Pontifícia Universidade Católica
do Rio de Janeiro

Jéferson Rômulo Pereira Coêlho

Uma Metodologia baseada em Otimização Quadrática para Geração de Malhas Geomecânicas de Reservatórios

Tese de Doutorado

Tese apresentada ao Programa de Pós–graduação em Informática do Departamento de Informática da PUC–Rio como requisito parcial para obtenção do grau de Doutor em Informática

Orientador: Prof. Marcelo Gattass

Rio de Janeiro
Abril de 2018

Pontifícia Universidade Católica
do Rio de Janeiro

Jéferson Rômulo Pereira Coêlho

Uma Metodologia baseada em Otimização Quadrática para Geração de Malhas Geomecânicas de Reservatórios

Tese apresentada ao Programa de Pós–graduação em Informática do Departamento de Informática do Centro Técnico Científico da PUC–Rio como requisito parcial para obtenção do grau de Doutor em Informática. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Marcelo Gattass
Orientador
Departamento de Informática — PUC–Rio

Prof. Anselmo Antunes Montenegro
Departamento de Informática — UFF

Prof. Ivan Fabio Mota de Menezes
Departamento de Engenharia Mecânica — PUC–Rio

Prof. Paulo Cezar Pinto Carvalho
FGV

Prof. Waldemar Celes Filho
Departamento de Informática — PUC–Rio

Prof. Márcio da Silveira Carvalho
Coordenador Setorial do Centro Técnico Científico — PUC–Rio

Rio de Janeiro, 16 de Abril de 2018

Jéferson Rômulo Pereira Coêlho

Possui graduação em Ciência da Computação pela Universidade Federal de Viçosa em 2011 e mestrado em Informática, com ênfase em computação gráfica, pela Puc-Rio em 2013.

Ficha Catalográfica

Coêlho, Jéferson Rômulo Pereira

Uma Metodologia baseada em Otimização Quadrática para Geração de Malhas Geomecânicas de Reservatórios / Jéferson Rômulo Pereira Coêlho; orientador: Prof. Marcelo Gattass. — Rio de Janeiro : PUC–Rio, Departamento de Informática, 2018.

v., 121 f: il. ; 29,7 cm

1. Tese (doutorado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática.

Inclui referências bibliográficas.

1. Informática – Tese. 2. Modelagem Geométrica; Malhas Geomecânicas de Reservatórios; Otimização Geométrica; Otimização Quadrática; Otimização Convexa.. I. Gattass, Marcelo. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

À Eva Pereira dos Santos

Agradecimentos

Existe uma quantidade imensa que pessoas que contribuíram de alguma forma para essa caminhada e que infelizmente não poderei citá-las nominalmente pela limitação de espaço.

Gostaria de começar agradecendo a CNPq e ao Instituto Tecgraf/PUC-Rio por financiarem parcialmente esta pesquisa.

A todos os meus amigos, de todos os lugares que passei, que entenderam a minha ausência durante esse longo período que foi a pós graduação.

A todos os familiares que deixei de visitar inúmeras vezes por falta de tempo, mas que mesmo assim não perderam o carinho por mim.

À família Martins pelo apoio e carinho durante o ensino médio. Em especial à Maria Aparecida Sampaio Bastos e Jarisch de Almeida Bastos que contribuíram de maneira determinante neste importante passo do processo que foi a minha formação acadêmica.

A todos os alunos que tive e me ajudaram a ser um educador melhor, em especial Caroline Redlich, Fernando da Senhora, Carolina Esporcatte, Tito Bruni e Giovana Mathias.

Aos meus orientadores da graduação, Valérias Mattos, Catarina Mendes de Jesus e Marcus Vinícius Alvim de Andrade pelos valorosos conhecimentos e conselhos que me foram passados e que foram fundamentais para o meu desenvolvimento no mestrado e doutorado.

Aos amigos que fiz no Instituto Tecgraf/PUC-Rio e na PUC-Rio que suportaram, e as vezes até se divertiram (ou não), com as inúmeras discussões sobre problemas acadêmicos e teorias para soluções dos mesmos. Em especial Rafael Martinelli, Eder Perez e Suellen Caduda.

Um agradecimento especial a Roberto Quispe, que contribuiu diretamente com esta tese através de valorosas discussões sobre as aplicações das malhas em simulações geomecânicas de reservatórios e com o apoio na escrita da Seção 4.1 contida nesta tese.

Ao meu orientador, Marcelo Gattass, por todos os momentos de valorosas discussões. Por abrir espaço para ideias diferentes, mesmo que muitas vezes contrárias às suas. Por todos os conselhos e paciência. O aprendizado que levarei transcende a esfera acadêmica.

E por final, um agradecimento mais que especial para os meus pais Romilda Pereira dos Santos e Hélio Gilson Coêlho, por toda luta e sofrimento que passaram para que eu pudesse ter condições de estudar.

Resumo

Coêlho, Jéferson Rômulo Pereira; Gattass, Marcelo. **Uma Metodologia baseada em Otimização Quadrática para Geração de Malhas Geomecânicas de Reservatórios**. Rio de Janeiro, 2018. 121p. Tese de Doutorado — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

A geração de malhas geomecânicas de reservatórios ainda é uma tarefa tediosa que consome muito tempo. Para acelerar este processo, soluções que reconstroem analiticamente a geometria do reservatório têm sido propostas, mas essas soluções não são as mais adequadas para modelagem de objetos naturais. Este trabalho propõe uma modelagem discreta para a geometria do reservatório, onde os vértices da malha são posicionados por meio da solução de um problema de otimização quadrático e convexo. O problema de otimização é modelado de forma a garantir que as malhas geomecânicas de saída sejam suaves e que ao mesmo tempo respeitem as restrições do reservatório e dos horizontes presentes. Além disso, a metodologia proposta permite uma implementação eficiente, paralelizável e de baixo consumo de memória. Casos de teste com milhões de variáveis são apresentados para validar essa abordagem. Finalmente, a metodologia proposta neste trabalho para malhas de geomecânica pode ser naturalmente estendida para a modelagem estrutural de sub-superfícies na interpretação sísmica e de restauração geológica.

Palavras–chave

Modelagem Geométrica; Malhas Geomecânicas de Reservatórios; Otimização Geométrica; Otimização Quadrática; Otimização Convexa.

Abstract

Coêlho, Jéferson Rômulo Pereira; Gattass, Marcelo (Advisor). **A quadratic optimization approach for the reservoir Geomechanical mesh generation**. Rio de Janeiro, 2018. 121p. Tese de Doutorado — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Geomechanical mesh generation of complex reservoirs remains a tedious task prone to errors. Recently proposed solutions based on analytical reconstruction of the sub-surfaces are not capable to represent all the geometric details of natural objects. This work proposes a discrete model where the mesh vertices are positioned based on a convex quadratic optimization process. The optimization problem seeks to guarantee smooth meshes that conform with prescribed constraints. The resulting mesh therefore respects, as far as possible, the finite volume mesh of the reservoir pay zone and the existing horizons. Finally, the proposed methodology for Geomechanical meshes can be easily extend to model sub-surfaces present in the structural interpretation and geological restauration.

Keywords

Geometric Modeling; Reservoir Geomechanical Mesh; Geometric Optimization; Quadratic Optimization; Convex Optimization.

Sumário

Lista de figuras

Lista de tabelas

Lista de Algoritmos

1
Introdução

Soluções para um grande número de problemas enfrentados pela indústria petrolífera durante a exploração, avaliação e produção requerem integração de conhecimento e fluxos de trabalho de uma variedade de disciplinas, incluindo geologia, engenharia de petróleo, geomecânica, física de rochas, petrofísica e geofísica. No futuro, os projetos de armazenamento de CO_2 em cavernas submarinas também exigirão uma abordagem integrada. Em 2008 o consórcio multidisciplinar IPEGG (*Integrated Petroleum Engineering, Geomechanics and Geophysics*) foi criado com o principal objetivo de desenvolver e aplicar simulação acoplada de fluidos e geomecânica e integrar-se à modelagem sísmica para ajudar a prever o comportamento do reservatório (Angus *et al.*, 2011).

A etapa preliminar na geração de um modelo geomecânico é a geração da malha de elementos finitos com base no modelo geológico do reservatório. A malha deve ser expandida em torno do reservatório e deve ser conforme, ou seja, não devem haver *gaps* e cada nó deve ser conectado com os nós vizinhos. De acordo com (Angus *et al.*, 2011) falhas e outras descontinuidades nas subsuperfícies apresentam o maior desafio para geração da malha. Na tentativa de resolver este problema, soluções baseadas em reconstruções analíticas tem sido propostas. No entanto, estas abordagens não são as mais adequadas para lidar com objetos naturais, como aquelas encontrados em biologia e geologia. Ferramentas como NURBS (Piegl e Tiller, 1997), que foram desenvolvidas para superfícies de automóveis, são muito suaves para representar fronteiras geológicas. Decomposições espaciais, como são as malhas, são mais adequadas.

Um dos primeiros métodos de geração de malhas são os mapeamentos transfinitos (Gordon e Hall, 1993). Esta classe de métodos estabelece sistemas de coordenadas curvilíneas definidos pelo contorno de domínios arbitrários. Mallet (Mallet, 1989) foi um dos precursores da modelagem matemática de objetos naturais. Nos seus trabalhos, Mallet desenvolveu uma teoria que embasou o Consórcio GOCAD (Mallet, 1992) que, por sua vez, deu origem ao atual software de modelagem de reservatórios da Paradigm[1]. Uma análise mais aprofundada dos trabalhos do Mallet (Mallet, 1989, 2002) revela que ele revisitou as áreas

[1] http://www.pdgm.com/

de geometria computacional e otimização quadrática clássicas com uma notação própria que por um tempo manteve a comunidade de geometria computacional alheia a eles. A interação deste grupo de pesquisa com a comunidade de geometria computacional internacional se iniciou a partir de um artigo no SIGGRAPH de 1998 (Levy e Mallet, 1998) do Mallet com um dos seus alunos, Bruno Levy. Esse assunto de modelagem de superfícies naturais foi grandemente impulsionado pelo escaneamento de esculturas importantes, como o Davi de Michelangelo, que produziu uma quantidade enorme de nuvens de pontos em 3D. Os cursos nas conferências do Eurographics (Botsch *et al.*, 2006b) e do SIGGRAPH (Botsch *et al.*, 2006) confirmam esse foco. A ênfase, entretanto, foi na modelagem a partir de nuvens de pontos 3D escaneados de objetos reais. A relação destas modelagens geométricas com a geração de malhas não é tão direta. As malhas buscam representar o domínio de EDP (equação diferencial parcial) no objeto todo e não só na sua superfície.

Na PUC-Rio, a modelagem de objetos naturais teve sua origem nos trabalhos de geração de malhas feitos na Universidade de Cornell (Martha, 1989) que também tinham como base modelagens contínuas das superfícies. A ideia básica dessa estratégia é particionar o objeto complexo em regiões homeomorfas a cubos e tetraedros onde um mapeamento de geração de malhas é mais simples. As primeiras evoluções desta linha de pesquisa no Tecgraf foram os sistemas MG (Coelho, 1998), MTool (Celes, 1995) e Sigma3D[2]. Este trabalho revisita esta linha de pesquisa em busca de uma visão integrada dos modelos Geofísicos, Geológicos, de Reservatório e Geomecânico, todos modelados por decomposição espacial: voxel, volumes finitos, elementos finitos e outras representações equivalentes. O objetivo difuso desta tese é estabelecer as bases para o desenvolvimento desta nova ênfase da Linha de Pesquisa.

Este trabalho tem como objetivo específico gerar uma malha geomecânica de elementos finitos a partir de uma malha de fluxo de volumes finitos com descontinuidades geométricas. Ao eliminar as descontinuidades, a malha é suavizada, para garantir uma melhor transição entre os elementos, gerando assim uma malha válida e de qualidade.

A principal contribuição deste trabalho é uma técnica de reconstrução da malha geomecânica de reservatórios por meio de uma abordagem formal baseada em otimização quadrática convexa modelada diretamente nos domínios discretos das malhas. Esta abordagem suaviza a malha de fluxo de volumes finitos eliminando as descontinuidades e a estende para regiões onde as condições de contorno geomecânicas podem ser estimadas. Este trabalho propõe também uma

[2] `https://www.tecgraf.puc-rio.br/publicproductdetail/iaJckCFhrM7owypcx`

solução eficiente, paralelizável e com baixo custo de memória para o problema de otimização quadrático proposto. Além disso, ao contrário das abordagem de reconstrução analíticas, onde uma superfície contínua é construída com base nas informações de entrada, a abordagem proposta é modelada sobre uma discretização da malha do reservatório. Outra contribuição importante é uma infraestrutura criada para modelagem geométrica através de uma aplicação de otimização quadrática.

Esta tese está organizada da seguinte forma: o Capítulo 2 apresenta o conceitos básicos de otimização convexa. Nesse capítulo, são apresentadas propriedades importantes que serão usadas posteriormente para demonstrar a convexidade do problema de otimização proposto. No Capítulo 3, é feita uma revisão das propriedades e das diversas discretizações do operador de suavização de Laplace-Beltrami, que será usado como operador de suavização da malha resultante. No capítulo 4 os trabalhos mais recentes na geração de malhas geomecânicas de reservatórios são apresentados. A modelagem do problema de otimização proposto é apresentado no Capítulo 5. Nesse mesmo capítulo são demonstradas propriedades importantes, como a convexidade estrita do problema de otimização proposto, o que é uma propriedade necessária para que uma solução eficiente possa ser projetada. Um método eficiente para a solução do problema de otimização proposto no Capítulo 5 é apresentado no Capítulo 6. Este método resolve um sistema de equações lineares sem a necessidade de alocar a matriz em memória. No Capítulo 7, é feita uma análise dos resultados obtidos e, no Capítulo 8, são apresentadas as conclusões e são propostos alguns trabalhos futuros.

Para uma leitura mais linear do texto, dois assuntos estão expostos em apêndices. O Apêndice A apresenta alguns conceitos básicos de geometria diferencial que são utilizados no Capítulo 3. O Apêndice B apresenta a aplicação para modelagem geométrica baseada em otimização quadrática desenvolvida durante a tese. Esse apêndice traz alguns exemplos de aplicações com os respectivos códigos escritos na aplicação.

2
Otimização Convexa

Os problemas abordados nesta tese são modelados como problemas de otimização quadrática convexa. Portanto, neste capítulo apresenta-se uma introdução de otimização matemática, com foco nos casos especiais de otimização convexa e em otimização quadrática convexa. Convexidade tem um importante papel em otimização não linear. Uma razão para isso é que quando uma função f é convexa, não existe distinção entre mínimo local e mínimo global.

Apesar de otimização convexa e seus casos especiais, como otimização quadrática convexa, serem um assunto bastante extenso, a descrição feita aqui tem um escopo limitado ao conteúdo e propriedades necessárias para o desenvolvimento deste trabalho. A maioria dos resultados é dada sem prova, no entanto, uma descrição mais aprofundada, formal e geral dos assuntos aqui expostos pode ser encontrada nas referências (Boyd e Vandenberghe, 2004; Bertsekas, 2015, 2016; Luenberger e Ye, 2016), nas quais este capítulo é baseado.

2.1
Conjuntos Afins e Convexos

Tome $\mathbf{x}_1 \neq \mathbf{x}_2$ dois pontos em $\mathbb{R}^n$. Os pontos da forma

$$\mathbf{y} = \theta\mathbf{x}_1 + (1-\theta)\mathbf{x}_2, \tag{2-1}$$

onde $\theta \in \mathbb{R}$, formam uma *reta* passando pelos pontos $\mathbf{x}_1$ e $\mathbf{x}_2$. Os valores de $\theta = 0$ e $\theta = 1$ correspondem, respectivamente, aos pontos $\mathbf{x}_2$ e $\mathbf{x}_1$. Além disso, valores do parâmetro θ entre 0 e 1 correspondem ao *segmento de reta* entre $\mathbf{x}_1$ e $\mathbf{x}_2$.

No caso em que $\mathbf{y}$ é expressado da forma

$$\mathbf{y} = \mathbf{x}_2 + \theta(\mathbf{x}_1 - \mathbf{x}_2) \tag{2-2}$$

existe uma interpretação alternativa. Neste caso, $\mathbf{y}$ é a soma do ponto base $\mathbf{x}_2$ e a direção $\mathbf{x}_1 - \mathbf{x}_2$ escalado pelo parâmetro θ. Desta forma, θ representa a fração do caminho de $\mathbf{x}_2$ para $\mathbf{x}_1$ em que $\mathbf{y}$ está. À medida que θ aumenta de 0 para 1, o ponto $\mathbf{y}$ move de $\mathbf{x}_2$ para $\mathbf{x}_1$. Para $\theta > 1$, o ponto $\mathbf{y}$ está além do ponto $\mathbf{x}_1$, como ilustrado na Figura 2.1.

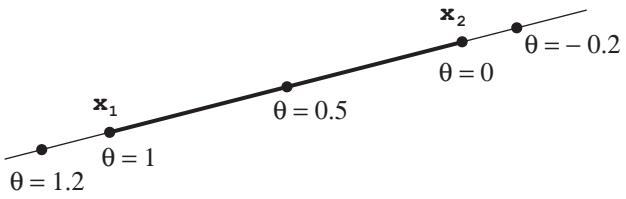

Figura 2.1: Reta passando através dos pontos $\mathbf{x}_1$ e $\mathbf{x}_2$ é descrita de forma paramétrica por $\theta\mathbf{x}_1 + (1-\theta)\mathbf{x}_2$. O segmento de reta relativo a $0 \leq \theta \leq 1$ é mostrado em negrito.

2.1.1 Conjunto Afim

Um conjunto $C \subseteq \mathbb{R}^n$ é *afim* se a reta que passa através de dois pontos distintos quaisquer em C está em C, ou seja, se para quaisquer $\mathbf{x}_1, \mathbf{x}_2 \in C$ and $\theta \in \mathbb{R}$, tem-se que

$$\theta\mathbf{x}_1 + (1-\theta)\mathbf{x}_2 \in C \quad \theta \in \mathbb{R} \tag{2-3}$$

Isto significa que as combinações lineares de quaisquer dois pontos em C, quando os coeficientes da combinação linear somam um, também estão em C.

Esta definição pode ser generalizada para mais de dois pontos, de forma que os pontos da forma $\theta_1\mathbf{x}_1 + \ldots + \theta_k\mathbf{x}_k$, onde $\theta_1 + \ldots + \theta_k = 1$, são uma *combinação afim* dos pontos $\mathbf{x}_1, \ldots, \mathbf{x}_k$. Por indução matemática na definição de conjunto afim, pode ser demonstrado que um conjunto afim contém todas as combinações afins de seus pontos, ou seja, se C é um conjunto afim, $\mathbf{x}_1, \ldots, \mathbf{x}_k \in C$ e os coeficientes $\theta_1 + \ldots + \theta_k = 1$, então o ponto $\theta_1\mathbf{x}_1 + \ldots + \theta_k\mathbf{x}_k$, também pertence a C (Boyd e Vandenberghe, 2004).

2.1.2 Conjunto Convexo

Um conjunto C é *convexo* se o segmento de reta entre quaisquer dois pontos em C está em C, ou seja, se para quaisquer $\mathbf{x}_1, \mathbf{x}_2 \in C$ e $0 \leq \theta \leq 1$, tem-se que

$$\theta\mathbf{x}_1 + (1-\theta)\mathbf{x}_2 \in C \quad 0 \leq \theta \leq 1 \tag{2-4}$$

Todo conjunto afim é também convexo. Uma vez que ele contém a reta inteira entre quaisquer par de pontos, ele contém também os segmentos de reta entre estes pontos.

O ponto da forma $\theta_1\mathbf{x}_1 + \ldots + \theta_k\mathbf{x}_k$, com $\theta_1 + \ldots + \theta_k = 1$ e $\theta_i \geq 0$, $i = 0, \ldots, k$ é uma *combinação convexa* dos pontos $\mathbf{x}_1, \ldots, \mathbf{x}_k$. Assim como no caso dos conjuntos afins, pode ser demonstrado que um conjunto é convexo se, e somente se, ele contém toda as combinações convexas dos seus pontos. A Figura 2.2 ilustra dois conjuntos, um convexo e outro não convexo.

Algumas operações sobre conjuntos preservam a convexidade, abaixo são listadas três delas (Bertsekas, 2015):

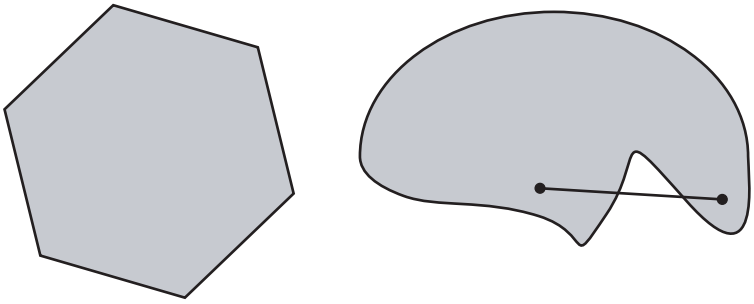

Figura 2.2: A esquerda um hexágono convexo, à direita um conjunto não convexo, uma vez que a reta que conecta dois pontos no conjunto não está contida no conjunto.

a) A interseção $\bigcap_{i \in I} C_i$ de qualquer coleção $\{C_i | i \in I\}$ de conjuntos convexos, é um conjunto convexo.

b) A soma vetorial $C_1 + C_2$ de dois conjuntos convexos C_1 e C_2 é um conjunto convexo.

c) O conjunto λC é convexo para qualquer conjunto convexo C e escalar λ. Além disso, se C é um conjunto convexo e λ_1, λ_2 são escalares positivos,

$$(\lambda_1 + \lambda_2)C = \lambda_1 C + \lambda_2 C \tag{2-5}$$

2.1.3 Cones

Um conjunto C é chamado de *cone* se para todo $\mathbf{x} \in C$ e $\theta \geq 0$, tem-se que $\theta\mathbf{x} \in C$. Um conjunto C é um *cone convexo* se ele é convexo e um cone ao mesmo tempo, ou seja, para quaisquer $\mathbf{x}_1, \mathbf{x}_2 \in C$ e $\theta_1, \theta_2 \geq 0$, tem-se que

$$\theta_1\mathbf{x}_1 + \theta_2\mathbf{x}_2 \in C \tag{2-6}$$

Pontos da forma de 2-6 podem ser descritos geometricamente como o conjunto de pontos entre as duas semi-retas que saem da origem e passam pelos pontos $\mathbf{x}_1$ e $\mathbf{x}_2$, como apresentado na Figura 2.3.

Um ponto da forma $\theta_1\mathbf{x}_1 + \ldots + \theta_k\mathbf{x}_k$, com $\theta_1 + \ldots + \theta_k \geq 0$ é chamado de *combinação cônica* de $\mathbf{x}_1, \ldots, \mathbf{x}_k$. Se $\mathbf{x}_i$ está em um cone convexo C, então toda combinação cônica de $\mathbf{x}_i$ está em C. Por outro lado, um conjunto C é um cone convexo se, e somente se, ele possui todas as combinações cônicas de seus elementos.

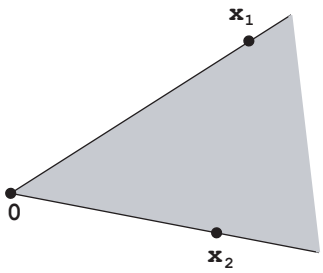

Figura 2.3: Cone formado pelos pontos bidimensionais $\mathbf{x}_1$ e $\mathbf{x}_2$ Para $\theta_1 = \theta_2 = 0$ tem-se a origem e as arestas correspondem a $\theta_1 = 0$ ou $\theta_2 = 0$.

2.1.4 Função Afim

Uma função $f : \mathbb{R}^n \to \mathbb{R}$ é dita *afim* se ela é da forma $f(\mathbf{x}) = \mathbf{a}^T\mathbf{x} + b$ para algum $\mathbf{a} \in \mathbb{R}^n$ e $b \in \mathbb{R}$. Da mesma forma $f : \mathbb{R}^n \to \mathbb{R}^m$ é dita *afim* se ela tem a forma $f(\mathbf{x}) = \mathbf{A}\mathbf{x}+\mathbf{b}$, onde $\mathbf{A}$ é uma matriz de dimensão $m \times n$ e $b \in \mathbb{R}^m$ (Bertsekas, 2015). Se $b = 0$, f é uma *função linear* ou *transformação linear*. Neste texto, com um pequeno abuso de terminologia, equações ou inequações como $\mathbf{a}^T\mathbf{x} = b$ ou $\mathbf{a}^T\mathbf{x} \leq b$ serão referidas como equações ou inequações lineares.

2.2 Funções Convexas

Uma função $f : \mathbb{R}^n \to \mathbb{R}$ é *convexa* se o **dom** f (domínio de f) é um conjunto convexo e se para todo $\mathbf{x}$, $\mathbf{y} \in$ **dom** f e $0 \leq \theta \leq 1$, tem-se

$$f(\theta\mathbf{x} + (1-\theta)\mathbf{y}) \leq \theta f(\mathbf{x}) + (1-\theta)f(\mathbf{y}) \tag{2-7}$$

Geometricamente, esta desigualdade significa que o segmento de reta entre $(\mathbf{x}, f(\mathbf{x}))$ e $(\mathbf{y}, f(\mathbf{y}))$, o qual é uma corda conectando $f(\mathbf{x})$ a $f(\mathbf{y})$, permanece acima do gráfico de f, como ilustrado na Figura 2.4.

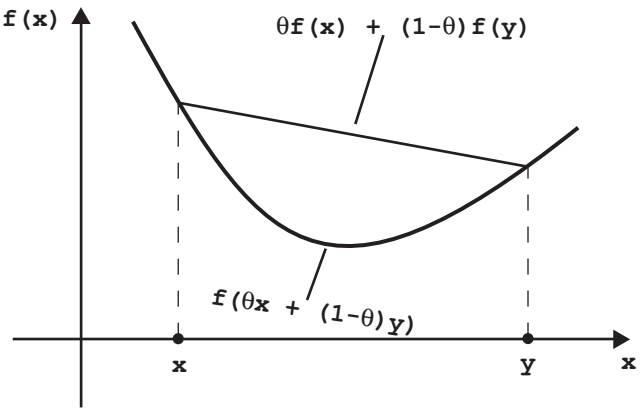

Figura 2.4: Gráfico de uma função convexa. A corda conectando os dois pontos na função está sempre acima da função.

Uma função f é *estritamente convexa* se a desigualdade 2-7 é estritamente satisfeita sempre que $\mathbf{x} \neq \mathbf{y}$ e $0 < \theta < 1$. A função f é dita *côncava* se $-f$ é convexa e *estritamente côncava* se $-f$ é estritamente convexa.

Para toda função afim, a desigualdade 2-7 é satisfeita na igualdade, logo todas as funções afins são convexas e côncavas ao mesmo tempo. Por outro lado, toda função que é convexa e côncava é afim (Boyd e Vandenberghe, 2004).

2.2.1 Soma ponderada por pesos não negativos

Se uma função f é convexa e $\alpha \geq 0$, então a função αf também é convexa. Se f_1 e f_2 são funções convexas, então a soma $f_1 + f_2$ é uma função convexa (Boyd e Vandenberghe, 2004). Combinando a escala por um peso não negativo com a soma de funções convexas, nota-se que o conjunto de funções convexas é por si só um cone convexo, ou seja, uma soma ponderada por pesos não negativos de funções convexas

$$f = w_1 f_1 + ... + w_n f_n \tag{2-8}$$

é convexa com domínio dado por

$$\mathbf{dom}\ f = \bigcap_{i=1}^{n} \mathbf{dom}\ g_i \tag{2-9}$$

Da mesma forma, a soma ponderada de funções côncavas por pesos não negativos é côncava. A soma ponderada por pesos estritamente positivos de funções estritamente convexas (côncavas) é estritamente convexa (côncava).

2.2.2 Composição com funções afins

Sejam $f : \mathbb{R}^n \to \mathbb{R}$, $h : \mathbb{R}^m \to \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{n \times m}$, e $b \in \mathbb{R}^n$. Tome $g : \mathbb{R}^m \to \mathbb{R}$ dada por

$$g(\mathbf{x}) = f(h(\mathbf{x})) = f(\mathbf{A}\mathbf{x} + b) \tag{2-10}$$

com $\mathbf{dom}\ g = \{\mathbf{x} | h(\mathbf{x}) \in \mathbf{dom}\ f\}$. Se f é convexa, então g também é convexa. Da mesma forma, se f é côncava, então g também é côncava (Bertsekas, 2015).

Prova: Note que h é uma função afim, logo para $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$ é válido que

$$h(\theta\mathbf{x} + (1-\theta)\mathbf{y}) = \theta h(\mathbf{x}) + (1-\theta)h(\mathbf{y}) \tag{2-11}$$

Como f é uma função convexa, tem-se,

$$\begin{aligned} g(\theta\mathbf{x} + (1-\theta)\mathbf{y}) &= f(h(\theta\mathbf{x} + (1-\theta)\mathbf{y})) \\ &= f(\theta h(\mathbf{x}) + (1-\theta)h(\mathbf{y})) \\ &\leq \theta f(h(\mathbf{x})) + (1-\theta)g(h(\mathbf{y})) \end{aligned} \tag{2-12}$$

Logo, conclui-se que para $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$, h uma função afim e f uma função convexa, a composição $g(\mathbf{x}) = f(h(\mathbf{x}))$ também é convexa pois

$$f(h(\theta\mathbf{x} + (1-\theta)\mathbf{y})) \leq \theta f(h(\mathbf{x})) + (1-\theta)g(h(\mathbf{y})) \tag{2-13}$$

2.2.3
Condição de primeira ordem e convexidade

Suponha que f seja diferenciável, ou seja, o seu gradiente ∇f existe em todos os pontos de **dom** f. Então f é convexa se, e somente se, o **dom** f é convexo e

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^T(\mathbf{y} - \mathbf{x}) \tag{2-14}$$

é valido para todo $\mathbf{x}, \mathbf{y} \in$ **dom** f. Esta desigualdade é ilustrada na Figura 2.5.

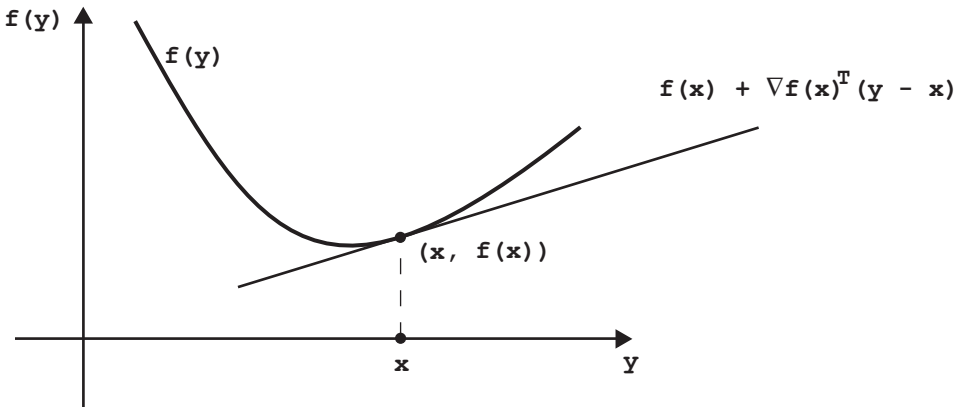

Figura 2.5: Se f é uma função convexa e diferenciável, então $f(\mathbf{x}) + \nabla f(\mathbf{x})^T(\mathbf{y} - \mathbf{x}) \leq f(\mathbf{y})$ para todo $\mathbf{x}, \mathbf{y}$ no **dom** f.

A função afim $f(\mathbf{x}) + \nabla f(\mathbf{x})^T(\mathbf{y} - \mathbf{x})$ é aproximação de primeira ordem de $f(\mathbf{y})$ em torno de $\mathbf{x}$ usando a série de Taylor. A desigualdade 2-14 estabelece que para uma função convexa, a aproximação de primeira ordem pela série de Taylor é um "subestimador" global da função. De outra forma, se a aproximação de primeira ordem pela série de Taylor é sempre um "subestimador" global da função, então a função é convexa (Bertsekas, 2016).

A convexidade estrita também pode ser caracterizada pela aproximação de primeira ordem, ou seja, f é estritamente convexa se, e somente se, o **dom** f é convexo e para todo $\mathbf{x}, \mathbf{y} \in$ **dom** f, $\mathbf{x} \neq \mathbf{y}$, tem-se

$$f(\mathbf{y}) > f(\mathbf{x}) + \nabla f(\mathbf{x})^T(\mathbf{y} - \mathbf{x}) \tag{2-15}$$

Da mesma forma, f é côncava se, e somente se, o **dom** f é convexo e

$$f(\mathbf{y}) \leq f(\mathbf{x}) + \nabla f(\mathbf{x})^T(\mathbf{y} - \mathbf{x}) \tag{2-16}$$

é valido para todo $\mathbf{x}, \mathbf{y} \in$ **dom** f. Além disso, f e estritamente côncava sempre que o **dom** f é convexo e para todo $\mathbf{x}, \mathbf{y} \in$ **dom** f, $\mathbf{x} \neq \mathbf{y}$, tem-se

$$f(\mathbf{y}) < f(\mathbf{x}) + \nabla f(\mathbf{x})^T(\mathbf{y} - \mathbf{x}) \tag{2-17}$$

Condições de Otimalidade

Condições de otimalidade podem ser obtidas a partir da aproximação de primeira ordem. Seja $\mathbb{D}$ um conjunto convexo e $f : \mathbb{R}^n \to \mathbb{R}$ uma função convexa definida sobre $\mathbb{D}$, então

a) Se f é diferenciável, então

$$\nabla f(\mathbf{x}^*)^T(\mathbf{x} - \mathbf{x}^*) \geq 0, \quad \forall \mathbf{x} \in \mathbb{D} \tag{2-18}$$

é uma condição necessária e suficiente para o vetor $\mathbf{x}^* \in \mathbb{D}$ seja um mínimo global de f em $\mathbb{D}$ (Bertsekas, 2016).

Prova: Usando o fato de f ser convexa, então

$$f(\mathbf{x}) \geq f(\mathbf{x}^*) + \nabla f(\mathbf{x}^*)^T(\mathbf{x} - \mathbf{x}^*) \quad \forall \mathbf{x} \in \mathbb{D} \tag{2-19}$$

Assumindo que $\nabla f(\mathbf{x}^*)^T(\mathbf{x} - \mathbf{x}^*) \geq 0$ é verdade para todo $\mathbf{x} \in \mathbb{D}$, então $f(\mathbf{x}) \geq f(\mathbf{x}^*)$ para todo $\mathbf{x} \in \mathbb{D}$, logo $\mathbf{x}^*$ minimiza f sobre o $\mathbb{D}$.

Por outro lado, buscando uma contradição, assuma que $\mathbf{x}^*$ minimiza f sobre o $\mathbb{D}$ e que $\nabla f(\mathbf{x}^*)^T(\mathbf{x} - \mathbf{x}^*) < 0$ para algum $\mathbf{x} \in \mathbb{D}$, então

$$\lim_{\alpha \to 0} \frac{f(\mathbf{x}^* + \alpha(\mathbf{x} - \mathbf{x}^*)) - f(\mathbf{x}^*)}{\alpha}$$

representa a derivada direcional de f em $\mathbf{x}^*$ na direção de $(\mathbf{x} - \mathbf{x}^*)$.

Se o limite for multiplicado e dividido pela constante $(\mathbf{x} - \mathbf{x}^*)$, tem-se

$$\lim_{\alpha \to 0} \frac{(\mathbf{x} - \mathbf{x}^*)[f(\mathbf{x}^* + \alpha(\mathbf{x} - \mathbf{x}^*)) - f(\mathbf{x}^*)]}{\alpha(\mathbf{x} - \mathbf{x}^*)}$$

Quando $\alpha \to 0$, $\alpha(\mathbf{x} - \mathbf{x}^*) \to 0$, logo, ao substituir $h = \alpha(\mathbf{x} - \mathbf{x}^*)$, o limite se torna

$$(\mathbf{x} - \mathbf{x}^*) \lim_{h \to 0} \frac{f(\mathbf{x}^* + h) - f(\mathbf{x}^*)}{h} = \nabla f(\mathbf{x}^*)^T(\mathbf{x} - \mathbf{x}^*) < 0 \tag{2-20}$$

Desta forma, $f(\mathbf{x}^* + \alpha(\mathbf{x} - \mathbf{x}^*))$ diminui estritamente para um valor de $\alpha > 0$ suficientemente pequeno, o que contraria a hipótese de otimalidade de $\mathbf{x}^*$ (Bertsekas, 2016).

b) Se $\mathbb{D}$ é aberto e f é diferenciável em $\mathbb{D}$, então $\nabla f(\mathbf{x}^*) = 0$ é uma condição necessária e suficiente para o vetor $\mathbf{x}^* \in \mathbb{D}$ seja um mínimo global de f sobre $\mathbb{D}$ (Bertsekas, 2016).

Prova: Se $\nabla f(\mathbf{x}^*) = 0$, a otimalidade de $\mathbf{x}^*$ segue diretamente de 2-19.

De uma maneira mais formal, $\mathbf{x}^*$ minimiza f sobre $\mathbb{D}$, logo $\mathbf{x}^*$ é viável. Para todo $\mathbf{x}$ viável, tem-se que $\nabla f(\mathbf{x}^*)^T(\mathbf{x} - \mathbf{x}^*) \geq 0$ (provado no item a)). Uma vez que f é diferenciável, seu domínio é aberto por definição, então todo $\mathbf{x}$ suficientemente próximo de $\mathbf{x}^*$ é viável. Tome $\mathbf{x} = \mathbf{x}^* - t\nabla f(\mathbf{x}^*)$, onde $t \in \mathbb{R}$ é um parâmetro. Para t pequeno e positivo, e $\mathbf{x}^*$ viável, tem-se

$$\begin{aligned} \nabla f(\mathbf{x}^*)^T(\mathbf{x} - \mathbf{x}^*) &= \nabla f(\mathbf{x}^*)^T(\mathbf{x}^* - t\nabla f(\mathbf{x}^*) - \mathbf{x}^*) \\ &= -t\nabla f(\mathbf{x}^*)^T\nabla f(\mathbf{x}^*) \\ &= -t||\nabla f(\mathbf{x}^*)||^2 \geq 0 \end{aligned} \tag{2-21}$$

o que leva a concluir que $\nabla f(\mathbf{x}^*) = 0$. Logo, $\nabla f(\mathbf{x}^*) = 0$ é uma condição necessária e suficiente para o vetor $\mathbf{x}^*$ seja o mínimo de f (Boyd e Vandenberghe, 2004).

De fato estas propriedades são vitais para muitos métodos numéricos que minimizam uma dada função. É muito comum que métodos iterativos usem o módulo do gradiente como condição de parada, ou seja, a solução é tida como ótima quando

$$||\nabla f(\mathbf{x})|| < \epsilon \tag{2-22}$$

com ϵ suficientemente pequeno.

2.2.4 Condição de segunda ordem e convexidade

Suponha que f seja duas vezes diferenciável, ou seja, sua *Hessiana* ou segunda derivada $\nabla^2 f$ existe em cada ponto no **dom** f. Então, f é convexa se, e somente se, o **dom** f é convexo e sua Hessiana é positiva semi-definida (Boyd e Vandenberghe, 2004).

Esta condição pode ser interpretada geometricamente como uma condição em que o gráfico da função tenha uma curvatura positiva em $\mathbf{x}$. Da mesma forma, f é côncava se, e somente se, o **dom** f é convexo e sua Hessiana é negativa semi-definida para todo $\mathbf{x} \in$ **dom** f. A convexidade restrita pode ser parcialmente caracterizada pela condição de segunda ordem. Se a Hessiana de uma função f for positiva definida para todo $\mathbf{x}$ no **dom** f, então f é estritamente convexa, no entanto, o contrário não é verdadeiro. Por exemplo, a função $f : \mathbb{R} \to \mathbb{R}$ dada por $f(x) = x^4$ é estritamente convexa, mas tem sua segunda derivada nula em $x = 0$.

2.3 Programação Matemática

Um *problema de otimização matemática*, ou simplesmente *problema de otimização*, tem a forma geral

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & f(\mathbf{x}) \\ \text{sujeito a} \quad & g_i(\mathbf{x}) \leq 0,\ i = 1, \ldots, m. \\ & h_i(\mathbf{x}) = 0,\ i = 1, \ldots, p. \end{aligned} \tag{2-23}$$

onde o vetor $\mathbf{x} = (x_1, ..., x_n)$ são as *variáveis de projeto* do problema. A função $f : \mathbb{R}^n \to \mathbb{R}$ é a *função objetivo*. As funções $g_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, ..., m$ são *restrições* de desigualdade. As funções $h_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, ..., p$ são *restrições* de igualdade. Se não houver restrições, o problema é dito ser *irrestrito*.

O conjunto de pontos $\mathbb{D}$ no qual a função objetivo e todas as funções de restrições são definidos

$$\mathbb{D} = \bigcap_{i=0}^{m} \mathbf{dom}\ g_i \bigcap_{i=1}^{p} \mathbf{dom}\ h_i \tag{2-24}$$

é chamado de *domínio* do problema de otimização 2-23. Um ponto $\mathbf{x} \in \mathbb{D}$ é dito *viável* se ele satisfaz todas as restrições $g_i(\mathbf{x}) \leq 0,\ i = 1, \ldots, m$ e $h_i(\mathbf{x}) = 0,\ i = 1, \ldots, p$. O problema é dito viável se existe ao menos um ponto viável, caso contrário, o problema é dito *inviável*. O conjunto de todos os pontos viáveis será denotado por X.

2.3.1 Mínimo local e global

Um vetor $\mathbf{x}^*$ é um *mínimo local* de um problema de otimização irrestrito, se $f(\mathbf{x}^*)$ tem o valor mínimo dentre seus vizinhos, ou seja, existe um $\epsilon > 0$ tal que

$$f(\mathbf{x}^*) \leq f(\mathbf{x}), \qquad \forall \mathbf{x} \in \mathbb{D} \quad \text{com} \quad ||\mathbf{x} - \mathbf{x}^*|| < \epsilon. \tag{2-25}$$

Um vetor $\mathbf{x}^*$ é um *mínimo global* de um problema de otimização irrestrito se $f(\mathbf{x}^*)$ tem o valor mínimo dentre todos os vetores no **dom** f, ou seja,

$$f(\mathbf{x}^*) \leq f(\mathbf{x}), \qquad \forall \mathbf{x} \in \mathbb{D} \tag{2-26}$$

O ponto de mínimo local ou global irrestrito $\mathbf{x}^*$ é dito *único* ou *estrito* se as inequações acima são estritamente satisfeitas sempre que $\mathbf{x}^* \neq \mathbf{x}$. A Figura 2.6 ilustra essas definições.

Estas definições de mínimo global e local podem ser estendidas para o caso onde a função f deve ser minimizada sujeita a um conjunto de restrições. Neste caso, o espaço de soluções é restrito ao conjunto viável $X \subset \mathbb{R}^n$. Desta forma, um

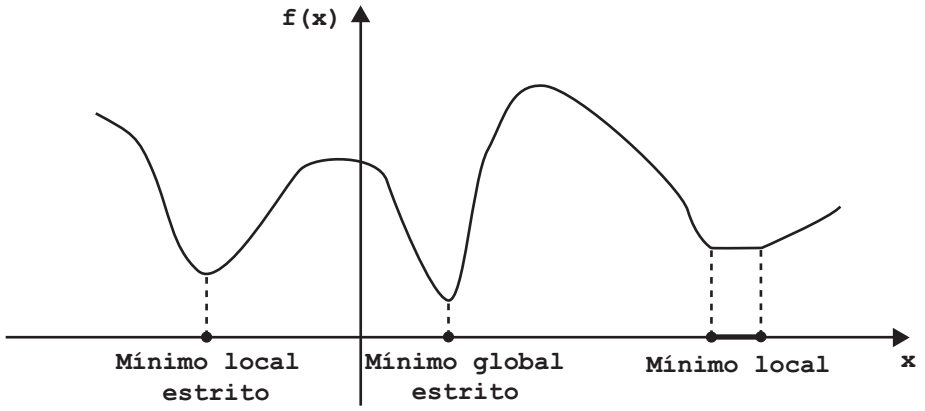

Figura 2.6: Mínimos local e global de um problema de otimização irrestrito.

vetor $\mathbf{x}^*$ é um *local mínimo* de f sobre o conjunto X se $\mathbf{x}^* \in X$ e existe um $\epsilon > 0$ tal que

$$f(\mathbf{x}^*) \leq f(\mathbf{x}), \qquad \forall \mathbf{x} \in X \quad \text{com} \quad ||\mathbf{x} - \mathbf{x}^*|| < \epsilon. \tag{2-27}$$

como ilustrado na Figura 2.7.

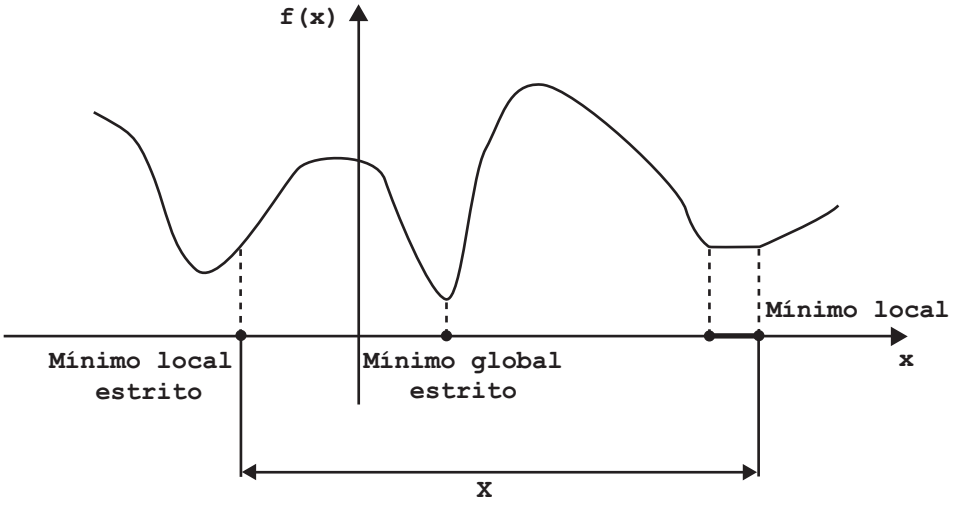

Figura 2.7: Mínimos local e global de um problema de otimização restrito.

As definições para mínimo global e mínimo estrito de f sobre X são análogas.

2.4 Otimização Convexa

Um *problema de otimização convexa* é da forma

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} \quad & f(\mathbf{x}) \\ \text{sujeito a} \quad & g_i(\mathbf{x}) \leq 0,\ i = 1, \ldots, m. \\ & \mathbf{a}_i^T \mathbf{x} = b_i,\ i = 1, \ldots, p. \end{aligned} \tag{2-28}$$

onde $f, \ldots, f_m$ são funções convexas. Em relação à forma padrão do problema de otimização matemática 2-23, um problema de otimização convexa tem as seguintes restrições extras:

a) a função objetivo deve ser convexa,

b) as restrições de desigualdade devem ser convexas,

c) as restrições de igualdade $h_i(\mathbf{x}) = \mathbf{a}_i^T\mathbf{x} - b_i$ devem ser funções afim.

Note que o conjunto viável de um problema de otimização convexa é convexo, uma vez que ele é dado pela interseção dos domínios das funções envolvidas no problema

$$\mathbb{D} = \bigcap_{i=0}^{m} \mathbf{dom}\ g_i \tag{2-29}$$

que é um conjunto convexo. Desta forma, em um problema de otimização convexa, a função objetivo é minimizada sobre um conjunto convexo.

Uma propriedade fundamental de um problema de otimização convexa é que qualquer ponto que seja um ótimo local, é também um ótimo global.

2.5 Otimização quadrática Convexa

Um caso especial do problema de otimização convexa é o *problema de otimização quadrática convexo*. Neste caso, a função objetivo é quadrática e convexa e as restrições são funções afins. Esta classe de problema pode ser expressa da seguinte forma

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} \quad & \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{q}^T\mathbf{x} + c \\ \text{sujeito a} \quad & \mathbf{G}\mathbf{x} \leq \mathbf{h} \\ & \mathbf{A}\mathbf{x} = \mathbf{b} \end{aligned} \tag{2-30}$$

onde $\mathbf{Q}$ é uma matriz quadrada simétrica e positiva semidefinida, $\mathbf{G} \in \mathbb{R}^{m\times n}$ e $\mathbf{A} \in \mathbb{R}^{p\times n}$. Em um problema de otimização quadrática, uma função quadrática é minimizada sobre um hiperpoliedro, como ilustrado na Figura 2.8.

Quando o problema de otimização 2-30 possui também restrições quadráticas, ele é conhecido como *problema de otimização quadrática quadraticamente restrito*. Neste caso, o problema não seria minimizado sobre um poliedro, e sim sobre uma região resultante da interseção de elipsoides. Porém, esta classe de problema não é alvo de estudo neste trabalho.

Note que no caso de funções quadráticas, a matriz Hessiana $\nabla^2 f(\mathbf{x}) = \mathbf{Q}$ é constante para todo $\mathbf{x}$. Logo, f é convexa se e somente se $\mathbf{Q}$ é positiva semi-definida e côncava se e somente se $\mathbf{Q}$ é negativa semi-definida. Diferentemente do caso geral, que foi ilustrado na Seção 2.2.4, a convexidade restrita para um

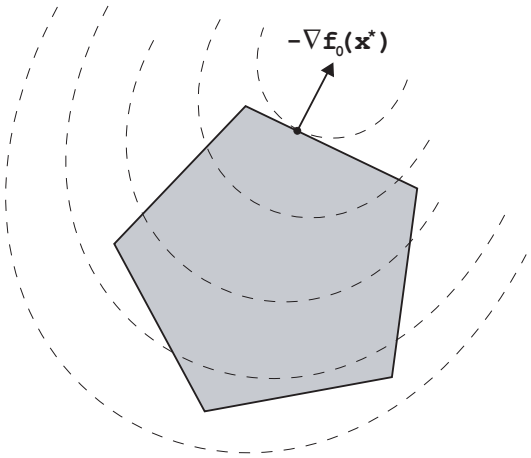

Figura 2.8: Região viável, que é um poliedro, do problema de otimização quadrático mostrado de forma sombreada. As curvas de nível da função objetivo são mostradas pontilhadas.

problema de otimização quadrática e irrestrito é completamente caracterizada pela condição de segunda ordem, ou seja, f é estritamente convexa se e somente se $\mathbf{Q}$ é positiva definida e estritamente côncava se e somente se $\mathbf{Q}$ é positiva negativa (Boyd e Vandenberghe, 2004).

No caso em que o problema de otimização quadrático convexo é irrestrito, ou naqueles casos em que ele possa ser reduzido a um problema irrestrito equivalente, uma condição necessária e suficiente para que $\mathbf{x}$ seja uma solução de f (Seção 2.2.3) é

$$\nabla f(\mathbf{x}) = \mathbf{Q}\mathbf{x} + \mathbf{q} = 0 \tag{2-31}$$

Basicamente três casos podem acontecer, dependendo se esta equação linear não tem solução, tem solução única ou tem múltiplas soluções:

a) Se $\mathbf{q}$ não pertence ao *range* de $\mathbf{Q}$, ou seja, $\mathbf{q}$ não pode escrito como combinação linear das colunas de $\mathbf{Q}$, então não existe solução e f é ilimitada inferiormente.

b) Se $\mathbf{Q} \succ 0$, então f é estritamente convexa e a solução ótima é única e dada por $\mathbf{x}^* = -\mathbf{Q}^{-1}\mathbf{q}$.

c) Se $\mathbf{Q}$ é singular, mas $\mathbf{q}$ pertence ao *range* de $\mathbf{Q}$, então existe um conjunto afim de soluções.

3
Operador Laplaciano para Suavização de Malhas

A maneira mais simples e comum de posicionar um vértice de uma superfície durante um processo de suavização de uma malha é movê-lo para o centroide de seus vértices vizinhos (Cheng *et al.*, 2012). Este método é chamado de *suavização laplaciana* por seu operador discreto ser interpretado como um Laplaciano calculado por diferenças finitas. No entanto, esta é só uma das muitas formas discretas do operador de Laplace para superfícies. A discretização do operador de Laplace tem desempenhado um papel central em um amplo espectro de aplicações relacionadas à computação gráfica como filtragem, suavização, parametrização, reconstrução, compressão, deformação e edição de malhas. Este capítulo introduz alguns dos conceitos fundamentais do operador de Laplace-Beltrami, assim como as propriedades de suas diferentes formas de discretização, além da sua relação com os problemas de difusão de fluxo, superfícies mínimas e minimização da Energia de Dirichlet.

3.1
Operador de Laplace

De forma geral, o operador de Laplace é definido como o divergente do gradiente, ou seja

$$\Delta f = \nabla^2 f = \nabla \cdot (\nabla f). \tag{3-1}$$

Para uma função escalar de dois parâmetros $f(u, v)$ no espaço Euclidiano, este operador diferencial de segunda ordem pode ser escrito como a soma das segundas derivadas parciais

$$\Delta = \text{div}\, \nabla f(u, v) = f_{uu} + f_{vv}, \tag{3-2}$$

onde $f_{uu} = \dfrac{\partial^2 f(u,v)}{\partial u^2}$ e $f_{vv} = \dfrac{\partial^2 f(u,v)}{\partial v^2}$.

O operador de *Laplace-Beltrami* estende esse conceito para funções definidas sobre superfícies. Para uma dada função f definida em uma superfície *manifold* $\mathcal{S}$, o operador Laplace-Beltrami é definido como

$$\Delta_{\mathcal{S}} f = \text{div}\, {}_{\mathcal{S}}\nabla_{\mathcal{S}} f, \tag{3-3}$$

o qual requer uma definição adequada dos operadores gradiente e divergente em *manifolds*. Quando aplicado sobre a função das coordenadas $\mathbf{x}$ da superfície, o operador avalia a *curvatura normal média* (Do Carmo, 1976), ou seja

$$\Delta_{\mathcal{S}}\mathbf{x} = -2H\mathbf{n}, \tag{3-4}$$

onde $\mathbf{n}$ é normal no vértice e H sua curvatura normal.

3.2 Difusão de Fluxo

As discretizações do operador de Laplace-Beltrami são intimamente ligadas ao problema de difusão de fluxo. Difusão de fluxo é um problema matematicamente bem compreendido para um processo de suavização de um sinal $f(\mathbf{x}, t)$ ao longo do tempo t (Botsch *et al.*, 2010). Processos físicos, como difusão de calor, podem ser descritos pela difusão de fluxo.

A difusão de fluxo é modelada pela *equação de difusão*

$$\frac{\partial f(\mathbf{x}, t)}{\partial t} = \lambda \Delta f(\mathbf{x}, t) \tag{3-5}$$

A Equação 3-5 é uma equação diferencial parcial (EDP) linear de segunda ordem, que define que a função f varia ao longo do tempo por um coeficiente escalar de difusão λ vezes o seu laplaciano $\Delta f(\mathbf{x}, t)$. Se, por exemplo, problema de difusão de calor fosse modelado, $f(\mathbf{x}, t)$ denotaria a temperatura de um ponto material $\mathbf{x}$ no tempo t e a equação seria chamada de *equação de calor*.

É possível empregar a equação de difusão para suavizar uma função arbitrária $f : \mathcal{S} \rightarrow \mathbb{R}$ em uma superfície *manifold* $\mathcal{S}$, apenas substituindo o operador de Laplace pelo operador de Laplace-Beltrami. Uma vez que a Equação 3-5 é uma EDP contínua que depende do tempo, é comum discretizá-la tanto no espaço quanto no tempo.

Para discretização espacial, a função f pode ser substituída pelos valores amostrados nos vértices da malha $(f(v_1, t), ..., f(v_n, t))$ e calcular o Laplace-Beltrami usando uma de suas discretizações (Seção 3.4). Isto leva a uma equação para a evolução do valor da função no tempo em cada vértice

$$\frac{\partial}{\partial t} f(v_i, t) = \lambda \Delta_{\mathcal{S}} f(v_i, t),\ i = 1, ..., n \tag{3-6}$$

Se o operador Laplace-Beltrami for discretizado em uma matriz $\mathbf{L}$, a equação acima pode ser escrita como:

$$\frac{\partial}{\partial t}\mathbf{f}(t) = \lambda \mathbf{L}\mathbf{f}(t) \tag{3-7}$$

Já a discretização temporal pode ser feita dividindo-se o eixo do tempo de forma regular em intervalos de tamanho h, ou seja, $\{t, t+h, t+2h, ...\}$. Se diferenças

finitas forem usadas para aproximar a derivada do tempo, pode-se aproximar:

$$\frac{\partial \mathbf{f}(t)}{\partial t} \approx \frac{\mathbf{f}(t+h) - \mathbf{f}(t)}{h} \tag{3-8}$$

e resolver para $\mathbf{f}(t+h)$. Isto leva à *integração explícita de Euler*

$$\mathbf{f}(t+h) = \mathbf{f}(t) + h\frac{\partial \mathbf{f}(t)}{\partial t} = \mathbf{f}(t) + h\lambda \mathbf{L}\mathbf{f}(t) \tag{3-9}$$

Para suavizar a função da geometria $\mathbf{x}$ da malha, em vez de uma função arbitrária $\mathbf{f}$, basta aplicar a regras de atualização acima para as posições dos vértices. A atualização explícita por vértice é conhecida como *suavização laplaciana* (Botsch *et al.*, 2010):

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + h\lambda \Delta_{\mathcal{S}} \mathbf{x}_i \tag{3-10}$$

Uma vez que o Laplace-Beltrami das posições dos vértices correspondem à curvatura normal média ($\Delta \mathbf{x} = -2H\mathbf{n}$), todos os vértices se movem na direção da normal por uma quantidade determinada pela curvatura normal H.

3.3 Fairing

Na equação de difusão, o fluxo ocorre com maior intensidade onde a taxa de variação é alta, com isso ele tende a suavizar as altas frequências que comumente são os ruídos. Infelizmente os detalhes também são altas frequências e a suavização tende a eliminá-los também. Já o *fairing* tem o objetivo de calcular formas o mais suaves possível (Botsch *et al.*, 2010). A maneira que se mede suavidade ou o nível de *fairing* depende da aplicação, mas em geral *fairing* produz superfícies com o *princípio da forma mais simples*, ou seja, a superfície não deve ter detalhes ou oscilações desnecessárias (Welch e Witkin, 1992). Isto pode ser modelado por uma energia que penaliza comportamentos não desejados da superfície. A minimização desta energia levaria à superfície desejada. Uma energia que é frequentemente usada é a energia de membrana, que para uma superfície paramétrica suave $\mathbf{x} : \Omega \rightarrow \mathcal{S}$ é dada por

$$E_M(\mathbf{x}) = \int \int_{\Omega} \sqrt{\det(\mathbf{I})}\ dudv \tag{3-11}$$

onde $\mathbf{I}$ é a primeira forma fundamental. A energia de membrana $E_M(\mathbf{x})$ mede a área da superfície $\mathcal{S}$ (Seção A.1.1). Esta energia é minimizada sob restrições na fronteira $\partial\Omega$ que tipicamente fixam os vértices da borda da superfície (Botsch *et al.*, 2010). A superfície resultante é dita *superfície mínima*.

Garantir a robustez e eficiência da minimização de $E_M(\mathbf{x})$ é uma tarefa muito difícil, uma vez que $E_M(\mathbf{x})$ é altamente não linear, contendo a raiz quadrada do determinante da primeira forma fundamental, que também não é uma expressão linear. Por isso, a energia de membrana é simplificada, substituindo o determinante da primeira forma fundamental pelas derivadas de primeira ordem, o que leva à

Energia de Dirichlet (Botsch *et al.*, 2010)

$$E_D(\mathbf{x}) = \int \int_{\Omega} (||\mathbf{x}_u||^2 + ||\mathbf{x}_v||^2)\ dudv \tag{3-12}$$

onde $\mathbf{x}_u = \dfrac{\partial \mathbf{x}}{\partial u}$ e $\mathbf{x}_v = \dfrac{\partial \mathbf{x}}{\partial v}$.

Note que $E_D(\mathbf{x})$ é nula para funções constantes, não negativa e quadrática, uma vez que o operador de derivada parcial é linear. Empregando cálculo variacional para minimizar $E_D(\mathbf{x})$ (Kobbelt, 1997), pode ser demonstrado que, se $E_D(\mathbf{x})$ é mínima, então

$$\begin{aligned} \Delta\mathbf{x}(u,v) &= 0 \qquad && (u,v) \in \Omega \setminus \partial\Omega \\ \mathbf{x}(u,v) &= \mathbf{x}_0(u,v) && (u,v) \in \partial\Omega \end{aligned} \tag{3-13}$$

onde $\mathbf{x}_0(u,v)$ são os valores da função no bordo.

A Equação 3-13 é conhecida como *Equação de Laplace*. Em um paralelo com o problema de difusão de calor, a solução para esta equação definiria um estado estacionário, como ilustrado na Figura 3.1.

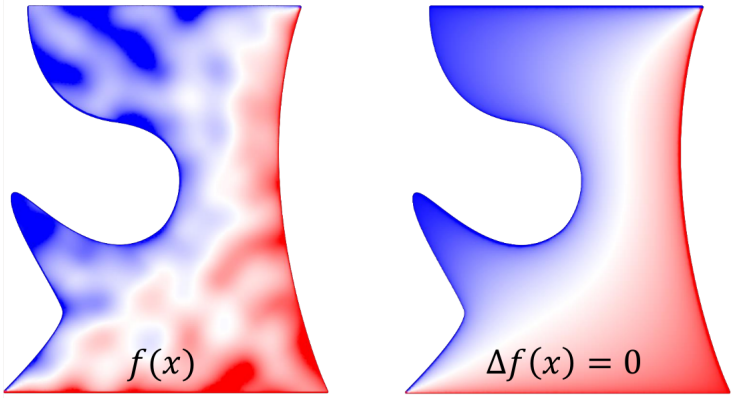

Figura 3.1: À esquerda os valores iniciais de uma função em um domínio. À direita, os valores da função após a minimização da energia de Dirichlet respeitando os valores da fronteira.

Funções que são duas vezes diferenciáveis e satisfazem a Equação de Laplace são ditas *funções harmônicas* (Axler *et al.*, 2001). Dentre suas propriedades, a seguintes são as mais importantes para este trabalho

- se Ω é limitado e a função é contínua e harmônica em Ω, então seus valores de máximo e mínimo estão em $\partial\Omega$ (Axler *et al.*, 2001).
- se Ω é conexo, então a função possui apenas mínimos ou máximos globais (*o princípio máximo*) (Axler *et al.*, 2001).
- se Ω é limitado, o princípio máximo implica que se uma solução para o problema de Dirichlet existe, então ela é única (Axler *et al.*, 2001).

3.4 Discretizações do Operador de Laplace-Beltrami

O operador de Laplace-Beltrami desempenha um papel central em muitas aplicações, como processamento de imagens (Kimmel *et al.*, 1997; Weickert *et al.*, 1998; Bertalmio *et al.*, 2000; Sapiro, 2006), no estudo de equações diferenciais geométricas (Bertalmio *et al.*, 2000; Sapiro, 2006) e principalmente no processamento geométrico de superfícies, como filtragem de malhas (Vallet e Levy, 2008; Kovnatsky *et al.*, 2012), parametrização (Floater, 1997; Gotsman *et al.*, 2003), deformação e edição de malhas (Lipman *et al.*, 2004; Sorkine *et al.*, 2004; Zhou *et al.*, 2005), segmentação, reconstrução, *remeshing*, compressão de malhas (Karni e Gotsman, 2011; Sorkine *et al.*, 2003), suavização de malhas (Taubin, 1995; Desbrun *et al.*, 1999; Zhang e Fiume, 2003), simulação, interpolação, entre outras aplicações. Nestas aplicações, as superfícies a serem processadas são normalmente representadas como malhas discretas, o que leva à necessidade de discretizar o operador de Laplace-Beltrami (Li *et al.*, 2015).

3.4.1 Forma geral do operador de Laplace-Beltrami

Considere uma malha $\mathcal{M}$. De forma geral, o operador discreto de Laplace-Beltrami sobre $\mathcal{M}$ é definido em um vértice v_i como

$$\Delta_{\mathcal{S}} f(v_i) = w_i \sum_{v_j \in N_1(v_i)} w_{ij}(f_{v_j} - f_{v_i}) \tag{3-14}$$

onde $N_1(v_i)$ representa o conjunto dos vértices na estrela do vértice v_i, w_{ij} são pesos atribuídos às arestas que conectam os vértices v_i e v_j e w_i um peso atribuído ao vértice central v_i.

Existem muitas formas de definir os pesos em questão, cada uma delas gera uma discretização diferente do operador de suavização Laplace-Beltrami. Em geral, uma aplicação frequentemente requer certas propriedades estruturais do operador. (Wardetzky *et al.*, 2007) citam algumas propriedades desejáveis, dentre elas:

- **Simetria**: $w_{ij} = w_{ji}$, uma vez que matrizes simétricas tem autovalores reais e autovetores ortogonais.
- **Localidade**: os pesos são associados às arestas da malha de forma que $w_{ij} = 0$ se v_i e v_j não compartilham aresta em $\mathcal{M}$. Desta forma, ao mudar o valor de f_{v_j}, o valor de $\Delta_{\mathcal{S}} f(v_i)$ não será alterado se não existir uma aresta conectando v_i e v_j.
- **Precisão Linear**: $\Delta_{\mathcal{S}} f(v_i) = 0$ para todos os vértices interiores sempre que $\mathcal{M}$ é uma linha reta imersa em um plano e f é uma função linear no plano. Isto é equivalente a

$$\Delta_{\mathcal{S}} f(v_i) = w_i \sum_{v_j \in N_1(v_i)} w_{ij}(f_{v_j} - f_{v_i}) = 0 \tag{3-15}$$

– **Pesos Positivos**: $w_{ij} \geq 0$ sempre que $i \neq j$. Além disso, para cada vértice v_i, existe ao menos um vértice v_j de modo que $w_{ij} > 0$. Isto é uma condição suficiente para o *princípio de máximo*, ou seja, não existem máximos locais nos vértices interiores. Fisicamente, em problemas de difusão, esta propriedade garante que o fluxo flui das regiões de potencial mais alto para as regiões de potencial mais baixo. Além disso, isto estabelece uma conexão com *coordenadas baricêntricas* uma vez que podemos calcular

$$\lambda_{ij} = \frac{w_{ij}}{\sum_{i \neq j} w_{ij}} \tag{3-16}$$

de modo que

$$\sum_{i \neq j} \lambda_{ij} = 1 \tag{3-17}$$

Considerando o caso geral de uma malha $\mathcal{M}$, ou seja, onde não são impostas restrições, como regularidade, sobre $\mathcal{M}$, (Wardetzky *et al.*, 2007) demonstraram que não existe discretização do operador de Laplace-Belatrami que satisfaça, simultaneamente, as propriedades de simetria, localidade, linearidade e positividade. Isto explica a diversidade de discretizações propostas na literatura, uma vez que diferentes aplicações exigem diferentes discretizações.

De forma simplificada, podemos dividir esses operadores em dois grupos: operadores combinatórios e operadores geométricos. Os operadores combinatórios são aqueles que definem os pesos baseando-se apenas nas conexões dos vértices, já os operadores geométricos, codificam também propriedades geométricas da malha de entrada.

3.4.2 Operadores Combinatórios

Os operadores combinatórios definem os pesos baseando-se apenas nas conexões dos vértices, como por exemplo: operador uniforme ou *umbrella operator* (Taubin, 1995; Kobbelt *et al.*, 1998; Gotsman *et al.*, 2003) e operador Kirchhoff (Zhang, 2004). Outros operadores combinatórios podem ser encontrados em (Zhang, 2004).

Todos estes operadores são definidos para o vértice v_i usando apenas as informações de vizinhança da malha, como na Figura 3.2.

Os pesos para o operador Kirchhoff podem ser definidos da seguinte forma:

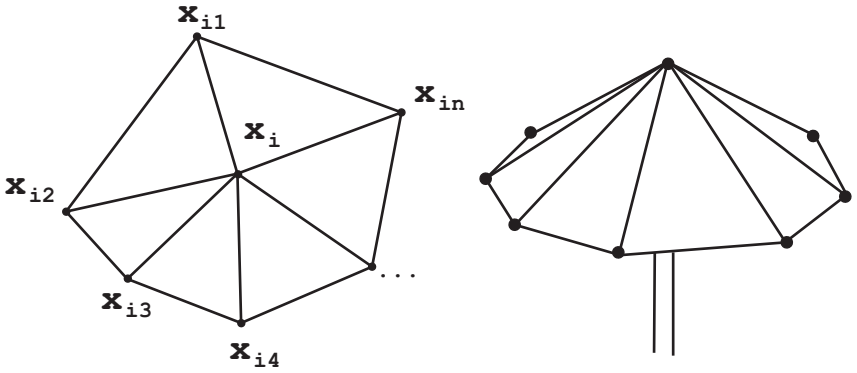

Figura 3.2: Para calcular o valor do Laplace-Beltrami em um vértice $\mathbf{x}_i$, as informações dos vértices $\mathbf{x}_{ij}$ na estrela de $\mathbf{x}_i$ são levadas em conta. O operador uniforme é comumente chamado de *umbrela* porque conjunto de vértices que compõem o operador, em uma malha triangular, se assemelha a uma sombrinha.

$$L_{v_i,v_j} = \begin{cases} -|N_1(v_i)| & \text{se } v_i = v_j \\ 1 & \text{se } i \neq j \text{ e } v_j \in N_1(v_i) \\ 0 & \text{caso contrário} \end{cases} \tag{3-18}$$

onde $|N_1(v_i)|$ é a valência do vértice v_i. O valor de $\Delta_{\mathcal{S}} f(v_i)$ usando o operador Kirchhoff é dado por

$$\Delta_{\mathcal{S}} f(v_i) = \sum_{v_j \in N_1(v_i)} (f_{v_j} - f_{v_i}) \tag{3-19}$$

Se for usado o operador uniforme, os pesos são definidos como

$$L_{v_i,v_j} = \begin{cases} -1 & \text{se } v_i = v_j \\ \dfrac{1}{|N_1(v_i)|} & \text{se } v_i \neq v_j \text{ e } v_j \in N_1(v_i) \\ 0 & \text{caso contrário} \end{cases} \tag{3-20}$$

e o valor de $\Delta_{\mathcal{S}} f(v_i)$, neste caso, é dado como a seguir

$$\Delta_{\mathcal{S}} f(v_i) = \frac{1}{|N_1(v_i)|} \sum_{v_j \in N_1(v_i)} (f_{v_j} - f_{v_i}) \tag{3-21}$$

Note que a diferença entre os dois operadores é apenas uma normalização pela valência do vértice central. Em geral, estas diferenças são causadas pela busca de diferentes propriedades na matriz $\mathbf{L}$ usada no passo de suavização da Equação 3-9. Neste caso, por exemplo, é sabido que os autovalores da matriz simétrica gerada pelo operador Kirchhoff estão no intervalo $[0, 2d_{max}]$, onde d_{max} é a máxima valência da malha. Já no caso do operador uniforme, os autovalores estão no intervalo $[0, 2]$ e a matriz não é simétrica em geral, o que faz com que seus autovalores não sejam ortogonais (Zhang, 2004). (Taubin, 1995) notou que, no caso

da matriz do operador uniforme, seus autovetores representam a vibração natural dos nós da malha, enquanto seus autovalores capturam sua frequência natural, assemelhando-se ao cenário da transformada discreta de Fourier.

Quando aplicado sobre a função das coordenadas **x**, o *operador laplaciano uniforme* avalia o vetor que aponta do vértice central $\mathbf{x}_i$ para a média dos vértices vizinhos $\mathbf{x}_j$. Apesar de serem simples e eficientes de serem avaliados, ambos operadores falham em serem geométricos, ou seja, violam a propriedade da precisão linear. Isto significa que o vetor resultante pode ser não nulo mesmo para configurações planares dos vértices. No entanto, o esperado nesta configuração seria um laplaciano nulo, uma vez que a curvatura média sobre a malha é nula. Isto indica que estes operadores não fornecem discretizações precisas para malhas não uniformes. Como o operador depende da conectividade dos vértices, ele não se adapta a todas as distribuições espaciais dos vértices, porém isto pode ser uma vantagem em aplicações que buscam melhorar a distribuição local dos vértices da malha (Botsch *et al.*, 2010).

3.4.3 Operadores geométricos

Os operadores geométricos levam em conta a geometria corrente da malha, além de sua conexão. Uma discretização mais precisa para o operador de Laplace-Beltrami, para malhas triangulares, foi derivada por (Meyer *et al.*, 2003) usando um misto de elementos finitos e volumes finitos. Para tal, o divergente do gradiente de uma função linear por partes é integrado sobre uma área $A_i = A(v_i)$ em torno de um vértice v_i, ou seja

$$\int_{A_i} \text{div } \mathbf{F}(\mathbf{u}) dA \tag{3-22}$$

resultando na seguinte discretização:

$$\Delta f(v_i) := \frac{1}{2A_i} \sum_{j \in N_1(v_i)} (\cot \alpha_{i,j} + \cot \beta_{ij})(f_j - f_i) \tag{3-23}$$

onde a vizinhança, os ângulos e área envolvidos são ilustrados na Figura 3.3. Uma demonstração da discretização da cotangente do operador de Laplace-Beltami pode ser encontrada em (Botsch *et al.*, 2010).

A Equação 3-23 é conhecida como *método da cotangente* e apareceu pela primeira vez em (Pinkall e Polthier, 1993). Desde então, existe uma quantidade imensa de operadores que são variações do método da cotangente derivados de diversas formas diferentes, como elementos finitos, volumes finitos, cálculo exterior, entre outros. A Equação 3-23 é provavelmente a mais utilizada para discretização do operador de Laplace-Beltrami para malhas triangulares em computação gráfica e é tipicamente aplicada em várias tarefas de processamento

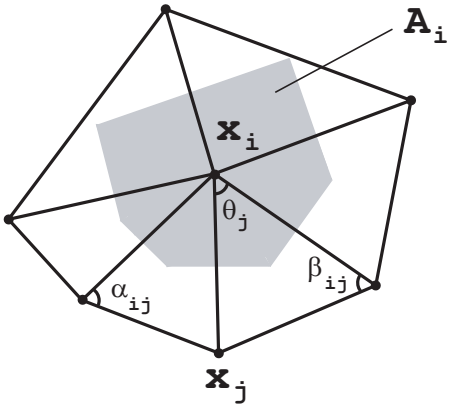

Figura 3.3: Ilustração das quantidades utilizadas na discretização do operador de Laplace-Beltrami utilizando a fórmula da cotangente. Neste caso, o peso atribuído à aresta é dado por $w_{ij} = \cot \alpha_{i,j} + \cot \beta_{ij}$. Já o peso atribuído ao vértice é dado por $w_i = \frac{1}{2A_i}$.

geométrico como suavização de superfícies, parametrização e modelagem de formas (Botsch *et al.*, 2010). No entanto, este operador viola a propriedade de positividade, uma vez que $w_{ij} = \cot \alpha_{i,j} + \cot \beta_{ij}$ torna-se negativo quando $\alpha_{i,j} + \beta_{ij} > \pi$. Para tentar contornar o problema, tenta-se avaliar o operador em uma triangulação de Delaunay. No entanto, esta solução mostra que o operador não é intrínseco à superfície, uma vez que é dependente da triangulação.

Operadores Geométricos para malhas não triangulares

O operador clássico de Laplace em espaços planos é geralmente aproximado por diferenças finitas. No entato, devido à complexidade e à diversidade das discretizações de superfícies *manifold*, uma discretização precisa do operador de Laplace-Beltrami é muito mais complicada (Li *et al.*, 2015). Existem muitas propostas de discretizações geométricas na literatura para malhas de triângulos (Pinkall e Polthier, 1993; Taubin, 1995; Desbrun *et al.*, 1999; Mayer, 2001; Meyer *et al.*, 2003; Xu, 2004), a maioria delas são variações do método da cotangente primeiramente proposto por (Pinkall e Polthier, 1993).

Para malha de quadriláteros, ou malhas poligonais gerais, existem poucos resultados na literatura. (Liu *et al.*, 2008) descrevem uma discretização do operador de Laplace-Beltrami em malhas de quadriláteros baseada em uma interpolação bilinear do quadrilátero. (Xiong *et al.*, 2011) obtiveram uma discretização do operador de Laplace-Beltrami em malhas de quadriláteros usando o esquema da cotangente para obter uma média sobre todas as possíveis triangulações. (Alexa e Wardetzky, 2011) estenderam o esquema da cotangente para o caso geral de malhas poligonais.

3.4.4
Operadores combinatórios versus operadores geométricos

Como visto na Seção 3.2, ao aplicar o passo de suavização laplaciana da Equação 3-9, os vértices deveriam se mover na direção da sua normal. Este movimento é verdadeiro (de forma aproximada) para a discretizações usando o método da cotangente. No entanto, ele não é verdadeiro para a discretização uniforme do operador de Laplace-Beltrami. O motivo disso, é que este operador não leva em consideração a geometria da malha. A suavização com o operador uniforme tenta mover cada vértice para o baricentro dos seus vértices vizinhos, desta forma, a malha é suavizada e regularizada ao mesmo tempo. Dependendo da aplicação, esta propriedade pode ser desejável ou não. A Figura 3.4 ilustra a diferença dos dois operadores quando aplicados em uma malha não uniforme.

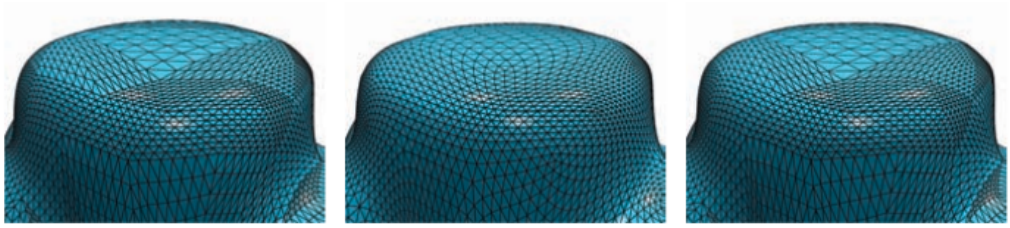

Figura 3.4: Quando a malha de entrada à esquerda é suavizada com o operador uniforme, ela é regularizada, gerando a malha do centro da imagem. Quando a mesma malha é suavizada com o método da cotangente, a malha preserva a distribuição dos seus vértices, resultando na malha da direita. Fonte: (Botsch *et al.*, 2010)

Desta forma, os operadores combinatórios são mais indicados para problema de geração de malhas, uma vez que não se tem conhecimento da geometria dos vértices da malha no instante inicial, de forma que todos os vértices poderiam ter suas posições iniciais definidas arbitrariamente. Em contrapartida, os operadores geométricos são mais indicados para problemas como o de suavização da malha de entrada, removendo assim as altas frequências enquanto as baixas frequências são preservadas. Como cada operador tem a deficiência de ao menos uma das propriedades desejáveis, um conjunto de discretizações diferentes podem ser usadas em etapas diferentes do processo de geração de malhas.

A grande maioria dos operadores geométricos construídos são para malhas triangulares. No entanto, os operadores combinatórios são facilmente estendidos para qualquer tipo de malha, uma vez que dependem apenas da topologia das malhas envolvidas. Além disso, os operadores geométricos precisam ser recalculados a cada vez que a geometria muda, o que resulta em um custo computacional maior. Porém, a dependência apenas da conectividade torna os operadores combinatórios menos

robustos e precisos. Este é um *trade-off* que as aplicações precisam gerenciar ao tratar malhas gerais.

4
Geração de malhas geomecânicas de reservatórios

Este capítulo faz uma revisão sobre modelagem geomecânica de reservatórios de óleo e gás, bem como métodos para geração de malhas de hexaedros e de geração de malhas de hexaedros para reservatórios.

4.1
Modelagem Geomecânica de Reservatórios de óleo e gás

A maioria dos problemas na indústria de óleo e gás são decorrentes de fenômenos de fluxo e deformação. Indiscutivelmente, ambos fenômenos acontecem de forma acoplada afetando simultaneamente o desenvolvimento de um sobre o outro. As reduções de pressão dentro do reservatório devido à produção de óleo, por exemplo, podem gerar deformações e alterações de tensão que podem afetar o comportamento geomecânico de um campo, ocasionando riscos de problemas de compactação, subsidência superficial, reativação de falhas geológicas, produção de areia e instabilidade de poços. Por sua vez, as deformações geradas pelos processos geomecânicos, podem alterar significativamente a porosidade e a permeabilidade do reservatório, afetando a produção em um campo de óleo e gás.

Este tipo de problemas, em que os processos de fluxo e deformação estão extremamente ligados entre si, são denominados de problemas de acoplamento hidromecânico. Para sua modelagem, duas equações governantes devem ser solucionadas, uma correspondente à conservação de massa, para fluxo, e a outra correspondente ao equilíbrio de forças, para geomecânica. Idealmente, ambas equações devem ser solucionadas simultaneamente através de um único sistema de equações considerando, não apenas os termos inerentes a cada processo, mas também os termos de acoplamento entre um processo e outro. Na literatura, esta abordagem é conhecida como solução de acoplamento total, verdadeiro ou implícito. Desafortunadamente, para aplicações práticas, o desenvolvimento de um simulador completamente acoplado é geralmente complicado e muitas vezes inviável por várias questões. Em primeiro lugar, o tratamento da não linearidade das relações constitutivas em cada processo pode ser extremamente complexo em função da ocorrência de deformações inelásticas e/ou fluxo multifásico. Além disso, uma solução completamente acoplada, quando possível, é computacionalmente

mais cara, tanto pelo mal condicionamento do sistema de equações resultante quanto pelo elevado número de graus de liberdade, da ordem de milhões em modelos tridimensionais.

A fim de evitar estes problemas, estratégias de solução alternativas têm sido desenvolvidas para tratar problemas de acoplamento hidromecânico. A principal caraterística destas abordagens é isolar os processos de fluxo e deformação, considerando soluções sequenciais em que aproxima-se o efeito que um exerce sobre o outro ao longo de uma simulação (Quevedo, 2012; Sheng *et al.*, 2003). Dentre as vantagens destas abordagens, destaca-se o uso de simuladores existentes, seja para modelagem de fluxo em reservatórios ou para geomecânica, que levam em consideração os métodos mais apropriados e inerentes a cada processo. Desta forma, em muitas situações, estas estratégias alternativas de acoplamento permitem obter resultados aproximados com menor esforço computacional e consequentemente, menores tempos de simulação.

Dentre as estratégias de acoplamento mais conhecidas destacam-se aquelas feitas através de duas vias, podendo ser iterativas ou explícitas. No caso do acoplamento iterativo, ambas equações governantes são solucionadas independentemente, mas para um mesmo intervalo de tempo, várias iterações com atualização dos termos de acoplamento são realizadas até que a solução atinja uma tolerância de erro aceitável (Settari e Mourits, 1998; Kim *et al.*, 2013). Por sua vez, o acoplamento explícito representa um acoplamento mais fraco que o anterior, já que apenas precisa-se de uma iteração. Em contraposição, dependendo da evolução de um processo em relação ao outro, esta estratégia pode se tornar muito atrativa no sentido que um processo pode ser solucionado em diferentes intervalos de tempo do outro, reduzindo substancialmente os tempos de simulação (Dean *et al.*, 2006; Fu *et al.*, 2013).

Uma outra abordagem relativamente mais simples para a solução de problemas hidromecânicos consiste no acoplamento de uma via. Nesta abordagem, as variações de pressão determinadas por um simulador de fluxo de reservatórios em diferentes intervalos de tempo são introduzidas como carregamentos em um simulador geomecânico. No entanto, diferentemente das estratégias de duas vias, as deformações e as alterações de tensão determinadas pelo simulador geomecânico não são consideradas pelo simulador de fluxo. Apesar da sua simplicidade, as estratégias de uma via têm produzido bons resultados, particularmente em reservatórios normalmente consolidados cujas propriedades hidráulicas não são significativamente afetadas pelas deformações (Marchina e Onaisi, 2005; De Gennaro *et al.*, 2010; HajNasser, 2012; De Sousa *et al.*, 1999). Esta abordagem também tem sido utilizada para aproveitar os resultados obtidos por simuladores comerciais de fluxo composicional em reservatórios, como o IMEX da Computer

Modelling Group (CMG) ou o ECLIPSE da Schlumberger, na modelagem do comportamento geomecânico de campos de óleo e gás. Este último, por exemplo, é baseado no método das diferenças finitas, cuja formulação garante a conservação da massa. A principal vantagem deste simulador é que pode ser utilizado com geometrias totalmente irregulares, como aquelas encontradas nos reservatórios de óleo e gás, incluindo descontinuidades como falhas geológicas, afinamentos ou estreitamentos progressivos de camadas em uma direção até sua extinção (*pinch-out*), e regiões ativas ou inativas para produção. Para isto, algumas células da grade do reservatório podem apresentar formas totalmente irregulares o que não representa dificuldade nenhuma na simulação de fluxo. Por outro lado, para as análises geomecânicas empregam-se simuladores baseados no método dos elementos finitos (MEF). Esta metodologia é bastante aceita na geomecânica pela sua capacidade de tratar com problemas altamente não lineares, como as relações constitutivas de tensão-deformação. Adicionalmente, o MEF permite a aplicação de condições inicias e de contorno com diversas aplicações na engenharia. Basicamente, o MEF consiste em discretizar o meio contínuo em um número discreto de elementos finitos os quais são analisados individualmente em um nível local. Estes elementos são conectados através de nós formando uma malha de elementos finitos. A contribuição de cada elemento é utilizada para construir um sistema global de equações o qual deve ser resolvido a fim de encontrar diretamente os correspondentes deslocamentos nodais, e indiretamente as deformações e tensões nos pontos de Gauss de cada elemento.

Contudo, a principal dificuldade dos simuladores geomecânicos é que o MEF requer que os elementos que pertencem à malha não sejam ser muito distorcidos, caso contrário, a solução local realizada em cada elemento não pode ser determinada. Por este motivo, em geral o grid de células de um modelo de fluxo de reservatórios não é diretamente utilizado como malha de elementos finitos do simulador geomecânico, demandando um tratamento para os elementos muito distorcidos. Outra dificuldade dos simuladores geomecânicos é que os modelos apresentam um domínio espacial consideravelmente maior daquele utilizado pelos simuladores de fluxo. Isto porque a malha de elementos finitos não deve considerar apenas o reservatório, mas também as camadas sobrejacentes, adjacentes e subjacentes. Isto demanda um outro tratamento para estender o domínio do reservatório em todas as direções, considerando inclusive a presença de camadas estratificadas fora do reservatório, aumentando significativamente o número de elementos das malhas geomecânicas. Alguns simuladores geomecânicos comerciais consideram estes tratamentos através de fases de pré-processamento de dados (Longuemare *et al.*, 2002; Monaco *et al.*, 2011). No entanto, métodos para geração de uma malha geomecânica adequada, no sentido de graus de

refinamento apropriados e elementos pouco distorcidos, têm sido pouco explorados. Recentemente, alguns trabalhos têm chamado a atenção para estes tratamentos na geração de malhas geomecânicas e proposto algumas alternativas (Florez *et al.*, 2014, 2016), mas este tema ainda se encontra em aberto para novas pesquisas.

4.2 Geração de malhas de hexaedros

Existe uma variedade de trabalhos que lidam com malhas com predominância de hexaedros, como (Ran *et al.*, 2012; Botella *et al.*, 2016). Nesta classe de problemas, a malha resultante possui, em sua maioria, elementos hexaédricos. No entanto, é permitida a existência de outros tipos de elementos.

Outra classe de problema lida com malhas que possuem apenas elementos hexaédricos. (Knupp, 2003) propõe um método para otimizar a forma dos elementos de uma malha de hexaedros de forma a garantir que não seja gerado nenhum elemento invertido. Para tal, cada hexaedro é divido em oito tetraedros associados com cada um dos oito nós do hexaedro, com as arestas coincidindo com as arestas do hexaedro. Dado um vetor $\mathbf{x}$ da origem a qualquer um dos oito nós e $\mathbf{x}_1$, $\mathbf{x}_2$ e $\mathbf{x}_3$ coordenadas de três vértices vizinhos escolhidos com uma orientação adequada, podem ser formadas três arestas $\mathbf{e}_1 = \mathbf{x}_1 - \mathbf{x}$, $\mathbf{e}_2 = \mathbf{x}_2 - \mathbf{x}$ e $\mathbf{e}_3 = \mathbf{x}_3 - \mathbf{x}$ e definida uma matriz $\mathbf{T}$ formada a partir de três vetores coluna $\mathbf{T} = [\mathbf{e}_1|\mathbf{e}_2|\mathbf{e}_3]$, ou seja, se as componentes de $\mathbf{x}$ são x, y, z então

$$T = \begin{pmatrix} x_1 - x & x_2 - x & x_3 - x \\ y_1 - y & y_2 - y & y_3 - y \\ z_1 - z & z_2 - z & z_3 - z \end{pmatrix} \tag{4-1}$$

Existem oito matrizes T_k para cada um dos vértices do hexaedro. Se $\alpha_k = \det(A_k) > 0$ então o elemento tem jacobiano positivo. Desta forma, a função objetivo proposta por (Knupp, 2003) minimiza o número de condição da matriz T_k para cada tetraedro, onde o número de condição é dado por

$$\kappa(T_k) = |T_k||T_k^{-1}|. \tag{4-2}$$

A norma da matriz em questão é a norma de Frobenius. A função de minimização global da malha utilizada é dada por

$$F = \frac{1}{N} \sum_n \sum_k \left(\frac{\kappa(T_{n,k})}{3} \right)^2 - 1 \tag{4-3}$$

sujeita a restrição de que $\alpha_{n,k} = \det(A_{n,k}) > 0 \, \forall n = \{0, ..., N-1\}$ e $k = \{0, ..., 7\}$, onde N é o número de elementos hexaédricos na malha.

Esta formulação garante que nenhum elemento com jacobiano negativo é gerado durante o processo de suavização da malha. No entanto, de acordo com

(Knupp, 2003) nenhuma tentativa foi realizada para provar a convexidade do problema de otimização. A solução é dada pelo primeiro mínimo local encontrado.

Apesar de garantir que todos os elementos da malha tenham sempre elementos válidos, (Knupp, 2003) reporta apenas resultados com malhas pequenas. No maior caso de teste reportado, o método leva 395 segundos para otimizar as formas do elementos de uma malha de 6749 nós.

4.3 Geração de malhas geomecânicas de reservatórios

O cálculo da compactação e subsidência de um reservatório normalmente envolve resolver um problema mecânico e de fluxo por uma técnica de acoplamento iterativo (Kim *et al.*, 2009). Como já existe uma malha de fluxo quando essas questões são levantadas, tem-se o problema de criar uma malha geomecânica compatível com essa. No entanto, o uso de uma malha que modela apenas a região do reservatório requer as condições de Neumann das EDPs que as são tensões na fronteira (Bell, 2003). Porém, estimar ou medir essas tensões é uma problema difícil. Uma outra abordagem mais comumente utilizada consiste em estender a malha em torno do reservatório de forma a que as condições de contorno sejam conhecidas. Esta última abordagem é mais tratável na prática, apesar do maior custo computacional (Florez *et al.*, 2011).

Existe um esforço nos últimos anos para gerar malhas de hexaedros e modelos adequados para simulações acopladas de fluxo e mecânica. (Gai, 2004; Dean *et al.*, 2006) embutiram a malha de fluxo em uma malha geomecânica estendida conforme e definiram os valores de porosidade e permeabilidade como zero fora da região do reservatório. As ferramentas Visage/Virage da Schlumberger seguem uma abordagem similar (Florez *et al.*, 2014). Nesta abordagem, a malha é estendida ao redor do reservatório com a restrição de que a malha na região do reservatório seja estritamente honrada. No entanto, para malhas de reservatório gerais, com falhas e auto-interseções, esta abordagem não pode ser realizada porque alguns elementos são degenerados e as falhas precisam ser suavizadas para evitar de ter que lidar com problemas de contato nas descontinuidades. A malha resultante deve ser *water tight*, ou seja, não devem existir lacunas entre os elementos e os vértices devem ser conectados aos seus vizinhos propriamente. Desta forma, as falhas e outras descontinuidades representam o maior desafio na geração destas malhas (Angus *et al.*, 2011).

O dado de entrada contém a geometria do reservatório e a topologia descrita por um conjunto de hexaedros. Cada hexaedro pode ser associado a um índice i, j e k, ou seja, a topologia é sempre a de uma grade. No entanto, a malha pode conter falhas em sua geometria. Estas falhas podem ser identificadas verificando

se a face em comum, do ponto de vista topológico, de células vizinhas na grade tem todos os vértices congruentes. Uma vez que ao menos um par de vértices da face em comum não seja congruente, uma falha é identificada. Além disso, existem algumas propriedades estáticas como porosidade, permeabilidade, entre outras. Uma informação muito importante é um *booleano* atribuído a cada célula que determina se a célula é ativa ou não. Em geral, as informações físicas do reservatório estão armazenadas nas células ativas, de forma que as células inativas existem apenas para tornar a topologia da malha do reservatório equivalente a uma grade. A Figura 4.1 ilustra o modelo Brugge (Peters *et al.*, 2013) na forma em que ele é dado como entrada. Note que as células inativas, em azul, não tem qualquer compromisso com a geometria do reservatório e portanto devem ter suas informações geométricas desconsideradas em qualquer algoritmo de reconstrução.

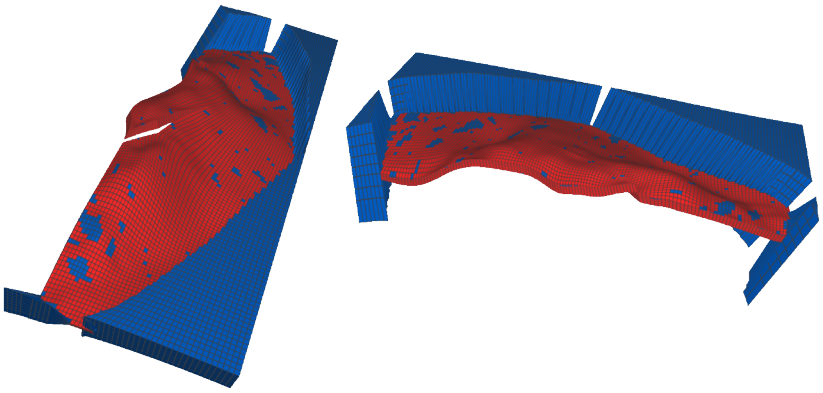

Figura 4.1: Modelo Brugge em dois diferentes ângulos. As células em vermelho são ativas enquanto as células em azul são células inativas. As células ativas, em geral, trazem as informações importantes do modelo. Os algoritmos de reconstrução devem ignorar as informações geométricas das células inativas.

4.3.1 Reconstrução por B-Splines cúbicas

(Florez *et al.*, 2014) propuseram uma técnica capaz de lidar com as descontinuidades nos modelos de reservatório. Nesta técnica eles reconstroem a malha na região do reservatório de forma contínua usando *patches* de B-Splines cúbicas e depois discretizam em uma malha de hexaedros. Para gerar a malha estendida em torno do reservatório a malha é projetada lateralmente e então uma interpolação linear transfinita é utilizada para gerar malhas abaixo e acima do reservatório. Como este trabalho aborda exatamente o tema desta tese, ele é descrito a seguir para efeito de comparação.

De forma simplificada, o algoritmo de reconstrução por B-Splines proposto por (Florez *et al.*, 2014) pode ser dividido nos três seguintes passos:

– criação de um polígono de controle que é projetado sobre o reservatório para capturar a sua geometria;
– interpolação do polígono de controle para gerar curvas contínuas que passem pelos pontos capturados no reservatório e posteriormente a criação de uma superfície usando *patches* cúbicos de B-Splines;
– extensão lateral da superfície pela projeção lateral das curvas da borda do reservatório e, por fim, uma extensão vertical do reservatório para povoar de células o espaço entre o reservatório e os horizontes de topo e base.

Estes três passos são descritos com mais detalhes a seguir.

Criação do polígono de controle

No primeiro passo, cria-se o polígono de controle que será interpolado depois de capturar a geometria do reservatório. Para isso, o reservatório de entrada é rotacionado e transladado de forma que seu eixo longitudinal seja paralelo ao eixo horizontal x. A caixa envolvente, alinhada com o reservatório, de controle fique restrito a ele, como ilustrado na Figura 4.2.

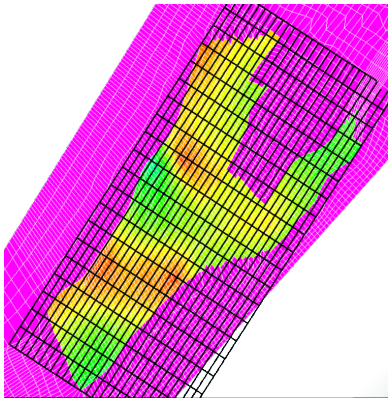

Figura 4.2: Polígono de controle restrito à região com células ativas. Imagem retirada de (Florez *et al.*, 2014).

O polígono pode ter espaçamento constante ou ter uma distribuição variável para capturar melhor alguma característica local do reservatório. O passo seguinte atrai o polígono para as células ativas, ou seja, os valores de z dos nós do polígono são definidos de acordo com o de z da célula onde o ponto é projetado (ver Figura 4.3). No entanto, isto pode levar a mudanças bruscas na curvatura da superfície resultante. Em alguns reservatórios isto ocorre quando mudanças geológicas locais levam a grandes mudanças nas características geométricas. Em geral, uma intervenção do usuário é necessária para ajustar o algoritmo para cada reservatório (Florez *et al.*, 2014), como impor um limite máximo no declive do polígono ao atraí-lo para as células ativas, como ilustrado na Figura 4.3.

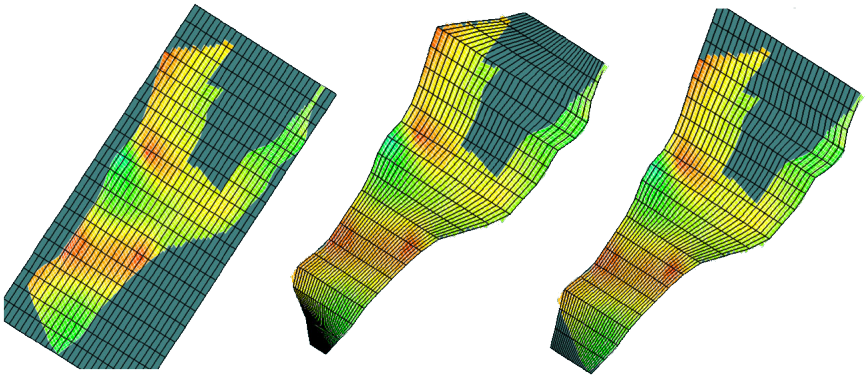

Figura 4.3: Geração do polígono de controle. À esquerda, o polígono restrito às células ativas. No centro, o polígono atraído pelas células ativas, apresentando grandes declives. À direita, o polígono atraído pelas células ativas com um limite imposto no declive para evitar grandes mudanças na curvatura. Imagem retirada de (Florez *et al.*, 2014).

A Figura 4.4 ilustra o caso de um polígono processado sobre o modelo Brugge. Este polígono resultante é aquele que será interpolado no próximo passo do algoritmo para gerar uma superfície na região do reservatório.

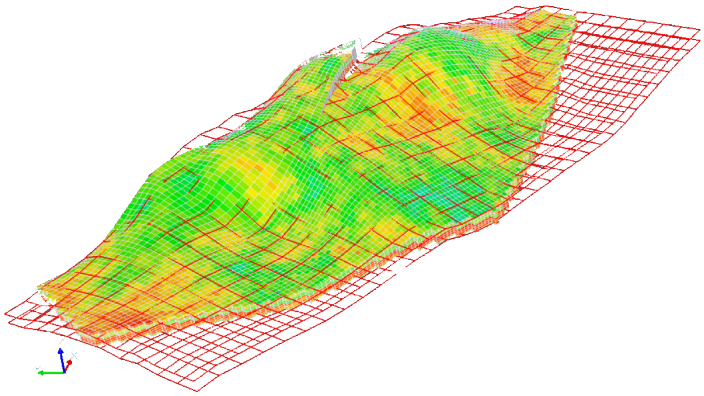

Figura 4.4: Polígono a ser interpolado para gerar as superfícies no modelo Brugge. Imagem retirada de (Florez *et al.*, 2014).

Criação da superfície B-Spline

Uma vez gerado o polígono de controle, seus pontos são ordenados e organizados em uma estrutura de dados, que em geral é uma matriz. Então, um conjunto de curvas B-Splines, nas direções u e v, são criadas. Estas curvas interpolam o polígono de controle e são chamadas de u-interpolantes e v-interpolantes, umas vez que são geradas fixando o valor de u e interpolando os pontos na direção de v e vice-versa. Por fim, as curvas u-interpolantes e v-interpolantes são usadas para obter os *patches* bicúbicos. Estes passos são ilustrados na Figura 4.5.

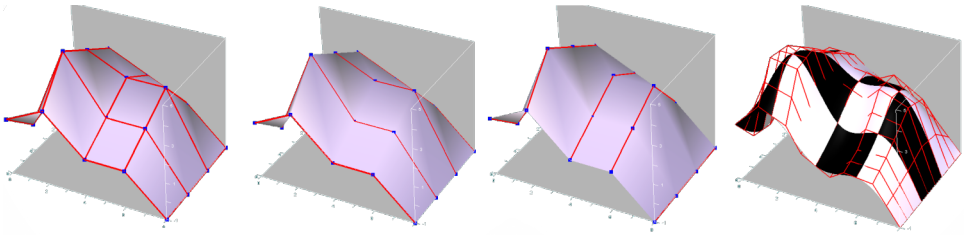

Figura 4.5: Polígono de controle à esquerda seguido pelas curvas u-interpolantes e v-interpolantes. À direita os *patches* bicúbicos resultantes. Imagem retirada de (Florez *et al.*, 2014).

Após este passo, o reservatório é representado por uma entidade geométrica analítica que o descreve. Desta forma, o modelo pode ser reconstruído com amostragens diferentes, se necessário. A Figura 4.6 ilustra a reconstrução do modelo Brugge.

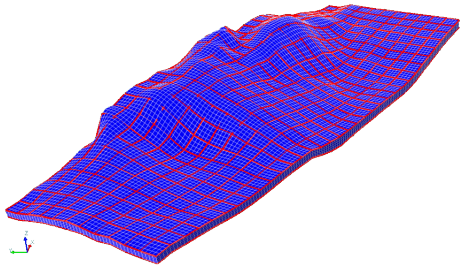

Figura 4.6: Superfície criada a partir do modelo Brugge. Imagem retirada de (Florez *et al.*, 2014).

Extensão do reservatório

Nesta etapa o reservatório é propagado lateralmente, uma vez que a malha precisa ser estendida em torno do reservatório. Para isso, as curvas na borda do reservatório (aquelas curvas com $u = 0$, $u = u_{\text{max}}$, $v = 0$ e $v = v_{\text{max}}$) são extrudadas. Isto é ilustrado na Figura 4.7.

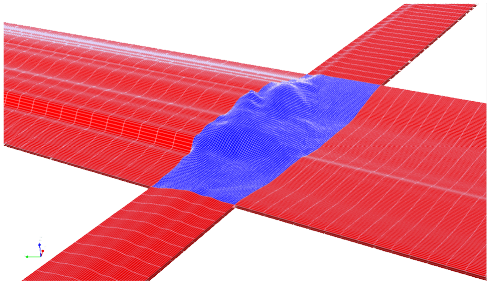

Figura 4.7: Cruvas estendidas lateralmente no reservatório. Imagem retirada de (Florez *et al.*, 2014).

As quatro regiões restantes nos cantos são calculadas utilizando superfícies translacionais (Piegl e Tiller, 1997). A ideia é literalmente transladar uma curva α

ao longo de outra curva β

$$s(u,v)^T = \alpha(u) + \beta(v) \tag{4-4}$$

Note que extrudar uma curva é equivalente a transladá-la ao longo de uma linha reta. Este processo é repetido para cada nível k do reservatório. Após este processo, uma superfície estruturada pode ser gerada para cada nível e então duas superfícies paralelas $\sigma^t(u,v)$ e $\sigma^b(u,v)$ podem ser interpoladas usando uma interpolação linear transfinita (Liseikin, 2009)

$$x_i(u,v,t) = (1-t)\sigma^t(u,v) + t\sigma^b(u,v). \tag{4-5}$$

onde $x_i(u,v,t)$ é o resultado da interpolação entre as duas superfícies. Esta interpolação criará camadas entre as duas superfícies para definir uma malha de hexaedros válida. Na Figura 4.8 uma malha de hexaedros é criada após a interpolação de duas superfícies.

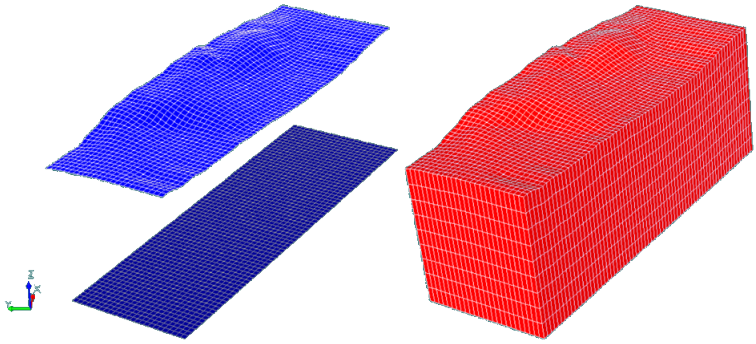

Figura 4.8: Malha de hexaedros criada a partir de uma interpolação transfinita. Imagem retirada de (Florez *et al.*, 2014).

Após todo o processo, um volume é gerado em torno do reservatório como ilustrado na Figura 4.9.

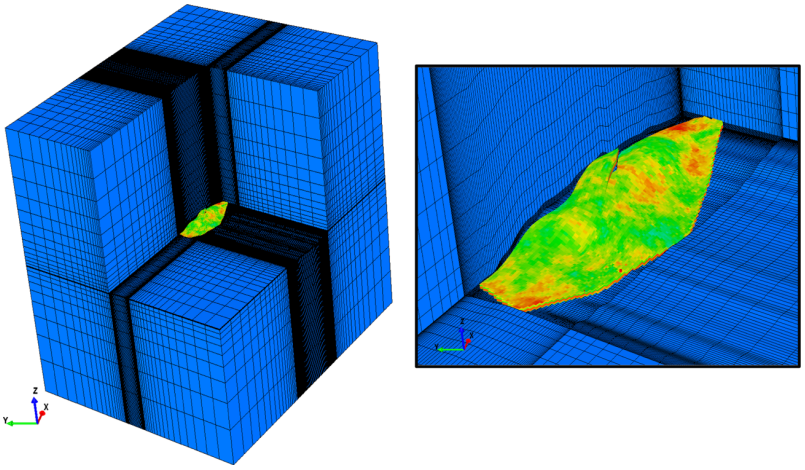

Figura 4.9: Malha de hexaedros gerada em torno do modelo Brugge. Imagem retirada de (Florez *et al.*, 2014).

4.3.2
Reconstrução por Elasticidade

Em um trabalho mais recente para geração de malhas geomecânicas de reservatórios, (Florez *et al.*, 2016) usam uma aborgadem baseada em elasticidade. Esta abordagem se limita a gerar a malha no entorno do reservatório. A malha de entrada do reservatório é assumida como sendo a malha já suavizada pelo métodos das B-Splines (Florez *et al.*, 2014). Uma outra malha, com um buraco no centro, é construída de forma que cada ponto do buraco tenha um ponto correspondente na fronteira do reservatório. Os deslocamentos destes pontos são medidos em relação ao reservatório e definidos para os pontos da malha externa. Valores diferentes de rigidez são definidos em regiões diferentes na malha externa, como por exemplo $E_{vermelho} = 0.5$ e $E_{azul} = 0.001$, e então um problema de elasticidade é resolvido até que se chegue ao equilíbrio, como ilustrado na Figura 4.10.

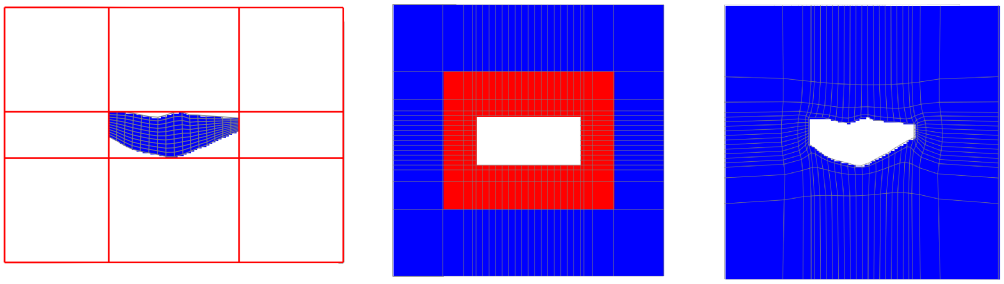

Figura 4.10: Malha suavizada do reservatório, à esquerda, é imersa em outra de referência, no centro. Após definidos os deslocamentos e os parâmetros para a matriz de rigidez elástica ($E_{vermelho} = 0.5$, $E_{azul} = 0.001$ e $v = 0.3$) um problema de elasticidade é resolvido, gerando a malha da direita. Imagem retirada de (Florez *et al.*, 2014).

Ao final, as duas malhas são conectadas gerando assim a malha em toda a região. Um resultado comum ao empregar este método é ilustrado na Figura 4.11.

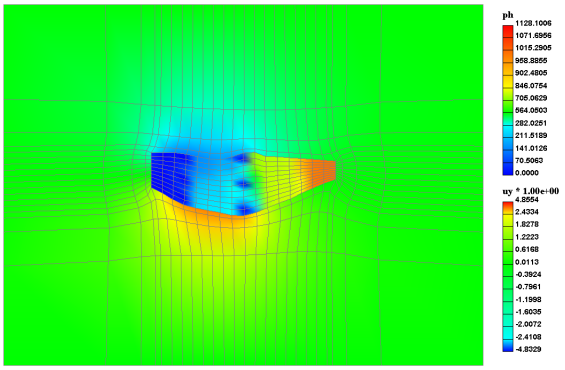

Figura 4.11: Resultado típico obtido ao empregar o método de geração de malhas geomecânicas por elasticidade. Imagem retirada de (Florez *et al.*, 2014).

De forma simplificada, pode-se dizer que (Florez *et al.*, 2016) propõe, um método para substituir os métodos de interpolação transfinita e superfícies translacionais empregadas no método anterior para gerar as malhas ao redor do reservatório.

5
Modelo de Otimização Quadrática para malhas de reservatórios

Este capítulo descreve a modelagem do problema de otimização quadrática para resolver o problema de geração de malhas geomecânicas de reservatórios, assim como os dados esperados na entrada do problema. A convexidade do problema resultante também é discutida, uma vez que esta é uma propriedade vital para que uma solução eficiente possa ser projetada.

5.1
Descrição do problema

O problema de geração de malhas geomecânicas de reservatórios consiste em construir uma malha de hexaedros que engloba a região do reservatório, o seu entorno e um conjunto de horizontes (subsuperfícies que separam camadas geológicas distintas) que pode existir acima e abaixo do reservatório, a partir das informações de uma malha de volumes finitos. A malha gerada não deve conter as falhas presentes na malha de volumes finitos. No entanto, as posições das células ativas devem ser respeitadas o máximo possível, uma vez que estas células carregam informações importantes do modelo. Uma vez minimizado o deslocamento destas células, são também minimizados os erros nas transferências dos atributos entre as malhas.

Uma vez que será utilizada para simulações, a malha gerada deve ser suave. Desta forma, o processo de eliminação das descontinuidades, juntamente com a restrição de manter as posições das células ativas sempre que possível, força a necessidade de suavizar o modelo para evitar mudanças muito bruscas nas regiões de falhas. Em geral é desejável que o grau de suavidade possa ser controlado pelo usuário. Desta forma, é possível ponderar a importância entre gerar uma malha suave e minimizar os deslocamentos das células ativas.

A Figura 5.1 ilustra uma entrada típica para o problema consistindo de um reservatório e múltiplos horizontes. O usuário deve fornecer as informações necessárias para estender o modelo, como distância e número de elementos na região estendida e número de camadas de elementos entre as superfícies. Desta

forma são conhecidas as subsuperfícies da malha resultante que devem respeitar os horizontes da entrada e as células que farão parte da condição de contorno.

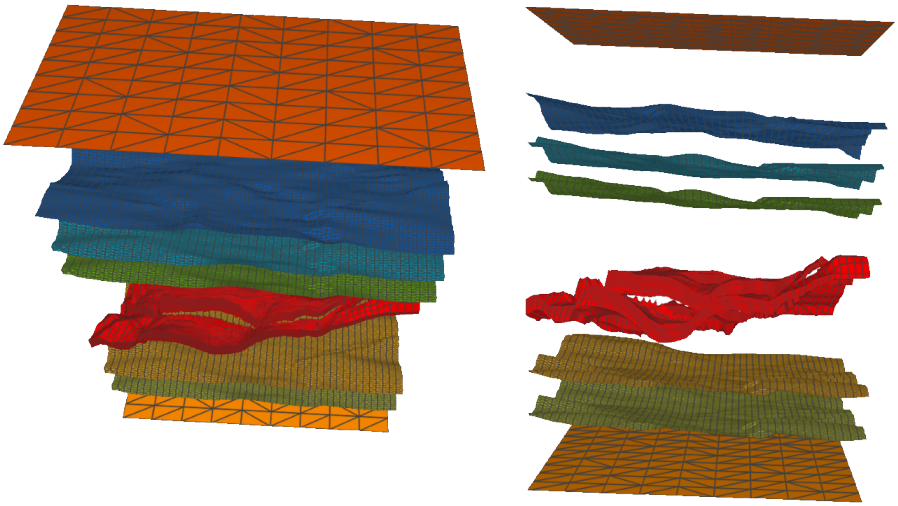

Figura 5.1: Modelo de entrada composto de um reservatório, em vermelho, e um conjunto de horizontes vistos por dois pontos de vista. O reservatório em questão é o modelo real Norne (`https://opm-project.org/?page_id=559`).

5.2 Modelo de Otimização

O processo de geração de malhas geomecânicas de reservatórios precisa realizar duas tarefas: eliminar as descontinuidades do modelo e, por consequência, suavizar a malha resultante para evitar variações muito bruscas e/ou elementos degenerados ou impróprios, como elementos com o jacobiano negativo. É desejável que o grau de suavização do modelo possa ser controlado pelo usuário. Além disso, ao eliminar as descontinuidades, é desejável manter as células ativas nas mesmas posições sempre que possível, uma vez que estas células carregam as informações importantes do reservatório. O modelo de otimização proposto na Seção 5.2.3 busca realizar as duas tarefas ao mesmo tempo.

5.2.1 Eliminação das descontinuidades do modelo

Em geral o modelo de entrada apresenta descontinuidades em sua geometria, como ilustrado na Figura 5.2. Estas descontinuidades precisam ser eliminadas com a restrição de conservar, sempre que possível, a posição dos vértices das células ativas. A seguir são descritas duas abordagens para eliminar tais descontinuidades, uma no domínio do modelo de entrada e outra no domínio do modelo de saída.

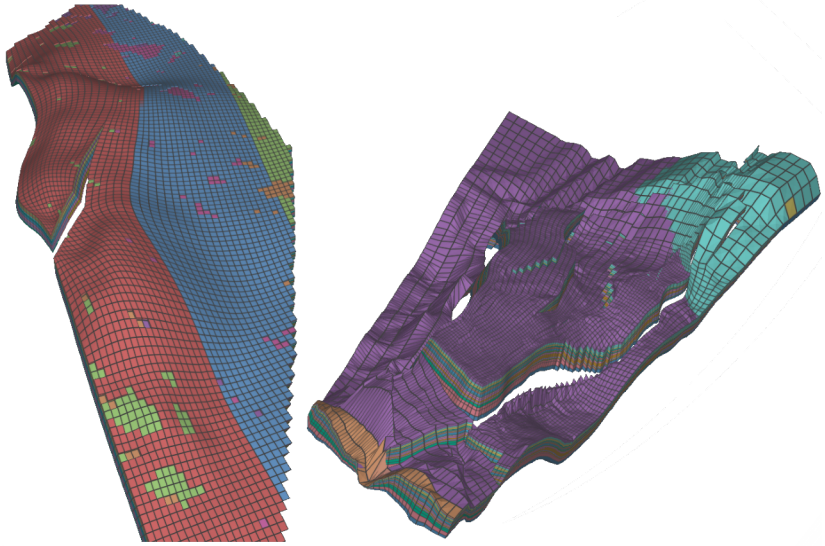

Figura 5.2: Exemplos de descontinuidades nos modelos de entrada. À esquerda o modelo Brugge e à direita o modelo Norne.

Modelagem sobre o domínio de entrada

No domínio da entrada existem elementos que são vizinhos topológicos, uma vez que a topologia do reservatório é sempre uma grade, porém com uma descontinuidade entre eles. Estes elementos possuem vértices correspondentes. Estes vértices são aqueles, que ao eliminar as descontinuidades, terão as mesmas posições. Como descrito no Capítulo 4, estas descontinuidades podem ser identificadas facilmente usando o fato de que a malha de entrada é sempre organizada topologicamente como uma grade. Desta forma, basta verificar se a face comum entre duas células que são vizinhas topologicamente tem os vértices congruentes em ambas as células. A Figura 5.3 ilustra, em duas dimensões, dois elementos vizinhos separados por uma descontinuidade.

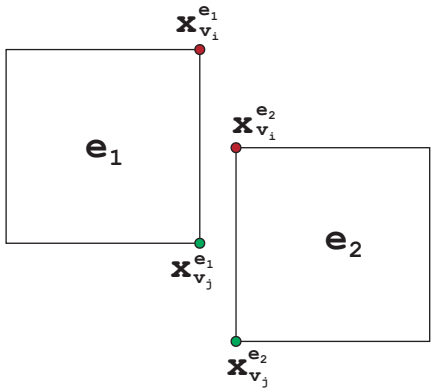

Figura 5.3: Dois elementos e_1 e e_2, em duas dimensões, vizinhos topológicos com uma descontinuidade entre eles. Os vértices $\mathbf{x}_{v_i}^{e_1}$ e $\mathbf{x}_{v_i}^{e_2}$ são vértices equivalentes, assim como os vértices $\mathbf{x}_{v_j}^{e_1}$ e $\mathbf{x}_{v_j}^{e_2}$.

Para gerar uma malha sem descontinuidades a partir da malha de entrada, pode-se criar variáveis para representar vértices correspondentes e definir restrições para garantir que estes tenham a mesma posição ao final do processo. Desta forma, para cada par de vértices v_i correspondentes que estejam em elementos e_1 e e_2 distintos e vizinhos, porém separados por uma descontinuidade, a seguinte restrição de igualdade deve ser adicionada ao problema de otimização

$$\mathbf{x}_{v_i}^{e_1} - \mathbf{x}_{v_i}^{e_2} = 0. \tag{5-1}$$

onde $\mathbf{x}_{v_i}$ é um vetor com uma variável para cada dimensão do vértice v_i.

Como o deslocamento das células ativas deve ser minimizado, uma restrição de penalização por deslocamento deve ser adicionada para cada vértice que esteja em uma célula ativa, ou seja, supondo que $\mathbf{x}_{v_i}^{e_1}$ seja o vetor de variáveis que representa um vértice de uma célula ativa e que $\mathbf{p}_{v_i}^{e_1}$ seja sua posição no modelo de entrada, a seguinte restrição *soft*

$$\mathbf{x}_{v_i}^{e_1} - \mathbf{p}_{v_i}^{e_1} \approx 0 \tag{5-2}$$

deve ser incluída no problema de otimização. Estas restrições são chamadas de *soft* porque elas não precisam ser estritamente honradas e devem ser garantidas no sentido de mínimos quadrados, por isso, estas restrições são incluídas na função objetivo como um termo da forma

$$k(\mathbf{x}_{v_i}^{e_1} - \mathbf{p}_{v_i}^{e_1})^2 \tag{5-3}$$

onde $k \geq 0$ é um fator de penalização. Quanto maior o valor de k, maior será o custo de desviar um vértice de sua posição original.

Em um cenário ideal, as restrições *soft* seriam restrições de igualdade e desta forma nenhum vértice em células ativas seria movido. No entanto, uma vez utilizada uma restrição de igualdade em uma descontinuidade onde e_1 e e_2 são elementos que representam células ativas, o problema de otimização resultante seria inviável, uma vez que a mesma variável deveria assumir dois valores distintos, o que contraria a restrição 5-1. A Figura 5.4 ilustra uma situação onde há células ativas em ambos os lados de uma descontinuidade.

A Figura 5.5 ilustra um exemplo, em duas dimensões, onde a descontinuidade entre dois elementos é eliminada utilizando as restrições discutidas acima.

Apesar desta abordagem eliminar as descontinuidades do modelo de entrada, a criação de múltiplas variáveis para representar um mesmo vértice aumenta o custo computacional para a solução do problema de otimização resultante. Uma vez que um vértice interno de uma grade se conecta a outros seis vértices, no pior caso pode acontecer de serem necessários seis vetores de variáveis para o mesmo vértice,

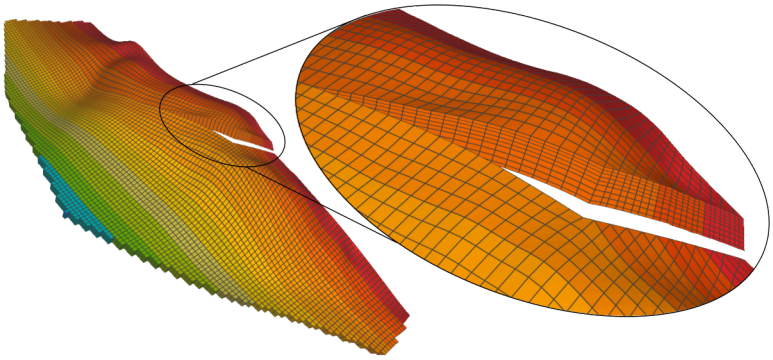

Figura 5.4: Células ativas do modelo Brugge. Note que existem células ativas e vizinhas em lados opostos de uma descontinuidade. Nestes casos obrigatoriamente as células ativas devem mudar de posição para eliminar a descontinuidade.

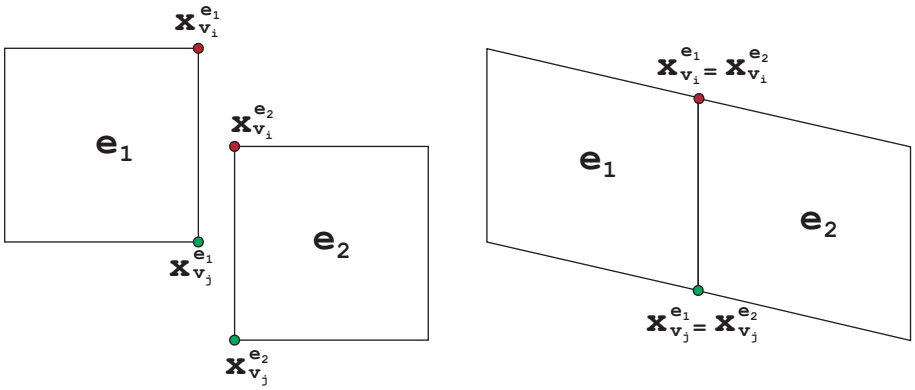

Figura 5.5: À esquerda dois elementos vizinhos representando células ativas com uma descontinuidade entre eles. À direita uma solução para as restrições de eliminação de descontinuidades.

além das respectivas restrições. As desvantagens dessa abordagem levam à segunda abordagem, mudando o domínio onde o problema é modelado.

Modelagem sobre o domínio de saída

O modelo de saída é representado por uma malha estruturada em forma de uma grade, sem qualquer descontinuidade. As características topológicas da malha resultante são conhecidas, uma vez que as incógnitas do problema se resumem à geometria dos vértices. Logo, este fato pode ser explorado para modelar o problema de otimização sobre o domínio de saída, em vez de seguir o caminho natural de modelar sobre a malha de entrada.

Nesta abordagem, as variáveis $\mathbf{x}_{v_i}$ são associadas aos vértices da malha de saída, desta forma sempre existirá um único vetor de variáveis, com uma variável por dimensão, para cada vértice. Isto elimina o custo computacional extra gerado pela abordagem anterior ao criar múltiplas variáveis para um mesmo vértice. Isto é ilustrado na Figura 5.6.

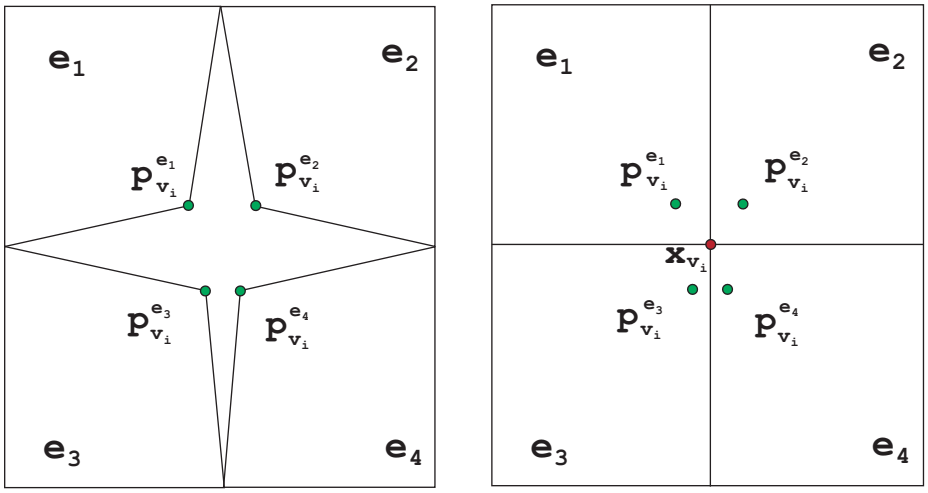

Figura 5.6: Exemplo de modelagem do problema de otimização sobre o modelo da saída. À esquerda o modelo de entrada e à direita o de saída. Um único vetor de variáveis $\mathbf{x}_{v_i}$ é criado para cada vértice. A posição do vértice é determinada pelas penalizações de deslocamento dos vértices em células ativas.

Como a malha de saída é sempre conforme, esta abordagem também elimina a necessidade de adicionar restrições do tipo 5-1 para eliminar descontinuidades. Desta forma restam apenas as restrições *sotf* do tipo 5-2 para penalizar os deslocamentos dos vértices que pertencem à células ativas.

Em resumo, ao modelar o problema de otimização para eliminar descontinuidades no domínio de saída, basta adicionar as restrições *soft* para penalizar o deslocamento de vértices que estão em células ativas. Estas penalizações vão determinar a nova posição do vértice. Se todos os elementos da Figura 5.6 representassem células ativas, bastaria atribuir as variáveis $\mathbf{x}_{v_i}$ para o vértice central e adicionar as seguintes restrições

$$\begin{aligned} \mathbf{x}_{v_i} - \mathbf{p}_{v_i}^{e_1} &\approx 0 \\ \mathbf{x}_{v_i} - \mathbf{p}_{v_i}^{e_2} &\approx 0 \\ \mathbf{x}_{v_i} - \mathbf{p}_{v_i}^{e_3} &\approx 0 \\ \mathbf{x}_{v_i} - \mathbf{p}_{v_i}^{e_4} &\approx 0 \end{aligned} \tag{5-4}$$

para determinar sua posição final.

Note que, como as restrições *soft* integram a função objetivo, e as restrições do tipo 5-1 não são mais necessárias, uma consequência imediata desta abordagem é que o problema de otimização que elimina as descontinuidades do modelo é irrestrito.

Dadas todas as vantagens sobre a opção de eliminar as descontinuidades sobre o domínio de entrada, a modelagem do problema de otimização sobre o domínio de saída é a mais adequada por ser mais simples e eficiente.

5.2.2 Suavização do Modelo

Eliminadas as descontinuidades, o próximo passo é suavizar o modelo resultante para tentar evitar elementos impróprios nas regiões onde foram eliminadas as descontinuidades e buscar transições mais suaves na malha resultante. Como discutido no Capítulo 3, uma forma natural de fazer isso é usando o operador de suavização laplaciano. Como é um problema de reconstrução, uma vez que a geometria das células inativas não podem ser usadas como referência, o uso dos operadores combinatórios é o mais indicado. Além disso, estes operadores são naturalmente estendidos para diferentes tipos de malha, o que não acontece com a fórmula da cotangente, por exemplo.

Do ponto de vista de otimização, tanto o operador Kirchhoff quanto o operador uniforme posicionam o vértice central no centroide dos seus vizinhos, quando minimizados. Desta forma, as duas opções seriam equivalentes, no entanto, para este trabalho será escolhido o operador de Kirchhoff pela relação dos seus pesos com a valência dos vértices centrais. Nas referências seguintes no texto, este operador será referido como operador laplaciano, ou simplesmente laplaciano.

Os pesos do operador laplaciano podem ser definidos como uma máscara em um vértice v_i sobre seus vizinhos $v_j \in N_1(v_i)$, como a seguir:

$$w_m(v_i) = \begin{cases} -|N_1(v_i)| & \text{se } v_i = v_j \\ 1 & \text{se } v_i \neq v_j \text{ e } v_j \in N_1(v_i) \end{cases} \tag{5-5}$$

onde $N_1(v_i)$ é o conjunto de vértices na estrela de v_i e $|N_1(v_i)|$ é a cardinalidade deste conjunto, ou seja, a valência do vértice v_i, como ilustrado na Figura 5.7.

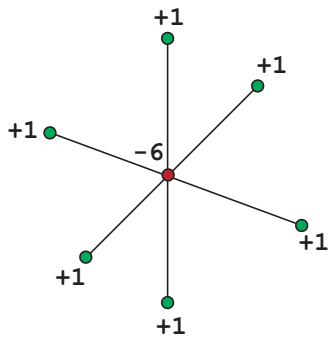

Figura 5.7: Estrela de um vértice na máscara do operador laplaciano com seus respectivos pesos.

Essa máscara é comumente utilizada como um método de suavização sem restrição, ou seja, ela é aplicada sobre a malha repetidas vezes até que os vértices da malha convirjam para uma posição fixa.

Como definido na Equação 3-13, uma superfície é suavizada quando possui o valor do laplaciano zero em seus vértices. Por isso, pode-se definir um problema de otimização no qual a função objetivo minimiza o quadrado do módulo do valor do operador laplaciano para cada vértice, ou seja:

$$\mathbf{x}^* = \min \sum_{v_i} ||\mathbf{L}(v_i)||^2 \tag{5-6}$$

onde $\mathbf{x}^*$ é a solução da minimização e $\mathbf{L}(v_i) = \sum_{v_j \in N_1(v_i)} \left(\mathbf{x}_{v_j} - \mathbf{x}_{v_i}\right)$.

5.2.3 Problema de otimização quadrática

A eliminação das descontinuidades e a suavização do modelo podem ser realizadas ao mesmo tempo por um problema de otimização quadrática. Suponha que V seja o conjunto de todos os vértices da malha e $V_a \subset V$ o conjunto dos vértices que pertencem à células ativas, então o problema de otimização pode ser escrito como

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathbf{L}(V) + k\mathbf{P}(V_a) \\ \text{sujeito a} \quad & \mathbf{x}_f = \mathbf{p}_f \end{aligned} \tag{5-7}$$

onde $k \geq 0$ é um fator de penalização,

$$\mathbf{L}(V) = \sum_{v_i \in V} ||\mathbf{L}(v_i)||^2 \tag{5-8}$$

e

$$\mathbf{P}(V_a) = \sum_{v_i \in V_a} \sum_{e} (\mathbf{x}_{v_i} - \mathbf{p}^e_{v_i})^2 \tag{5-9}$$

são as penalizações para os vértices das células ativas da forma 5-3 discutidas na Seção 5.2.1. As restrições de igualdade fixam vértices que tem posições conhecidas, como aqueles que definem as condições de contorno necessárias para o operador laplaciano, os vértices das subsuperfícies da malha que serão projetados nos horizontes ou vértices de células ativas em regiões sem descontinuidade. Neste último caso, fica a cargo do usuário definir se todos os vértices da malha tem liberdade para serem posicionados pelo problema de otimização. Caso contrário, o usuário deve fixar os vértices que desejar.

As condições de contorno são definidas fixando os vértices das células que pertencem ao borda malha estendida em torno do reservatório. Em geral, a distribuição das células pode ser feita regularmente. No entanto, melhores resultados são alcançados se as condições de contorno acompanharem o comportamento das células do bordo do reservatório, ou seja, as posições no bordo da malha de entrada do reservatório são projetadas na no bordo da malha estendida.

Para garantir que a malha gerada honre os horizontes do dado de entrada, em tempo de pré-processamento a subsuperfície da malha que deve ser mapeada em um determinado horizonte é projeta neste horizonte e tem seus vértices fixados nestas posições através das restrições de igualdade.

O fator de penalização k é uma forma do usuário definir um fator de suavização da malha. Quanto maior for o valor de k, mais importante será honrar as posições dos vértices nas células ativas e menos suavizada será a malha nas regiões de descontinuidade. Ao usar um valor de k pequeno, o usuário prioriza a suavização do modelo e permite que os vértices que integram as células ativas tenham mais liberdade para se moverem.

Uma vez que $\mathbf{L}(v_i) = \sum_{v_j \in N_1(v_i)} \left(\mathbf{x}_{v_j} - \mathbf{x}_{v_i}\right)$, com $\mathbf{x}_{v_i}$ e $\mathbf{x}_{v_j}$ vetores de variáveis, com uma variável para cada dimensão, $\mathbf{L}(v_i)$ pode ser definido como um vetor n-dimensional dado por $\mathbf{L}(v_i) = (y_{i0}, y_{i1}, ..., y_{id})$. Usando esta nova representação de $\mathbf{L}(v_i)$ a minimização pode ser reescrita como

$$\min \sum_{v_i \in V} ||\mathbf{L}(v_i)||^2 = \min \sum_{v_i \in V} \mathbf{L}(v_i) \cdot \mathbf{L}(v_i) = \min \sum_i \sum_j^d y_{ij}^2 \qquad (5\text{-}10)$$

Desta forma, o termo da função objetivo que minimiza o módulo do laplaciano para cada vértice da malha é uma soma de termos quadráticos das dimensões. Como é uma soma de termos positivos e não existem produtos entre variáveis que representam dimensões distintas, a otimização do operador laplaciano pode ser feita de forma desacoplada, ou seja, cada dimensão pode ser minimizada separadamente.

Note que o termo da penalização $\mathbf{P}(V_a)$ que integra a função objetivo tem propriedade semelhante. Cada termo de penalização é da forma $(\mathbf{x}_{v_i} - \mathbf{p}_{v_i}^e)^2$. Logo, ao minimizar este termo, o módulo do vetor que liga a posição corrente dada pelas variáveis $\mathbf{x}_{v_i}$ ao ponto $\mathbf{p}_{v_i}^e$ é minimizado. Como no caso do laplaciano, este termo pode ser reescrito como a soma de termos quadráticos por dimensão. Uma forma mais simples de ver isso é perceber que a restrição *soft* para penalização pode ser definida em um vértice para cada uma de suas dimensões de forma separada. Logo, o termo das penalizações também pode ser minimizados por dimensão. Desta forma, todo o problema é desacoplado entre suas dimensões.

Em resumo, quando modelado sobre o domínio da malha de saída, o problema de otimização quadrática 5-7 elimina as descontinuidades e suaviza a malha ao mesmo tempo. Um parâmetro de entrada k modula a importância entre suavizar a malha e conservar, o quanto possível, a posição das células ativas. Além disso, as únicas restrições do problema de otimização em questão são restrições triviais para fixar vértices em posições conhecidas e a solução do problema pode ser feita de forma independente para cada uma das dimensões da malha.

5.3 Convexidade do Modelo de Otimização Proposto

Um problema de otimização quadrática é convexo se sua função objetivo é convexa, já que todas as suas restrições são lineares.

Uma vez que a soma ponderada por pesos não negativos de funções convexas é uma função convexa (Seção 2.2.1), podemos analisar a convexidade da função objetivo do problema de otimização 5-7 por partes, ou seja, uma vez provadas as convexidades de $\mathbf{P}(V_a)$ e $\mathbf{L}(V)$, a função objetivo do problema de otimização proposto é convexa.

5.3.1 Convexidade de $\mathbf{P}(V_a)$

O termo $\mathbf{P}(V_a)$ na função objetivo é definido como

$$\mathbf{P}(V_a) = \sum_{v_i \in V_a} \sum_{e} (\mathbf{x}_{v_i} - \mathbf{p}^e_{v_i})^2 \tag{5-11}$$

ou seja, para cada vértice v_i da malha de saída, é somada uma penalização para cada posição $\mathbf{p}^e_{v_i}$ dos vértices correspondentes a v_i em células ativas e. Como discutido na seção anterior, este termo pode ser reescrito como uma soma de termos quadráticos por dimensão da seguinte forma

$$P(v^d_i) = (x_d - p_d)^2 \tag{5-12}$$

onde d é uma das dimensões do vértice v_i, x_d é a variável corresponde e p_d é o valor o qual o problema de otimização será penalizado se x_d desviar dele. Como este é um termo quadrático de uma única variável, a sua segunda derivada é estritamente positiva, logo esta função é estritamente convexa (Seções 2.2.4 e 2.5).

Uma vez que $\mathbf{P}(V_a)$ pode ser escrita como uma soma de termos da forma dada em 5-12, então $\mathbf{P}(V_a)$ é estritamente convexa, uma vez que a soma de funções convexas é uma função convexa, como descrito na Seção 2.2.1.

5.3.2 Convexidade de $\mathbf{L}(V)$

O termo $\mathbf{L}(V)$ da função objetivo pode ser definido como

$$\mathbf{L}(V) = \sum_{v_i \in V} \mathbf{L}(v_i)^2 = \sum_{v_i \in V} \sum_{v_j \in N_1(v_i)} \left(\mathbf{x}_{v_j} - \mathbf{x}_{v_i}\right)^2 \tag{5-13}$$

Note que $\mathbf{L}(v_i)$ é uma função afim, logo cada termo quadrático de $\mathbf{L}(V)$ é uma composição de uma função afim com a função estritamente convexa $f(x) = x^2$. Como discutido na Seção 2.2.2, a composição de uma função afim com uma função convexa é também uma função convexa, logo cada termo de $\mathbf{L}(V)$ é também

convexo. Recorrendo mais uma vez ao fato de que que a soma de funções convexas é uma função convexa, conclui-se que $\mathbf{L}(V)$ é uma função convexa. Porém, exitem infinitas configurações que zeram o termo $\mathbf{L}(v_i)$, ou seja, apesar de $\mathbf{L}(v_i)$ ser convexo, a função não é estritamente convexa, uma vez que a solução não é única. Uma forma de gerar mais de uma solução ótima para $\mathbf{L}(v_i)$ é posicionar os vértices na estrela de v_i igualmente espaçados sobre um círculo e variar o raio do círculo. Desta forma, todas as soluções geradas seriam ótimas, uma vez que o vértice central sempre está no ponto médio dos seus vizinhos, como ilustrado na Figura 5.8.

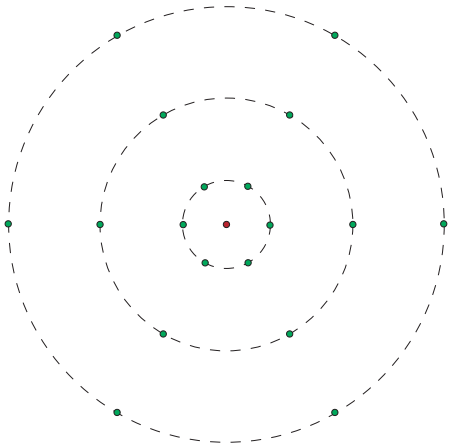

Figura 5.8: Distintas configurações de vértices (em verde) com o mesmo vértice central. Todas as soluções zeram o valor do laplaciano para o vértice central.

Uma vez que a minimização do operador laplaciano implica na minimização da energia de Dirichlet e esta tem solução única se definidas condições de contorno adequadas, uma forma de garantir que a minimização do operador laplaciano tenha solução única é fixar os vértices da fronteira da malha. Estes vértices são fixados usando as restrições de igualdade no problema de otimização 5-7. Note que ao fixar os vértices na estrela de um vértice v_i, a minimização do operador laplaciano tem solução única. Logo, ao fixar a borda de uma malha, garante-se que $\mathbf{L}(v_i)$ tem solução única, e uma vez que a composição é feita com uma função estritamente convexa, então a função resultante é estritamente convexa.

Como $\mathbf{L}(V)$ e $\mathbf{P}(V_a)$ são funções estritamente convexas e $k \geq 0$, a função objetivo do problema de otimização 5-7 é estritamente convexa e portanto possui solução única.

6
Solução do Modelo de Otimização

Neste capítulo será descrita a solução do problema de otimização proposto no capítulo 5. O problema de otimização 5-7 é irrestrito, uma vez que as restrições presentes podem ser eliminadas ao substituir as variáveis envolvidas pelos valores atribuídos nas restrições. Desta forma, é necessário minimizar um problema da forma

$$\min_{\mathbf{x}} \quad f_0 = \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{q}^T\mathbf{x} + c \tag{6-1}$$

Como visto na Seção 2.5, uma vez que f_0 é estritamente convexa, uma condição necessária e suficiente para que $\mathbf{x}$ minimize f_0 é

$$\nabla f_0(\mathbf{x}) = \mathbf{Q}\mathbf{x} + \mathbf{q} = 0 \tag{6-2}$$

o que leva ao seguinte sistema de equações lineares

$$\mathbf{Q}\mathbf{x} = -\mathbf{q}. \tag{6-3}$$

onde $\mathbf{Q}$ é uma matriz quadrada, simétrica e positiva definida, uma vez que o problema a ser minimizado é estritamente convexo. Além disso, como o operador laplaciano é um operador local, a matriz $\mathbf{Q}$ é também esparsa.

Este capítulo apresenta duas soluções para o problema de otimização, ambas resolvem o sistema de equações 6-3. No entanto, a primeira solução utiliza a matriz $\mathbf{Q}$ alocada em memória e a segunda abordagem reconstrói apenas as linhas de $\mathbf{Q}$ à medida que elas são necessárias, portanto a matriz $\mathbf{Q}$ não é armazenada em memória. Desta forma, as duas soluções são semelhantes com uma diferença no montante de memória necessário.

6.1
Método dos Gradientes Conjugados

Se $\mathbf{Q}$ fosse uma matriz densa, a melhor forma de resolver o sistema 6-3 provavelmente seria fatorar a matriz $\mathbf{Q}$ e resolver o sistema por retro substituição. O tempo que leva para fatorar uma matriz densa é próximo do tempo que leva para resolver um sistema de equações lineares iterativamente, e uma vez fatorada a matriz $\mathbf{Q}$, o sistema pode ser resolvido rapidamente para diferentes valores de

q (Shewchuk, 1994). Este seria o caso do problema em questão, uma vez que a função objetivo, de onde é derivada a matriz **Q**, é definida com base nas conexões dos vértices, que por sua vez é igual para todas as dimensões [1]. A matriz triangular resultante da fatoração de uma matriz esparsa tem, em geral, uma quantidade muito maior de elementos não nulos do que a própria matriz. Esta fatoração pode não ser possível devido a limitações de memória e tempo de processamento para matrizes muito grandes, o que pode tornar até mesmo o passo de retro substituição mais lento do que uma solução iterativa (Shewchuk, 1994). Em contrapartida, a maioria dos métodos iterativos são eficientes, quanto ao consumo de memória e tempo de processamento, para matrizes esparsas.

Um dos métodos iterativos mais populares para resolver grandes sistemas de equações lineares é o *método dos gradientes conjugados*. Este método foi desenvolvido para matrizes com as características da matriz **Q**, ou seja, quadrada, simétrica e positiva definida. Além disso, a despeito de problemas relacionados à precisão numérica, o método converge em no máximo n iterações (Shewchuk, 1994; Press *et al.*, 2007), onde n é o número de variáveis do problema. No entanto, em geral o método converge com menos que n iterações para uma dada tolerância.

Uma forma de acelerar a convergência de métodos iterativos é pelo uso de precondicionadores. Neste caso, uma transformação linear é aplicada ao sistema para melhorar suas propriedades espectrais (Press *et al.*, 2007). O *raio espectral* é definido como o valor do maior autovalor da matriz **Q**. Este valor determina a velocidade da convergência, de forma que quanto maior o raio espectral, mais lenta é a convergência. Na prática, ao aplicar um precondicionador **M**, o sistema original é substituído por um sistema equivalente mais simples de resolver

$$\mathbf{MQx} = -\mathbf{Mq}. \tag{6-4}$$

Como o objetivo é aumentar a eficiência na solução do sistema de equações, o precondicionador deve ser simples e eficiente de computar. Um dos precondicionadores mais simples e eficientes de computar é o *precondicionador de Jacobi*. Este é dado por $\mathbf{M} = \mathbf{D}^{-1}$, onde **D** é a matriz construída usando os elementos da diagonal de **Q**. Apesar da simplicidade, este precondicionador apresenta um ótimo ganho de performance a um custo baixo, por isso as soluções apresentadas a seguir são baseadas no método dos gradientes conjugados com precondicionador de Jacobi.

O Algoritmo 1 apresenta o pseudo-código do método dos gradientes conjugados com precondicionador. A implementação da biblioteca LIS (Nishida *et al.*, 2009), utilizada para resolver o sistema de equações neste trabalho, é idêntica ao do Algoritmo 1. Neste pseudo-código k representa a iteração corrente e r é o

[1] Isso não é verdade se algumas dimensões de um determinado vértice forem fixadas e outras não.

vetor residual. Note que o residual mede o valor de $-\nabla f_0(\mathbf{x})$, logo a condição de otimalidade é alcançada quando a norma do residual for menor que uma tolerância dada. Uma análise mais aprofundada sobre o método dos gradientes conjugados com uso de precondicionadores pode ser encontrada em (Press *et al.*, 2007). O pseudo código foi inserido aqui porque será necessário para análise na sequência do capítulo.

***Algoritmo* 1** Método dos gradientes conjugados com precondicionador

```
1  ****************************************************
2  * Preconditioned Conjugate Gradient                *
3  ****************************************************
4  r(0)    = -(q + Qx(0))
5  rho(-1) = 1
6  p(0)    = (0,...,0)^T
7  ****************************************************
8  for k=1,2,...
9    z(k-1)    = M^-1 * r(k-1)
10   rho(k-1)  = <r(k-1),z(k-1)>
11   beta      = rho(k-1) / rho(k-2)
12   p(k)      = z(k-1) + beta*p(k-1)
13   y(k)      = Q * p(k)
14   dot_py    = <p(k),y(k)>
15   alpha     = rho(k-1) / dot_py
16   x(k)      = x(k-1) + alpha * p(k)
17   r(k)      = r(k-1) - alpha * y(k)
18 ****************************************************
```

6.2 Montagem do sistema de equações

Nesta seção será descrito o método para construir o sistema de equações, ou seja, a matriz **Q** e o vetor **q** a partir da malha de entrada. Como visto no Capítulo 5, a função objetivo é composta por duas funções, uma para suavização da malha e outra para penalização dos deslocamentos das células ativas. A seguir será mostrado como cada uma destas funções contribuem para a montagem da matriz **Q** e do vetor **q**, para isso será considerado que o problema será resolvido para cada dimensão separadamente.

6.2.1 Função de Penalização

Como visto no Capítulo 5, cada restrição *soft* para penalização dos deslocamentos dos vértices das células ativas adiciona na função objetivo um termo, por dimensão, da forma

$$f(x_d) = k(x_d - p_d)^2 = kx_d^2 - 2kp_dx_d + kp_d^2 \tag{6-5}$$

onde k é o fator de penalização, x_d é a variável que representa a d-ésima dimensão do vértice e p_d é um valor escalar de entrada o qual o problema de otimização será penalizado caso o valor de x_d se desvie de p_d. Como a matriz **Q** é a matriz *Hessiana* da função, ela é composta pelos valores das segundas derivadas parciais. A única segunda derivada parcial não nula do termo acima é

$$\frac{\partial^2 f(x_d)}{\partial x_d^2} = 2k \tag{6-6}$$

logo, este termo é somado na diagonal da matriz na posição referente à variável x_d.

Já o vetor **q** é composto pelos escalares que multiplicam os termos lineares de $f(x_d)$, ou de forma equivalente, os termos constantes do gradiente de $f(x_d)$. Como o gradiente do termo acima é dado por

$$\nabla f(x_d) = 2kx_d - 2kp_d \tag{6-7}$$

basta somar o valor $-2kp_d$ na posição referente a x_d no vetor **q**.

Logo, para adicionar as contribuições da função de penalização na matriz **Q** e no vetor **q**, basta percorrer a malha de entrada e somar $2k$ na diagonal de **Q** para cada vértice não fixado em uma célula ativa. Da mesma forma, o vetor **q** recebe a contribuição de $-2kp_d$. É claro que estas contribuições devem ser somadas nas posições referentes as das variáveis dos vértices analisados.

6.2.2 Função de Suavização

De forma geral, todas as discretizações do operador de Laplaci-Beltrami utilizam a seguinte máscara

$$w_m(v_i) = \begin{cases} -\sum_{v_j \in N_1(v_i)} w_{ij} & \text{se } v_i = v_j \\ w_{ij} & \text{se } v_i \neq v_j \end{cases} \tag{6-8}$$

O termo de suavização da função objetivo deste trabalho faz uso do operador de Kirchhoff, onde todo peso $w_{ij} = 1$ para $i \neq j$. De modo que é somado um termo na função objetivo para cada vértice da forma

$$\mathbf{L}(v_i)^2 = \left(\sum_{v_j \in N_1(v_i)} \left(\mathbf{x}_{v_j} - \mathbf{x}_{v_i} \right) \right)^2 \tag{6-9}$$

no entanto, para as derivações a seguir serão assumidos pesos gerais, ou seja, suponha que cada termo da função objetivo seja da forma

$$\mathbf{L}(v_i)^2 = \left(\sum_{v \in N_{0,1}(v_i)} w_v \mathbf{x}_v \right)^2 \tag{6-10}$$

onde $N_{0,1}(v) = N_1(v_i) \cup \{v\}$.

Desta forma, as contribuições deste termo para matriz **Q** são dadas por todas as combinações possíveis de segundas derivadas, ou seja, dados $v_j, v_k \in N_{0,1}(v_i)$ vértices desconhecidos na malha, o termo correspondente a v_j e v_k é dado por

$$\frac{\partial^2 L(v_i)^2}{\partial x_{v_j} \partial x_{v_k}} = 2 w_{v_j} w_{v_k} \tag{6-11}$$

Note que um mesmo vértice contribui em linhas distintas da matriz **Q**. Além disso, se existir um vértice $v \in N_{0,1}(v_i)$ com valor conhecido, um valor constante aparecerá em $\nabla \mathbf{L}(v_i)^2$, desta forma, para cada vértice $v_j \in N_{0,1}(v_i)$ que tenha sua posição p_{v_j} conhecida, todos os vértices não conhecidos $v_k \in N_{0,1}(v_i)$ adicionam a contribuição de $2 w_{v_j} w_{v_k} p_{v_j}$ ao vetor **q** na posição relativa à variável do vértice v_k. Este resultado é obtido diretamente de $\nabla \mathbf{L}(v_i)^2$. Ao diferenciar $\mathbf{L}(v_i)^2$ em relação a v_k, o termo da variável v_j é constante na expressão e a contribuição a **q** é obtida substituindo a variável, com valor conhecido, pelo seu valor p_{v_j}.

Como um exemplo, considere a discretização de Kirchhoff do operador Laplace-Beltrami na Figura 6.1. Os vértices d e e tem suas posições conhecidas, logo, não são variáveis do problema. Estas posições são $p_{v_d} = 2$ e $p_{v_e} = -1$

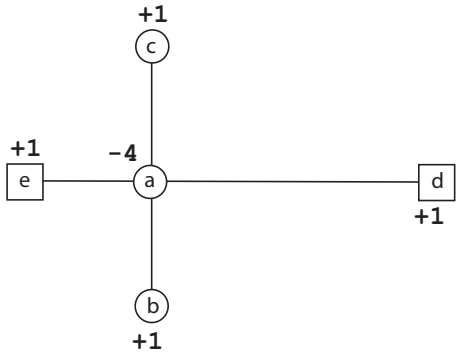

Figura 6.1: Exemplo bidimensional para a discretização de Kirchhoff do operador Laplace-Beltrami. Os vértices d e e são conhecidos e portanto não são variáveis do problema. $p_{v_d} = 2$ e $p_{v_e} = -1$.

Seguindo as derivações apresentadas acima, a matriz Hessiana desta máscara é dada por

$$\begin{bmatrix} \frac{\partial L(a)^2}{\partial x_a^2} & \frac{\partial L(a)^2}{\partial x_a \partial x_b} & \frac{\partial L(a)^2}{\partial x_a \partial x_c} \\ \frac{\partial L(a)^2}{\partial x_b \partial x_a} & \frac{\partial L(a)^2}{\partial x_b^2} & \frac{\partial L(a)^2}{\partial x_b \partial x_c} \\ \frac{\partial L(a)^2}{\partial x_c \partial x_a} & \frac{\partial L(a)^2}{\partial x_c \partial x_b} & \frac{\partial L(a)^2}{\partial x_c^2} \end{bmatrix} = \begin{bmatrix} 2w_{v_a}^2 & 2w_{v_a}w_{v_b} & 2w_{v_a}w_{v_c} \\ 2w_{v_b}w_{v_a} & 2w_{v_b}^2 & 2w_{v_b}w_{v_c} \\ 2w_{v_c}w_{v_a} & 2w_{v_c}w_{v_b} & 2w_{v_c}^2 \end{bmatrix} = \begin{bmatrix} 32 & -8 & -8 \\ -8 & 2 & 2 \\ -8 & 2 & 2 \end{bmatrix} \tag{6-12}$$

Já o vetor **q** é dado por

$$\begin{bmatrix} 2w_{v_a}(w_{v_d}p_{v_d} + w_{v_e}p_{v_e}) \\ 2w_{v_b}(w_{v_d}p_{v_d} + w_{v_e}p_{v_e}) \\ 2w_{v_c}(w_{v_d}p_{v_d} + w_{v_e}p_{v_e}) \end{bmatrix} = \begin{bmatrix} -8(2-1) \\ 2(2-1) \\ 2(2-1) \end{bmatrix} = \begin{bmatrix} -8 \\ 2 \\ 2 \end{bmatrix} \tag{6-13}$$

Para verificar o resultado obtido, basta calcular as contribuições para a matriz **Q** e o vetor **b** através do cálculo de $\mathbf{L}(v_i)^2$, como a seguir

$$\begin{aligned} \mathbf{L}(v_i)^2 &= (b + c + 2 - 1 - 4a)^2 \\ &= 16a^2 - 8ab - 8ac + b^2 + 2bc + c^2 - 8a + 2b + 2c + 1 \end{aligned} \tag{6-14}$$

Ao diferenciar a Equação 6-14 verifica-se os resultados obtidos para as contribuições à matriz **Q** e o vetor **b**.

Note que para uma grade estruturada, estes valores são constantes, uma vez que as conexões dos vértices internos são idênticas. Desta forma, a discretização de Kirchhoff é constante para a malha do reservatório e sempre tem os pesos iguais aos apresentados na Figura 5.7 para todos os vértices. Desta forma as contribuições da suavização do vértice central à matriz **Q** podem ser derivadas com regras bem simples, como a seguir:

- A contribuição do vértice central para o próprio vértice central é dada por $2w_{v_a}^2 = 72$.
- A contribuição paras as linhas e colunas do vértice central é dada por $2w_{v_a}w_{v_b} = -12$ para todo $v_b \in N_{0,1}(v_a)$.
- Todas as outras contribuições são dadas por $2w_{v_b}w_{v_c} = 2$ para todo $v_b, v_c \in N_{0,1}(v_a)$.

A mesma lógica pode ser seguida para definir o vetor **q**. Desta forma, para montar o sistema de equações lineares, basta percorrer a malha somando a contribuição de cada vértice na matriz **Q** e no vetor **q**.

6.3 Solução com alocação da matriz Q

Uma vez conhecida a forma de montar o sistema de equações, a biblioteca LIS (*Library of Iterative Solvers for Linear Systems*) (Nishida *et al.*, 2009) foi utilizada para resolver o sistema de equações lineares. A biblioteca permite a seleção de vários métodos de solução com diferentes precondicionadores, além de formatos que permitem representar matrizes esparsas armazenando apenas os seus elementos não nulos.

Para resolver o problema em questão foi utilizado o método dos gradientes conjugados com precondicionador de Jacobi e a matriz foi armazenada utilizando o formato CSR(*Compressed Sparse Row*). O formato CSR utiliza três vetores para armazenar os elementos não nulos de uma matriz, estes vetores serão chamados de *ptr*, *index* e *value*. Supondo que a matriz **Q** tenha nnz elementos não nulos e n variáveis, estes vetores são descritos como

value: vetor com nnz elementos do tipo *double* que armazena os elementos não nulos de **Q**.

index: vetor com nnz elementos inteiros. Cada posição armazena a coluna na matriz que o respectivo elemento gravado no vetor *value* ocupa.

ptr: vetor com $n+1$ elementos inteiros que indica o início de cada uma das linhas no vetor *index*.

A seguinte matriz

$$\begin{bmatrix} 11 & 0 & 0 & 0 \\ 21 & 22 & 0 & 0 \\ 0 & 32 & 33 & 0 \\ 41 & 0 & 43 & 44 \end{bmatrix} \tag{6-15}$$

é representada em formato CSR da seguinte forma

$$\begin{aligned} \textit{value} &= [11, 21, 22, 32, 33, 41, 43, 44] \\ \textit{index} &= [0, 0, 1, 1, 2, 0, 2, 3] \\ \textit{ptr} &= [0, 1, 3, 5, 8] \end{aligned} \tag{6-16}$$

Apesar de resolver o problema proposto, esta forma de solução apresenta uma limitação de memória para modelos grandes, mesmo representando a matriz em formatos eficientes, como CSR. Para ilustrar isso, considere um reservatório com

dimensões dadas por $409 \times 321 \times 161$ o qual os autores deste trabalho tiveram contato durante o desenvolvimento do mesmo. Isso resultaria em $21.137.529$ variáveis para cada dimensão, o que leva a uma matriz com $528.438.225$ elementos não nulos. Ao representar essa matriz no formato CSR, usando inteiros de 4 *bytes* para os vetores *index* e *ptr*, e *double* (8 *bytes*) para o vetor *value*, seriam necessários aproximadamente 6 GB para representar apenas a matriz para cada uma das dimensões[2]. Como a matriz **Q** permanece estática no Algoritmo 1 e é usada apenas para multiplicações por vetor, uma redução no montante de memória pode ser alcançado ao representar o vetor *values* por uma representação inteira de 8 *bits*, uma vez que o operador de Kirchhoff gera pesos inteiros e dentro intervalo $[-128, 127]$ nos vértices internos da grade do reservatório. Com esta melhoria, a matriz **Q** ocuparia cerca de 2.5 GB em memória. Esta opção não é possível ao utilizar a LIS e em geral também não é possível em outras bibliotecas, sendo necessário assim uma implementação própria do método dos gradientes conjugados com o precondicionador de Jacobi.

A análise acima foi feita com base apenas na malha do entorno da região do reservatório. Na presença de horizontes no modelo, o número de variáveis do problema aumentará ainda mais. Note a importância do uso da discretização de Kirchhoff para essa redução no montante de memória necessário para representar a matriz **Q**. A forma como a montagem da matriz foi descrita na Seção 6.2 independe dos pesos em questão, logo essa otimização não é valida para o caso geral. A simples mudança para o operador uniforme obrigaria o uso de uma representação em ponto flutuante para o vetor *value*.

6.4 Solução sem alocação da matriz Q

O método dos gradientes conjugados utiliza a matriz **Q** apenas em multiplicações por vetores. Se fosse possível reconstruir as linhas de **Q** de maneira eficiente, estas multiplicações poderiam ser feitas sem a necessidade de manter a matriz em memória. Buscando eliminar a necessidade de grandes montantes de memória para otimização do modelo proposto, será descrita nesta seção uma solução do sistema de equações lineares 6-3 que realiza as multiplicações presentes no método dos gradientes conjugados iterando sobre a malha, sem armazenar a matriz **Q** em memória. Esta técnica é similar à uma antiga técnica utilizada em elementos finitos conhecida como elemento-por-elemento (Wathen, 1989; Augarde *et al.*, 2006), onde problemas muito grandes são resolvidos sem o passo de *assembly* da matriz. Em geral, cada elemento contribui para um bloco da matriz global.

[2]O cálculo do número de variáveis, e consequentemente do montante de memória necessário, não levam em conta os vértices que foram fixados, no entanto, em geral a economia de memória ao desconsiderar tais vértices é desprezível.

Desta forma, em um método iterativo, como o método dos gradiente conjugados, essas contribuições podem ser multiplicadas pelo vetor solução corrente e o valor incrementado em um vetor auxiliar para cálculo posterior do residual. Isto faz com que grandes problemas possam ser resolvidos usando montantes pequenos de memória, o que habilita soluções usando GPUs (Duarte *et al.*, 2015).

Essa seção descreve um método baseado na técnica de elemento-por-elemento para reconstruir cada linha da matriz, independente dos pesos utilizados. Depois este método é aplicado ao caso específico do operador Kirchhoff na grade do reservatório e então, uma análise do montante de memória utilizado pelo método dos gradiente conjugados é feita.

6.4.1 Reconstrução das linhas da matriz Q

Cada vértice a não conhecido na malha gera uma linha na matriz $\mathbf{Q}$, logo, para reconstruir individualmente cada linha de $\mathbf{Q}$ é necessário entender como os vértices vizinhos ao vértice a influenciam na geração da linha de a. Cada vértice da malha soma o seguinte termo

$$\mathbf{L}(a)^2 = \left(\sum_{v \in N_{0,1}(a)} w_v \mathbf{x}_v \right)^2 \tag{6-17}$$

na função objetivo. A Figura 6.2 ilustra todas as máscaras que contém o vértice a em uma grade bidimensional.

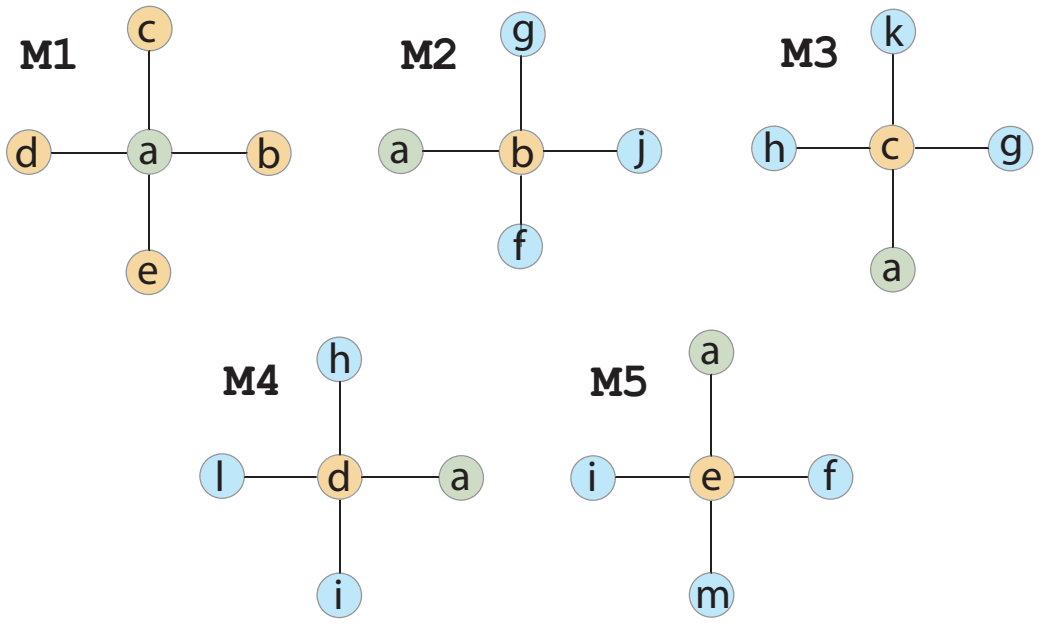

Figura 6.2: Todas as máscaras possíveis que contém o vértice a em uma grade bidimensional.

Note que a união destas máscaras formam uma máscara maior apresentada na Figura 6.3. Este é o conjunto de vértices aparecem na linha do vértice a na matriz $\mathbf{Q}$. Como o operador laplaciano relaciona um vértice e seus vizinhos, o vértice central a é relacionado a todos os vértices que distam até duas arestas. Esta conexão é feita pelas mascaras dos vértices vizinhos ao vértice a, em amarelo na Figura 6.3.

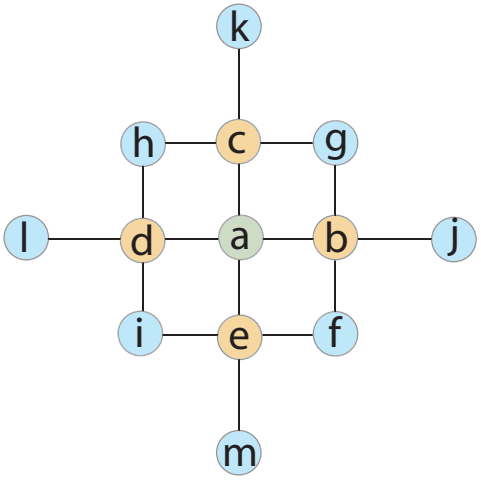

Figura 6.3: Vértices que integram as colunas na linha do vértice a para o caso de uma grade bidimensional. O vértices vizinhos ao vértice central a fazem com que as colunas não nulas na linha de a sejam todos os vértices que distam até duas arestas do vértice a.

Uma vez que um vértice se relaciona a outros vértices que que distam até duas arestas do vértice central, conclui-se que o número de elementos não nulos na linha é, no máximo, o número total de vértices nessa vizinhança. O número de elementos não nulos pode ser menor caso algum destes vértices tenha valor conhecido.

Da mesma forma que esta máscara foi obtida unindo todas as máscaras que contém o vértice central a, os pesos dos seus vértices podem ser obtidos somando as contribuições de todas as máscaras. Como visto na Seção 6.2, dado um vértice v em uma das máscaras que contém o vértice a, a contribuição deste termo é dada por

$$2w_a w_v \tag{6-18}$$

Desta forma, o valor da diagonal referente à variável do vértice a é dada por

$$2\sum_m (w_a^m)^2 \tag{6-19}$$

onde m é uma das máscaras que contém a e w_a^m é o peso do atribuído ao vértice a na mascara m. Para o exemplo da grade bidimensional da Figura 6.2, este valor para o vértice a seria

$$2(w_a^{M1})^2 + 2(w_a^{M2})^2 + 2(w_a^{M3})^2 + 2(w_a^{M4})^2 + 2(w_a^{M5})^2 \tag{6-20}$$

Ainda no mesmo exemplo, o valor da coluna do vértice b na linha da matriz referente ao vértice a, seria dado por

$$2w_a^{M1}w_b^{M1} + 2w_a^{M2}w_b^{M2} \tag{6-21}$$

Note, que este procedimento pode ser realizado para quaisquer pesos em qualquer tipo de malha, ou seja, é aplicável a todas as discretizações possíveis do operador laplaciano. Para malhas não estruturadas o acesso à vizinhança necessária pode levar à uma solução ineficiente, porém, para malhas estruturadas, o acesso à essa vizinhança pode ser feita em tempo constante.

Assumindo a discretização de Kirchhoff, os vértices centrais em cada mascara do exemplo da Figura 6.2 tém peso igual a -4 e os outros vértices das figuras tem peso igual a 1, logo, os valores das colunas de cada um dos vértices na matriz **Q**, na linha de a, são dados por

$$\begin{aligned}
col_a(a) &= 2(-4)^2 + 2(1)^2 + 2(1)^2 + 2(1)^2 + 2(1)^2 = 40 \\
col_a(b) &= col_a(c) = col_a(d) = col_a(e) = 2(-4 \times 1) = -8 \\
col_a(f) &= col_a(g) = col_a(h) = col_a(i) = 2(1+1) = 4 \\
col_a(j) &= col_a(k) = col_a(l) = col_a(m) = 2 \times 1 = 2
\end{aligned} \tag{6-22}$$

onde $col_{v_i}(v_j)$ representa o valor do elemento na linha do vértice v_i e na coluna do vértice v_j. Note que ao usar uma malha estuturada e uma mesma discretização do operador laplaciano para todos os vértices, as linhas da matriz **Q** podem ser pré-computadas, como ilustrado na Figura 6.4, onde todos os pesos para o caso de uma grade bidimensional usando a discretizaçao de Kirchhoff são apresentados.

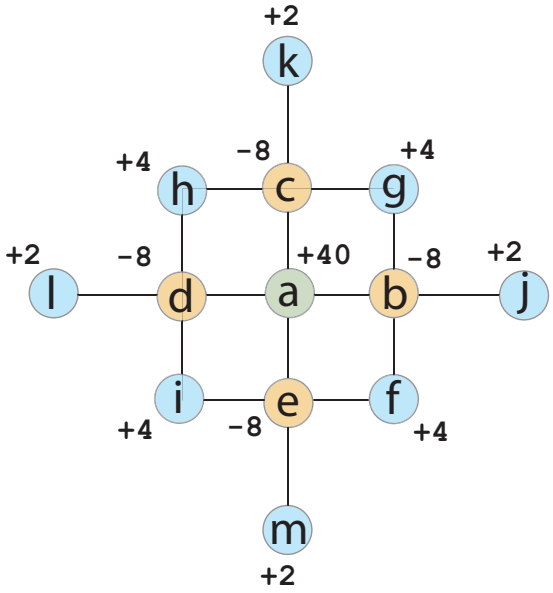

Figura 6.4: Pesos pré-computados para uma grade bidimensional usando a discretização de Kirchhoff para o operador de Laplace-Beltrami.

Seguindo a mesma ideia, a Figura 6.5 ilustra a vizinhança em uma grade tridimensional dos vértices que integram a linha do vértice central na matriz **Q**. Como existem 25 vértices nessa máscara, cada linha da matriz tem no máximo 25 elementos não nulos.

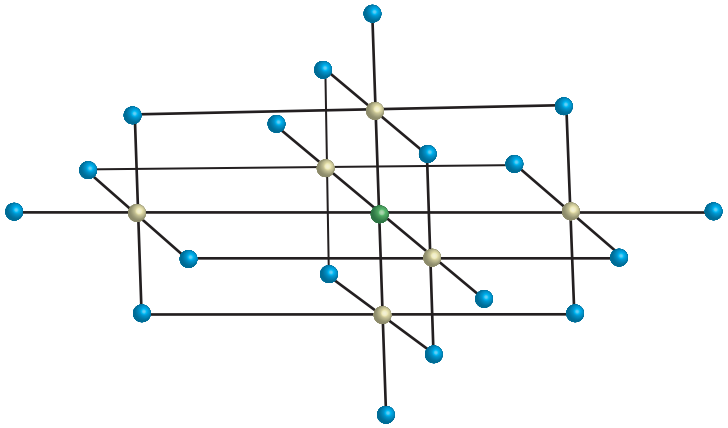

Figura 6.5: Vizinhança que integra a linha de um vértice em uma grade tridimensional.

Para uma grade tridimensional, a discretização de Kirchhoff é da forma como da Figura 5.7 para todos os vértices. Desta forma, as linhas da matriz **Q** podem ser pré-computadas da mesma forma do caso bidimensional. A Figura 6.6 apresenta os valores das colunas dos respectivos vértices na linha do vértice central.

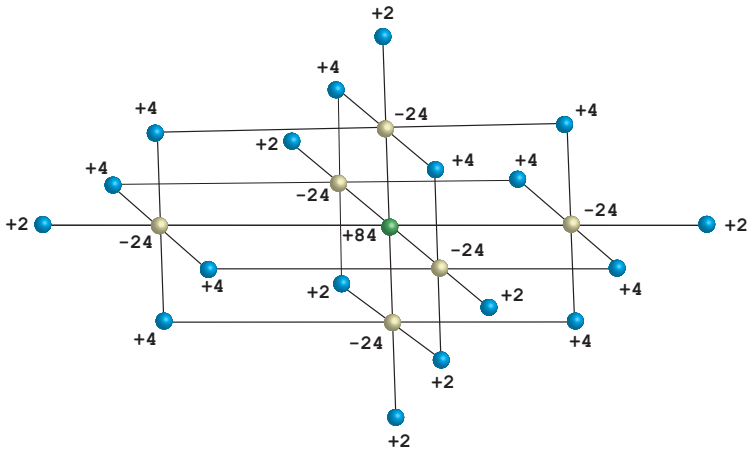

Figura 6.6: Pesos pré-computados para uma grade tridimensional usando a discretização de Kirchhoff para o operador de Laplace-Beltrami. Os pesos são os elementos que compõem as colunas do vértice central na matriz **Q**.

A parte correspondente à função de penalização que integra a função objetivo é mais simples de computar. Como o fator de suavização k é constante para todas as penalizações, e cada penalização de um vértice soma $2k$ na diagonal da matriz **Q**, como visto na Seção 6.2.1, basta calcular o número p de penalizações que cada vértice sofre, em tempo de pre-processamento, e em tempo de execução somar $2kp$

na diagonal referente ao vértice para reconstruir corretamente a linha. Como o vetor **b** precisa ser calculado apenas uma vez por dimensão, a descrição anterior para a sua construção é satisfatória.

Uma vez conhecido um método para reconstruir em tempo de execução cada linha da matriz **Q**, o método dos gradientes conjugados pode ser executado sem a necessidade de persistir tal matriz em mémoria, eliminando assim a principal limitação de memória na solução do método.

7
Resultados

Este capítulo descreve os resultados obtidos pela abordagem proposta no modelo Brugge (Peters *et al.*, 2013) e o no modelo Norne[1]. A Figura 7.1 ilustra os dados de entrada.

A versão atual do modelo Brugge possui dimensões aproximadas de $10.000 \times 3.000 \times 61$ m^3 e uma malha com 60048 células, estruturadas em uma grade com $139 \times 48 \times 9$. Deste total, 43.474 células são ativas. O Brugge é um modelo sintético criado no ATW (*SPE Applied Technology Workshop*) em 2008 com o objetivo de estudar estratégias de injeção para otimizar a produção do reservatório. Desde então este modelo é uma referência em testes de simulação de fluxo em reservatórios. Os autores deste trabalhem agradecem a TNO[2] por fornecer o dado para testes.

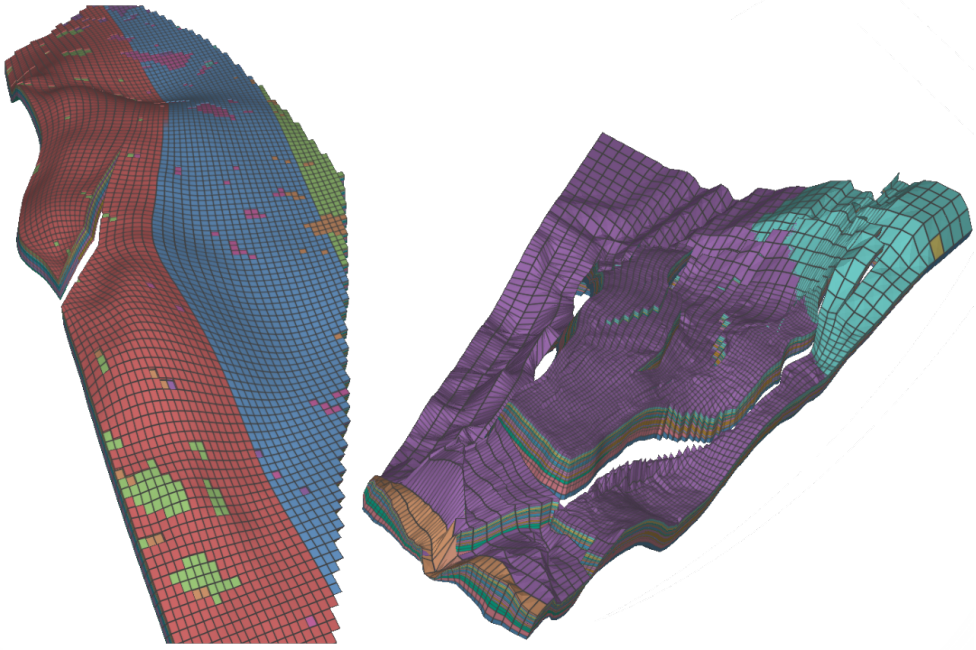

Figura 7.1: Modelos utilizados para teste. À esquerda o modelo Brugge e à direita o modelo Norne.

O modelo Norne é um dado real público do Mar da Noruega com dimensões aproximadas de $12.000 \times 5.500 \times 350$ m^3. A malha do modelo tem 113.344 células

[1] https://opm-project.org/?page_id=559
[2] https://www.tno.nl/en/

estruturadas em um grid de dimensão $46 \times 112 \times 22$. Deste total, 44.927 são células ativas. As imagens exibidas neste capítulo foram geradas usando o software Pos3D desenvolvido no Instituto Tecgraf[3].

7.1 Exemplo em duas dimensões

Antes de exibir os resultados nos dados de teste, o comportamento do algoritmo será analisado em um dado simples de duas dimensões. O exemplo selecionado é dado por uma malha de quadriláteros composta de 45 elementos e 65 nós. A malha é separada em duas por uma descontinuidade, como ilustrado na Figura 7.2.

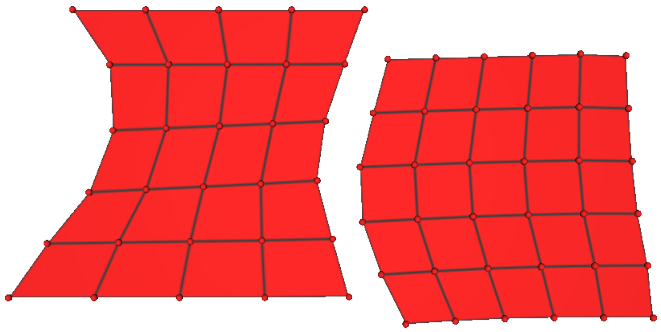

Figura 7.2: Exemplo bidimensional para ilustração do funcionamento do algoritmo. Malha composta por 45 elementos e 65 nós. A grade com dimensão de 9×5 elementos é separada em duas por uma descontinuidade.

Os exemplos exibidos a seguir foram gerados usando uma grade de saída com dimensão 27×15 elementos, resultando em uma malha de 405 elementos e 420 nós. Os elementos correspondentes aos elementos da malha de entrada foram posicionados no centro da grade de saída.

7.1.1 Exemplo com todos os elementos ativos

Em um primeiro exemplo foi considerado que todos os elementos elementos da malha de entrada são ativos, logo o deslocamento de todos os vértices deve ser penalizado. O algoritmo foi executado para penalizações dadas por $k = 0.0$, $k = 0.5$, $k = 5.0$ e $k = 100.0$. A Figura 7.3 ilustra as malhas geradas para cada valor de k. Como esperado, quando $k = 0$ a malha resultante é uma grade regular, uma vez que, neste caso, a função objetivo minimiza apenas a função de suavização. À medida que k aumenta, a penalização dos deslocamentos das células ativas se torna mais importante que a suavização da malha resultante, logo a malha resultante se adapta melhor à malha de entrada.

[3] `https://www.tecgraf.puc-rio.br/`

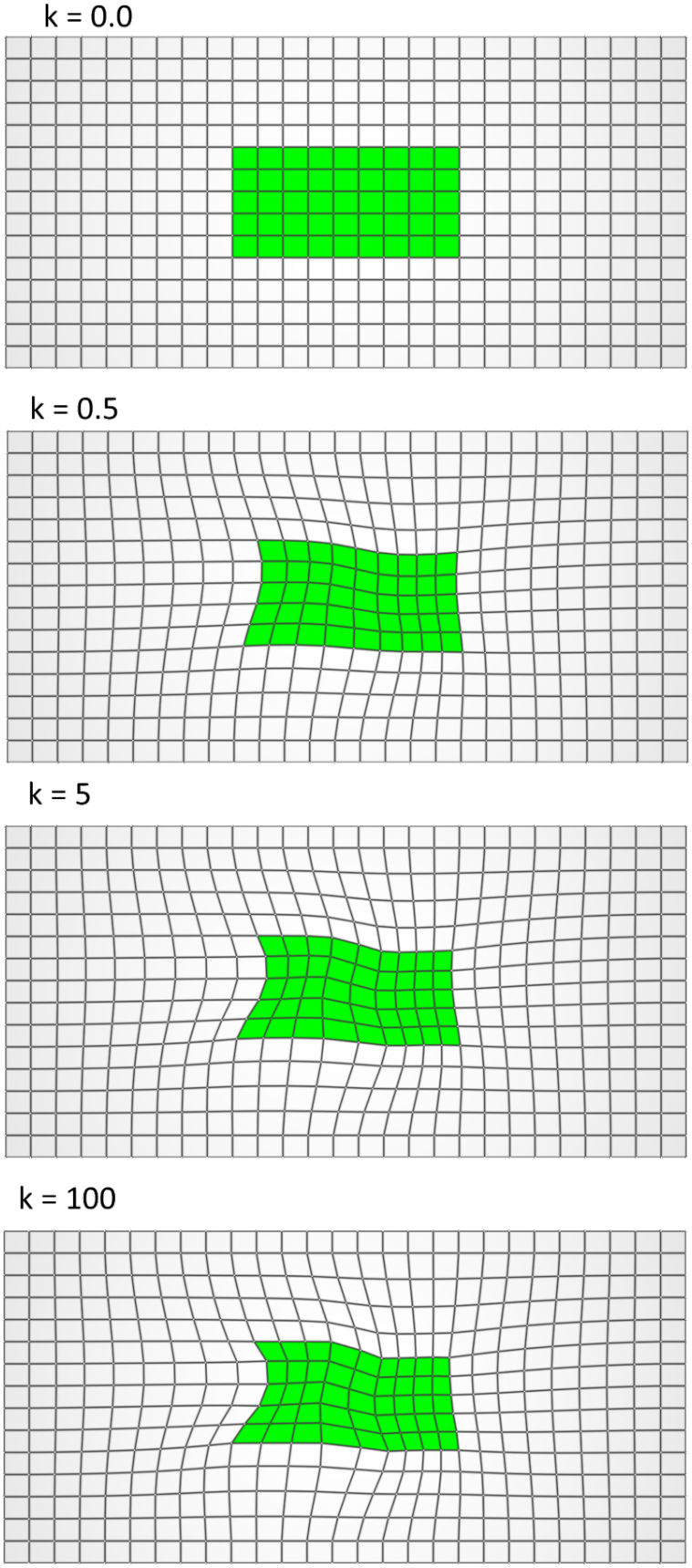

Figura 7.3: Malhas resultantes para $k = 0.0$, $k = 0.5$, $k = 5.0$ e $k = 100.0$.

A Figura 7.4 ilustra a comparação entra a malha de entrada e a malha resultante. As duas malhas foram sobrepostas e uma transparência aplicada de forma que as duas malhas pudessem ser visualizadas. Note que à medida que o valor da penalização aumenta o erro de aproximação da malha resultante diminui. Isso se dá ao fato do fator de penalização poder ser interpretado como um fator de suavização, desta forma, quanto menor o valor de k, mais suave é a malha resultante. À medida que a penalização pelo deslocamento se sobrepõe à suavização, os pontos distantes das regiões de falha da malha resultante e o seus correspondentes da malha de entrada tendem a coincidir.

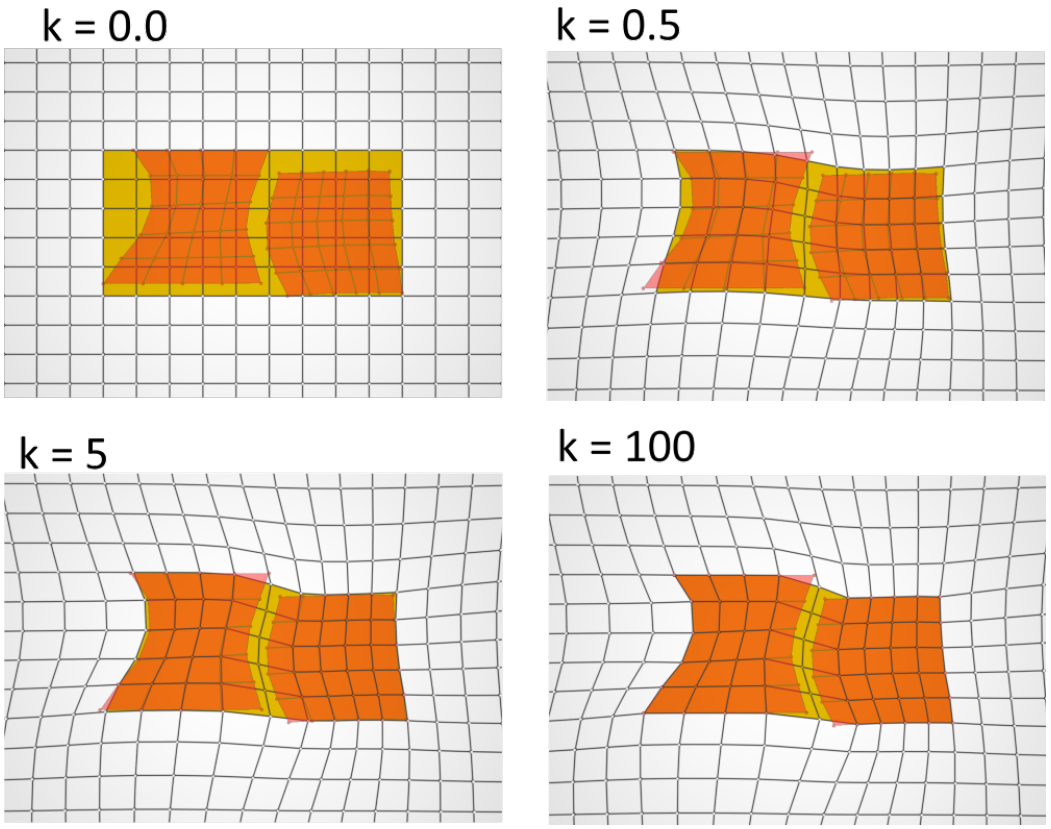

Figura 7.4: Malhas de entrada e saída sobrepostas. Em vermelho a malha de entrada e amarelo a malha de saída. Na sobreposição das duas malhas, a malha de entrada pode aparecer em laranja pelo efeito da transparência.

7.1.2 Exemplo com apenas uma componente de elementos ativos

Em um segundo exemplo, apenas a componente à direita da malha de entrada foi considerada contendo elementos ativos. Desta forma, apenas estes elementos contribuem com penalizações para o problema de otimização.

Note que neste exemplo não há descontinuidades nas células ativas. No entanto, a penalização ainda é somada para cada vértice em células ativas. Desta forma a malha resultante é uma versão suavizada da malha de entrada, como

ilustrado na Figura 7.5. Se o usuário não desejar que a malha seja suavizada neste casos, basta fixar as células distantes das regiões de falha. No exemplo da Figura 7.5, todos os vértices poderiam ser fixados para o problema de otimização. Consequentemente nenhuma penalização seria necessária.

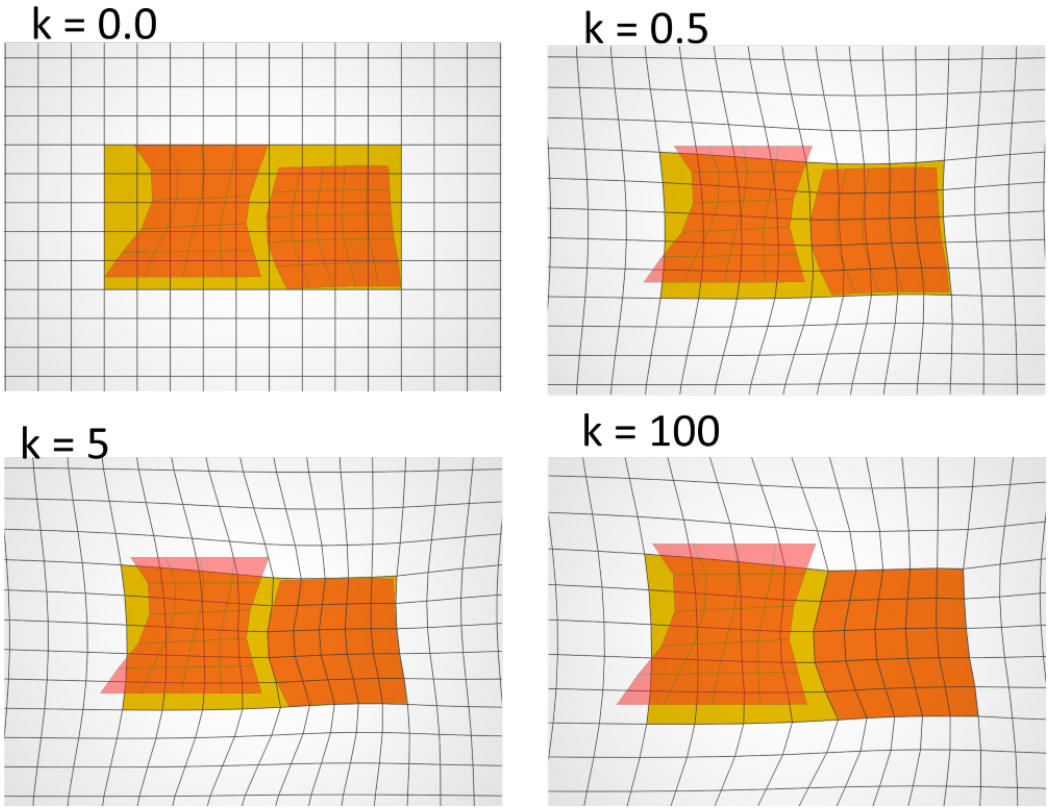

Figura 7.5: Malhas de entrada e saída sobrepostas. Neste exemplo apenas a componente direita da malha é considerada ativa. Desta forma, não há descontinuidades entre as células ativas. Todas as células ativas poderiam ser fixadas se não fosse desejada uma versão suavizada da malha de entrada.

7.2 Avaliação Visual

A Figura 7.6 apresenta uma visão global do modelo Norne com horizontes acima e abaixo do reservatório. Os horizontes apresentados foram importados de uma sísmica e adaptados para testes no modelo, uma vez que o dado não possui horizontes. Para geração da malha o modelo foi dobrado nas dimensões laterais e o último horizonte do topo foi colocado a 2.800 m acima do reservatório, já o horizonte da base foi colocado 2.000 m abaixo do reservatório. A dimensão da grade do modelo resultante é $59 \times 145 \times 199$.

A Figura 7.7 apresenta um horizonte de entrada e o seu mapeamento na malha resultante.

A Figura 7.8 ilustra uma região com múltiplas falhas do modelo Norne e as respectivas malhas produzidas pelo método proposto nestas regiões. Note que as

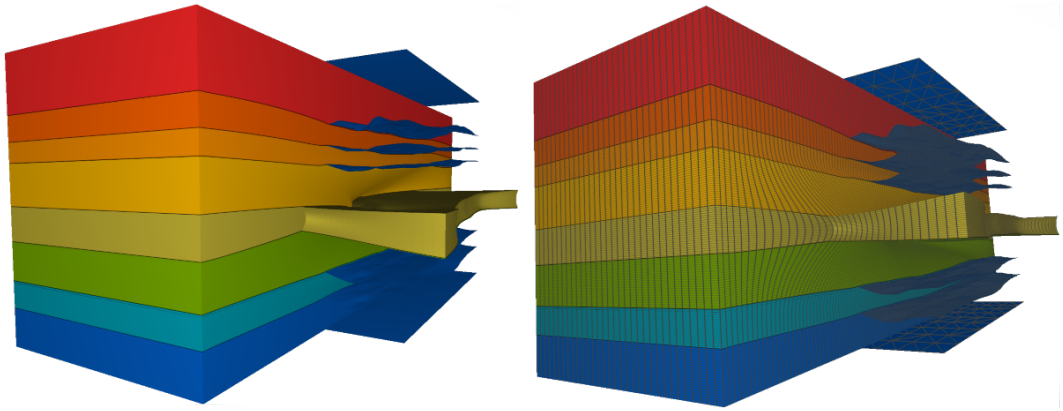

Figura 7.6: Modelo Norne com horizontes com horizontes acima e abaixo do reservatório.

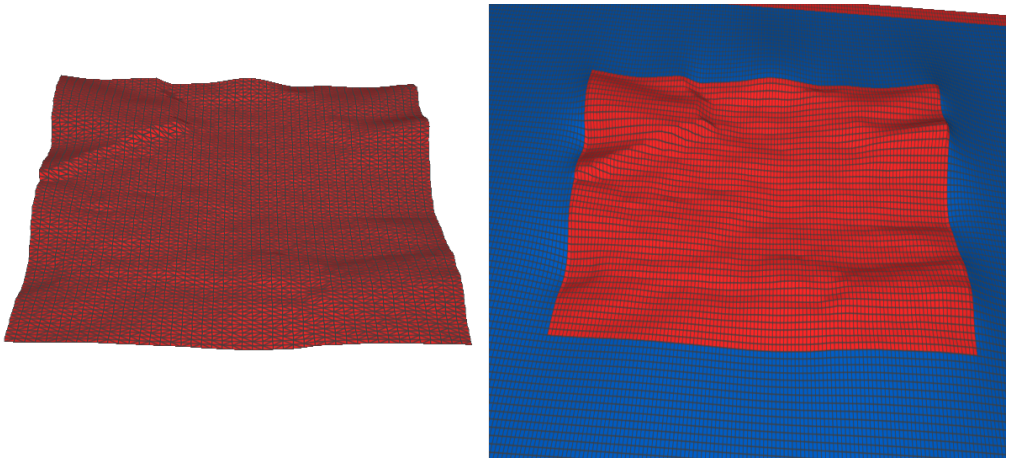

Figura 7.7: À esquerda uma malha triangular representando um horizonte do dado de entrada. À direita o horizonte mapeado na malha resultante.

células ativas (em vermelho) que possuem vizinhas inativas permanecem imóveis após o processo de geração da malha. Isto ocorre porque as células inativas (em azul) não tem seus deslocamentos penalizados, logo a descontinuidade pode ser eliminada movendo apenas as células inativas. Isto ocorre dos lados direito e esquerdo da malha apresentada na Figura 7.8.

Os exemplos a seguir são de imagens da malha gerada para o modelo Brugge. A malha foi estendida pelas laterais usando uma distância de $1.5l$, onde l é a largura do modelo. Foi estendida para cima e para baixo usando fatores de sete e 15 vezes a espessura do reservatório, respectivamente. Foram usadas 45 células para estender o modelo em cada um dos sentidos de i, 30 células em cada um dos sentidos de j, 50 células abaixo e 25 células acima do reservatório, resultando em uma grade de dimensão $229 \times 108 \times 84$. O fator k, que controla a suavização da malha resultante, foi definido como $k = 0.3$.

A Figura 7.9 dá uma visão geral da malha gerada, assim como sua discretização em torno do reservatório. À direita o reservatório é ilustrado no centro

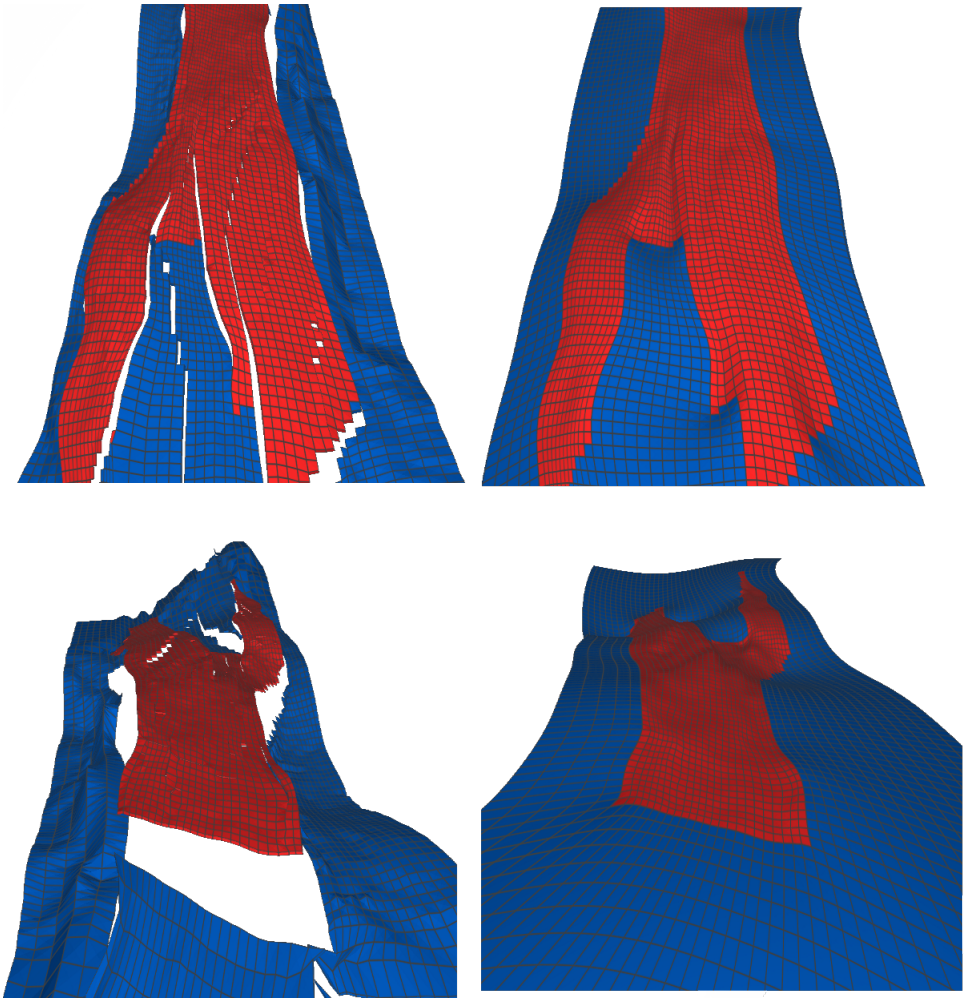

Figura 7.8: Trechos do dado Norne com múltiplas falhas. À esquerda é mostrada a entrada e à direita o resultado obtido. Descontinuidades que possuem células ativas (em vermelho) somente de um lado são eliminadas movendo apenas as células inativas (em azul).

da malha. Nas imagens do Brugge a seguir nesta seção, as células em amarelo são células ativas, as células em laranja são células inativas do reservatório e as demais são células usadas na extensão do modelo.

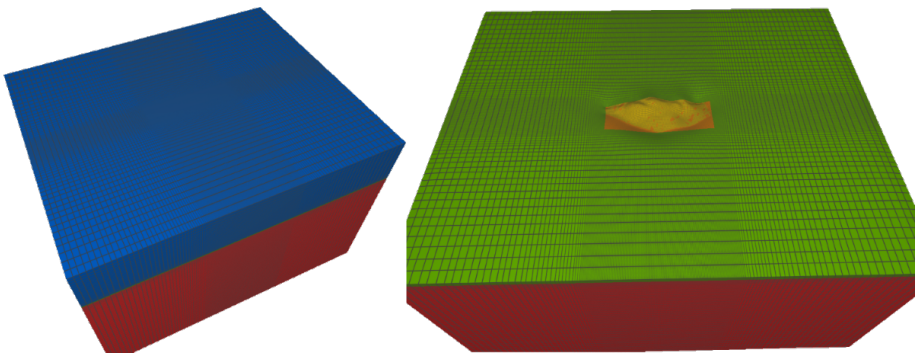

Figura 7.9: Bloco gerado em torno da malha do reservatório.

A Figura 7.10 mostra três vistas diferentes da região do reservatório na malha resultante.

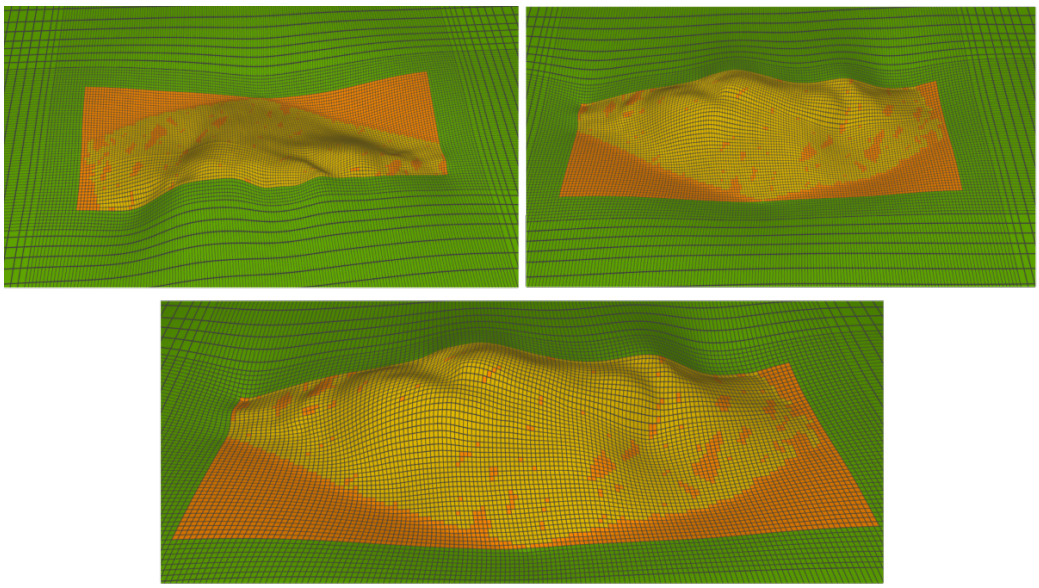

Figura 7.10: Três diferentes visões da malha resultante do reservatório.

Já a Figura 7.11 mostra como ficou a região, onde antes existia uma falha, na malha resultante. Note que a região foi suavizada, gerando elementos de boa aparência com transições suaves. As regiões de falha são as regiões onde potencialmente elementos ruins são geradas, uma vez que é comum existir grandes declives nestas regiões.

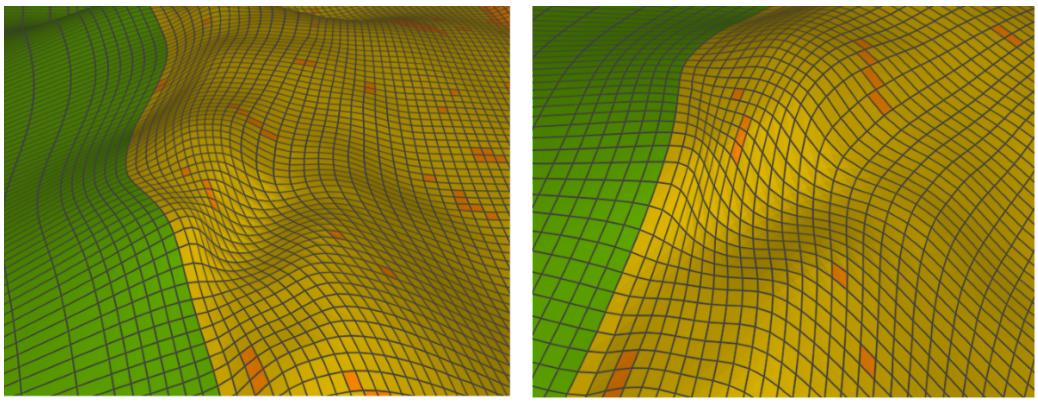

Figura 7.11: A região da falha suavizada na malha resultante.

7.3 Influência do fator de suavização nos dados de teste

Um dos parâmetros de entrada do método proposto é o fator k que controla a suavização do modelo de resultante. Como dito anteriormente no texto, quanto menor o valor de k, mais suave será a malha resultante. No entanto, quanto maior a

suavização, maior o deslocamento das células ativas nas regiões de descontinuidade. Em contrapartida, ao fornecer um valor alto para k, permite-se mudanças bruscas na curvatura da malha, melhorando assim a aproximação nas regiões de falha. Porém, isso pode levar a elementos distorcidos. A Figura 7.12 ilustra o modelo Brugge gerado para quatro valores diferentes de k. Da esquerda para a direita, $k = 0.0$, $k = 0.3$, $k = 1.0$ e $k = 10.0$. Para o valor $k = 0.0$, a função objetivo minimiza apenas a parte que controla a suavização da malha, portanto, é esperada uma malha totalmente suave, mas que não respeita as células ativas. Para o valor de $k = 10.0$, espera-se a maior variação na região de falha do modelo.

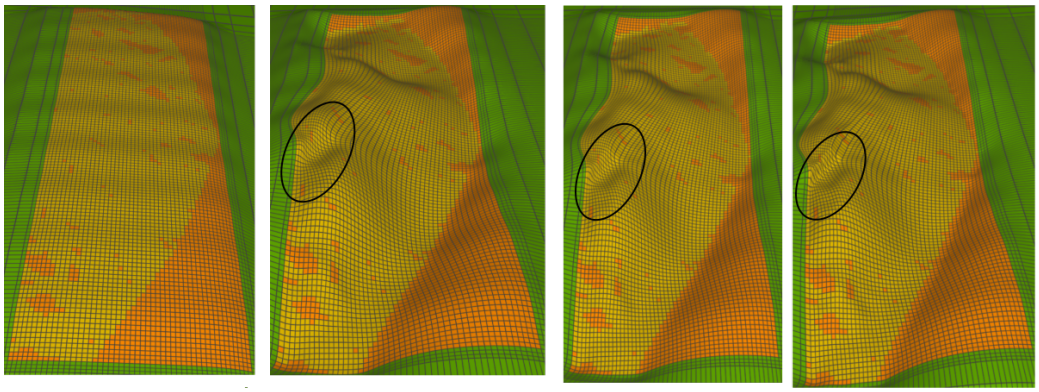

Figura 7.12: Modelo Brugge gerado variando o valor de k. Da esquerda para a direita, $k = 0.0$, $k = 0.3$, $k = 1.0$ e $k = 10.0$. À medida que k aumenta varições mais bruscas na região de falha são notadas.

7.4 Qualidade dos elementos gerados e condições de contorno

Uma vez que a malha gerada pelo método proposto deverá ser utilizada em uma simulação de elementos finitos, não devem existir elementos com jacobiano negativo. Na modelagem do método em si, não existe nenhuma restrição específica para evitar a formação de elementos não convexos ou com auto interseções. Durantes os testes não foi incomum a geração de elementos inválidos nos modelos. No entanto, a quantidade observada destes elementos sempre foi menor que 1% do total de elementos da malha. Em geral, estes elementos aparecem em torno das falhas. Uma forma de lidar com isso é usar um pós-processamento para tratar as células com jacobiano negativo. Este pós-processamento ainda não foi implementado.

A qualidade da malha gerada é diretamente proporcional à qualidade das condições de contorno utilizada. A geometria das células ativas deve servir de guia para definir as posições das células da borda da malha.

7.5 Custo de memória na solução do sistema de equações lineares

O método dos gradientes conjugados apresentado no Algoritmo 1, além da matriz, faz o uso de sete vetores: r, q, x, p, y, z e outro vetor para o precondicionador de Jacobi. Com o objetivo de diminuir o montante de memória requerido pelo método, o pseudo-código do Algoritmo 1 foi modificado conhecendo as propriedades específicas do problema de otimização a ser resolvido.

O precondicionador de Jacobi é calculado sobre a diagonal da matriz Q. Esta diagonal é sempre dada por $84 + 2kp$, onde k é o fator de suavização e p é o número de penalizações do vértice. Em tempo de preprocessamento um vetor c pode ser usado para calcular o número de penalizações de cada vértice. Como um vértice pode sofrer no máximo seis penalizações, uma vez que cada vértice interno se conecta a outros seis vértices, um vetor de inteiros de 8 *bits* pode ser utilizado. Desta forma, o vetor auxiliar z pode ser descartado. Além disso, o vetor b é usado apenas para calcular o valor do residual inicial, logo, este vetor pode ser reutilizado como o vetor p no restante do algoritmo. Também não é necessário um novo vetor x, uma vez que é possível atualizar diretamente o dado de entrada. Em resumo, para resolver o problema de otimização proposto utilizando o método dos gradientes conjugados são necessários apenas quatro vetores: r, p, c, e y. O mais importante: não é necessário armazenar a matriz em memória para realizar as multiplicações.

Além dos vetores utilizados diretamente no método dos gradientes conjugados, um vetor auxiliar s para reindexar as variáveis é necessário, uma vez que os vértices fixados não geram variáveis no sistema de equações. Como é possível resolver o problema separadamente para cada dimensão, supondo que o problema de otimização de entrada tenha n variáveis, todos os vetores envolvidos no algoritmo terão seus tamanhos divididos por três. Logo, no pior caso, a quantidade de memória necessária é proporcional a $\frac{5n}{3}$, ou seja, menos que o dobro do tamanho do dado de entrada. No entanto, o montante de memória necessário é ainda menor, uma vez que o sistema é montado e resolvido apenas para $k < n$ variáveis desconhecidas do problema. Logo, o tamanho dos vetores r, p, c e y é $\frac{k}{3}$ e do vetor s é $\frac{n}{3}$. Os vetores r, p, c são do tipo *double* (8 *bytes*), o vetor s é de inteiros de 4 *bytes* e o vetor c de inteiros de 1 *byte*, logo o montante de memória auxiliar necessário, em bytes, é dado por

$$\frac{25k}{3} + \frac{4n}{3} \tag{7-1}$$

Como a entrada para o método que resolve o sistema de equações lineares é uma grade de pontos, com cada coordenada representada em *double*, o montante de memória necessário para representar a grade é dado por $8n$. Um limitante superior para o montante de memória auxiliar utilizado pelo algoritmo é $\frac{29n}{3}$, o que

representa aproximadamente 1.21 vezes o montante de memória necessário para representar a entrada dos dados do sistema, comprovando assim a eficiência no uso da memória da solução proposta.

7.6 Análise de tempo de processamento

Alguns modelos de diferentes tamanhos foram gerados e os respectivos tempos de processamento foram medidos. Para os testes realizados, foi utilizada uma máquina com processador Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz com 40 cores. O método proposto necessita de um pequeno montante de memória para realizar a otimização. No entanto, uma vez que a solução é paralelizável, quanto maior o poder de paralelismo, mais rápido o problema será resolvido. Para os teste abaixo, foram utilizadas 36 threads no processo de multiplicação de matriz por vetor no método dos gradientes conjugados.

A Tabela 7.1 mostra o tempo necessário para otimizar vários modelos de diferentes dimensões, assim como o montante de memória utilizado pelo *solver*. Todos os testes foram realizados com condição de parada para quando a norma do gradiente for menor que $\epsilon = 1$ e $\epsilon = 10^{-3}$.

Tabela 7.1: Tempos de execução do método de otimização utilizando uma tolerância para condição de parada dada por $\epsilon = 1$ e $\epsilon = 10^{-3}$.

Num. de Variáveis	**Dimensão**	**Pontos Fixos**	**Memória (MB)**	**Tempo(s)** $\epsilon = 1$	**Tempo(s)** $\epsilon = 10^{-3}$
348.210	152×59×21	72.258	4,92	5,0	7,0
1.366.596	180×83×40	142.068	17,70	27,0	40,0
2.681.280	202×101×54	207.948	33,92	107,0	175,0
2.681.280	222×117×67	277.722	54,79	234,0	431,0
6.264.000	238×131×78	343.884	77,61	381,0	694,0
8.358.096	254×143×88	410.304	103,00	756,0	1326,0
10.126.242	264×153×95	461.826	124,39	1017,0	1.905,0

A Figura 7.13 ilustra a evolução dos tempos da Tabela 7.1 para ambas as tolerâncias utilizadas.

A Figura 7.14 ilustra o gráfico para a evolução do tempo para ambas as tolerâncias. Para que fossem comparáveis, ambos foram colocados na mesma escala. Para isso os eixos foram normalizados pelo valor máximo. Note que praticamente não há diferença entre os gráficos, o que mostra que não houve problemas significativos relacionados a erros numéricos.

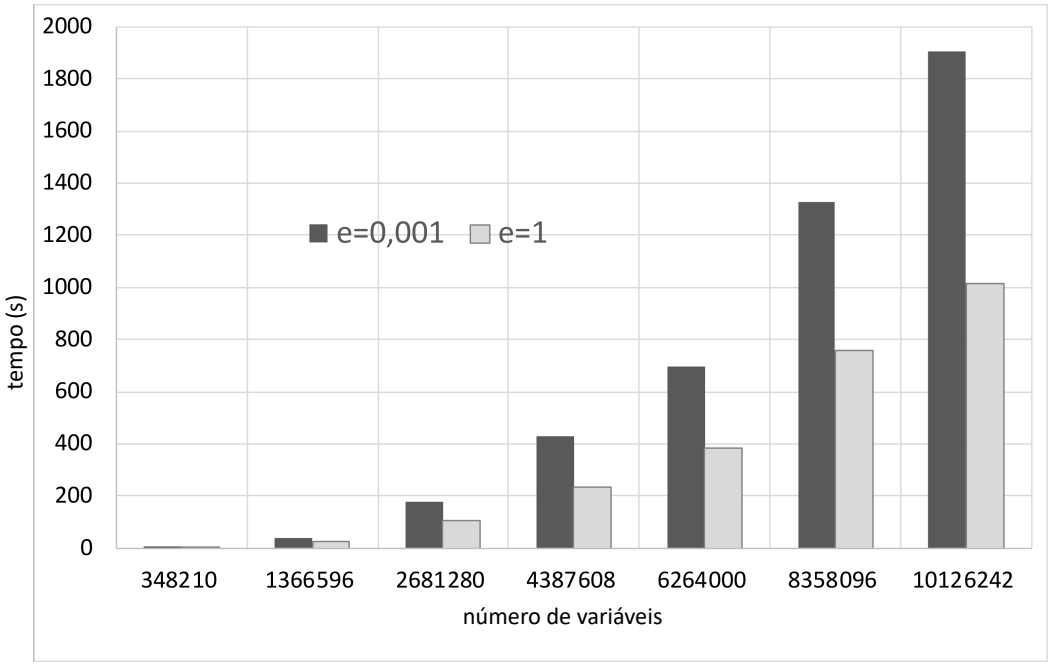

Figura 7.13: Evolução do tempo gasto para otimização das diferentes instâncias do problema.

O método dos gradientes conjugados converge em no máximo n iterações, e uma vez que cada iteração tem custo linear (a matriz Q tem na ordem de $25n$ elementos), a complexidade esperada é da ordem de $O(n^2)$. Como as curvas de tendência da Figura 7.14 mostram, esta complexidade se verifica.

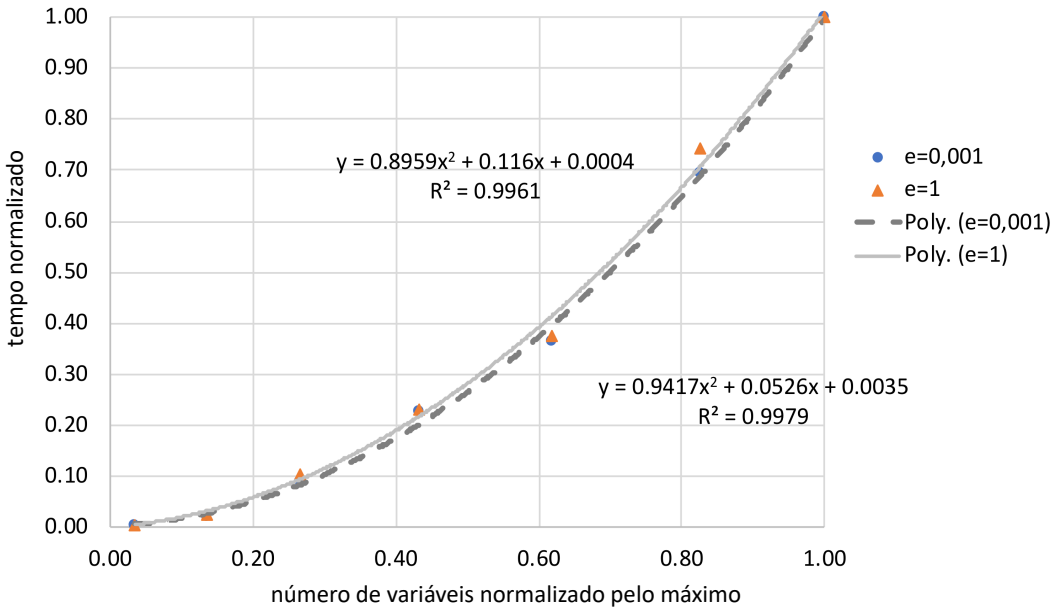

Figura 7.14: Evolução do tempo gasto para otimização das diferentes instâncias do problema.

7.7
Simulação usando malhas geradas pelo método proposto

Uma vez que as malhas resultantes do método proposto serão usadas em simulações, nesta seção o modelo gerado para o Brugge na Seção 7.2 foi utilizado para simular a subsidência e compactação do reservatório.

A malha do reservatório de entrada é composta por 15.296 e nós 43.474 elementos hexaédricos de interpolação linear. No total o modelo geomecânico é constituído por 2.130.950 nós e 2.077.488 elementos. O campo de Brugge apresenta no total 30 poços verticais (20 produtores e 10 injetores) espalhados em distintas regiões do reservatório. Como consequência da produção e dos processos de recuperação, diferentes distribuições de poropressão foram geradas através de um simulador de fluxo ao longo de 10 anos. Estas pressões foram extraídas e interpoladas nos centroides dos elementos finitos correspondentes ao reservatório. A Figura 7.15, apresenta as poropressões (kPa) em quatro instantes de tempo durante a fase de produção. Podem ser observadas duas regiões com mudanças significativas de pressão. Uma à direita da região central do reservatório em que concentram-se os poços de produção e outra à esquerda da região superior na qual concentram-se os poços de injeção. Após 10 anos de produção, na região depletada observam-se reduções de poropressão da ordem de 11000 kPa, enquanto que na região de injeção observam-se incrementos de poropressão da ordem de 4000 kPa.

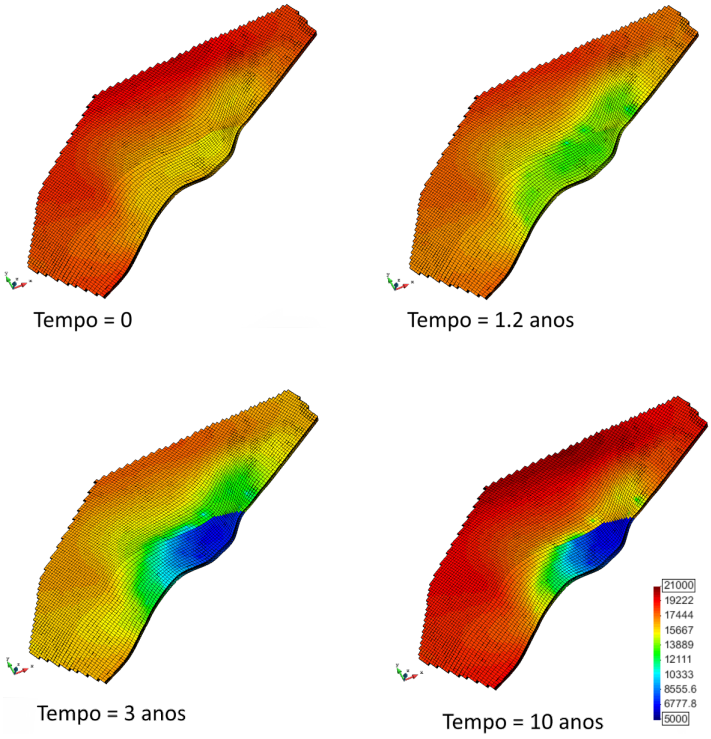

Figura 7.15: Campos de poropressões impostos em diferentes instantes de tempo durante o período de produção do campo de Brugge.

Estas poropressões foram transformadas em forças nodais equivalentes e aplicadas no modelo Geomecânico apresentado nesta tese. Este modelo foi separado em três regiões, uma região para o reservatório e outras três regiões: sobrejacente, adjacente e subjacente. As propriedades elásticas utilizadas na simulação encontram-se listadas na Tabela 7.2.

Tabela 7.2: Propriedades elásticas utilizadas no modelo Geomecânico.

Região	Módulo de Young (kPa)	Coeficiente de Poisson
Sobrejacente	138.000	0,30
Adjacente	68.000	0,25
Reservatório	68.000	0,25
Sobjacente	207.000	0,30

A simulação geomecânica foi conduzida através de um simulador de elementos finitos desenvolvido no instituto Tecgraf. A Figura 7.16 apresenta os deslocamentos verticais obtidos no topo do reservatório para cada um dos instantes de tempo indicados anteriormente. No início da simulação foram considerados deslocamentos nulos. Nos tempos posteriores, observa-se um movimento descendente que se concentra principalmente na região de maior depleção. Os deslocamentos verticais dessa região atingiram -4.53 m no topo do reservatório enquanto que na base do reservatório atingiram +2.4 m. Isto corresponde a uma compactação máxima correspondente a 2.13 m.

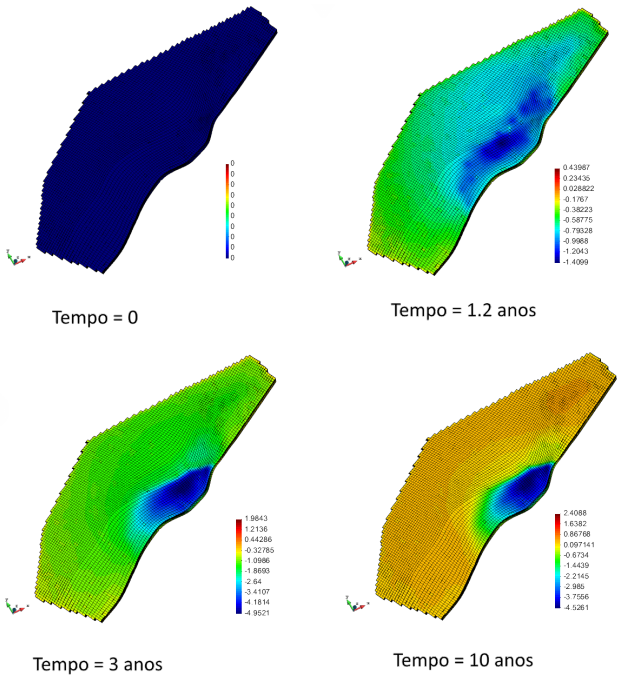

Figura 7.16: Campos de deslocamento vertical resultantes em diferentes instantes de tempo durante o período de produção do campo de Brugge.

A Figura 7.17 apresenta os deslocamentos verticais tanto no reservatório quanto no topo do modelo Geomecânico. Observam-se que a maior subsidência ocorre nas regiões acima da região com maior depleção, como era esperado. A Figura 7.18 apresenta os deslocamentos verticais obtidos ao longo das linhas A-A' e B-B' indicadas na Figura 7.17. Pode se observar que a subsidência máxima obtida após 10 anos de produção foi de 1.79 m e localizou-se numa região acima da região depletada do reservatório. Além disto, também pode se observar regiões que apresentaram um movimento ascendente devido aos processos de injeção de fluidos dentro do reservatório. Contudo, o máximo deslocamento vertical ascendente foi de 0.24 m.

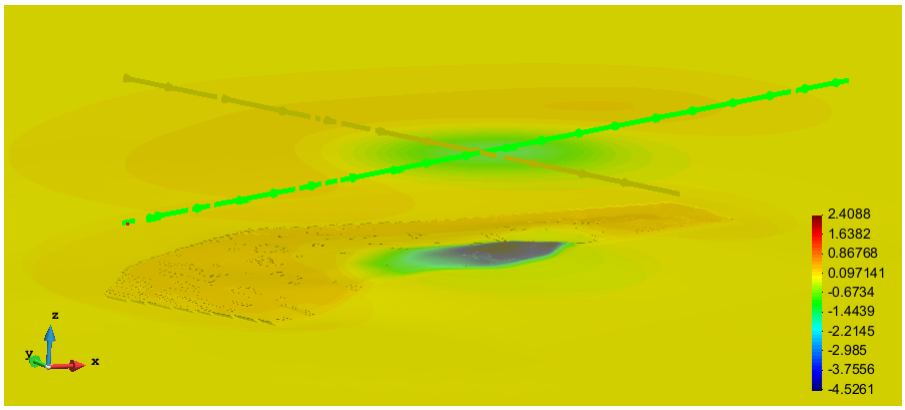

Figura 7.17: Campos de deslocamento vertical tanto no reservatório quanto no topo do modelo.

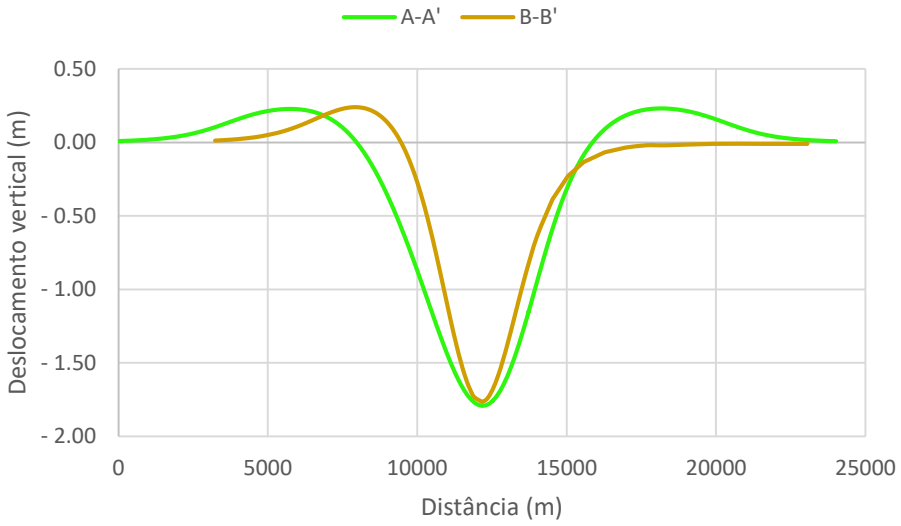

Figura 7.18: Deslocamentos verticais ao longo das linhas A-A' e B-B' localizadas no topo do modelo.

Infelizmente os resultados apresentados aqui não puderam ser comparados numericamente com os apresentados por (Florez *et al.*, 2016), uma vez que os

valores de poropressão apresentados são inconsistentes com os dados de entrada fornecidos. O autor deste trabalha supõe que os dados de entrada tenham sido multiplicado por algum fator não informado.

8
Conclusões e trabalhos futuros

Nesta tese foi apresentada uma abordagem baseada em otimização quadrática convexa para geração de malhas geomecânicas de reservatório. O modelo de otimização proposto elimina as descontinuidades do modelo de entrada e suaviza a malha resultante, ao mesmo tempo. A suavização é realizada através do operador de Laplace-Beltrami. Diferentemente das abordagens comuns em suavização de malhas, que constroem uma matriz e iteram sobre a malha de entrada até que um estado estacionário seja atingido, a abordagem por otimização permite adicionar restrições sobre o resultado da suavização, como foi o caso das restrições *soft* utilizadas para penalizar os deslocamentos das células ativas. Estas restrições poderiam ser de qualquer natureza. Quando lineares, o problema de otimização resultante é dito quadrático.

O problema de otimização proposto foi reformulado, gerando um problema de otimização equivalente, porém irrestrito. Esta reformulação, juntamente com a propriedade do problema ser estritamente convexo, permitiu que uma solução eficiente fosse projetada. Esta solução resolve um sistema de equações lineares sem precisar armazenar a matriz em memória. Para isso cada linha da matriz é reconstruída em tempo de execução e então utilizada na multiplicação necessária. Foi mostrado que a mesma abordagem pode ser aplicada para qualquer discretização do operador de Laplace-Beltrami, porém um acesso à vizinhança da malha é necessário, o que pode acarretar em um algoritmo ineficiente para malhas gerais. Em contrapartida, para o caso de malhas estruturadas, a vizinhança de um vértice pode ser acessada sempre em tempo constante e uma vez utilizada a mesma máscara do operador de Laplace-Beltrami para todos os vértices, a contribuição da função de suavização para as linhas da matriz pode ser pré-computada. Foi mostrado ainda que o montante de memória necessário para resolver tal sistema de equações é menor que 1.21 vezes o tamanho do dado de entrada. Além disso, a solução se mostrou eficiente, resolvendo modelos de milhões de variáveis em poucos minutos.

As abordagens presentes na literatura buscam uma reconstrução analítica dos modelos de entrada usando ferramenta de CAD/CAM, como B-Splines. Os células ativas do dado de entrada são usadas para construir superfícies que representam o modelo. Porém, nas regiões onde não existem células ativas as superfícies precisam

ser extrapoladas. No entanto, essas superfícies não são controladas fora da regiões que não tem pontos controle, logo, extrapolações não são confiáveis. Em contraste, a abordagem proposta é modelada sobre a discretização do dado de entrada e independe da configuração das células ativas no modelo de entrada.

Uma desvantagem da abordagem proposta é a dependência das condições de contorno. As células ativas devem ser utilizadas como guia para definir boas condições de contorno para o modelo de entrada, uma vez que a qualidade da malha de saída é diretamente proporcional à qualidade das condições de contorno. Se, por exemplo, uma camada de células tiver sua condição de contorno definida em uma altura muito diferente da sua, isso poderá provocar artefatos indesejáveis na malha para que a condição da fronteira seja satisfeita.

Esta tese também produziu uma implementação de otimização quadrática que, quando aplicada a malhas, substitui as formulações do DSI do Mallet com a vantagem de ser baseada em conceitos e notação conhecidos. Como subproduto desta implementação foi produzida uma aplicação para auxiliar na solução de problemas de otimização para modelagem geométrica em geral. Apesar desta aplicação ainda estar em desenvolvimento, ela já tem uma versão funcional e foi utilizada no desenvolvimento desta tese. Alguns exemplos estão ilustrados no Apêndice B.

Como trabalhos futuros, os autores propõem:

- um estudo de outros operadores de Laplace-Beltrami que possam ser aplicados à malhas tridimensionais.
- restrições indiretas que impeçam que o modelo resultante tenha elementos impróprios, evitando assim um passo de pós-processamento. A simples restrição de o jacobiano da célula ser positivo não é viável, uma vez que o jacobiano é uma função cúbica e não convexa dos vértices da célula.
- A finalização da aplicação de otimização, com uma possível extensão para otimização não convexa.
- A aplicação de modelos de otimização semelhantes em outros problemas, como a *restauração de modelos geológicos*, onde as camadas geológicas são modelas por superfícies e ao retroceder no tempo, as camadas são descompactadas e as superfícies envolvidas precisam se adaptar ao novo modelo e eliminar as descontinuidades de falhas que existiam, mas que ao voltar no tempo não existem mais.
- Implementar o solver em GPU e avaliar a eficiência e precisão numérica da solução.
- Implementar solver com suporte a *quad double* para melhorar a precisão numérica em modelos em caso de modelos muito grandes. Esta operação

dobraria a quantidade de memória necessário, no pior caso, mas mesmo assim o método ainda utilizaria pouca memória.

– Avaliar a possibilidade de tornar o fator de penalização adaptativo e dependente da malha de entrada.

– Formulações não convexas que levem em conta métricas diretas de qualidade dos elementos.

Referências bibliográficas

ALEXA, M.; WARDETZKY, M.. **Discrete laplacians on general polygonal meshes**. ACM Trans. Graph., 30(4):102:1–102:10, jul 2011. 3.4.3

ANGUS, D.; VERDON, J.; FISHER, Q.; KENDALL, J.; SEGURA, J.; KRISTIANSEN, T.; CROOK, A.; SKACHKOV, S.; YU, J. ; DUTKO, M.. **Integrated fluid-flow, geomechanic and seismic modelling for reservoir characterisation**. Can Soc Explor Geophys Rec, 36(4):18–27, 2011. 1, 4.3

AUGARDE, C.; RAMAGE, A. ; STAUDACHER, J.. **An element-based displacement preconditioner for linear elasticity problems**. Computers & Structures, 84(31):2306 – 2315, 2006. 6.4

SHELDON AXLER, PAUL BOURDON, W. R.. **Harmonic Function Theory**, volumen 137 de **Graduate Texts in Mathematics 137**. Springer, 2 edition, 2001. 3.3

BELL, J.. **Practical methods for estimating in situ stresses for borehole stability applications in sedimentary basins**. Journal of Petroleum Science and Engineering, 38(3):111 – 119, 2003. Borehole Stability. 4.3

BERTALMIO, M.; SAPIRO, G.; CHENG, L.-T. ; OSHER, S.. **A framework for solving surface partial differential equations for computer graphics applications**. CAM Report 00-43, UCLA, Mathematics Department, 2000. 3.4

BERTSEKAS, D.. **Convex optimization algorithms**. Athena Scientific, Belmont, Mass, 1 edition, 2015. 2, 2.1.2, 2.1.4, 2.2.2

BERTSEKAS, D.. **Nonlinear programming**. Athena Scientific, Belmont, Mass, 3 edition, 2016. 2, 2.2.3, a), a), b)

BOTSCH, M.; PAULY, M.; RÖSSL, C.; BISCHOFF, S. ; KOBBELT, L.. **Geometric modeling based on triangle meshes**. In: Magnenat-Thalmann, N.; Bühler, K., editors, EUROGRAPHICS 2006: TUTORIALS. The Eurographics Association, 2006. 1

BOTSCH, M.; PAULY, M.; ROSSL, C.; BISCHOFF, S. ; KOBBELT, L.. **Geometric modeling based on triangle meshes**. In: ACM SIGGRAPH 2006 COURSES, SIGGRAPH '06, New York, NY, USA, 2006. ACM. 1

BOTSCH, M.; KOBBELT, L.; PAULY, M.; ALLIEZ, P. ; LEVY, B.. **Polygon mesh processing**. A K Peters, Natick, Mass, 2010. (document), 3.2, 3.2, 3.3, 3.3, 3.4.2, 3.4.3, 3.4.3, 3.4, A.1

BOTELLA, A.; LÉVY, B. ; CAUMON, G.. **Indirect unstructured hex-dominant mesh generation using tetrahedra recombination**. Computational Geosciences, 20(3):437–451, Jun 2016. 4.2

BOYD, S.; VANDENBERGHE, L.. **Convex Optimization**. Cambridge University Press, New York, NY, USA, 2004. 2, 2.1.1, 2.2, 2.2.1, b), 2.2.4, 2.5

CELES, W.. **Modelagem configurável de subdivisões planares hierárquicas**. PhD thesis, Pontifical Catholic University of Rio de Janeiro, 1995. 1

CHENG, S.-W.; DEY, T. K. ; SHEWCHUK, J.. **Delaunay Mesh Generation**. Chapman & Hall/CRC, 1st edition, 2012. 3

COELHO, L. C. G.. **Modelagem de Cascas com Interseções Paramétricas**. PhD thesis, Pontifical Catholic University of Rio de Janeiro, 1998. 1

DE GENNARO, S.; SCHUTJENS, P.; FRUMAU, M.; FUERY, M.; ITA, J. ; FOKKER, P.. **The role of geomechanics in the development of an hpht field**. 01 2010. 4.1

DE SOUSA JUNIOR, L. C.; DOS SANTOS, L. O. S.; DE SOUZA RIOS, V.; ARAÚJO, C.; CELES, W. ; SEPEHRNOORI, K.. **Methodology for geomechanically controlled transmissibility through active natural fractures in reservoir simulation**. Journal of Petroleum Science and Engineering, 147:7 – 14, 2016. 4.1

DEAN, R. H.; GAI, X.; STONE, C. M.; MINKOFF, S. E. ; OTHERS. **A comparison of techniques for coupling porous flow and geomechanics**. Spe Journal, 11(01):132–140, 2006. 4.1, 4.3

DESBRUN, M.; MEYER, M.; SCHRÖDER, P. ; BARR, A. H.. **Implicit fairing of irregular meshes using diffusion and curvature flow**. In: PROCEEDINGS OF THE 26TH ANNUAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, SIGGRAPH '99, p. 317–324, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co. 3.4, 3.4.3

CARMO, M.. **Differential geometry of curves and surfaces**. Prentice-Hall, Upper Saddle River, N.J, 1976. 3.1, A

DUARTE, L. S.; CELES, W.; PEREIRA, A.; M. MENEZES, I. F. ; PAULINO, G. H.. **Polytop++: an efficient alternative for serial and parallel topology optimization on cpus & gpus**. Structural and Multidisciplinary Optimization, 52(5):845–859, Nov 2015. 6.4

FLOATER, M. S.. **Parametrization and smooth approximation of surface triangulations**. Comput. Aided Geom. Des., 14(3):231–250, apr 1997. 3.4

FLOREZ, H.; WHEELER, M.; RODRIGUEZ, A. ; PALOMINO MONTEAGUDO, J.. **Domain decomposition methods applied to coupled flow-geomechanics reservoir simulation**. In: SPE RESERVOIR SIMULATION SYMPOSIUM. Society of Petroleum Engineers, January 2011. 4.3

FLOREZ, H.; MANZANILLA-MORILLO, R.; FLOREZ, J. ; WHEELER, M. F.. **Spline-based reservoir's geometry reconstruction and mesh generation for coupled flow and mechanics simulation**. Computational Geosciences, 18(6):949–967, Dec 2014. (document), 4.1, 4.3, 4.3.1, 4.2, 4.3.1, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.9, 4.3.2, 4.10, 4.11

FLOREZ, H.; CEBERIO, M.. **A novel mesh generation algorithm for field-level coupled flow and geomechanics simulations**. In: 50TH U.S. ROCK MECHANICS/GEOMECHANICS SYMPOSIUM, número 9. American Rock Mechanics Association, Jun 2016. 4.1, 4.3.2, 4.3.2, 7.7

FU, P.; JOHNSON, S. M. ; CARRIGAN, C. R.. **An explicitly coupled hydro-geomechanical model for simulating hydraulic fracturing in arbitrary discrete fracture networks**. International Journal for Numerical and Analytical Methods in Geomechanics, 37(14):2278–2300, 2013. 4.1

GAI, X.. **A coupled geomechanics and reservoir flow model on parallel computers**. PhD thesis, The University of Texas at Austin, 2004. 4.3

GORDON, W. J.; HALL, C. A.. **Construction of curvilinear coordinate systems and applications to mesh generation**. International Journal for Numerical Methods in Engineering, 7(4):461–477, 1973. 1

GOTSMAN, C.; GU, X. ; SHEFFER, A.. **Fundamentals of spherical parameterization for 3d meshes**. ACM Trans. Graph., 22(3):358–363, jul 2003. 3.4, 3.4.2

GUROBI OPTIMIZATION, I.. **Gurobi optimizer reference manual**, 2016. B

HAJNASSER, Y.. **The implications of shale geomechanics and pressure diffusion for 4D interpretation**. PhD thesis, Institute of Petroleum Engineering, Heriot-Watt University, 2012. 4.1

KARNI, Z.; GOTSMAN, C.. **Spectral compression of mesh geometry**. In: PROCEEDINGS OF THE 27TH ANNUAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, SIGGRAPH '00, p. 279–286, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co. 3.4

KIMMEL, R.; MALLADI, R. ; SOCHEN, N.. **Image processing via the beltrami operator**. In: Chin, R.; Pong, T.-C., editors, COMPUTER VISION — ACCV'98, p. 574–581, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg. 3.4

KIM, J.; TCHELEPI, H. A.; JUANES, R. ; OTHERS. **Stability, accuracy and efficiency of sequential methods for coupled flow and geomechanics**. In: SPE RESERVOIR SIMULATION SYMPOSIUM. Society of Petroleum Engineers, Feb 2009. 4.3

KIM, J.; TCHELEPI, H. ; JUANES, R.. **Rigorous coupling of geomechanics and multiphase flow with strong capillarity**. 18, 11 2013. 4.1

KNUPP, P. M.. **A method for hexahedral mesh shape optimization**. International Journal for Numerical Methods in Engineering, 58(2):319–332, 2003. 4.2, 4.2, 4.2

KOBBELT, L.. **Discrete fairing**. In: IN PROCEEDINGS OF THE SEVENTH IMA CONFERENCE ON THE MATHEMATICS OF SURFACES, p. 101–131, 1997. 3.3

KOBBELT, L.; CAMPAGNA, S.; VORSATZ, J. ; SEIDEL, H.-P.. **Interactive multi-resolution modeling on arbitrary meshes**. In: PROCEEDINGS OF THE 25TH ANNUAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, SIGGRAPH '98, p. 105–114, New York, NY, USA, 1998. ACM. 3.4.2

KOVNATSKY, A.; BRONSTEIN, M. M. ; BRONSTEIN, A. M.. **Stable spectral mesh filtering**. In: PROCEEDINGS OF THE 12TH INTERNATIONAL CONFERENCE ON COMPUTER VISION - VOLUME PART I, ECCV'12, p. 83–91, Berlin, Heidelberg, 2012. Springer-Verlag. 3.4

LÉVY, B.; MALLET, J.-L.. **Non-distorted texture mapping for sheared triangulated meshes**. In: PROCEEDINGS OF THE 25TH ANNUAL

CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, SIGGRAPH '98, p. 343–352, New York, NY, USA, 1998. ACM. 1

LI, X.; XU, G. ; ZHANG, Y. J.. **Localized discrete laplace–beltrami operator over triangular mesh**. Computer Aided Geometric Design, 39:67 – 82, 2015. 3.4, 3.4.3

LIPMAN, Y.; SORKINE, O.; COHEN-OR, D.; LEVIN, D.; ROSSI, C. ; SEIDEL, H.-P.. **Differential coordinates for interactive mesh editing**. In: PROCEEDINGS OF THE SHAPE MODELING INTERNATIONAL 2004, SMI '04, p. 181–190, Washington, DC, USA, 2004. IEEE Computer Society. 3.4

LISEIKIN, V. D.. **Grid Generation Methods**. Springer Publishing Company, Incorporated, 2nd edition, 2009. 4.3.1

LIU, D.; XU, G. ; ZHANG, Q.. **A discrete scheme of laplace–beltrami operator and its convergence over quadrilateral meshes**. Computers & Mathematics with Applications, 55(6):1081 – 1093, 2008. 3.4.3

LONGUEMARE, P.; MAINGUY, M.; LEMONNIER, P.; ONAISI, A.; GÉRARD, C. ; KOUTSABELOULIS, N.. **Geomechanics in reservoir simulation: Overview of coupling methods and field case study**. 57:471–483, 09 2002. 4.1

LUENBERGER, D.; YE, Y.. **Linear and Nonlinear Programming**. International Series in Operations Research & Management Science. Springer US, 2016. 2

MALLET, J.-L.. **Discrete smooth interpolation**. ACM Trans. Graph., 8(2):121–144, Apr. 1989. (document), 1, B.2.1, B.1, B.2.1

MALLET, J. L.. **GOCAD: A Computer Aided Design Program for Geological Applications**, p. 123–141. Springer Netherlands, Dordrecht, 1992. 1

MALLET, J.-L.. **Geomodeling**. Oxford University Press, 2002. 1

MARTHA, L. F. C. R.. **Topological and Geometrical Modeling Approach to Numerical Discretization and Arbitrary Crack Propagation in Three-Dimensions**. PhD thesis, Cornel University, 1989. 1

MARCHINA, P.; ONAISI, A.. **Reservoir-geomechanics coupled simulations: A powerful tool for well design and operation in an hp-ht environment**. 02 2005. 4.1

MAYER, U. F.. **Numerical solutions for the surface diffusion flow in three space dimensions**. Computational and Applied Mathematics, 20(3):361–379, 2001. 3.4.3

MEYER, M.; DESBRUN, M.; SCHRÖDER, P. ; BARR, A. H.. **Discrete differential-geometry operators for triangulated 2-manifolds**. In: Hege, H.-C.; Polthier, K., editors, VISUALIZATION AND MATHEMATICS III, p. 35–57, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. 3.4.3, 3.4.3

S. MONACO, S.; CAPASSO, G.; DATYE, D.; MANTICA, S. ; VITALI, R.. **Field scale geomechanical modelling using a new automated workflow in abaqus**. 2, 01 2011. 4.1

NISHIDA, A.; SUDA, R.; HASEGAWA, H.; NAKAJIMA, K.; TAKAHASHI, D.; KOTAKEMORI, H.; KAJIYAMA, T.; NUKADA, A.; FUJII, A.; HOURAI, Y.; ZHANG, S.; ABE, K.; ITOH, S. ; SOGABE, T.. **The scalable software infrastructure for scientific computing project**. Technical report, 2009. 6.1, 6.3

PETERS, E.; CHEN, Y.; LEEUWENBURGH, O. ; OLIVER, D.. **Extended brugge benchmark case for history matching and water flooding optimization**. Computers & Geosciences, 50:16 – 24, 2013. Benchmark problems, datasets and methodologies for the computational geosciences. 4.3, 7

PIEGL, L.; TILLER, W.. **The NURBS Book (2Nd Ed.)**. Springer-Verlag New York, Inc., New York, NY, USA, 1997. 1, 4.3.1

PINKALL, U.; POLTHIER, K.. **Computing discrete minimal surfaces and their conjugates**. Experimental Mathematics, 2(1):15–36, 1993. 3.4.3, 3.4.3

PRESS, W. H.; TEUKOLSKY, S. A.; VETTERLING, W. T. ; FLANNERY, B. P.. **Numerical Recipes 3rd Edition: The Art of Scientific Computing**. Cambridge University Press, New York, NY, USA, 3 edition, 2007. 6.1, 6.1

QUEVEDO, R.. **Three-dimensional analysis of hydromechanical problems in partially saturated soils**. PhD thesis, Pontifical Catholic University of Rio de Janeiro, 2012. 4.1

RAN, L.; BOROUCHAKI, H.; BENALI, A. ; BENNIS, C.. **Hex-dominant mesh generation for subterranean formation modeling**. Engineering with Computers, 28(3):255–268, Jul 2012. 4.2

SAPIRO, G.. **Geometric Partial Differential Equations and Image Analysis**. Cambridge University Press, New York, NY, USA, 2006. 3.4

SETTARI, A.; M. MOURITS, F.. **A coupled reservoir and geomechanical simulation system**. 3:219–226, 09 1998. 4.1

SHEWCHUK, J. R.. **An introduction to the conjugate gradient method without the agonizing pain**. Technical report, Pittsburgh, PA, USA, 1994. 6.1

SHENG, D.; SLOAN, S. W.; GENS, A. ; SMITH, D. W.. **Finite element formulation and algorithms for unsaturated soils. part i: Theory**. International Journal for Numerical and Analytical Methods in Geomechanics, 27(9):745–765, 2003. 4.1

SORKINE, O.; COHEN-OR, D. ; TOLEDO, S.. **High-pass quantization for mesh encoding**. In: PROCEEDINGS OF THE 2003 EUROGRAPHICS/ACM SIGGRAPH SYMPOSIUM ON GEOMETRY PROCESSING, SGP '03, p. 42–51, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association. 3.4

SORKINE, O.; COHEN-OR, D.; LIPMAN, Y.; ALEXA, M.; RÖSSL, C. ; SEIDEL, H.-P.. **Laplacian surface editing**. In: PROCEEDINGS OF THE 2004 EUROGRAPHICS/ACM SIGGRAPH SYMPOSIUM ON GEOMETRY PROCESSING, SGP '04, p. 175–184, New York, NY, USA, 2004. ACM. 3.4

TAUBIN, G.. **A signal processing approach to fair surface design**. In: PROCEEDINGS OF THE 22ND ANNUAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, SIGGRAPH '95, p. 351–358, New York, NY, USA, 1995. ACM. 3.4, 3.4.2, 3.4.2, 3.4.3

VALLET, B.; LEVY, B.. **Spectral geometry processing with manifold harmonics**. Computer Graphics Forum, 2008. 3.4

WARDETZKY, M.; MATHUR, S.; KÄLBERER, F. ; GRINSPUN, E.. **Discrete laplace operators: No free lunch**. In: PROCEEDINGS OF THE FIFTH EUROGRAPHICS SYMPOSIUM ON GEOMETRY PROCESSING, SGP '07, p. 33–37, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association. 3.4.1, 3.4.1

WATHEN, A.. **An analysis of some element-by-element techniques**. Computer Methods in Applied Mechanics and Engineering, 74(3):271 – 287, 1989. 6.4

WEICKERT, J.. **Anisotropic diffusion in image processing**, volumen 1. Teubner Stuttgart, 1998. 3.4

WELCH, W.; WITKIN, A.. **Variational surface modeling**. SIGGRAPH Comput. Graph., 26(2):157–166, jul 1992. 3.3

XIONG, Y.; LI, G. ; HAN, G.. **Mean Laplace–Beltrami Operator for Quadrilateral Meshes**, p. 189–201. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. 3.4.3

XU, G.. **Discrete laplace–beltrami operators and their convergence**. Computer Aided Geometric Design, 21(8):767 – 784, 2004. Geometric Modeling and Processing 2004. 3.4.3

ZHANG, H.; FIUME, E.. **Butterworth filtering and implicit fairing of irregular meshes**. In: 11TH PACIFIC CONFERENCE ON COMPUTER GRAPHICS AND APPLICATIONS, 2003. PROCEEDINGS., p. 502–506, Oct 2003. 3.4

ZHANG, H.. **Discrete combinatorial laplacian operators for digital geometry processing**. In: SIAM CONFERENCE ON GEOMETRIC DESIGN, 2004, p. 575–592. Press, 2004. 3.4.2, 3.4.2

ZHOU, K.; HUANG, J.; SNYDER, J.; LIU, X.; BAO, H.; GUO, B. ; SHUM, H.-Y.. **Large mesh deformation using the volumetric graph laplacian**. ACM Trans. Graph., 24(3):496–503, jul 2005. 3.4

A
Geometria Diferencial

Este capítulo apresenta, de forma superficial, conceitos de geometria diferencial que serviram de base para o desenvolvimento da tese. Um estudo aprofundado sobre o assunto pode ser encontrado em (Do Carmo, 1976).

A.1
Propriedades métricas das superfícies

Tome uma superfície contínua $S \subset \mathbb{R}^3$ dada na forma paramétrica

$$\mathbf{x}(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix}, \quad (u,v) \in \Omega \subset \mathbb{R}^2 \tag{A-1}$$

onde x, y e z são funções diferenciáveis em u e v, e Ω é o domínio dos parâmetros. Os escalares (u,v) são coordenadas no espaço paramétrico.

As duas derivadas parciais

$$\mathbf{X}_u(u_0,v_0) = \frac{\partial \mathbf{x}}{\partial u}(u_0,v_0) \qquad \text{e} \qquad \mathbf{X}_v(u_0,v_0) = \frac{\partial \mathbf{x}}{\partial v}(u_0,v_0) \tag{A-2}$$

são, respectivamente, os vetores tangentes das duas curvas paramétricas

$$\mathbf{C}_u(t) = \mathbf{x}(u_0+t, v_0) \qquad \text{e} \qquad \mathbf{C}_v(t) = \mathbf{x}(u_0, v_0+t) \tag{A-3}$$

no ponto $\mathbf{x}(u_0,v_0) \in S$.

Se a parametrização for regular, ou seja, $\mathbf{x}_u \times \mathbf{x}_v \neq 0$, o plano tangente de S é gerado pelos dois vetores tangentes $\mathbf{x}_u$ e $\mathbf{x}_v$. O vetor normal à superfície é ortogonal a ambos os vetores e dado por

$$\mathbf{n} = \frac{\mathbf{x}_u \times \mathbf{x}_v}{||\mathbf{x}_u \times \mathbf{x}_v||} \tag{A-4}$$

Dado um vetor direção $\bar{\mathbf{w}} = (u_p, v_p)^T$ definido no espaço dos parâmetros, considere uma reta parametrizada por t, passando por (u_0,v_0) e orientada por $\bar{\mathbf{w}}$ dada por $(u,v) = (u_0,v_0) + t\bar{\mathbf{w}}$. A imagem dessa reta em $\mathbf{x}$ (Fig. A.1) é a curva

dada por

$$\mathbf{C_w}(t) = \mathbf{x}(u_0 + tu_p, v_0 + tv_p) \tag{A-5}$$

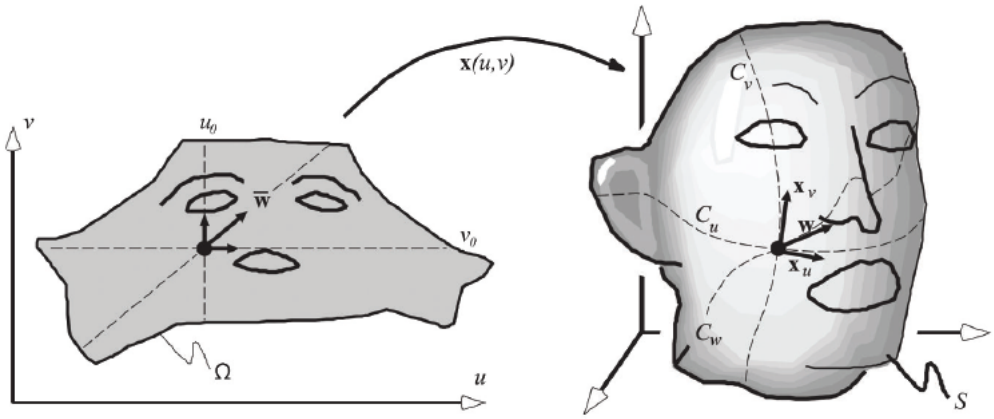

Figura A.1: Transformação de vetores do espaço paramétrico para vetores tangentes na superfície. Fonte: (Botsch *et al.*, 2010)

A derivada direcional $\mathbf{w}$ de $\mathbf{x}$ em (u_0, v_0) relativa à derivada direcional $\bar{\mathbf{w}}$ é definida como sendo a tangente de $\mathbf{C_w}$ em $t = 0$, dada por

$$\begin{aligned} \mathbf{w} &= \frac{\partial \mathbf{C_w}(t)}{\partial t} \\ &= \frac{\partial \mathbf{x}}{\partial u}\frac{\partial u}{\partial t} + \frac{\partial \mathbf{x}}{\partial v}\frac{\partial v}{\partial t} \\ &= \mathbf{x}_u u_p + \mathbf{x}_v v_p \\ &= \begin{bmatrix} \mathbf{x}_u & \mathbf{x}_v \end{bmatrix} (u_p, v_p) \end{aligned} \tag{A-6}$$

Logo,

$$\mathbf{w} = \mathbf{J}\bar{\mathbf{w}} \tag{A-7}$$

Onde,

$$\mathbf{J} = \begin{bmatrix} \frac{\partial x}{\partial u} & \frac{\partial x}{\partial v} \\ \frac{\partial y}{\partial u} & \frac{\partial y}{\partial v} \\ \frac{\partial z}{\partial u} & \frac{\partial z}{\partial v} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_u & \mathbf{x}_v \end{bmatrix} \tag{A-8}$$

é a *matrix jacobiana* de $\mathbf{x}$.

A matriz jacobiana de $\mathbf{x}$ corresponde a um mapeamento linear que transforma um vetor $\bar{\mathbf{w}}$ no espaço dos parâmetros, em um vetor tangente $\mathbf{w}$ na superficíe. De modo geral, a matriz jacobiana codifica a métrica da superfície permitindo medir como ângulos, distâncias e áreas são transformadas pelo mapeamento do domínio

dos parâmetros para a superfície.

A.1.1 Primeira Forma Fundamental

Tomando dois vetores diretores $\bar{\mathbf{w}}_1$ e $\bar{\mathbf{w}}_2$ unitários no espaço dos parâmetros, o cosseno do ângulo entre estes dois vetores é dado por $\bar{\mathbf{w}}_1^T\bar{\mathbf{w}}_2$. O produto escalar entre os vetores tangentes correspondentes na superfície é então dado por

$$\mathbf{w}_1^T\mathbf{w}_2 = (\mathbf{J}\bar{\mathbf{w}}_1)^T(\mathbf{J}\bar{\mathbf{w}}_2) = \bar{\mathbf{w}}_1^T(\mathbf{J}^T\mathbf{J})\bar{\mathbf{w}}_2. \tag{A-9}$$

O produto de matrizes $\mathbf{J}^T\mathbf{J}$ é conhecido como **primeira forma fundamental** de $\mathbf{x}$ e pode ser escrito como

$$\mathbf{I} = \mathbf{J}^T\mathbf{J} = \begin{bmatrix} \mathbf{x}_u^T\mathbf{x}_u & \mathbf{x}_u^T\mathbf{x}_v \\ \mathbf{x}_u^T\mathbf{x}_v & \mathbf{x}_v^T\mathbf{x}_v \end{bmatrix} = \begin{bmatrix} E & F \\ F & G \end{bmatrix} \tag{A-10}$$

Desta forma, o produto escalar entre os vetores tangentes na superfície pode ser também escrito da seguinte forma:

$$\begin{aligned} \mathbf{w}_1^T\mathbf{w}_2 &= \bar{\mathbf{w}}_1^T\mathbf{I}\bar{\mathbf{w}}_2 \\ &= \begin{bmatrix} u_p & v_p \end{bmatrix} \begin{bmatrix} E & F \\ F & G \end{bmatrix} \begin{bmatrix} u_p \\ v_p \end{bmatrix} \\ &= Eu_p^2 + 2Fu_pv_p + Gv_p^2 \end{aligned} \tag{A-11}$$

Comprimento de Arco

A primeira forma fundamental $\mathbf{I}$ define um produto interno no espaço tangente de S. Além de medir ângulos, este produto interno pode ser usado para medir a norma ao quadrado do vetor $\mathbf{w}$ como

$$||\mathbf{w}||^2 = \bar{\mathbf{w}}^T\mathbf{I}\bar{\mathbf{w}}. \tag{A-12}$$

Desta forma, podemos medir o comprimento de uma curva $\mathbf{x}(t) = \mathbf{x}(\mathbf{u}(t))$, definida como a imagem de uma curva regular $\mathbf{u}(t) = (u(t), v(t)) = (u_0 + tu_p, v_0 + tv_p)$ no espaço dos parâmetros. O vetor tangente da curva, como calculado na Equação A-6, é dado por

$$\frac{\partial\mathbf{x}(\mathbf{u}(t))}{\partial t} = \frac{\partial\mathbf{x}}{\partial u}\frac{\partial u}{\partial t} + \frac{\partial\mathbf{x}}{\partial v}\frac{\partial v}{\partial t} = \mathbf{x}_u u_p + \mathbf{x}_v v_p. \tag{A-13}$$

Sendo **w** o vetor tangente à curva na superfície S, o comprimento $l(a,b)$ de $\mathbf{x}(\mathbf{u}(t))$ para um intervalo $[a,b]$ no espaço dos parâmetros é dado por

$$\begin{aligned} l(a,b) &= \int_a^b ||u't()||dt \\ &= \int_a^b \sqrt{< u'(t), u'(t) >}dt \\ &= \int_a^b \sqrt{\mathbf{w}^T\mathbf{w}}dt \\ &= \int_a^b \sqrt{(u_p, v_p)\mathbf{I}(u_p, v_p)^T}dt \\ &= \int_a^b \sqrt{Eu_p^2 + 2Fu_pv_p + Gv_p^2}dt \end{aligned} \tag{A-14}$$

Uma vez que a primeira forma fundamental é usada para medir comprimento de arco, não surpreende que ela esteja intimamente ligada com isometria de superfícies. Isometrias são homeomorfismos de uma superfície S para outra superfície S' que mapeia curvas em S para curvas em S' com mesmo comprimento.

Teorema A.1 *Um patch de coordenadas das superfícies S e S' são isométricas, se e somente se, existem parametrizações* ***r*** $: V \to \mathbb{R}^3$ *e* ***r'*** $: V \to \mathbb{R}^3$ *de S e S', respectivamente, com a mesma primeira forma fundamental.*

Área da Superfície

A área A de uma superfície parametrizada correspondente à uma certa região $U \subseteq \Omega$ é dada por

$$A = \int\int_U ||\mathbf{x}_u \times \mathbf{x}_v||dudv \tag{A-15}$$

Pode-se desenvolver o termo $||\mathbf{x}_u \times \mathbf{x}_v||$ para que este possa ser escrito em termos da primeira forma fundamental da seguinte maneira:

$$
\begin{aligned}
||\mathbf{x}_u \times \mathbf{x}_v|| &= ||\mathbf{x}_u||||\mathbf{x}_v|| \sin\theta \\
||\mathbf{x}_u \times \mathbf{x}_v||^2 &= ||\mathbf{x}_u||^2||\mathbf{x}_v||^2 \sin^2\theta \\
&= ||\mathbf{x}_u||^2||\mathbf{x}_v||^2(1-\cos^2\theta) \\
&= ||\mathbf{x}_u||^2||\mathbf{x}_v||^2 - ||\mathbf{x}_u||^2||\mathbf{x}_v||^2\cos^2\theta \\
&= ||\mathbf{x}_u||^2||\mathbf{x}_v||^2 - (||\mathbf{x}_u||||\mathbf{x}_v||\cos\theta)^2 \\
&= < \mathbf{x}_u, \mathbf{x}_u >< \mathbf{x}_v, \mathbf{x}_v > - < \mathbf{x}_u, \mathbf{x}_v >^2 \\
&= EG - F^2 = \det(\mathbf{I}) \\
||\mathbf{x}_u \times \mathbf{x}_v|| &= \sqrt{\det(\mathbf{I})}
\end{aligned}
\tag{A-16}
$$

Logo, a área da superfície pode ser definida como

$$
A = \int\int_U \sqrt{\det(\mathbf{I})} du dv \tag{A-17}
$$

B
Aplicação para otimização geométrica

Um subproduto desta tese é uma aplicação que resolve modelos de otimização quadrática com foco em problemas geométricos. Esta aplicação foi pensada para ter uma interface simples e clara, de forma que o usuário consiga escrever modelos de otimização de forma rápida. Além disso, a aplicação oferece um gerenciador robusto do problema de otimização (ou modelo de otimização) e provê pré-processamentos eficientes a fim de reduzir a dimensão do problema de entrada e verificar a viabilidade do mesmo. A aplicação é capaz de decidir, dentre os *solvers* que possui, qual é o mais adequado para resolver o problema em questão, levando em conta questões de eficiência e memória disponível. Desta forma, o objetivo é sempre buscar a solução mais eficiente possível. O usuário é capaz de definir qual *solver* deve ser usado, assim como estender a aplicação implementando seus próprios *solvers*. A aplicação implementada é inspirada em *software* de otimização comerciais, como IBM CPLEX e GUROBI (Gurobi, 2016). Alguns deles fornecem suporte à otimização quadrática. No entanto, o foco principal são problemas de otimização lineares inteiros.

No momento, a aplicação dá suporte a problemas de otimização quadrática sem uma solução eficiente para restrições de desigualdade. Mesmo assim, foi de grande utilidade para o desenvolvimento deste trabalho, uma vez que ela foi extensamente utilizada no estudo da solução proposta nesta tese.

B.1
Entidades da aplicação da otimização

Na aplicação de otimização implementada, existem quatro entidades que o usuário precisa manipular:

- Modelo: é o que define e gerencia todo o problema de otimização. Todas as informações a respeito do problema devem ser consultadas ou definidas nele.
- Variável: é a representação de tudo que é incógnita para o problema. Uma variável tem um nome, valor, *lower bound* e *upper bound* que podem ser definidos pelo usuário.
- Restrições: impõe o conjunto de restrições sobre quais valores as variáveis podem assumir. Podem ser do tipo igual ($=$), menor/maior ou igual ($\leq$, $\geq$), ou

aproximadamente igual ($\simeq$). O último tipo de restrição é satisfeito no sentido de mínimos quadrados.

– Função objetivo: define qual função objetivo o modelo deve minimizar.

Um dos principais objetivos da aplicação é permitir que o usuário defina este problema de otimização de forma simples. Isso quer dizer que ele deve ser capaz de rapidamente definir qual é a função objetivo e quais são as restrições do problema. Para isso, existem classes de apoio, como manipuladores de expressões quadráticas e lineares que são usadas para construir a função objetivo e as restrições. Suponha que se queira encontrar o ponto mínimo do seguinte paraboloide elíptico:

$$f(x, y) = 2x^2 + y^2 - 2xy + 4x - 2y + 6$$

O Algoritmo 2 traz uma implementação em C++ para a solução deste problema e sua saída é dada em seguida.

***Algoritmo* 2** Exemplo simples do uso da aplicação de otimização.

```
1  //Instancia modelo.
2  Model *m = new OptimizationModel( );
3
4  //Adiciona variaveis ao problema de otimizacao.
5  Variable x = m->addVariable( "x" );
6  Variable y = m->addVariable( "y" );
7
8  //Constroi funcao objetivo e a define no modelo.
9  QuadraticExpression obj = 2*x*x + y*y - 2*x*y + 4*x - 2*y + 6;
10 m->setObjectiveFunction( obj );
11
12 //Adiciona restricao 2x -3y = 5 ao problema de otimizacao.
13 m->addConstraint( 2 * x - 3 * y, EQUAL, 5 );
14
15 //Imprime e otimiza modelo.
16 cout << *m << endl;
17
18 m->optimize( );
19 //Imprime valor das variaveis.
20 printf( "\nx = %.3lf\ny = %.3lf\n", x.getValue( ), y.getValue( ) );
21
22 //Imprime valor da funcao.
23 printf( "\nMinimo: %.3lf\n", obj.getValue( ) );
24 delete m;
```

O tipo *Model* centraliza toda interação do usuário com o problema de otimização. Invariavelmente o usuário vai começar por criar um objeto do tipo *Model*. Após isso, deve-se solicitar ao modelo a criação de variáveis e então, o problema de otimização deve ser definido sobre elas. O tipo *Variable* foi implementado de forma que o usuário possa tratá-lo como outra variável primitiva qualquer e para isso todo o conjunto de operadores aritméticos foi sobrecarregado.

***Algoritmo* 3** Saída produzida pelo Algoritmo 2.

```
Minimize + 4 * x - 2 * y + 6 + [ + 4 x * x  + 2 y * y
- 4 x * y ] / 2
Subject to:
        C1: + 2 * x - 3 * y = + 5
x = -1.700
y = -2.800
Minimo: 8.900
```

Na prática, o tipo *Variable* armazena apenas um ponteiro para um objeto interno que representa de fato as informações da variável. Isso traz os seguintes benefícios:

- O custo de cópia é igual à cópia de uma variável simples.
- Todas as cópias de uma mesma variável definem as informações da mesma variável, evitando duplicidades. Assim sendo, quando o *solver* define o valor de uma variável, o usuário pode consultar o valor corrente da variável em sua cópia.
- Operadores podem ser sobrecarregados sobre as variáveis e mesmo que elas apareçam em múltiplas expressões diferentes, o modelo saberá identificar a qual variável do modelo uma dada variável de uma expressão faz referência.

Estas propriedades são ilustradas no Algoritmo 4.

***Algoritmo* 4** Modelo é capaz de entender que variáveis são equivalentes.

```
//Instancia um solver e cria um modelo.
LBFGSOptimizationSolver solver;
OptimizationModel m( &solver );

//Adiciona variaveis ao problema de otimizacao.
Variable x = m.addVariable( "x" );
Variable y = m.addVariable( "y" );

//Cria uma copia da variavel x.
Variable z = x;

//Define uma solucao inicial.
x.setValue( 1.0 ); y.setValue( 0.5 );
//Muda o valor da variavel x tambem.
z.setValue( -1.0 );

//O modelo sabe que x = z e define a restricao 4x = 5.
m.addConstraint(7 * x - 3 * z, EQUAL, 5);

//Modelo entendera a fun. objetivo como 4x^2 + y^2 + 4x - 2y + 6
m.setObjectiveFunction( 2*x*x + y*y + 2*x* z + 4*x - 2*y + 6 );

m.optimize();
cout << m << endl;
printf( "\nx = %.3lf\ny = %.3lf\n", x.getValue( ), y.getValue( ) );
```

***Algoritmo* 5** Saída produzida pelo Algoritmo 4.

```
1 Minimize + 4 * x - 2 * y + 6 + [ + 8 x * x  + 2 y * y ] / 2
2 Subject to:
3         C1:    + 4 * x = + 5
4 x = 1.250
5 y = 1.000
6 Minimo: 16.250
```

B.2
Exemplos usando a aplicação de otimização

Para ilustrar a API da aplicação de otimização desenvolvida, nesta seção alguns exemplos serão ilustrados.

B.2.1
Superfície com falhas

Em um primeiro exemplo, considere o exemplo ilustrado na Figura B.1 proposto no trabalho em que Mallet (Mallet, 1989) propõe o método DSI (*Discrete Smooth Interpolation*). Neste exemplo, uma superfície, com duas falhas, é reconstruída a partir de 27 pontos dados como entrada. Para isso foi utilizado um grid regular de 35×26 pontos. Em seu paper, Mallet também utiliza o operador Kirchhoff para reconstruir uma malha suave.

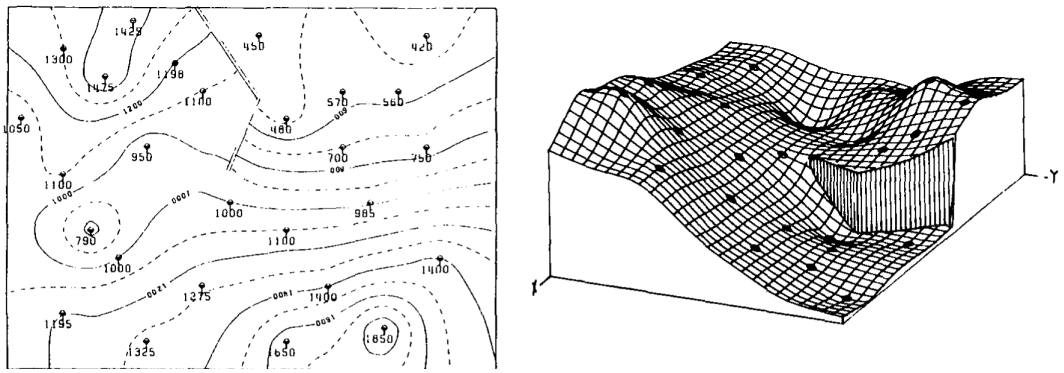

Figura B.1: À esquerda, as curvas de nível do exemplo modelado. À direita, a superfície gerada pelo método DSI. A superfície foi reconstruída a partir de vinte e sete pontos conhecidos. Fonte: (Mallet, 1989).

Para reproduzir o exemplo, foi adaptado um grid regular de dimensão 35×26 sobre os pontos dados na curva de nível. Os 27 pontos de entrada tiveram suas alturas fixadas e o operador Kirchhoff foi aplicado para todos os vértices. No entanto, este operador deve respeitar as falhas, ou seja, dois vértices em lados opostos de uma falha não são considerados vizinhos. Assim como no paper (Mallet, 1989), apenas as alturas dos pontos do grid são ajustadas. Para a implementação na aplicação de otimização serão assumidas que as estruturas apresentadas no

Algoritmo 6 já estão inicializados, de forma que os pontos conhecidos na grade tem o valor definido, assim como a *flag know* definida como *true*.

***Algoritmo* 6** Estruturas assumidas para reprodução do exemplo proposto por Mallet.

```
1  struct Point
2  {
3      double z;
4      bool known;
5  };
6
7  //Grid de pontos e variavies.
8  std::vector< std::vector< Point    > > grid;
9  std::vector< std::vector< Variable > > v;
```

O Algoritmo 7 aloca as variáveis necessárias e adiciona as restrições para os vértices que tem altura conhecida na malha.

***Algoritmo* 7** Gera variáveis no modelo e adiciona restrições para os valores conhecidos na superfície.

```
1  ConjugateGradientLISOptimizationSolver s;
2  OptimizationModel m( &s );
3  int ni = 35, nj = 26;
4  for ( int i = 0; i < ni; i++ )
5  {
6      for ( int j = 0; j < nj; j++ )
7      {
8          //Solicta ao modelo a criacao de uma variavel e armazena
9          //a copia retornada.
10         v[i][j] = m.addVariable( );
11         if ( grid[i][j].known )
12         {
13             m.addConstraint( v[i][j], EQUAL, grid[i][j].z );
14         }
15     }
16 }
```

O próximo passo é calcular a função objetivo usando as variáveis alocadas, como ilustrado no Algoritmo 8.

***Algoritmo* 8** Calcula função objetivo usando as variáveis retornadas pelo modelo.

```
1  QuadraticExpression& obj = m.getObjectiveFunction( );
2  for ( int i = 0; i < ni; i++ )
3  {
4      for ( int j = 0; j < nj; j++ )
5      {
6          LinearExpression l;
7          //Verifica se os vertices de indice (i,j) e (i-1, j) estao
8          //dentro do grid e nao estao separados  por uma falha.
9          if ( isNeighbour( i, j, i - 1, j ) )
10             l += v[i - 1][j];
```

```
        if ( isNeighbour( i, j, i + 1, j ) )
            l += v[i + 1][j];

        if ( isNeighbour( i, j, i, j - 1 ) )
            l += v[i][j - 1];

        if ( isNeighbour( i, j, i, j + 1 ) )
            l += v[i][j + 1];

        //Adiciona o vertice central ponderado
        //pelo numero de vizinhos validos.
        l -= l.size( ) * v[i][j];

        //Soma o termo quadratico à funcao objetivo.
        obj += l * l;
    }
}
```

A função *isNeighbour* verifica se dois vértices pertencem à mesma máscara do laplaciano. Uma vez que as falhas devem ser modelas, dois vértices vizinhos na grade não são vizinhos na máscara se uma falha os separa. Desta forma, a função *isNeighbour* retorna *true* sempre que não há falhas separando os dois vértices em questão e ambos pertencem à grade.

Uma vez definidas as restrições e a função objetivo, basta otimizar o modelo e obter a solução, como ilustrado no Algoritmo 9.

***Algoritmo* 9** Otimiza o modelo e obtém a solução.

```
//Otimiza o modelo.
m.optimize( );

//Copia a solucao.
for ( int i = 0; i < ni; i++ )
{
    for ( int j = 0; j < nj; j++ )
    {
        //Obtem valor atribuido a variavel.
        grid[i][j].z = v[i][j].getValue( );
    }
}
```

Ao otimizar o modelo, o *solver* atualiza o valor corrente de cada variável, logo a solução pode ser obtida diretamente da cópia das variáveis armazenadas pelo usuário.

A Figura B.2 ilustra a grade de entrada utilizada para reproduzir o exemplo antes e depois do processo de otimização. Os vértices em vermelhos são os vértices com altura conhecida. A Figura B.3 ilustra a malha triangular resultante. Em todas as figuras os triângulos que atravessam uma falha foram removidos. Em seu paper Mallet não cita quais as ordens de gradeza das coordenadas na dimensões x e y. As

imagens apresentadas foram geradas multiplicando os índices i e j por um fator de 150.0.

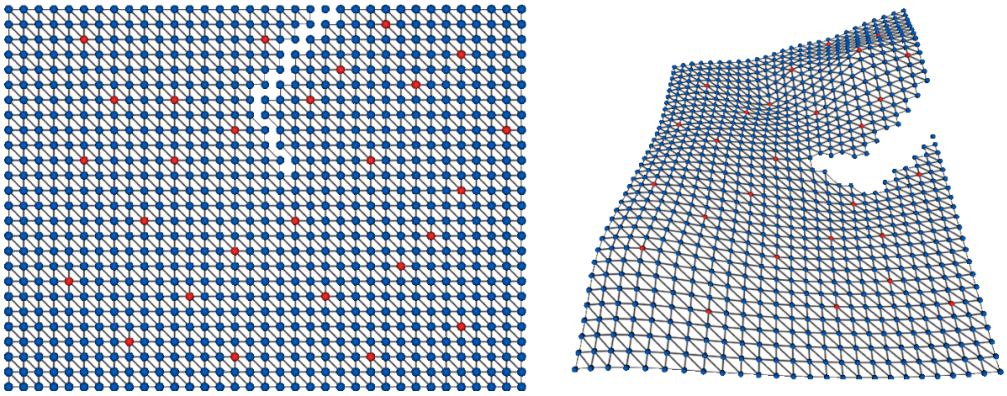

Figura B.2: Imagem à esquerda ilustra a grade de entrada com os pontos conhecidos destacados em vermelho. À direita, a grade resultante após o cálculo da altura de cada nó.

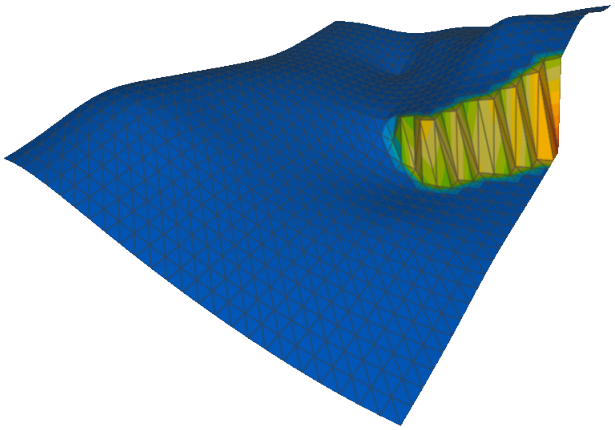

Figura B.3: Duas vistas da malha triangular gerada para o exemplo. Os triângulos que cortam as falhas foram removidos da malha final.

B.2.2 Suavização de uma grade bidimensional

O primeiro problema modelado nesta aplicação de otimização foi a suavização de uma grade bidimensional. Essa grade foi obtida de um reservatório e os vértices correspondentes à células ativas são fixados. Foi este problema que deu início ao trabalho desta tese.

A Figura B.4 ilustra o dado de entrada. Os vértices em verde pertencem à células ativas do reservatório e tem posição fixa. Os vértices em vermelho fazem parte da condição de contorno.

Para a implementação na aplicação de otimização serão assumidas que as estruturas apresentadas no Algoritmo 10 já estão inicializadas, de forma que os

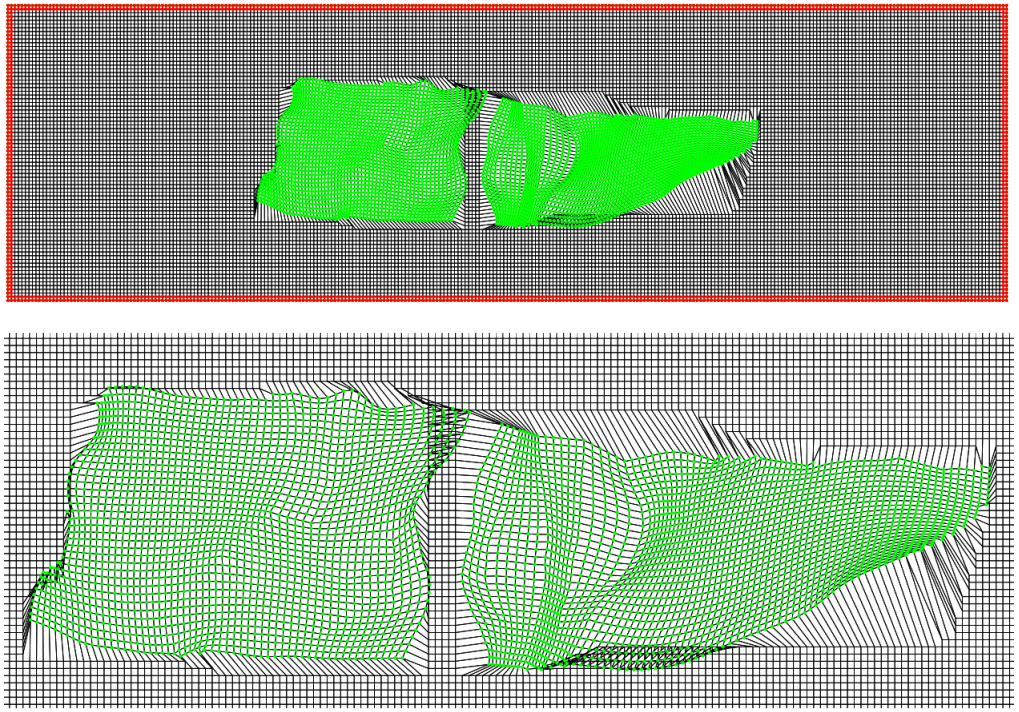

Figura B.4: Dado de entrada. Em verde os pontos fixos nas células ativas e em vermelho os pontos fixos pelas condições de contorno.

pontos conhecidos na grade tem o valor definido, assim como a *flag know* definida como *true*.

***Algoritmo* 10** Estruturas assumidas para implementação da suavização da grade bidimensional.

```
1  struct Point
2  {
3      double x, y;
4      bool known;
5  };
6
7  //Uma variavel para cada dimensao.
8  struct VarPoint
9  {
10     Variable x, y;
11 };
12
13 //Grid de pontos e variavies.
14 std::vector< std::vector< Point    > > grid;
15 std::vector< std::vector< VarPoint > > v;
16
17 //Cria modelo.
18 OptimizationModel m;
```

O Algoritmo 11 cria todas as variáveis e define um valor inicial para cada uma delas. Além disso, os vértices que fazem parte de células ativas são fixados.

***Algoritmo* 11** Alocação de variáveis; definição de solução inicial; restrições para vértices em células ativas.

```
for (unsigned int j = 0; j < nj; j++)
{
    for (unsigned int i = 0; i < ni; i++)
    {
        //Adiciona variaveis ao modelo.
        v[i][j].x = m.addVariable( );
        v[i][j].y = m.addVariable( );

        //Define uma solucao inicial.
        v[i][j].x = grid[i][j].x;
        v[i][j].y = grid[i][j].y;

        //Adiciona restricao para os pontos que estao em
        //celulas ativas.
        if(grid[i][j].known)
        {
            m.addConstraint( v[i][j].x, EQUAL, grid[i][j].x );
            m.addConstraint( v[i][j].y, EQUAL, grid[i][j].y );
        }
    }
}
```

Após criação das variáveis, o Algoritmo 12 mostra a criação da função objetivo. Mais uma vez a função objetivo minimiza o operador de Laplace-Beltrami para cada vértice utilizando a discretização de Kirchhoff.

***Algoritmo* 12** Construção da função objetivo para minimizar o operador de Laplace-Beltrami numa grade bidimensional.

```
//Calcula funcao objetivo
QuadraticExpression &obj = m.getObjectiveFunction( );
for (int j = 0; j < nj; j++)
{
    for (int i = 0; i < ni; i++)
    {
        LinearExpression x, y;
        if (i - 1 >= 0)
        {
            x += v[i - 1][j + 0].x;
            y += v[i - 1][j + 0].y;
        }
        if (i + 1 < ni)
        {
            x += v[i + 1][j + 0].x;
            y += v[i + 1][j + 0].y;
        }

        if (j - 1 >= 0)
        {
            x += v[i + 0][j - 1].x;
            y += v[i + 0][j - 1].y;
        }
```

```
        if (j + 1 < nj)
        {
            x += v[i + 0][j + 1].x;
            y += v[i + 0][j + 1].y;
        }

        x -= x.size() * v[i][j].x;
        y -= y.size() * v[i][j].y;

        //Adiciona termos quadraticos na funcao objetivo.
        obj += x * x + y * y;
    }
}
```

Um último passo da construção do modelo de otimização é a definição das condições de fronteira, como ilustrado no Algoritmo 13.

***Algoritmo* 13** Definição das condições de contorno do grid.

```
//Adiciona restricoes para fixar uma celula na borda.
for (int j = 0; j < nj; j++)
{
    for (int i = 0; i < ni; i++)
    {
        if (i > 1 && i < ni - 2 &&
            j > 1 && j < nj - 2)
        {
            continue;
        }
        m.addConstraint( v[i][j].x, EQUAL, grid[i][j].x );
        m.addConstraint( v[i][j].y, EQUAL, grid[i][j].y );
    }
}
```

Uma vez construído o modelo de otimização, basta otimizá-lo e obter a solução a partir das variáveis, como ilustrado no Algoritmo 14.

***Algoritmo* 14** Otimização e recuperação da solução.

```
//Otimiza o modelo.
m.optimize();

//Copia a solucao do modelo.
for (int j = 0; j < nj; j++)
{
    for (int i = 0; i < ni; i++)
    {
        grid[i][j].x = (double) v[i][j].x;
        grid[i][j].y = (double) v[i][j].y;
    }
}
```

A Figura B.5 ilustra o resultado obtido pela suavização da grade bidimensional.

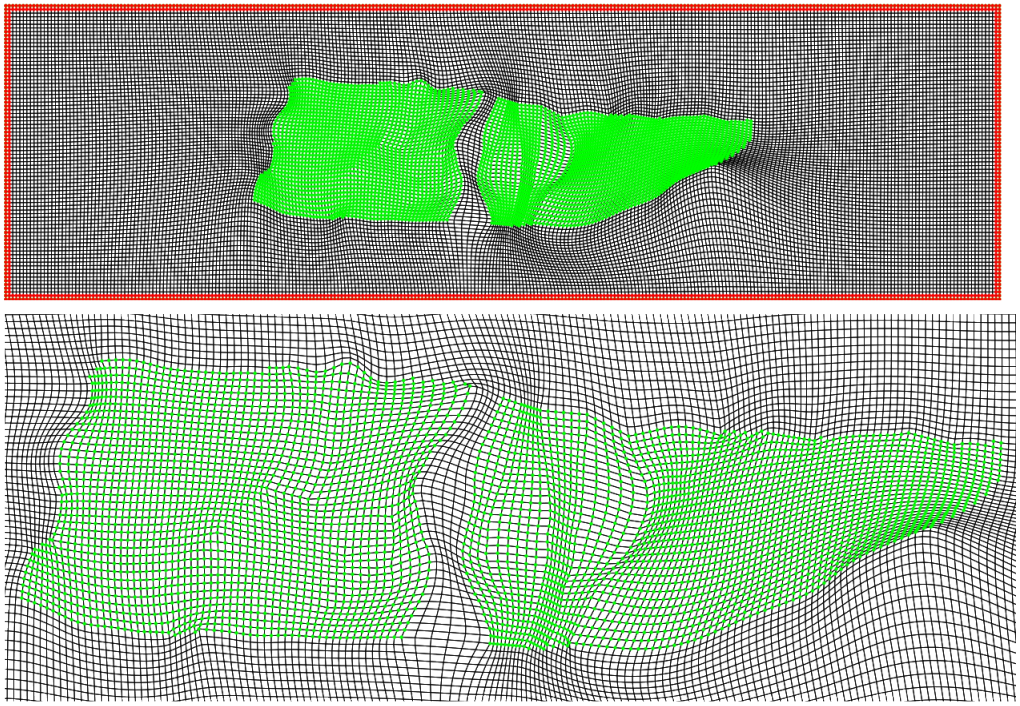

Figura B.5: Solução da suavização da grade bidimensional .