproximação do gradiente usando um método de quadratura

numérica. Para cada aresta A_i, de $\bar{F}$ escrevemos s e d utilizando pontos de quadratura $\bar{p}_{ij}$ e pesos correspondentes w_{ij}.

Uma vez que queremos minimizar a distância com sinal, é preciso determinar o sinal s de um ponto, ou seja, dado um ponto da malha fina, o sinal irá determinar se este ponto está dentro ou fora da malha grosseira. Para isso será usado o conceito de Winding Number, ou índice de um ponto, que determina o número de voltas que uma curva dá em torno de um ponto. Se este for nulo, o ponto estará no exterior da malha grosseira. Este conceito será detalhado na Seção 2.2. Antes disso, será abordado o conceito de quadratura numérica (2.1), necessário para a aproximação do gradiente de Φ em (2.1).

2.1 Quadratura

Nesta seção, vamos tratar do método de Quadratura Gaussiana no caso geral.
A quadratura gaussiana é um dos esquemas de aproximação de integral que serviu bem para este propósito. Para outros esquemas de quadratura numérica, veja [4].

Para um intervalo $[a,b]$, a quadratura gaussiana escolhe pontos, ou nós $x_1, x_2, \ldots, x_n$ em $[a,b]$, e coeficientes $c_1, c_2, \ldots, c_n$, de forma a minimizar o erro esperado obtido na aproximação

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} c_i f(x_i) \tag{2.3}$$

Os coeficientes $c_1, c_2, \ldots, c_n$ e os nós $x_1, x_2, \ldots, x_n$ são desconhecidos, então temos $2n$ parâmetros para determinar. Se os coeficientes de um polinômio também são considerados parâmetros, a classe dos polinômios de grau no máximo $2n-1$ também contém $2n$ parâmetros. Com uma escolha adequada

de valores e constantes, podemos obter a exatidão nesse conjunto.
Após a determinação dos coeficientes e nós para integração no intervalo $[a,b]$, adaptaremos esta metodologia para o cálculo da integral de linha em 2.1.

2.1.1 Quadratura Gaussiana no intervalo $[-1,1]$

Para ilustrar, vamos mostrar como encontrar os coeficientes e nós quando $n=2$ no intervalo $[-1,1]$. Suponha que queremos determinar c_1, c_2, x_1 e x_2 tal que

$$\int_{-1}^{1} f(x)dx \approx c_1 f(x_1) + c_2 f(x_2) \tag{2.4}$$

forneça um resultado exato quando $f(x)$ é um polinômio de grau menor ou igual a 3, ou seja,

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 \tag{2.5}$$

para constantes a_0, a_1, a_2 e a_3.

Teorema 2.2. *Se $f(x)$ for 1, x, x^2 ou x^3, a integral*

$$\int_{-1}^{1} f(x)dx \approx c_1 f(x_1) + c_2 f(x_2)$$

nos dá um resultado exato, se as constantes c_1 e c_2 e os nós x_1 e x_2 forem dados por $c_1 = c_2 = 1$, $x_1 = -\frac{\sqrt{3}}{3}$ e $x_2 = \frac{\sqrt{3}}{3}$.

Demonstração: Da linearidade da integral, temos

$$\int (a_0 + a_1 x + a_2 x^2 + a_3 x^3)dx = a_0 \int 1dx + a_1 \int xdx + a_2 \int x^2 + a_3 \int x^3 dx \tag{2.6}$$

Então

$$2 = \int_{-1}^{1} 1dx = c_1.1 + c_2.1$$

$$0 = \int_{-1}^{1} x dx = c_1 x_1 + c_2 x_2$$

$$\frac{2}{3} = \int_{-1}^{1} x^2 dx = c_1 x_1^2 + c_2 x_2^2$$

$$0 = \int_{-1}^{1} x^3 dx = c_1^3 + c_2 x_2^3$$

Desta forma, as constantes c_1 e c_2 e os nós x_1 e x_2 são dados pela solução do sistema

$$\begin{cases} c_1 + c_2 = 2 & (1) \\ c_1 x_1 + c_2 x_2 = 0 & (2) \\ c_1 x_1^2 + c_2 x_2^2 = \frac{2}{3} & (3) \\ c_1 x_1^3 + c_2 x_2^3 = 0. & (4) \end{cases}$$

Isolando c_1 em (1) e substituindo em (2) temos

$$c_1 = 2 - c_2$$

$$(2 - c_2)x_1 + c_2 x_2 = 0$$

$$c_2 x_2 = -2x_1 + x_1 c_2$$

Assim,

$$x_2 = x_1 - \frac{2x_1}{c_2} \tag{2.7}$$

Substituindo c_1 em (3) temos:

$$c_1 x_1^2 + c_2 x_2^2 = \frac{2}{3}$$

$$(2 - c_2)x_1^2 + c_2 \left(x_1 - \frac{2x_1}{c_2} \right)^2 = \frac{2}{3}$$

$$2x_1^2 - c_2 x_1^2 + c_2 \left(x_1^2 - \frac{4x_1^2}{c_2} + \frac{4x_1^2}{c_2^2} \right) = \frac{2}{3}$$

$$2x_1^2 - c_2x_1^2 + c_2x_1^2 - 4x_1^2 + \frac{4x_1^2}{c_2} = \frac{2}{3}$$

$$-2x_1^2 + \frac{4x_1^2}{c_2} = \frac{2}{3}$$

Multiplicando ambos os lados por c_2:

$$-2c_2x_1^2 + 4x_1^2 = \frac{2c_2}{3}$$

$$x_1^2(4 - 2c_2) = \frac{2c_2}{3}$$

Isolando x_1^2 e supondo $4 - 2c_2 \neq 0$ temos:

$$x_1^2 = \frac{2c_2}{3(4 - 2c_2)} \tag{2.8}$$

De (4),

$$c_1x_1^3 + c_2x_2^3 = 0$$

Substituindo c_1 e x_2, temos

$$(2 - c_2)x_1^3 + c_2\left(x_1 - \frac{2x_1}{c_2}\right)^3 = 0$$

$$2x_1^3 - x_1^3c_2 + c_2\left(x_1^3 - 3x_1^2\frac{2x_1}{c_2} + 3x_1\frac{4x_1^2}{c_2^2} - \frac{8x_1^3}{c_2^3}\right) = 0$$

$$2x_1^3 - x_1^3c_2 + c_2x_1^3 - 6x_1^3 + \frac{12x_1^3}{c_2} - \frac{8x_1^3}{c_2^2} = 0$$

$$-4x_1^3 + \frac{12x_1^3}{c_2} - \frac{8x_1^3}{c_2^2} = 0.$$

Divindo ambos os lados por 4:

$$-x_1^3 + \frac{3x_1^3}{c_2} - \frac{2x_1^3}{c_2^2} = 0$$

$$x_1^3\left(-1+\frac{3}{c_2}-\frac{2}{c_2^2}\right)=0 \tag{2.9}$$

De (2.8),

$$x_1^2=\frac{2c_2}{3(4-2c_2)}$$

Então

$$x_1=\sqrt{\frac{2c_2}{3(4-2c_2)}} \tag{2.10}$$

Substituindo (2.10) em (2.9):

$$\sqrt{\frac{2c_2}{3(4-2c_2)}}^3\left(-1+\frac{3}{c_2}-\frac{2}{c_2^2}\right)=0.$$

Assim devemos ter

$$\frac{2c_2}{3(4-2c_2)}=0$$

o que implica que $c_2=0$, ou

$$-1+\frac{3}{c_2}-\frac{2}{c_2^2}=0.$$

Neste caso,

$$-c_2^2+3c_2-2=0$$

Resolvendo a equação de segundo grau, temos que $c_2=2$ ou $c_2=1$.

Agora vamos analisar em casos.

Caso 1: Se $c_2=0$, por (1), devemos ter $c_1=2$ e $x_1=0$, mas isso contradiz (3).

Caso 2: Se $c_2=2$, por (1) devemos ter $c_1=0$. Substituindo em (2),temos $2x_2=0$. Isso nos diz que $x_2=0$. Mas isso contradiz (3).

Caso 3: Se $c_2=1$, por (1) devemos ter $c_1=1$. De (2), $x_1+x_2=0$, o que

implica que $x_1 = -x_2$ e de (3),

$$x_1^2 + x_2^2 = \frac{2}{3}$$

$$x_1^2 + (-x_1)^2 = \frac{2}{3}$$

$$2x_1^2 = \frac{2}{3}$$

$$x_1^2 = \frac{1}{3}$$

Assim, devemos ter $x_1 = -\frac{\sqrt{3}}{3}$ e $x_2 = \frac{\sqrt{3}}{3}$.
Então, uma solução do sistema é dada por $c_1 = c_2 = 1$ e $x_1 = -\frac{\sqrt{3}}{3}$ e $x_2 = \frac{\sqrt{3}}{3}$.
Desta forma, temos que

$$\int_{-1}^{1} f(x)dx \approx f\left(-\frac{\sqrt{3}}{3}\right) + f\left(\frac{\sqrt{3}}{3}\right)$$

e o resultado é exato quando f é um polinômio de grau menor ou igual a 3.
■

2.1.2 Quadratura Gaussiana sobre um intervalo arbitrário $[a,b]$

Para determinar os pontos de quadratura em um intervalo arbitrário $[a,b]$ devemos fazer uma mudança de variáveis no parâmetro t. Assim, devemos ter

$$\begin{cases} x = b \quad se \quad t = 1 \\ x = a \quad se \quad t = -1 \end{cases}$$

Então,

$$x = \frac{1}{2}((b-a)t + a + b) \tag{2.11}$$

Mutiplicando ambos os lados por 2, temos

$$2x = ((b-a)t + a + b)$$

$$2x - a - b = (b - a)t$$

Isolando t:

$$t = \frac{2x - a - b}{b - a}$$

ou ainda,

$$t = \frac{x - a}{b - a} + \frac{x - b}{b - a}$$

Desta forma

$$\int_a^b f(x)dx = \int_{-1}^{1} f\left(\frac{1}{2}((b-a)t + a + b)\right)\frac{(b-a)}{2}dt$$

$$\int_a^b f(x)dx = \frac{b-a}{2}\int_{-1}^{1} f\left(\frac{1}{2}((b-a)t + a + b)\right)dt$$

O mesmo procedimento pode ser usado para determinar a fórmula geral para determinar os coeficientes e nós de (2.3) para mais pontos de quadratura.

Para integrais de linha sobre segmentos de reta ligando $a \in \mathbb{R}^2$ e $b \in \mathbb{R}^2$ fazemos a mesma mudança que em (2.11) para determinar os pontos de quadratura sobre o segmento $\overline{ab}$.
Na figura (2.3), temos um exemplo de quadratura gaussiana com 2 pontos.

Para quadratura de maior ordem, alguns valores de x_k e c_k são dados na Tabela (2.1). Esses valores tabelados podem ser encontrados em diversos livros de referência, entre eles pode-se citar [4] e [7].

Agora, voltando à equação (2.1) e aplicando as regras de quadratura gaussiana, temos

$$\Phi(\bar{F}) = \int_{\bar{F}} s(\bar{p})d(\bar{p})dC = \sum_{i=1}^{m_F}\int_{\bar{p}\in A_i} s(\bar{p})d(\bar{p})dC \approx \sum_{i=1}^{m_F}\sum_{j=1}^{h} w_{ij}s(\bar{p}_{ij})d(\bar{p}_{ij}) \tag{2.12}$$

em que h é o número de pontos de quadratura e m_F é o número de arestas da malha. Usando quadratura gaussiana com dois pontos, $\Phi(\bar{F})$ é aproximada

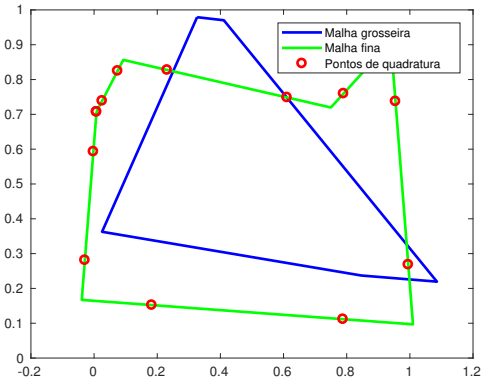

Figura 2.3: Exemplo de malha fina com dois pontos de quadratura por aresta.

n	x_k	c_k
2	$-\frac{\sqrt{3}}{3}$	1
	$\frac{\sqrt{3}}{3}$	1
3	$-0,77459667$	$5/9$
	0	$8/9$
	$0,77459667$	$5/9$
4	$-0,86113631$	$0,34785485$
	$-0,33998104$	$0,65214515$
	$0,86113631$	$0,34785485$
	$0,33998104$	$0,64214515$
5	$-0,90617985$	$0,23692689$
	$-0,53846931$	$0,47862867$
	0	$0,56888889$
	$0,90617985$	$0,23692689$
	$0,53846931$	$0,47862867$

Tabela 2.1: Pesos e pontos de quadratura para os graus $n = 2,3,4,5$. Pra $n = 4$ e $n = 5$ os valores são aproximados conforme [4].

como uma soma sobre todas arestas de $\bar{F}$:

$$\Phi(\bar{F}) \approx \sum_{i=1}^{m_F} w_{i1} s(\bar{p}_{i1}) d(\bar{p}_{i1}) + w_{i2} s(\bar{p}_{i2}) d(\bar{p}_i 2). \tag{2.13}$$

Conforme a Tabela (2.1), os coeficientes w_1 e w_2 são dados por $w_1 = w_2 = 1$. Então,

$$\Phi(\bar{F}) \approx \sum_{i=1}^{m_F} s(\bar{p}_1)d(\bar{p}_1) + s(\bar{p}_2)d(\bar{p}_2) \tag{2.14}$$

Veremos nas seções (2.2) e (2.3) como determinar o sinal $s(\bar{p}_j)$ e a distância $d(\bar{p}_j)$.

2.2 Winding Number

Nesta seção, vamos abordar o conceito de Winding Number, ou índice de um ponto. Dado um ponto da malha fina, o índice irá determinar o sinal deste ponto, ou seja, se ele está no interior ou no exterior da malha grosseira. Para o entendimento desta seção, assume-se que o leitor tenha familiaridade com números complexos, funções holomorfas e integrais sobre curvas. Para mais detalhes sobre esses conceitos, tem-se como referência [2] e [3].

Definição 2.3. *Seja γ um caminho fechado que não passa por $\alpha \in \mathbb{C}$. O winding number com respeito ao ponto α é definido como*

$$w(\gamma, \alpha) = \frac{1}{2\pi i} \int_\gamma \frac{1}{z-\alpha} dz$$

Se γ é uma curva definida em um intervalo $[a,b]$ então, de acordo com a definição de integral de linha, essa integral pode ser escrita da forma

$$\int_\gamma \frac{1}{z-\alpha} dz = \int_a^b \frac{\gamma'(t)}{\gamma(t)-\alpha} dt$$

Basicamente, o winding number é o número de voltas que uma curva fechada γ dá em torno de um ponto α. A seguir veremos três proposições relacionadas ao winding number.

Proposição 2.4. *Se γ é um caminho fechado então $w(\gamma, \alpha)$ é um número*

inteiro.

Demonstração: Sejam $\{\gamma_1, \gamma_2, \ldots, \gamma_n\} = \gamma$ uma família de curvas em que γ_i está definida no intervalo $[a_i, b_i]$. Depois de uma reparametrização de cada curva, se necessário, podemos assumir sem perda de generalidade que $b_i = a_{i+1}$ para $i = 1, \ldots, n-1$.
Desta forma, temos que γ é contínua e está definida em um intervalo $[a,b]$, em que $a = a_1$ e $b = b_n$ e γ é diferenciável em cada intervalo (a_i, b_i). Seja

$$F(t) = \int_a^t \frac{\gamma'(s)}{\gamma(s) - \alpha} ds$$

Temos que F é contínua em $[a,b]$ e diferenciável $\forall t \neq a_i, b_i$. Então

$$F'(t) = \frac{\gamma'(t)}{\gamma(t) - \alpha}$$

Intuitivamente, $F(t) = \log(\gamma(t) - \alpha)$.
Note que

$$\frac{d}{dt} e^{-F(t)}(\gamma(t) - \alpha) = e^{-F(t)}\gamma'(t) - F'(t)e^{-F(t)}(\gamma(t) - \alpha)$$

$$= e^{-F(t)}\gamma'(t) - \frac{\gamma'(t)}{\gamma(t) - \alpha} e^{-F(t)}(\gamma(t) - \alpha) = 0$$

Então existe uma constante c tal que

$$e^{-F(t)}(\gamma(t) - \alpha) = c$$

ou seja,

$$\gamma(t) - \alpha = ce^{F(t)}$$

Como γ é uma curva fechada, $\gamma(a) = \gamma(b)$ e

$$ce^{F(b)} = \gamma(b) - \alpha = \gamma(a) - \alpha = ce^{F(a)}$$

Como $\gamma(a) - \alpha \neq 0$, então $c \neq 0$ e $e^{F(a)} = e^{F(b)}$. Desta forma, existe um

inteiro k tal que

$$F(b) = F(a) + 2\pi ik.$$

Como $F(a) = 0$, então $F(b) = 2\pi ik$. Assim, temos que

$$w(\gamma, \alpha) = \frac{1}{2\pi i} \int_\gamma \frac{\gamma'(t)}{\gamma(t) - \alpha} = \frac{2\pi ik}{2\pi i} = k$$

■

Proposição 2.5. *Seja γ um caminho. Então a função de α definida por*

$$\alpha \to \int_\gamma \frac{1}{z - \alpha} dz$$

para γ não passando por α é uma função contínua de α.

A demonstração desta proposição possui alguas tecnicalidades que fogem do escopo deste trabalho, pos isso deixamos como referência [2].

Proposição 2.6. *Seja γ um caminho fechado. Seja S um conjunto conexo que não intersecta γ. Então a função*

$$\alpha \to \frac{1}{2\pi i} \int_\gamma \frac{1}{z - \alpha} dz$$

é constante para α em S. Se S é não limitado, então essa constante é zero.

Demonstração: Das Proposições (2.4) e (2.5), sabemos que $w(\gamma, \alpha)$ é um número inteiro e é uma função contínua então ela é constante para alguma curva e consequentemente constante para um conjunto conexo por caminhos. Se S é não limitada , então para algum α suficientemente grande, o integrando $\frac{1}{|z-\alpha|}$ tem valor absoluto muito pequeno. Estimando esta integral, temos que $w(\gamma, \alpha) = 0$.■

O conceito de Winding Number pode ser ilustrado na figura (2.4), retirada de [11]. Mais detalhes podem ser encontrados em livros de referência de

Análise Complexa como [2] e [3], por exemplo.

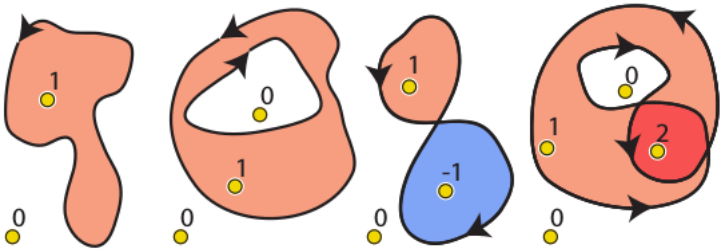

Figura 2.4: Winding Number para curvas. O valor 0 indica que o ponto em questão é exterior à curva. Valores positivos inidicam o número de voltas que a curva deu em torno do ponto no sentido anti-horário. Valores negativos indicam o número de voltas que a curva deu em torno do ponto no sentido horário. Imagem retirada de [11].

Aqui, vamos focar neste conceito para uma malha poligonal.

Partindo da definição de winding number,

$$w(\gamma,\alpha)=\frac{1}{2\pi i}\int_\gamma \frac{1}{z-\alpha}dz$$

para curvas, queremos generalizar esta definição para malhas poligonais. Podemos supor, sem perda de generalidade que $\alpha=0$. A integral de uma função f sobre uma curva γ é dada por

$$\int_\gamma f(\gamma(t))\gamma'(t)dt$$

Escrevendo z em sua forma polar, temos $z=r(t)e^{i\theta(t)}$. Assim,

$$\frac{1}{2\pi i}\int_\gamma \frac{1}{z}dz=\frac{1}{2\pi i}\int_a^b \frac{r'(t)e^{i\theta(t)}+i\theta'(t)r(t)e^{i\theta(t)}}{r(t)e^{i\theta(t)}}dt$$

$$=\frac{1}{2\pi i}\int_a^b \frac{r'(t)}{r(t)}+i\theta'(t)dt$$

$$= \frac{1}{2\pi}\int_a^b \theta'(t) - \frac{ir'(t)}{r(t)}dt$$

$$= \frac{1}{2\pi}\int_a^b \theta'(t)dt - \frac{i}{2\pi}\int_a^b \frac{r'(t)}{r(t)}dt$$

Mas

$$\frac{-i}{2\pi}\int_a^b \frac{r'(t)}{r(t)}dt = \frac{-i}{2\pi}(\log(r(b)) - \log(r(a))) = 0$$

pois γ é uma curva fechada. Desta forma,

$$\frac{1}{2\pi i}\int_\gamma \frac{1}{z}dz$$

é equivalente a

$$\frac{1}{2\pi}\int_a^b \theta'(t)dt = \frac{1}{2\pi}\oint_{\mathscr{C}} d\theta \tag{2.15}$$

Os valores 0 e maiores do que zero nos dizem se α está fora ou dentro de $\mathscr{C}$, respectivamente.

A integral em (2.15) fornece uma discretização exata se $\mathscr{C}$ for linear por partes

$$w(p) = \frac{1}{2\pi}\sum_{i=1}^{n} \theta_i \tag{2.16}$$

em que θ_i é o ângulo entre os vetores de dois vértices consecutivos c_i e c_{i+1} com vértice em p conforme podemos ver na figura (2.5). Adiante, vamos ver que $w(p)$ vai nos dar o sinal de um ponto da malha fina em relação à malha mais grosseira.

Dado um ponto p, vamos definir $u = c_i - p$ e $v = c_{i+1} - p$, em que $u = (u_x, u_y)^T$ e $v = (v_x, v_y)^T$. Então

$$\tan(\theta_i(p)) = \frac{det[u,v]}{\langle u, v\rangle} = \frac{u_x v_y - u_y v_x}{u_x v_x + u_y v_y} \tag{2.17}$$

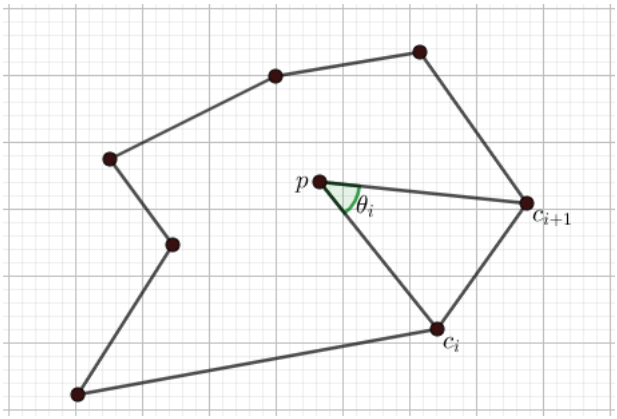

Figura 2.5: Ângulo entre dois vértices consecutivos.

Para determinar os ângulos θ_i, vamos usar a função atan2 do MatLab.

$$atan2(\theta_i(p)) = \frac{u_x v_y - u_y v_x}{u_x v_x + u_y v_y} = \frac{y}{x}$$

Caso 1: $x > 0$. Neste caso, devemos ter $\theta_i \in (-\frac{\pi}{2}, \frac{\pi}{2})$, conforme figura 2.6. Então $\theta_i = \arctan(\frac{y}{x})$.
Caso 2: $x < 0$ e $y < 0 \Rightarrow \tan(\theta_i) > 0$;
Neste caso, $\theta_i \in (-\pi, -\frac{\pi}{2})$ e $\theta_i = \arctan(\frac{y}{x}) - \pi$, que satisfaz $\tan(\theta_i) > 0$.
Caso 3: $x < 0$ e $y \geqslant 0 \Rightarrow \tan(\theta_i) \leqslant 0$
Neste caso, $\theta_i \in (\frac{\pi}{2}, -\pi]$ e

$$\theta_i = \arctan\left(\frac{y}{x}\right) + \pi$$

que satisfaz $\tan(\theta_i) \geqslant 0$.

Para os pontos tais que $x = 0$ e $y < 0$, definimos $\theta_i = -\frac{\pi}{2}$ e para $x = 0$ e $y > 0$ definimos $\theta_i = \frac{\pi}{2}$.

Desta forma, temos que

$$atan2(y,x) = \begin{cases} \arctan\left(\frac{y}{x}\right), & \text{se} \quad x > 0 \\ \arctan\left(\frac{y}{x}\right) - \pi & \text{se} \quad x < 0 \quad \text{e} \quad y < 0 \\ \arctan\left(\frac{y}{x}\right) + \pi & \text{se} \quad x < 0 \quad \text{e} \quad y \geqslant 0 \\ -\frac{\pi}{2} & \text{se} \quad x = 0 \quad \text{e} \quad y < 0 \\ \frac{\pi}{2} & \text{se} \quad x = 0 \quad \text{e} \quad y > 0 \end{cases}$$

Já vimos que $w(p)$ é um número inteiro. Como estamos considerando sempre malhas poligonais orientadas no sentido anti-horário, temos que $w(p)$ será sempre positivo ou zero. Desta forma, se $w(p) > 0$, então p está no interior da malha poligonal e definimos $s(p) = -1$. Se $w(p) = 0$, entãp p está no exterior da malha e definimos $s(p) = 1$.

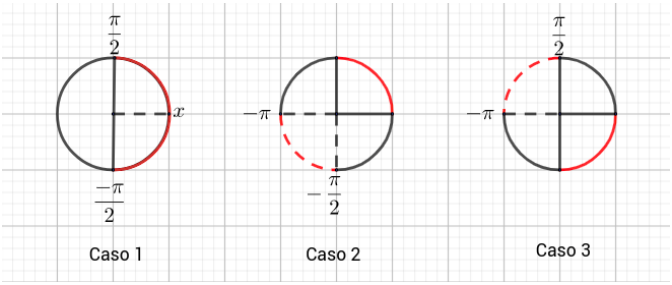

Figura 2.6: Ilustração de $atan2$

2.3 Projeção

Para encontrar a distância $d(p)$ de um um ponto p da malha fina até a malha mais grosseira $\hat{G}$, projetamos p ortogonalmente sobre as arestas A_i de $\hat{G}$.

Na figura (2.7), temos os pontos de quadratura da malha fina projetados sobre as arestas da malha grosseira.

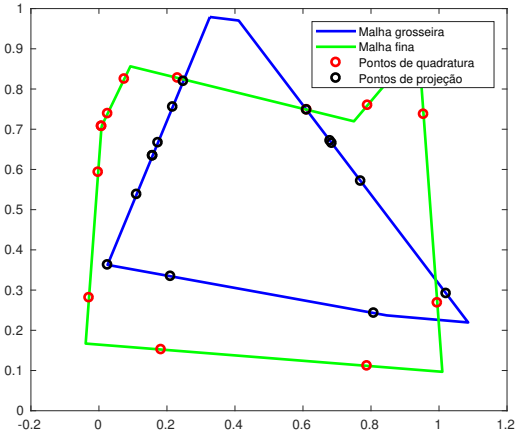

Figura 2.7: Projeção dos pontos de quadratura da malha fina sobre as arestas da malha grosseira. Cada ponto da malha grosseira (em preto) é a projeção ortogonal de um ponto de quadratura da malha fina (em vermelho).

Vamos considerar uma aresta A_i de $\hat{G}$ com vértices x_1 e x_2 e x_0 um ponto da malha fina. A distância $d(x_0, \hat{G})$ do ponto x_o à aresta $\hat{G}$ é dada por

$$d(x_0, \hat{G}) = min\{d(x_0, A_i) | A_i \in \hat{G}\}$$

A menor distância de x_0 para A_i será dada pela projeção ortogonal p_{x0}, conforme figura 2.8.

O ponto p_{x0} pertence a aresta A_i se $px_0 = x_1 + (x_2 - x_1)t$ tal que $0 \leqslant t \leqslant 1$. Queremos definir t tal que $\overline{x_0 p_{x0}}$ seja perpendicular a $\overline{x_1 x_2}$.

Temos que

$$\langle x_2 - x_1, x_0 - px_0 \rangle = 0$$

$$\langle x_2 - x_1, x_0 - (x_1 + (x_2 - x_1)t) \rangle = 0$$

$$\langle x_2 - x_1, x_0 - x_1 > - < x_2 - x_1, (x_2 - x_1)t \rangle = 0$$

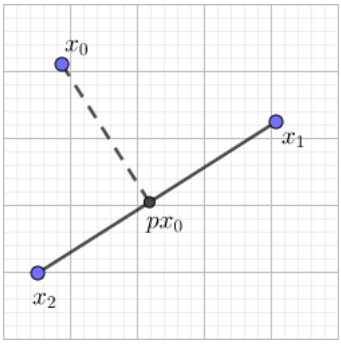

Figura 2.8: Projeção Ortogonal de x_o sobre a aresta $\overline{x_1x_2}$

$$\langle x_2 - x_1, x_0 - x_1 \rangle = < x_2 - x_1, x_2 - x_1 \rangle t$$

Então

$$t = \frac{\langle x_2 - x_1, x_0 - x_1 \rangle}{\langle x_2 - x_1, x_2 - x_1 \rangle}$$

Se $t < 0$ ou $t > 1$, a projeção cai fora do segmento $\overline{x_1x_2}$, por isso, temos que se $0 \leqslant t \leqslant 1$, $px_0 = x_1 + (x_2 - x_1)t$. Se $t < 0$, definimos $px_0 = x_1$ e se $t > 1$, definimos $px_0 = x_2$.

2.4 Etapa de Fluxo

Vamos considerar duas malhas poligonais $\bar{F}$ e $\hat{G}$ em que $\bar{F}$ é mais fina em relação a $\hat{G}$. Queremos mover todos os vértices de $\bar{F}$ ao longo de um fluxo que minimiza distâncias com sinal para $\hat{G}$ integrando sobre todos os pontos $\bar{p} \in \bar{F}$. Considere

$$\Phi(\bar{F}) = \int_{\bar{F}} s(\bar{p}) d(\bar{p}) dC, \tag{2.18}$$

em que $s(\bar{p})$ denota o sinal apropriado do ponto e $d(\bar{p})$ é a distância de $\bar{p}$ para a malha mais grosseira $\hat{G}$.

Vamos denotar por $\bar{f}$ um vértice de $\bar{F}$. Como queremos minimizar distâncias,

minimizamos Φ tomando pequenos passos de tempo na direção oposta ao gradiente. O fluxo na direção oposta ao gradiente é dado por, para cada vértice $\bar{f} \in \bar{F}(t)$:

$$\frac{\partial \bar{f}}{\partial t} = -\nabla_{\bar{f}} \Phi(\bar{F}) \tag{2.19}$$

Discretizando o lado esquerdo de (2.19),

$$\frac{\partial \bar{f}}{\partial t} \approx \frac{\bar{f}(t+\Delta t) - \bar{f}(t)}{\Delta t}.$$

Então, pelo método de Euler explícito

$$-\nabla_{\bar{f}} \Phi(\bar{F}) \approx \frac{\bar{f}(t+\Delta t) - \bar{f}(t)}{\Delta t}$$

$$\bar{f}(t+\Delta t) \approx -\nabla_{\bar{f}} \Phi(\bar{F}) \Delta t + \bar{f}(t)$$

Uma vez que Φ é uma função não-linear e difícil de calcular numericamente, aproximamos o gradiente usando um método de quadratura. Aqui vamos usar a Quadratura Gaussiana, conforme descrito na Seção (2.1).

Considere uma aresta A_i de $\bar{F}$ com vértices a e b. Escrevemos s e d em pontos de quadratura $\bar{p}_{ij}$ e pesos correspondentes w_{ij}, com $j = 1, \ldots, h$ em que h, representa o número de pontos de quadratura. Então,

$$\Phi(\bar{F}) = \sum_{i=1}^{m_F} \int_{\bar{p}_j \in A_i} s(\bar{p}) d(\bar{p}) dC \approx \sum_{i=1}^{m_F} \sum_{j=1}^{h} w_{ij} s(\bar{p}_{ij}) d(\bar{p}_{ij}) \tag{2.20}$$

em que m_F é o número de arestas de $\bar{F}$. Podemos escrever cada $\bar{p}$ como uma combinaçao linear dos vértices da sua respectiva aresta, ou seja, se $\bar{p}$ pertence a aresta de vértices $\bar{f}_a$ e $\bar{f}_b$, então

$$\bar{p}_{ij} = \lambda_a \bar{f}_a + \lambda_b \bar{f}_b \tag{2.21}$$

em que λ_a e λ_b são dados pela solução do sistema

$$\begin{cases} \bar{f}_{ax}\lambda_a + \bar{f}_{b_x}\lambda_b &= \bar{p}_{jx} \\ \bar{f}_{ay}\lambda_a + \bar{f}_{b_y}\lambda_b &= \bar{p}_{jy} \end{cases}$$

Por simplicidade, vamos descrever o método usando $h = 2$. Desta forma temos

$$\Phi(\bar{F}) \approx \sum_{i=1}^{m_F} (w_{i1}s(p_{i1})d(p_{i1}) + w_{i2}s(p_{i2})d(p_{i2})) \tag{2.22}$$

Para contornar a dificuldade de diferenciar a função distância, assumimos que para cada ponto de quadratura $\bar{p}_j$, o ponto $\bar{q}_j$ mais próximo a ele na malha mais grosseira e o sinal $s(\bar{p}_j)$ permanecem constantes em cada passo de tempo de fluxo, ou seja, $\bar{q}_j = \bar{q}_j^*$ e $s(\bar{p}_j) = s_j^*$.
Desta forma, para cada vértice $\bar{f}$ de $\bar{F}$, temos o gradiente de Φ escrito como um somatório de termos.

$$\nabla_{\bar{f}}\Phi(\bar{F}) \approx \sum_{i=1}^{m_F}\sum_{j=1}^{h} w_{ij}s_{ij}^*\nabla_{\bar{f}}||p_{ij} - q_{ij}^*|| \tag{2.23}$$

Pela Regra da Cadeia

$$\nabla_{\bar{f}}\Phi(\bar{F}) \approx \sum_{i=1}^{m_{\bar{F}}}\sum_{j=1}^{h} w_{ij}s_{ij}^*(J_{\bar{f}}\bar{p}_{ij})^T\nabla_{\bar{p}_{ij}}||\bar{p}_{ij} - \hat{q}_{ij}^*|| \tag{2.24}$$

Por (2.21),como $\bar{p}_{ij} = \lambda_a\bar{f}_a + \lambda_b\bar{f}_b$, então

$$J_{\bar{f}}\bar{p}_{ij} = \begin{pmatrix} \lambda_a & 0 \\ 0 & \lambda_a \end{pmatrix},$$

se $\bar{f} = \bar{f}_a$;

$$J_{\bar{f}}\bar{p}_{ij} = \begin{pmatrix} \lambda_b & 0 \\ 0 & \lambda_b \end{pmatrix},$$

se $\bar{f} = \bar{f}_b$; e

$$J_{\bar{f}}\bar{p}_{ij} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix},$$

se $\bar{f} \neq \bar{f}_b$ e $\bar{f} \neq \bar{f}_a$.

Temos que

$$\nabla_{\bar{p}_{ij}} ||\bar{p}_{ij} - \hat{q}^*_{ij}||^2 = 2(\bar{p}_{ij} - \hat{q}^*_{ij}) \tag{2.25}$$

Assim, escrevendo

$$||\bar{p}_j - \hat{q}^*_j|| = \sqrt{||\bar{p}_j - \hat{q}^*_j||^2}$$

Então, por (2.25),

$$\nabla_{\bar{p}_{ij}} ||\bar{p}_{ij} - \hat{q}^*_{ij}|| = \nabla_{\bar{p}_{ij}} \sqrt{||\bar{p}_{ij} - \hat{q}^*_{ij}||^2} = \frac{2(\bar{p}_{ij} - \hat{q}^*_{ij})}{2\sqrt{||\bar{p}_{ij} - \hat{q}^*_{ij}||^2}} = \frac{\bar{p}_{ij} - \hat{q}^*_{ij}}{||\bar{p}_{ij} - \hat{q}^*_{ij}||}$$

Assim, temos

$$\nabla_{\bar{f}}\Phi(\bar{F}) \approx \sum_{i=1}^{m_{\bar{F}}} \sum_{j=1}^{h} w_{ij} s^*_{ij} (J_{\bar{f}}\bar{p}_{ij})^T \nabla_{\bar{p}_{ij}} ||\bar{p}_{ij} - \hat{q}^*_{ij}|| \approx \sum_{i \in N(\bar{f})} \sum_{j=1}^{h} w_{ij} \lambda_{\bar{f}} s^*_{ij} \frac{\bar{p}_{ij} - \hat{q}^*_{ij}}{||\bar{p}_{ij} - \hat{q}^*_{ij}||} \tag{2.26}$$

em que $N(\bar{f})$ é o conjunto de arestas adjacentes a $\bar{f}$. Na figura 2.10, vemos um exemplo em que as malhas grosseira e fina possuem 7 e 8 vértices, respectivamente. Na figura 2.10, podemos observar algumas etapas de fluxo, com,45, 90, 150 e 195 passos de fluxo.

Após o fluxo, obtemos um estado final em que a malha fina está totalmente no interior da malha grosseira. Nosso objetivo agora é reverter o fluxo

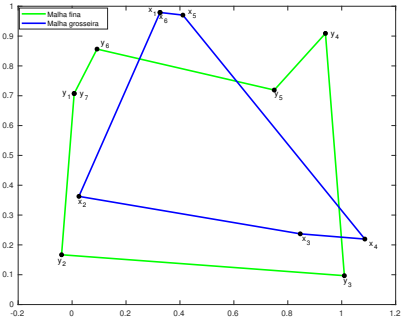

Figura 2.9: Malhas fina e grosseira com intersecções entre si antes do fluxo.

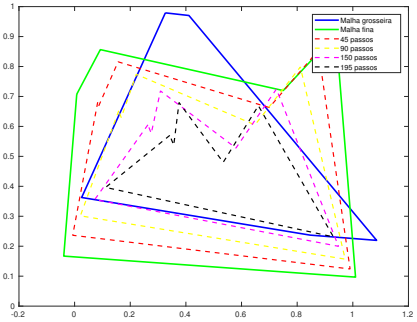

Figura 2.10: Fluxo com 45, 90,150 e 195 passos

em que a malha fina volta para sua posição inicial empurrando a malha grosseira. Vamos ver como fazer isso no próximo capítulo.

3 *Re-inflação*

Nesta fase já temos a malha fina totalmente no interior da malha grosseira. Este capítulo trata da etapa de re-inflação, que consiste em levar a malha fina de volta para a sua posição inicial empurrando a malha grosseira e detectando e respondendo colisões em cada passo de tempo. Este processo é ilustrado na Figura 3.1. Ao final, a malha grosseira será totalmente exterior à malha fina. Para cada passo que a malha fina andou no processo de fluxo, ela deve andar de volta no processo de re-inflação. Na Seção 3.1, vamos abordar o conceito de detecção de colisão, baseado em [8]. Na Seção 3.2 vamos tratar do problema de resposta a colisões com restrições de desigualdade e igualdade, baseado em [9] e [10]. Na Seção 3.3 vamos descrever o problema de otimização, que será resolvido para levar a malha fina de volta para sua posição inicial empurrando a malha grosseira, seguido da rotina *fmincon* do MatLab, que será usada para resolver o problema de otimização. Finalmente, na Seção 3.4, vamos tratar das diferenças com o método proposto por Sacht *et al* [1] no artigo intitulado *Nested Cages*, que trata de malhas triangulares exteriores em 3 dimensões.

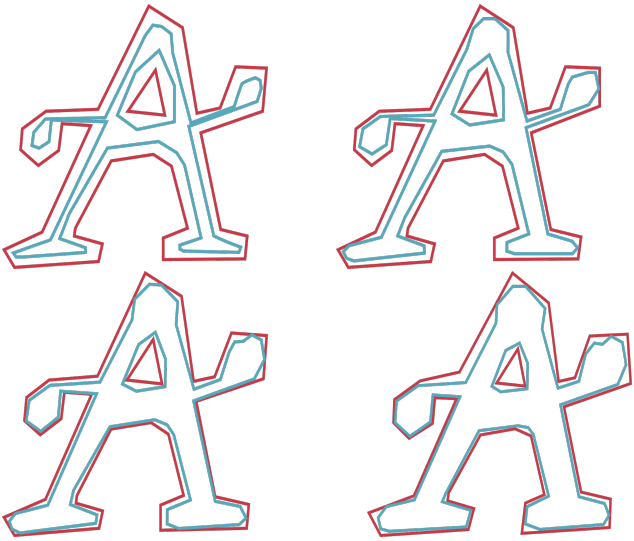

Figura 3.1: Ilustração do processo de re-inflação, obtendo um estado final com a malha fina totalmente no interior da malha grosseira. Imagem oriunda de [1].

3.1 Detecção de colisões

A detecção de colisões consiste em identificar se uma ou mais colisões ocorrem em um dado intervalo de tempo. Aqui vamos tratar do caso de colisões ponto-aresta, quando um vértice da malha fina colide com uma aresta da malha grosseira.

Considere o instante de tempo t_0 como o instante em que ocorre uma colisão no intervalo de tempo $[t_0, t_0 + \Delta t]$. Conhecendo o tempo e a velocidade de cada ponto no tempo t_0 é possível determinar sua posição no instante de tempo $t_0 + \Delta t$.

Sejam $P(t) = p_0 + t\overrightarrow{v}$ um vértice em movimento da malha fina e A e B as extremidades de uma aresta da malha grosseira e seja $\overrightarrow{v}$ a velocidade de $P(t)$ no intervalo de tempo $[t_0, t_0 + \Delta t]$. Se houver colisão, então existe $t \in [t_0, t_0 + \Delta t]$

tal que $P(t) \in \overline{AB}$, ou seja, existe $u \in [0,1]$ tal que

$$\overline{AP}(t) = u\overline{AB} \tag{3.1}$$

Uma vez que $P(t)$ deve pertencer à aresta no momento da colisão, então a condição

$$\langle \overline{AP}(t), N \rangle = 0 \tag{3.2}$$

deve ser satisfeita, em que N e a normal da aresta. Definindo

$$\overline{AB} = \begin{pmatrix} B_x - A_x \\ B_y - A_y \end{pmatrix}$$

podemos escrever N como

$$N = \begin{pmatrix} B_y - A_y \\ -B_x + A_x \end{pmatrix}$$

Escrevendo $P(t) = p_0 + tv$, em que p_0 é o ponto inicial, devemos ter

$$\langle (p_0 + tv - A)^T, N \rangle = 0 \tag{3.3}$$

$$\Updownarrow$$

$$((p_0 + tv - A)_x, (p_0 + tv - A)_y) \begin{pmatrix} B_y - A_y \\ -B_x + A_x \end{pmatrix} = 0$$

$$\Updownarrow$$

$$(p_0 + tv - A)_x.(B_y - A_y) + (p_0 + tv)_y.(-B_x + A_x) = 0$$

$$\Updownarrow$$

$$p_{0x}(B_y - A_y) + tv_x(B_y - A_y) - A_x(B_y - A_y) + p_{0y}(-B_x + A_x) + tv_y(-B_x + A_x)$$
$$-A_y(-B_x + A_x) = 0$$

$$\Updownarrow$$

$$t(v_x(B_y-A_y)+v_y(-B_x+A_x))+p_{0x}(B_y-A_y)-A_x(B_y-A_y)+p_{0y}(-B_x+A_x)$$
$$-A_y(-B_x+A_x)=0$$

$$\Updownarrow$$

$$t=\frac{(A_x-p_{0x})(B_y-A_y)+(A_y-p_{0y})(-B_x+A_x)}{v_x(B_y-A_y)+v_y(-B_x+A_x)} \tag{3.4}$$

Se $t \in [t_0, t_0+\Delta t]$ então, substituindo t em (3.1), podemos determinar $0 \leqslant u \leqslant 1$ tal que o par (t,u) é a solução de (3.1).

3.2 Resposta a colisões

Uma vez detectadas todas as colisões que ocorrem em um dado passo de tempo, deve-se responder a elas. A resposta à colisão deve levar em conta que a malha fina deve voltar a sua posição anterior em cada passo de tempo, ao mesmo tempo que empurra a malha grosseira, de forma que esta fique exterior à malha fina em cada passo de tempo. Para que isto seja satisfeito, adicionamos restrições de igualdade e desigualdade ao problema. Para ilustrar, considere as malhas conforme a Figura 3.2. Vamos tratar do problema de re-inflação para estas malhas.

Após o fluxo, obtemos um estado em que a malha fina está totalmente interior a malha grosseira, conforme Figura 3.3. Considere

$$x^z=(x^z_{1x},x^z_{2x},\ldots,x^z_{6x},x^z_{1y},x^z_{2y},\ldots,x^z_{6y})^T \in \mathbb{R}^{12}$$

e

$$y^z=(y^z_{1x},y^z_{2x},\ldots,y^z_{7x},y^z_{1y},y^z_{2y},\ldots,y^z_{7y})^T \in \mathbb{R}^{14}$$

os vetores contendo todas as posições dos vértices das malhas grosseira e fina respectivamente após z passos do fluxo. Por exemplo, (y^z_{1x},y^z_{1y}) é a posição do ponto y_1 na figura 3.3.

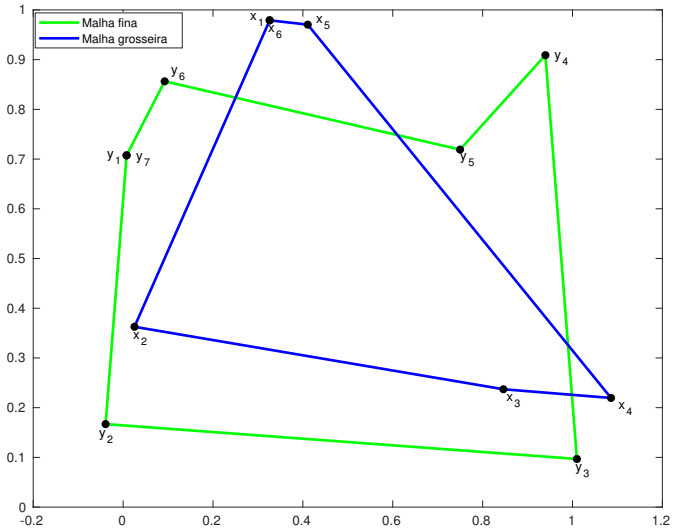

Figura 3.2: Estado inicial das duas malhas, anterior ao fluxo

Queremos determinar um novo vetor de posições

$$X = \begin{bmatrix} x \\ y \end{bmatrix}$$

após revertermos z passos de fluxo satisfazendo as seguintes condições:

- A malha fina deve voltar para a sua posição inicial y^0;

- A malha fina deve empurrar a malha grosseira, de forma que esta fique totalmente exterior à malha fina.

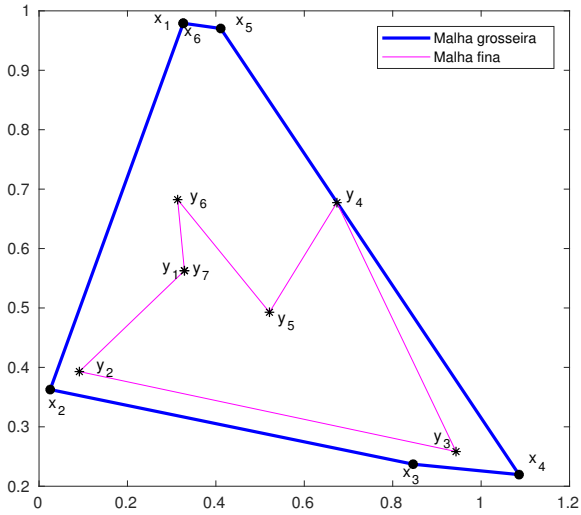

Figura 3.3: Malha fina interior à malha grosseira após o fluxo

3.2.1 Restrições de igualdade

Para que isso aconteça, a primeira condição deve ser satisfeita em cada passo z, ou seja, a malha fina deve voltar para a posição que tinha no passo anterior do fluxo, empurrando a malha grosseira em cada passo. Para garantir a primeira condição devemos ter

$$\begin{bmatrix} 0_{14\times 12} & Id_{14\times 14} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} y^{z-1} \end{bmatrix}$$

Definindo $A_{eq} = \begin{bmatrix} 0_{14\times 12} & Id_{14\times 14} \end{bmatrix}$ e $b_{eq} = \begin{bmatrix} y^{z-1} \end{bmatrix}$ devemos ter

$$A_{eq}X = b_{eq} \tag{3.5}$$

Esta restrição de igualdade irá garantir que em cada passo a malha fina volte para a sua posição anterior. Além disso, impomos que $x_{1x} - x_{6x} = 0$ e $x_{1y} - x_{6y} = 0$ para que a malha permaneça fechada durante a re-inflação. Uma vez que impomos que a malha fina volte para a sua posição anterior ela deve empurrar a malha grosseira de forma que não ocorra interpenetração entre um vértice de $\bar{F}$ e uma aresta de $\hat{G}$.

3.2.2 Restrições de Desigualdade

A Figura 3.3 nos mostra o estado final das duas malhas após o fluxo. No primeiro passo de re-inflação é possível encontrar um par (t,u) satisfazendo (3.1) tal que o vértice $y_4 \in \bar{F}$ colide com a aresta $\overline{x_4x_5}$ de $\hat{G}$. Vamos definir $p_a = y_4$ e p_b o ponto da aresta $\overline{x_4x_5}$ em que ocorre a colisão. Para vizualização deste caso, veja Figura 3.4. Para garantir que não ocorra interpenetração entre o vértice e a aresta, devemos impor que

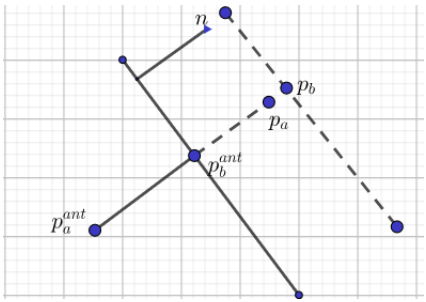

Figura 3.4: Ilustração da restrição de desigualdade. N representa o vetor normal da aresta. Os pontos p_a^{ant} e p_b^{ant} são os pontos antes de reverter o passo de fluxo. p_a e p_b são as novas posições de p_a^{ant} e p_b^{ant}, respectivamente, após resolver reverter o fluxo em um dado passo.

$$\langle N, (p_a - p_b) \rangle \leqslant 0 \tag{3.6}$$

em que n é a normal exterior de $\overline{x_4x_5}$. Vamos denotar $N = \begin{pmatrix} n_1 \\ n_2 \end{pmatrix}$. Como $p_b \in \overline{x_4x_5}$, podemos escrevê-lo como $p_b = ux_4 + (1-u)x_5$, em que u é dado em (3.1). Assim, devemos ter

$$\langle N, (p_a - p_b) \rangle \leqslant 0$$

$$\Updownarrow$$

$$\begin{pmatrix} n_1 & n_2 \end{pmatrix} \begin{pmatrix} (p_a - p_b)_x \\ (p_a - p_b)_y \end{pmatrix} \leqslant 0$$

$$\Updownarrow$$

$$\begin{pmatrix} n_1 & n_2 \end{pmatrix} \begin{pmatrix} y_{4x} - ux_{4x} - (1-u)x_{5x} \\ y_{4y} - ux_{4y} - (1-u)x_{5y} \end{pmatrix} \leqslant 0$$

$$\Updownarrow$$

$$n_1y_{4x} - un_1x_{4x} - (1-u)n_1x_{5x} + n_2y_{4y} - un_2x_{4y} - (1-u)n_2x_{5y} \leqslant 0.$$

Em forma matricial, devemos ter

$$\begin{pmatrix} A_1 & A_2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \leqslant 0$$

em que

$$A_1 = (0,0,0,-un_1,-(1-u)n_1,0,0,0,0,-un_2,-(1-u)n_2,0,0)$$

e

$$A_2 = (0,0,n_1,0,0,0,0,0,0,n_2,0,0,0)$$

ou seja,

$$\Updownarrow$$

$$AX \leqslant b$$

em que $A = (A_1, A_2)$, $X = \begin{pmatrix} x \\ y \end{pmatrix}$ e $b = 0$.

No exemplo que usamos para ilustrar as restrições de desigualdade, ocorre apenas uma colisão no primeiro passo. Por isso, a matriz A possui apenas uma linha. Para outros problemas, pode acontecer mais de uma colisão em cada passo de re-inflação. Cada colisão define uma nova linha de A e b construída de maneira idêntica à que acabamos de detalhar.

3.2.3 Colisão aresta-ponto

No exemplo da Figura 3.3, todos os casos de colisão que ocorreram se tratavam de colisões do tipo ponto-aresta, ou seja, quando um ponto $p \in \bar{F}$ colide com uma aresta $\overline{AB} \in \hat{G}$.
Vamos considerar agora o caso de colisão aresta-ponto, quando uma aresta $\overline{AB} \in \bar{F}$ colide com um ponto $p \in \hat{G}$. Um exemplo deste caso encontra-se nas Figuras 3.5 e 3.6. A Figura 3.6 nos mostra o estado final das duas malhas da figura 3.5 após o fluxo em que, ao reverter o último passo acontece apenas uma colisão do tipo aresta-ponto.

Vamos considerar A e B os vértices de uma aresta $\overline{AB} \in \bar{F}$ e $p \in \hat{G}$ um vértice da malha grosseira. Para que ocorra a colisão, devemos encontrar $t \in [0,1]$ tal que

$$\overline{A(t)p} = u\overline{A(t)B(t)} \tag{3.7}$$

No momento da colisão, a condição

$$\langle \overline{A(t)p}, N(t) \rangle = 0 \tag{3.8}$$

deve ser satisfeita, em que $N(t)$ é a normal da aresta $\overline{A(t)B(t)}$ em cada passo de tempo.

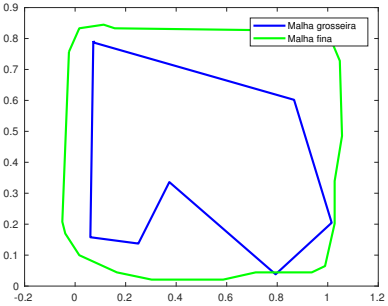

Figura 3.5: Estado inicial antes do fluxo

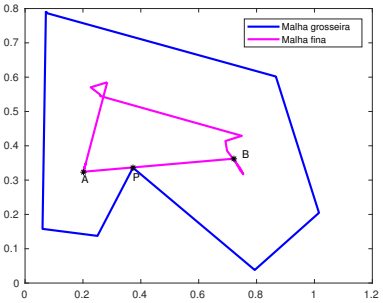

Figura 3.6: Estado final após o fluxo. Caso de colisão aresta-ponto

Defina $A(t) = A_0 + tv_A$ e $B(t) = B_0 + tv_B$ em que v_a e v_b são as velocidades dos vértices A e B respectivamente. Desta forma,

$$\overline{A(t)B(t)} = \begin{pmatrix} B_{0x} + tv_{Bx} - A_{0x} - tv_{Ax} \\ B_{0y} + tv_{By} - A_{0y} + tv_{Ay} \end{pmatrix}$$

e podemos escrever $N(t)$ como

$$N(t) = \begin{pmatrix} B_{0y} + tv_{By} - A_{0y} + tv_{Ay} \\ -B_{0x} - tv_{Bx} + A_{0x} + tv_{Ax} \end{pmatrix}$$

Assim, de acordo com (3.8), devemos ter

$$\langle (p - A_0 - tv_A), N(t) \rangle = 0$$

$$\Updownarrow$$

$$((p_x - A_{0x} - tv_{A_x}), (p_y - A_{0y} - tv_{Ay})) \begin{pmatrix} B_{0y} + tv_{By} - A_{0y} + tv_{Ay} \\ -B_{0x} - tv_{Bx} + A_{0x} + tv_{Ax} \end{pmatrix} = 0$$

$$\Updownarrow$$

$$p_x B_{0y} + t p_x v_{By} - p_x A_{0y} - t p_x v_{Ay} - A_{0x} B_{0y} - t A_{0x} v_{By} + A_{0x} A_{0y} + t A_{0x} v_{Ay} - t B_{0y} v_{Ax}$$

$$-t^2 v_{Ax} v_{By} + t v_{Ax} A_{0y} + t^2 v_{Ax} v_{Ay} - p_y B_{0x} - t p_y v_{Bx} + p_y A_{0x} + t p_y v_{Ax} + A_{0y} B_{0x} + t A_{0y} v_{Bx}$$

$$-A_{0x} A_{0y} - t v_{Ax} A_{0y} + t B_{0x} v_{Ay} + t^2 v_{Bx} v_{Ay} - t v_{Ay} A_{0x} - t^2 v_{Ax} v_{Ay} = 0$$

$$\Updownarrow$$

$$t^2 (v_{Bx} v_{Ay} - v_{Ax} v_{By}) + t(p_x v_{By} - p_x v_{Ay} - A_{0x} v_{By} - B_{0y} v_{Ax} - p_y v_{Bx} + p_y v_{Ax} + A_{0y} v_{Bx}$$

$$+B_{0x} v_{Ay}) + p_x B_{0y} - p_x A_{0y} + p_y A_{0x} - p_y B_{0x} - A_{0x} B_0 y + A_{0y} B_{0x} = 0$$

que é uma equação de segundo grau na variável t. Desta forma, definindo

$$\tilde{a} = (v_{Bx} v_{Ay} - v_{Ax} v_{By})$$

$$\tilde{b} = p_x v_{By} - p_x v_{Ay} - A_{0x} v_{By} - B_{0y} v_{Ax} - p_y v_{Bx} + p_y v_{Ax} + A_{0y} v_{Bx}$$

$$+B_{0x} v_{Ay}$$

e

$$\tilde{c} = p_x B_{0y} - p_x A_{0y} + p_y A_{0x} - p_y B_{0x} - A_{0x} B_0 y + A_{0y} B_{0x}$$

as raízes da equação de segundo grau são dadas por

$$t_1 = \frac{-\tilde{b} - \sqrt{\tilde{b}^2 - 4\tilde{a}\tilde{c}}}{2\tilde{a}}$$

e

$$t_2 = \frac{-\tilde{b} + \sqrt{\tilde{b}^2 - 4\tilde{a}\tilde{c}}}{2\tilde{a}}$$

Primeiro verificamos se t_1 e t_2 estão no intervalo $[0,1]$. Em caso afirmativo, se apenas um deles estiver no intervalo tomamos t como sendo o valor no intervalo. Se houver dois, tomamos $t = \min(t_1, t_2)$. Substituindo t em (3.7), encontramos $u \in [0,1]$ tal que o par (t,u) é solução de (3.7). Para cada caso de colisão aresta-ponto, é definida uma restrição do tipo

$$\langle N, (p_b - p_a) \rangle \leqslant 0,$$

em que n é a normal da aresta ao fim do passo, p_a é o ponto da aresta em questão e p_b o vértice da malha grosseira. Para cada colisão do tipo aresta-ponto é definida uma nova linha da matriz de restrições de desigualdade, como abordado na Seção 3.2.1.

3.3 Problema de Otimização

Uma vez que temos todas as restrições que devem ser satisfeitas, nosso problema consiste em minimizar uma função energia $\mathscr{E}(X)$ em cada passo satisfazendo as restrições de igualdade e desigualdade impostas na Seção 3.2, ou seja, queremos

$$\begin{aligned} &\underset{s.a}{\text{minimizar}}\, \mathscr{E}(X) \\ &AX \leqslant b \\ &A_{eq}X = b_{eq} \end{aligned} \tag{3.9}$$

Aqui, definimos

$$\mathscr{E}(X) = ||X - X_{ant}||^2 \tag{3.10}$$

em que X_{ant} é o vetor com todas as posições do passo anterior da re-inflação. Esse problema de minimização é resolvido em cada passo de re-inflação. O

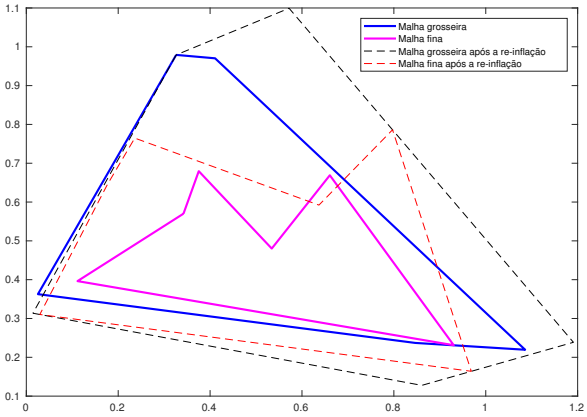

Figura 3.7: Estado das duas malhas após 95 passos de re-inflação

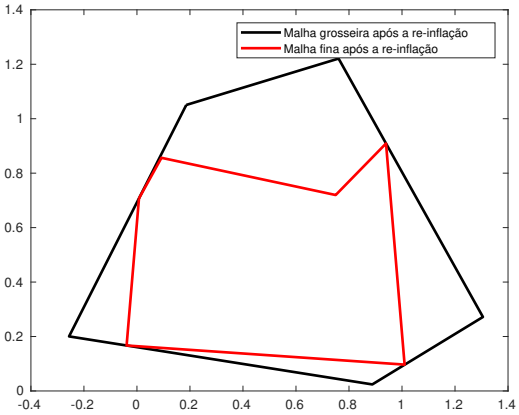

Figura 3.8: Estado final das duas malhas após a re-inflação em que a malha grosseira está totalmente exterior a malha fina

exemplo da Figura 3.2 faz o fluxo da malha fina para o interior da malha grosseira em 195 passos, até atingir o estado final como mostrado na Figura

3.3. Na Figura 3.7 apresentamos o estado das duas malhas após 95 passos e na Figura 3.8, temos o estado final das duas malhas após todos os passos da re-inflação. Neste estado temos a malha grosseira totalmente exterior a malha fina. O problema (3.9) é resolvido usando o solver *fmincon* do MatLab. *fmincon* encontra o mínimo de uma função de várias variáveis de um problema com restrições. No nosso caso, queremos determinar

$$X = fmincon(\mathscr{E}(X), A, b, A_{eq}, b_{eq})$$

Esta rotina é implementada para quatro diferentes algoritmos, são eles, pontos interiores, programação quadrática sequencial(SQP), restrições ativas e região de confiança. Neste trabalho, selecionamos pontos interiores, que funciona bem para os nossos exemplos. Uma abordagem mais detalhada onde compara a eficiência e viabilidade de cada método é deixada como trabalho futuro. Detalhes sobre cada método podem ser encontrados em [5].

3.4 Nested cages

O trabalho *Nested Cages*, proposto em 2015 por Sacht *et al* trata de malhas triangulares exteriores em 3 dimensões. A implementação de Nested Cages consiste em encontrar X que satisfaça todas as restrições, caminhando na direção oposta ao gradiente da função objetivo e projetando no conjunto viável, até obter a convergência. Isto é feito em dois passos:

1. $step : X \leftarrow X - \alpha \nabla \mathscr{E}(X)$;
2. Project: Remove todas as colisões no novo estado.

Em ambos os casos para a remoção das colisões acima, foi usada a biblioteca de simulações físicas *eltopo* disponível no link `https://github.com/tysonbrochu/eltopo`, que trabalha apenas com malhas triangulares em 3D

e, por isso, não foi usada neste trabalho. Esta metodologia se inspira no método do Gradiente Projetado, cujos detalhes podem ser encontrados em [5],mas a adaptação do método usando simulações físicas para determinar estados viáveis não tem garantias de convergência, visto que não minimiza nenhuma função objetivo. Com os métodos implementados por *fmincon* temos melhores garantias de convergência.

4 Resultados e Códigos

Neste capítulo apresentamos alguns resultados numéricos testados com malhas de diferentes tamanhos, bem como os códigos implementados. Estes códigos estão publicamente disponíveis no link `http://mtm.ufsc.br/~leo/dissertacao_aline.zip`.

4.1 Resultados

Aqui vamos apresentar alguns resultados numéricos. A implementação foi feita em linguagem de programação *MatLab*® e os teste feitos em uma máquina com processador Intel Core i5-6200U 2.3GHz e memória 8GB DDR3 L.

Iniciamos apresentando a Tabela 4.1 que apresenta o tempo de fluxo e re-inflação e o número de passos dados para malhas com diferentes números de vértices. Para melhor entendimento da tabela, os seguintes valores representam:

- v_{pqs}: Número de vértices da malha grosseira;
- v_{ps}: Número de vértices da malha fina;

- t_f: tempo de fluxo (em segundos);
- t_r: tempo de re-inflação (em segundos);
- z: Número de passos de fluxo e re-inflação.

Para gerar os resultados apresentados na Tabela 4.1, adotamos o passo de tempo para o fluxo $\Delta t = 10^{-3}$ e número de pontos de quadratura $n = 2$ para cada aresta.

linha	v_{psq}	v_{ps}	t_f	t_r	z
1	14	18	3.1457	41.6390	374
2	12	37	2.0677	18.7905	114
3	10	50	2.4493	26.6275	121
4	28	129	25.1687	39.1442	163
5	27	208	16.8370	21.1772	69
6	9	190	3.5635	10.2439	54
7	9	150	2.7518	12.3372	74
8	35	113	175.6568	228.7938	990

Tabela 4.1: Testes numéricos

Pode-se notar na Tabela 4.1 que o número de passos z independe do número de vértices das duas malhas. Como por exemplo nas linhas 1, em que as malhas grosseira e fina possuem 14 e 18 vértices respectivamente fazendo 374 passos de fluxo, enquanto na linha 7, as malhas grosseira e fina possuem 9 e 150 vértices, respectivamente, fazendo 74 passos de fluxo. A quantidade de passos depende apenas da geometria das malhas, se estas posssuem mais ou menos concavidades, como ilustrado nas Figuras 4.1 e 4.2.

Até aqui, todos os teste envolviam apenas um par de malhas. Veremos agora um exemplo envolvendo várias malhas, cada malha exterior a anterior. Este exemplo é ilustrado na figura 4.3.

Quando trabalhamos com mais de duas malhas, o processo de fluxo e

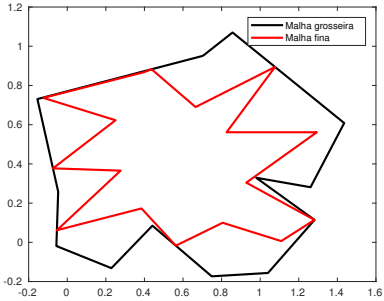

Figura 4.1: Malhas grosseira e fina com 14 e 18 vértices respectivamente após 374 passos de fluxo.

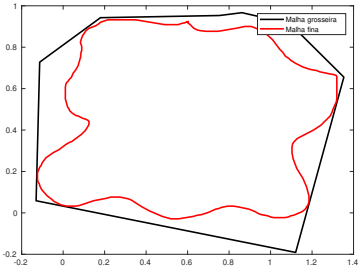

Figura 4.2: Malhas grosseira e fina com 9 e 150 vértices respectivamente após 74 passos de fluxo

re-inflação é feito aos pares. Em nosso exemplo com 4 malhas, na figura 4.3, iniciamos com as malhas M_1 e M_2, em que M_1 é mais fina que M_2. Fazemos o fluxo de M_1 para o interior de M_2 e em seguida, a re-inflação de M_1 empurrando M_2. Uma vez que M_1 retorna a sua posição inicial e M_2 está totalmente exterior a M_1, contruímos uma nova malha M_3, que seja mais grosseira que M_2. Fazemos o fluxo de M_2 para o interior de M_3 e em seguida a re-inflação, de forma que M_3 fique totalmente exterior a M_2. Após a re-inflação, M_2 voltou para sua posição inicial, que era exterior a M_1. Desta forma, temos que

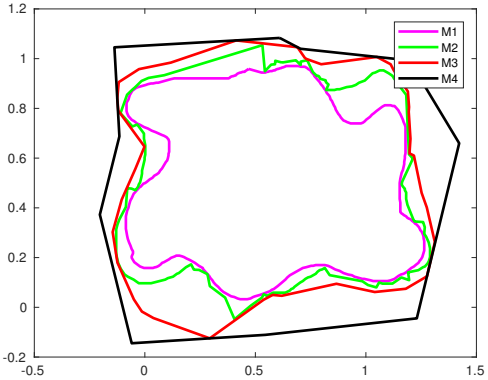

Figura 4.3: Sequência de malhas exteriores em que cada uma é exterior e tem menor vértices do que a anterior.

M_3 é exterior a M_2 e esta exterior a M_1. O mesmo acontece entre M_3 e M_4.

Na Tabela 4.2 apresentamos os resultados numéricos referentes a este exemplo.

Par	v_{ps}	v_{psq}	t_f	t_r	z
M_1M_2	260	146	313.4135	23.4506	47
M_2M_3	146	38	226.2827	24.9817	73
M_3M_4	38	11	303.1580	14.3799	98

Tabela 4.2: Resultados numéricos de tempo de fluxo, re-inflação e número de passos envolvendo 4 malhas.

Apesar de funcionar bem para malhas convexas ou com concavidades não muito acentuadas, o sucesso do método não é garantido para todos os casos. Isso deve ao fato de o fluxo levar pontos de quadratura da malha fina para o interior da malha grosseira, mas isto não garante que a malha fina fique totalmente interior. Uma solução (parcial) para este problema seria usar uma quantidade maior de pontos de quadratura por aresta.

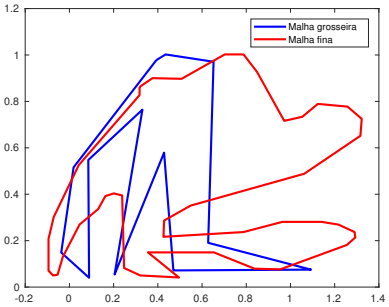

Figura 4.4: Estado inicial das duas malhas com muitas concavidades.

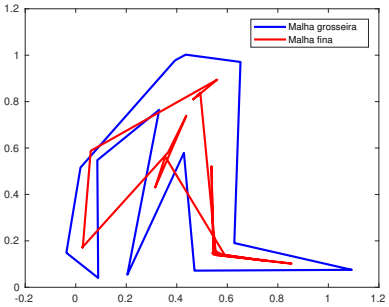

Figura 4.5: Estado final das duas malhas após 1500 passos de fluxo.

Na Figura 4.4, podemos ver um exemplo em que as malhas grosseira e fina possuem 14 e 44 vértices, respectivamente. Paramos a execução do fluxo após 1500 passos, obtendo o estado das malhas conforme a figura 4.5, em que a malha fina não é interior a malha grosseira. Isto acontece porque este método leva os pontos de quadratura para o interior da malha grosseira, mas isto não garante que a malha toda seja interior.

4.2 Códigos

Esta seção apresenta os códigos das etapas de fluxo e re-inflação, bem como os códigos auxiliares, como quadratura, winding number, projeção e gradiente, discutidos nos Capítulos 2 e 3, implementados na linguagem de programação *MatLab*®.
Inicialmente definimos

- *ps* =matriz contendo todas as posições dos vértices da malha fina $\bar{F}$
- *psq* =matriz contendo todas as posições dos vértices da malha grosseira $\hat{G}$

Veremos mais adiante como definir essas matrizes.

Iniciamos apresentando o código referente à quadratura gaussiana, discutida na Seção 2.1. Esta função tem como parâmetros de entrada a matriz de vértices *ps* e *n*, que determina a quantidade de nós em cada aresta da malha fina. Como saída, obtemos uma matriz *Q* contendo todas as posições dos vértices dos pontos de quadratura de $\bar{F}$.

```
function [Q]=quadratura(ps,n)

[m,~]=size(ps);
Q=zeros(n*(m-1),2);

switch n
  case 3
    for i=1:m-1
      for k=1:3
```

```
        a1=ps(i,1);
        b1=ps(i+1,1);
        xk=[-0.77499667;0;0.77499667];
        a2=ps(i,2);
        b2=ps(i+1,2);
        v1=((b1-a1)*xk(k)+a1+b1)/2;
        v2=((b2-a2)*xk(k)+a2+b2)/2;
        v=[v1,v2];
        Q(n*(i-1)+k,:)=v;
      end
    end

  case 2
    for i=1:m-1
      for k=1:2
        a1=ps(i,1);
        b1=ps(i+1,1);
        xk=[-0.57735027;0.57735027];
        a2=ps(i,2);
        b2=ps(i+1,2);
        v1=((b1-a1)*xk(k)+a1+b1)/2;
        v2=((b2-a2)*xk(k)+a2+b2)/2;
        v=[v1,v2];
        Q(n*(i-1)+k,:)=v;
      end
    end
end
```

Na etapa de fluxo queremos minimizar a distância com sinal entre as duas malhas. O sinal é determinado de acordo com o conceito de Win-

ding Number, discutido na Seção 2.2 e implementado como segue. O sinal é cálculado sobre todos os pontos de quadratura da malha fina, que foram determinados no código anterior, em relação a malha grosseira.

```
function [S]=sinal(Q,psq)

[m,~]=size(Q);
[g,~]=size(psq);
S=zeros(m,1);
for i=1:m
  angulos=zeros(m,1);
  p=Q(i,:);
  for k=2:g
    a=psq(k-1,:)-p;
    b=psq(k,:)-p;
    prod=a*b';
    M=[a' b'];
    deter=det(M);
    theta=atan2(deter,prod);
    angulos(k)=theta;
  end
  soma=sum(angulos);
  ang=soma/(2*pi);
  if ang>1-1e-5
    S(i)=-1;
  else
  S(i)=1;
  end
end
```

A distância, por sua vez é determinada pela conceito de projeção, discutido na Seção 2.3. A função dist_proj consiste em, dado um ponto de quadratura de $\bar{F}$, determinar o ponto de $\hat{G}$ mais próximo a ele, ou seja, sua projeção ortogonal sobre a malha. Isto é, projetando o ponto de quadratura sobre todas as arestas da malha grosseira. O ponto mais próximo é definido como sendo aquele que possui a menor distância. Esta função tem como parâmetros de saída uma matriz P contendo as posições dos vértices dos pontos de projeção e um vetor D contendo a distância do ponto de quadratura ao seu ponto de projeção.

```
function [D,P]=dist_proj(Q,psq)

[m,~]=size(Q);
[l,~]=size(psq);
P=zeros(m,2);
D=zeros(m,1);
for k=1:m
  x0=Q(k,:);
  Pontos=zeros(l-1,2);
  distancias=zeros(l-1,1);
  for j=2:l
    x1=psq(j-1,:);
    x2=psq(j,:);
    t0=((x2-x1)*(x0-x1)')/((x2-x1)*(x2-x1)');
    if t0<0 || t0>1
      d1=norm(x0-x1);
      d2=norm(x0-x2);
      Y=[x1;x2];
      F=[d1;d2];
      [dist,q]=min(F);
```

```
      ponto=Y(q,:);
    else
      ponto=x1+(x2-x1)*t0;
      dist=norm(x0-ponto);
    end
  Pontos(j-1,:)=ponto;
  distancias(j-1)=dist;
  end
  [menor_distancia,linha]=min(distancias);
  D(k)=menor_distancia;
  point=Pontos(linha,:);
  P(k,:)=point;
end
```

Vimos na seção 2.4 que o fluxo de $\bar{F}$ para o interior de $\hat{G}$ é dada tomando passo na direção oposta ao gradiente da função

$$\Phi(\bar{F}) = \int_{\bar{F}} s(\bar{p})d(\bar{p})dC$$

A função grad determina o gradiente de Φ, dado em (2.26). O gradiente é cálculado sobre todos os vértices de $\bar{F}$ e leva em consideração os pontos de quadratura, o sinal e distância, calculados anteriormente.

```
function [G]=grad(n,Q,ps,S,P,D)

[w,~]=size(Q);
[m,~]=size(ps);
G1=zeros(m-2,2);
G2=zeros(m-2,2);

switch n
```

```
case 3
  L=[0.8875;0.5;0.1125];
  A1=S(1)*L(1)*(P(1,:)-Q(1,:))/D(1)+S(2)*L(2)*...
  ...*(P(2,:)-Q(2,:))/D(2) + S(3)*L(3)*(P(3,:)-Q(3,:))/D(3);
  A2=S(w)*L(3)*(P(w,:)-Q(w,:))/D(w) +...
  ...+ S(w-1)*L(2)*(P(w-1,:)-Q(w-1,:))/D(w-1) +
  S(w-2)*L(1)*(P(w-2,:)-Q(w-2,:))/D(w-2);
  Gp=A1;
  Gpo=A2;
  G6=Gp+Gpo;

  for j=2:m-1
    A=[0,0;0,0];
    for k=1:n
      A1=(P(n*(j-1)+k,:)-Q(n*(j-1)+k,:))*(S(n*(j-1)+k)*...
      ...*L(k))/D(n*(j-1)+k);
      A(k,:)=A1;
    end
    G1(j-1,:)=sum(A);
  end

  for j=1:m-2
    B=[0,0;0,0];
    for k=1:n
      A2=S(n*(j-1)+k)*L(4-k)*(P(n*(j-1)+k,:)-...
      ...-Q(n*(j-1)+k,:))/D(n*(j-1)+k);
      B(k,:)=A2;
    end
    G2(j,:)=sum(B);
  end
```

```
    G=[G6;G1+G2];
  case 2
    L=[0.7887;0.2113];
    A1=S(1)*L(1)*(P(1,:)-Q(1,:))/D(1) +...
    ...+ S(2)*L(2)*(P(2,:)-Q(2,:))/D(2);
    A2=S(w)*L(2)*(P(w,:)-Q(w,:))/D(w) +...
     ...+S(w-1)*L(1)*(P(w-1,:)-Q(w-1,:))/D(w-1);
    Gp=A1;
    Gpo=A2;
    G6=Gp+Gpo;

    for j=2:m-1
      A=[0,0;0,0];
      for k=1:2
        A1=(P(n*(j-1)+k,:)-Q(n*(j-1)+k,:))*(S(n*(j-1)+k)...
        ...*L(k))/D(n*(j-1)+k);
        A(k,:)=A1;
      end
      G1(j-1,:)=sum(A);
    end

    for j=1:m-2
      B=[0,0;0,0];
      for k=1:2
        A2=S(n*(j-1)+k)*L(3-k)*(P(n*(j-1)+k,:)-...
        ...-Q(n*(j-1)+k,:))/D(n*(j-1)+k);
        B(k,:)=A2;
      end
      G2(j,:)=sum(B);
```

```
    end
    G=[G6;G1+G2];
end
y=G(1,:);
G=[G;y];
```

Agora que o gradiente de Φ foi determinado, o fluxo de $\bar{F}$ para o interior da malha grosseira é dado tomando passos de tamanho δ na direção oposta ao gradiente de Φ. Em nossos testes, adotamos $\delta = 10^{-3}$.
Em nosso trabalho usamos a biblioteca de processamento geométrico *gptoolbox*, disponibilizado por Alec Jacobson em
`https://github.com/alecjacobson/gptoolbox`.
Para determinar as malhas, usamos a função *get_pencil_curves*, que permite desenhar curvas. Para simplificar essas curvas em malhas poligonais, usamos a função *dpsimplify* que, dada uma certa tolerência, utiliza o algoritmo de simplificação recursiva de Douglas Peucker para reduzir o número de vértices de uma curva linear por partes. Mais informações sobre o algoritmo de Douglas Peucker pode ser encontrado em `http://en.wikipedia.org/wiki/Ramer-Douglas-Peucker_algorithm`.

A função *fluxo*, tem como parâmetros de entrada n, δ, $tol1$ e $tol2$, que representam o número de pontos de quadratura, o tamanho de passo, a tolerância de simplificação da malha fina e a tolerância de simplificação da malha grosseira, respectivamente. A função retorna como parâmetros de saída um número z, que indica quantos passos a malha fina deu na direção oposta ao gradiente até que estivesse totalmente no interior da malha grosseira, a matriz psq de vértices de $\hat{G}$, as z matrizes ps, em cada passo de tempo z e a matriz psf, com as posições de todos os vértices da malha fina após o fluxo. Nosso primeiro critério de parada consiste em verificar se todos os pontos de quadratura de $\bar{F}$ estão no interior de $\hat{G}$. Em caso afirmativo, verifica-se

as intersecções entre as duas malhas. Caso alguma aresta da malha fina intersecte alguma aresta da malha grosseira, o fluxo continua, até que sejam removidas todas as intersecções. O resultado final é a malha fina no fina totalmente interior a malha grosseira.

```
function [ps,psf,psq,z]=fluxo(n,delta,tol1,tol2)

[V,~,~]=get_pencil_curves(tol1);
[Vq,~,~]=get_pencil_curves(tol2);
ps(:,:,1)=dpsimplify(V,tol1);
psq=dpsimplify(Vq,tol2);

linha1=ps(1,:);
linha1q=psq(1,:);
ps=[ps;linha1];
psq=[psq;linha1q];
[Q]=quadratura(ps(:,:,1),n);
[S]=sinal(Q,psq);
[D,P]=dist_proj(Q,psq);
[G]=grad(n,Q,ps(:,:,1),S,P,D);
t=length(S);
b=-ones(t,1);
contador=0;

if S==b
r=1;
else
r=0;
end
```

```
ps(:,:,2)=ps(:,:,1)+delta*G;
z=2;

while r==0
  z=z+1;
  [Q]=quadratura(ps(:,:,z-1),n);
  [S]=sinal(Q,psq);
  [D,P]=dist_proj(Q,psq);
  [G]=grad(n,Q,ps(:,:,z-1),S,P,D);
  ps(:,:,z)=ps(:,:,z-1)+delta*G;
  psf=ps(:,:,z);
  contador=contador + 1;
  if S==b
    r=1;
  else
    r=0;
  end
end

psf=ps(:,:,z);
c=0;
[aval]=interseccao(psf,psq);

while aval==1
  z=z+1;
  [Q]=quadratura(ps(:,:,z-1),n);
  [S]=sinal(Q,psq);
  [D,P]=dist_proj(Q,psq);
  [G]=grad(n,Q,ps(:,:,z-1),S,P,D);
  ps(:,:,z)=ps(:,:,z-1)+delta*G;
```

```
  psf=ps(:,:,z);
  [aval]=interseccao(psf,psq);
  c=c+1;
end
```

A função *interseccao*, utiliza a função *distLinSeg*, disponível no link `https://www.mathworks.com/matlabcentral/fileexchange/49502-fast-shortest-distance-between-two-line-segments--in-n-dimensions-?focused=3860112&tab=function`, para verificar se as duas malhas se intersectam. Isto é feito testando todas as arestas de $\bar{F}$ com todas as arestas de $\hat{G}$

```
function [aval]=interseccao(psf,psq)
[l,~]=size(psf);
[m,~]=size(psq);
for i=1:m-1
  point1s=psq(i,:);
  point1e=psq(i+1,:);
  for j=1:l-1
    point2s=psf(j,:);
    point2e=psf(j+1,:);
    [dist,~] = distLinSeg(point1s,point1e,point2s,point2e);
    if dist<1e-12
      aval=1;
      return
    else
      aval=0;
    end
  end
```

```
end
```

Finalmente, a função *colisão* trata do processo de re-inflação discutido no capítulo 3. Iniciamos detectando todas as colisões do tipo ponto-aresta, conforme a seção 3.1. Para cada colisão detectada, definimos uma linha da matriz de restrições de desigualdade, conforme a subseção 3.2.1. Em seguida fazemos a detecção de todas as colisões do tipo aresta-ponto, conforme a subseção 3.2.3 e para cada colisão detectada, definimos uma nova linha da matriz de restrições.
A matriz de restrição de igualdade é definida conforme 3.2.1. Definidas as matrizes de restrições, usamos *fmincon* para determinar o vetor X de posições dos vértices das duas malhas. Atulizamos as matrizes *ps* e *psq* e repetimos o processo até que sejam dados z passos de re-inflação.

```
function [pteste,psq]=colisao(ps,psq,z)
tic
[l,~]=size(ps);
[m,~]=size(psq);
pteste=ps(:,:,z);
for k=1:z-1
  B=[];
  c=[];
  for i=1:l
    A=zeros(1,2*(l+m)+2);
    pteste=ps(:,:,z);
    V1=ps(:,:,z-1)-ps(:,:,z);
    v=V1(i,:)';
    p0=pteste(i,:)';
    tt=zeros(m,1);
    for j=1:m-1
```

```
      a=psq(j,:)';
      b=psq(j+1,:)';
      s=v(1)*(b(2)-a(2))+v(2)*(-b(1)+a(1));
      t=(a(1)-p0(1))*(b(2)-a(2))+(a(2)-p0(2))*(-b(1)+a(1));
      t=t/s;
      if t>1e-8 && t<1
        p=p0+t*v;
        a1=p-a;
        a2=b-a;
        u=a1(1)/a2(1);
        if u<=1&& u>1e-8
          tt(j)=t;
          aresta=psq(j+1,:)'-psq(j,:)';
          N=[aresta(2) -aresta(1)];
          A(j)=-N(1)*(1-u);
          A(j+1)=-(u)*N(1);
          A(j+m)=-N(2)*(1-u);
          A(j+1+m)=-(u)*N(2);
          A(2*m+i)=N(1);
          A(2*m+i+l)=N(2);
          c=[c;0];
          B=[B;A];
        end
      else
        tt(j)=1050;
      end
    end
  end

  for i=1:m
```

```
V=ps(:,:,z-1)-ps(:,:,z);
p=psq(i,:)';
tt=zeros(m,1);
for j=1:l-1
  va=V(j,:)';
  vb=V(j+1,:)';
  a0=pteste(j,:)';
  b0=pteste(j+1,:)';
  AA = va(2)*vb(1) - va(1)*vb(2);
  BB=(-b0(2)*va(1) + p(2)*va(1) + b0(1)*va(2) - p(1)*va(2) +...
  ...+ a0(2)*vb(1) - p(2)*vb(1) - a0(1)*vb(2) + p(1)*vb(2));
  CC = (a0(2)*b0(1) - a0(1)*b0(2) - a0(2)*p(1) + b0(2)*p(1) +..
  ...+ a0(1)*p(2) - b0(1)*p(2));
  delta=BB^2-4*(AA*CC);
  if (delta>=0)
    t1 = (-BB - sqrt(delta))/(2*AA);
    t2 = (-BB + sqrt(delta))/(2*AA);
    if ((t1>0 && t1<1) && ~(t2>0 && t2<1))
      t = t1;
    elseif (~(t1>0 && t1<1) && (t2>0 && t2<1))
      t = t2;
    else
      t=min(t1,t2);
    end
  else
    t = -1000;
  end

  if t>1e-8 && t<1
    a=a0+t*va;
```

```
        b=b0+t*vb;
        a1=p-a;
        a2=b-a;
        u=a1(1)/a2(1);
        if u<=1&& u>1e-8
          A=zeros(1,2*(l+m)+2);
          a=a0+1*va;
          b=b0+1*vb;
          are=b-a;
          N=[are(2) -are(1)];
          A(i)=-N(1);
          A(i+m)=-N(2);
          A(2*m+j)=N(1)*u;
          A(2*m+j+1)=N(1)*(1-u);
          A(2*m+l+j)=N(2)*u;
          A(2*m+l+j+1)=N(2)*(1-u);
          c=[c;0];
          B=[B;A];
        end
      else
        tt(j)=1050;
      end
    end
  end

  l3=zeros(1,2*(l+m));
  l3(1,m)=-1;
  l3(1,1)=1;
  l4=zeros(1,2*(l+m));
  l4(1,2*m)=-1;
```

```
  l4(1,m+1)=1;
  pl=[0;0];
  linha=zeros(2*l+2,1);
  beq=ps(:,:,z-1);
  beq=beq(:);
  beq=[beq;pl];
  Aeq=[zeros(2*l,2*m) eye(2*l,2*l)];
  Aeq=[Aeq;l3;l4];
  Aeq=[Aeq linha linha];
  fun = @(X)(norm(X-[psq(:,1);psq(:,2);beq])^2);
  options = optimoptions('fmincon','Display','iter','Algorithm',...
  ...,'interior-point');
  X = fmincon(fun,[psq(:,1);psq(:,2);beq],B,c,..
  ..,Aeq,beq,[],[],[],options);
  c1=X(1:m);
  c2=X(m+1:2*m);
  d1=X(2*m+1:2*m+l);
  d2=X(2*m+l+1:2*(m+l));
  pteste=[d1 d2];
  psq=[c1 c2];
  z=z-1;
end
```

Conclusão e Trabalhos Futuros

Apresentamos neste trabalho um método para geração de malhas poligonais exteriores em duas dimensões. No Capítulo 2, detalhamos todos os conceitos necessários para realizar o fluxo de uma malha fina para o interior de uma malha grosseira. No Capítulo 3 descrevemos a re-inflação da malha fina de volta para sua posição inicial empurrando a malha grosseira. Para resolver o problema de otimização restrito usamos o solver *fmincon*. Por fim, no Capítulo 4, apresentamos códigos implementados, bem como alguns resultados numéricos.

Em trabalhos futuros, pretendemos testar outras funções objetivo no problema de minimização restrita, bem como testar outros métodos para resolver o problema de otimização, como por exemplo, região de confiança e restrições ativas.

Os códigos implementados estão disponíveis em
`http://mtm.ufsc.br/~leo/dissertacao_aline.zip`.

Referências Bibliográficas

[1] Sacht, L. K. and Vouga E. and Jacobson A. *Nested Cages*. Siggraph Asia 2015.

[2] Lang, S., *Complex analysis*. 4. ed. New York: Springer, c1999. xiv, 485 p. ISBN 0387985921.

[3] Conway, J. B. *Functions of one complex variable*. New York: Springer, c1973. xi, 313p. (Graduate texts in mathematics; 11).)) ISBN 0387900624 : 0387900616 : (broch.).

[4] Burden, R. L. *Numerical Analysis*. Boston: PWS, 1981. 598p.

[5] Nocedal, J. and Wright, S. J., *Numerical Optimization*. 2nd ed. New York: Springer, c2006. xxii, 664 p. ISBN 9780387303031

[6] Izmailov, A. and Solodov, M., *Otimização, volume 2: métodos computacionais*. Rio de Janeiro: Instituto de Matemática Pura e Aplicada, c2007. 448p. ISBN 8524402685 (broch.)

[7] Franco, N. M. B., *Cálculo numérico* Universidade de São Paulo. Instituto de Ciências Matemáticas e de Computação.

[8] Provot, X., *Colisions and self-collision handling in model dedicated to design garments* Institut National de Recherche en Informatique et Automatique (INRIA) B.P. 105, 78153 Le Chesnay Cedex, France.

[9] Baraff, D. and Witkin, A. *Large Steps in Cloth Simulation*. Computer Graphics Proceedings, Annual Conference Series. Siggraph, Orlando 1998.

[10] Otaduy, M. and Tanstorf, R. and Steinemann, D. and Gross, Markus, *Implicit Contact Handling for Deformable Objects* Eurographics, 2009.

Volume 28, Number 2 .

[11] Jacobson, A. and Kavan, L. and Sorkine-Hornung, O. *Robust Inside-Outside Segmentation using Generalized Winding Numbers* ACM Transactions on Graphics (TOG) - SIGGRAPH 2013 Conference Proceedings. New York, 2013.

[12] DeRose, T. and Meyer, M. *Harmonic Coordinates* Pixar Technical Memo 06,02. Pixar Animation Studios.

[13] Harmon, D. and Vouga, E. and Smith, B. and Tamstorf, R. and Grinspun E., *Asynchronous Contact Mechanics* SIGGRAPH '09 (ACM Transactions on Graphics). New York, 2009. ISBN: 978-1-60558-726-4.

[14] Sander, P. V. and Gu, X. and Gortler, S. and Hoppe, H. and Snyder, J. *Silhouette Clipping* ACM SIGGRAPH 2000 Proceedings, 327-334.

[15] Xian. C. and Lin, H. and Gao, S. *Automatic cage generation by improved OBBs for mesh deformation* Vis Comput (2012) 28:21–33. DOI 10.1007/s00371-011-0595-6.

[16] Ainsley, S., Vouga, E., Grinspun, E., Tamstorf, R., *Speculative Parallel Asynchronous Contact Mechanics* SIGGRAPH ASIA (ACM Transactions on Graphics) 2012. issn: 0730-0301.

[17] Jacobson, A. *gptoolbox* https://github.com/alecjacobson/gptoolbox