Universidade Federal da Paraíba
Centro de Informática
Programa de Pós-Graduação em Modelagem Matemática e Computacional

APLICAÇÃO DO MÉTODO DAS SOLUÇÕES FUNDAMENTAIS EM PROBLEMAS INVERSOS GEOMÉTRICOS

Manuel Esteban Ramírez Carrillo

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Modelagem Matemática e Computacional, UFPB, da Universidade Federal da Paraíba, como parte dos requisitos necessários à obtenção do título de Mestre em Modelagem Matemática e Computacional.

Orientadores: Jairo Rocha de Faria

Thiago José Machado

João Pessoa
Março de 2020

Catalogação na publicação
Seção de Catalogação e Classificação

R173a Ramírez Carrillo, Manuel Esteban.
Aplicação do Método das Soluções Fundamentais em Problemas Inversos Geométricos / Manuel Esteban Ramírez Carrillo. - João Pessoa, 2020.
145 f. : il.

Orientação: Jairo Rocha de Faria, Thiago José Machado.
Dissertação (Mestrado) - UFPB/CI/PPGMMC.

1. Problemas Inversos Geométricos. 2. Otimização de Forma. 3. Método das Soluções Fundamentais. I. Faria, Jairo Rocha de. II. Machado, Thiago José. III. Título.

UFPB/BC

APLICAÇÃO DO MÉTODO DAS SOLUÇÕES FUNDAMENTAIS EM PROBLEMAS INVERSOS GEOMÉTRICOS

Manuel Esteban Ramírez Carrillo

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM MATEMÁTICA E COMPUTACIONAL (PPGMMC) DO CENTRO DE INFORMÁTICA DA UNIVERSIDADE FEDERAL DA PARAÍBA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM MODELAGEM MATEMÁTICA E COMPUTACIONAL.

Examinada por:

Prof. Jairo Rocha de Faria, D.Sc.
Jairo Rocha de Faria
SIAPE 1565613

Prof. Thiago José Machado, D.Sc.

Prof.ª Ana Paula Pintado Wyse, D.Sc.

Prof. Sidarta Araújo de Lima, D.Sc.

JOÃO PESSOA, PB – BRASIL
MARÇO DE 2020

Agradecimentos

Agradeço primeiramente a minha família, pelo total apoio e confiança.

Aos meus orientadores, o professor Jairo Rocha, pela sua permanente disposição, aconselhamento e paciência durante o desenvolvimento deste trabalho, ao professor Thiago Machado, pelos valiosos conselhos e lições que me permitiram explorar a linha de pesquisa deste trabalho de dissertação.

Ao professor Antônio Boness e à professora Ana Wyse, pelo total apoio altruísta o tempo todo, o que me permitiu continuar e concretizar este projeto.

Por fim, agradeço o apoio financeiro da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) e a todos que contribuíram direta ou indiretamente para meu sucesso.

Resumo da Dissertação apresentada ao PPGMMC/CI/UFPB como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

APLICAÇÃO DO MÉTODO DAS SOLUÇÕES FUNDAMENTAIS EM PROBLEMAS INVERSOS GEOMÉTRICOS

Manuel Esteban Ramírez Carrillo

Março/2020

Orientadores: Jairo Rocha de Faria

Thiago José Machado

Programa: Modelagem Matemática e Computacional

O objetivo do presente trabalho é explorar algumas aplicações do Método das Soluções Fundamentais (MSF) na resolução de problemas inversos geométricos. Inicialmente, estudam-se as reconstruções de um subdomínio contido em um conjunto limitado para as equações de Helmhotz modificada e Laplace. Finalmente apresentam-se duas formas de utilização do método das soluções fundamentais em um problema de Tomografia por Impedância Elétrica (EIT) que visa reconstruir a área úmida de um recipiente a partir de leituras de voltagem obtidas de um sistema de eletrodos alimentados com corrente elétrica e colocados sobre as paredes que contêm o referido líquido. A ideia básica da metodologia usada para resolver ambos os problemas supracitados consiste em minimizar um funcional de custo, que relaciona as leituras reais lidas e os dados numéricos obtidos através do MSF, sendo os últimos dados dependentes da fronteira a ser reconstruída. Vários exemplos numéricos são apresentados para a verificação da validade dos resultados. Para uma simulação mais próxima da realidade, incorporamos ruído gaussiano branco (WGN) nos dados.

Palavras-chave: Problemas Inversos Geométricos, Otimização de Forma, Método das Soluções Fundamentais.

Abstract of Dissertation presented to PPGMMC/CI/UFPB as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

THESIS TITLE

Manuel Esteban Ramírez Carrillo

March/2020

Advisors: Jairo Rocha de Faria

Thiago José Machado

Program: Computational Mathematical Modelling

The aim of this work is to explore some applications of the Method of Fundametal Solutions (MFS) to solve geometric problems. Initially, we study the reconstructions of a subdomain contained in a bounded set for the modified Helmhotz and Laplace equations. Finally, two ways of using the Fundamental Solutions Method for a problem of Electrical Impedance Tomography (EIT) are presented. Such methods are aimed at reconstructing the wet area of water from voltage measurements obtained from a system of electrodes fed with a certain electrical current and placed on the walls containing that liquid. The basic idea of the methodology used to solve both of the above problems is to minimize a cost function, which relates the measurements and the numerical data obtained through the MFS, the last data being dependent on the boundary to be reconstructed. Several numerical examples are presented to check the validity of the results. For a simulation more realistic, we incorporate White Gaussian Noise (WGN) into the data.

Keywords: Geometric Inverse Problems, Shape Optimization, Method of Fundamental Solutions.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Fenômenos físicos são modelados a partir de dados de entrada, dados de saída e parâmetros que descrevem o modelo. Quando os parâmetros do modelo são conhecidos, tem-se também um conjunto de dados de entrada e deseja-se obter dados de saída; então tem-se um problema direto. Por outro lado, um problema inverso pretende conhecer os parâmetros do modelo a partir dos dados de entrada e de saída ou vista conhecer os dados de entrada a partir dos parâmetros do modelo e os dados de saída. Os problemas inversos vêm sendo estudados em muitos campos da ciência, especialmente em áreas de engenharia [58] e medicina [19]. Estes problemas são geralmente mais difíceis de resolver analiticamente e numericamente do que um problema direto, uma vez que são não lineares e, em geral, mal postos no sentido de Hadamard [73]. Nas diferentes vertentes dos problemas inversos, os problemas inversos geométricos visam determinar uma parte da fronteira do domínio da solução de um fenômeno, ou seja, identificar defeitos como obstáculos, cavidades ou fissuras. Existem aplicações na física [28, 32, 36, 76], eletrodinâmica [31], acústica [22, 25], astronomia [54] ou bem na detecção de tumores, no campo da medicina através da tomografia [18].

O presente capítulo está dividido em seis subseções. Na primeira, descreve-se brevemente o método das soluções fundamentais, que será utilizado para a solução numérica dos problemas estudados. Na segunda subseção, apresenta-se o método de regularização de Tikhonov, bastante utilizado na resolução de problemas inversos. Na terceira subseção é definido o conceito de problema inverso geométrico. A quarta subseção apresenta uma breve revisão bibliográfica da metodologia que está sendo usada para abordar este tipo de problemas inversos; em seguida, na quinta subseção expõe-se a estratégia utilizada para tratar o problema e finalmente, na sexta subseção descreve-se a estrutura do presente trabalho.

1.1 Método das Soluções Fundamentais (MSF)

Existe uma variedade de diferentes métodos numéricos para resolver equações diferenciais parciais. Podem ser citados, por exemplo: o método dos elementos finitos (MEF) [43], o método das diferenças finitas (MDF) [72], o método dos elementos de contorno ou de fronteira (MEC) [13] e o método dos volumes finitos (MVF) [45]. Estes métodos têm demonstrado ser capazes de produzir aproximações fiéis para várias equações diferenciais; no entanto, existem certas limitações e dificuldades ao aplicá-las. Por exemplo, o MEF, o MDF e o MVF, requerem que seja gerada uma malha sobre o domínio da solução, podendo ser um processo computacionalmente caro, além disso, há dificuldades adicionais quando a fronteira do domínio varia no tempo. O MEC apenas exige a formação de uma malha sobre a fronteira do domínio, o qual reduz a dimensão do problema em comparação com o MEF e o MDF. No entanto, é preciso integrar sobre essa fronteira.

O Método das Soluções Fundamentais (MSF), introduzido inicialmente por Kupradze e Aleksidze em [47, 48] com o nome do método das equações funcionais, vem sendo um importante método amplamente usado para solucionar equações diferenciais parciais (EDP) com condições de contorno, cuja solução fundamental é conhecida explicitamente. O MSF é um método de colocação, ou seja, as condições de fronteira são impostas apenas a um número finito de pontos. Além disso, tem como principal vantagem o fato de não requerer discretizar o domínio envolvido no problema tal como no MEF ou MDF, pois o MSF é um método sem malha.

Para ilustrar como este método é implementado, vamos assumir que temos um domínio $\Omega \subset \mathbb{R}^n$, $n \in \mathbb{N}$, limitado, com fronteira denotada por Γ e tem-se a EDP

$$\mathcal{L}u = 0, \quad \text{em } \Omega, \tag{1.1}$$

com condições de contorno

$$\mathcal{B}_i u = f_i, \quad \text{sobre } \Gamma_i, \ i = 1, \ldots, P, \tag{1.2}$$

onde $\mathcal{L}$ é um operador diferencial, $\Gamma = \cup_{i=1}^{P} \Gamma_i$, $\cap_{i=1}^{P} \Gamma_i = \phi$, $\mathcal{B}_i$ denotam as condições de Dirichlet, Neumann ou Robin e f_i são funções dadas. O MSF consiste em aproximar a solução do problema (1.1), através de uma combinação linear das soluções fundamentais, ou seja

$$u_{MSF}(x) = \sum_{j_1}^{N} a_j F(x, y_j), \quad x \in \overline{\Omega} = \Omega \cup \Gamma, \tag{1.3}$$

onde F é a solução fundamental do problema (1.1), $(y_j)_{j=1}^{N}$ são chamados de pontos

de fonte (singularidades), localizados fora do domínio $\overline{\Omega}$ da solução do problema, enquanto $(a_j)_{j=1}^{N}$ são coeficientes constantes a determinar.

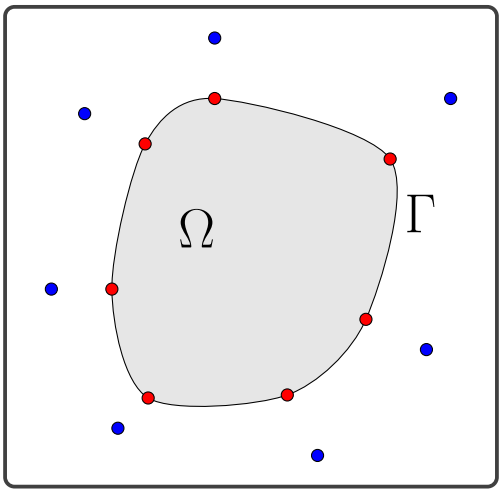

Figura 1.1: Representação de um esquema possível para os pontos de colocação (pontos vermelhos) e pontos de fonte (pontos azuis).

A determinação dos valores dos coeficientes é feita através da imposição das condições de contorno (1.2) considerando-se um número finito de pontos, chamados pontos de colocação. A Figura 1.1 mostra um possível esquema dos pontos fonte e colocação.

Assim, se M pontos de colocação fossem escolhidos sobre Γ, então o sistema de equações gerado pela imposição das condições de contorno é um sistema de M equações lineares com N incógnitas, o qual pode ser escrito como

$$Aa = b, \tag{1.4}$$

onde A é uma matriz de $M \times N$, a um vetor de $N \times 1$ incógnitas que é preciso determinar para obter a solução numérica (1.3) e b é o vetor dos valores das funções f_i nos pontos de colocação.

No melhor dos cenários possíveis, o sistema de equações (1.4) pode ser resolvido através da eliminação gaussiana sempre que $M = N$ e a matriz A tenha posto completo. Por sua vez, se $M > N$, a solução do sistema de equações (1.4) pode ser obtida usando o método dos mínimos quadrados. No entanto, o MSF muitas vezes gera matrizes mal condicionadas e é necessário utilizar algum método de regularização, como a regularização de Tikhonov [28], que é utilizada no presente trabalho.

1.2 O Método de Regularização de Tikhonov

Conforme mencionado na seção anterior, o MSF usualmente gera matrizes mal condicionadas. Este tipo de sistema é geralmente chamado de um problema discreto linear mal posto.

O método de regularização de Tikhonov substitui o sistema (1.4), pelo problema de otimização

$$\min_{x\in\mathbb{R}^N} \|Ax-b\|^2 + \lambda\|P^{(k)}x\|^2, \tag{1.5}$$

onde $\lambda > 0$ é um parâmetro de regularização e a matriz $P^{(k)} \in \mathbb{R}^{(N-k)\times N}$ é uma aproximação discreta de um operador derivada que determina a ordem de regularização, o que, por sua vez, induz uma restrição de continuidade C^k à solução a_λ do problema (1.5) (para mais informações, consulte [28]). Por conseguinte, as regularizações de Tikhonov da ordem $k = 0, 1, 2$ são dadas respectivamente pelas matrizes $P^{(0)} = \mathbb{I} \in \mathbb{R}^{N\times N}$,

$$P^{(1)} = \begin{pmatrix} -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & -1 & 1 \end{pmatrix} \in \mathbb{R}^{(N-1)\times(N)},$$

$$P^{(2)} = \begin{pmatrix} -1 & 2 & 1 & 0 & \dots & 0 \\ 0 & -1 & 2 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & -1 & 2 & 1 \end{pmatrix} \in \mathbb{R}^{(N-2)\times(N)}.$$

A escolha da ordem de regularização k depende principalmente das necessidades do problema em que é utilizada [55] (as ordens de regularização 0, 1, 2, enfatizam respectivamente a suavidade da solução, a suavidade das derivadas espaciais da solução ou a suavidade da curvatura espacial). Neste trabalho são utilizadas duas regularizações, a primeira para as soluções associadas ao MSF (com ênfase na suavidade das soluções, ou seja, $k = 0$, já que não é necessário exigir maiores requerimentos para obter soluções com adequada aproximação [55]) e a segunda para a regularização dos pontos da fronteira a reconstruir (com ênfase nas ordens de regularização $k = 1, 2$, assumindo que se deseja reconstruir superfícies lisas [10]).

Para obter a solução do sistema (1.5), basta igualar o seu gradiente a zero.

Dessa forma, obtemos

$$a_\lambda = (A^\intercal A + \lambda P^{(k)\intercal} P^{(k)})^{-1} A^\intercal b. \tag{1.6}$$

Observe que o termo residual $\|Ax - b\|^2$ no problema de minimização (1.5) pode ser visto como um indicador de quão bem a solução a_λ do problema se aproxima do sistema (1.4).

Por sua vez, note que, se o parâmetro λ for muito pequeno, o termo $\|Ax-b\|^2$ será levado mais em conta no problema de minimização, o que produzirá uma solução que se encaixa bem no sistema. No entanto, esta será dominada pelas contribuições de possíveis erros contidos nos dados (vetor b), portanto, o termo $\|P^{(k)}x\|^2$ será muito grande. Pelo contrário, se λ for muito grande, o problema de minimização será focado no termo $\|P^{(k)}x\|^2$, produzindo uma solução excessivamente suave; pois, este termo controla o comportamento da frequência de oscilação da solução causada pelo ruído nos dados. Além disso, o termo residual $\|Ax - b\|^2$ será muito grande e a solução não se ajustará adequadamente aos dados.

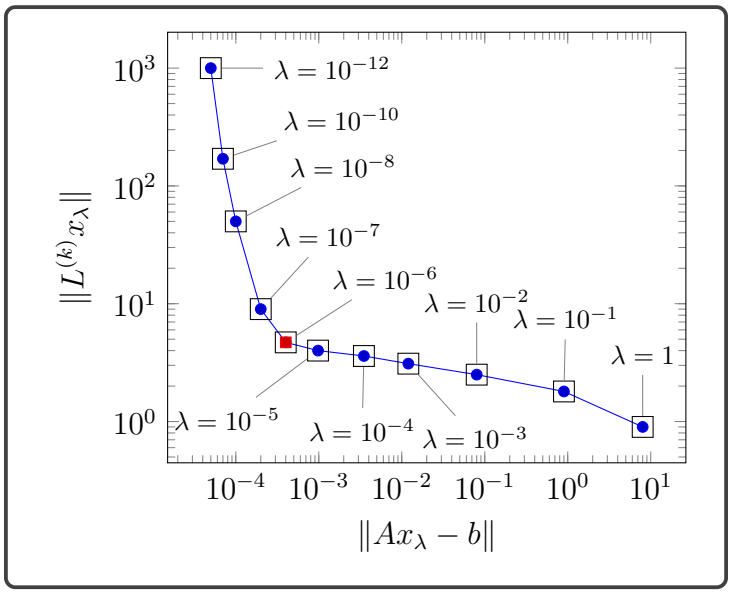

Figura 1.2: Exemplo de um gráfico log-log da curva L para a regularização de Tikhonov.

No que se segue ao presente trabalho, o critério da curva L [29] será considerado para a escolha do parâmetro de penalização λ. Este critério consiste em eleger um valor para λ, que corresponde à "esquina" do gráfico $\|P^{(k)}a_\lambda\|$ vs $\|Aa_\lambda - b\|$, ou seja, o parâmetro correspondente ao ponto de máxima curvatura no gráfico. Assim, a curva L é de fato uma curva de balanço entre quantidades que devem ser controladas. A Figura 1.2 mostra um exemplo típico de um gráfico da curva L, para $P^{(0)}$. Para este caso, uma possível escolha do termo de regularização seria $\lambda = 10^{-6}$. Para mais

detalhes, bem como uma análise do critério da curva L, ver [30].

1.3 Problemas Inversos Geométricos

Dado um modelo matemático de um fenômeno físico, este pode ser escrito em termos de um operador $A : X \longrightarrow Y$ como

$$Ax = y, \tag{1.7}$$

onde X e Y são espaços funcionais e o domínio de definição X é um conjunto não vazio. O problema (1.7) é dito bem posto no sentido de Hadamard se forem satisfeitas as três condições seguintes:

- O operador A é sobrejetivo, isto é, para cada $y \in Y$, existe pelo menos um $x \in X$ (existência de solução).
- O operador A é injetivo, em outras palavras, se $Ax_1 = Ax_2$ para $x_1, x_2 \in X$, então $x_1 = x_2$ (unicidade da solução).
- O operador inverso A^{-1} existe e é contínuo em Y (estabilidade).

Logo, um problema é dito mal posto se não satisfizer pelo menos uma das condições acima referidas. Uma definição de um problema inverso é dada por Kubo [46], onde um problema inverso é considerado o oposto de um problema direto. Um problema direto é proposto, sabendo com antecedência as seguintes informações:

1. a fronteira e/ou domínio da solução do problema;
2. a equação que governa o domínio do problema;
3. as condições de contorno sobre toda a fronteira do domínio da solução, bem como as condições iniciais, se necessário;
4. as propriedades materiais do domínio do problema;
5. as forças agindo no domínio do problema.

Quando uma ou mais das informações acima são desconhecidas, ou não totalmente especificadas, isto leva a um problema inverso, os quais são geralmente não lineares e mal postos no sentido de Hadamard.

Assim, o foco na obtenção de cada uma das informações ausentes acima leva a um tipo específico de problema inverso. O presente trabalho centra-se no tipo de problema inverso relativo ao item 1, que visa determinar a fronteira, uma parte dela ou o domínio da solução de um problema que modela certo fenômeno.

Como previamente comentado, um problema inverso geométrico visa identificar parte do domínio ou fronteira da solução de um determinado fenômeno, ou seja, identificar defeitos como obstáculos, cavidades ou fissuras, utilizando técnicas como a tomografia por impedância elétrica (EIT) (ver [76]); tomografia por radiação de raios gamma (GRET) (ver [15]), ou imagens por ressonância magnética (MRI) (ver [8]).

1.4 Revisão Bibliográfica

Os problemas inversos geométricos têm sido um tema de interesse na última década, pois eles modelam defeitos, tais como: obstáculos, cavidades, inclusões, deficiências, falhas e fissuras, cuja identificação é de grande importância na engenharia e na indústria.

Para a solução do problema inverso, as condições na parte conhecida da fronteira devem ser sobredeterminadas, uma vez que, nestes problemas a localização e/ou a forma de alguma parte da fronteira do domínio da solução são desconhecidas. Uma metodologia bastante utilizada na literatura para superar a dificuldade de trabalhar com problemas sobredeterminados, que são inerentes no contexto de problemas inversos, é reescrevê-lo como um problema de otimização. Em particular, problemas inversos geométricos estão associados a problemas de otimização de forma, os quais podem ser escritos como a minimização de um funcional de forma ou função objetivo J, a qual é avaliada sobre $\Omega \subset \mathbb{R}^d$, onde $d = 1, 2, 3$ e Ω pertence ao conjunto de formas admissíveis $\mathcal{U}_{ad}$. Isto é

$$\min_{\Omega \in \mathcal{U}_{ad}} J(\Omega);$$

além disso, é comum na maioria dos casos que $J(\Omega)$ dependa de uma função, a qual é solução de uma equação diferencial parcial definida em Ω.

Entre os métodos que lidam com problemas inversos geométricos, destacam-se aqueles que empregam a Análise à Mudança de Forma (SSA : *Shape Sensitivity Analysis*), baseados em expansões assintóticas, como, por exemplo: o Método da Velocidade, originado a partir da mecânica dos fluidos [70]; e o método da derivada topológica, que é baseado em perturbações singulares do domínio envolvido [51]. Dentre as aplicações destes métodos, podem ser citadas, por exemplo: detecção de fissuras [2], reconstrução de obstáculos [59], o problema inverso de espalhamento (*Scattering Problem*), que consiste na determinação da forma ou composição interna de um objeto através do estudo do comportamento das ondas (reflexão e refração), sejam mecânicas ou eletromagnéticas, emitidas ao corpo em estudo [22, 25]. Tais exemplos mostram que o método apresenta importantes resultados para vários problemas inversos geométricos. Contudo, eles envolvem muitas vezes cálculos adi-

cionais, por exemplo, a solução numérica de várias EDP's associadas ao método numérico e/ou funcional de forma usado, o que resulta em uma maior complexidade temporal e espacial dos algoritmos computacionais a serem implementados para a resolução do problema.

O Método da Velocidade também tem-se mostrado eficaz para fazer frente a problemas inversos de espalhamento [12, 69], assim como, problemas inversos de interface [7, 53], que consistem em determinar a forma de um corpo ou substância contidos noutro corpo ou material, com base nas suas características materiais (coeficiente de condutividade).

Além disso, o Método da Velocidade também tem sido usado recentemente no problema de tomografia por impedância elétrica aplicada à análise da umidade em estruturas [33]. Este problema é um caso particular do problema de tomografia de impedância elétrica (EIT), que constitui uma técnica para a resolução de problemas de interface através do uso de eletrodos e medidas elétricas (corrente ou voltagem) feitas na fronteira externa do domínio do problema em questão.

O presente trabalho centra-se em particular no problema de EIT para a análise da umidade em estruturas. Como alguns exemplos da aplicação do método da velocidade para o tratamento deste problema, pode-se citar principalmente o trabalho de Rymarczyk et al. [64–66], quem emprega a otimização topológica (derivada topológica) e também a otimização de forma (derivada de forma ou Método da Velocidade) em todos os seus trabalhos. Em [66], seu algoritmo de otimização é baseado no método de Gauss-Newton além de um híbrido deste método e o método de conjuntos de nível (*Level set Method* [14]), alcançando resultados efetivos; em [64] os autores usam redes neurais artificiais (ANN) [42], onde além da eficácia, mostram a eficiência da nova abordagem utilizada. Finalmente, no trabalho [65], Rymarczyk apresenta os resultados de estudos com diferentes algoritmos de aprendizado de máquinas (*Machine Learning*). A ideia geral que é utilizada consiste em construir uma malha fixa no interior da parede de forma a utilizar cada elemento da malha como um *pixel* correspondente a uma cor que está associada à condutividade elétrica desse elemento. As medições de voltagem e corrente são então realizadas em cada par de eletrodos que são colocados em cada face da parede e estas leituras são usadas para treinar em cada *pixel* da rede um sistema baseado na aprendizagem supervisada [11] (ANN, o método de regressão de menor ângulo (LAR) [62] e o método de rede elástica (*ElasticNet* [77])). O autor utiliza os três métodos supracitados, mostrando apenas resultados eficazes com ANN e LAR. Além disso, os autores utilizam EIDORS, uma *toolbox* do MATLAB, para modelar o domínio do problema e algoritmos de otimização. Também usa o MEF para resolver o problema inverso e melhorar a eficiência dos seus resultados. A principal desvantagem é que a qualidade da reconstrução da distribuição da umidade depende da quantidade de *pixels* considerada no momento

da geração da malha, ou seja, para obter reconstruções de melhor qualidade é necessário considerar uma malha com uma grande quantidade de elementos, produzindo um elevado custo computacional do algoritmo. Outro ponto fraco é a quantidade de redes neurais ou funcionais envolvidas na formação do sistema inteligente, uma vez que uma é treinada para cada *pixel*, mostrando melhores resultados quanto maiores forem o número de medidas consideradas.

1.5 Proposta do Trabalho

De modo a evitar lidar com cálculos adicionais envolvidos na utilização dos métodos já mencionados na seção anterior para a solução de problemas inversos geométricos, este trabalho apresenta uma forma alternativa de abordar estes problemas sem o uso da análise à mudança de forma. Além disso, o MSF é utilizado para evitar o problema da discretização iterativa do domínio que envolve o uso de métodos malha-dependentes, como o MEF, usado na maioria dos problemas inversos já mencionados, com ênfase especial no problema de EIT para análise da umidade de estruturas.

A ideia básica proposta para resolver os problemas de reconstrução supracitados consiste na modelagem do problema, reformulando-o como um problema de otimização não linear mediante a minimização de um funcional de custo, que relaciona as leituras lidas na fronteira do domínio do problema e os dados numéricos obtidos através do MSF, sendo estes dados, dependentes da fronteira a ser reconstruída. Pretende-se mostrar vários exemplos numéricos incorporando em alguns deles, ruído gaussiano branco (WGN). Além disso, é usada a *toolbox* de otimização do MATLAB.

Mais precisamente, a proposta do trabalho será apresentada através de exemplos numéricos para dois tipos de problemas inversos geométricos.

O primeiro problema consiste na reconstrução de um obstáculo e para a sua resolução, é utilizada a proposta acima mencionada. Aborda-se a equação de Helmholtz modificada e para minimizar o funcional são utilizados os algoritmos dos pontos interiores [3] e *Active-Set* [50] da *toolbox* de otimização do MATLAB. Posteriormente, a título de comparação, uma tentativa de resolução alternativa é mostrada usando a análise de mudança de forma, mais precisamente o Método da Velocidade e a derivada de forma de primeira ordem. Além disso, de modo a simplificar os cálculos que envolvem a utilização deste método, a equação de Laplace é considerada no problema de reconstrução. No entanto, no presente trabalho, não foram obtidos resultados eficazes para o problema da reconstrução através deste método, o que será mostrado no capítulo 2.

O segundo problema considerado é o de EIT aplicado à análise da umidade em estruturas. Para resolvê-lo, quatro principais exemplos numéricos são apresentados. No primeiro exemplo, um algoritmo é implementado em MATLAB. Seguindo

a proposta, um funcional custo que depende, ao mesmo tempo, de um conjunto de pontos do obstáculo a ser reconstruído e dos coeficientes associados ao MSF é minimizado por meio do algoritmo dos pontos interiores da *toolbox* de otimização do MATLAB. No segundo exemplo, para resolver o mesmo problema de EIT inverso, é proposto outro algoritmo que reduz o número de variáveis do funcional custo para otimizar apenas um conjunto de pontos da fronteira do obstáculo a ser reconstruído. Esta estratégia reduz significativamente a complexidade do algoritmo e, portanto, o número de iterações para reconstruir a fronteira procurada. O terceiro exemplo mostra a reconstrução da fronteira livre do problema EIT para uma fronteira alvo mais complexa do que a utilizada nos dois exemplos anteriores, para a qual o MSF é utilizado para obter dados fictícios que servem para resolver o problema utilizando o mesmo algoritmo proposto no exemplo anterior. O último exemplo estende o exemplo anterior a um caso em três dimensões e mostra a eficácia da ideia proposta, mesmo poluindo os dados com WGN.

1.6 Organização do Trabalho

Os capítulos seguintes estão organizados da seguinte forma: no capítulo 2, mostra-se como o MSF é aplicado aos problemas de reconstrução através de alguns exemplos, considerando as equações modificadas de Helmholtz e Laplace, comparando a proposta de trabalho com o uso da análise de mudança de forma (Método da Velocidade). O capítulo 3 centra-se no problema de EIT para a análise da umidade estrutural, utilizando dois métodos derivados da primeira proposta de resolução exposta no capítulo 1. Apresentam-se exemplos numéricos em duas e três dimensões com ruído moderado nas medições. A umidade nas paredes é detectada de forma satisfatória mediante uma abordagem simples e da *toolbox* de otimização do MATLAB. Por fim, o capítulo 4 apresenta as conclusões gerais do trabalho e os possíveis trabalhos futuros.

Capítulo 2

Exemplo de Reconstrução de um Obstáculo Usando o MSF

Neste capítulo mostra-se um exemplo da solução de um problema de reconstrução mediante a aplicação do MSF. O problema consiste na determinação da forma de um conjunto interno contido em um domínio regido pela equação de Helmholtz modificada.

2.1 Formulação Matemática do Problema

Seja $\Omega \subset \mathbb{R}^n$, $n = 1, 2, 3$, um conjunto simplesmente conexo, aberto e limitado, com fronteira Γ_0 Lipschitz contínua. Considere D um subdomínio fixo não vazio aberto e limitado de Ω, também com fronteira $\partial D = \Sigma$, Lipschitz contínua e $\Omega^* = \Omega \backslash \overline{D}$.

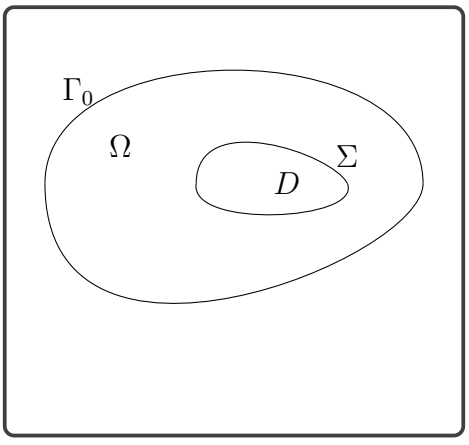

Figura 2.1: Domínio Ω e o subdomínio alvo D.

Considere também o seguinte problema de contorno

$$\begin{cases} \mathcal{L}u = b, & \text{em } \Omega^*, \\ u = g, & \text{sobre } \Gamma_0, \\ u = h, & \text{sobre } \Sigma, \end{cases} \tag{2.1}$$

onde $\mathcal{L}$ é um operador diferencial linear, $b \in \mathbb{R}$ é constante, $g \in H^{1/2}(\Gamma_0)$ e $h \in H^{1/2}(\Sigma)$, sendo $H^{1/2}(\Gamma_0)$ o espaço dos traços das funções em $H^1(\Omega)$, sobre a fronteira Γ_0 e Σ. se o conjunto D for conhecido e o operador $\mathcal{L}$ corresponder à equação de Laplace ou Helmholtz modificada (equações utilizadas neste trabalho), então pode ser provado, através da formulação variacional e do teorema de Lax-Milgram, que o problema (2.1) tem uma solução única $u \in H^1(\Omega^*)$, ver [17, 21].

No entanto, o problema geométrico inverso a ser considerado consiste em encontrar o conjunto D. O objetivo, portanto, é reconstruir D ou a sua fronteira Σ. Para tanto, será considerada também a condição de Neumann sobre a fronteira Γ_0

$$\frac{\partial u}{\partial n} = f, \text{ sobre } \Gamma_0, \tag{2.2}$$

para $f \in H^{-1/2}(\Gamma_0)$, onde $H^{-1/2}(\Gamma_0)$ denota o espaço dual associado a $H^{1/2}(\Gamma_0)$. A condição (2.2) pode ser obtida a partir da realização de leituras do fenômeno em estudo sobre a fronteira Γ_0.

Assim, o problema inverso geométrico consiste em reconstruir o domínio D (Σ) que satisfaça ao problema

$$\begin{cases} \mathcal{L}u = b, & \text{em } \Omega^*, \\ u = g, & \text{sobre } \Gamma_0, \\ u = h, & \text{sobre } \Sigma, \\ \dfrac{\partial u}{\partial n} = f, & \text{sobre } \Gamma_0, \end{cases} \tag{2.3}$$

O problema inverso geométrico (2.3) é mal posto, uma vez que não depende continuamente dos dados (condições g e f sobre Γ_0), ver [38]. No entanto, o problema inverso possui solução única. A existência e unicidade (ver [38]) é provada utilizando a teoria do potencial para problemas inversos [37] e o método da ortogonalidade [57]. Para detalhes da existência e unicidade de soluções para problemas geométricos inversos mais gerais, ver [52].

2.2 Primeiro Método de Resolução: O Método das Soluções Fundamentais para a Equação de Helmholtz Modificado

Nesta seção será exposta uma forma de aplicar o método das soluções fundamentais para resolver o problema da determinação de um defeito D contido em um domínio Ω limitado e simplesmente conexo, baseado no método apresentado em [10].

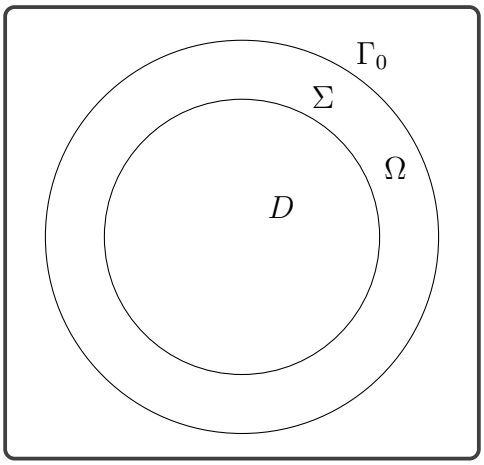

Figura 2.2: Domínio $\Omega = B(0,1)$ e o subdomínio procurado D.

2.2.1 Reformulação Matemática do Problema usando o MSF para a Equação de Helmholtz

A seguir, apresenta-se uma forma de aplicar o método das soluções fundamentais para a solução do problema inverso, considerando um domínio $\Omega \subset \mathbb{R}^2$ simplesmente conexo, com fronteira suave, um subdomínio D suave de Ω, tal que $\overline{D} \subset \Omega$ e $\Omega \backslash D$ é conexo (ver Figura 2.2). Além disso, vamos considerar a equação de Helmholtz modificada

$$\Delta u - k^2 u = 0\,, \quad \text{em } \Omega \backslash \overline{D}, \tag{2.4}$$

onde k^2 é uma constante conhecida e as seguintes condições de contorno:

$$u = g \text{ sobre } \Gamma_0, \tag{2.5}$$

$$u = h \text{ sobre } \Sigma, \tag{2.6}$$

onde $g \in H^{1/2}(\Gamma_0)$ e $h \in H^{1/2}(\Sigma)$ são funções dadas. Neste caso considera-se que não se conhece D e, além das condições dadas (2.5), (2.6); suponha que dispõe-se

da seguinte informação:

$$\left.\frac{\partial u}{\partial n}\right|_{\Gamma_0} = f, \tag{2.7}$$

para $f \in H^{-1/2}(\Gamma_0)$.

De modo a utilizar o MSF para resolver o problema inverso, procede-se a aproximar a solução do problema (2.4) pela seguinte combinação linear:

$$u_{MSF}(x) = \sum_{i=1}^{M+N} a_i \phi(x, \xi_i), \quad x \in \overline{\Omega} \backslash D, \tag{2.8}$$

onde ϕ é a solução fundamental da equação de Helmholtz modificada (2.4) em $\mathbb{R}^2$, ou seja,

$$\phi(x, y) = \frac{1}{2\pi} K_0(k\|x - y\|), \tag{2.9}$$

com $\|x - y\| = ((x_1 - y_1)^2 + (x_2 - y_2)^2)^{1/2}$, $x = (x_1, x_2)$, $y = (y_1, y_2)$ e o ponto y denota a singularidade da solução fundamental. No caso da expressão (2.8), as $M + N$ singularidades ξ_i, $i = 1, ..., M + N$, também chamados pontos fonte, são escolhidas de modo que fiquem fora do domínio da solução de (2.4), ou seja, $\xi_i \in \mathbb{R}^2 \backslash (\overline{\Omega} \backslash D)$, $i = 1, ..., M + N$. Dessa forma, a solução u_{MSF} está bem definida em todo $\overline{\Omega} \backslash D$.

Por outro lado, K_0 denota a função de Bessel modificada de segundo tipo de ordem zero e os coeficientes a_i, $i = 1, ..., M + N$, são números reais a serem determinados mediante o uso das condições de fronteira (2.5), (2.6) e (2.7). Note que na expressão (2.8), a constante $1/2\pi$ da solução fundamental ϕ foi inserida nos coeficientes $(a_i)_{i=1}^{M+N}$, para cada termo do somatório.

Para resolver o problema inverso acima, é necessário escolher um conjunto de pontos de fonte para a expressão (2.8), os quais devem ser pontos que não pertençam ao domínio de solução do problema, ou seja, os pontos fonte devem pertencer ao conjunto $(\mathbb{R}^2 \backslash \overline{\Omega}) \cup D$. Por outro lado, como nosso objetivo é encontrar a solução u_{MSF}, será necessário considerar outro conjunto de pontos, chamados pontos de colocação, que serão utilizados para se impor as condições de contorno e determinar as constantes $(a_i)_{i=1}^{M+N}$ da combinação linear (2.8).

Por uma questão de simplicidade, vamos considerar Ω sendo a bola centrada na origem de raio 1, isto é, $\Omega = B(0, 1)$ e a fronteira do domínio desconhecido D definida como

$$\Sigma = \{x \in \mathbb{R}^2 | (r(\theta)\cos(\theta), r(\theta)\sin(\theta)),\ \theta \in (0, 2\pi]\}, \tag{2.10}$$

onde $r : [0, 2\pi] \longrightarrow [0, 1]$ é uma função contínua suave.

Assim, são tomados os seguintes M pontos de colocação

$$x_i = \left(\cos\left(\frac{2\pi i}{M}\right), \sin\left(\frac{2\pi i}{M}\right)\right), \quad i = 1, ..., M, \tag{2.11}$$

sobre a fronteira $\Gamma_0 = \partial B(0,1)$ e os seguintes N pontos sobre Σ

$$x_j = \left(r_j \cos\left(\frac{2\pi j}{N}\right), r_j \sin\left(\frac{2\pi j}{N}\right)\right), \quad j = M+1, ..., M+N. \tag{2.12}$$

Note-se que o conjunto de pontos dado pela expressão anterior (2.12), pertence à fronteira Σ, cuja forma foi dada na expressão (2.10).

Quanto aos pontos fonte, por motivos de simplicidade, são apontados M pontos da fronteira fictícia $\partial B(0,R)$,

$$\xi_i = Rx_i = \left(R\cos\left(\frac{2\pi i}{M}\right), R\sin\left(\frac{2\pi i}{M}\right)\right), \quad i = 1, ..., M, \tag{2.13}$$

onde $R > 1$, assim como N pontos pertencentes ao domínio desconhecido D cuja forma é dada por

$$\xi_j = \frac{1}{\rho}x_j = \left(\frac{r_j}{\rho}\cos\left(\frac{2\pi j}{N}\right), \frac{r_j}{\rho}\sin\left(\frac{2\pi j}{N}\right)\right), \quad j = M+1, ..., M+N, \tag{2.14}$$

onde $\rho > 1$. A escolha dos pontos fonte (2.14) poderia ter sido mais arbitrária na condição de pertencer ao interior da suposta fronteira D, no entanto, para efeitos de simplicidade, são utilizados os pontos que figuram na expressão (2.14). Note que essa escolha dos pontos fonte assegura que eles pertençam ao interior de D, os quais são necessários para a aplicação do MSF já que tanto (2.13) como (2.14) são pontos fora do domínio de solução $\overline{\Omega}\backslash D$.

Para determinar os coeficientes $(a_i)_{i=1}^{M+N}$ da solução u_{MSF}, basta resolver o problema de colocação associado ao MSF usando as condições conhecidas (2.5) e (2.6), isto é, resolver o seguinte sistema de equações

$$Aa = b, \tag{2.15}$$

onde $a = (a_1, a_2, \ldots, a_{M+N})^\intercal$, $(A)_{ij} = \phi(x_i, \xi_j)$, $i, j = 1, \ldots, M+N$ e $b = (f(x_1), \ldots, f(x_M), h(x_{M+1}), \ldots, h(x_{M+N}))^\intercal$. Repare que as primeiras M linhas do sistema linear anterior estão associadas a (2.5), sendo esta uma condição sobre Γ_0, a qual é uma parte da fronteira do domínio de solução $\Gamma_0 \cup \Sigma$; enquanto as restantes N linhas correspondem à outra parte da fronteira da solução Σ dadas pela condição (2.6).

No entanto, como nosso principal objetivo é reconstruir o obstáculo D, o seguinte

funcional será usado

$$\overline{F}(a,r) = \|u-g\|^2_{L^2(\Gamma_0)} + \left\|\frac{\partial u}{\partial n} - f\right\|^2_{L^2(\Gamma_0)} + \|u-h\|^2_{L^2(\Sigma)}, \tag{2.16}$$

sendo $a = (a_1, \ldots, a_{M+N})$ e $r = (r_1, \ldots, r_N)$, onde a condição adicional (2.7) utilizada garante a resolução do problema geométrico inverso como já mencionado na formulação matemática do problema inverso. Note que o funcional (2.16) permite encontrar a solução para o problema e reconstruir o obstáculo ao mesmo tempo.

Logo, a expressão do funcional (2.16) resulta em

$$F(a,r) = \sum_{i=1}^{M} \left\{ \left[\sum_{j=1}^{M+N} a_j \phi(x_i, \xi_j) - g(x_i) \right]^2 + \left[\sum_{l=1}^{M+N} a_l \frac{\partial \phi}{\partial n}(x_i, \xi_l) - f(x_i) \right]^2 \right\} + \sum_{i=1+M}^{M+N} \left[\sum_{j=1}^{M+N} a_j \phi(x_i, \xi_j) - h(x_i) \right]^2 \tag{2.17}$$

Assim, tem-se um problema de otimização não linear com $M + 2N$ variáveis a serem determinadas. Em resumo, precisamos resolver o problema

$$\begin{cases} \min\limits_{(a,r)\in\mathbb{R}^{M+2N}} \quad F(a,r), \\ \text{sujeito a} \\ \qquad\qquad 0 < r_i < 1,\ i = 1, \ldots, N, \quad r = (r_1, \ldots, r_N). \end{cases} \tag{2.18}$$

Neste trabalho, são utilizados três métodos de otimização da *toolbox* de otimização do MATLAB (método dos pontos interiores, método do conjunto ativo e algoritmos genéticos). Estes métodos serão brevemente descritos, antes de mostrar os resultados numéricos que envolvem sua utilização.

Os resultados numéricos que serão apresentados na seção 2.2.3 utilizam o método dos pontos interiores.

Apresenta-se a seguir uma breve descrição do algoritmo de pontos interiores.

2.2.2 Método dos Pontos Interiores

O método dos pontos interiores (*Interior Point Method*), na otimização não linear, foi introduzido pela primeira vez por Anthony V. Fiacco e Garth P. McCormick em 1960 [56]. A base deste método é incorporar as restrições do problema de otimização na função objetivo, criando uma função de barreira (*Barrier Function*). Para resolver o problema, além dos típicos métodos de Karush-Kuhn-Tucker (KKT) [60], também é utilizado um fator de perturbação, que limita a busca de potenci-

ais soluções numa região viável, resultando num algoritmo eficiente em termos de complexidade temporal.

Por tanto, uma das principais vantagens deste método é ter uma complexidade assintótica temporal polinomial (alta velocidade de execução do algoritmo). Também é melhor para resolver problemas de otimização com um grande número de variáveis alvo, pois a álgebra linear envolvida em sua formulação produz algoritmos mais rápidos do que outros métodos (sua matriz Jacobiana tem uma estrutura especial que permite que seja facilmente manipulada).

Por outro lado, o método dos pontos interiores, em geral, não se comporta como um otimizador global, uma vez que pode considerar como suficiente, critérios locais de otimização para encontrar uma solução. Ou seja, cada valor inicial das variáveis impostas pelo usuário poderia produzir um mínimo local, o que não é tão apropriado se o problema a ser resolvido contiver vários mínimos locais. Para mais detalhes do método, ver [3], por exemplo.

2.2.3 Resultados Numéricos

Nesta seção serão mostrados alguns resultados numéricos para o problema inverso geométrico (2.18) para a reconstrução do obstáculo D.

O seguinte exemplo com solução analítica foi extraído da tese de Bin-Mohsin, B. [10], onde o autor resolve o problema inverso usando a rotina E04FCF que faz parte da biblioteca NAG (software para resolução de otimização e problemas relacionados).

Neste trabalho, será utilizada a função *fmincon* da *toolbox* de otimização do *software* MATLAB, a qual permite resolver problemas de otimização restrita não lineares. Dos métodos de otimização fornecidos pela função *fmincon*, foi escolhido o método de pontos interiores, devido ao fato de lidar bem com problemas de otimização que envolvem várias variáveis e apresentar uma maior velocidade de execução. Tal método é utilizado em todos os casos dos testes numéricos desta seção.

De modo a resolver o problema inverso geométrico, considerem-se as seguintes condições de contorno:

$$u(1,\theta) = f(\theta) = e^{\cos(\theta)+\sin(\theta)}, \qquad \text{sobre } \partial B(0,1), \tag{2.19}$$

$$\frac{\partial u}{\partial n}(1,\theta) = g(\theta) = (\cos(\theta)+\sin(\theta))e^{\cos(\theta)+\sin(\theta)}, \qquad \text{sobre } \partial B(0,1), \tag{2.20}$$

$$u(r(\theta),\theta) = h(\theta) = e^{0.7(\cos(\theta)+\sin(\theta))}, \qquad \text{sobre } \Sigma, \tag{2.21}$$

Convém assinalar que o domínio Σ da condição (2.21) é desconhecido para o problema inverso (2.18), sendo sua fronteira parametrizada pela expressão (2.10).

Assume-se que o domínio que se deseja encontrar tem a forma de uma bola de

raio 0.7, a saber, $D_r = B(0, 0.7)$. As condições de contorno (2.19), (2.20), (2.21), produzem a solução analítica

$$u_a(r, \theta) = e^{r(\cos(\theta)+\sin(\theta))}, \tag{2.22}$$

portanto, ao resolver o problema inverso geométrico composto por (2.18), (2.19), (2.20), (2.21), espera-se obter uma solução numérica u_{MSF} da forma (2.8), que aproxime à solução analítica (2.22) e um conjunto de raios que, junto com a forma assumida da fronteira de D, aproximem-se da forma D_r.

Também, em todos os testes numéricos serão considerados $M = N$ pontos, o raio da fronteira fictícia para os pontos fonte (2.13) fora do domínio Ω sendo $R = 2$ e $\rho = 2$ parâmetro para os pontos fonte (2.14) no interior de D. A seguir são mostrados alguns testes para diferentes chutes iniciais D_0 e $(a_i)_{i=1}^{M+N}$.

- **Teste 1** : Neste primeiro caso considera-se como valores iniciais para o uso do algoritmo de otimização $D_0 = B(0, 0.3)$, $(a_i)_{i=1}^{M+N} = 1$.

  ⋆ **Caso M=N=5**:

  No gráfico do lado esquerdo da Figura 2.3 mostra-se a distribuição tanto dos pontos fonte quanto dos pontos de colocação. Com relação aos pontos fonte, $M = 5$ estão localizados sobre a fronteira fictícia $\partial B(0, 2)$ e $N = 5$ estão localizados sobre $\partial B(0, 0.3/2)$. Por sua vez, mostra-se a distribuição dos pontos de colocação ($M + N = 5 + 5 = 10$); $M = 5$ sobre $\Gamma_0 = \partial B(0, 1)$ e $N = 5$ sobre $D_0 = B(0, 0.3)$. Quanto ao gráfico do lado direito, mostra-se a fronteira real Σ_r (círculo azul), a qual deve ser determinada; a fronteira inicial escolhida (círculo vermelho descontínuo) e a fronteira determinada através da minimização do funcional (2.17), polígono verde.

  Observa-se que para $M = N = 5$, a aproximação D^* para D_r, dada pelo termo $r = (r_1, \ldots, r_{N=5})$ da solução (a, r) aproximada do funcional (2.17) correspondentes ao raio r, não é precisa.

  A Figura 2.4 mostra as comparações da solução analítica u_a e sua derivada normal $\partial u_a/\partial n$ com suas aproximações numéricas, dadas respectivamente por u_{MSF} e sua derivada normal $\partial u_{MSF}/\partial n$, obtidas pelo vetor $a = (a_1, \ldots, a_{M+N=10})$ da solução (a, r). Como pode ser observado, não se aproximam com precisão em alguns pontos, que em coordenadas polares correspondem a $\theta \in [0, 2]$, sobre a fronteira $\partial B(0, 0.85)$ a qual está contida no domínio de solução $\overline{B(0, 1)}\backslash D_r$.

  ⋆ **Caso M=N=10**:

  Ao se aumentar o número de pontos fonte e colocação, isto é, $M = N = 10$, nota-se uma melhoria tanto na forma final D^* como na solução

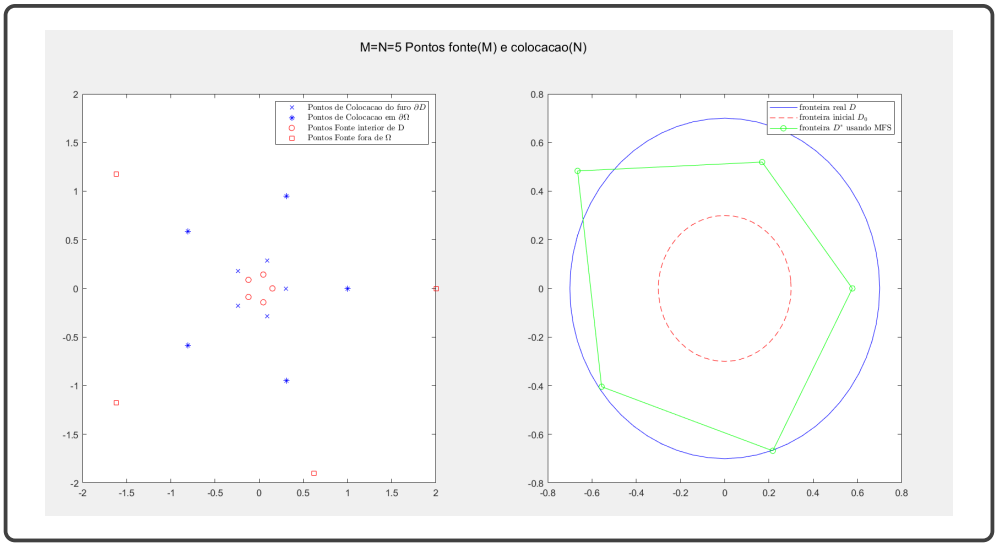

Figura 2.3: Distribuição dos pontos fonte e de colocação, domínio inicial D_0 e aproximação D^* para $M = N = 5$ no Teste 1.

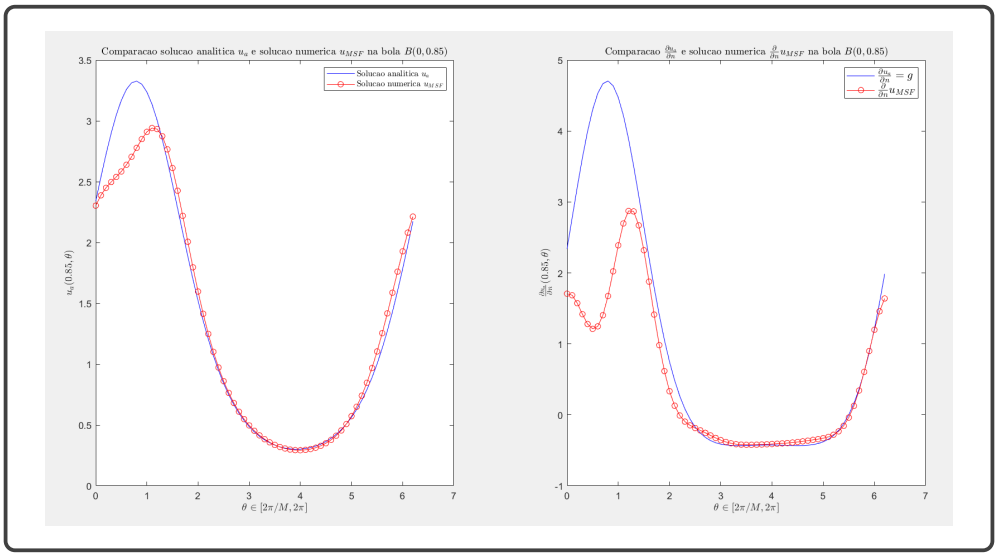

Figura 2.4: Comparação da solução analítica u_a e a solução numérica u_{MSF} sobre o conjunto $\partial B(0, 0.85)$ para $M = N = 5$ no Teste 1.

numérica u_{MSF} e sua derivada $\partial u_{MSF}/\partial n$, tal como indicado nas Figuras (2.5) e (2.6) respectivamente.

⋆ **Caso M=N=20**:
Se o número de pontos de fonte e colocação for aumentado para $M = N = 20$, é possível perceber uma melhoria apenas da solução numérica, mas não da forma do domínio D^*.

A Tabela 2.1 mostra o valor do funcional (2.17) avaliado na solução aproximada (a, r) e os erros das soluções numéricas e derivadas sobre $\partial B(0, 0.85) \subset \overline{B(0, 1)} \backslash D_r$.

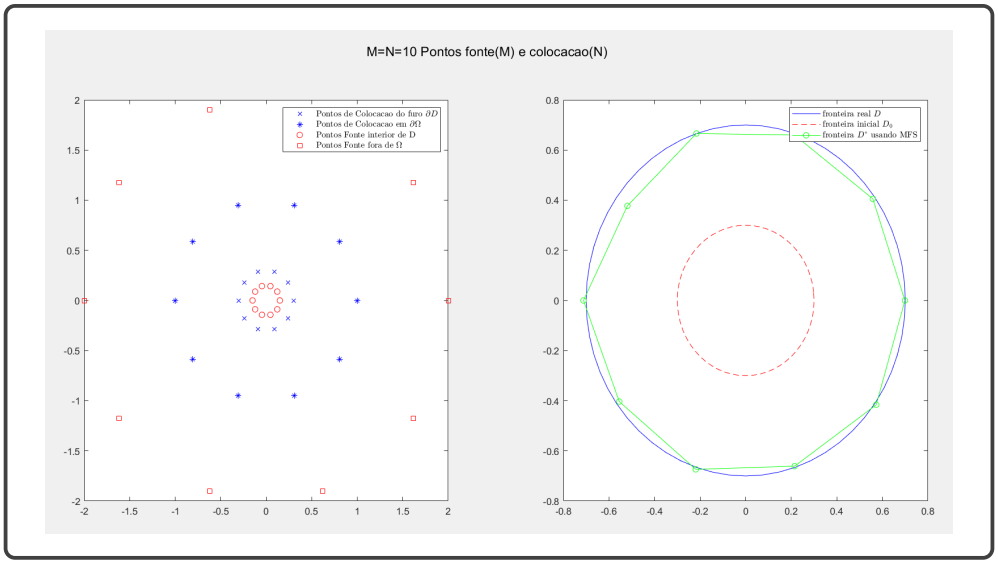

Figura 2.5: Distribuição dos pontos fonte e de colocação, domínio inicial D_0 e aproximação D^* para $M = N = 10$ no Teste 1.

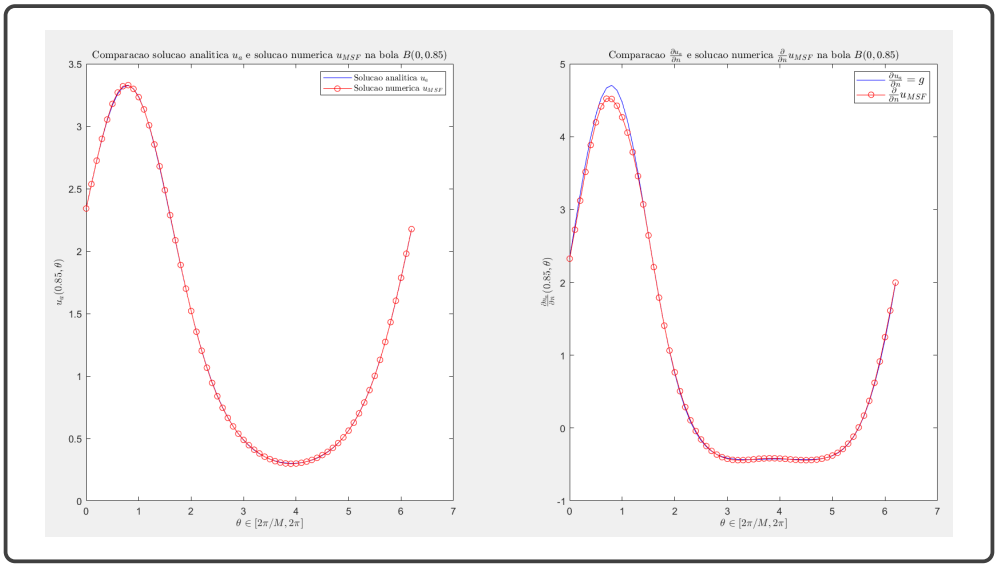

Figura 2.6: Comparação da solução analítica u_a e a solução numérica u_{MSF} sobre o conjunto $\partial B(0, 0.85)$ para $M = N = 10$ no Teste 1.

- **Teste 2** : Como segundo teste são consideradas as seguintes condições $D_0 = B(0, 0.5)$, $(a_i)_{i=1}^{M+N} = 0.5$ como valores iniciais para o algoritmo de otimização.

  ⋆ **Caso M=N=5**:

  As Figuras (2.9) e (2.10), mostram uma aproximação muito semelhante à do Caso 2.2.3, tanto para a forma final D^* como para a solução numérica u_{MSF} e sua derivada $\partial u_{MSF}/\partial n$, embora pontos de partida diferentes tenham sido tomados.

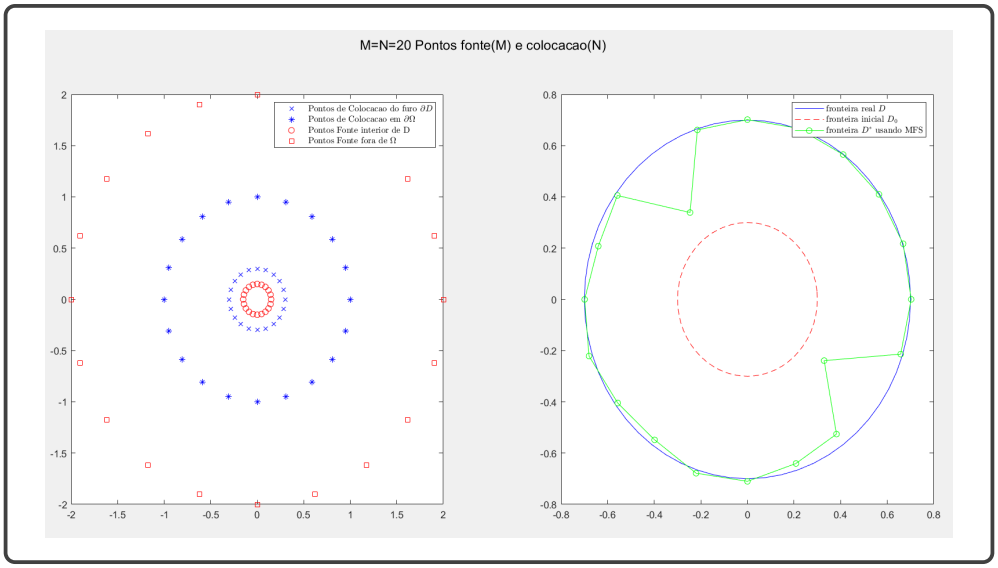

Figura 2.7: Distribuição dos pontos fonte e de colocação, domínio inicial D_0 e aproximação D^* para $M = N = 20$ no Teste 1.

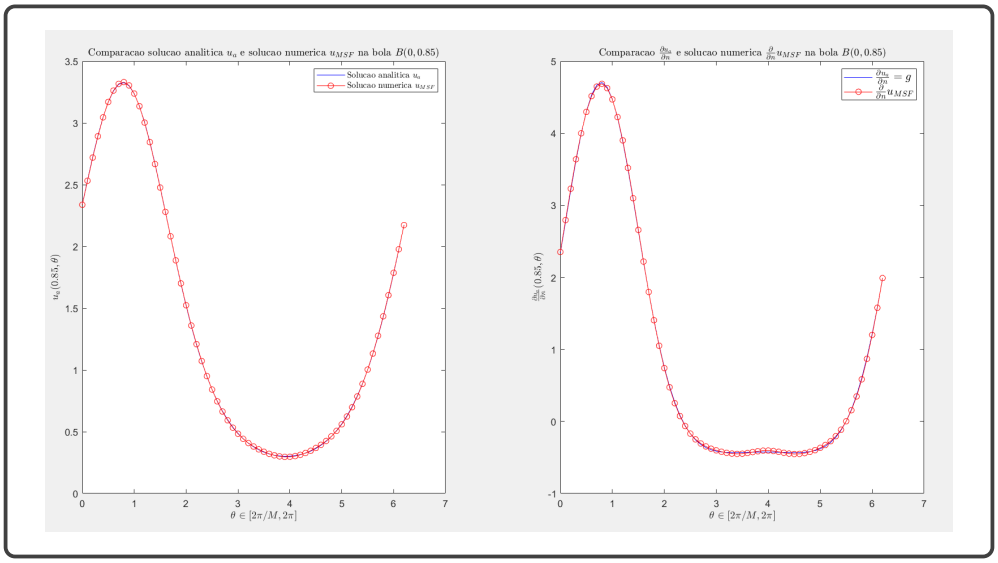

Figura 2.8: Comparação da solução analítica u_a e a solução numérica u_{MSF} sobre o conjunto $\partial B(0, 0.85)$ para $M = N = 20$ no Teste 1.

⋆ **Caso M=N=10**:

Quando o número de pontos é aumentado para $M = N = 10$, nota-se que eles melhoram tanto a solução numérica u_{MSF} como a forma final D^*, como pode ser visto nas Figuras (2.11) e (2.12) respectivamente.

⋆ **Caso M=N=20**:

Para um maior aumento no número de pontos, apenas se pode ver uma melhora na solução numérica u_{MSF} e $\partial u_{MSF}/\partial n$, mas não na aproximação do domínio buscado, como aconteceu no Caso 2.2.3. Ver Figuras (2.13), (2.14).

Tabela 2.1: Valores de $F(a,r)$, $\|u_a - u_{MSF}\|$ e $\|\frac{\partial u_a}{\partial n} - \frac{\partial u_{MSF}}{\partial n}\|$ para $M = N \in \{5, 10, 20\}$ no Teste 1.

$M = N$	$F(a,r)$	$\|u_a - u_{MSF}\|_{\partial B(0,0.85)}$	$\|\frac{\partial u_a}{\partial n} - \frac{\partial u_{MSF}}{\partial n}\|_{\partial B(0,0.85)}$
5	2.0000×10^{-2}	1.5857	8.5488
10	2.5605×10^{-5}	0.0295	0.4990
20	9.2492×10^{-5}	0.0227	0.0980

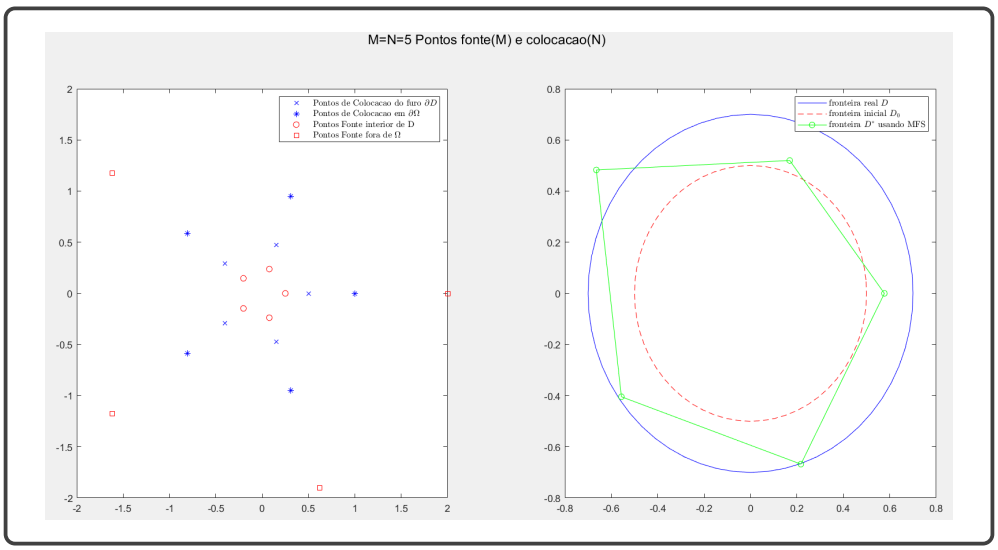

Figura 2.9: Distribuição dos pontos fonte e de colocação, domínio inicial D_0 e aproximação D^* para $M = N = 5$ no Teste 2.

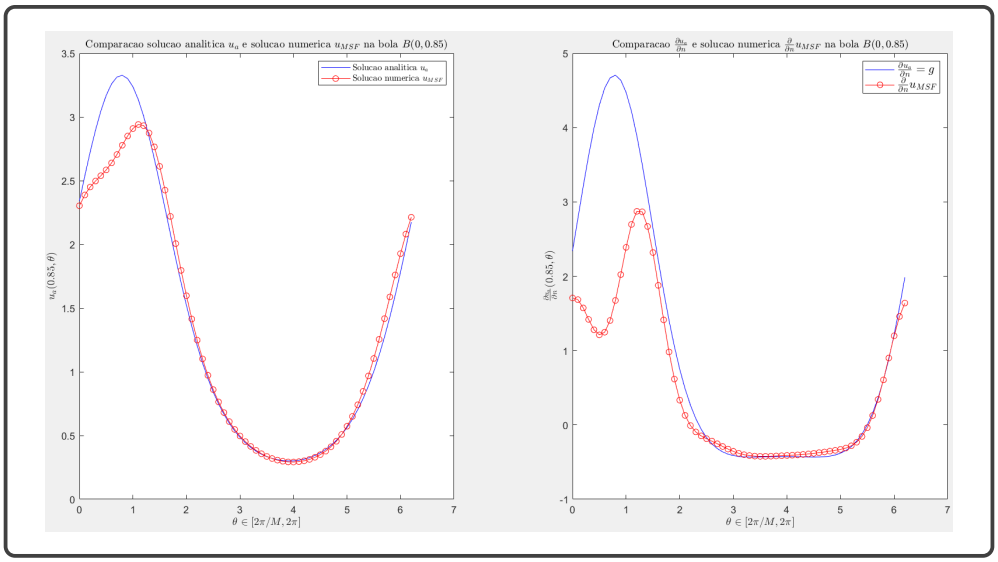

Figura 2.10: Comparação da solução analítica u_a e a solução numérica u_{MSF} sobre o conjunto $\partial B(0, 0.85)$ para $M = N = 5$ no Teste 2.

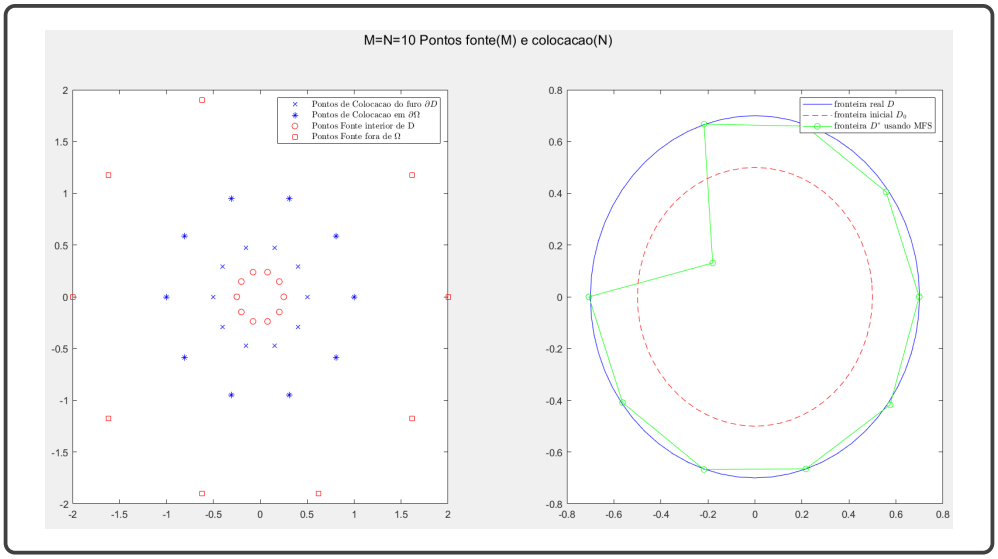

Figura 2.11: Distribuição dos pontos fonte e de colocação, domínio inicial D_0 e aproximação D^* para $M = N = 10$ no Teste 2.

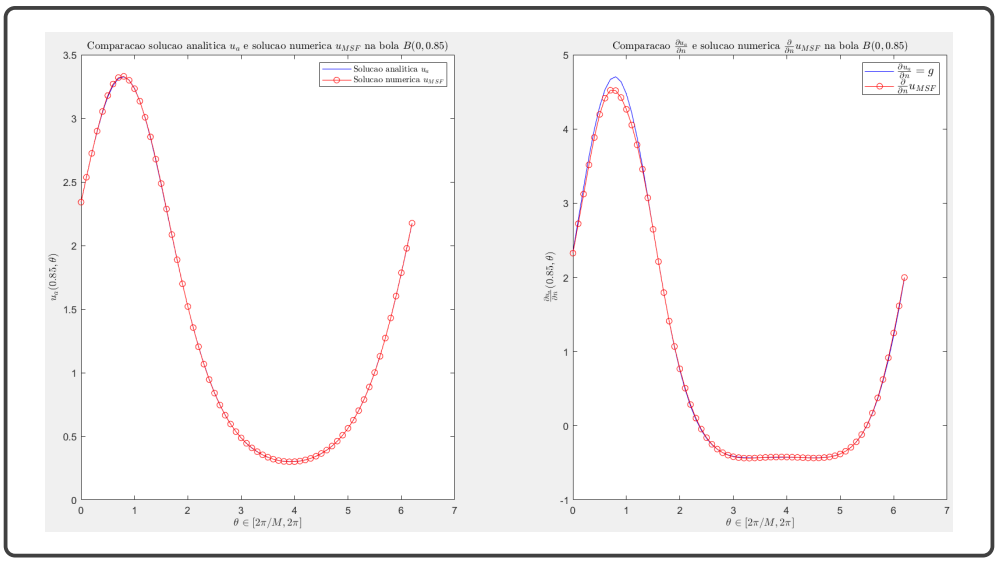

Figura 2.12: Comparação da solução analítica u_a e a solução numérica u_{MSF} sobre o conjunto $\partial B(0, 0.85)$ para $M = N = 10$ no Teste 2.

A melhoria das soluções numéricas pode ser vista na Tabela 2.2.

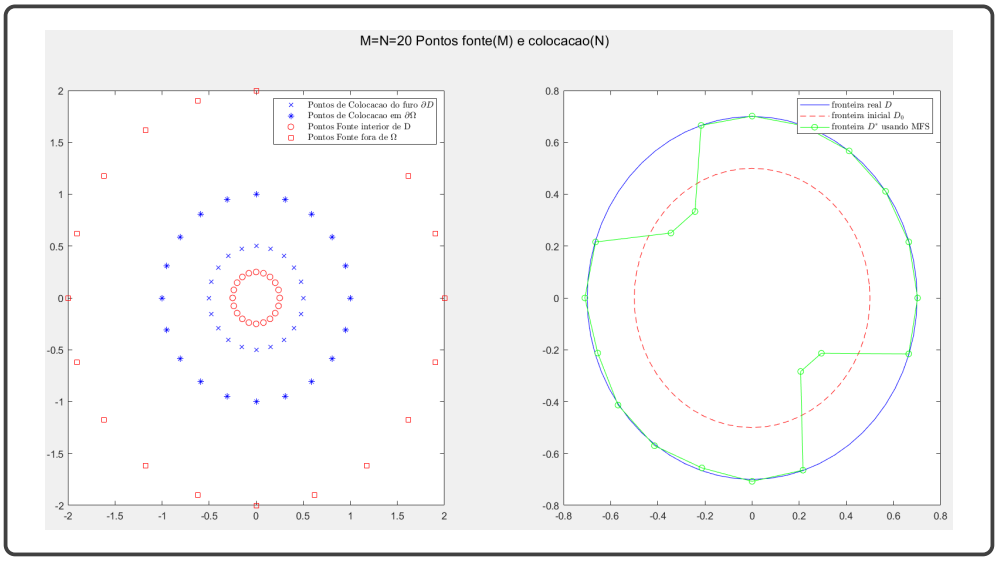

Figura 2.13: Distribuição dos pontos fonte e de colocação, domínio inicial D_0 e aproximação D^* para $M = N = 20$ no Teste 2.

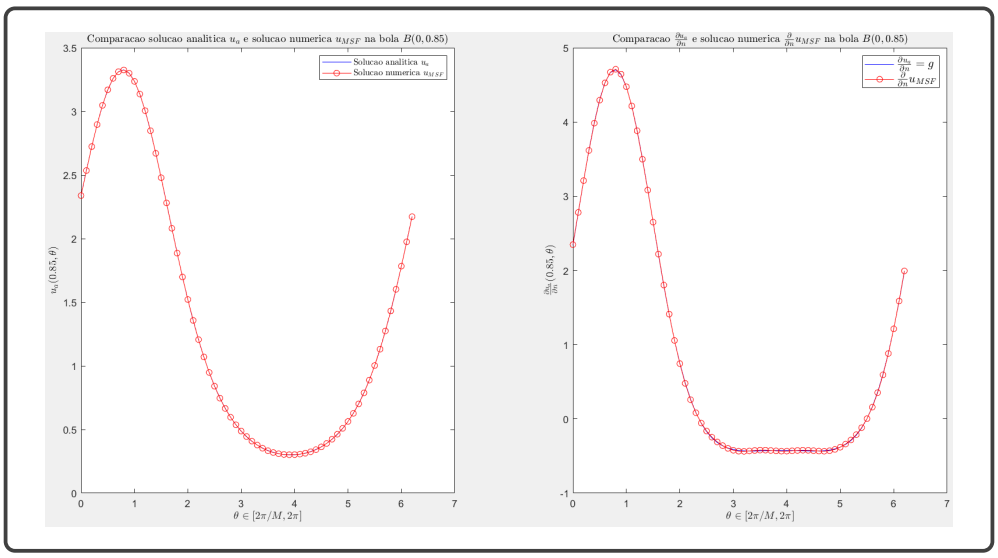

Figura 2.14: Comparação da solução analítica u_a e a solução numérica u_{MSF} sobre o conjunto $\partial B(0, 0.85)$ para $M = N = 20$ no Teste 2.

Tabela 2.2: Valores de $F(a, r)$, $\|u_a - u_{MSF}\|$ e $\|\frac{\partial u_a}{\partial n} - \frac{\partial u_{MSF}}{\partial n}\|$ para $M = N \in \{5, 10, 20\}$ no Teste 2.

$M = N$	$F(a, r)$	$\|u_a - u_{MSF}\|_{\partial B(0,0.85)}$	$\|\frac{\partial u_a}{\partial n} - \frac{\partial u_{MSF}}{\partial n}\|_{\partial B(0,0.85)}$
5	2.0000×10^{-2}	1.5856	8.5489
10	6.6334×10^{-6}	0.0265	0.4999
20	5.6148×10^{-6}	0.0072	0.0493

- **Teste 3** : Neste terceiro teste são considerados os seguintes valores iniciais $D_0 = B(0, 0.85)$, $(a_i)_{i=1}^{M+N} = 1$.

  Este caso difere dos casos anteriores uma vez que, neles, as fronteiras iniciais foram tomadas como bolas contidas na fronteira real D_r. Neste caso, a fronteira inicial D_0 é tomada como uma bola contendo a fronteira real.

  ⋆ **Caso M=N=5**:
  Para este número de pontos de colocação e fonte, os resultados são análogos aos obtidos nos casos anteriores. Ver Figuras (2.15) e (2.16).

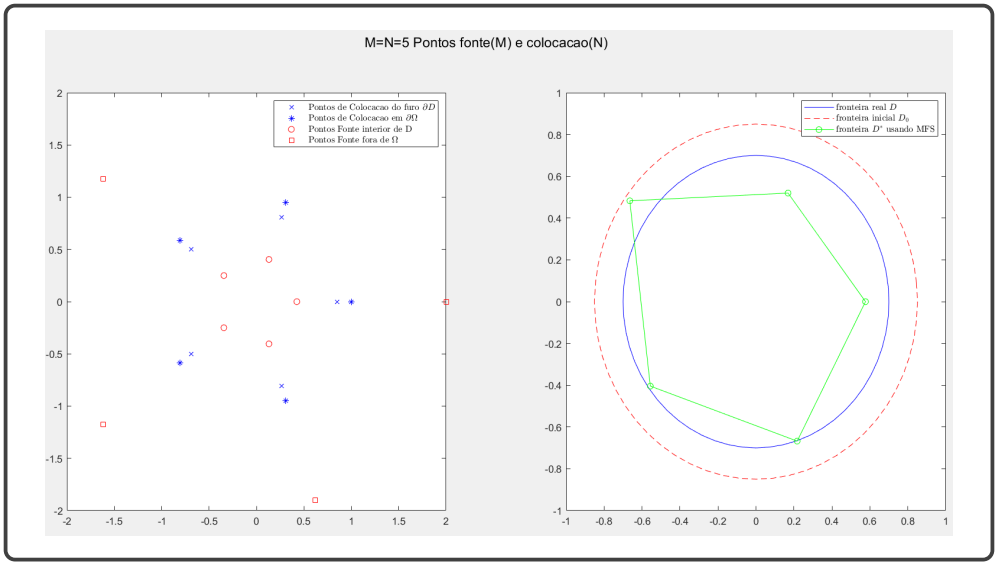

Figura 2.15: Distribuição dos pontos fonte e de colocação, domínio inicial D_0 e aproximação D^* para $M = N = 5$ no Teste 3.

  ⋆ **Caso M=N=10**:
  Nessa situação, os resultados melhoraram tanto para a solução numérica quanto para o domínio aproximado D^*, como ocorreu nos casos anteriores, como pode ser visto nas Figuras (2.17) e (2.18).

  ⋆ **Caso M=N=20**:
  Para um maior aumento no número de pontos, apenas as aproximações para a solução numérica u_{MSF} e sua derivada $\partial u_{MSF}/\partial n$ melhoraram, mas não para o domínio reconstruído D^*. Fato que se repetiu como nos casos anteriores. Ver (2.19) e (2.20).

  Na Tabela 2.3 pode-se observar a melhoria da solução numérica u_{MSF} e sua derivada $\partial u_{MSF}/\partial n$ à medida que o número de pontos de fonte e de colocação aumentam.

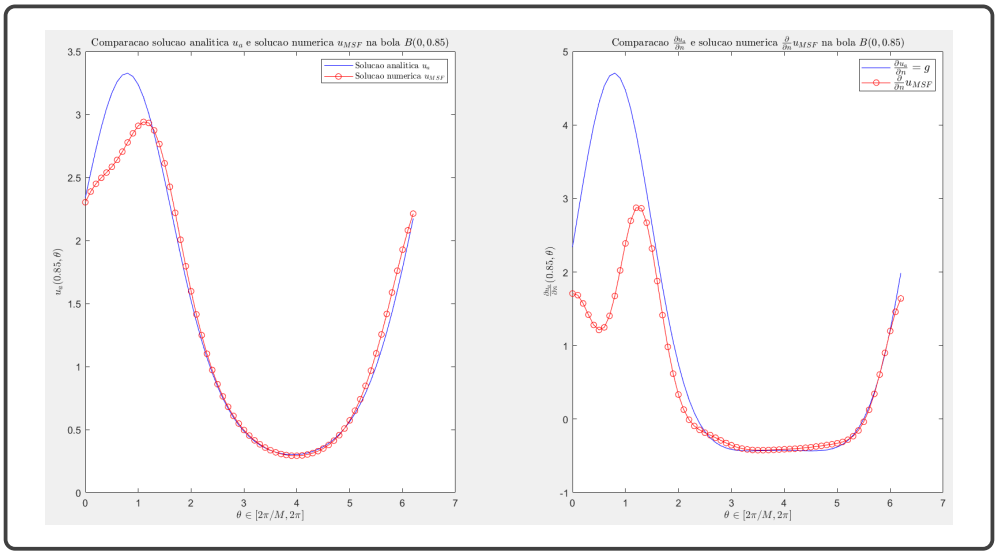

Figura 2.16: Comparação da solução analítica u_a e a solução numérica u_{MSF} sobre o conjunto $\partial B(0, 0.85)$ para $M = N = 5$ no Teste 3.

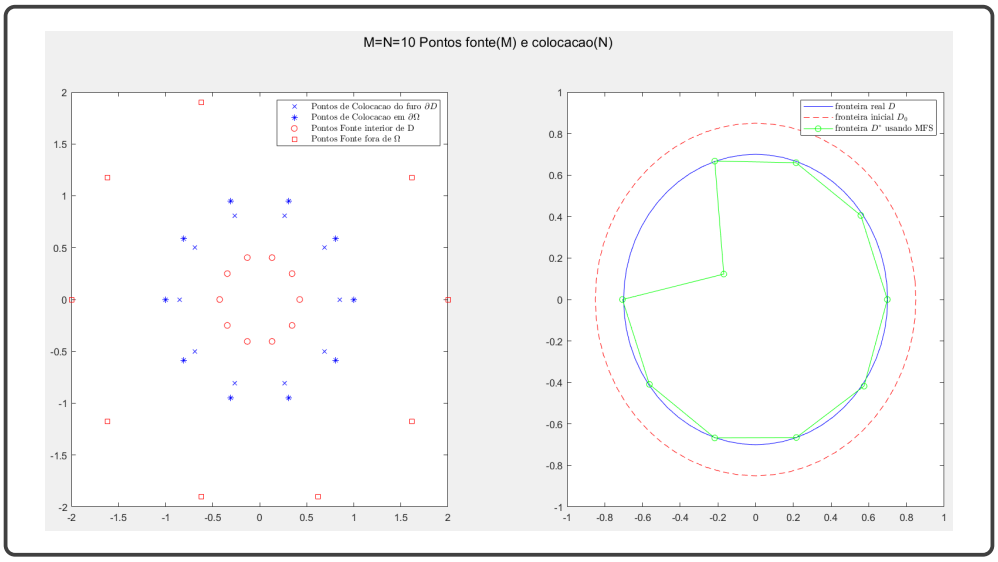

Figura 2.17: Distribuição dos pontos fonte e de colocação, domínio inicial D_0 e aproximação D^* para $M = N = 10$ no Teste 3.

Tabela 2.3: Valores de $F(a, r)$, $\|u_a - u_{MSF}\|$ e $\|\frac{\partial u_a}{\partial n} - \frac{\partial u_{MSF}}{\partial n}\|$ para $M = N \in \{5, 10, 20\}$ no Teste 3.

$M = N$	$F(a,r)$	$\|u_a - u_{MSF}\|_{\partial B(0,0.85)}$	$\|\frac{\partial u_a}{\partial n} - \frac{\partial u_{MSF}}{\partial n}\|_{\partial B(0,0.85)}$
5	2.0000×10^{-2}	1.5856	8.5489
10	6.2830×10^{-6}	0.0264	0.5000
20	8.0979×10^{-6}	0.0084	0.0559

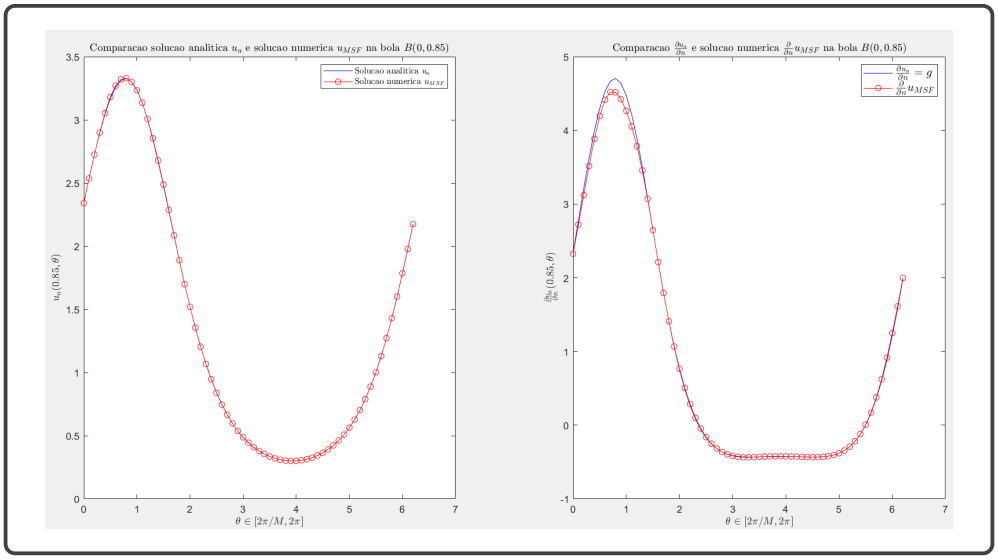

Figura 2.18: Comparação da solução analítica u_a e a solução numérica u_{MSF} sobre o conjunto $\partial B(0, 0.85)$ para $M = N = 10$ no Teste 3.

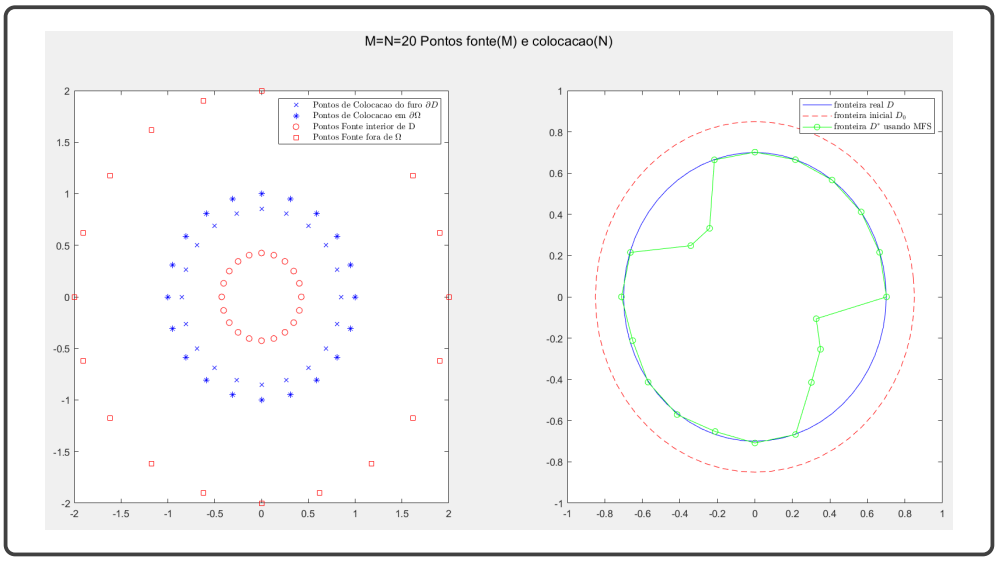

Figura 2.19: Distribuição dos pontos fonte e de colocação, domínio inicial D_0 e aproximação D^* para $M = N = 20$ no Teste 3.

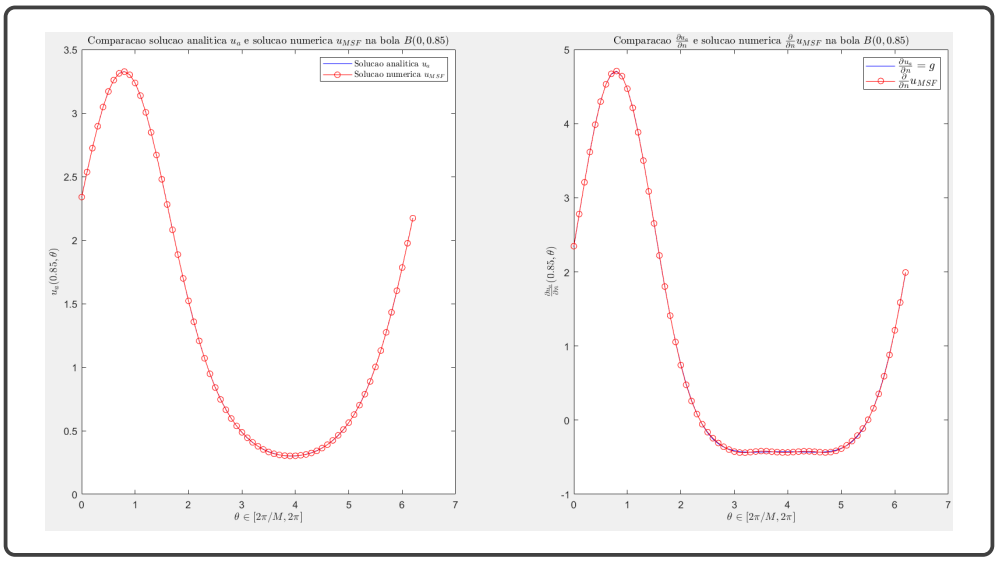

Figura 2.20: Comparação da solução analítica u_a e a solução numérica u_{MSF} sobre o conjunto $\partial B(0, 0.85)$ para $M = N = 20$ no Teste 3.

Os experimentos apresentados acima foram obtidos com um critério de parada de 1000 iterações para o algoritmo utilizado. A partir desses experimentos, pode-se concluir que este procedimento formulado, que utiliza o método das soluções fundamentais para a resolução do problema geométrico inverso (2.18), apesar de ter como intenção encontrar ao mesmo tempo, a solução numérica u_{MSF} e a forma do domínio desconhecido; só melhora gradualmente a solução numérica à medida que o número de pontos fonte e de colocação aumenta. Porém, em relação à reconstrução do domínio desconhecido, ele não é eficaz. Os melhores resultados em cada teste foram obtidos considerando um número relativamente pequeno de pontos de fonte e colocação, como visto nos três testes mostrados, quando $M = N = 10$. Além disso, em todos os casos acima, foi observado um comportamento decrescente do funcional objetivo, o que indicou que o processo de minimização é convergente (se isto não tivesse acontecido, isso poderia ter indicado que o processo de minimização é divergente ou, mais provavelmente, que o código computacional contém erros).

Como já foi evidenciado, surge um inconveniente com o funcional apresentado, pois ele reconstrói o domínio objetivo com certa irregularidade, ou seja, as distâncias entre pontos consecutivos que reconstroem o domínio alvo D são muito grandes, fato que sugere a regularização da distância entre esses pontos (componentes do vetor r) no funcional (2.18), pois como mencionado, este problema encontra tanto os pontos associados à solução numérica quanto os pontos que determinam a forma do domínio desconhecido ao mesmo tempo (componentes dos vetores b e a respectivamente).

- **Teste 4** : Neste teste, um termo de regularização foi incorporado ao funcional (2.16). Logo, a nova expressão a ser minimizada é dada por:

$$F_{\lambda_H} = F(a, r) + \lambda_H \|P^{(1)} r\|^2, \tag{2.23}$$

onde $P^{(1)}$ é a matriz associada à regularização de Tikhonov de ordem 1, isto é

$$P^{(1)} = \begin{pmatrix} -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & -1 & 1 \end{pmatrix} \in \mathbb{R}^{(N-1)\times(N)}. \tag{2.24}$$

Note que, ao incorporar o termo de regularização no funcional F_{λ_H}, a matriz $P^{(1)}$ contribuirá diminuindo as distâncias entre os pontos da fronteira a

otimizar, o que permitirá uma reconstrução mais suave da fronteira alvo.

Deve esclarecer-se que a escolha do parâmetro de regulação λ_H na expressão acima não pode ser feita através do critério da curva L, porque o termo $F(a, r)$ no funcional (2.23) não é linear. Um possível recurso para a escolha do parâmetro lambda poderia ser o critério da superfície L [6]. No entanto, os procedimentos que envolvem a sua utilização são muito complicados de implementar computacionalmente, pelo que, neste trabalho, a escolha dos parâmetros de regularização associados aos parâmetros funcionais regularizados baseia-se no critério de prova e erro para vários valores do parâmetro, ou seja, testando primeiro com valores pequenos e aumentando-os gradualmente até se perceber uma maior suavidade nas reconstruções da fronteira desconhecida, considerando várias formas diferentes como estimativas iniciais para o método utilizado na minimização do funcional regularizado (2.23).

É também importante referir que o critério da curva L, exposto no capítulo 1, será utilizado na seção 3.4 do capítulo 3, para a regularização das soluções associadas ao MSF, o que é possível porque será apresentada uma nova proposta de implementação do algoritmo de reconstrução que, primeiro, encontra as soluções numéricas associadas ao MSF (onde o critério da curva L é utilizado) e, depois, minimiza um funcional não linear que depende apenas dos pontos que reconstroem a superfície alvo.

Foram realizados testes para todos os casos anteriormente apresentados, obtendo-se melhores resultados. De modo a não prolongar este capítulo, apresenta-se em seguida o resultado para o caso: $M = N = 10$ e $D_0 = B(0, 0.3)$ e $(a_i)_{i=1}^{M+N} = 1$, que são as mesmas condições no segundo caso do teste 1 para fins de comparação. O parâmetro de regularização λ_H, para este caso é igual a 1, foi escolhido por teste e erro, como mencionado anteriormente, no intervalo $\{10^{-5}, 10^{-4}, \ldots, 10^{0}, \ldots, 10^{4}, 10^{5}\}$.

Com o exposto, é de se esperar uma melhora nos resultados experimentais; o que é conferido na Fig. 2.21, onde um resultado próximo ao domínio alvo foi alcançado com apenas 96 iterações usando o mesmo método de minimização não-linear dos testes anteriores, onde o processo para este quarto teste terminou tendo um funcional minimizado de ordem de 10^{-5}. Assim, a eficiência do método utilizado nesta seção foi melhorada (de 1000 iterações, obtidas no mesmo caso do Teste 1, para apenas 96 iterações).

Na Fig. 2.22, mostra-se a comparação das soluções analíticas e numéricas depois da incorporação do termo de regularização, observando-se uma ligeira discrepância, mas isso se deve ao uso da regularização que, como mencionado, está focada nos pontos que reconstruem o domínio.

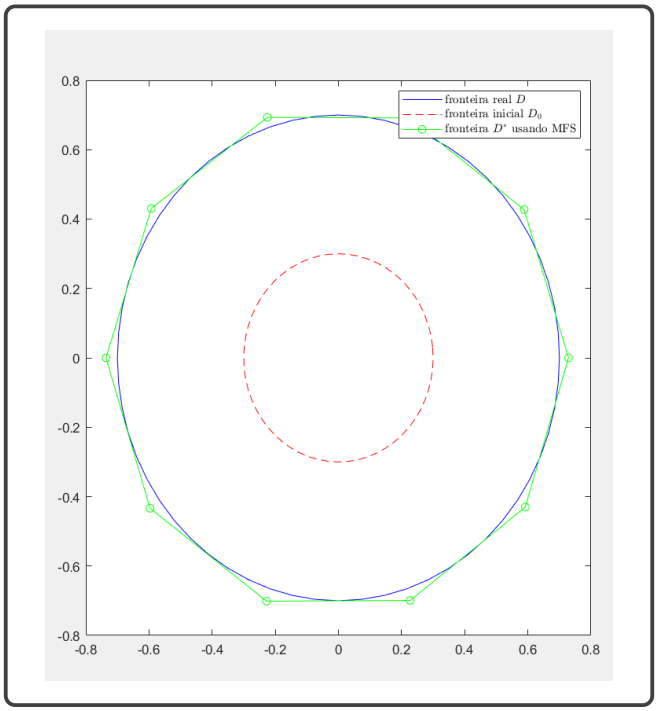

Figura 2.21: Domínio inicial $D_0 = B(0, 0.3)$ e aproximação D^* para $M = N = 10$ no Teste 4.

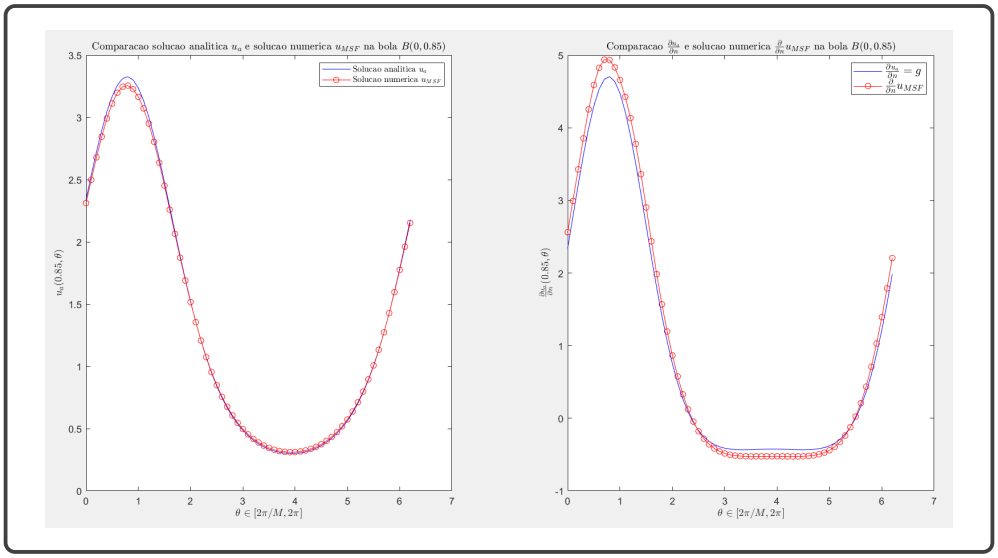

Figura 2.22: Comparação da solução analítica u_a e a solução numérica u_{MSF} sobre o conjunto $\partial B(0, 0.85)$ para $M = N = 10$ no Teste 4.

- **Teste 5** : Para terminar esta seção, mostra-se o resultado da reconstrução do mesmo obstáculo D dos testes anteriores, considerando também o funcional regularizado (2.23) e como chute inicial D_0 um domínio elíptico de semi-eixos 0.5 e 0.2. Além disso, $M = N = 10$, $(a_i)_{i=1}^{M+N} = 1$ e $\lambda_H = 1$.

  O algoritmo atinge o resultado representado na Fig. 2.23 depois de apenas 140 iterações, com um funcional minimizado com um valor da ordem de 10^{-3}.

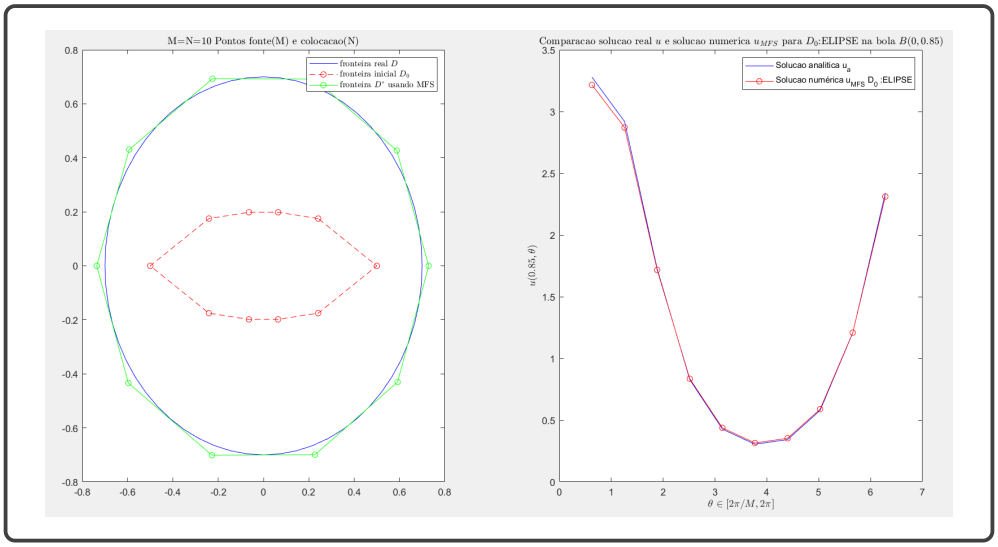

Figura 2.23: Domínio inicial Elipse (círculos vermelhos), reconstrução D^* e comparação das soluções analítica u_a e numérica u_{MSF} sobre o conjunto $\partial B(0, 0.85)$ para $M = N = 10$ no Teste 5.

2.3 Segundo Método de Resolução: Analise à Mudança de Forma aplicada à equação de Laplace

Na presente seção, outra maneira de abordar o problema inverso geométrico (2.3) será considerada, fazendo-se uso da análise de sensibilidade à mudança de forma através do método da velocidade. Por razões de simplicidade nos cálculos envolvidos no método, a equação de Laplace será utilizada para o problema inverso.

2.3.1 Reformulação Matemática do Problema para a Equação de Laplace

Nesta seção, é apresentado um método para abordar o problema inverso 2.3, utilizando a análise da mudança de forma através do método da velocidade. Para esse fim, utilizaremos o funcional de Kohn-Vogelius [44], uma vez que é conhecido que tal funcional conduz a procedimentos de otimização robustos e eficientes para problemas de reconstrução [1, 79].

O funcional de Kohn-Vogelius é definido como

$$J_{KV}(D) = \frac{1}{2}\int_{\Omega^*} |\nabla u^D(x) - \nabla u^N(x)|^2 \, dx, \tag{2.25}$$

onde $u^D \in H^1(\Omega^*)$ é a solução para o seguinte problema de valor de contorno com condições de Dirichlet

$$\begin{cases} -\Delta u^D = 0, & \text{em } \Omega^*, \\ u^D = g, & \text{sobre } \Gamma_0, \\ u^D = h, & \text{sobre } \Sigma \end{cases} \tag{2.26}$$

e $u^N \in H^1(\Omega^*)$ é a solução para o seguinte problema de valor de contorno com condições de Dirichlet e Neumann

$$\begin{cases} -\Delta u^N = b, & \text{em } \Omega^*, \\ \dfrac{\partial u^N}{\partial n} = f, & \text{sobre } \Gamma_0, \\ u^N = h, & \text{sobre } \Sigma. \end{cases} \tag{2.27}$$

Portanto, o objetivo consiste em minimizar o funcional (2.25) e obter

$$D^* = \underset{\Omega \in \mathcal{U}_{ad}}{\operatorname{argmin}} \, J_{KV}(D), \tag{2.28}$$

onde

$$\mathcal{U}_{ad} = \{D \subset \Omega | D \text{ é aberto, } \Sigma \in C^2 \text{ e } \Omega \backslash D \text{ é conexo } \}. \tag{2.29}$$

2.3.2 Análise à Sensibilidade à Mudança de Forma

Para resolver problemas de otimização de forma através do método da velocidade, é necessário conhecer o Gradiente de Forma (derivada de forma de primeira ordem ou simplesmente derivada de forma) e/ou a Hessiana de Forma (derivada de forma de segunda ordem). Em particular, o cálculo da Hessiana é mais complicado, ver [41]. Assim, a derivada de forma de primeira ordem será utilizada neste trabalho.

O procedimento para realizar a análise de sensibilidade à mudança de forma visa aproximar o funcional (2.25) da seguinte forma:

$$J_{KV}(D_t) \approx J_{KV}(D) + t dJ_{KV}(D; V), \tag{2.30}$$

onde V é um campo de velocidade de deformação, que está associado a uma transformação $T_t : \Omega \to \mathbb{R}^2$ definida por

$$T_t(x) = x + tV, \tag{2.31}$$

para $t \in [0, \epsilon[$, para algum ϵ positivo e $D_t = T_t(D)$.

Além disso, $dJ_{KV}(D; V)$ é a derivada de forma associada à velocidade V.

Note que a expressão (2.30), tem a forma de uma expansão de Taylor de ordem 1. Assim, é necessário encontrar uma expressão para a derivada de forma $dJ_{KV}(D; V)$, de modo a garantir a desigualdade

$$J(D_t, V) \leq J(D), \tag{2.32}$$

para todo $t > 0$ em um intervalo não vazio.

A derivada $dJ_{KV}(D; V)$ é dada por

$$dJ_{KV}(D; V) = -\frac{1}{2} \int_{\Sigma} \left[(\nabla u^D - \nabla u^N) \cdot n \right]^2 (V \cdot n) ds, \tag{2.33}$$

onde n é o vetor normal unitário externo sobre Σ [70].

Agora, considerando-se a velocidade dada pela expressão

$$V = h \left[(\nabla u^D - \nabla u^N) \cdot n \right]^2 n, \tag{2.34}$$

onde $h > 0$ é uma constante, a desigualdade desejada (2.32) é satisfeita, o que garante a minimização do funcional (2.25).

Portanto, o algoritmo proposto para resolver o problema inverso é como segue:

1. Considerar um chute inicial para a fronteira desconhecida D;

2. Resolver as EDPs (2.26) e (2.27) para o domínio D interior considerado, por meio do MSF;

3. Se o funcional (2.25) for menor que uma tolerância dada, finaliza-se o processo. Se não, vá para o passo seguinte 4;

4. Atualizar os pontos da fronteira incógnita D, com a expressão $x = x + tV$ e voltar ao passo 2.

2.3.3 Resultados Numéricos

A fim de minimizar o funcional (2.25), o algoritmo proposto na seção anterior foi codificado em MATLAB. Neste caso, não é utilizado nenhum programa da *toolbox* de otimização do MATLAB, pois a análise de sensibilidade à mudança de forma fornece uma direção de descida para os pontos localizados na fronteira desconhecida, que dependem, ao mesmo tempo da solução associada a cada novo sistema de EDPs que se forma.

Com o acima exposto, para este teste, será utilizado o seguinte exemplo:

$$\begin{cases} \Delta u = 0, & \text{em } \Omega^*, \\ u = \cos(\theta), & \text{sobre } \Gamma_0, \\ u = 0, & \text{sobre } \Sigma, \end{cases} \tag{2.35}$$

para o caso em que $\Omega = B(0,1)$ e o domínio alvo é $D = B(0, 0.75)$.

Uma vez que as matrizes associadas às soluções numéricas dadas pelo MSF apresentavam um elevado número de condicionamento, uma regularização Tikhonov de ordem 2 também foi utilizada e uma sub-rotina foi codificada em MATLAB para escolher cada parâmetro de regularização associado a tais matrizes, para um intervalo $\{10^{-10}, 10^{-9}, \ldots, 10^{-2}, 10^{-1}\}$ em cada iteração do algoritmo exposto acima. Além disso, esta sub-rotina é baseada no critério da curva L.

A Fig. 2.24 mostra o resultado da tentativa de reconstrução do domínio D considerado para 250 iterações. Foram utilizados 20 pontos fonte localizados na bola $B(0,2)$, e 20 pontos de colocação sobre a fronteira de Ω. Pode-se observar que, apesar do uso de uma regularização de Tikhonov de ordem 2 para MSF, o algoritmo não mostra um resultado desejado.

Se o número de pontos fonte e colocação fosse aumentado para 40 e mesmo que fosse testado com maiores distâncias dos pontos fonte em relação ao domínio Ω (pontos fonte na bola $B(0,4)$), apenas o número de iterações diminui em relação ao

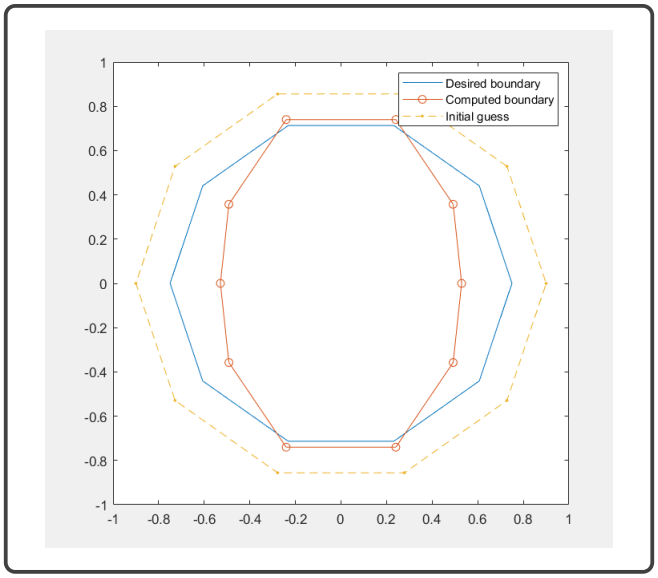

Figura 2.24: Fronteira inicial considerada (laranja), fronteira reconstruída (vermelha) e fronteira alvo em azul. para $M = N = 20$.

primeiro ensaio para 165 iterações para atingir uma tolerância próxima à anterior. Ainda assim, não obtemos uma solução adequada, ver Fig. 2.25.

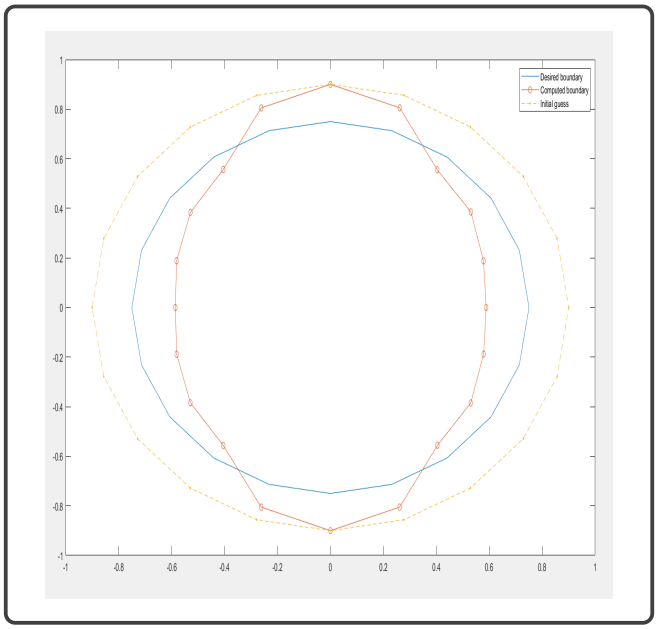

Figura 2.25: Fronteira inicial considerada (laranja), fronteira reconstruída (vermelha) e fronteira alvo em azul. para $M = N = 40$.

Uma das desvantagens deste método apresentado está na escolha de muitos parâmetros determinantes, como é o caso do tamanho do passo e da constante h, na expressão da velocidade escolhida. A velocidade presente na equação acima é também um vetor de escolha aberta, desde que satisfaça as condições do método, ver [70]. Portanto, a utilização deste método para este caso precisa ser estudado em

maior profundidade.

2.4 Conclusões para os Métodos Apresentados

Neste capítulo, o problema geométrico inverso, associado às equações de Helmholtz modificado e Laplace, que consiste na determinação de uma fronteira interna de um domínio circular, foi estudado com o MSF. Para o primeiro método, os primeiros resultados sugeriram a incorporação de um termo de regularização para obter soluções estáveis. Desta forma, foi observada a eficácia e eficiência do método baseado no MSF sem utilizar análise à mudança de forma. No segundo método, aplicado à equação de Laplace, para o mesmo tipo de problema inverso, observou-se uma das desvantagens do uso do método da velocidade, que envolve a escolha de vários parâmetros, portanto, este método precisa ser estudado com mais profundidade para este problema inverso geométrico.

Capítulo 3

Problema Inverso da Tomografia por Impedância Elétrica (EIT) para a Análise da Umidade em Estruturas

A umidade nas paredes é um problema sério, quer para a indústria da construção, quer especialmente nas estruturas históricas, não havendo neste caso a possibilidade de as demolir e substituir por novas estruturas. A umidade cria danos não só nas paredes mas também na saúde humana, uma vez que promove a progressão de doenças reumáticas ou alérgicas devido à formação de fungos nas paredes [4]. A razão mais comum da umidade nas paredes é a falta de isolamento adequado para separar a umidade do chão nos tijolos. A água contida nos tijolos sobe através dos poros da parede num processo denominado de capilaridade. A capilaridade é um fenômeno no qual as moléculas são atraídas eletroquimicamente para a superfície da parede, permitindo que a água se mova verticalmente através de poros de certos tamanhos, apesar da força da gravidade na direção oposta [27].

Existem vários métodos a secagem de paredes (secagem com barreiras impermeáveis, injeção de produtos hidrofugantes, etc.), no entanto, antes de utilizar qualquer um destes métodos, é necessário diagnosticar exaustivamente tanto a extensão da umidade como a sua intensidade. Atualmente, diversos métodos são utilizados para medir a concentração de umidade em paredes, que geralmente podem ser divididos em dois grupos: métodos destrutivos e não-destrutivos.

Entre os métodos destrutivos, podemos citar, por exemplo, o método do peso seco (*dry-weight method*) e o método do carboneto (*carbide method*), para mais detalhes de tais métodos, ver [34]. Estes métodos não são recomendados, pois, envolvem a coleta de uma amostra da parede, o que significa danificá-la e, no caso de uma estrutura histórica, poderia ser inaceitável. Assim, os métodos não destrutivos são mais convenientes e têm um maior valor de aplicação, veja [61, 63].

Os métodos não destrutivos incluem, por exemplo, métodos baseados em termovisão [40] ou ultra-som [67]. A desvantagem do uso da termovisão é principalmente a impossibilidade de penetrar na superfície da estrutura em investigação. A abordagem ultra-sonográfica é de pouca utilidade devido à alta porosidade dos materiais contendo células (poros) cheias de ar ou água. Em resumo, o principal problema de tais métodos é que eles não permitem a possibilidade de fornecer informações sobre a distribuição espacial da água contida dentro do muro.

O método não destrutivo utilizado neste trabalho faz parte do conjunto de métodos baseados na tomografia por impedância elétrica, que é uma técnica que visa reconstruir a distribuição da condutividade elétrica dentro do objeto de teste, a partir de medidas de corrente ou voltagem no exterior do domínio de definição do problema [34]. Assim, este método permite a determinação espacial não invasiva do nível de umidade.

3.1 O Método da Tomografia por Impedância Elétrica

O objetivo do método de EIT é descobrir a distribuição da condutividade interna do objeto baseado nas medidas de voltagem e corrente elétrica na superfície do corpo investigado. A seguir, uma breve descrição da técnica de EIT, retirada do artigo [9] de Jan Sikora e Katarzyna Biernat, é exposta.

Em geral, considera-se uma disposição de eletrodos em torno do objeto em estudo, para depois aplicar a corrente elétrica, medir a voltagem e reconstruir a distribuição da condutividade elétrica dentro do mesmo. Além disso, é possível reconstruir essa distribuição de corrente aplicando os potenciais elétricos aos eletrodos e depois lendo a corrente elétrica, devido à dualidade entre voltagem e corrente.

A relação entre os potenciais e a distribuição da condutividade é expressa por

$$v = T(j, \gamma), \tag{3.1}$$

onde

v: potencial elétrico,

j: densidade de corrente,

γ: distribuição da condutividade elétrica,

T: Operador não linear baseado na equação de Laplace.

A expressão acima representa de forma abstrata o problema direto de EIT, que é interpretado como: dada uma disposição de eletrodos submetidos a uma corrente

e se a distribuição das condutividades elétricas internas do corpo é conhecida, é possível obter os potenciais elétricos.

De modo a aumentar o número de informações sobre a distribuição da condutividade, os dados são tomados de diferentes eletrodos. Esta relação é descrita pela seguinte forma matricial:

$$V = A\boldsymbol{\gamma}, \tag{3.2}$$

onde

V: vetor de potenciais medidos,

$\boldsymbol{\gamma}$: vetor condutividade elétrica,

A: matriz associada ao operador T.

Então, o problema inverso pode ser escrito como

$$\boldsymbol{\gamma} = A^{-1}V. \tag{3.3}$$

A expressão (3.3) representa de forma abstrata o problema inverso de EIT, o qual visa encontrar a distribuição de condutividade conhecendo a intensidade da corrente elétrica e tendo medido os potenciais elétricos. Este problema inverso é geralmente mal posto, pois, se a condutividade elétrica é considerada anisotrópica, não existe solução única [71]. Os métodos numéricos são aplicados para obter uma solução apropriada, para maiores detalhes ver [9].

3.2 Formulação Matemática do Problema de EIT para a Determinação da Superfície Livre da Água

De modo a estudar de forma prática o problema do EIT para a análise da umidade em estruturas, nesta seção aborda-se o problema que consiste na determinação da fronteira livre de uma quantidade de água contida em um recipiente em cujas paredes foram colocados eletrodos. Para o caso bidimensional, ver Fig. 3.1.

Em primeiro lugar, será apresentado o modelo matemático do problema direto, o problema inverso associado, seguido de dois testes para a resolução do problema inverso.

Considere-se o conjunto de eletrodos $E = \{e_1, \ldots, e_k\}$ colocado em duas paredes, como mostra a Fig. 3.1. Também, para garantir a solução única do problema de EIT [5, 74, 75], considera-se a condutividade elétrica isotrópica, constante em partes, isto

é

$$\gamma(x) = \begin{cases} \gamma_1, \text{ em } \Omega_1, \\ \gamma_2, \text{ em } \Omega_2. \end{cases} \tag{3.4}$$

Na prática, aplicam-se potenciais elétricos nos eletrodos E. Por conseguinte, o

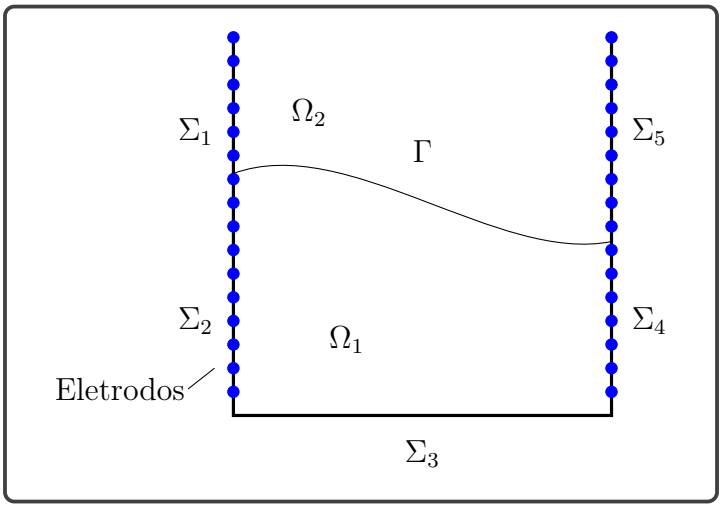

Figura 3.1: Representação do problema Direto.

potencial elétrico (voltagem) em $\Omega = \Omega_1 \cup \Omega_2$ satisfaz a seguinte EDP:

$$\begin{cases} -div(\gamma(x)\nabla u) = 0, & \text{em } \Omega, \\ u = f(x), & \text{sobre } E, \\ u = 0, & \text{sobre } \Sigma_3. \end{cases} \tag{3.5}$$

A equação acima é obtida a partir das equações de Maxwell (ver [16]).

Os potenciais aplicados produzem uma densidade de corrente nos eletrodos ($q_i = \gamma \frac{\partial u}{\partial n}(e_i)$). Esta corrente pode ser medida nos eletrodos. Portanto, o problema inverso visa reconstruir a fronteira livre Γ, através dos potenciais nos eletrodos $f(e_i)$, que são dados, e as medidas correspondentes q_i no conjunto de eletrodos E.

A formulação variacional do problema (3.5) produz as seguintes condições de continuidade

$$u_1|_\Gamma = u_2|_\Gamma \tag{3.6}$$

e salto da derivada normal das soluções

$$\gamma_1 \left.\frac{\partial u_1}{\partial n}\right|_\Gamma = \gamma_2 \left.\frac{\partial u_2}{\partial n}\right|_\Gamma, \tag{3.7}$$

na fronteira Γ, onde u_1, u_2 são soluções em Ω_1, Ω_2, respectivamente.

Para resolver o problema inverso, o procedimento geral será o seguinte:

1. Uma curva inicial com forma qualquer será assumida como a fronteira Γ.

2. O problema direto (3.5) será resolvido usando também as condições de contorno (3.6) e (3.7).

3. As medidas ou dados $(q_i)_{i=1}^k$ serão comparadas com as obtidas numericamente $\left(\frac{\partial u_{MSF}}{\partial n}\right)$ dadas pelo MSF, através de um funcional, que dependerá da forma da curva desconhecida, aproximando tal curva da fronteira alvo pela minimização deste funcional, processo que resultará em um problema de otimização não-linear.

Deve-se assinalar que o modelo (3.5) também pode ser proposto considerando uma condição de Neumann em vez da condição de Dirichlet, o que significaria ter aplicado uma corrente nos eletrodos e depois ter feito medições dos potenciais (voltagens). Também, uma questão importante a mencionar é a influência da distribuição de eletrodos na superfície da parede e a distância entre as paredes. No artigo [65], foram realizados testes para a análise da umidade nas estruturas utilizando o EIT, considerando 16 e 32 eletrodos distribuídos de duas formas diferentes numa parede. Com base nos resultados desse estudo, este trabalho considerará o esquema que foi apresentado na Figura 3.1 considerando $k = 32$ eletrodos, com 16 eletrodos igualmente espaçados colocados em cada parede e a distância entre as paredes, igual ou inferior à altura delas, uma vez que estas considerações asseguraram a obtenção de melhores resultados em [65]. Mais informações serão fornecidas nos resultados numéricos apresentados mais adiante neste trabalho.

Na seção seguinte apresentam-se duas estratégias que foram utilizadas para resolver este problema inverso. Ambas se baseiam no método descrito na seção 2.2, pois é o método que obteve os melhores resultados para os problemas de reconstrução geométrica.

3.3 Primeira Proposta de Resolução

Como primeira estratégia, considera-se a minimização de um funcional multiobjetivo, onde os cinco primeiros termos envolvem as condições de contorno do problema direto, que permitem a satisfação das soluções do problema direto e os demais termos comparam os valores numéricos com os dados reais medidos para o controle dos pontos de contorno a serem reconstruídos. Desta forma, o funcional proposto é o seguinte

$$
\begin{aligned}
J_1(a,b,g) = &\|u_1 - f\|^2_{L^2(\Sigma_2^g \cup \Sigma_4^g)} + \|u_2 - f\|^2_{L^2(\Sigma_1^g \cup \Sigma_5^g)} \\
&+ \|u_1\|^2_{L^2(\Sigma_3)} + \|u_1 - u_2\|^2_{L^2(G(g))} \\
&+ \left\| \gamma_1 \frac{\partial u_1}{\partial n} - \gamma_2 \frac{\partial u_2}{\partial n} \right\|^2_{L^2(G(g))} \\
&+ \sum_{i \in \{2,4\}} \left\| \gamma_1 \frac{\partial u_1}{\partial n} - q_{E_i} \right\|^2_{L^2(\Sigma_i^g)} \\
&+ \sum_{j \in \{1,5\}} \left\| \gamma_2 \frac{\partial u_2}{\partial n} - q_{E_j} \right\|^2_{L^2(\Sigma_j^g)},
\end{aligned}
\tag{3.8}
$$

onde as fronteiras $\{\Sigma_i^g\},\ i \in \{1,2,4,5\}$, dependem do gráfico $G(g) = \{(x, g(x)) | x \in X\}$ da função injetora $g : X \to Y;\ X,\ Y \subset \mathbb{R}$, sendo X um conjunto finito de pontos. Além disso, os valores q_{E_i} representam as medições de corrente correspondentes ao conjunto de eletrodos $E_i \subset \Sigma_i,\ i \in \{1,2,4,5\}$, sobre as paredes e $E = E_1 \cup E_2 \cup E_4 \cup E_5$.

Note que o funcional J_1 depende, além dos pontos que compõem a fronteira desconhecida g, dos coeficientes $a = (a_1, \ldots, a_M),\ b = (b_1, \ldots, b_M)$ associados às soluções numéricas dadas pelo MSF

$$
u_1 = \sum_{i=1}^{M} a_i \phi(x, \xi_i),\ \text{em } \Omega_1, \tag{3.9}
$$

$$
u_2 = \sum_{i=1}^{M} b_i \phi(x, \xi_i),\ \text{em } \Omega_2. \tag{3.10}
$$

Além disso, os primeiros cinco termos envolvem os coeficientes associados às soluções numéricas para o problema, enquanto os dois últimos termos comparam as densidades de corrente elétrica obtidas numericamente com as medidas (dados).

O objetivo do funcional apresentado é, portanto, encontrar ao mesmo tempo os coeficientes para as soluções numéricas u_1, u_2 e reconstruir a fronteira alvo Γ (aproximada pela função g).

Uma vez que a função apresentada envolve um processo de otimização com vários alvos e após os resultados mostrados na seção 2.2, onde o mesmo tipo de função objetivo é utilizada, optou-se por incorporar um termo de regularização centrado nos pontos que compõem o gráfico de g (pontos da fronteira a ser reconstruída). Além disso, considerando a possibilidade de ter soluções numéricas regulares, foram adicionados mais dois termos de regularização, no total foram utilizados 3 novos termos. Assim, o funcional final proposto é dado por

$$
J_{r_1}(a,b,g) = J_1(a,b,g) + \lambda_1 \|a\|^2 + \lambda_2 \|b\|^2 + \lambda_3 \|g\|^2, \tag{3.11}
$$

onde λ_1, λ_2, λ_3 são parâmetros de penalização e g é a imagem da função g. Como para cada teste numérico no funcional, apenas os pontos no eixo vertical são movidos para qualquer distribuição de pontos na fronteira desconhecida, apenas é suficiente focar nos pontos da imagem de g na regularização.

Seguem-se alguns resultados obtidos utilizando o funcional (3.11).

3.3.1 Resultados Numéricos

Em todos os exemplos numéricos que serão mostrados a seguir serão considerados $k = 32$ eletrodos colocados 16 em cada parede. Para fins didáticos, as condutividades elétricas da água e do ar serão consideradas respectivamente como $\gamma_1 = 1$ e $\gamma_2 = 0.01$.

Exemplo 3.3.1 (Teste sem ruído)**.** Como primeiro teste, considera-se um exemplo com solução analítica. Se considerarmos o problema (3.1) com fronteira alvo Γ sendo a linha constante $y = s$ e a condição de Dirichlet sobre a fronteira é dada por

$$f(x) = \begin{cases} -\dfrac{\gamma_2}{\gamma_1}(y+1), & \text{se } y \leq s, \\ (y+1) - \left(\dfrac{\gamma_2}{\gamma_1}\right)(s+1), & \text{caso contrário.} \end{cases} \tag{3.12}$$

Ademais, se consideramos o domínio do problema como $\Omega_1 \cup \Omega_2 = (-0.5, 0.5) \times (-1, 1)$, com base nas considerações anteriores, a solução analítica para o problema direto (3.5) é dada por

$$u^{an}(x,y) = \begin{cases} u_1^{an} = -\dfrac{\gamma_2}{\gamma_1}(y+1), & \text{em } \Omega_1, \\ u_2^{an} = (y+1) - \left(\dfrac{\gamma_2}{\gamma_1}\right)(s+1), & \text{em } \Omega_2. \end{cases} \tag{3.13}$$

Levando-se em consideração o retângulo fictício $(-1, 1) \times (-2, 2)$ para a distribuição dos pontos fonte $M = 36$ e considerando $N = 78$ pontos de colocação sobre $\Sigma_1 \cup \Sigma_2 \cup \Sigma_4 \cup \Sigma_5$, onde 32 deles estão distribuídos, 16 em cada lado das paredes como mostrado nas Figs. 3.1 e 13 eletrodos sobre a fronteira desconhecida $g(x)$, sendo esta considerada inicialmente como $g_0(x) = s = 0.1$, $x \in X = (-0.5, 0.5)$. Os parâmetros de penalização também foram considerados sendo $\lambda_1 = 10^{-10}$, $\lambda_2 = 10^{-5}$, $\lambda_3 = 0$, escolhidos pelo critério de prova e erro na faixa de $\{10^{-12}, 10^{-1}, 0\}$ e como chute inicial para os coeficientes associados às soluções numéricas sendo $a = b = (1, \ldots, 1) \in \mathbb{R}^{36}$. O funcional (3.11) foi minimizado utilizando o algoritmo de pontos interiores da *toolbox* de otimização do MATLAB.

O resultado obtido é apresentado na Fig. 3.2, onde uma boa reconstrução pode ser observada. No entanto, cerca de 1000 iterações foram necessárias para que o funcional atingisse um valor de ordem 10^{-6}.

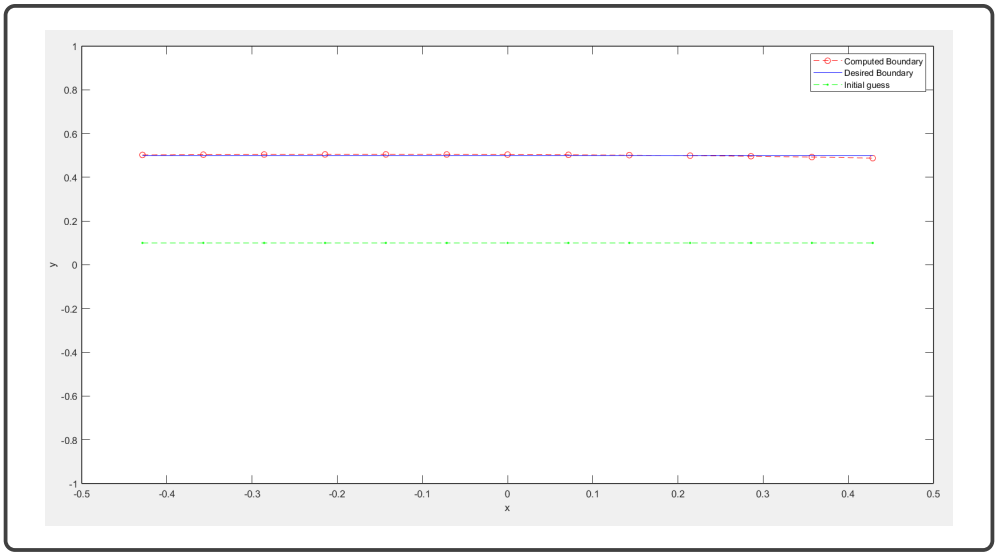

Figura 3.2: Fronteira inicial considerada (verde), fronteira reconstruída (vermelho) e fronteira alvo em azul.

Exemplo 3.3.2 (Teste com ruído 1%)**.** Tendo-se em conta as mesmas considerações do caso anterior, desta vez adiciona-se ruído gaussiano branco (WGN) de 1% de intensidade às medições. A Fig. 3.3 mostra uma aproximação menos precisa que a anterior (caso sem ruído) com 1000 iterações. A função custo atinge valores de ordem 10^{-4}.

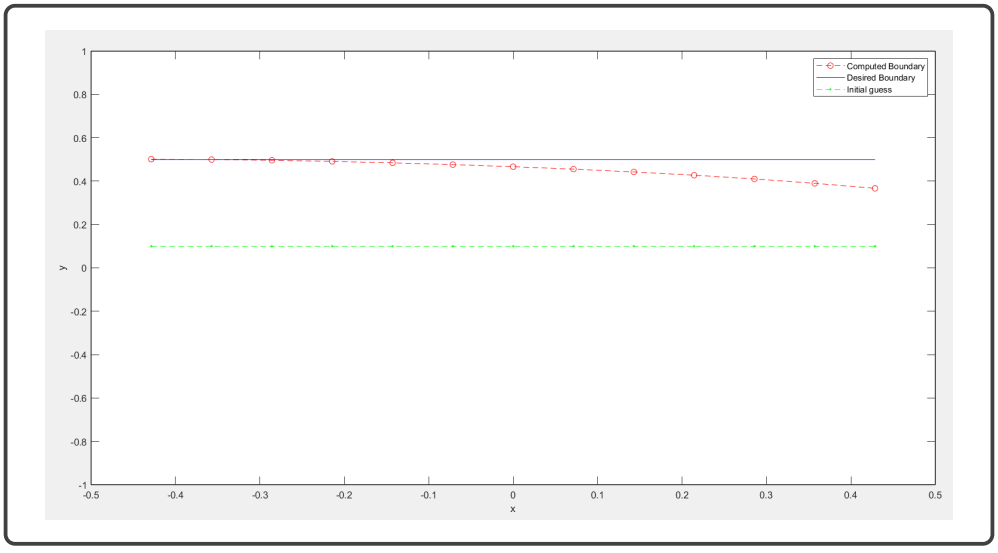

Figura 3.3: Fronteira inicial considerada (verde), fronteira reconstruída (vermelho) e fronteira alvo em azul, para dados poluídos com 1% de WGN.

Exemplo 3.3.3 (Teste com ruído 5%)**.** Como é de se esperar, se considerarmos o caso de dados poluídos com um nível de 5% de ruído, o algoritmo anterior resulta numa reconstrução pior da fronteira alvo, considerando-se o mesmo limite de 1000 iterações e a função objetivo atinge valores da ordem 10^{-2}, ver Fig. 3.4.

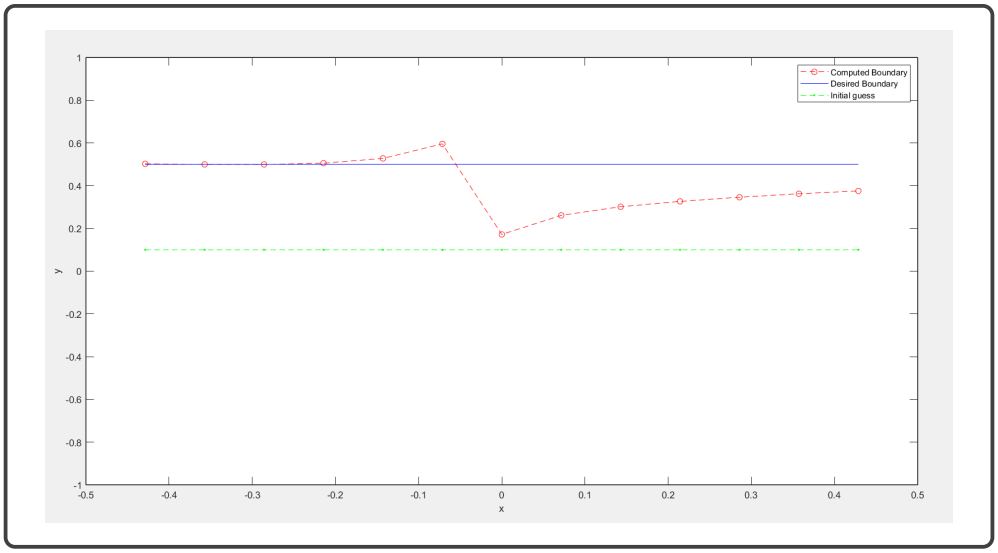

Figura 3.4: Fronteira inicial considerada (verde), fronteira reconstruída (vermelho) e fronteira alvo em azul, para dados poluídos com 5% de WGN.

O algoritmo acima apresenta certa dependência do chute inicial, mesmo para dados sem ruídos. Assim, para superar algumas desvantagens da utilização do funcional (3.11) utilizado nos exemplos acima, na seção seguinte, propomos a utilização de um funcional que depende apenas dos pontos da fronteira a ser reconstruída, este procedimento diminui o custo computacional do algoritmo utilizado, como será demonstrado nos experimentos.

3.4 Segunda Proposta de Resolução

A seguir, será utilizado um novo funcional que depende apenas dos pontos da fronteira desconhecida. Isto é, o código implementado em MATLAB primeiro calcula as soluções numéricas (3.9), (3.10), onde foi incorporado um termo de regularização de Tikhonov de ordem 1 para cada uma das soluções, onde uma sub-rotina escolhe, utilizando o critério da curva L, os parâmetros de penalização no conjunto $\{10^{-10}, 10^{-9}, \ldots, 10^{-6}, 10^{-5}\}$, em cada iteração, o que garante que em cada iteração sejam escolhidos os melhores parâmetros para o processo de otimização. Depois encontra as derivadas normais correspondentes sobre $g(x)$, finalmente minimiza o funcional

$$\begin{aligned} J_2(g) = &\sum_{i\in\{2,4\}} \left\| \gamma_1 \frac{\partial u_1}{\partial n} - q_{E_i} \right\|^2_{L^2(\Sigma_i^g)} \\ &+ \sum_{j\in\{1,5\}} \left\| \gamma_2 \frac{\partial u_2}{\partial n} - q_{E_j} \right\|^2_{L^2(\Sigma_j^g)}. \end{aligned} \tag{3.14}$$

Novamente, como na seção anterior, para obter melhores resultados, um termo de regularização foi introduzido. Portanto, o funcional final proposto é

$$J_{r_2}(g) = J_2(g) + \lambda_F \|P^{(m)} g(X)\|^2. \tag{3.15}$$

Onde $m = 0, 1, 2$ é a ordem da matriz associada à regularização de Tikhonov. Isto é $P^{(0)} = I$, a matriz $P^{(1)}$ para este funcional é de dimensão $(n-1) \times (n)$ da forma

$$P^{(1)} = \begin{pmatrix} -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & -1 & 1 \end{pmatrix} \in \mathbb{R}^{(n-1)\times(n)}, \tag{3.16}$$

onde n é o cardinal de $g(X)$, ou seja, o número de pontos que é usado sobre a fronteira desconhecida a ser reconstruída e

$$P^{(2)} = \begin{pmatrix} -1 & 2 & 1 & 0 & \dots & 0 \\ 0 & -1 & 2 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & -1 & 2 & 1 \end{pmatrix} \in \mathbb{R}^{(n-2)\times(n)}. \tag{3.17}$$

3.4.1 Resultados Numéricos

A seguir, apresenta-se brevemente o método do conjunto ativo, que é utilizado nos resultados numéricos para o caso bidimensional proposto.

3.4.1.1 Método do Conjunto Ativo

O método do conjunto ativo (*Active Set* [50]), consiste basicamente em: primeiro escolher um conjunto ativo (pontos que satisfaçam a igualdade em todas as restrições do problema mesmo que sejam desigualdades), este conjunto também é chamado de conjunto de trabalho; depois, é aproximada uma solução para o problema no conjunto ativo e, em seguida, nesse ponto, um novo conjunto ativo é escolhido.

Uma vantagem deste método é que, uma vez que em cada iteração são consideradas as restrições ativas, o problema a ser resolvido no conjunto ativo normalmente tem poucas restrições e pode ser resolvido rapidamente. Além disso, em muitos casos, os conjuntos ativos variam pouco em cada passo iterativo, tornando tal método

eficiente.

Por outro lado, este método pode tornar-se mais lento perto da solução ótima e portanto, computacionalmente lento devido à possibilidade de muitas das expressões nas condições de Karush-Kuhn-Tucker poderem ser altamente não lineares e difíceis de resolver.

No entanto, este método apresenta resultados mais robustos em comparação com o método dos pontos interiores [23, 24], ou seja, converge com maior velocidade com dados fracos (dados com maior ruído). Para mais detalhes sobre este método, ver [26].

3.4.1.2 Exemplos para o Caso Bidimensional

Nesta seção, apresenta-se alguns resultados produzidos usando o funcional (3.15). Em todos os exemplos a seguir, por questões de simplicidade na implementação do algoritmo, será utilizado agora o domínio quadrado $\Omega_1 \cup \Omega_2 = (0,1) \times (0,1)$.

Nos três primeiros exemplos apresentados (Testes: 1,2,3), o mesmo exemplo analítico com solução analítica, usado na seção 3.3 será considerado adaptado a este novo domínio, ou seja, considera-se agora

$$f(x) = \begin{cases} -\dfrac{\gamma_2}{\gamma_1} y, & \text{se } y \leq s, \\ y - \left(\dfrac{\gamma_2}{\gamma_1}\right) s, & \text{caso contrário.} \end{cases} \tag{3.18}$$

Com solução analítica

$$u^{an}(x,y) = \begin{cases} u_1^{an} = -\dfrac{\gamma_2}{\gamma_1} y, & \text{em } \Omega_1, \\ u_2^{an} = y - \left(\dfrac{\gamma_2}{\gamma_1}\right) s, & \text{em } \Omega_2. \end{cases} \tag{3.19}$$

Também vamos considerar $s = 0.5$. São utilizados 56 pontos fonte e 96 pontos de colocação. Onde os pontos fonte são colocados sobre o quadrado fictício de raio $R = 2$ com centro em $(0.5, 0.5)$ e, como no método precedente, 16 pontos de colocação pertencem a cada parede ($\Sigma_1 \cup \Sigma_2$ e $\Sigma_4 \cup \Sigma_5$) e 16 pontos sobre a fronteira desconhecida $g(x)$, $X =]0,1[$, $g(X) \in]0,1[$. A faixa de escolha para os parâmetros de regularização de Tikhonov associados ao MSF é $\{10^{-10}, 10^{-9}, \ldots, 10^{-6}, 10^{-5}\}$, cuja escolha é dada pelo critério da curva L em cada iteração.

Cabe mencionar que, todos os exemplos que se seguem foram testados utilizando o método dos pontos interiores e o método do conjunto ativo; no entanto, o método dos pontos interiores demonstrou exigir mais iterações em comparação com o método do conjunto ativo para atingir um valor final da mesma ordem do funcional de custo. Por conseguinte, a seguir apenas serão apresentados os resultados relativos ao uso

do método do conjunto ativo (*Active Set*) para a minimização do funcional, através da *toolbox* de otimização do MATLAB.

Convém esclarecer que nos 3 primeiros exemplos, a sensibilidade do método proposto para a variável inicial é mostrada mudando a forma da superfície inicial objetivo $g_0(x)$ utilizada. Portanto, o parâmetro de regularização λ_F da função objetivo (3.15) para o termo associado aos pontos de colocação na curva desconhecida $g(x)$ é o mesmo em todos esses exemplos ($\lambda_F = 5$), escolhido pelo critério de prova e erro na faixa de $\{10^{-2}, 10^{-1}, 0, 1, 2, 3, 4, 5, \ldots, 20\}$ e $P^{(1)}$.

Exemplo 3.4.1 (Teste 1)**.** Como primeiro teste é utilizada como estimativa inicial a fronteira $g_0(x) = 0.1$, $\forall x \in X$ constante.

Teste 1 sem ruído: A Fig. 3.5 mostra o resultado obtido. Note-se que o novo funcional utilizado, junto com o método do conjunto ativo, requereu apenas 49 iterações para realizar a reconstrução mostrada.

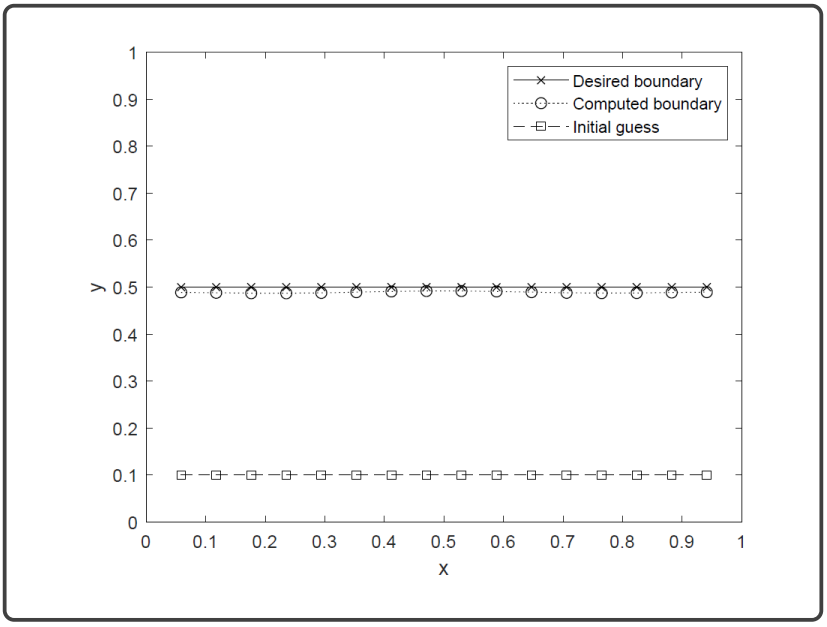

Figura 3.5: Fronteira inicial considerada (□), fronteira reconstruída (∘) e fronteira alvo (×).

Por outro lado, a Fig. 3.6 mostra um gráfico com a evolução do funcional utilizado em relação ao número de iterações. Uma diminuição da velocidade de convergência pode ser percebida a partir da iteração 40, que era esperada de acordo com o exposto na seção 3.4.1.1, onde foi apresentado o método do conjunto ativo.

Teste 1 com ruído: Agora vamos estudar a robustez do método utilizado, adicionando ruído Gaussiano branco (WGN). A Fig. 3.7 mostra o resultado para 1% de ruído sobre os dados, enquanto a Fig. 3.8 mostra o resultado correspondente para 5% de aumento de ruído.

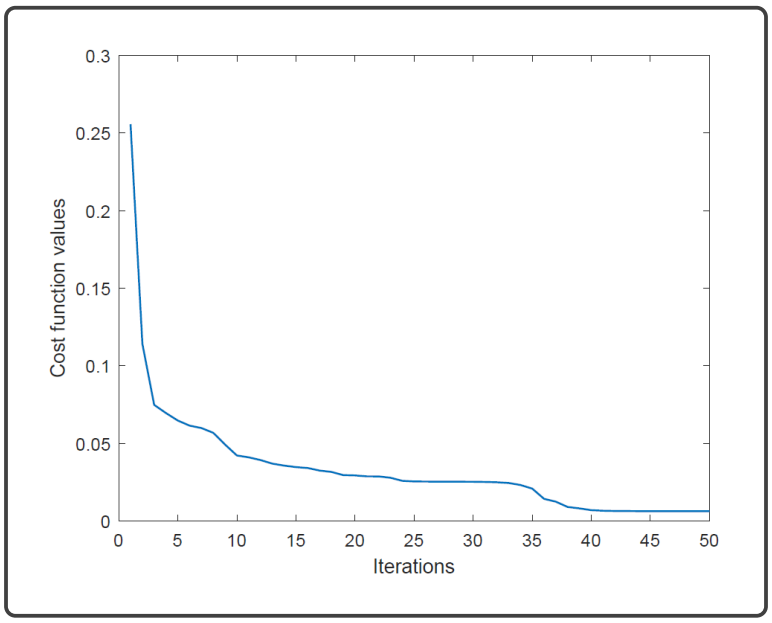

Figura 3.6: Função custo vs Iterações.

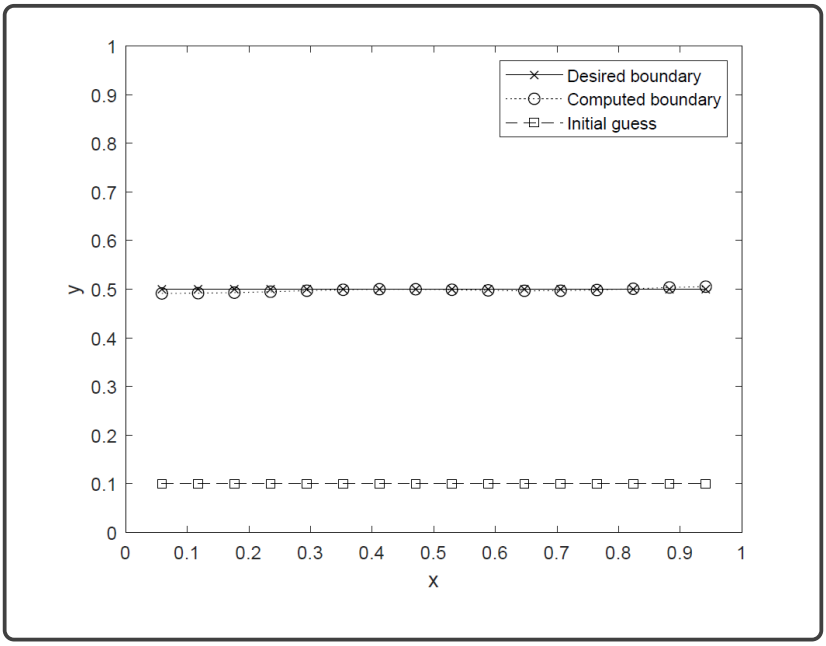

Figura 3.7: Fronteira inicial considerada (□), fronteira reconstruída (∘) e fronteira alvo (×) para dados poluídos com 1% de WGN.

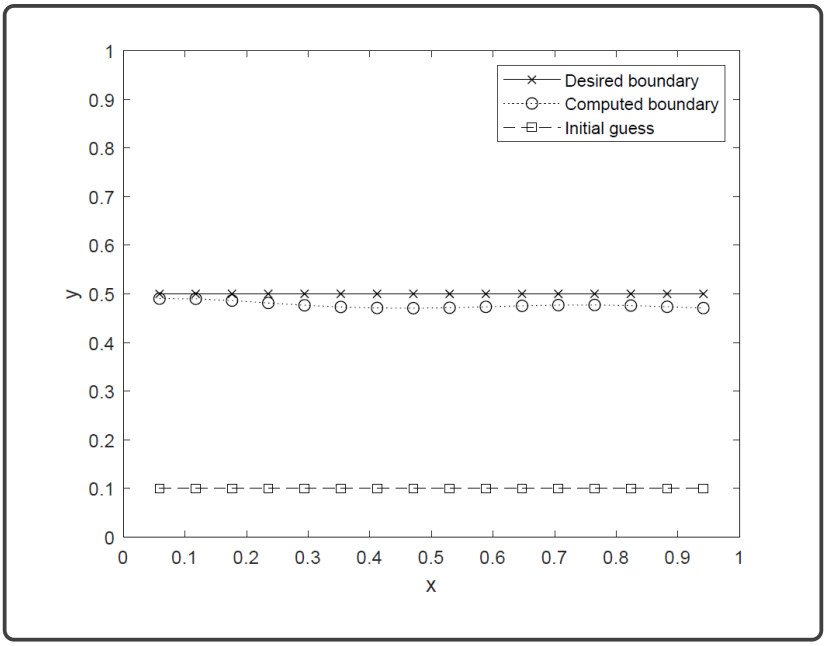

Figura 3.8: Fronteira inicial considerada (□), fronteira reconstruída (∘) e fronteira alvo (×) para dados poluídos com 5% de WGN.

Como esperado, à medida que o ruído aumenta, é necessário aumentar o número de iterações para conseguir as reconstruções desejadas, ver a Tabela 3.1, onde, para além do número de iterações, mostra o valor final do funcional e o erro entre a superfície alvo e a reconstrução. Note-se que, ao incorporar 1% de ruído, são necessárias mais iterações para obter um erro próximo ao teste sem incorporação de ruído.

, conforme mostrado na Tabela 3.1.

Tabela 3.1: Número de iterações, valor funcional de custo e erro de reconstrução para o Teste 1.

WGN %	Iterações	J_2	$\|s - g(x)\|_{L^2(\Gamma)}$
0	49	0.00632996	0.044716
1	68	0.0703142	0.038997
5	57	0.197282	0.095015

Exemplo 3.4.2 (Teste 2)**.** Neste ensaio é considerada uma estimativa inicial para a fronteira de forma diferente, dada por

$$g_0(x) = 0.2 + 0.1 * \sin 2\pi x, \ x \in]0, 1[. \tag{3.20}$$

Teste 2 sem ruído: O método atual atinge o resultado mostrado na Fig. 3.9 com apenas 48 iterações.

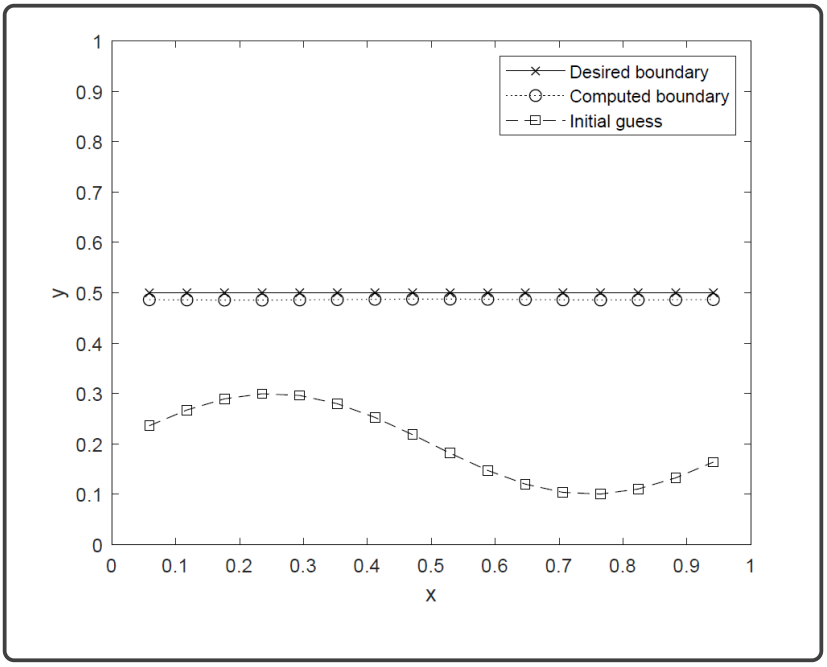

Figura 3.9: Fronteira inicial considerada (□), fronteira reconstruída (∘) e fronteira alvo (×).

Teste 2 com ruído: Neste caso, o ruído branco de 1% e 5% também é adicionado, com resultados mostrados nas Figuras 3.10, 3.11, respectivamente.

Na Tabela 3.2 é possível observar uma situação semelhante ao Teste 1. A pior reconstrução é dada pelo nível de ruído mais alto.

Tabela 3.2: Número de iterações, valor funcional de custo e erro de reconstrução para o Teste 2.

WGN %	Iterações	J_2	$\|s - g(x)\|_{L^2(\Gamma)}$
0	31	0.00633001	0.04332
1	36	0.0061555	0.06517
5	35	0.0621342	0.070894

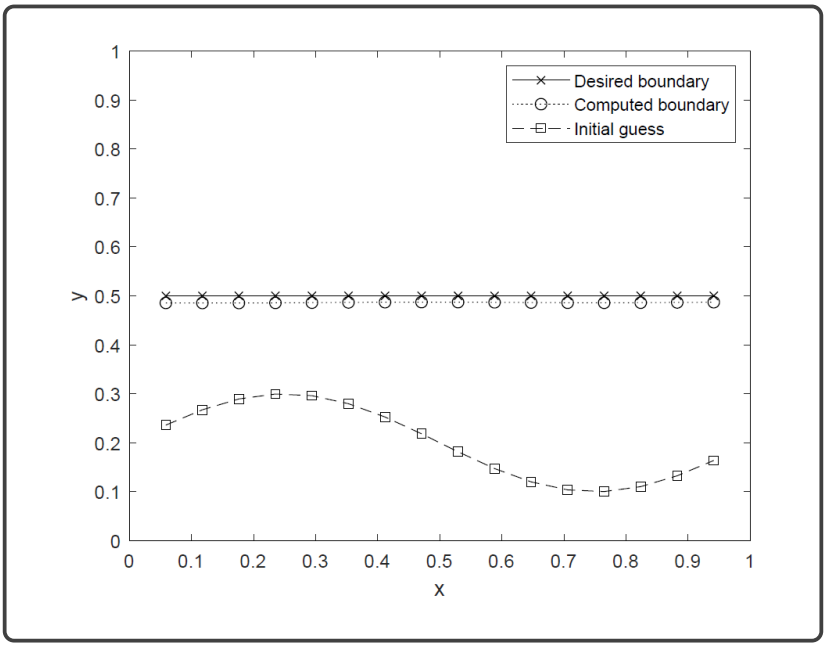

Figura 3.10: Fronteira inicial considerada (□), fronteira reconstruída (∘) e fronteira alvo (×) para dados poluídos com 1% de WGN.

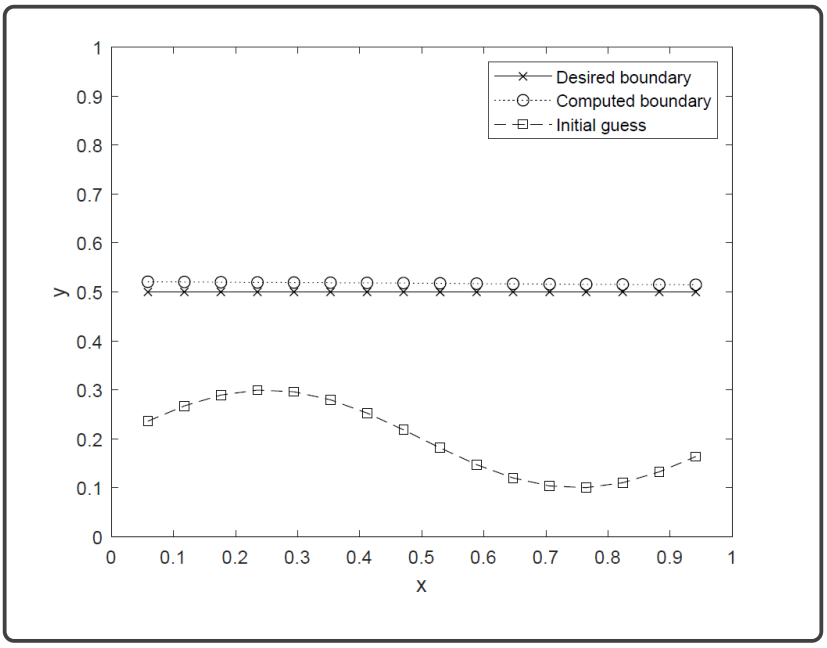

Figura 3.11: Fronteira inicial considerada (□), fronteira reconstruída (∘) e fronteira alvo (×) para dados poluídos com 5% de WGN.

Exemplo 3.4.3 (Teste 3)**.** Neste ensaio é considerada uma estimativa inicial para a fronteira dada por

$$g_0(x) = 0.2 + 0.1 * \sin 16\pi x,\ x \in]0, 1[. \tag{3.21}$$

Teste 3 sem ruído: O algoritmo atinge o resultado mostrado na Fig. 3.12 com 24 iterações.

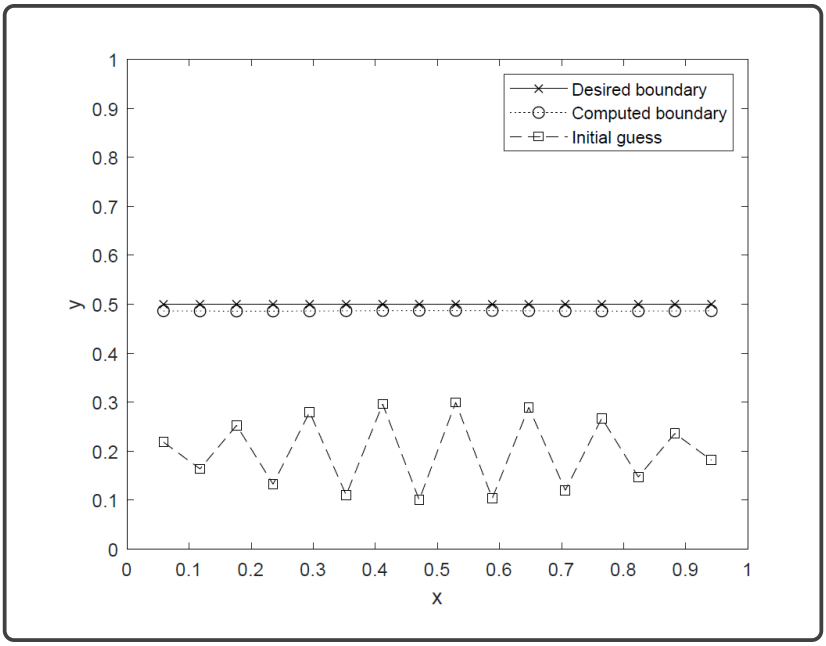

Figura 3.12: Fronteira inicial considerada (□), fronteira reconstruída (∘) e fronteira alvo (×).

Teste 3 com ruído: as Figuras 3.13 e 3.14 mostram os resultados obtidos pela incorporação dos níveis de ruído 1% e 5%.

A Tabela 3.3 mostra o comportamento dos resultados com tais níveis de ruído adicionados.

Tabela 3.3: Número de iterações, valor funcional de custo e erro de reconstrução para o Teste 3.

WGN %	Iterações	J_2	$\|s - g(x)\|_{L^2(\Gamma)}$
0	24	0.00636919	0.054298
1	23	0.0641294	0.104858
5	42	0.214811	0.122469

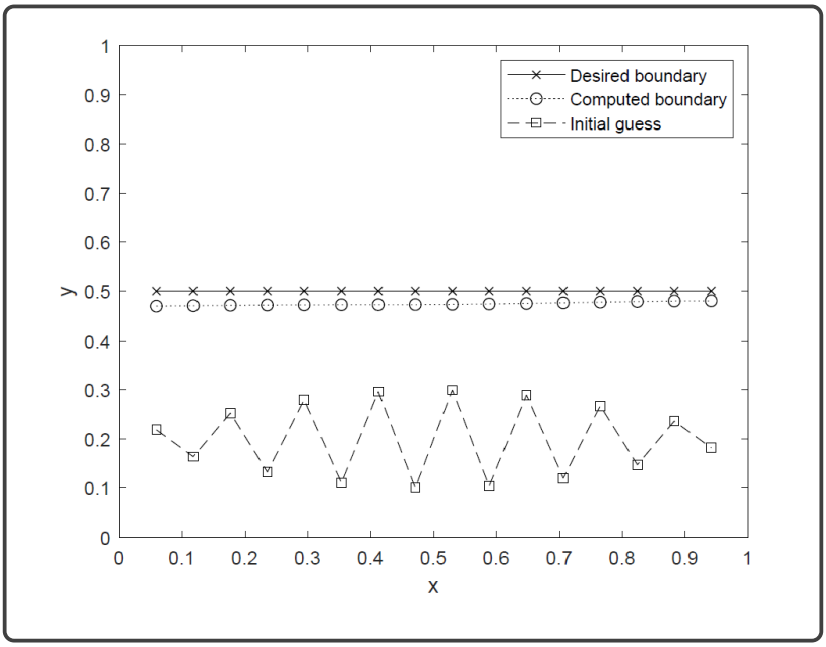

Figura 3.13: Fronteira inicial considerada (□), fronteira reconstruída (∘) e fronteira alvo (×) para dados poluídos com 1% de WGN.

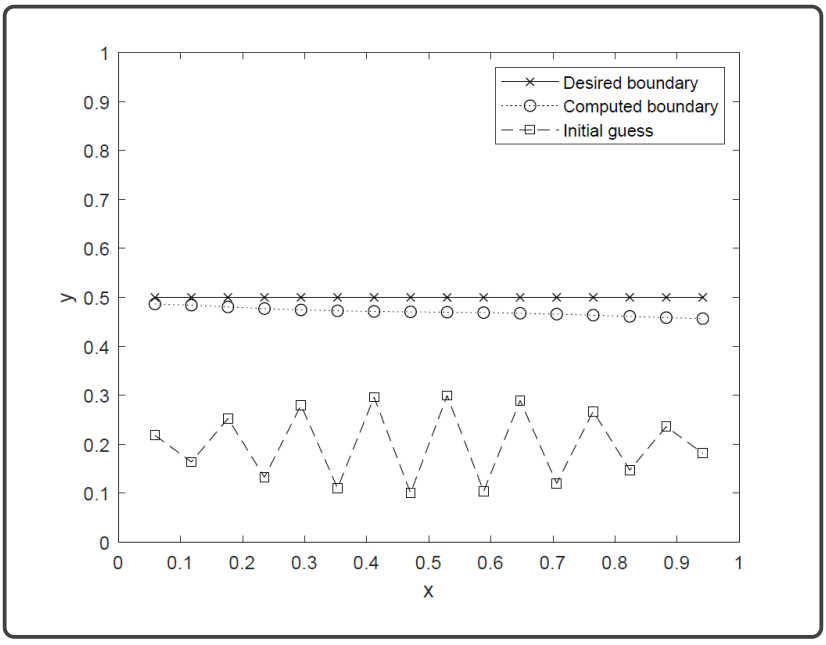

Figura 3.14: Fronteira inicial considerada (□), fronteira reconstruída (∘) e fronteira alvo (×) para dados poluídos com 5% de WGN.

Exemplo 3.4.4 (Teste 4)**.** A fim de se realizar mais um teste para a eficácia do método proposto, serão criados dados de medição fictícios para uma determinada fronteira alvo. Mais especificamente, será realizado o seguinte procedimento:

- uma superfície arbitrária Γ será considerada fixada como superfície alvo a reconstruir.
- O problema direto associado (3.5) será resolvido numericamente usando MSF.
- As medições fictícias são obtidas tomando a derivada normal das soluções numéricas encontradas nos eletrodos.
- As medições fictícias obtidas serão utilizadas como dados de medição a utilizar através do método proposto na seção 3.4 anterior para resolver o problema inverso.

Para a criação dos dados fictícios, serão consideradas as mesmas condições de Dirichlet (3.18) e a fronteira $g_a :]0,1[\to]0,1[$,

$$g_a = 0.5 + 0.1 * \sin 2\pi x. \tag{3.22}$$

Para resolver o problema direto associado à fronteira g_a. 64 pontos fonte foram localizados no quadrado de lado 8 centrado em $(0.5, 0.5)$, com 64 pontos de colocação e também foi aplicada uma regularização de Tikhonov de ordem 2 à matriz do sistema associado à solução via o MSF porque a matriz era mal condicionada. Para a escolha do parâmetro de regularização, foi utilizado o critério da curva L.

Para resolver o problema inverso associado, foram usados 64 pontos fonte localizados no quadrado centrado em $(0.5, 0.5)$ de raio 4, 64 pontos de colocação, 16 em cada parede, como mostrado na Fig. 3.1, e 16 pontos na fronteira a ser reconstruída. A fronteira inicial considerada foi constante $g_0(x) = 0.1212$

Também foi incorporada uma regularização Tikhonov de ordem 2 para todas as iterações, o parâmetro de regularização em cada iteração foi escolhido através do uso de uma sub-rotina criada para uma faixa de $\{10^{-8}, 10^{-7}, \ldots, 10^{-5}\}$. Além disso, o parâmetro de regularização para o funcional (3.15) utilizado foi $\lambda_F = 10$ (escolhido pelo critério de prova e erro no intervalo $\{10^{-2}, 10^{-1}, 1, 2, 3, \ldots, 10\}$) e $P^{(2)}$ foi utilizado no referido funcional, uma vez que foi essa matriz que permitiu obter um melhor resultado.

O resultado da reconstrução da fronteira considerada é mostrado na Fig. 3.15. Foram necessárias 400 iterações para atingir um valor do funcional de 0.0531742.

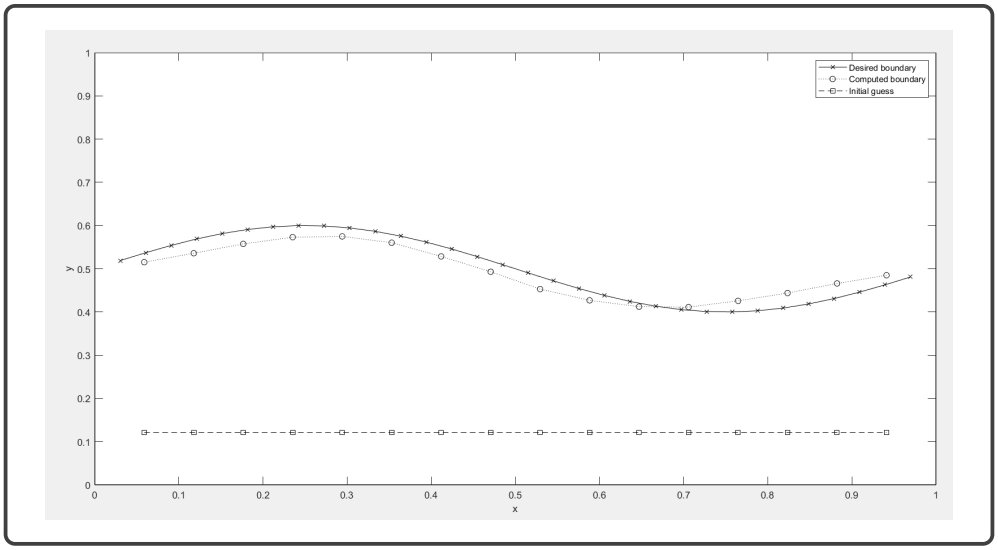

Figura 3.15: Fronteira inicial considerada (□), fronteira reconstruída (∘) e fronteira alvo (×).

Antes de apresentar os resultados numéricos para o caso tridimensional, será dada uma breve descrição do método de otimização utilizado nesta seção: algoritmos genéticos.

3.4.1.3 Algoritmos Genéticos

Baseado em processos de seleção natural, os algoritmos genéticos (GA), introduzidos pela primeira vez por John Holland [35], são um tipo de algoritmo estocástico para resolver problemas de otimização (com ou sem restrições). O algoritmo compõe-se principalmente de uma população inicial e uma função de adaptabilidade (função objetivo), que compara os membros da população através de técnicas inspiradas na teoria da evolução natural de Darwin [20], tais como: herança, mutação, seleção e cruzamento (também chamada de recombinação). A evolução usualmente parte da população inicial, que é composta por indivíduos gerados aleatoriamente e acontece ao longo de gerações. Em cada geração, a adaptabilidade de cada indivíduo é avaliada e então uma nova população dos indivíduos mais adequados é formada.

Uma das principais vantagens dos algoritmos genéticos é a capacidade de lidar com problemas complexos através do paralelismo (execução de duas ou mais rotinas computacionais simultaneamente). Estes algoritmos podem resolver eficazmente problemas de otimização, onde a função objetivo é estacionária ou não (mudanças com o tempo), linear ou não-linear, contínua ou descontínua, mesmo considerando dados com ruído. Como vários indivíduos descendentes em uma população agem de maneira independente, a população ou algum subgrupo pode explorar o espaço de busca em várias direções simultaneamente. Parâmetros diferentes e até grupos diferentes de sequências codificadas podem ser manipulados ao mesmo tempo.

No entanto, os algoritmos genéticos também têm algumas desvantagens. O tamanho da população, a escolha de alguns parâmetros como a probabilidade de mutação, cruzamento e critérios de seleção da nova população são decisivos em muitos casos. Uma escolha inadequada poderia dificultar a convergência do algoritmo ou talvez produzir resultados sem sentido. Apesar desses inconvenientes, os algoritmos genéticos continuam sendo uma boa alternativa amplamente utilizada, especialmente em problemas de otimização não-linear [78].

O uso de algoritmos genéticos associados ao método das soluções fundamentais tem aplicações relevantes, por exemplo, podemos citar: Jopek e Kołodziej [39], onde algoritmos genéticos são aplicados para determinar a localização ótima dos pontos de origem; em Santos et al. [68], simulações numéricas de sistemas de proteção catódica são realizadas combinando MSF e GA. Para detalhes de mais aplicações de GA em problemas inversos, ver [49].

3.4.1.4 Exemplos para o Caso Tridimensional

Para concluir esta seção de exemplos numéricos, serão apresentados resultados relativos ao caso tridimensional do problema inverso em análise.

Foram feitos testes para vários tipos de distribuição dos pontos fonte (esférica, cúbica, paralelepipédica). A distribuição esférica foi considerada a mais apropriada. A Figura 3.16 mostra um exemplo da distribuição dos pontos fonte (40 pontos localizados $40/8 = 5$ em cada octante da esfera) da rotina que foi implementada para tal caso.

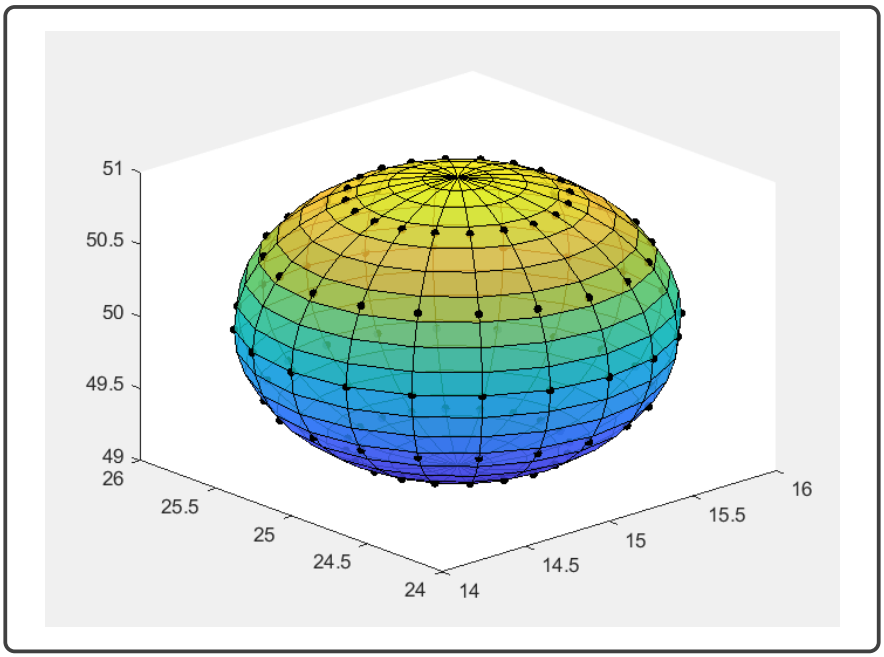

Figura 3.16: Exemplo de distribuição de 40 pontos fonte sobre uma superfície esférica de raio 1 e centro $(15, 25, 50)$.

Foi considerado um domínio ortoédrico de dimensões $[0, 30] \times [0, 50] \times [0, 100]$. Também foram considerados $k = 32$ eletrodos colocados 16 em cada parede e as condutividades elétricas da água e do ar dadas por $\gamma_1 = 1$ e $\gamma_2 = 0.01$, respetivamente. Foram considerados 62 pontos de colocação: 32 sobre as paredes, 15 na base, e 15 sobre a superfície a ser reconstruída. Além disso, foram criados dados fictícios usando o mesmo procedimento descrito no exemplo 3.4.4, considerando duas fronteiras alvo. Quanto aos dados Dirichlet utilizados, foram considerados os valores $(1, 2, 3, ..., 15, 16)$ e $(4, 5, 6, ..., 18, 19)$ para os 16 eletrodos de cada lado, respectivamente.

Para uma melhor análise comparativa do método proposto, tanto para a criação de dados fictícios como para a resolução dos problemas auxiliares envolvidos no algoritmo implementado que resolve o problema inverso, a quantidade de pontos fonte como o raio esférico de distribuição dos mesmos é diferente. Considerando o centro $(15, 25, 50)$, 56 pontos fonte foram utilizados na esfera de raio 360 para a criação dos dados fictícios e 40 pontos fonte, distribuídos na esfera de raio 400 para o algoritmo do método de resolução.

O conjunto $\{10^{-8}, 10^{-7}, ..., 10^{-2}\}$ foi usado para a eleição do parâmetro de regularização para as soluções do MSF, através de uma rotina implementada em MATLAB que procura a melhor escolha por meio do critério da curva L. Tal rotina foi usada para a obtenção dos dados fictícios da fronteira alvo (regularização das soluções associadas a cada tipo de fronteira alvo) e para os exemplos (regularização das soluções associadas ao MSF em cada iteração do algoritmo utilizado na solução do problema inverso). Considerando em todos os casos a regularização de Tikhonov de ordem zero.

Vale a pena mencionar que, todos os exemplos numéricos que serão apresentados foram testados utilizando os métodos dos pontos interiores, conjunto ativo e algoritmos genéticos, sendo este último mais eficaz em termos do número de iterações (menos iterações) para a sua convergência. Por conseguinte, a seguir, apenas serão apresentados os resultados correspondentes à utilização dos algoritmos genéticos através da função (*ga*) de algoritmos genéticos da *toolbox* do MATLAB com uma população de tamanho 100, com um limite de 50 iterações (gerações).

O funcional objetivo utilizado (função de adaptabilidade) é o mesmo proposto na seção 3.4 (J_2), uma vez que, em termos de complexidade computacional, se demonstra ser superior ao proposto na seção 3.3 (J_1). Em primeiro lugar, o funcional J_2 permite realizar uma regularização das soluções associadas ao MSF em cada iteração, o que significa uma maior precisão na comparação das soluções, visto que, em cada etapa iterativa é utilizado o critério da curva L através de uma sub-rotina, pelo que o parâmetro de regularização não é necessariamente o mesmo em cada etapa do processo iterativo. Porém, no funcional J_1, os parâmetros de regularização,

se fossem considerados (usando J_{r_1}), teriam de ser escolhidos através do critério de prova e erro, e são considerados fixos ao longo de todo o processo iterativo, o que poderia, a dada altura, produzir soluções menos precisas. Outra vantagem da utilização do funcional J_2 é o fato de lidar com um menor número de variáveis de optimização (apenas o número de pontos na fronteira a reconstruir), em comparação com o funcional J_1 que, além dos pontos na fronteira a reconstruir, contém também os coeficientes associados às soluções numéricas associadas ao MSF, o que implica um maior tempo de execução, independentemente do método utilizado na minimização do funcional objetivo.

Seguem-se dois exemplos numéricos para duas superfícies alvo diferentes considerando diferentes superfícies iniciais e WGN.

Exemplo 3.4.5 (Fronteira alvo constante)**.** Neste primeiro exemplo, para a criação das medidas fictícias, a fronteira alvo $g_a :]0, 30[\times]0, 50[\rightarrow]0, 100[$ foi considerada constante $g_a = 60$.

1. **Caso**: Como primeiro caso, foi tomado como chute inicial, uma fronteira constante $g_0(x, y) = 15$.

   A Figura 3.17 mostra o resultado obtido pelo algoritmo, as Figuras 3.18 e 3.19 representam os resultados após a adição de 3% e 5% de WGN, respectivamente.

   Para este primeiro caso, o ruído não afeta significativamente a resposta. A Tabela 3.4 apresenta a relação entre o ruído e o valor final da função minimizada depois de 50 iterações em todos os casos.

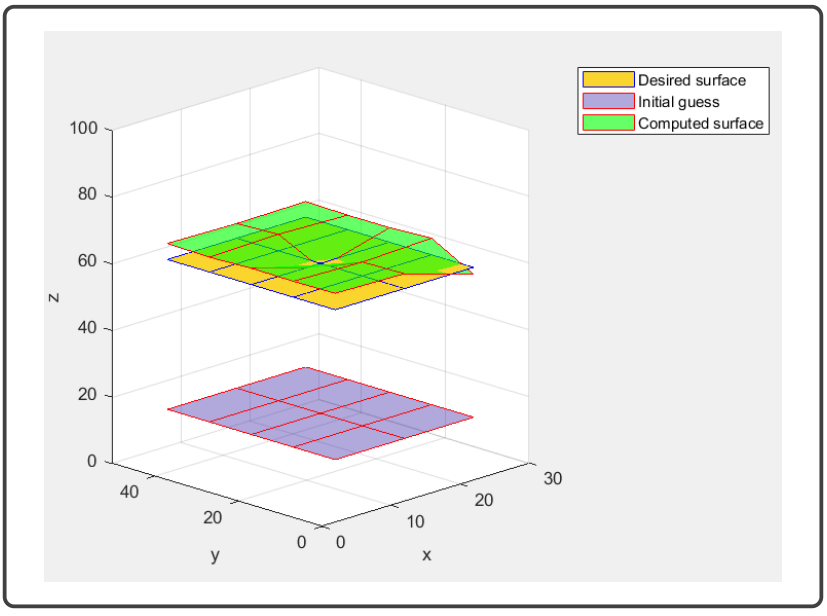

Figura 3.17: Fronteira inicial considerada (azul), fronteira reconstruída (verde) e fronteira alvo (amarelo).

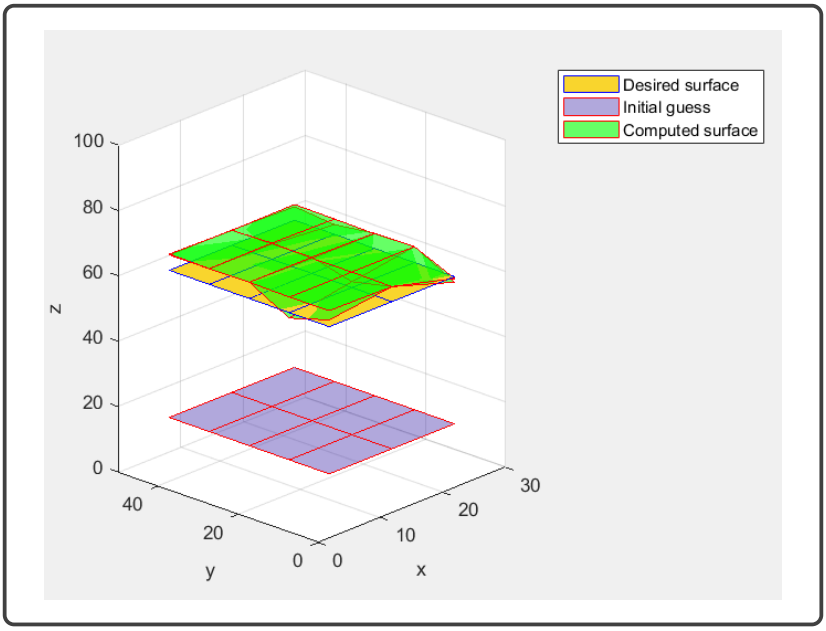

Figura 3.18: Fronteira inicial considerada (azul), fronteira reconstruída (verde) e fronteira alvo (amarelo) para dados poluídos com 3% de WGN.

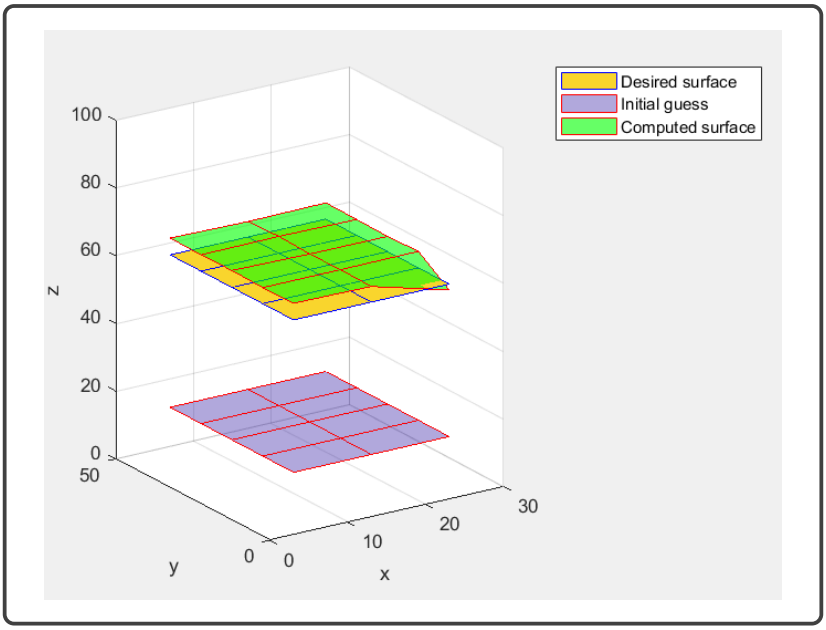

Figura 3.19: Fronteira inicial considerada (azul), fronteira reconstruída (verde) e fronteira alvo (amarelo) para dados poluídos com 5% de WGN.

Tabela 3.4: Ruído e valor final do funcional de custo para o Caso 1.

WGN %	J
0	0.00004975
3	0.0001007
5	0.0001216

2. **Caso**: Para este caso, como estimativa inicial, foi considerada a superfície $g_0 :]0, 30[\times]0, 50[\rightarrow]0, 100[$, dada por

$$g_0(x, y) = 5(\sin x + \cos y) + 10. \tag{3.23}$$

A Fig. 3.20 mostra o resultado atingido pelo algoritmo para 50 iterações. Como no caso anterior, o ruído foi incorporado com valores de 3% e 5% (Figuras 3.21 e 3.22 respectivamente) uma vez que, para valores superiores, os resultados não são satisfatórios. Para comparar a relação entre ruído e valor funcional para 50 iterações em todos os casos, ver Tabela 3.5.

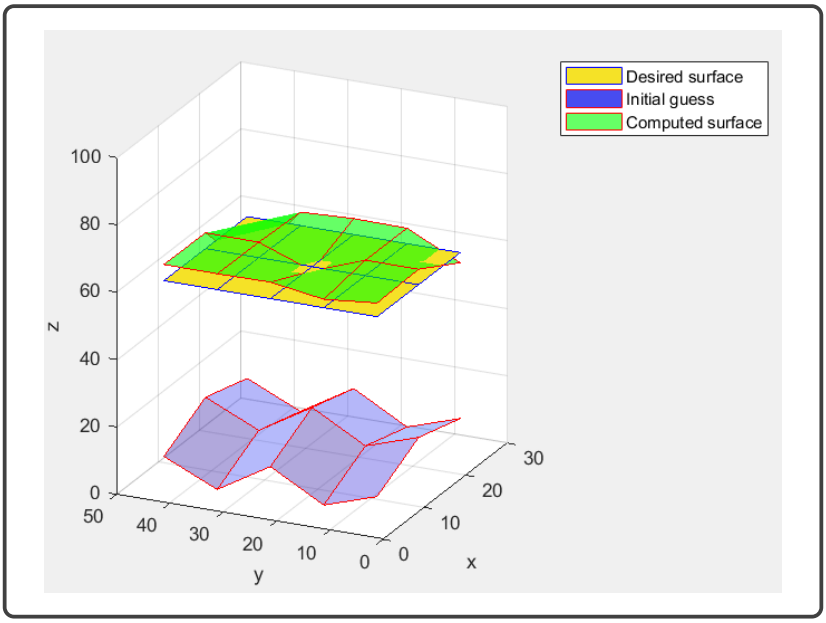

Figura 3.20: Fronteira inicial considerada (azul), fronteira reconstruída (verde) e fronteira alvo (amarelo).

Tabela 3.5: Ruído e valor final do funcional de custo para o Caso 2.

WGN %	J
0	0.00004976
3	0.00007847
5	0.00008175

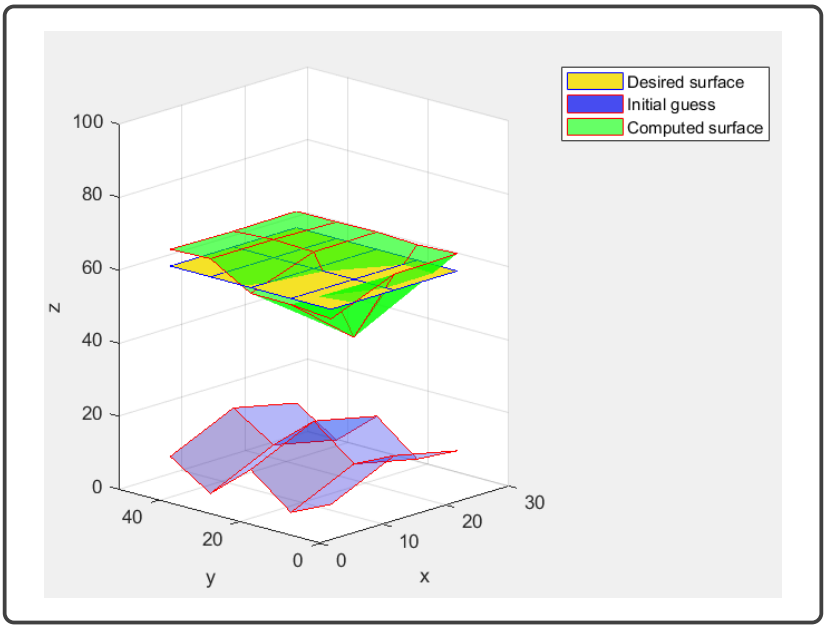

Figura 3.21: Fronteira inicial considerada (azul), fronteira reconstruída (verde) e fronteira alvo (amarelo) para dados poluídos com 3% de WGN.

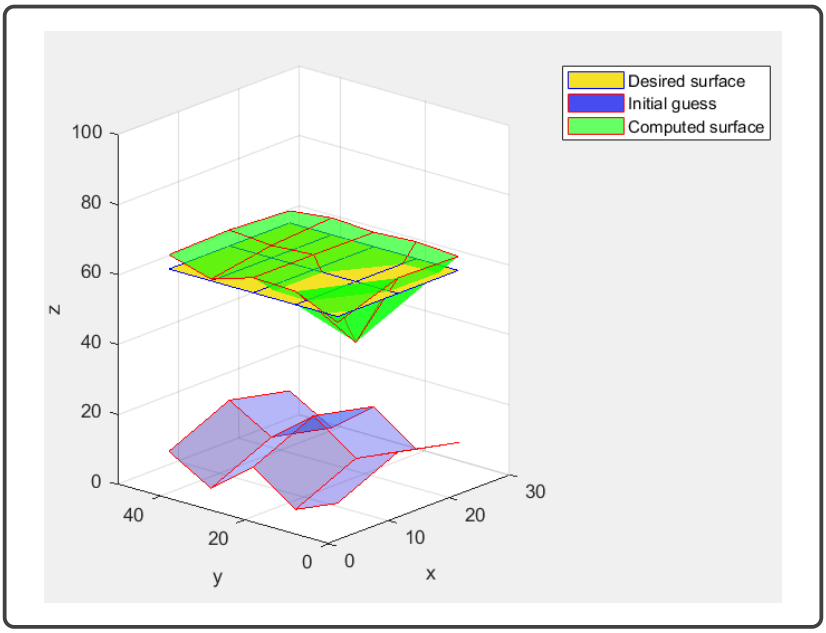

Figura 3.22: Fronteira inicial considerada (azul), fronteira reconstruída (verde) e fronteira alvo (amarelo) para dados poluídos com 5% de WGN.

Exemplo 3.4.6. Considerando a fronteira alvo $g_a :]0, 30[\times]0, 50[\rightarrow]0, 100[$, sendo a função

$$g_a(x, y) = 5(\sin x + \cos y) + 60. \tag{3.24}$$

Todos os resultados a seguir foram atingidos com 50 iterações.

1. **Caso**: Com estimativa inicial constante $g_0 = 15$, as Figuras 3.23, 3.24 e 3.25 representam os resultados para as soluções sem ruído, e com ruído de 1% e 3% respectivamente.

   Devido à complexidade da superfície alvo, neste exemplo, era de esperar-se que os resultados fossem favoráveis até 3% de WGN. Relação que se pode verificar na Tabela 3.6

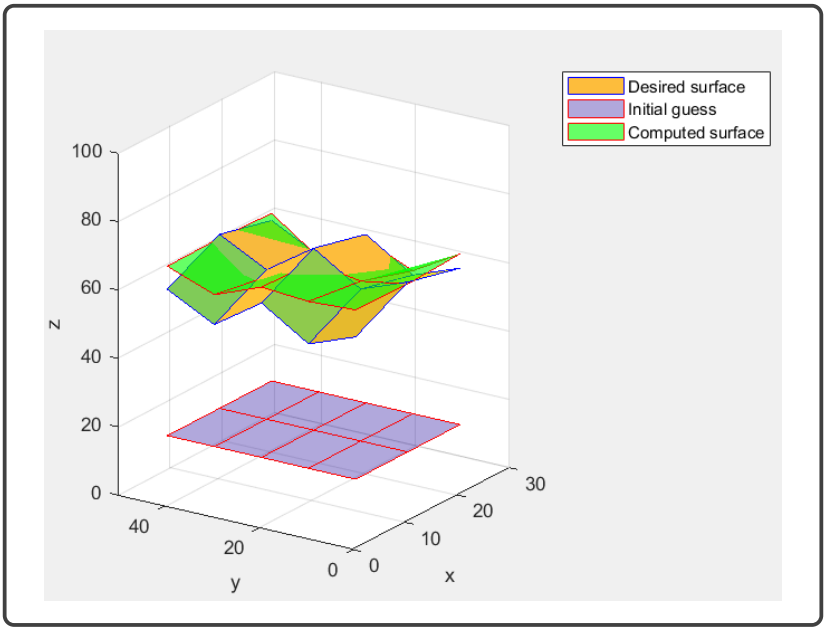

Figura 3.23: Fronteira inicial considerada (azul), fronteira reconstruída (verde) e fronteira alvo (amarelo).

Tabela 3.6: Ruído e valor final do funcional de custo para o Caso 1.

WGN %	J
0	0.00005196
1	0.00005685
3	0.00007337

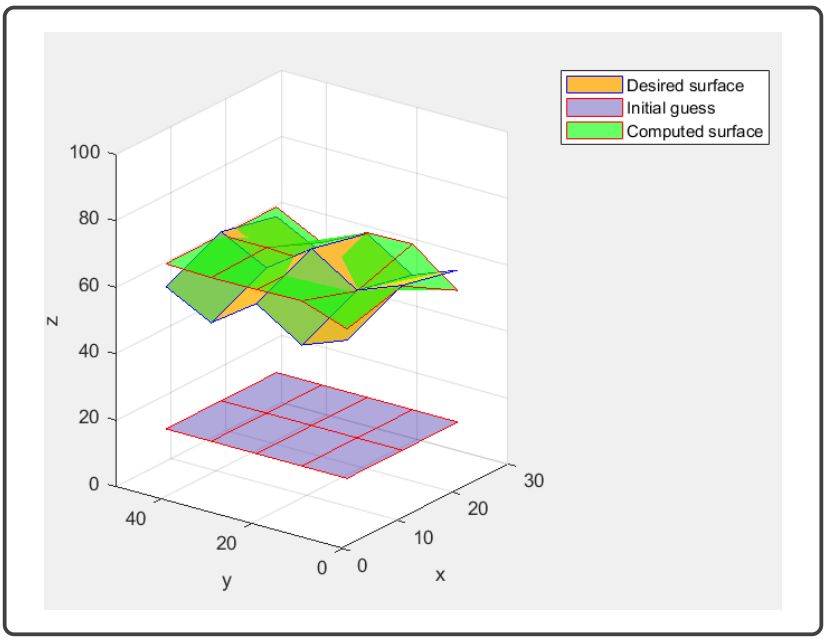

Figura 3.24: Fronteira inicial considerada (azul), fronteira reconstruída (verde) e fronteira alvo (amarelo) para dados poluídos com 1% de WGN.

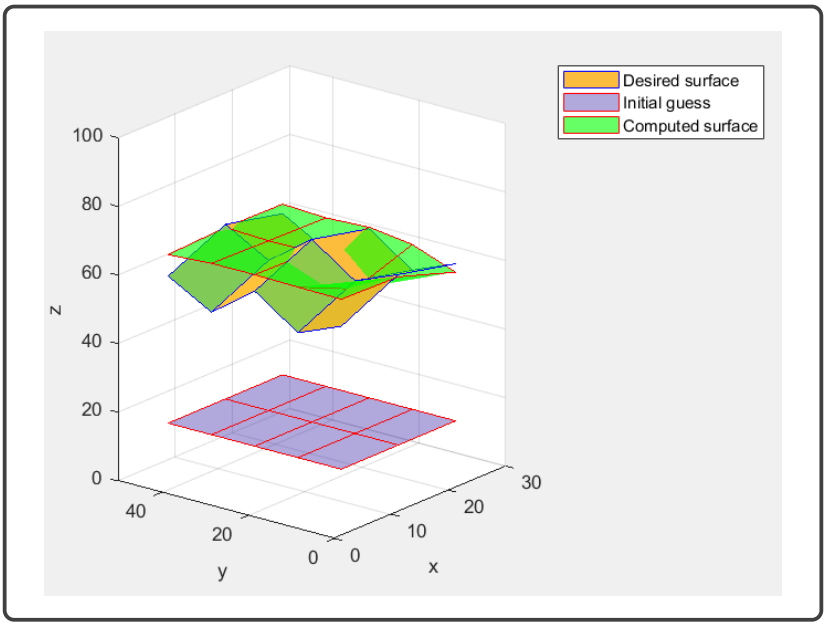

Figura 3.25: Fronteira inicial considerada (azul), fronteira reconstruída (verde) e fronteira alvo (amarelo) para dados poluídos com 3% de WGN.

2. **Caso**: Como último caso, é considerada a superficie inicial g_0 dada por

$$g_a(x, y) = 5(\sin x + \cos y) + 10, \tag{3.25}$$

cujos resultados são mostrados nas Figuras 3.26, 3.27 e 3.28 considerando as mesmas quantidades de ruído que no caso anterior e atingindo valores piores para o funcional minimizado, comparado com o primeiro caso, ver Tabela 3.7

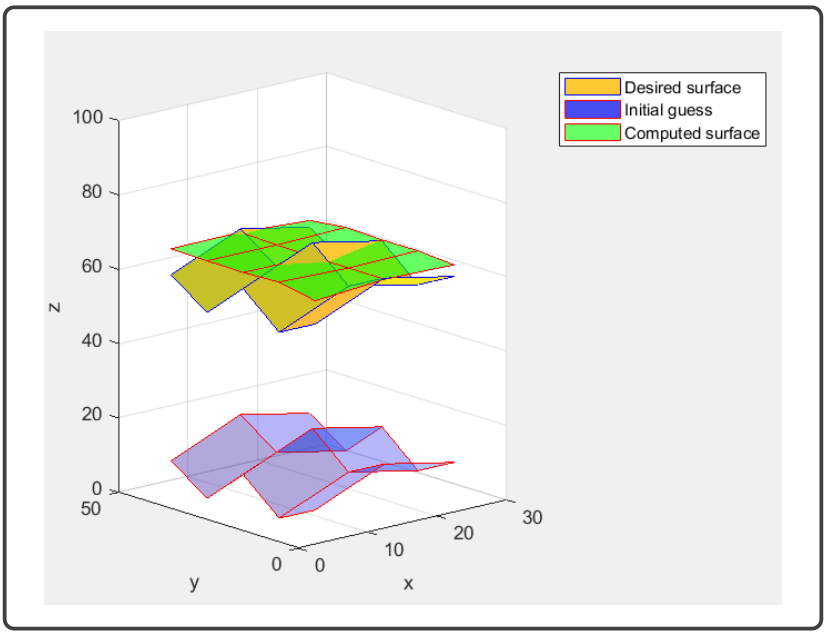

Figura 3.26: Fronteira inicial considerada (azul), fronteira reconstruída (verde) e fronteira alvo (amarelo).

Tabela 3.7: Ruído e valor final do funcional de custo para o Caso 2.

WGN %	J
0	0.00005194
1	0.00005738
3	0.00009358

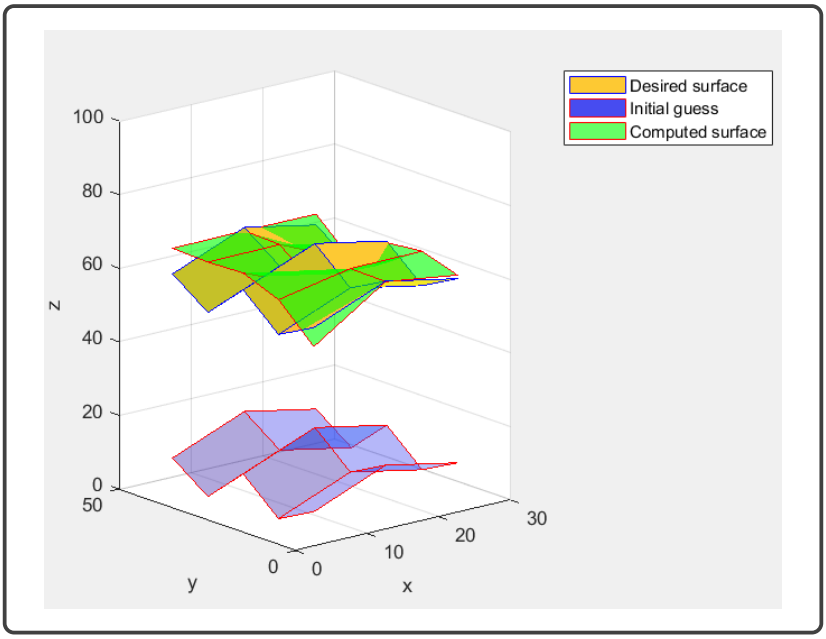

Figura 3.27: Fronteira inicial considerada (azul), fronteira reconstruída (verde) e fronteira alvo (amarelo) para dados poluídos com 1% de WGN.

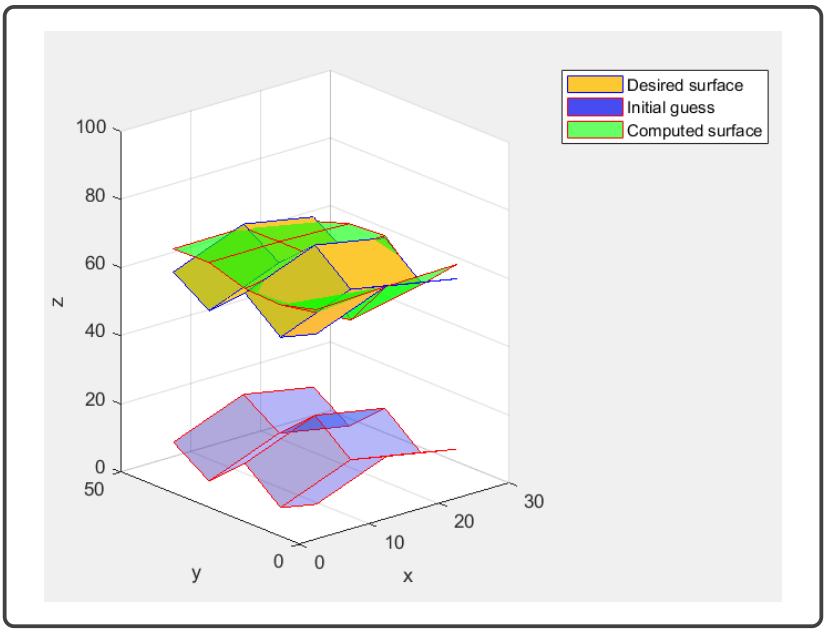

Figura 3.28: Fronteira inicial considerada (azul), fronteira reconstruída (verde) e fronteira alvo (amarelo) para dados poluídos com 3% de WGN.

3.5 Conclusões para as Propostas de Resolução Apresentadas

Neste capítulo, a principal contribuição deste trabalho foi apresentada com o objeto da aplicação do primeiro método apresentado no capítulo anterior para a análise da umidade em estruturas através de um caso generalizado que visa reconstruir a distribuição interna da superfície livre da água.

Nas primeiras seções foram mostrados vários exemplos em duas dimensões. Como primeira proposta de resolução, foram apresentados exemplos relacionados com o primeiro método do capítulo anterior, devido à maior praticidade na implementação e eficiência demonstrada em comparação com o método da velocidade. A primeira proposta mostrou eficácia mesmo com dados contaminados com WGN, mas também baixa eficiência devido ao número de iterações necessárias para atingir resultados favoráveis.

Como segunda proposta, o número de variáveis de otimização foi reduzido através da utilização de um funcional menor e da reestruturação do algoritmo implementado no MATLAB. Para esta proposta, foram apresentados quatro exemplos para o caso bidimensional. Os três primeiros exemplos visavam reconstruir uma superfície constante, e o último considerado uma superfície alvo em forma senoidal para dados fictícios criados numericamente. A proposta mostrou eficiência, bem como maior eficácia em relação à primeira proposta apresentada.

Finalmente, motivado pelos resultados da segunda proposta, foi implementado um algoritmo para a resolução do problema em três dimensões. Devido à falta de exemplos com soluções analíticas na literatura sobre este tipo de problemas, foi decidido considerar também dados fictícios, criados para a análise de dois tipos de superfícies a serem reconstruídas. Assim, foram apresentados dois exemplos, o primeiro para uma superfície alvo constante e o segundo para uma superfície mais complexa. Para ambos os casos, a WGN foi incorporada, obtendo resultados eficazes quanto à reconstrução do nível e distribuição interna de água considerando no máximo 50 iterações, permitindo o objetivo de determinar a distribuição de água numa estrutura através da técnica não invasiva: EIT.

Capítulo 4

Conclusões Gerais

Neste trabalho foi proposta a aplicação do Método das Soluções Fundamentais (MSF) para a resolução de problemas geométricos inversos, devido à maior facilidade de implementação e redução do custo computacional envolvido, por exemplo, em relação aos métodos numéricos malha-dependentes, que aplicados a este tipo de problema, implicam em remalhar o domínio em cada fase iterativa, independentemente do método utilizado para resolver o problema inverso.

Duas categorias de problemas inversos foram abordados neste trabalho. O primeiro, a título de exemplo, para comparar dois tipos de métodos de resolução, teve como objetivo a reconstrução de um domínio interno. De modo que dois métodos foram expostos, o primeiro com a característica de reduzir o problema inverso a um problema de otimização e o segundo método abordado através análise de sensibilidade à mudança de forma por meio do método da velocidade. O primeiro método demonstrou-se mais eficaz para este tipo de problema de reconstrução através da utilização do método das soluções fundamentais, pois o segundo método apresenta baixa taxa de convergência e alta dependência do chute inicial e parâmetros envolvidos na sua formulação.

Em seguida, estudou-se o problema Inverso da Tomografia por Impedância Elétrica (EIT) para a análise da umidade em estruturas. O problema foi abordado através da solução do problema da EIT para a reconstrução da superfície livre de água entre duas paredes. Duas propostas de resolução foram apresentadas. Comprovou-se mediante vários exemplos numéricos bidimensionais a eficácia da segunda proposta, mesmo com a adição de ruído aos dados. Também foi realizado um teste para o caso em três dimensões, onde duas fronteiras objetivo foram estudadas considerando dados fictícios criados por soluções numéricas obtidas através do MSF e poluindo esses dados com diferentes graus de WGN.

Foi possível verificar a eficácia do método das soluções fundamentais na solução dos problemas inversos abordados neste trabalho, não só pelas soluções apresentadas, mas pelo fato de ser um método sem malhas, o que diminui a complexidade espacial

e/ou temporal computacional, porém, uma desvantagem muito comum neste método é o fato de acabar sempre na solução de sistemas de equações mal condicionados, o que obriga à incorporação de algum método de regularização que envolve a busca do melhor parâmetro de regularização.

4.1 Trabalhos Futuros

Como possível trabalho futuro, pretende-se identificar formas mais gerais para o problema inverso geométrico discutido no segundo capítulo, ampliar os resultados para o caso da equação de Helmholtz. Além de incorporar a estratégia utilizada na seção 3.4 (segunda proposta de resolução) para reduzir o número de variáveis de otimização e, assim, melhorar a complexidade do algoritmo.

Para o problema de EIT, pretende-se considerar mais condições de fronteira para aumentar a precisão das soluções do problema direto auxiliar associado ao método de resolução do problema inverso utilizado, pois na maioria dos artigos relacionados a este problema inverso, são considerados dados de Dirichlet não só sobre os eletrodos mas também fora deles. Também se supõe uma possível melhora dos resultados obtidos pela incorporação de mais medições feitas a partir dos eletrodos no funcional de custo, pois geralmente, fatores externos como o ruído em cada coleta de dados alteram seu valor como medida.

Por fim, pretende-se trabalhar com dados reais para o problema da tomografia por impedância elétrica, abordado no capítulo três, de modo a se ter uma maior confiabilidade do método proposto.

Referências Bibliográficas

[1] Afraites, L., Dambrine, M., and Kateb, D. (2007). Conformal mappings and shape derivatives for the transmission problem with a single measurement. *Numerical Functional Analysis and Optimization*, 28:519–551.

[2] Amstutz, S., Horchani, I., and Masmoudi, M. (2005). Crack detection by the topological gradient method. *Control and Cybernetics*, 34.

[3] A.Potra, F. and J.Wright, S. (2000). Interior-point methods. *Journal of Computational and Applied Mathematics*, 124:281–302.

[4] Arthur, R. (2005). Damp Indoor Spaces and Health. Institute of Medicine: Committee on Damp Indoor Spaces and Health. The National Academy Press. ISBN 0-309-09193-4. Washington, D.C. 2004, pp. 355. *Journal of Public Health*, 27(2):234–234.

[5] Astala, K. and Päivärinta, L. (2006). Calderon's inverse conductivity problem in plane. *Annals of mathematics, ISSN 0003-486X, Vol. 163, Nº 1, 2006, pags. 265-299*, 163.

[6] Belge, M., Kilmer, M. E., and Miller, E. L. (2002). Efficient determination of multiple regularization parameters in a generalized l-curve framework. *Inverse Problems*, 18(4):1161–1183.

[7] Belhachmi, Z. and Meftahi, H. (2013). Shape sensitivity analysis for an interface problem via minimax differentiability. *Applied Mathematics and Computation*, 219:6828–6842.

[8] Bertero, M., Boccacci, P., and Koenig, A. (2001). Introduction to inverse problems in imaging. *Optics Photonics News*, 12:46–47.

[9] Biernat, K., Sikora, J., and Wójtowicz, S. (2012). Nondestructive impedance method of brickwork damp identification.

[10] Bin-Mohsin, B. (2013). *The method of fundamental solutions for Helmholtz-type problems*. PhD thesis, Department of Applied Mathematics; University of Leeds, Leeds, West Yorkshire, England.

[11] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.

[12] Bochniak, M. and Cakoni, F. (2000). Domain sensitivity analysis for acoustic scattering problems.

[13] Brebbia, C. A. (1984). The solution of continuum mechanics problems using boundary elements.

[14] Burger, M. (2001). A level set method for inverse problems. *Inverse Problems*, 17(5):1327–1355.

[15] Cattle, B. A. (2005). Limited-access remote-sensing using inverse problem techniques.

[16] Cheney, M., Isaacson, D., and Newell, J. C. (1999). Electrical impedance tomography. *SIAM Review*, 41:85–101.

[17] Colton, D. and Kress, R. (2013). *Inverse Acoustic and Electromagnetic Scattering Theory*. Springer-Verlag, New York.

[18] Colton, D. and Reemtsen, R. (1984). The numerical solution of the inverse stefan problem in two space variables. *SIAM Journal on Applied Mathematics*, 44(5):996–1013.

[19] Council, N. R. (1996). *Mathematics and Physics of Emerging Biomedical Imaging*. The National Academies Press, Washington, DC.

[20] Darwin, C. (1859). *On the origin of Species by Means of Natural Selection*. John Murray, Londres.

[21] Evans, L. C. (2010). *Partial differential equations*. American Mathematical Society, Providence, R.I.

[22] Feijóo, G., Oberai, A., and Pinsky, P. (2004). An application of shape optimization in the solution of inverse acoustic scattering problems. *Inverse Problems*, 20:199–228.

[23] Floudas, C. A. (1995). *Nonlinear and mixed-integer optimization*. Oxford university press, New York.

[24] Grossmann, I. (2002). Review of non-linear mixed integer and disjunctive programming techniques for process systems engineering. *Optim Eng.*, 3:227–252.

[25] Guzina, B. and Bonnet, M. (2006). Small-inclusion asymptotic of misfit functionals for inverse problems in acoustics. *Inverse Problems*, 22.

[26] Hager, W. and Zhang, H. (2016). An active set algorithm for nonlinear optimization with polyhedral constraints. *Science China Mathematics*, 59.

[27] Hall, C. and Hoff, W. (2007). Hoff, w.d.: Rising damp: capillary rise dynamics in walls. proc. r. soc. a math. phy. 463, 1871-1884. *Proceedings of The Royal Society A: Mathematical, Physical and Engineering Sciences*, 463:1871–1884.

[28] Hansen, P. (2010). *Discrete inverse problems. Insight and algorithms*, volume 7. Society for Industrial and Applied Mathematics.

[29] Hansen, P. C. (1990). Truncated singular value decomposition solutions to discrete ill-posed problems with ill-determined numerical rank. *SIAM J. Scientific Computing*, 11:503–518.

[30] Hansen, P. C. (1992). Analysis of discrete ill-posed problems by means of the l-curve. *SIAM Rev.*, 34(4):561–580.

[31] Hao, D., Xuan, T. P., and Lesnic, D. (2012). A boundary element method for a multidimensional inverse heat conduction problem. *International Journal of Computer Mathematics - IJCM*, 89:1–15.

[32] Hasanov, A. and Mueller, J. L. (2001). A numerical method for backward parabolic problems with non-selfadjoint elliptic operators. *Applied Numerical Mathematics*, 37(1):55 – 78.

[33] Hintermüller, M., Laurain, A., and Novotny, A. (2012). Second-order topological expansion for electrical impedance tomography. *Advances in Computational Mathematics*, 36:235–265.

[34] Hola, J., Matkowski, Z., and Sikora, J. (2002). New method of investigation of rising damp in brick walls by means of impedance tomography.

[35] Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI.

[36] Hon, Y. and Li, M. (2008). A computational method for inverse free boundary determination problem. *International Journal for Numerical Methods in Engineering*, 73:1291 – 1309.

[37] Isakov, V. (1990). *Inverse Source Problems*. Mathematical surveys and monographs. American Mathematical Society.

[38] Isakov, V. (2009). Inverse obstacle problems. *Inverse Problems*, 25(12):123002.

[39] Jopek, H. and Kołodziej, J. (2008). Application of genetic algorithms for optimal positions of source points in method of fundamental solutions. *Computer Assisted Mechanics and Engineering Sciences*, 15.

[40] Karpov, D. and Kapnob (2019). Thermovision method of determining the moisture fields of building surfaces of construction structures of buildings and constructions. *Bulletin of Belgorod State Technological University named after. V. G. Shukhov*, pages 28–33.

[41] Khludnev, A. M. and Sokolowski, J. (1997). *Modeling and Control in Solid Mechanics*, volume 122. Birkhauser Verlag, Basel.

[42] Klosowski, G. and Rymarczyk, T. (2017). Using neural networks and deep learning algorithms in electrical impedance tomography. *Informatics Control Measurement in Economy and Environment Protection*, 7:99–102.

[43] Knowles, N. (1984). Finite element analysis. *Computer-Aided Design*, 16(3):134 – 140.

[44] Kohn, R. and Vogelius, M. (1984). Determining conductivity by boundary measurements. *Communications on Pure and Applied Mathematics*, pages 289–298.

[45] Kolditz, O. (2002). *Finite Volume Method*. Springer Berlin Heidelberg, Berlin, Heidelberg.

[46] Kubo, S. (1988). Inverse problems related to the mechanics and fracture of solids and structures. *JSME international journal. Ser. 1, Solid mechanics, strength of materials*, 31(2):157–166.

[47] Kupradze, V. (1964). A method for the approximate solution of limiting problems in mathematical physics. *USSR Computational Mathematics and Mathematical Physics*, 4(6):199 – 205.

[48] Kupradze, V. and Aleksidze, M. (1964). The method of functional equations for the approximate solution of certain boundary value problems. *USSR Computational Mathematics and Mathematical Physics*, 4(4):82 – 126.

[49] Neto, A. S. and Becceneri, J. (2012). Técnicas de inteligência computacional inspiradas na natureza - aplicação em problemas inversos e transferência radiativa. *SBMAC*, 41, 2a Edição. Notas em Matemática Aplicada.

[50] Nocedal, J. and J., W. S. (2006). *Numerical Optimization*. Springer-Verlag, New York.

[51] Novotny, A. A. and Sokołowski, J. (2013). *Topological Derivatives in Shape Optimization*. Springer-Verlag Berlin, Berlin.

[52] O'Dell, S. L. (2008). Inverse scattering for schrödinger-type operators with interface conditions across smooth surfaces. *SIAM J. Math. Analysis*, 39:1595–1626.

[53] Paganini, A. (2015). *Approximate Shape Gradients for Interface Problems*, volume 166, pages 217–227. Birkhäuser, Cham.

[54] Pijpers, F. (2006). *Image reconstruction as an inverse problem*, volume 92, pages 241–254. Oxford Academic.

[55] Potyagaylo, D., Schulze, W. H. W., and Doessel, O. (2012). A new method for choosing the regularization parameter in the transmembrane potential based inverse problem of ecg. *Computing in Cardiology*, 39:29–32.

[56] Powell, M. J. D. (1969). Nonlinear Programming—Sequential Unconstrained Minimization Techniques. *The Computer Journal*, 12(3):207–a–207.

[57] P.S., N. (1938). Sur le problem inverse du potentiel. *Dokl. Akad. Nauk SSSR*, 18(3):165 – 168.

[58] Ramm, A. G. (2005). *Inverse Problems, Mathematical and Analytical Techniques with Applications to Engineering*, volume 1. Springer US.

[59] Rocha, S. and Novotny, A. (2016). Obstacles reconstruction from partial boundary measurements based on the topological derivative concept. *Structural and Multidisciplinary Optimization*.

[60] Rosen, J., Mangasarian, O., and Ritter, K. (1970). *Nonlinear Programming*. Wiley Interscience, New Jersey.

[61] Rybak, G., Romanowski, A., and Sankowski, D. (2014). Non–invasive methods of industrial processes control. *Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Środowiska*, 4(3):41–45.

[62] Rymarczyk, T., Adamkiewicz, P., and Kozłowski, E. (2018a). Application of least angle regression methods for image reconstruction in eit.

[63] Rymarczyk, T. and Kłosowski, G. (2018a). Application of neural reconstruction of tomographic images in the problem of reliability of flood protection facilities. *Eksploatacja i Niezawodność*, Vol. 20, no. 3:425–434.

[64] Rymarczyk, T. and Kłosowski, G. (2018b). The use of a neural controller in masonry tomography.

[65] Rymarczyk, T., Kłosowski, G., and Kozłowski, E. (2018b). Non-destructive system based on electrical tomography and machine learning to analyze moisture of buildings.

[66] Rymarczyk, T., Tchórzewski, P., and Sikora, J. (2018c). Detect dampness in building walls by solving the inverse problem in electrical impedance tomography.

[67] Sandoz, J. L. (1993). Moisture content and temperature effect on ultrasound timber grading. *Wood Science and Technology*, 27:373–380.

[68] Santos, W., Santiago, J., and Telles, J. C. (2012). An application of genetic algorithms and the method of fundamental solutions to simulate cathodic protection systems. *Computer Modeling in Engineering and Sciences*, 87:23–40.

[69] Sargheini, S. (2016). Shape sensitivity analysis of electromagnetic scattering problems.

[70] Sokolowski, J. and Zolesio, J.-P. (1992). *Introduction to Shape Optimization*. Springer-Verlag, Berlin.

[71] Sylvester, J. (1990). An anisotropic inverse boundary value problem. *Communications on Pure and Applied Mathematics*, 43(2):201–232.

[72] Taflove, A. and Umashankar, K. (1990). *Finite Element and Finite Difference Methods in Electromagnetic Scattering*. Elsevier, Oxford.

[73] Tikhonov, A. N. and Arsenin, V. Y. (1977). *Solutions of Ill-posed problems*. W.H. Winston.

[74] Uhlmann, G., Päivärinta, L., and Panchenko, A. (1980). On an inverse boundary value problem, in seminar on numerical analysis and its applications to continuum physics.

[75] Uhlmann, G., Päivärinta, L., and Panchenko, A. (2003). Complex geometical optics for lipschitz conductivities. *Revista matemática iberoamericana, ISSN 0213-2230, Vol. 19, Nº 1, 2003, pags. 57-72*, 19.

[76] Vauhkonen, P. J. (2004). *Image Reconstrucion in Three-dimensional Electrical Impedance Tomography*. PhD thesis, Department of Applied Physics; University of Kuopio, Finland. PhD thesis.

[77] Wang, J., Han, B., and Wang, W. (2019). Elastic-net regularization for nonlinear electrical impedance tomography with a splitting approach. *Applicable Analysis*, 98(12):2201–2217.

[78] Yang, X.-S. (2014). *Chapter 5 - Genetic Algorithms*, pages 77 – 87. Elsevier, Oxford.

[79] Zhang, Y., Gulliksson, M., and Cheng, X. (2016). An adjoint method in inverse problems of chromatography. *Inverse Problems in Science and Engineering*, pages 1–26.

Apêndice A

Códigos em Matlab

A.1 Códigos dos Exemplos do Capítulo 2

A.1.1 Primeiro Método

Equação de Helmholtz modificada para o problema inverso de reconstrução.

```
%% Example 1 pag83 Bin Mohsin tese.
    % problema
    % \Delta u - k^2 u=0 in \Omega \ D
    % u = f on \partial \Omega
    % u = h on \partial D
    % du/dn = g on \partial \Omega (measured data)
%% Descripcao das variaveis usadas
clear all; close all;
global x_cO y_cO x_cD y_cD x_sO y_sO x_sD y_sD M N vtheta k ...
    lambF PF
k = sqrt(2);
lambF=1;  % regularizacao do funcional
r0=0.7;     % radio real de cavidad a identificar D
rOmega = 1.0; % radio dominio Omega
R_se = 2.0; % radio puntos fuente exterior
s=2;        % divisor radio puntos fuente interior
r_0=0.3;    % radio interno D valor inicial dado
a_0=1;    % valor inicial para todos los coeficientes a0
M = 10; N = M;  % numero de puntos fuente y de colocacion

for i = 1:M
   x_cO(i) = cos(2*pi*i/M);
   y_cO(i) = sin(2*pi*i/M);
   vtheta(i)=2*pi*i/M;
end
for l = 1:N
```

```
26      x_cD(l) = cos(2*pi*(l)/N);
27      y_cD(l) = sin(2*pi*(l)/N);
28  end
29  x_sO= R_se*x_cO; y_sO = R_se*y_cO;
30  x_sD= (1/s)*x_cD; y_sD = (1/s)*y_cD;
31  figure(1)
32  subplot(1,2,1);
33  plot(r_0*x_cD,r_0*y_cD,'bx',x_cO,y_cO,'b*',r_0*x_sD,...
34  r_0*y_sD,'ro',x_sO,y_sO,'rs');
35  legend('Pontos de Colocacao do furo $\partial D$','Pontos de ...
        Colocacao em $\partial\Omega$','Pontos Fonte interior de ...
        D','Pontos Fonte fora de $\Omega$','interpreter','latex');
36  %% Matriz regularizacao Funcional
37  % Matriz de regularizacao Tikhnov ordem 1 para o funcional
38              m1=diag(-1*ones(1,N));
39              m1=m1(1:end-1,:);
40              m2=diag(ones(1,N),1);
41              m2=m2(1:N-1,1:N);
42              PF=m1+m2;
43  %% Resolvendo funcao a otimizar
44  % Tipo de algoritmo
45  %'Algorithm','active-set','interior-point' (default),'sqp'
46  % example de options : options = ...
        optimset('Algorithm','active-set','Display',...
47  %     'iter-detailed','MaxFunEvals',30000,'TolFun',1.e-16);
48  %options = ...
        optimset('Algorithm','active-set','Display','iter-detailed',...
49  'MaxFunEvals',10000,'TolFun',1.e-16);
50  %options = optimset('Algorithm','interior-point',...
51  'Display','iter-detailed','MaxFunEvals',10000,'TolFun',1.e-10,...
52  'TolX',1.e-10,'TolCon',1.e-10);
53  options = ...
        optimoptions('fmincon','Display','iter','MaxFunEvals',40000,...
54  'TolFun',1.e-10);%para detallar las iteraciones
55  lb=[-Inf*ones(M,1);-Inf*ones(N,1);0.01*ones(N,1)];
56  ub=[Inf*ones(M,1);Inf*ones(N,1);0.99*ones(N,1)];
57  a0=a_0*ones(M+N,1); % valor inicial a(coeficientes de u_mfs=\sum ...
        a_i FundSolution)
58  vr0=r_0*ones(N,1); % fronteira inicial D_0 bola B(0,r_0)
59  x0=[a0;vr0];
60  [x fval]=fmincon(@objectfun,x0,[],[],[],[],lb,ub,[],options); ...
        %cacula os coeficientes e o raio
61  fprintf('|Dr-Ds|=%f\n',norm(vr0-x(M+N+1:end)));
62  %% Grafica das fronteiras D real e obtida usando MFS
63  %pontos da fronteira D usando MFS
64  xsol=x(M+N+1:end).*(x_cD)';
65  ysol=x(M+N+1:end).*(y_cD)';
```

```
%pontos da fronteira real D (circulo de raio r0=0.7)
t=1:60;
xr=r0*[cos(2*pi*t/60) cos(2*pi/60)];
yr=r0*[sin(2*pi*t/60) sin(2*pi/60)];
%pontos da fronteira inicial usada (circulo de raio r_0)
xin=r_0*[cos(2*pi*t/60) cos(2*pi/60)];
yin=r_0*[sin(2*pi*t/60) sin(2*pi/60)];
%figure(2)
subplot(1,2,2);
plot(xr,yr,'b-',xin,yin,'r--',[xsol;xsol(1)]',[ysol;ysol(1)]',...
'g-o');
legend('fronteira real $D$','fronteira inicial $D_0$','fronteira ...
    $D^*$ usando MFS','interpreter','latex');
suptitle(['M=N=' num2str(M) ' Pontos fonte(M) e colocacao(N)']);

%% Comparando solucao analitica ur e solucao numerica umfs na ...
    bola B(0,0.85)
%{
vt=[0.85*x_c0' 0.85*y_c0']; % M pontos na fronteira da bola B(0,1.5)
yyumfs=zeros(1,M);
for i=1:M
    yyumfs(i)=umfs(x,vt(i,:));
end
%}
vvtheta=0:0.1:2*pi;
xp=0.85*cos(vvtheta);
yp=0.85*sin(vvtheta);
vp=[xp' yp'];
te=size(vp);
for i=1:te(1)
    yyumfs(i)=umfs(x,vp(i,:));
end
yur=ur(0.85,vvtheta);
figure(2)
subplot(1,2,1)
plot(vvtheta,yur,'b-',vvtheta,yyumfs,'r-o');
xlabel('$\theta \in [2\pi/M,2\pi]$','interpreter','latex');
ylabel('$u_a (0.85,\theta)$','interpreter','latex');
title('Comparacao solucao analitica $u_a$ e solucao numerica ...
    $u_{MSF}$ na bola $B(0,0.85)$','interpreter','latex');
legend('Solucao analitica $u_a$','Solucao numerica ...
    $u_{MSF}$','interpreter','latex');
%% comparando dur/dn real com dumfs/dn na bola B(0,0.85)
yydumfs=zeros(1,M);
for i=1:te(1)
    yydumfs(i)=dumfs(x,vp(i,:),vvtheta(i));
end
```

```
ydur=dur(0.85,vvtheta);
figure(2)
subplot(1,2,2)
plot(vvtheta,ydur,'b-',vvtheta,yydumfs,'r-o');
xlabel('$\theta \in [2\pi/M,2\pi]$','interpreter','latex');
ylabel('$\frac{\partial u_a}{\partial ...
    n}(0.85,\theta)$','interpreter','latex');
title('Comparacao $\frac{\partial u_a}{\partial n}$ e solucao ...
    numerica $\frac{\partial}{\partial n} u_{MSF}$ na bola ...
    $B(0,0.85)$','interpreter','latex');
ll=legend('$\frac{\partial u_a}{\partial ...
    n}=g$','$\frac{\partial}{\partial n} ...
    u_{MSF}$','interpreter','latex');
set(ll,'FontSize',12);

%{
%% Experimentando para D_0 inicial forma de elipse semieixos ...
    a=0.5 b=0.4
%%{
for i=1:M
    vr02(i)=(0.5*0.4)/(norm([0.4*cos(vtheta(i)) ...
        0.5*sin(vtheta(i))]));
end
x02=[a0;vr02']; %vr02 fronteira inicial D_0 elipse
[x2 fval2]=fmincon(@objectfun,x02,[],[],[],[],lb,ub);

%% Grafica das fronteiras D real e obtida para elipse usando MFS
%pontos da fronteira D usando MFS
xsol2=x2(M+N+1:end).*(x_cD)';
ysol2=x2(M+N+1:end).*(y_cD)';
%pontos da fronteira inicial usada (elipse)
xin2=vr02.*x_cD;
yin2=vr02.*y_cD;
figure(3)
subplot(1,2,1)
plot(xr,yr,'b-',xin2,yin2,'r--',xsol2',ysol2','g-o');
legend('fronteira real $D$','fronteira inicial $D_0$','fronteira ...
    $D^*$ usando MFS','interpreter','latex');
title(['M=N=' num2str(M) ' Pontos fonte(M) e colocacao(N)']);
%% Comparando solucao analitica ur e solucao numerica umfs2 na ...
    bola B(0,1.5)
vt2=[1.5*x_cO' 1.5*y_cO'];
yumfs2=zeros(1,M);
for i=1:M
    yumfs2(i)=umfs(x2,vt2(i,:));
end
yur=ur(1.5,vtheta);
```

```
figure(3)
subplot(1,2,2)
plot(vtheta,yur,'b-',vtheta,yumfs2,'r-o');
xlabel('theta in [2pi/M,2pi]');
ylabel('u(1.5,theta)');
title('Comparacao solucao real u e solucao numerica umfs2 para ...
    D0:ELIPSE na bola B(0,1.5)');
legend('Solucao real u','Solucao numerica umfs2 D0:ELIPSE');
%}

%% Definicao das funcoes de fronteira
function yf=f(theta) %funcao f
    yf=exp(cos(theta)+sin(theta));
end
function yh=h(theta)  %funcao h
    yh=exp(0.7*(cos(theta)+sin(theta)));
end
function yg=g(theta) %funcao dada medida na fronteira de Omega du/dn
    yg=(cos(theta)+sin(theta)).*exp(cos(theta)+sin(theta));
end
%% Solucao real ur do problema
function yur=ur(r,theta)
    yur=exp(r*(cos(theta)+sin(theta)));
end
%% Derivada normal da sol real dur/dn do problema
function ydur=dur(ra,ttheta)
    ydur=(cos(ttheta)+sin(ttheta)).*exp(ra*(cos(ttheta)+...
    sin(ttheta)));
end
%% Definicao da solucao u_msf fundamental para Helmholtz Modificado
function yumfs=umfs(soll,vx) % (vetor coluna)soll:solucao obtida ...
    de fmincon ,(vetor fila)vx:valor a avaliar em u_mfs
global x_cO y_cO x_cD y_cD x_sO y_sO x_sD y_sD M N vtheta k
    vs=[x_sO' y_sO'; soll(M+N+1:end).*x_sD' soll(M+N+1:end).*y_sD'];
    yumfs=0;
    for i=1:M+N
        yumfs=yumfs+soll(i)*besselk(0,k*norm(vx-vs(i,:)));
    end
end
%% Definicao da derivada normal dumsf/dn para Helmholtz Modificado
function ydumfs=dumfs(soll,vx,ang)
%(vetor coluna)soll:solucao obtida de fmincon,
%(vetor fila)vx:valor a avaliar em du_mfs,
% angulo polar do ponto vx
global x_cO y_cO x_cD y_cD x_sO y_sO x_sD y_sD M N vtheta k
    vs=[x_sO' y_sO'; soll(M+N+1:end).*x_sD' soll(M+N+1:end).*y_sD'];
    ydumfs=0;
```

```
192     for i=1:M+N
193         ydumfs=ydumfs+soll(i)*(-k/(norm(vx-vs(i,:))))*...
194         ((vx-vs(i,:))*[cos(ang) ...
                sin(ang)]')*besselk(1,k*norm(vx-vs(i,:)));
195     end
196 end
197 %% Definicao da funcao objetivo
198 function yobjectfun=objectfun(x) %funcao objetivo
199     global x_c0 y_c0 x_cD y_cD x_s0 y_s0 x_sD y_sD M N vtheta k ...
            lambF PF
200     s1=0;
201     s2=0;
202     s3=0;
203     for i=1:M
204         aux1=zeros(M+N,1); % 1 termino
205         aux2=zeros(M+N,1); % 2 termino
206         for j=1:M+N
207             if j<=M
208                 aux1(j)=x(j)*besselk(0,k*norm([x_c0(i) ...
                        y_c0(i)]-[x_s0(j) y_s0(j)])); % 1 termino
209                 aux2(j)=x(j)*(-k/norm([x_c0(i) y_c0(i)]-[x_s0(j) ...
                        y_s0(j)]))*(([x_c0(i) y_c0(i)]-[x_s0(j) ...
                        y_s0(j)])*[cos(vtheta(i)) ...
                        sin(vtheta(i))]')*besselk(1,k*norm([x_c0(i) ...
                        y_c0(i)]-[x_s0(j) y_s0(j)])); % 2 termino
210             else
211                 aux1(j)=x(j)*besselk(0,k*norm([x_c0(i) ...
                        y_c0(i)]-[x(j+N)*x_sD(j-M) ...
                        x(j+N)*y_sD(j-M)])); % 1 termino
212                 aux2(j)=x(j)*(-k/norm([x_c0(i) ...
                        y_c0(i)]-[x(j+N)*x_sD(j-M) ...
                        x(j+N)*y_sD(j-M)]))*(([x_c0(i) ...
                        y_c0(i)]-[x(j+N)*x_sD(j-M) ...
                        x(j+N)*y_sD(j-M)])*[-cos(vtheta(i)) ...
                        -sin(vtheta(i))]')*besselk(1,k*norm([x_c0(i) ...
                        y_c0(i)]-[x(j+N)*x_sD(j-M) ...
                        x(j+N)*y_sD(j-M)])); % 2 termino
213             end
214         end
215         s1=s1+(sum(aux1)-f(vtheta(i)))^2; % 1 termino
216         s2=s2+(sum(aux2)-g(vtheta(i)))^2; % 2 termino
217     end
218     %tercer termino funcional
219     for i=1:N
220         aux3=zeros(M+N,1); % 3 termino
221         for j=1:M+N
222             if j<=M
```

```
223                 aux3(j)=x(j)*besselk(0,k*norm([x(i+M+N)*x_cD(i) ...
                        x(i+M+N)*y_cD(i)]-[x_sO(j) y_sO(j)])); % 3 ...
                        termino
224             else
225                 aux3(j)=x(j)*besselk(0,k*norm([x(i+M+N)*x_cD(i) ...
                        x(i+M+N)*y_cD(i)]-[x(j+N)*x_sD(j-M) ...
                        x(j+N)*y_sD(j-M)])); % 3 termino
226             end
227         end
228         s3=s3+(sum(aux3)-h(vtheta(i)))^2; % 3 termino
229     end
230     yobjectfun=s1+s2+s3+lambF*norm(PF*x(M+N+1:end))^2;
231 end
```

A.1.2 Segundo Método

Equação de Laplace para o problema inverso de reconstrução usando o Método da Velocidade

```
1  %% Example B 4.1.5.2 pag107 TOPOLOGICAL DERIVATIVES IN SHAPE ...
       OPTIMIZATION NOVOTNY, SOKOLOWSKI
2      % problema
3      % \Delta u =0 in \Omega \ D
4      % u = cos(\theta) on \partial \Omega
5      % u = 0 on \partial D
6      % du/dn = g= c1*cos(\theta)(1+c2/r^2) on \partial \Omega ...
           (measured data)
7      % g=[c1(1-c2*(x^2+y^2)-2*x^2)/(x^2+y^2)^2 , ...
           c1*c2*(2*x*y)/(x^2+y^2)^2 ]*n'
8      % c1=p/(p^2-ep^2) , c2=ep^2
9      % u= (p/r)((r^2-ep^2)/(p^2-ep^2))cos(\theta)
10
11 %% Solucoes usando MFS
12 clear all; clc
13 %clear all clc
14 global M  theta
15 p=1;
16 ep=0.5;
17 c1=p/(p^2-ep^2);
18 c2=ep^2;
19 M=10;
20 m=2;
21 r0=0.9;
22
23 % Pontos colocacao e fonte
```

```
for i=1:M
    x(i)=cos(2*i*pi/M);
    y(i)=sin(2*i*pi/M);
    theta(i)=2*i*pi/M;
end
xce=p*[x',y'];%pontos coloc em \partial\Omega
xci=r0*xce;%pontos colocacao em \partial D_0
xfe=m*xce;%pontos fonte fora de \Omega
xfi=(1/m)*xci;%pontos fonte interior D

xci_0=xci; %chute inicial
x_alvo=ep*[x',y']; %fronteira alvo

for i=1:M
    for j=1:M
        A1(i,j)=g(xce(i,:),xfe(j,:));
        B1(i,j)=dg(xce(i,:),xfe(j,:),[cos(theta(i)),sin(theta(i))]);
    end
end

bd=[cos(theta)';zeros(M,1)];
bn=c1*(1+c2)*[cos(theta)';zeros(M,1)];

cont=0;
h=0.01;
t=0.01;
d=zeros(1,2*M);
n=zeros(1,2*M);
hold on
figure(1)
while cont < 600
    for i=1:M
        for j=1:M
            A2(i,j)=g(xce(i,:),xfi(j,:));
            A3(i,j)=g(xci(i,:),xfe(j,:));
            A4(i,j)=g(xci(i,:),xfi(j,:));
            B2(i,j)=dg(xce(i,:),xfi(j,:),[cos(theta(i)),...
            sin(theta(i))]);
            B3(i,j)=dg(xci(i,:),xfe(j,:),[cos(theta(i)),...
            sin(theta(i))]);
            B4(i,j)=dg(xci(i,:),xfi(j,:),[cos(theta(i)),...
            sin(theta(i))]);
        end
    end

    A=[A1,A2;A3,A4];
    B=[B1,B2;B3,B4];
```

```
71
72      d=A\bd;
73      n=B\bn;
74
75      plot([xci(:,1);xci(1,1)],[xci(:,2);xci(1,2)]);
76
77      for k=1:M
78         V(k,:)=v(d,n,xci(k,:),xfe,xfi,[-cos(theta(k)),...
79         -sin(theta(k))],h);
80      end
81
82      xci=xci+t*V;
83      xfi=(1/m)*xci;
84      cont=cont+1;
85  end
86  hold off
87  figure(2)
88  plot([x_alvo(:,1);x_alvo(1,1)],[x_alvo(:,2);x_alvo(1,2)],...
89  [xci(:,1);xci(1,1)],[xci(:,2);xci(1,2)],[xci_0(:,1);xci_0(1,1)],...
90  [xci_0(:,2);xci_0(1,2)]);
91  legend('Desired boundary','Computed boundary','Initial guess')
92
93  %% VELOCIDADE
94  function yv=v(dd,nn,xx,e_e,e_i,vn,h) %dd,nn solucoes, xx ponto,
95  global M theta                 %vn normal a xx, ee pontos fontes
96  ee=[e_e;e_i];
97  s=0; % vn vetor normal h mult velocidade
98  for i=1:2*M
99      s=s+(-1/(2*pi*(norm(xx-ee(i,:)))^2))*(dd(i)-nn(i))*...
100     ((xx-ee(i,:))*vn');
101 end
102 yv=h*(s^2)*vn';
103 end
104 %% SOL FUNDAMENTAL G(x,\xi)
105 function yg=g(xx,eth) % xx ponto, eth fonte i-esima
106 global M theta
107 yg=(-1/(2*pi))*log(norm(xx-eth));
108 end
109 %% DERIV FUNDAMENTAL \nabla G .n
110 function ydg=dg(xx,eth,vn)% vn vetor normal a xx
111 global M
112 ydg=(-1/(2*pi*(norm(xx-eth))^2))*((xx-eth)*vn');
113 end
```

A.2 Códigos dos Exemplos do Capítulo 3

A.2.1 Primeira Proposta

Uso do funcional que determina simultaneamente os coeficientes de MFS e os pontos de fronteira a reconstruir.

```
%% Problema exemplo ruido
clear all; clc;
% \gamma_1\Delta u_1=0 em \Omega_1
% \gamma_2\Delta u_2=0 em \Omega_2
% u_1 = -\gamma (y+1) sobre \Gamma_1
% u_1 = 0 sobre \Gamma_2
% u_1 = -\gamma*(y+1) sobre \Gamma_3
% u_2 = y+1-(\gamma+1)*s sobre \Sigma_1
% u_2 = y+1-(\gamma+1)*s sobre \Sigma_2
% u_1 = u_2 sobre \Gamma  (continuidade)
% \gamma_1 du_1/n = \gamma_2 du_2/n  sobre \Gamma (salto da ...
    derivada normal)
% Dominio considerado (-1/2,1/2)x(-1,1)
% \gamma=\gamma2/\gamma1
% solucao analitica :
% u1= -\gamma*(y+1) ; u2=(y+1)-(\gamma+1)(s+1)
global M N tc tf pcs1 pcs2 pcg1 pcg3 pfi pfd pfa pfb pcg2 pf f1 ...
    f2 f3 g1 g2 hs1 hg1 hs2 hg3 gamma1 gamma2 fronteira vruido ...
    lamb1 lamb2 lamb3

%% PARAMETROS
% ESCOLHA DOS PONTOS
% o numero de pontos de colocacao M e fonte N deve ser tal que:
% 11*N/6 ≥ 2*M+4+N/6  ou  5*N/6 - 2 ≥ M
% para ter mais equacoes do que incognitas no sistema nao linear do
% funcional e existir solucoes
% Exemplos: (M=36,N=42,48,...) (M=42,N=54,60,...) ;  ...
    (M=48,N=60,66,...) ; (M=54,N=72,78,...)
M= 36; %numero de pontos fonte deve ser multiplo de 6
N= 78; %numero de pontos colocacao deve ser multiplo de 6
tc=N/6;% quantidade de pontos colocacao por lado e quantidade de ...
    pontos na fronteira desconhecida
tf=M/6;
pcs1=zeros(1,tc);pcs2=pcs1;
pcg1=pcs1; pcg2=pcs1; pcg3=pcs1;
fronteira= 0.5; %  -1 < fronteira < 1


```

```
34 %pontos fonte sobre (-1,1)x(-2,2)
35 hfi=4/(2*tf+1);
36 pfiy=2:-hfi:-2;
37 pfiy=pfiy(1,2:(end-1));
38 pfix=-1*ones(2*tf,1);
39 pfi=[pfix, pfiy']; %pontos fonte lado esquerdo de acima para abaixo
40
41 hfb=2/(tf+1);
42 pfbx=-1:hfb:1;
43 pfbx=pfbx(1,2:(end-1));
44 pfb=[pfbx' , -2*ones(tf,1)];%pontos fonte sobre o lado de abaixo
45
46 hfd=4/(2*tf+1);
47 pfdy=-2:hfd:2;
48 pfdy=pfdy(1,2:(end-1));
49 pfdx=1*ones(2*tf,1);
50 pfd=[pfdx, pfdy']; %pontos fonte lado direito de abaixo para acima
51
52 hfa=2/(tf+1);
53 pfax=1:-hfa:-1;
54 pfax=pfax(1,2:(end-1));
55 pfa=[pfax' , 2*ones(tf,1)];%pontos fonte sobre o lado de acima
56
57 pf=[pfi;pfb;pfd;pfa]; %pontos fonte totais comecando do lado ...
       esquerdo
58
59 %pontos de colocacao sobre (-0.5,0.5)x(-1,1)
60 hg2=1/(tc+1);
61 xcg2=-0.5:hg2:0.5;
62 xcg2=xcg2(1,2:(end-1));
63 pcg2=[xcg2' , ones(tc,1)];%pontos colocacao sobre gamma_2 (nunca ...
       mudam em cada iteracao)
64
65 %% SOLUCAO DO PROBLEMA
66 % RESTRICOES
67 lb=[-Inf*ones(M,1);-Inf*ones(M,1);...
68 -0.9*ones(2,1);-0.9*ones(tc,1);-0.9*ones(2,1)];
69 ub=[Inf*ones(M,1);Inf*ones(M,1);0.9*ones(2,1);...
70 0.9*ones(tc,1);0.9*ones(2,1)];
71 % valores iniciais para as incognitas
72 lamb1=1.e-10 ; %parametro regularizacao solucao no dominio \Omega1
73 lamb2=1.e-05; %parametro regularizacao solucao no dominio \Omega2
74 lamb3=0;      %regularizacao fronteira
75 ruido=5;                          % ...
       ---------------------------------------- RUIDO
76 vruido=ruido*(2*rand(1,4*tc)-1)/100;  % ...
       ---------------------------------------- VETOR RUIDO
```

```
77 f2=0;
78 hs1=0; hg1=0; hs2=0; hg3=0;
79 gamma1=1; gamma2=0.01;
80 a=0.1*ones(M,1);%valor inicial para coeficientes da solucao u_1
81 b=0.1*ones(M,1);%valor inicial para coeficientes da solucao u_2
82 ygs1_0=0.1; % Considerar : ygs1_0-ygg1_0 ≤ 0.1
83 ygg1_0=0;
84 ygs2_0=0.1; % Considerar : ygs2_0-ygg3_0 ≤ 0.1
85 ygg3_0=0;
86 % para inicio recta
87 yg=0.1*ones(tc,1);
88 x_0=[a;b;ygs1_0;ygg1_0;yg;ygs2_0;ygg3_0]; % fronteira inicial ...
       considerada
89 %----------------------
90 % para inicio sin
91 %yg=0.1*sin(2*pi*1*xcg2);
92 %x_0=[a;b;ygs1_0;ygg1_0;yg';...
93 ygs2_0;ygg3_0]; % fronteira inicial considerada
94 %----------------------
95 options = optimoptions('fmincon','Display',...
96 'iter','MaxFunEvals',100000,'TolFun',1.e-6);% acrescentar ...
       MaxFunEvals para mais iteracoes
97 [x fval]=fmincon(@objectf,x_0,[],[],[],[],lb,ub,@nonlcon,options);
98
99 %% COMPARACAO GRAFICAS
100 freal=fronteira*ones(1,tc);
101 figure(1)
102 %plot(xcg2,x(2*M+3:(end-2)),'c-x',xcg2,...
103 freal,'.--',xcg2,yg','b-',[-0.5 0.5 0.5 -0.5 -0.5],[1 1 -1 -1 ...
       1],'k-',[-0.5 0.5],[x(2*M+1) x(end-1)],'ro',[-0.5 ...
       0.5],[x(2*M+2) x(end)],'go');
104 plot(xcg2,x(2*M+3:(end-2)),'ro--',xcg2,freal,...
105 'b-',xcg2,yg','g.--',[-0.5 0.5 0.5 -0.5 -0.5],[1 1 -1 -1 1],'k-');
106 xlabel('x');
107 ylabel('y');
108 legend('Computed Boundary','Desired Boundary','Initial guess');
109 %title('Reconstrucao das superficies real e reconstruida');
110 fprintf('|sr-sa|:%f\n',norm(x(2*M+3:(end-2))-freal));
111
112 %% COMPARACAO SOLUCOES
113 %{
114 %pontos de teste sobre as fornteiras
115
116 %para a parte \Sigma_1
117 ycs1p=x(2*M+1);
118 hycs1p=(ycs1p-1)/(tc);
119 ycs1yp=1:hycs1p:ycs1p;
```

```
ycs1yp=ycs1yp(1,2:(end));
pcs1p=[-0.5*ones(tc,1),ycs1yp'];

%para a parte \Gamma_1
ycg1p=x(2*M+2);
hycg1p=(-1-ycg1p)/(tc);
ycg1yp=ycg1p:hycg1p:-1;
ycg1yp=ycg1yp(1,1:(end-1));
pcg1p=[-0.5*ones(tc,1),ycg1yp'];

% para \Gamma_2 ja foi definido fora

%para \Gamma_3
ycg3p=x(end);
hycg3p=(ycg3p+1)/(tc);
ycg3yp=-1:hycg3p:ycg3p;
ycg3yp=ycg3yp(1,2:(end));
pcg3p=[-0.5*ones(tc,1),ycg3yp'];

%para \Sigma_2
ycs2p=x(end-1);
hycs2p=(1-ycs2p)/(tc);
ycs2yp=ycs2p:hycs2p:1;
ycs2yp=ycs2yp(1,1:(end-1));
pcs2p=[-0.5*ones(tc,1),ycs2yp'];

%% PARA A FRONTEIRA /Sigma_1
s1pa=fao1(pcs1p(:,2));
tam=size(pcs1p);
vee=1:tam;
for i=1:tam(1)
    s1pn(i)=fno2(x,pcs1p(i,:));
end
plot(vee,s1pa','r-',vee,s1pn,'b-o');
legend('Solucao analitica','Solucao numerica');
%}

%% PARA O CIRCULO /partial B((0,1/2),1/4)=B_2 contido em ...
    \Omega_1 e B((0,-1/2),1/4)=B_1 contido em \Omega_2
cp=30; ynumericao1=zeros(1,cp); ynumericao2=ynumericao1;
ynumericano1=ynumericao1; ynumericano2=ynumericao1;
yanaliticano1=ynumericao1;yanaliticano2=ynumericao1;
for j=1:cp
    pppo1(j,:)=[(1/4)*cos(2*pi*j/cp),(1/4)*sin(2*pi*j/cp)-1/2]; ...
        % circulo B_1 em \Omega_1
    pppo2(j,:)=[(1/4)*cos(2*pi*j/cp),(1/4)*sin(2*pi*j/cp)+1/2]; ...
        % circulo B_2 em \Omega_2
```

```
164     appp(j)=2*pi*j/cp;
165     % solucoes numericas
166     ynumericao1(j)=fno1(x,pppo1(j,:));
167     ynumericao2(j)=fno2(x,pppo2(j,:));
168     % derivada normal das solucoes numericas
169     ynumericano1(j)=fndno1(x,pppo1(j,:),[cos(appp(j)) , ...
            sin(appp(j))]);
170     ynumericano2(j)=fndno2(x,pppo2(j,:),[cos(appp(j)) , ...
            sin(appp(j))]);
171     % derivada normal das solucoes analiticas
172     yanaliticano1(j)=fadno1([cos(appp(j)) , sin(appp(j))]);
173     yanaliticano2(j)=fadno2([cos(appp(j)) , sin(appp(j))]);
174 end
175 % solucoes analiticas
176 yanaliticao1=fao1(pppo1(:,2));
177 yanaliticao2=fao2(pppo2(:,2));
178
179 figure(2)
180 subplot(2,2,1)
181 plot(appp,yanaliticao1,'r-',appp,ynumericao1,'b-o');
182 legend('Solucao analitica u_1','Solucao numerica u_{1MFS}');
183 subplot(2,2,2)
184 plot(appp,yanaliticano1,'r-',appp,ynumericano1,'b-o');
185 legend('Solucao analitica \partial u_1/\partial n','Solucao ...
        numerica \partial u_{1MFS}/\partial n');
186
187 subplot(2,2,3)
188 plot(appp,yanaliticao2,'r-',appp,ynumericao2,'b-o');
189 legend('Solucao analitica u_2','Solucao numerica u_{2MFS}');
190 subplot(2,2,4)
191 plot(appp,yanaliticano2,'r-',appp,ynumericano2,'b-o');
192 legend('Solucao analitica \partial u_2/\partial n','Solucao ...
        numerica \partial u_{2MFS}/\partial n');
193 suptitle('Comparacao das solucoes analiticas e numericas sobre ...
        \partial B_1 , \partial B_2 contidas em \Omega_1 e \Omega_2');
194
195
196 %% Definicao das funcoes envolvidas no problema
197 %% Condicoes de fronteira
198 function sf=f(y) % Para /Gamma1 e /Gamma3
199 global gamma1 gamma2
200     sf=-(gamma2/gamma1)*(y+1);
201 end
202
203 function sg=g(y) % Para /Sigma1 e /Sigma2
204 global gamma1 gamma2 fronteira
205     sg=y+1-(gamma2/gamma1+1)*(fronteira+1);
```

```
206 end
207 %% Funcoes analiticas
208 % Funcao analitica do problema u1
209 function yfao1=fao1(yy) %% yy : componente y do vetor (x,y) a ...
       avaliar
210 global gamma1 gamma2
211     yfao1=-(gamma2/gamma1)*(yy+1);
212 end
213 % Funcao analitica derivada normal du1/dn
214 function yfadno1=fadno1(vcn) % vcn : vetor normal fila
215 global gamma1 gamma2
216     yfadno1=[0 , -(gamma2/gamma1)]*vcn';
217 end
218
219 % Funcao analitica do problema u2
220 function yfao2=fao2(yy)
221 global gamma1 gamma2 fronteira
222     yfao2=yy+1-(gamma2/gamma1+1)*(fronteira+1);
223 end
224 % Funcao analitica derivada normal du2/dn
225 function yfadno2=fadno2(vcn) % vcn : vetor normal fila
226 global gamma1 gamma2
227     yfadno2=[0 , 1]*vcn';
228 end
229 %% Funcoes numericas
230 % Funcao numerica u1_MFS
231 function yfno1=fno1(solu,vet) % vet: vetor de  onde avaliar u1_MFS
232 global pf
233     ss=0; tt=size(pf);
234     for k=1:tt(1)
235        ss=ss+solu(k)*log(norm(vet-pf(k,:)));
236     end
237     yfno1=ss;
238 end
239 % Funcao numerica derivada normal du1_MFS/dn
240 function yfndno1=fndno1(solu,vet,vetn) % vet: vetor a avaliar
241 global pf                              % vetn vetor normal fila
242     ss=0; tt=size(pf);
243     for k=1:tt(1)
244         ss=ss+solu(k)*(1/(norm(vet-pf(k,:)))^2)*...
245         ((vet-pf(k,:))*vetn');
246     end
247     yfndno1=ss;
248 end
249
250 % Funcao numerica u2_MFS
251 function yfno2=fno2(solu,vet) % vet: vetor onde avaliar u2_MFS
```

```
252 global pf M
253     ss=0; tt=size(pf);
254     for k=1:tt(1);
255        ss=ss+solu(k+M)*log(norm(vet-pf(k,:)));
256     end
257     yfno2=ss;
258 end
259
260 % Funcao numerica derivada normal du2_MFS/dn
261 function yfndno2=fndno2(solu,vet,vetn) % vet: vetor a avaliar
262 global pf M                             % vetn vetor normal fila
263     ss=0; tt=size(pf);
264     for k=1:tt(1)
265         ss=ss+solu(k+M)*(1/(norm(vet-pf(k,:)))^2)*...
266         ((vet-pf(k,:))*vetn');
267     end
268     yfndno2=ss;
269 end
270
271
272 %% Definicao da funcao objetivo
273 function yobjectf=objectf(x)
274 global M N tc tf pcs1 pcs2 pcg1 pcg3 pfi pfd pfa pfb pcg2 pf f1 ...
       f2 f3 g1 g2 hs1 hg1 hs2 hg3 gamma1 gamma2 vruido lamb1 lamb2 ...
       lamb3
275 %pontos colocacao
276 ycs1=x(2*M+1);
277 hycs1=(ycs1-1)/(tc);
278 ycs1y=1:hycs1:ycs1;
279 ycs1y=ycs1y(1,2:(end));
280 pcs1=[-0.5*ones(tc,1),ycs1y']; %para a parte \Sigma_1
281
282 ycg1=x(2*M+2);
283 hycg1=(-1-ycg1)/(tc);
284 ycg1y=ycg1:hycg1:-1;
285 ycg1y=ycg1y(1,1:(end-1));
286 pcg1=[-0.5*ones(tc,1),ycg1y']; %para a parte \Gamma_1
287
288 % para \Gamma_2 ja foi definido fora
289
290 ycg3=x(end);
291 hycg3=(ycg3+1)/(tc);
292 ycg3y=-1:hycg3:ycg3;
293 ycg3y=ycg3y(1,2:(end));
294 pcg3=[-0.5*ones(tc,1),ycg3y']; %para \Gamma_3
295
296 ycs2=x(end-1);
```

```
hycs2=(1-ycs2)/(tc);
ycs2y=ycs2:hycs2:1;
ycs2y=ycs2y(1,1:(end-1));
pcs2=[-0.5*ones(tc,1),ycs2y']; %para \Sigma_2

pcg=[pcg2(:,1), x(2*M+3:(end-2))];

s1=0; s2=0; s3=0; s4=0; s5=0; s6=0; s7=0; s8=0; s9=0; s10=0; s11=0;
ra=norm(x(1:M))^2; % regularizacao MFS para solucao em \Omega_1
rb=norm(x((M+1):(2*M)))^2; % regularizacao MFS em \Omega_2
rf=norm(x((2*M+3):(end-2)))^2; % regularizacao MFS \Gamma
for i=1:tc
    auxs1=zeros(M,1);auxs2=zeros(M,1);auxds3=zeros(M,1);...
    auxds4=zeros(M,1);auxs5=zeros(M,1);
    auxs6=zeros(M,1);auxs7=zeros(M,1);auxs8=zeros(M,1);...
    auxs9=zeros(M,1);auxs10=zeros(M,1);auxs11=zeros(M,1);
    for j=1:M
       auxs1(j)=x(j)*log(norm(pcg1(i,:)-pf(j,:)));
       auxs2(j)=x(j)*log(norm(pcg2(i,:)-pf(j,:)));
       auxs3(j)=x(j)*log(norm(pcg3(i,:)-pf(j,:)));
       auxs4(j)=x(j+M)*log(norm(pcs1(i,:)-pf(j,:)));
       auxs5(j)=x(j+M)*log(norm(pcs2(i,:)-pf(j,:)));
       auxs6(j)=(x(j)-x(j+M))*log(norm(pcg(i,:)-pf(j,:)));
       auxs7(j)=(gamma1*x(j)-gamma2*x(j+M))*...
       (1/(norm(pcg(i,:)-pf(j,:)))^2)*((pcg(i,:)-pf(j,:))*[0,1]');
       auxs8(j)=gamma1*x(j)*(1/(norm(pcg1(i,:)-pf(j,:)))^2)*...
       (pcg1(i,:)-pf(j,:))*[-1,0]';
       auxs9(j)=gamma1*x(j)*(1/(norm(pcg3(i,:)-pf(j,:)))^2)*...
       (pcg3(i,:)-pf(j,:))*[1,0]';
       auxs10(j)=gamma2*x(j+M)*...
       (1/(norm(pcs1(i,:)-pf(j,:)))^2)*(pcs1(i,:)-pf(j,:))*[-1,0]';
       auxs11(j)=gamma2*x(j+M)*(1/(norm(pcs2(i,:)-pf(j,:)))^2)*...
       (pcs2(i,:)-pf(j,:))*[1,0]';
    end
    s1=s1+(sum(auxs1)-f(pcg1(i,2)))^2;
    s2=s2+(sum(auxs2)-f2)^2;
    s3=s3+(sum(auxs3)-f(pcg3(i,2)))^2;
    s4=s4+(sum(auxs4)-g(pcs1(i,2)))^2;
    s5=s5+(sum(auxs5)-g(pcs2(i,2)))^2;
    s6=s6+(sum(auxs6))^2; % cond fronteira desc continuidade
    s7=s7+(sum(auxs7))^2; % cond fronteira derivada assumindo ...
       que a fronteira desconhecida e escalonada por partes ...
       para ter vetor normal (0,1) ou (0-1)
    s8=s8+(sum(auxs8)-vruido(i))^2;
    s9=s9+(sum(auxs9)-vruido(i+tc))^2;
    s10=s10+(sum(auxs10)-vruido(i+2*tc))^2;
    s11=s11+(sum(auxs11)-vruido(i+3*tc))^2;
```

```
342 end
343 yobjectf=s1+s2+s3+s4+s5+s6+s7+s8+s9+s10+s11+...
344 lamb1*ra+lamb2*rb+lamb3*rf;
345 end
346 %% Restricoes
347 function [c , ceq]=nonlcon(x)
348 global M
349 c(1)=x(2*M+2)-x(2*M+1)+0.0001;
350 c(2)=x(end)-x(end-1)+0.0001;
351 c(3)=x(2*M+1)-x(2*M+2)-0.2;
352 c(4)=x(end-1)-x(end)-0.2;
353 ceq=[];
354 end
```

A.2.2 Segunda Proposta

Algoritmo modificado para otimizar a Primeira Proposta mediante o uso de um funcional que envolve apenas os pontos na fronteira alvo.

Caso 2D.

```
1  %% Funcao wetAreaf para testes
2  function [curvaSol history]=wetAreaf(orderMFS,orderF,reg_fun,noise)
3  %% Problema exemplo agua para funcional J= ...
       \sum_1_4(\gamma_jdu_MFS/dn - h_data)^2
4  % \gamma_1\Delta u_1=0 em \Omega_1
5  % \gamma_2\Delta u_2=0 em \Omega_2
6  % u_1 = -\gamma (y+1) sobre \Gamma_1
7  % u_1 = 0 sobre \Gamma_2
8  % u_1 = -\gamma*(y+1) sobre \Gamma_3
9  % u_2 = y+1-(\gamma+1)*s sobre \Sigma_1
10 % u_2 = y+1-(\gamma+1)*s sobre \Sigma_2
11 % u_1 = u_2 sobre \Gamma  (continuidade)
12 % \gamma_1 du_1/n = \gamma_2 du_2/n  sobre \Gamma (salto da ...
       derivada normal)
13 % Dominio considerado (-1,1)x(-1,1)
14 % \gamma=\gamma2/\gamma1
15 % solucao analitica :
16 % u1= -\gamma*(y) ; u2=(y)-(\gamma+1)(s)
17
18 %% PARAMETROS
19 global gamma1 gamma2 M N nf nc pf xc pcg2 fronteira lambr ...
       solfinal reg P1 vruido P2 regMFS
20 gamma1=1; gamma2=0.01;
21 R=2; % raio para os pontos fonte ( R > 0.5 )
```

```
22 M=56; % pontos fonte multiplo de 4 (ou mult 4 e 7 para igualdade ...
       N=10*M/7)
23 N=80; % pontos para gerar 6*N/5 ptos de colocacao. N multiplo de ...
       5.(N ≥0 existencia sol Funcional)
24       % N ≥ 10*M/7 para existencia solucoes MFS.
25 % Exemplos (M=28,N=40) (M=56,N=80) (M=84,N=120) (M=112,N=160)... ...
       Casos para igualdade N=10*M/7
26 nf=M/4; % pontos fonte por lado
27 nc=N/5; % pontos coloc por lado
28 %-------------------------------
29 % Ruido
30 ruido=noise;% Percentagem de ruido
31 vruido=ruido*(2*rand(1,4*nc)-1)/100; % Vetor Ruido
32 %-------------------------------
33 % Pontos Fonte sentido antihorario
34 ypf=linspace(0.5+R,0.5-R,nf+2);
35 xpf=linspace(0.5-R,0.5+R,nf+2);
36 pfli=[(0.5-R)*ones(nf,1),ypf(2:end-1)']; % pontos fonte lado ...
       esquerdo
37 pfld=[(0.5+R)*ones(nf,1),ypf(end-1:-1:2)']; %pontos fonte lado ...
       direito
38 pfb=[xpf(2:end-1)' , (0.5-R)*ones(nf,1)]; % pontos fonte abaixo
39 pfa=[xpf(end-1:-1:2)' , (0.5+R)*ones(nf,1)]; % pontos fonte acima
40 pf=[pfli;pfb;pfld;pfa]; %pontos fonte
41 %-----------------------------------------
42 % Coordenadas em x para pontos coloc em \Gamma e \Gamma_2 NAO ...
       mudam em cada iteracao
43 xcaux=linspace(0,1,nc+2);
44 xc=xcaux(2:end-1);
45 %---------------------------
46 % Pontos colocacao para \Gamma2 (nao mudam cada iter)
47 pcg2=[xc' , zeros(nc,1)];
48 %-------------------------
49 %% Matrizes de regularizacao
50 switch(orderF)
51     case 0
52             % Matriz de regularizacao Tikhnov ordem 0 para o ...
                   funcional
53             P1=eye(nc);
54     case 1
55             % Matriz de regularizacao Tikhnov ordem 1 para o ...
                   funcional
56             m1=diag(-1*ones(1,nc));
57             m1=m1(1:end-1,:);
58             m2=diag(ones(1,nc),1);
59             m2=m2(1:nc-1,1:nc);
60             P1=m1+m2;
```

```
61      case 2
62              % Matriz de regularizacao Tikhnov ordem 2 para o ...
                   funcional
63              P1a=eye(nc)+diag(-2*ones(1,nc-1),1)+...
64              diag(ones(1,nc-2),2);
65              P1=P1a(1:nc-2,:);
66  end
67  %-------------------------------------
68  switch(orderMFS)
69      case 0
70              % Matriz de regularizacao Tikhnov ordem 0 para o MFS
71              P2=eye(2*M);
72      case 1
73              % Matriz de regularizacao Tikhnov ordem 1 para o MFS
74              mm1=diag(-1*ones(1,2*M));
75              mm1=mm1(1:end-1,:);
76              mm2=diag(ones(1,2*M),1);
77              mm2=mm2(1:2*M-1,1:2*M);
78              P2=mm1+mm2;
79      case 2
80              % Matriz de regularizacao Tikhnov ordem 2 para o MFS
81              P2a=eye(2*M)+diag(-2*ones(1,2*M-1),1)+...
82              diag(ones(1,2*M-2),2);
83              P2=P2a(1:2*M-2,:);
84  end
85  %-------------------------------------------------------
86
87  %% SOLUCAO DO PROBLEMA
88  history=[];
89  %regMFS=1.0e-05;  % Parametro da Regularizacao do MFS
90  reg=reg_fun ;    % Parametro da Regularizacao do funcional
91  %lambr=1.0e-04;  % Parametro para regularizacao da matriz do MFS.
92  s_0=0.1*ones(1,nc);        % Fronteira inicial considerada constante
93  amp=0.1;                    % Amplitude fronteira inicial
94  T=1/1;                      % Periodo de oscilacao
95  %s_0=0.2+amp*sin((2*pi/T)*xc); % Fronteira inicial considerada
96  x_0=s_0';                   % Chute inicial
97  lb=0.1*ones(nc,1);             % Limite inferior para a fronteira
98  ub=0.9*ones(nc,1);          % Limite superior para a fronteira
99  fronteira=0.5; % Fronteira solucao s, 0 < fronteira < 1
100 options = optimset('Outputfcn',@outfun,...
101 'Algorithm','active-set','Display','iter-detailed',...
102 'MaxFunEvals',20000,'TolFun',1.e-10,'TolX',1.e-10,'TolCon',1.e-10);
103 %options = optimoptions('OutputFcn',@outfun,...
104 'fmincon','Display','iter','MaxFunEvals',50000,...
105 'TolFun',1.0e-10,'TolX',1.0e-30,'StepTolerance',1.0e-50,...
106 'MaxIterations',2000);
```

```
%options = optimoptions('OutputFcn',@outfun,'fmincon',...
'Display','iter','MaxFunEvals',20000,'TolFun',...
1.0e-30,'TolX',1.0e-30,'ConstraintTolerance',...
1e-30,'MaxIterations',2000);% acrescentar MaxFunEvals para mais ...
    iteracoes
[curvaSol, fval,exitflag,output]=...
fmincon(@objectf,x_0,[],[],[],[],lb,ub,[],options);
%% COMPARACAO DAS SUPERFICIES
figure(1);
%plot(xc,fronteira*ones(nc,1),'k-x',xc,x,'k-o',...
xc,s_0,'k-.',pf(:,1),pf(:,2),'k*',[0 0 1 1],[1 0 0 1],'k-');
plot(xc,fronteira*ones(nc,1),'k-x',xc,curvaSol,'k:o',xc,...
s_0,'k--s',[0 0 1 1],[1 0 0 1],'k-');
xlabel('x');
ylabel('y');
legend('Desired boundary','Computed boundary','Initial guess');
%saveas(gcf,'noise_s1_4.pdf');
fprintf('|s_0-xSol|= %f \n',norm(curvaSol-0.5));
%% COMPARACAO DAS SOLUCOES
%-------------------------------------------------------
% PARA O CIRCULO /partial B((0,1/2),1/4)=B_2 contido em \Omega_1 ...
    e B((0,-1/2),1/4)=B_1 contido em \Omega_2
%-------------------------------------------------------
cp=20; ynumericao1=zeros(1,cp); ynumericao2=ynumericao1;
ynumericano1=ynumericao1; ynumericano2=ynumericao1;
yanaliticano1=ynumericao1;yanaliticano2=ynumericao1;
for j=1:cp
    pppo1(j,:)=[(1/16)*cos(2*pi*j/cp)+1/2,(1/16)*...
    sin(2*pi*j/cp)+1/8]; % circulo B_1 em \Omega_1
    pppo2(j,:)=[(1/8)*cos(2*pi*j/cp)+1/2,(1/8)*...
    sin(2*pi*j/cp)+3/4]; % circulo B_2 em \Omega_2
    appp(j)=2*pi*j/cp;
    % solucoes numericas
    ynumericao1(j)=umfs(solfinal(1:M),pppo1(j,:));
    ynumericao2(j)=umfs(solfinal(M+1:end),pppo2(j,:));
    % derivada normal das solucoes numericas
    ynumericano1(j)=dnumfs(solfinal(1:M),pppo1(j,:),...
    [cos(appp(j)) , sin(appp(j))]);
    ynumericano2(j)=dnumfs(solfinal(M+1:end),pppo2(j,:),...
    [cos(appp(j)) , sin(appp(j))]);
    % derivada normal das solucoes analiticas
    yanaliticano1(j)=fadno1([cos(appp(j)) , sin(appp(j))]);
    yanaliticano2(j)=fadno2([cos(appp(j)) , sin(appp(j))]);
end
% solucoes analiticas
yanaliticao1=fao1(pppo1(:,2));
yanaliticao2=fao2(pppo2(:,2));
```

```
152
153 figure(2)
154 subplot(2,2,1)
155 plot(appp,yanaliticao1,'r-',appp,ynumericao1,'b-o');
156 legend('Solucao analitica u_1','Solucao numerica u_{1MFS}');
157 subplot(2,2,2)
158 plot(appp,yanaliticano1,'r-',appp,ynumericano1,'b-o');
159 legend('Solucao analitica \partial u_1/\partial n','Solucao ...
       nummerica \partial u_{1MFS}/\partial n');
160
161 subplot(2,2,3)
162 plot(appp,yanaliticao2,'r-',appp,ynumericao2,'b-o');
163 legend('Solucao analitica u_2','Solucao numerica u_{2MFS}');
164 subplot(2,2,4)
165 plot(appp,yanaliticano2,'r-',appp,ynumericano2,'b-o');
166 legend('Solucao analitica \partial u_2/\partial n','Solucao ...
       numerica \partial u_{2MFS}/\partial n');
167 suptitle('Comparacao das solucoes analiticas e numericas sobre ...
       \partial B_1 , \partial B_2 contidas em \Omega_1 e \Omega_2');
168
169 %% Funcao de saida
170     function stop = outfun(curvaSol,optimValues,state)
171         stop = false;
172         if isequal(state,'iter')
173             history = [history;optimValues.fval];
174         end
175     end
176 figure(3)
177 plot(history,'Linewidth',1);
178 xlabel('Iterations');
179 ylabel('Cost function values');
180 saveas(gcf,'fvalues_s0_1.pdf');
181 end
182 %% Definicao das funcoes envolvidas no problema
183 %% Funcoes analiticas
184 % Funcao analitica do problema u1
185 function yfao1=fao1(yy) %% yy : componente y do vetor (x,y) a ...
       avaliar
186 global gamma1 gamma2
187     yfao1=-(gamma2/gamma1)*(yy);
188 end
189 % Funcao analitica derivada normal du1/dn
190 function yfadno1=fadno1(vcn) % vcn : vetor normal fila
191 global gamma1 gamma2
192     yfadno1=[0 , -(gamma2/gamma1)]*vcn';
193 end
194
```

```
% Funcao analitica do problema u2
function yfao2=fao2(yy)
global gamma1 gamma2 fronteira
    yfao2=yy-(gamma2/gamma1+1)*(fronteira);
end
% Funcao analitica derivada normal du2/dn
function yfadno2=fadno2(vcn) % vcn : vetor normal fila
global gamma1 gamma2
    yfadno2=[0 , 1]*vcn';
end
%% Solucao Fundamental
% Solucao fundamental \phi Laplace
function yphi=phi(xx,xi) % xx ponto, xi singularidade
    yphi=log(norm(xx-xi));
end

% Derivada normal da sol fundamental \phi
function ydnphi=dnphi(xx,xi,vn) % xx ponto, xi singul, vn vetor ...
    normal
    ydnphi=(1/(norm(xx-xi)^2))*((xx-xi)*vn');
end
%% Solucao numerica
% Solucao numerica u_MFS
function yumfs=umfs(coefs,xxx) % coefs: a para u1, b para u2
global pf M                    % xxx: ponto a avaliar
yumfs=0;
    for i=1:M
        yumfs=yumfs+coefs(i)*phi(xxx,pf(i,:));
    end
end

% Solucao numerica du_MFS/dn
function ydnumfs=dnumfs(coefs,xxx,vvn)
global pf M
ydnumfs=0;
    for i=1:M
        ydnumfs=ydnumfs+coefs(i)*dnphi(xxx,pf(i,:),vvn);
    end
end
%% Condicoes de fronteira
function sf=f(y) % f1, f3 Para /Gamma1 e /Gamma3 (f1=f3)
global gamma1 gamma2
    sf=-(gamma2/gamma1)*(y);
end

function sg=g(y) % Para /Sigma1 e /Sigma2
global gamma1 gamma2 fronteira
```

```
241     sg=y-(gamma2/gamma1+1)*(fronteira);
242 end
243 %% L-curve parametro para MFS
244 function [ylambMFS]=lambMFS(AA,bb,PP)
245 lambdas=[1.0e-10 ;1.0e-09 ;1.0e-08 ;1.0e-07 ;1.0e-06 ;1.0e-05];
246 npo=size(lambdas);
247     for k=1:npo(1)
248         solaux=(AA'*AA + lambdas(k)*(PP')*PP)\((AA')*bb);
249         mvalores(k,:)=[norm(AA*solaux-bb)^2 , norm(solaux)^2];
250     end
251     dista=(mvalores(:,1).^2+mvalores(:,2).^2).^(1/2);
252     indmin=find(dista==min(dista),npo(1));
253     ylambMFS=lambdas(indmin(1));
254   %plot(mvalores(:,1),mvalores(:,2),'b-*',mvalores(indmin(1),1),...
255   mvalores(indmin(1),2),'rs');
256 end
257 %% Funcao objetivo
258 function yobjectf=objectf(x) % x=(x_1 ,..., x_nc) \in R^{nc}
259 global gamma1 gamma2 M N nf nc pf xc pcg2 fronteira lambr ...
       solfinal reg P1 vruido P2 regMFS
260 %-------------------------------------------------------------------
261 % Definicao pontos de colocacao para \Sigma_1 \Sigma_2 \Gamma_1 ...
       \Gamma3
262 %-------------------------------------------------------------------
263 % (pcg2: ja definido sobre \Gamma_2) preench no sentido antihorario
264 pcg=[xc',x]; % pontos coloc em \Gamma
265
266 ypcs1=linspace(1,x(1)+0.05,nc+2);
267 pcs1=[zeros(nc,1),ypcs1(2:end-1)']; % pontos coloc \Sigma_1
268
269 ypcg1=linspace(x(1)-0.05,0,nc+2);
270 pcg1=[zeros(nc,1),ypcg1(2:end-1)']; % pontos coloc \Gamma_1
271
272 ypcg3=linspace(0,x(end)-0.05,nc+2);
273 pcg3=[1*ones(nc,1),ypcg3(2:end-1)']; % pontos coloc \Gamma_3
274
275 ypcs2=linspace(x(end)+0.05,1,nc+2);
276 pcs2=[1*ones(nc,1),ypcs2(2:end-1)']; % pontos coloc \Sigma_2
277 %-------------------------------------------------------------
278 % Matriz do sistema
279 %    |  Au11 |  Au12|
280 % A=|  Au21 |  Au22|b= [f1G1,f2G2,f3G3,g1S1,g2S2,bf,bc]' ...
       SISTEMA: Ax=b
281 %    |  Af1  |  Af2 |
282 %    |  Ac1  |  Ac2 |
283 Au12=zeros(3*nc,M); Au21=zeros(2*nc,M); bf=zeros(nc,1); bc=bf;
284 f1G1=f(pcg1(:,2));  % condicao sobre \Gamma_1
```

```
f2G2=zeros(nc,1);   % Condicao sobre \Gamma_2
f3G3=f(pcg3(:,2));  % Condicao sobre \Gamma_3
g1S1=g(pcs1(:,2));  % Condicao sobre \Sigma_1
g2S2=g(pcs2(:,2));  % Condicao sobre \Sigma_2
for i=1:nc
    for j=1:M
        % Para Au11
        Au11G1(i,j)=phi(pcg1(i,:),pf(j,:));
        Au11G2(i,j)=phi(pcg2(i,:),pf(j,:));
        Au11G3(i,j)=phi(pcg3(i,:),pf(j,:));

        % Para Au22
        Au22S1(i,j)=phi(pcs1(i,:),pf(j,:));
        Au22S2(i,j)=phi(pcs2(i,:),pf(j,:));

        % Para as condicoes de igualdade na fronteira \Gamma
        Af1(i,j)=phi(pcg(i,:),pf(j,:));

        % Para as condicoes de continuidade sobre \Gamma
        Acaux(i,j)=dnphi(pcg(i,:),pf(j,:),[0,1]);
    end
end

Af2=-Af1;
Ac1=gamma1*Acaux;
Ac2=-gamma2*Acaux;
Au11=[Au11G1;Au11G2;Au11G3];
Au22=[Au22S1;Au22S2];
A=[Au11,Au12;Au21,Au22;Af1,Af2;Ac1,Ac2];
b= [f1G1;f2G2;f3G3;g1S1;g2S2;bf;bc];

%{
b = (A')*b;
%A = A'*A + lambr*eye(size(A'*A));  %Tikhonov reg zeroth pa MFS
A = A'*A + regMFS*(P2')*P2;     %Tikhonov reg first pa MFS
sol = A\b;
%}
regMFS=lambMFS(A,b,P2);
%fprintf('lambdaMFS: %f \n',regMFS);
sol = (A'*A + regMFS*(P2')*P2)\((A')*b);
solfinal=sol;

%-----------------------------------------
% Funcional a minimizar
s1=0; s2=s1; s3=s1; s4=s1;
auxs1=zeros(M,1);auxs2=zeros(M,1);auxs3=zeros(M,1);
auxs4=zeros(M,1);
```

```
for i=1:nc
    for j=1:M
        auxs1(j)=sol(j)*dnphi(pcg1(i,:),pf(j,:),[-1,0]);
        auxs2(j)=sol(j)*dnphi(pcg3(i,:),pf(j,:),[1,0]);
        auxs3(j)=sol(j+M)*dnphi(pcs1(i,:),pf(j,:),[-1,0]);
        auxs4(j)=sol(j+M)*dnphi(pcs2(i,:),pf(j,:),[1,0]);
    end
    s1=s1+(gamma1*sum(auxs1)-vruido(i))^2;
    s2=s2+(gamma1*sum(auxs2)-vruido(i+nc))^2;
    s3=s3+(gamma2*sum(auxs3)-vruido(i+2*nc))^2;
    s4=s4+(gamma2*sum(auxs4)-vruido(i+3*nc))^2;
end
% yobjectf=s1+s2+s3+s4;
yobjectf=s1+s2+s3+s4+reg*norm(P1*x)^2;  % Tikhnov ordem 1 pa ...
    funcional
end
```

```
% Cria target e encontra as leituras para fronteira geral
% N:Pontos fonte (multip de 4) nf=N/4 por lado
% M:Pontos de colocac nc=M/4 em ...
    \Sigma_1\cup\Sigma_2,\Sigma_4\cup\Sigma_5,
%   2*nc em \Sigma_3, 2*(2*nc) em \Gamma (salto e continuidade)
% 8nc ≥ 2N (Existencia sol MFS M≥N)
% R:raio de pontos fonte com centro em (1/2,1/2)
clear all
%-------------------------------------------------------------
% PARAMETROS
global gamma1 gamma2 N N0 pf pf0 nf nc nc0 yc_f ycfq xc xc0 pcs3 ...
    pcs3_0 fyc gyc pqs1s2_T pqs5s4_T P0 PF lambF rangoMFS_0
gamma1=1; gamma2=0.01;
rangoMFS_T=[1.0e-08;1.0e-07;1.0e-06;1.0e-05];
rangoMFS_0=[1.0e-08;1.0e-07;1.0e-06;1.0e-05];
R=4;
N=64;
M=64;
nf=N/4;
nc=M/4;
R0=4;
N0=40;
M0=48;
nf0=N0/4;
nc0=M0/3;
%-------------------------------------------------------------
% PONTOS FONTE sentido antihorario TARGET
ypf=linspace(1/2+R,1/2-R,nf+2);
ypfi=ypf(2:end-1);
```

```
28 pfi=[(1/2-R)*ones(nf,1),ypfi']; % lado esquerdo
29 ypfd=ypf(end-1:-1:2);
30 pfd=[((1/2+R)*ones(nf,1)),ypfd']; % lado direito
31 xpf=linspace(1/2-R,1/2+R,nf+2);
32 xpfb=xpf(2:end-1);
33 xpft=xpf(end-1:-1:2);
34 pfb=[xpfb',(1/2-R)*ones(nf,1)]; % bottom source points
35 pft=[xpft',(1/2+R)*ones(nf,1)]; % top source points
36 pf=[pfi;pfb;pfd;pft]; % pontos fonte
37 %-------------------------------------------------------------
38 % PONTOS FONTE sentido antihorario para incognita
39 ypf0=linspace(1/2+R0,1/2-R0,nf0+2);
40 ypfi0=ypf0(2:end-1);
41 pfi0=[(1/2-R0)*ones(nf0,1),ypfi0']; % lado esquerdo
42 ypfd0=ypf0(end-1:-1:2);
43 pfd0=[((1/2+R0)*ones(nf0,1)),ypfd0']; % lado direito
44 xpf0=linspace(1/2-R0,1/2+R0,nf0+2);
45 xpfb0=xpf0(2:end-1);
46 xpft0=xpf0(end-1:-1:2);
47 pfb0=[xpfb0',(1/2-R0)*ones(nf0,1)]; % bottom source points
48 pft0=[xpft0',(1/2+R0)*ones(nf0,1)]; % top source points
49 pf0=[pfi0;pfb0;pfd0;pft0]; % pontos fonte incognita
50 %-------------------------------------------------------------
51 % PONTOS COLOCAC PARA O TARGET
52 % Posicao impares para eletrodos f(e_i), pares fora para fazer ...
       leituras q_i
53 xc=linspace(0,1,2*nc+2);
54 xc=xc(2:end-1); % coord x para pontos colocacao
55 ycfq=linspace(1,0,2*nc+2);
56 ycfq=ycfq(2:end-1); % coord y para pontos coloc f e leituras q
57 yc_f=ycfq(1:2:end); % coord y para coloc sobre eletrodos f(e)
58 %yc_q=ycfq(2:2:end); % coord y para medidas q
59 %-------------------------------------------------------------
60 % Superficie Target
61 amp=0.1;
62 T=1;
63 s=0.5+amp*sin((2*pi/T)*xc);
64 %-------------------------------------------------------------
65 pcg=[xc',s'];                              % pontos coloc sobre \Gamma
66 pcs3=[xc',zeros(2*nc,1)];                % pontos coloc sobre \Sigma_3
67 lit=0.5+amp*sin((2*pi/T)*0);
68 ldt=0.5+amp*sin((2*pi/T)*1);
69
70 ipcs1_f=find(yc_f>lit);      % index pontos coloc \Sigma_1
71 ncs1_f=size(ipcs1_f);
72 ncs1_f=ncs1_f(2);            % num de pontos coloc para \Sigma_1
73 ypcs1_f=yc_f(1:ncs1_f);
```

```
74 pcs1=[zeros(ncs1_f,1),ypcs1_f'];    % pontos coloc sobre \Sigma_1
75
76 ipcs1_q=find(ycfq>lit);     % index pontos para leituras q
77 nqs1_q=size(ipcs1_q);
78 nqs1_q=nqs1_q(2);           % num de pontos para leituras q ...
      sobre \Sigma_1
79 ypqs1_q=ycfq(1:nqs1_q);
80 pqs1=[zeros(nqs1_q,1),ypqs1_q']; % pontos para leituras q sobre ...
      \Sigma_1
81
82 ypcs2_f=yc_f(ncs1_f+1:end);
83 ncs2_f=nc-ncs1_f;
84 pcs2=[zeros(ncs2_f,1),ypcs2_f'];    % pontos coloc sobre \Sigma_2
85
86 ypqs2_q=ycfq(nqs1_q+1:end);
87 nqs2_q=2*nc-nqs1_q;
88 pqs2=[zeros(nqs2_q,1),ypqs2_q'];      % pontos leituras sobre ...
      \Sigma_2
89
90 ipcs5_f=find(yc_f>ldt);     % index pontos coloc \Sigma_5
91 ncs5_f=size(ipcs5_f);
92 ncs5_f=ncs5_f(2);
93 ypcs5_f=yc_f(1:ncs5_f);
94 pcs5=[ones(ncs5_f,1),ypcs5_f'];        % pontos coloc sobre \Sigma_5
95
96 ipqs5_q=find(ycfq>ldt);     % index para leituras sobre \Sigma_5
97 nqs5_q=size(ipqs5_q);
98 nqs5_q=nqs5_q(2);
99 ypqs5_q=ycfq(1:nqs5_q);
100 pqs5=[ones(nqs5_q,1),ypqs5_q'];     % pontos leitura sobre \Sigma_5
101
102 ypcs4_f=yc_f(ncs5_f+1:end);
103 ncs4_f=nc-ncs5_f;
104 pcs4=[ones(ncs4_f,1),ypcs4_f'];       % pontos coloc sobre \Sigma_4
105
106 ypqs4_q=ycfq(nqs5_q+1:end);
107 nqs4_q=2*nc-nqs5_q;
108 pqs4=[ones(nqs4_q,1),ypqs4_q'];      % pontos leitura sobre \Sigma_4
109
110 %plot(xc,s,'*-',0,lit,'rp',1,ldt,'rs',pcs1(:,1),pcs1(:,2),'k*',...
111 %    pcs2(:,1),pcs2(:,2),'kh',pcs5(:,1),pcs5(:,2),'k*',...
112 %    pcs4(:,1),pcs4(:,2),'kh',pcs3(:,1),pcs3(:,2),'ko',...
113 %    pqs1(:,1),pqs1(:,2),'g*',...
114 %    pqs2(:,1),pqs2(:,2),'ch',pqs5(:,1),pqs5(:,2),'g*',...
115 %    pqs4(:,1),pqs4(:,2),'ch')
116 %-----------------------------------------
117 % Pontos de colocacao para incognita
```

```
118 xc0=linspace(0,1,nc0+2);
119 xc0=xc0(2:end-1);        % coord x para pontos colocacao
120 pcs3_0=[xc0',zeros(nc0,1)];         % pontos coloc sobre \Sigma_3
121 %-------------------------------------------------
122 %% Matrizes de regularizacao
123 orderF=2;  % Ordem regularizacao para o Funcional
124 orderMFS_0=2;  % Ordem regularizacao para o MFS
125 orderMFS_T=2; % Ordem regularizacao para o MFS no Target
126
127 switch(orderF)
128     case 0
129             % Matriz de regularizacao Tikhnov ordem 0 para o ...
                    funcional
130             PF=eye(nc0);
131     case 1
132             % Matriz de regularizacao Tikhnov ordem 1 para o ...
                    funcional
133             m1=diag(-1*ones(1,nc0));
134             m1=m1(1:end-1,:);
135             m2=diag(ones(1,nc0),1);
136             m2=m2(1:nc0-1,1:nc0);
137             PF=m1+m2;
138     case 2
139             % Matriz de regularizacao Tikhnov ordem 2 para o ...
                    funcional
140             P1a=eye(nc0)+diag(-2*ones(1,nc0-1),1)+...
141             diag(ones(1,nc0-2),2);
142             PF=P1a(1:nc0-2,:);
143 end
144 %-------------------------------------------------
145 switch(orderMFS_0)
146     case 0
147             % Matriz de regularizacao Tikhnov ordem 0 para o MFS
148             P0=eye(2*N0);
149     case 1
150             % Matriz de regularizacao Tikhnov ordem 1 para o MFS
151             mm1=diag(-1*ones(1,2*N0));
152             mm1=mm1(1:end-1,:);
153             mm2=diag(ones(1,2*N0),1);
154             mm2=mm2(1:2*N0-1,1:2*N0);
155             P0=mm1+mm2;
156     case 2
157             % Matriz de regularizacao Tikhnov ordem 2 para o MFS
158             P2a=eye(2*N0)+diag(-2*ones(1,2*N0-1),1)+...
159             diag(ones(1,2*N0-2),2);
160             P0=P2a(1:2*N0-2,:);
161 end
```

```
%-------------------------------------------------
switch(orderMFS_T)
    case 0
            % Matriz de regularizacao Tikhnov ordem 0 para o MFS T
            PT=eye(2*N);
    case 1
            % Matriz de regularizacao Tikhnov ordem 1 para o MFS T
            mt1=diag(-1*ones(1,2*N));
            mt1=mt1(1:end-1,:);
            mt2=diag(ones(1,2*N),1);
            mt2=mt2(1:2*N-1,1:2*N);
            PT=mt1+mt2;
    case 2
            % Matriz de regularizacao Tikhnov ordem 2 para o MFS T
            P3a=eye(2*N)+diag(-2*ones(1,2*N-1),1)+...
            diag(ones(1,2*N-2),2);
            PT=P3a(1:2*N-2,:);
end
%-------------------------------------------------
%% Solucao do sistema Target
%    |  As2   |  As20  |
% A=|  As3   |  As30  |b= [fs2,fs3,fs4,gs1,gs5,bc,bs]' SISTEMA: Ax=b
%    |  As4   |  As40  |Ao10=[As20;As30;As40]
%    |  As10  |  As1   |Ao20=[As10;As50]
%    |  As50  |  As5   |
%    |  Agc   |  -Agc  |
%    | g1*Ags | -g2*Ags|
fs2=f(pcs2(:,2)); % coloc sobre fronteira \Sigma_2
fs3=zeros(2*nc,1); % coloc sobre fronteira \Sigma_3
fs4=f(pcs4(:,2)); % coloc sobre fronteira \Sigma_4
gs1=g(pcs1(:,2)); % coloc sobre fronteira \Sigma_1
gs5=g(pcs5(:,2)); % coloc sobre fronteira \Sigma_5
bc=zeros(2*nc,1); bs=bc;
Ao10=zeros(2*nc+ncs2_f+ncs4_f,N);
Ao20=zeros(ncs1_f+ncs5_f,N);

for j=1:N
    for i=1:ncs2_f
        As2(i,j)=phi(pcs2(i,:),pf(j,:));
    end
    for k=1:ncs4_f
        As4(k,j)=phi(pcs4(k,:),pf(j,:));
    end
    for l=1:ncs1_f
        As1(l,j)=phi(pcs1(l,:),pf(j,:));
    end
    for m=1:ncs5_f
```

```
209         As5(m,j)=phi(pcs5(m,:),pf(j,:));
210     end
211     for n=1:2*nc % continuidade e salto e \Sigma3
212         As3(n,j)=phi(pcs3(n,:),pf(j,:));
213         Agc(n,j)=phi(pcg(n,:),pf(j,:));
214         Ags(n,j)=dnphi(pcg(n,:),pf(j,:),[0,1]);
215     end
216 end
217 Ao1=[As2;As3;As4]; Ao2=[As1;As5];
218 A=[Ao1,Ao10;Ao20,Ao2;Agc,-Agc;gamma1*Ags,-gamma2*Ags];
219 b=[fs2;fs3;fs4;gs1;gs5;bc;bs];
220 lamb_T=lambMFS(A,b,PT,rangoMFS_T);
221 sol_T= (A'*A + lamb_T*(PT')*PT)\((A')*b);
222 %-------------------------------------------------------
223 %% Leituras q na fronteira target
224 % Para \Sigma_1
225 for i=1:nqs1_q
226     pqs1(i,3)=gamma2*dnumfs(sol_T(N+1:end),pqs1(i,1:2),[-1,0],'t');
227 end
228 % Para \Sigma_2
229 for j=1:nqs2_q
230     pqs2(j,3)=gamma1*dnumfs(sol_T(1:N),pqs2(j,1:2),[-1,0],'t');
231 end
232 % Para \Sigma_5
233 for k=1:nqs5_q
234     pqs5(k,3)=gamma2*dnumfs(sol_T(N+1:end),pqs5(k,1:2),[1,0],'t');
235 end
236 % Para \Sigma_4
237 for l=1:nqs4_q
238     pqs4(l,3)=gamma1*dnumfs(sol_T(1:N),pqs4(l,1:2),[1,0],'t');
239 end
240 % pontos colunas 1,2 e leituras coluna 3, para lado esquero
241 pqs1s2_T=[pqs1;pqs2];
242 % pontos colunas 1,2 e leituras coluna 3, para lado direito
243 pqs5s4_T=[pqs5;pqs4];
244 %-------------------------------------------------------
245 % Valores das condicoes de fronteira
246 fyc=f(yc_f); % eletrodos avaliados na condicao f fronteira para ...
       \Omega_1
247 gyc=g(yc_f); % eletrodos avaliados na condicao g fronteira para ...
       \Omega_2
248 %-------------------------------------------------------
249
250 %% SOLUCAO DO PROBLEMA
251 %history=[];
252 lambF=10;    % Parametro da Regularizacao do funcional
253 sc=0.5+amp*sin((2*pi/T)*xc0);                    % Fronteira ...
```

```
    Target Para comparar com solucao
hl=ycfq(1)-ycfq(2);
lb=(0+3*hl)*ones(nc,1); % Limite inferior para a fronteira
ub=(1-4*hl)*ones(nc,1); % Limite superior para a fronteira
s_0=(0+4*hl)*ones(1,nc);                              % Fronteira ...
    inicial considerada constante
amp0=hl/2;   % Amplitude fronteira inicial
T0=1/1;     % Periodo de oscilacao
%s_0=4*hl+amp0*sin((2*pi/T0)*xc0);                    % Fronteira ...
    inicial considerada
x_0=s_0';                    % Chute inicial
options = optimset('Algorithm','active-set',...
'Display','iter-detailed','MaxFunEvals',50000,...
'TolFun',1.e-20,'TolX',1.e-20,'TolCon',1.e-20);
%options = optimset('Outputfcn',@outfun,...
'Algorithm','active-set','Display','iter-detailed',...
'MaxFunEvals',20000,'TolFun',1.e-10,'TolX',1.e-10,'TolCon',1.e-10);
%options = optimoptions('fmincon','Display','iter',...
'MaxFunEvals',50000,'TolFun',1.0e-50,'TolX',...
1.0e-50,'StepTolerance',1.0e-50,'ConstraintTolerance',1e-50,...
'MaxIterations',2000);
%options = optimoptions('OutputFcn',@outfun,...
'fmincon','Display','iter','MaxFunEvals',20000,...
'TolFun',1.0e-30,'TolX',1.0e-30,'ConstraintTolerance',...
1e-30,'MaxIterations',2000);% acrescentar MaxFunEvals para mais ...
    iteracoes
[curvaSol, fval,exitflag,output]=fmincon(@objectf,...
x_0,[],[],[],[],lb,ub,[],options);
%-------------------------------------------------------------
%% COMPARACAO DAS SUPERFICIES
figure(1);
plot(xc,s,'k-x',xc0,curvaSol,'k:o',xc0,s_0,'k--s',[0 0 1 1],[1 0 ...
    0 1],'k-');
xlabel('x');
ylabel('y');
legend('Desired boundary','Computed boundary','Initial guess');
%saveas(gcf,'noise_s1_4.png'); % en png
%set(gcf,'Units','Inches');    % en pdf
%pos = get(gcf,'Position');
%set(gcf,'PaperPositionMode','Auto','PaperUnits',...
'Inches','PaperSize',[pos(3), pos(4)])
%print(gcf,'prueba.pdf','-dpdf','-r0')
fprintf('|s_0-xSol|= %f \n',norm(curvaSol-sc'));
%-------------------------------------------------------------
%% Definicao das funcoes envolvidas no problema
%-------------------------------------------------------------
% Solucao fundamental \phi Laplace
```

```
296 function yphi=phi(xx,xi) % xx ponto, xi singularidade
297     yphi=log(norm(xx-xi));
298 end
299
300 % Derivada normal da sol fundamental \phi
301 function ydnphi=dnphi(xx,xi,vn) % xx ponto, xi singul, vn vetor ...
       normal
302     ydnphi=(1/(norm(xx-xi)^2))*((xx-xi)*vn');
303 end
304 %-------------------------------------------------------------
305 % Solucao numerica u_MFS
306 function yumfs=umfs(coefs,xxx,op) % coefs: a para u1, b para u2
307 global pf pf0 N N0                    % xxx: ponto a avaliar
308 yumfs=0;
309 switch op
310     case {'t','T'}
311         for inn=1:N
312             yumfs=yumfs+coefs(inn)*phi(xxx,pf(inn,:));
313         end
314     case {'x','X'}
315         for inn=1:N0
316             yumfs=yumfs+coefs(inn)*phi(xxx,pf0(inn,:));
317         end
318 end
319 end
320 % Solucao numerica du_MFS/dn
321 function ydnumfs=dnumfs(coefs,xxx,vvn,opp)
322 global pf pf0 N N0
323 ydnumfs=0;
324 switch opp
325     case {'t','T'}
326         for ind=1:N
327             ydnumfs=ydnumfs+coefs(ind)*dnphi(xxx,pf(ind,:),vvn);
328         end
329     case {'x','X'}
330         for ind=1:N0
331             ydnumfs=ydnumfs+coefs(ind)*dnphi(xxx,pf0(ind,:),vvn);
332         end
333 end
334 end
335 %-------------------------------------------------------------
336 % Condicoes de fronteira
337 function sf=f(y) % f1, f3 Para /Sigma_2 e /Sigma_4 (f1=f3)
338 global gamma1 gamma2
339     sf=-(gamma2/gamma1)*(y);
340 end
341
```

```
342 function sg=g(y) % Para /Sigma1 e /Sigma5
343 global gamma1 gamma2
344     sg=y-(gamma2/gamma1+1);
345 end
346 %-------------------------------------------------------
347 %% L-curve parametro para MFS
348 function [ylambMFS]=lambMFS(AA,bb,PP,rango)
349 lambdas=rango;
350 npo=size(lambdas);
351     for k=1:npo(1)
352         solaux=(AA'*AA + lambdas(k)*(PP')*PP)\((AA')*bb);
353         mvalores(k,:)=[norm(AA*solaux-bb)^2 , norm(solaux)^2];
354     end
355     dista=(mvalores(:,1).^2+mvalores(:,2).^2).^(1/2);
356     indmin=find(dista==min(dista),npo(1));
357     ylambMFS=lambdas(indmin(1));
358   %plot(mvalores(:,1),mvalores(:,2),'b-*',...
359   mvalores(indmin(1),1),mvalores(indmin(1),2),'rs');
360 end
361 %% Funcao objetivo
362 function yobjectf=objectf(x) % x=(x_1 ,..., x_nc) \in R^{nc}
363 global gamma1 gamma2 N0 nc0 xc0 pf0 yc_f ycfq pcs3_0 fyc gyc ...
       pqs1s2_T pqs5s4_T P0 PF lambF rangoMFS_0
364 % Definicao dos pontos de colocacao e leitura nas fronteiras
365 li0=x(1); % ponto limite para o lado esquerdo
366 ld0=x(end); % ponto limite para o lado direito
367 %-------------------------------------------------------
368 % Sobre \Sigma_3 nao mudam
369 %-------------------------------------------------------
370 % Sobre \Gamma
371 pcg_0=[xc0' x];
372 %-------------------------------------------------------
373 % Sobre \Sigma_1
374 ipcs1_f0=find(yc_f>li0);
375 ncs1_f0=size(ipcs1_f0);
376 ncs1_f0=ncs1_f0(2);           % num de pontos coloc para \Sigma_1
377 ypcs1_f0=yc_f(1:ncs1_f0);
378 pcs1_0=[zeros(ncs1_f0,1),ypcs1_f0']; % pontos coloc sobre \Sigma_1
379
380 ipcs1_q0=find(ycfq>li0);
381 nqs1_q0=size(ipcs1_q0);
382 nqs1_q0=nqs1_q0(2);   % num de pontos para leituras q sobre \Sigma_1
383 ypqs1_q0=ycfq(1:nqs1_q0);
384 pqs1_0=[zeros(nqs1_q0,1),ypqs1_q0'];  % pontos para leituras q ...
       sobre \Sigma_1
385 %-------------------------------------------------------
386 % Sobre \Sigma_2
```

```
387 ypcs2_f0=yc_f(ncs1_f0+1:end);
388 ncs2_f0=nc0-ncs1_f0;
389 pcs2_0=[zeros(ncs2_f0,1),ypcs2_f0'];  % pontos coloc sobre \Sigma_2
390
391 ypqs2_q0=ycfq(nqs1_q0+1:end);
392 nqs2_q0=2*nc0-nqs1_q0;
393 pqs2_0=[zeros(nqs2_q0,1),ypqs2_q0'];    % pontos leituras sobre ...
       \Sigma_2
394 %-------------------------------------------------------------
395 % Sobre \Sigma_5
396 ipcs5_f0=find(yc_f>ld0);  % index pontos coloc \Sigma_5
397 ncs5_f0=size(ipcs5_f0);
398 ncs5_f0=ncs5_f0(2);
399 ypcs5_f0=yc_f(1:ncs5_f0);
400 pcs5_0=[ones(ncs5_f0,1),ypcs5_f0'];   % pontos coloc sobre \Sigma_5
401
402 ipqs5_q0=find(ycfq>ld0);  % index para leituras sobre \Sigma_5
403 nqs5_q0=size(ipqs5_q0);
404 nqs5_q0=nqs5_q0(2);
405 ypqs5_q0=ycfq(1:nqs5_q0);
406 pqs5_0=[ones(nqs5_q0,1),ypqs5_q0'];  % pontos leitura sobre \Sigma_5
407 %-------------------------------------------------------------
408 % Sobre \Sigma_4
409 ypcs4_f0=yc_f(ncs5_f0+1:end);
410 ncs4_f0=nc0-ncs5_f0;
411 pcs4_0=[ones(ncs4_f0,1),ypcs4_f0'];    % pontos coloc sobre \Sigma_4
412
413 ypqs4_q0=ycfq(nqs5_q0+1:end);
414 nqs4_q0=2*nc0-nqs5_q0;
415 pqs4_0=[ones(nqs4_q0,1),ypqs4_q0'];     % pontos leitura sobre ...
       \Sigma_4
416 %-------------------------------------------------------------
417 % Solucao do sistema
418 %     |  As2_0   |  As20_0  |
419 % A_0=|  As3_0   |  As30_0  |    SISTEMA: A_0*x=b_0
420 %     |  As4_0   |  As40_0  |
421 %     |  As10_0  |  As1_0   |
422 %     |  As50_0  |  As5_0   |
423 %     |  Agc_0   |  -Agc_0  |
424 %     | g1*Ags_0 | -g2*Ags_0|
425 fs2_0=fyc(ncs1_f0+1:end)'; % coloc sobre fronteira \Sigma_2
426 fs3_0=zeros(nc0,1); % coloc sobre fronteira \Sigma_3
427 fs4_0=fyc(ncs5_f0+1:end)'; % coloc sobre fronteira \Sigma_4
428 gs1_0=gyc(1:ncs1_f0)'; % coloc sobre fronteira \Sigma_1
429 gs5_0=gyc(1:ncs5_f0)'; % coloc sobre fronteira \Sigma_5
430 bc_0=zeros(nc0,1); bs_0=bc_0;
431 Ao10_0=zeros(nc0+ncs2_f0+ncs4_f0,N0);
```

```
Ao20_0=zeros(ncs1_f0+ncs5_f0,N0);

for j=1:N0
    for i=1:ncs2_f0
        As2_0(i,j)=phi(pcs2_0(i,:),pf0(j,:));
    end
    for k=1:ncs4_f0
        As4_0(k,j)=phi(pcs4_0(k,:),pf0(j,:));
    end
    for l=1:ncs1_f0
        As1_0(l,j)=phi(pcs1_0(l,:),pf0(j,:));
    end
    for m=1:ncs5_f0
        As5_0(m,j)=phi(pcs5_0(m,:),pf0(j,:));
    end
    for n=1:nc0 % continuidade e salto e \Sigma3
        As3_0(n,j)=phi(pcs3_0(n,:),pf0(j,:));
        Agc_0(n,j)=phi(pcg_0(n,:),pf0(j,:));
        Ags_0(n,j)=dnphi(pcg_0(n,:),pf0(j,:),[0,1]);
    end
end
Ao1_0=[As2_0;As3_0;As4_0]; Ao2_0=[As1_0;As5_0];
A_0=[Ao1_0,Ao10_0;Ao20_0,Ao2_0;Agc_0,...
-Agc_0;gamma1*Ags_0,-gamma2*Ags_0];
b_0=[fs2_0;fs3_0;fs4_0;gs1_0;gs5_0;bc_0;bs_0];
lamb_0=lambMFS(A_0,b_0,P0,rangoMFS_0);
sol_0= (A_0'*A_0 + lamb_0*(P0')*P0)\((A_0')*b_0);
%-------------------------------------------------------------
% Funcional a minimizar
s1=0; s2=s1; s5=s1; s4=s1;
% para \Sigma_1
for ii=1:nqs1_q0
    s1=s1+(gamma2*dnumfs(sol_0(N0+1:end),...
    pqs1_0(ii,:),[-1,0],'x')-pqs1s2_T(ii,3))^2;
end
% para \Sigma_2
for jj=1:nqs2_q0
    s2=s2+(gamma1*dnumfs(sol_0(1:N0),...
    pqs2_0(jj,:),[-1,0],'x')-pqs1s2_T(jj+nqs1_q0,3))^2;
end
% para \Sigma_5
for kk=1:nqs5_q0
    s5=s5+(gamma2*dnumfs(sol_0(N0+1:end),...
    pqs5_0(kk,:),[1,0],'x')-pqs5s4_T(kk,3))^2;
end
% para \Sigma_4
for ll=1:nqs4_q0
```

```
479      s4=s4+(gamma1*dnumfs(sol_0(1:N0),...
480      pqs4_0(ll,:),[1,0],'x')-pqs5s4_T(ll+nqs5_q0,3))^2;
481  end
482  %yobjectf=s1+s2+s5+s4;
483  yobjectf=s1+s2+s5+s4+lambF*norm(PF*x)^2; ...
                                              % Tikhnov ordem 1 para ...
       o funcional
484  end
```

Caso 3D.

```
1   function ...
        wetArea3D_GA_prueba(superficie_Target,superficie_Inicial,pruido)
2   % Wet Area 3D using Genetic Algorithms
3   % superficie_Target: 1 (constante=60), 2 (5*sin(xxx)+5*cos(yyy)+60),
4   % superficie_Inicial: 1 (constante=15), 2 ...
        (5*sin(xxx)+5*cos(yyy)+10),
5   % pruido: percentagem ruido
6    clc;
7    %superficie_Target=2; superficie_Inicial=1; pruido=5;
8   %=======================================
9   % PARAMETROS
10  %---------------------------------------
11  global gamma1 gamma2 x_g y_g rangoMFS_0 lambF pf_t N_t pf_0 N_0 ...
        f_mg c_mg dados_C_xzA dados_C_xzB pc_s_xzA pc_s_xzB pc_s_yzA ...
        pc_s_yzB pc_s_yzAB xzA_1 xzB_1 b_t P0
12  % pf_0 N_0 vetor pontos fonte e numero de pontos fonte na incognita
13  %pf(centro,np,opc,r,vdim)
14  %opc: 1 esfera, np=mult 8 pontos fonte
15  %opc: 2 cubo, np= pontos fonte por lado, 2*r lado do cubo
16  %opc: 3 paralelepipedo, 2*np pontos sobre xz yz e np sobre xy ...
        (no usa r)
17  gamma1=1; gamma2=0.01;
18  lambF=0;
19  rangoMFS_T=[1.0e-08;1.0e-07;1.0e-06;1.0e-05;...
20  1.0e-04;1.0e-03;1.0e-02];
21  rangoMFS_0=[1.0e-08;1.0e-07;1.0e-06;1.0e-05;...
22  1.0e-04;1.0e-03;1.0e-02];
23  centro=[15 25 50];
24  %===========================
25  % PONTOS FONTE TARGET
26  %---------------------------
27  R_t=360;
28  N_t=56; % pontos fonte no target
29  opc_t=1;
30  vdim_t=[];
```

```
31 pf_t=pf(centro,N_t,opc_t,R_t,vdim_t);
32
33 %==============================
34 % PONTOS FONTE INCOGNITA
35 R_0=400;
36 N_0=40;
37 opc_0=1;
38 vdim_0=[];
39 pf_0=pf(centro,N_0,opc_0,R_0,vdim_0);
40 %==============================
41 % RUIDO
42 ruido=pruido;                                % Percentagem de ruido
43 vruido=1+ruido*(2*rand(1,32)-1)/10;    % Vetor Ruido
44 %======================================
45 %======================================
46 %% Matrizes de regularizacao
47 %--------------------------------------
48 orderMFS_T=0; % Ordem regularizacao para o MFS no Target
49 orderMFS_0=0;  % Ordem regularizacao para o MFS
50 %--------------------------------------
51 switch(orderMFS_T)
52     case 0
53             % PT Matriz de regularizacao Tikhnov ordem 0 para o ...
                   MFS T
54             PT=eye(2*N_t);
55     case 1
56             % PT Matriz de regularizacao Tikhnov ordem 1 para o ...
                   MFS T
57             mt1=diag(-1*ones(1,2*N_t));
58             mt1=mt1(1:end-1,:);
59             mt2=diag(ones(1,2*N_t),1);
60             mt2=mt2(1:2*N_t-1,1:2*N_t);
61             PT=mt1+mt2;
62     case 2
63             % PT Matriz de regularizacao Tikhnov ordem 2 para o ...
                   MFS T
64             P3a=eye(2*N_t)+diag(-2*ones(1,2*N_t-1),1)+...
65             diag(ones(1,2*N_t-2),2);
66             PT=P3a(1:2*N_t-2,:);
67 end
68 %-------------------------------
69 switch(orderMFS_0) % P0 Matriz de regularizacao Tikhnov
70     case 0
71             % Ordem 0 para o MFS
72             P0=eye(2*N_0);
73     case 1
74             % Ordem 1 para o MFS
```

```
75              mm1=diag(-1*ones(1,2*N_0));
76              mm1=mm1(1:end-1,:);
77              mm2=diag(ones(1,2*N_0),1);
78              mm2=mm2(1:2*N_0-1,1:2*N_0);
79              P0=mm1+mm2;
80      case 2
81              % Ordem 2 para o MFS
82              P2a=eye(2*N_0)+diag(-2*ones(1,2*N_0-1),1)+...
83              diag(ones(1,2*N_0-2),2);
84              P0=P2a(1:2*N_0-2,:);
85  end
86  %====================================
87  %% DATA FICTICIA NO TARGET
88  %====================================
89  % PONTOS COLOC NA SUPERFICIE TARGET \Gamma  pc_g
90  %----------------------------------------
91  [x_g,y_g]=meshgrid(5:10:25,5:10:45); % grid 15 pontos sobre ...
       plano XY
92  z_t=sup_t(x_g,y_g,superficie_Target); [f_mg, c_mg]=size(z_t);
93  % pontos colocacao paralelos a cara xzA, com x de zero a inf
94  pc_gt_xzA=[x_g(1,:)' y_g(1,:)' z_t(1,:)'];
95  % pontos colocacao paralelos a cara xzB, com x de zero a inf
96  pc_gt_xzB=[x_g(end,:)' y_g(end,:)' z_t(end,:)'];
97  % pontos colocacao paralelos a cara yzA, com y de zero a inf
98  pc_gt_yzA=[x_g(2:end-1,1) y_g(2:end-1,1) z_t(2:end-1,1)];
99  % pontos colocacao no meio paralelos a yz, com y de zero a inf
100 pc_gt_yzAB=[x_g(2:end-1,2) y_g(2:end-1,2) z_t(2:end-1,2)];
101 % pontos colocacao paralelos a cara yzB, com y de zero a inf
102 pc_gt_yzB=[x_g(2:end-1,end) y_g(2:end-1,end) z_t(2:end-1,end)];
103
104 %{
105 % GRAFICA DA FRONTEIRA /Gamma
106 mesh(x_g,y_g,z_t);
107 axis([0 30 0 50 0 100])
108 xlabel('x'); ylabel('y'); zlabel('z');
109 %}
110 %-------------------------------
111 % PONTOS COLOC NA BASE S
112 %-------------------------------
113 pc_s_xzA=pc_gt_xzA; pc_s_xzA(:,end)=zeros(size(x_g,2),1);
114 pc_s_xzB=pc_gt_xzB; pc_s_xzB(:,end)=pc_s_xzA(:,end);
115
116 pc_s_yzA=pc_gt_yzA; pc_s_yzA(:,end)=zeros(size(x_g,1)-2,1);
117 pc_s_yzB=pc_gt_yzB; pc_s_yzB(:,end)=pc_s_yzA(:,end);
118 pc_s_yzAB=pc_gt_yzAB; pc_s_yzAB(:,end)=(pc_s_yzA(:,end));
119
120 %-------------------------------
```

```
% PONTOS COLOC NAS FACES LATERAIS
%---------------------------------------
[xzA_0,xzA_1,xzA_2,xzB_0,xzB_1,xzB_2,yzA_0,yzA_1,yzB_0,yzB_1]=pc;

% PONTOS COLOCACAO GRAFICA TARGET
%{
plot3(xzA_0(:,1),xzA_0(:,2),xzA_0(:,3),'k.',xzA_1(:,1),...
xzA_1(:,2),xzA_1(:,3),...% cara xz
    'b.',xzA_2(:,1),xzA_2(:,2),xzA_2(:,3),'k.',...
    xzB_0(:,1),xzB_0(:,2),xzB_0(:,3),...
    'k.',xzB_1(:,1),xzB_1(:,2),xzB_1(:,3),...
    'b.',xzB_2(:,1),xzB_2(:,2),xzB_2(:,3),'k.',...
    yzA_0(:,1),yzA_0(:,2),yzA_0(:,3),...
    'k.',yzA_1(:,1),yzA_1(:,2),yzA_1(:,3),'k.',... % cara yz
    yzB_0(:,1),yzB_0(:,2),yzB_0(:,3),...
    'k.',yzB_1(:,1),yzB_1(:,2),yzB_1(:,3),'k.',...
    'MarkerSize',15);
%}
% PONTOS FONTE GRAFICA TARGET
% plot3(pf_t(:,1),pf_t(:,2),pf_t(:,3),'.','Color','r',...
'MarkerSize',15);
% grid on; axis([0 30 0 50 0 100]);
% xlabel('x'); ylabel('y'); zlabel('z');
%-------------------------------------------------
% PONTOS LIMITE PARA CARAS XZ E PONTOS COLOC POR CARA(SOLO ...
    CONSIDERANDO ELECTRODOS)
% CARA A
it_xzA1_o2=find(xzA_1(:,3)>pc_gt_xzA(2,3));
nt_xzA1_o2=it_xzA1_o2(end);
pc_t_xzA1_o2=xzA_1(it_xzA1_o2,:); % PTOS COLOC PARA /Omega_2

pc_t_xzA1_o1=xzA_1(nt_xzA1_o2+1:end,:); % PTOS COLOC PARA /Omega_1
% CARA B
it_xzB1_o2=find(xzB_1(:,3)>pc_gt_xzB(2,3));
nt_xzB1_o2=it_xzB1_o2(end);
pc_t_xzB1_o2=xzB_1(it_xzB1_o2,:); % PTOS COLOC PARA /Omega_2

pc_t_xzB1_o1=xzB_1(nt_xzB1_o2+1:end,:); % PTOS COLOC PARA /Omega_1
%-------------------------------------------------
%=================================================
% MATRIZ DO SISTEMA PARA O TARGET
%=================================================
% \Gamma cond contin y salto
Ap_gt=zeros(f_mg*c_mg,N_t);          % submatriz para cond ...
    continuidade em /Gamma
bAp_gt=zeros(f_mg*c_mg,1);              % termo b para continuid
Adp_gt=Ap_gt;                        % submatriz para cond salto ...
```

```
    deriv normal em /Gamma
bAdp_gt=bAp_gt;                          % termo b para salto
% \Omega_1
Ap_xzA_o1t=zeros(16-nt_xzA1_o2,N_t);     % submatriz cara xzA ...
    para \Omega_1
zeros_Ap_xzA_o1t=Ap_xzA_o1t;
Ap_xzB_o1t=zeros(16-nt_xzB1_o2,N_t);     % submatriz cara xzB ...
    para \Omega_1
zeros_Ap_xzB_o1t=Ap_xzB_o1t;
Ap_st=Ap_gt;                             % submatriz para base ...
    S \Omega_1
bAp_st=bAp_gt;                             % termo b para base S
zeros_Ap_st=Ap_gt;                % submtriz zeros para cond na ...
    base S
% \Omega_2
Ap_xzA_o2t=zeros(nt_xzA1_o2,N_t);        % submatriz cara xzA ...
    para \Omega_2
zeros_Ap_xzA_o2t=Ap_xzA_o2t;       % submatriz zeros
Ap_xzB_o2t=zeros(nt_xzB1_o2,N_t);        % submatriz cara xzB ...
    para \Omega_2
zeros_Ap_xzB_o2t=Ap_xzB_o2t;       % submatriz zeros
% termo b para condicoes sobre eletrodos
bA=xzA_1(:,4);
bB=xzB_1(:,4);
%-------------------------------------------------------------------------
% ESTRUTURA DA MATRIZ
% Ax=b
% A= |      Ap_gt             -Ap_gt      |
%    | \gamma_1*Adp_gt   -\gamma_2*Adp_gt | ...
    b=[bAp_gt;bAdp_gt;bAp_st;bA;bB]
%    |       Ap_st         zeros_Ap_st     |
%    | zeros_Ap_xzA_o2t       Ap_xzA_o2t   |
%    |     Ap_xzA_o1t    zeros_Ap_xzA_o1t |
%    |  zeros_Ap_xzB_o2t      Ap_xzB_o2t   |
%    |     Ap_xzB_o1t    zeros_Ap_xzB_o1t |
%-------------------------------------------------

for j=1:N_t
    % Pontos em \Gamma e base S
    for i=1:c_mg
        % Continuidade
        Ap_gt(i,j)=phi(pc_gt_xzA(i,:),pf_t(j,:)); % xzA
        Ap_gt(i+3,j)=phi(pc_gt_xzB(i,:),pf_t(j,:)); % xzB
        Ap_gt(i+6,j)=phi(pc_gt_yzA(i,:),pf_t(j,:)); % yzA
        Ap_gt(i+9,j)=phi(pc_gt_yzAB(i,:),pf_t(j,:)); % yzAB
        Ap_gt(i+12,j)=phi(pc_gt_yzB(i,:),pf_t(j,:)); % yzB
        % Salto
```

```
205         Adp_gt(i,j)=dphi(pc_gt_xzA(i,:),pf_t(j,:),[0,0,1]); % xzA
206         Adp_gt(i+3,j)=dphi(pc_gt_xzB(i,:),pf_t(j,:),[0,0,1]); % xzB
207         Adp_gt(i+6,j)=dphi(pc_gt_yzA(i,:),pf_t(j,:),[0,0,1]); % yzA
208         Adp_gt(i+9,j)=dphi(pc_gt_yzAB(i,:),pf_t(j,:),[0,0,1]); % ...
                yzAB
209         Adp_gt(i+12,j)=dphi(pc_gt_yzB(i,:),pf_t(j,:),[0,0,1]); % yzB
210         % Base S
211         Ap_st(i,j)=phi(pc_s_xzA(i,:),pf_t(j,:)); % xzA
212         Ap_st(i+3,j)=phi(pc_s_xzB(i,:),pf_t(j,:)); % xzB
213         Ap_st(i+6,j)=phi(pc_s_yzA(i,:),pf_t(j,:)); % yzA
214         Ap_st(i+9,j)=phi(pc_s_yzAB(i,:),pf_t(j,:)); % yzAB
215         Ap_st(i+12,j)=phi(pc_s_yzB(i,:),pf_t(j,:)); % yzB
216     end
217     % Pontos em xzA o2
218     for i=1:nt_xzA1_o2
219     Ap_xzA_o2t(i,j)=phi(pc_t_xzA1_o2(i,1:3),pf_t(j,:));
220     end
221     % Pontos em xzA o1
222     for i=1:(16-nt_xzA1_o2)
223     Ap_xzA_o1t(i,j)=phi(pc_t_xzA1_o1(i,1:3),pf_t(j,:));
224     end
225     % Pontos em xzB o2
226     for i=1:nt_xzB1_o2
227     Ap_xzB_o2t(i,j)=phi(pc_t_xzB1_o2(i,1:3),pf_t(j,:));
228     end
229     % Pontos em xzB o1
230     for i=1:(16-nt_xzB1_o2)
231     Ap_xzB_o1t(i,j)=phi(pc_t_xzB1_o1(i,1:3),pf_t(j,:));
232     end
233 end
234 A_t=[Ap_gt,-Ap_gt;...
235     gamma1*Adp_gt,-gamma2*Adp_gt;...
236     Ap_st,zeros_Ap_st;...
237     zeros_Ap_xzA_o2t,Ap_xzA_o2t;...
238     Ap_xzA_o1t,zeros_Ap_xzA_o1t;...
239     zeros_Ap_xzB_o2t,Ap_xzB_o2t;...
240     Ap_xzB_o1t,zeros_Ap_xzB_o1t];
241 b_t=[bAp_gt;bAdp_gt;bAp_st;bA;bB];
242
243 lamb_t=lambMFS(A_t,b_t,PT,rangoMFS_T);
244 sol_t= (A_t'*A_t+lamb_t*(PT')*PT)\((A_t')*b_t);
245 %=========================================
246 % GERACAO DOS DADOS FICTICIOS
247 %=========================================
248     % Pontos em xzA o2
249 for i=1:nt_xzA1_o2
250     pc_t_xzA1_o2(i,5)=gamma2*dumfs(sol_t(N_t+1:end),...
```

```
251     pc_t_xzA1_o2(i,1:3),[0,-1,0],'t');
252 end
253     % Pontos em xzA o1
254 for i=1:(16-nt_xzA1_o2)
255     pc_t_xzA1_o1(i,5)=gamma1*dumfs(sol_t(1:N_t),...
256     pc_t_xzA1_o1(i,1:3),[0,-1,0],'t');
257 end
258     % Pontos em xzB o2
259 for i=1:nt_xzB1_o2
260     pc_t_xzB1_o2(i,5)=gamma2*dumfs(sol_t(N_t+1:end),...
261     pc_t_xzB1_o2(i,1:3),[0,1,0],'t');
262 end
263     % Pontos em xzB o1
264 for i=1:(16-nt_xzB1_o2)
265     pc_t_xzB1_o1(i,5)=gamma1*dumfs(sol_t(1:N_t),...
266     pc_t_xzB1_o1(i,1:3),[0,1,0],'t');
267 end
268 dados_C_xzA=[pc_t_xzA1_o2(:,5);...
269 pc_t_xzA1_o1(:,5)].*vruido(1:16)';
270 dados_C_xzB=[pc_t_xzB1_o2(:,5);...
271 pc_t_xzB1_o1(:,5)].*vruido(16+1:end)';
272 %================================================
273 %% SOLUCAO DO PROBLEMA INVERSO
274 %================================================
275 % PONTOS DA FRONTEIRA INICIAL
276 %------------------------------------------------
277 z_0=sup_0(x_g,y_g,superficie_Inicial); %f_mg linhas c_mg colunas
278 % pontos colocacao paralelos a cara xzA, com x de zero a inf
279 p_g0_xzA=[x_g(1,:)' y_g(1,:)' z_0(1,:)'];
280 % pontos colocacao paralelos a cara xzB, com x de zero a inf
281 p_g0_xzB=[x_g(end,:)' y_g(end,:)' z_0(end,:)'];
282 % pontos colocacao paralelos a cara yzA, com y de zero a inf
283 p_g0_yzA=[x_g(2:end-1,1) y_g(2:end-1,1) z_0(2:end-1,1)];
284 % pontos colocacao no meio paralelos a yz, com y de zero a inf
285 p_g0_yzAB=[x_g(2:end-1,2) y_g(2:end-1,2) z_0(2:end-1,2)];
286 % pontos colocacao paralelos a cara yzB, com y de zero a inf
287 p_g0_yzB=[x_g(2:end-1,end) y_g(2:end-1,end) z_0(2:end-1,end)];
288 % VETOR FILA Fronteira inicial (so contem componentes z)
289 x_0=[p_g0_xzA(:,3);p_g0_xzB(:,3);p_g0_yzA(:,3);...
290 p_g0_yzAB(:,3);p_g0_yzB(:,3)];
291 %=====================================
292 % RESTRICAO DA COORDENADA Z PARA x_0
293 %-------------------------------------
294 lb=10*ones(f_mg*c_mg,1); % Limite inferior para z's em x_0
295 ub=65*ones(f_mg*c_mg,1); % Limite superior para z's em x_0
296 %=====================================
297 % SOLUCAO DO PROBLEM INVERSO
```

```
298 %====================================
299 % GENETIC ALGORITHMS
300 options=optimoptions(@ga,'PopulationSize',100,...
301 'MaxGenerations',50,'MaxStallGenerations', 100,'Display','iter');
302 %options=optimoptions(@ga,'PopulationSize',...
303 300,'MaxGenerations',50,'MaxStallGenerations', ...
       100,'Display','iter');
304
305 %optimoptions(@ga,'SelectionFcn',@selectiontournament,...
306 'FitnessScalingFcn',@fitscalingprop,'MutationFcn',...
307 @mutationadaptfeasible);
308 %thestate = rng; % obtener resultados anteriores. Primero ...
       ejecutar esta linea, despues el ga
309 %rng(thestate); % despues esta linea antes de volver a ejecutar ga
310
311 %x0 = x_0'; % Start point (row vector)
312 %options.InitialPopulationMatrix = x0;
313 [curvaSol,fval,exitflag,output]=ga(@objectf,15,[],[],[],[],...
314 lb,ub,[],options);
315 %==================================
316 % COMPARACAO DE SUPERFICIES
317 %------------------------------------
318 z_sol=z_0;
319 z_sol(1,:)=curvaSol(1:c_mg)'; % xzA
320 z_sol(end,:)=curvaSol(c_mg+1:2*c_mg)'; % xzB
321 z_sol(2:end-1,1)=curvaSol(2*c_mg+1:3*c_mg); % yzA
322 z_sol(2:end-1,2)=curvaSol(3*c_mg+1:4*c_mg); % yzAB
323 z_sol(2:end-1,3)=curvaSol(4*c_mg+1:5*c_mg); % yzAB
324 h1=surf(x_g,y_g,z_t,'EdgeColor','b');
325 hold on;
326 h2=surf(x_g,y_g,z_0,'FaceAlpha',0.4,'EdgeColor','r');
327 h3=surf(x_g,y_g,z_sol,'FaceAlpha',0.6,'EdgeColor','r',...
328 'FaceColor','g'); % 'EdgeColor','none'
329 xlabel('x'); ylabel('y'); zlabel('z');
330 axis([0 30 0 50 0 100]);
331 legend([h1, h2, h3], {'Desired surface','Initial ...
       guess','Computed surface'});
332 grid on;
333
334 fprintf('Error curva: %f\n',norm(z_t-z_sol));
335 end % FIN FUNCAO PRINCIPAL
336 %====================================
337 %% FUNCAO OBJETIVO
338 %------------------------------------
339 function yobjectf=objectf(x)
340 global gamma1 gamma2 x_g y_g rangoMFS_0 lambF pf_0 N_0 f_mg c_mg ...
       dados_C_xzA dados_C_xzB pc_s_xzA pc_s_xzB pc_s_yzA pc_s_yzB ...
```

```
      pc_s_yzAB xzA_1 xzB_1 b_t P0
%====================================
% PONTOS COLOCACAO
%====================================
% PONTOS COLOC NA SUPERFICIE \Gamma
%------------------------------------
% pontos colocacao paralelos a cara xzA, com x de zero a inf
pc_g0_xzA=[x_g(1,:)' y_g(1,:)' x(1:c_mg)'];
% pontos colocacao paralelos a cara xzB, com x de zero a inf
pc_g0_xzB=[x_g(end,:)' y_g(end,:)' x(c_mg+1:2*c_mg)'];
% pontos colocacao paralelos a cara yzA, com y de zero a inf
pc_g0_yzA=[x_g(2:end-1,1) y_g(2:end-1,1) x(2*c_mg+1:3*c_mg)'];
% pontos colocacao no meio paralelos a yz, com y de zero a inf
pc_g0_yzAB=[x_g(2:end-1,2) y_g(2:end-1,2) x(3*c_mg+1:4*c_mg)'];
% pontos colocacao paralelos a cara yzB, com y de zero a inf
pc_g0_yzB=[x_g(2:end-1,end) y_g(2:end-1,end) x(4*c_mg+1:end)'];
%------------------------------------------
% PONTOS COLOC NA BASE S pc_s_.z. (x,y),(A,B,AB)
%------------------------------------------
% PONTOS COLOC NAS FACES LATERAIS .z.1 (x,y),(A,B)
%------------------------------------------
% PONTOS LIMITE PARA CARAS XZ E PONTOS COLOC POR CARA(SOLO ...
    CONSIDERANDO ELECTRODOS)
%------------------------------------------
% CARA A
i0_xzA1_o2=find(xzA_1(:,3)>pc_g0_xzA(2,3));
n0_xzA1_o2=i0_xzA1_o2(end);
pc_0_xzA1_o2=xzA_1(i0_xzA1_o2,:); % PTOS COLOC PARA /Omega_2

pc_0_xzA1_o1=xzA_1(n0_xzA1_o2+1:end,:); % PTOS COLOC PARA /Omega_1
% CARA B
i0_xzB1_o2=find(xzB_1(:,3)>pc_g0_xzB(2,3));
n0_xzB1_o2=i0_xzB1_o2(end);
pc_0_xzB1_o2=xzB_1(i0_xzB1_o2,:); % PTOS COLOC PARA /Omega_2

pc_0_xzB1_o1=xzB_1(n0_xzB1_o2+1:end,:); % PTOS COLOC PARA /Omega_1
%=======================================
% MATRIZ DO SISTEMA
%=======================================
% \Gamma cond contin y salto
Ap_g0=zeros(f_mg*c_mg,N_0);          % submatriz para cond ...
    continuidade em /Gamma
Adp_g0=Ap_g0;                       % submatriz para cond salto ...
    deriv normal em /Gamma
% \Omega_1
Ap_xzA_o10=zeros(16-n0_xzA1_o2,N_0);      % submatriz cara xzA ...
    para \Omega_1
```

```
383 zeros_Ap_xzA_o10=Ap_xzA_o10;
384 Ap_xzB_o10=zeros(16-n0_xzB1_o2,N_0);      % submatriz cara xzB ...
       para \Omega_1
385 zeros_Ap_xzB_o10=Ap_xzB_o10;
386 Ap_s0=Ap_g0;                              % submatriz para base ...
       S \Omega_1
387 zeros_Ap_s0=Ap_g0;                 % submtriz zeros para cond na ...
       base S
388 % \Omega_2
389 Ap_xzA_o20=zeros(n0_xzA1_o2,N_0);         % submatriz cara xzA ...
       para \Omega_2
390 zeros_Ap_xzA_o20=Ap_xzA_o20;       % submatriz zeros
391 Ap_xzB_o20=zeros(n0_xzB1_o2,N_0);         % submatriz cara xzB ...
       para \Omega_2
392 zeros_Ap_xzB_o20=Ap_xzB_o20;       % submatriz zeros
393 %-----------------------------------------
394 % ESTRUTURA DA MATRIZ
395 % Ax=b
396 % A= |      Ap_g0             -Ap_g0      |
397 %    | \gamma_1*Adp_g0   -\gamma_2*Adp_g0 |, ...
       b=b_t=[bAp_gt;bAdp_gt;bAp_st;bA;bB]
398 %    |       Ap_s0        zeros_Ap_s0     |
399 %    | zeros_Ap_xzA_o20       Ap_xzA_o20  |
400 %    |     Ap_xzA_o10    zeros_Ap_xzA_o10 |
401 %    |  zeros_Ap_xzB_o20      Ap_xzB_o20  |
402 %    |     Ap_xzB_o10    zeros_Ap_xzB_o10 |
403 %-----------------------------------------
404 for j=1:N_0
405     % Pontos em \Gamma e base S
406     for i=1:c_mg
407         % Continuidade
408         Ap_g0(i,j)=phi(pc_g0_xzA(i,:),pf_0(j,:)); % xzA
409         Ap_g0(i+3,j)=phi(pc_g0_xzB(i,:),pf_0(j,:)); % xzB
410         Ap_g0(i+6,j)=phi(pc_g0_yzA(i,:),pf_0(j,:)); % yzA
411         Ap_g0(i+9,j)=phi(pc_g0_yzAB(i,:),pf_0(j,:)); % yzAB
412         Ap_g0(i+12,j)=phi(pc_g0_yzB(i,:),pf_0(j,:)); % yzB
413         % Salto
414         Adp_g0(i,j)=dphi(pc_g0_xzA(i,:),pf_0(j,:),[0,0,1]); % xzA
415         Adp_g0(i+3,j)=dphi(pc_g0_xzB(i,:),pf_0(j,:),[0,0,1]); % xzB
416         Adp_g0(i+6,j)=dphi(pc_g0_yzA(i,:),pf_0(j,:),[0,0,1]); % yzA
417         Adp_g0(i+9,j)=dphi(pc_g0_yzAB(i,:),pf_0(j,:),[0,0,1]); % ...
               yzAB
418         Adp_g0(i+12,j)=dphi(pc_g0_yzB(i,:),pf_0(j,:),[0,0,1]); % yzB
419         % Base S
420         Ap_s0(i,j)=phi(pc_s_xzA(i,:),pf_0(j,:)); % xzA
421         Ap_s0(i+3,j)=phi(pc_s_xzB(i,:),pf_0(j,:)); % xzB
422         Ap_s0(i+6,j)=phi(pc_s_yzA(i,:),pf_0(j,:)); % yzA
```

```
423         Ap_s0(i+9,j)=phi(pc_s_yzAB(i,:),pf_0(j,:)); % yzAB
424         Ap_s0(i+12,j)=phi(pc_s_yzB(i,:),pf_0(j,:)); % yzB
425     end
426     % Pontos em xzA o2
427     for i=1:n0_xzA1_o2
428     Ap_xzA_o20(i,j)=phi(pc_0_xzA1_o2(i,1:3),pf_0(j,:));
429     end
430     % Pontos em xzA o1
431     for i=1:(16-n0_xzA1_o2)
432     Ap_xzA_o10(i,j)=phi(pc_0_xzA1_o1(i,1:3),pf_0(j,:));
433     end
434     % Pontos em xzB o2
435     for i=1:n0_xzB1_o2
436     Ap_xzB_o20(i,j)=phi(pc_0_xzB1_o2(i,1:3),pf_0(j,:));
437     end
438     % Pontos em xzB o1
439     for i=1:(16-n0_xzB1_o2)
440     Ap_xzB_o10(i,j)=phi(pc_0_xzB1_o1(i,1:3),pf_0(j,:));
441     end
442 end
443 A_0=[Ap_g0,-Ap_g0;...
444     gamma1*Adp_g0,-gamma2*Adp_g0;...
445     Ap_s0,zeros_Ap_s0;...
446     zeros_Ap_xzA_o20,Ap_xzA_o20;...
447     Ap_xzA_o10,zeros_Ap_xzA_o10;...
448     zeros_Ap_xzB_o20,Ap_xzB_o20;...
449     Ap_xzB_o10,zeros_Ap_xzB_o10];
450 lamb_0=lambMFS(A_0,b_t,P0,rangoMFS_0);
451 sol_0= (A_0'*A_0+lamb_0*(P0')*P0)\((A_0')*b_t);
452 %=======================================
453 % FUNCAO OBJETIVO
454 %=======================================
455 s_xzA_o2=0; s_xzA_o1=0; s_xzB_o2=0; s_xzB_o1=0;
456 % Pontos em xzA o2
457 for i=1:n0_xzA1_o2
458     s_xzA_o2=s_xzA_o2+(gamma2*dumfs(sol_0(N_0+1:end),...
459     pc_0_xzA1_o2(i,1:3),[0,-1,0],'x')-dados_C_xzA(i))^2;
460 end
461 % Pontos em xzA o1
462 for i=1:(16-n0_xzA1_o2)
463     s_xzA_o1=s_xzA_o1+(gamma1*dumfs(sol_0(1:N_0),...
464     pc_0_xzA1_o1(i,1:3),[0,-1,0],'x')-dados_C_xzA(n0_xzA1_o2+i))^2;
465 end
466 % Pontos em xzB o2
467 for i=1:n0_xzB1_o2
468     s_xzB_o2=s_xzB_o2+(gamma2*dumfs(sol_0(N_0+1:end),...
469     pc_0_xzB1_o2(i,1:3),[0,1,0],'x')-dados_C_xzB(i))^2;
```

```
end
% Pontos em xzB o1
for i=1:(16-n0_xzB1_o2)
    s_xzB_o1=s_xzB_o1+(gamma1*dumfs(sol_0(1:N_0),...
    pc_0_xzB1_o1(i,1:3),[0,1,0],'x')-dados_C_xzB(n0_xzB1_o2+i))^2;
end
%=======================================
% TERMOS DE REGULARIZACAO
%=======================================
% MATRIZES DO GRID para alturas z
%x_r=x_g'; y_r=y_g';
z_r=zeros(c_mg,f_mg);
z_r(:,1)=x(1:c_mg); z_r(:,end)=x(c_mg+1:2*c_mg); % xzA, xzB
z_r(1,2:end-1)=x(2*c_mg+1:3*c_mg)'; % yzA
z_r(2,2:end-1)=x(3*c_mg+1:4*c_mg)'; % yzAB
z_r(3,2:end-1)=x(4*c_mg+1:end)';    % yzB
%-----------------------------------------------
% GRID QUADRADO
Gqs=0;
for i=1:c_mg % linhas
    for j=1:f_mg-1
        Gqs=Gqs+(z_r(i,j)-z_r(i,j+1))^2;
    end
end
for j=1:f_mg  % colunas
    for i=1:c_mg-1
        Gqs=Gqs+(z_r(i,j)-z_r(i+1,j))^2;
    end
end
%-----------------------------------------------
% GRID CRUZ
Gcs=0; Gcs=(z_r(2,1)-z_r(1,2))^2+(z_r(3,4)-z_r(2,5))^2+...
    (z_r(2,1)-z_r(3,2))^2+(z_r(1,4)-z_r(2,5))^2;
for i=1:c_mg % direcao (1,1)
    for j=1:c_mg-1
        Gcs=Gcs+(z_r(j,i+3-j)-z_r(j+1,i+3-j-1))^2;
    end
end
for i=1:c_mg % direcao (1,-1)
    for j=1:c_mg-1
        Gcs=Gcs+(z_r(j,j+i-1)-z_r(j+1,j+1+i-1))^2;
    end
end
%=======================================
yobjectf=s_xzA_o2+s_xzA_o1+s_xzB_o2+s_xzB_o1+lambF*sqrt(Gqs);

end
```

```
%==========================================
%% L-CURVE CRITERIO MFS
%==========================================
function [ylambMFS]=lambMFS(AA,bb,PP,rango)
lambdas=rango;
npo=size(lambdas);
    for k=1:npo(1)
        solaux=(AA'*AA + lambdas(k)*(PP')*PP)\((AA')*bb);
        mvalores(k,:)=[norm(AA*solaux-bb),norm(solaux)];
    end
    dista=(mvalores(:,1).^2+mvalores(:,2).^2).^(1/2);
    indmin=find(dista==min(dista),npo(1));
    ylambMFS=lambdas(indmin(1));
   %plot(mvalores(:,1),mvalores(:,2),'b-*',...
   mvalores(indmin(1),1),mvalores(indmin(1),2),'rs');
end


%==========================================
%% FUNCOES ENVOLVIDAS
%==========================================
% SUPERFICIE TARGET
%------------------------------------------------
function T=sup_t(xxx,yyy,superficieT_op) % xxx,yyy:matrizes do mesh
    [ff,cc]=size(xxx); T=ones(ff,cc);
    if superficieT_op==1
    T=T*60;
    else
    T=5*sin(xxx)+5*cos(yyy)+60;
    end
end
%==========================================
% SUPERFICIE INICIAL
%------------------------------------------------
function G0=sup_0(xxx,yyy,superficie0_op) % xxx,yyy:matrizes do mesh
    [ff,cc]=size(xxx); G0=ones(ff,cc);
    if superficie0_op==1
    G0=G0*15; % Constante
    else
    G0=5*sin(xxx)+5*cos(yyy)+10;
    end
end
%==========================================
% FUNCAO /phi = 1/r
%------------------------------------------------
function yphi=phi(va,vpf) % va,vpf:vetores fila 3d
    yphi=1/(norm(va-vpf));
```

```
564 end
565 %=========================================
566 % FUNCAO d/phi/dn =
567 %-----------------------------------------
568 function ydphi=dphi(va,vpf,vn) % va,vpf, vn:vetores fila 3d
569     ydphi=-((va-vpf)*vn')/((norm(va-vpf))^3);
570 end
571 %=========================================
572 % SOLUCAO NUMERICA u_MFS
573 %-----------------------------------------
574 function yumfs=umfs(coefs,xxx,op) % coefs: a para u1, b para u2
575 global pf_t N_t pf_0 N_0               % xxx: ponto a avaliar
576 yumfs=0;
577 switch op
578     case {'t','T'}
579         for inn=1:N_t
580             yumfs=yumfs+coefs(inn)*phi(xxx,pf_t(inn,:));
581         end
582     case {'x','X'}
583         for inn=1:N_0
584             yumfs=yumfs+coefs(inn)*phi(xxx,pf_0(inn,:));
585         end
586 end
587 end
588 %-----------------------------------------
589 % SOLUCAO NUMERICA du_MFS/dn
590 %-----------------------------------
591 function ydumfs=dumfs(coefs,xxx,vn,op)
592 global pf_t N_t pf_0 N_0
593 ydumfs=0;
594 switch op
595     case {'t','T'}
596         for ind=1:N_t
597             ydumfs=ydumfs+coefs(ind)*dphi(xxx,pf_t(ind,:),vn);
598         end
599     case {'x','X'}
600         for ind=1:N_0
601             ydumfs=ydumfs+coefs(ind)*dphi(xxx,pf_0(ind,:),vn);
602         end
603 end
604 end
605 %=========================================
606 %% FUNCAO GERACAO PONTOS DE COLOCACAO
607 %-----------------------------------------
608 function ...
        [xzA_0,xzA_1,xzA_2,xzB_0,xzB_1,xzB_2,yzA_0,yzA_1,yzB_0,yzB_1]=pc
609 % cada matriz saida contem coordenadas x,y,z,c para ...A,B_i ...
```

```
      ,(i=0,2 em xz, i=0,1 em yz)
% x,y,z,p,c para ...A,B_1 em xz  (llenado de arriba a abajo)
    % pontos ao longo do eixo z
    zpc=linspace(0,100,10); zpe=linspace(0,100,18);
    xz_x=linspace(0,30,5); % coordenada x para pontos sobre ...
       planos xzA xzB
    yz_y=linspace(0,50,4); % coordenada y para pontos sobre ...
       planos yzA yzB
    % leituras nas caras que contem eletrodos
    vpeA=(1:16)'; vceA=zeros(16,1); vpeB=(4:19)'; vceB=zeros(16,1);
    %-------------------------------------------------
    % planos xzA xzB
        % plano xzA
            % corrente
    xzA_0=zeros(8,4);
    xzA_0(:,1)=repmat(xz_x(2),8,1);xzA_0(:,3)=zpc(end-1:-1:2)';
    xzA_2=xzA_0; xzA_2(:,1)=repmat(xz_x(4),8,1);
            % eletrodos
    xzA_1=zeros(16,5); xzA_1(:,4)=vpeA; xzA_1(:,5)=vceA; % leituras
    xzA_1(:,1)=repmat(xz_x(3),16,1); xzA_1(:,3)=zpe(end-1:-1:2)';
    %-------------------------------------------------
        % plano xzB
            % corrente
    xzB_0=xzA_0; xzB_0(:,2)=repmat(50,8,1);
    xzB_2=xzA_2; xzB_2(:,2)=xzB_0(:,2);
            % eletrodos
    xzB_1=xzA_1; xzB_1(:,4)=vpeB; xzB_1(:,5)=vceB;
    xzB_1(:,2)=repmat(50,16,1);
    %-------------------------------------------------
    %-------------------------------------------------
    % planos yzA yzB
        % plano yzA
    yzA_0=zeros(8,4); yzA_0(:,2)=repmat(yz_y(2),8,1);
    yzA_0(:,3)=zpc(end-1:-1:2)';
    yzA_1=yzA_0; yzA_1(:,2)=repmat(yz_y(3),8,1);
        % plano yzB
    yzB_0=yzA_0; yzB_0(:,1)=repmat(30,8,1);
    yzB_1=yzA_1; yzB_1(:,1)=repmat(30,8,1);
end
%=====================================================


%=====================================================
% FUNCAO GERA PONTOS FONTE
%------------------ exemplos--------------------------
%{
```

```
654 pf_t=pf(centro,16,1,1);
655 figure(1)
656 plot3(pf_t(:,1),pf_t(:,2),pf_t(:,3),'.','Color','k',...
657 'MarkerSize',15);
658 hold on
659 [xx, yy, zz]=sphere;
660 surf(xx+centro(1),yy+centro(2),zz+centro(3),'FaceAlpha',0.8);
661 %surf(xx+centro(1),yy+centro(2),zz+centro(3),'FaceAlpha',0.2,...
662 'FaceColor', [1 1 1]);
663 %}
664
665 %{
666 pf_t=pf(centro,4,2,60);
667 figure(2)
668 plot3(pf_t(:,1),pf_t(:,2),pf_t(:,3),'.','Color','k',...
669 'MarkerSize',15);
670 xlabel('x'); ylabel('y'); zlabel('z');
671 axis equal
672 grid on
673 %box on
674 %ax = gca;
675 %ax.BoxStyle = 'full';
676 %}
677
678 %{
679 pf_t=pf(centro,4,3,1,[30,50,100]);
680 figure(2)
681 plot3(pf_t(:,1),pf_t(:,2),pf_t(:,3),'.','Color','k',...
682 'MarkerSize',15);
683 xlabel('x'); ylabel('y'); zlabel('z');
684 axis equal
685 grid on
686 %}
687 %-----------------------------------
688 function ypf=pf(centro,np,opc,r,vdim)
689 %opc: 1 esfera, np=mult 8 pontos fonte
690 %opc: 2 cubo, np= pontos fonte por lado, 2*r lado do cubo
691 %opc: 3 paralelepipedo, 2*np pontos sobre xz yz e np sobre xy ...
       (no usa r)
692 k=1;
693     switch opc
694         case 1 % esfera
695             ypf=zeros(np,3);
696             np_xy=np/2; np_cc=np/8;
697             for i=1:np_xy
698                 for j=1:np_cc
699                     ypf(k,:)=[r*sin(j*pi/(np_cc+1))*...
```

```
700                     cos(i*2*pi/(np_xy))...
701                         r*sin(j*pi/(np_cc+1))*sin(i*2*pi/(np_xy))...
702                         r*cos(j*pi/(np_cc+1))];
703                     k=k+1;
704                 end
705             end
706             ypf=ypf+centro;
707         case 2 % cubo
708             ypf=zeros(6*np,3);
709             % pontos sobre face paralela a z
710             maux_z=repmat(centro,np,1); maux_y=maux_z;
711             zp=linspace(centro(3)+r,centro(3)-r,np+2);
712             maux_z(:,end)=zp(2:end-1)';
713                 % matriz traslacao em y
714             mt_y=zeros(np,3); mt_y(:,2)=ones(np,1);
715                 % matriz traslacao em x
716             mt_x=zeros(np,3); mt_x(:,1)=ones(np,1);
717             % pontos sobre face paralela a y
718             yp=linspace(centro(2)+r,centro(2)-r,np+2);
719             maux_y(:,2)=yp(2:end-1)';
720                 % matriz traslacao em z
721             mt_z=zeros(np,3); mt_z(:,end)=ones(np,1);
722             for i=1:2
723                 % pontos trasl eixo y
724                 ypf((1+(i-1)*np):(i*np),:)=maux_z+(2*i-3)*r*mt_y;
725                 % pontos trasl eixo x
726                 ypf((1+(i+1)*np):((i+2)*np),:)=...
727                 maux_z+(2*i-3)*r*mt_x;
728                 % pontos trasl eixo z
729                 ypf((1+(i+3)*np):((i+4)*np),:)=...
730                 maux_y+(2*i-3)*r*mt_z;
731             end
732         case 3 % paralelepipedo
733             % lados sobre x,y,z:vdim(i=1,2,3)
734             ypf=zeros(10*np,3);
735             % pontos sobre face paralela a z
736             maux_z=repmat(centro,2*np,1);
737             zp=linspace(centro(3)+vdim(3)/2,centro(3)-...
738             vdim(3)/2,2*np+2);
739             maux_z(:,end)=zp(2:end-1)';
740                 % matriz traslacao em y
741             mt_y=zeros(2*np,3); mt_y(:,2)=ones(2*np,1);
742                 % matriz traslacao em x
743             mt_x=zeros(2*np,3); mt_x(:,1)=ones(2*np,1);
744             % pontos sobre face paralela a y
745             maux_y=repmat(centro,np,1);
746             yp=linspace(centro(2)+vdim(2)/2,centro(2)-...
```

```
747             vdim(2)/2,np+2);
748             maux_y(:,2)=yp(2:end-1)';
749                 % matriz traslacao em z
750             mt_z=zeros(np,3); mt_z(:,end)=ones(np,1);
751             for i=1:2
752                 % pontos trasl eixo y
753                 ypf((1+(2*i-2)*np):((2*i)*np),:)=maux_z+...
754                 (2*i-3)*(vdim(2)/2)*mt_y;
755                 % pontos trasl eixo x
756                 ypf((1+(2*i+2)*np):((2*(i+2))*np),:)=maux_z+...
757                 (2*i-3)*(vdim(1)/2)*mt_x;
758                 % pontos trasl eixo z
759                 ypf((1+(i+7)*np):((i+8)*np),:)=maux_y+...
760                 (2*i-3)*(vdim(3)/2)*mt_z;
761             end
762
763     end
764 end
765 %================================================
```