Universidade Federal de Minas Gerais

Escola de Engenharia

Programa de Pós-Graduação em Engenharia Elétrica

Algoritmos Eficientes em Métodos sem Malha

Alexandre Ramos Fonseca

Tese submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais, como requisito para obtenção do título de Doutor em Engenharia Elétrica.

Orientador: Prof. Renato Cardoso Mesquita

Co-orientador: Prof. Elson José da Silva

Belo Horizonte, março de 2011

Resumo

Diferentemente dos métodos baseados em malhas, os métodos sem malha são caracterizados pelo uso de um conjunto de nós espalhados pelo domínio do problema, ao invés de uma malha ou grade. Neste trabalho é discutido o MLPG (*Meshless Local Petrov Galerkin method*) para a resolução de problemas eletromagnéticos. O objetivo aqui é, além de entender o método, buscar formas de aumentar sua eficiência a fim de torná-lo competitivo com métodos tradicionais como o método de elementos finitos.

Como não existe uma malha para dar conectividade aos nós, um desafio para o método é determinar com eficiência quais nós pertencem à vizinhança de um determinado nó. Para resolver esse problema é usada uma árvore de busca denominada *kd-tree*.

Outra dificuldade dos métodos sem malha está na imposição das condições de contorno de Dirichlet quando as funções de forma não apresentam a propriedade do delta de Kronecker, o que é o caso para funções de forma construídas a partir do método de mínimos quadrados móveis (MLS). O MLS é um método eficiente, bastante conhecido na literatura e um dos mais utilizados em métodos sem malha, mas necessita de técnicas especiais para impor as condições de contorno de Dirichlet, como o método de penalidades ou multiplicadores de Lagrange. Além do MLS, investigamos funções de forma baseadas no método de interpolação de pontos (PIM) utilizando funções de base radial (RPIM) e associadas com termos polinomiais (RPIMp). O RPIMp possui a propriedade do delta de Kronecker, sendo uma alternativa ao MLS, dispensando técnicas especiais para impor as condições de contorno. Entretanto, o RPIMp apresenta um custo computacional maior que o MLS quando um número maior de nós é utilizado. Neste trabalho propomos um método misto que combina o RPIMp para nós da fronteira e o MLS para o interior do domínio. Dessa forma, geramos um método que associa os melhores atributos das duas funções de forma: a imposição direta das condições de contorno e

rápido processamento. Resultados obtidos mostram que o método misto possui boa precisão e custo computacional intermediário aos das funções de forma utilizadas.

Para tornar o método misto ainda mais atrativo, estudamos uma função de forma mais simples para o interior do domínio. As funções de Shepard (caso particular do MLS com consistência C^0) são funções de forma extremamente simples, de fácil implementação e custo computacional muito baixo. Nesse caso, mesmo construindo as funções de forma com poucos nós vizinhos, o método misto sempre possui custo computacional inferior quando se compara à utilização de apenas funções RPIMp.

Durante as implementações do MLPG percebeu-se que as contribuições dos nós para o sistema matricial global são independentes. Cada nó contribui com uma linha do sistema, não interferindo na contribuição dos demais nós. Aproveitando a chegada ao mercado de processadores com múltiplos núcleos e aproveitando essa característica do método, é proposta uma forma de paralelizar o processo de montagem do sistema linear que é a fase do método de maior custo computacional. Resultados indicam um ganho de desempenho nessa parte do processamento de até $3,78$ vezes para processadores com 4 núcleos.

Abstract

Unlike mesh based methods, meshfree methods are distinguished by the use of a set of scattered nodes over the domain instead of a mesh or grid. In this work the use of MLPG (*Meshless Local Petrov Galerkin method*) to solve electromagnetic problems is discussed.

The main objective is to improve the method's computational efficiency in order to make it competitive with traditional methods like the finite element method.

Due to the absence of a mesh there is no connectivity among nodes and it is a challenge for the method to determine efficiently which nodes belong to the neighborhood of a particular node. To solve this problem a kd-tree is used.

Another issue of meshfree methods is how to impose Dirichlet boundary conditions when the shape functions do not possess the Kronecker delta property, which is the case for shape functions constructed from the moving least-squares method (MLS). The MLS is an efficient method, well known in the literature and one of the most used in meshless methods but requires special techniques to impose the Dirichlet boundary conditions, such as the method of penalties or Lagrange multipliers. In addition to the MLS, we investigate shape functions based on the point interpolation method (PIM) using radial basis functions (RPIM) with polynomial terms (RPIMp). The RPIMp has the property of Kronecker delta, which makes it an alternative to the MLS dispensing special techniques for imposing boundary conditions. However, RPIMp presents a higher computational cost than the MLS when a large number of nodes is used. In this paper we propose a mixed method that combines RPIMp to border nodes and the MLS into the domain. Thus, we generate a method that combines the best attributes of two functions: a direct imposition of boundary conditions and fast processing. Results show that

the mixed method has good accuracy and intermediate computational cost between the shape functions used.

To make the mixed method even more attractive, we investigated a simpler shape function to use inside the domain. Shepard functions (particular case of MLS with C^0 consistency) are extremely simple functions, easy to implement and have low computational cost. In this case, even constructing the shape functions with few neighboring nodes, the mixed method always present lower computational cost when compared to using only RPIMp functions.

During MLPG implementations we realize that the nodes contributions for the global matrix system are independent. Each node contributes with a line on the matrix system, not interfering with the contribution of the other nodes. Taking advantage of multi-core processors, now easy to find in the market, we propose a way to parallelize the linear system assembling process which is the stage of the method with higher computational cost. Results indicate a performance gain up to 3.78 times in this part of the processing for 4-cores processors.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

α	Constante de valor elevado (método das penalidades)
α_q	Raio do subdomínio de quadratura normalizado por h
α_s	Raio de influência da função de forma normalizado por h
$\bar{t}$	Condição de contorno de Neumann imposta em Γ_t
ϵ	Permissividade elétrica
$\frac{\partial J(\mathbf{a}(\mathbf{x}))}{\partial a_i}$	Derivada parcial de $J(\mathbf{a}(\mathbf{x}))$ em relação ao i-ésimo componente de $\mathbf{a}(\mathbf{x})$
Γ_I	Fronteira entre materiais (interface)
Γ_{qi}	Fronteira do subdomínio do nó i
$\mathbf{\Phi}$	Vetor de funções de forma
$\mathbf{\Phi}_{,k}$	Derivada parcial de $\mathbf{\Phi}$ em relação a x_k
μ	Permeabilidade magnética
Ω_q^i	Subdomínio do nó i
Ω_q	Subdomínio
ϕ_i	Função de forma do nó i
ρ	Densidade de carga
ρ_s	Densidade de carga superficial
$\mathbf{A}$	Matrix $(m \times m)$ usada pelo método de mínimos quadrados móveis
$\mathbf{a}(\mathbf{x})$	Vetor de coeficientes

$\mathbf{A}^{-1}$	Matriz inversa de **A**
$\mathbf{A}^{-1}_{,k}$	Derivada parcial de $\mathbf{A}^{-1}$ em relação a x_k
$\mathbf{A}_{,k}$	Derivada parcial de **A** em relação a x_k
B	Matrix $(m \times n)$ usada pelo método de mínimos quadrados móveis
$\mathbf{B}_{,k}$	Derivada parcial de **B** em relação a x_k
F	Vetor $n \times 1$ do sistema linear **KU**=**F**
K	Matriz $n \times n$ do sistema linear **KU**=**F**
P	Matriz $(n \times m)$ contendo os vetores **p** avaliados para todos os pontos conhecidos
$\mathbf{p}(\mathbf{x})$	Vetor contendo os monômios de uma base polinomial completa
R	Vetor $(n \times 1)$ com as avaliações das funções RBF para um dado ponto
$\mathbf{R}_0$	Matriz $(n \times n)$ contendo as avaliações dos pontos $\mathbf{x}_i$ nas funções R_i
V	Vetor $(n \times 1)$ com valores da função V para cada ponto $\mathbf{x}_i$
W	Matriz diagonal $(n \times n)$ contendo as avaliações da função de peso w para todos os pontos conhecidos
A_z	Componente na direção z do potencial vetor magnético
d	Distância euclidiana entre o ponto de avaliação e o centro da RBF normalizado pelo raio de suporte
dim	Dimensão do problema
F_i	Elemento (i) do vetor **F**
g	Condição de contorno de Dirichlet imposta em Γ_u
h	Distância média entre os nós
$J(\mathbf{a}(\mathbf{x}))$	Funcional para determinar os coeficientes $\mathbf{a}(\mathbf{x})$
K_{ij}	Elemento (i, j) da matriz **K**
m	Número de termos de $\mathbf{p}^t(\mathbf{x})$

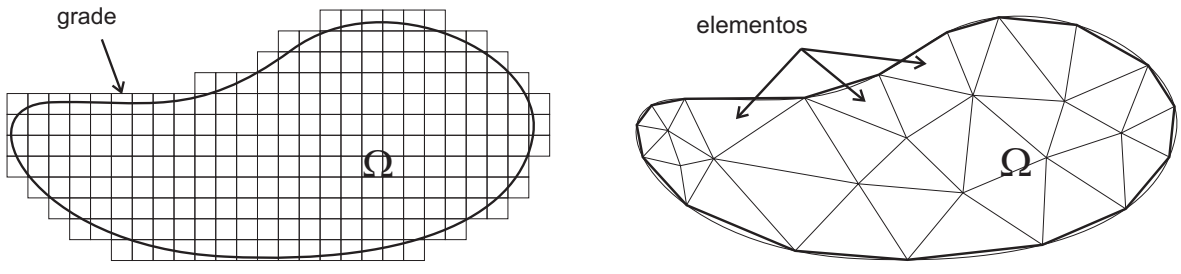
$p_j(\mathbf{x})$	j-ésimo termo da base polinominal $p(\mathbf{x})$
r	Raio suporte para uma função RBF
$R(.)$	Função de base radial
$R(.)_{,k}$	Derivada parcial de $R(.)$ em relação à x_k – $\frac{\partial R(.)}{\partial x_k}$
R_i	Raio da caixa de integração
$R_i(\mathbf{x})$	Avaliação do ponto **x** na RBF com centro em $\mathbf{x}_i$
R_q	Metade do lado da caixa de busca (domínio de suporte)
R_ϕ	Raio do domínio de influência dos nós
t	Ordem máxima dos monômios em $\mathbf{p}(\mathbf{x})$
V	Tensão elétrica
V^h	Aproximação para a função V
V_i	Valor conhecido da função V em $\mathbf{x}_i$
W_i	Função de teste para o nó i
x_k	k-ésimo componente de **x**

Capítulo 1

Introdução

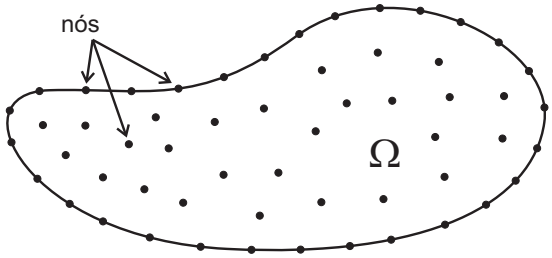
Em Engenharia, o projeto de sistemas avançados requer o uso de ferramentas computacionais (*computer-aided design - CAD*), nas quais técnicas de simulação computacional são freqüentemente usadas para modelar e investigar fenômenos físicos. Em vários sistemas, a etapa de simulação consiste na solução de equações diferenciais parciais que governam o fenômeno. Neste trabalho, será investigado o fenômeno eletromagnético, regido pelas equações de Maxwell. Computacionalmente, essas equações são resolvidas usando-se métodos numéricos, como o método de elementos finitos (FEM) (Hughes, 2000) e o método de diferenças finitas (FDM) (Taflove, 2000). Nesses métodos, o domínio espacial é discretizado segundo uma malha, cuja composição consiste em um conjunto de nós conectados entre si segundo um padrão pré-definido.

A malha é necessária para definir uma relação de dependência ou conectividade entre os nós, razão pela qual ela é a base para a formulação dos métodos numéricos convencionais. No FDM, usa-se uma malha estruturada, também conhecida como grade (*grid*), como ilustrado na Figura 1.1a. Em uma malha estruturada é simples determinar quais são os nós vizinhos de um dado nó i, já que os nós são ordenados pela grade de forma sistemática. Entretanto, a grade não se adapta bem a problemas com geometrias curvas ou até mesmo retas e planos inclinados. No FEM diz-se que a malha é composta de elementos e, apesar de cada elemento poder variar assumindo formas geométricas diversas, o mais comum é utilizar triângulos para problemas em duas dimensões, como ilustra a Figura 1.1b, ou tetraedros para definir os elementos em



(a) malha estruturada- FDM

(b) malha não estruturada- FEM

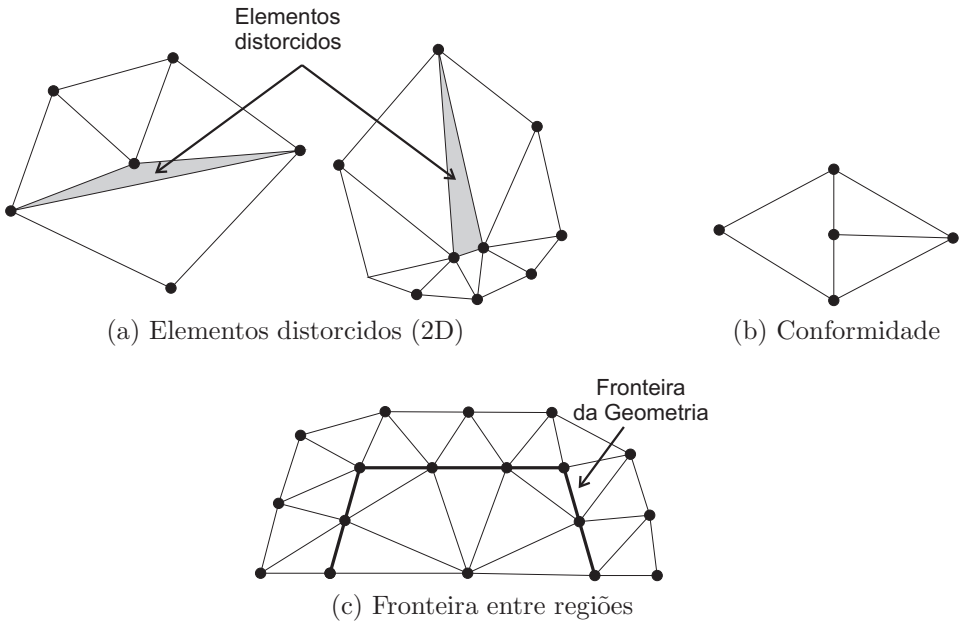


(c) nós - *Mfree*

Figura 1.1: Comparação entre discretização do domínio Ω. Em (a), o domínio é discretizado segundo uma grade regular. Nota-se, nesse caso, que a grade não consegue descrever a geometria com boa exatidão na fronteira do domínio. Para o método de elementos finitos, a malha é não estruturada, como ilustra (b). Esse tipo de malha consegue descrever com maior precisão geometrias complexas, mas pode apresentar problemas como elementos degenerados (ver Figura 1.2). Método sem malha não discretizam o domínio segundo uma malha. Nós são espalhados sobre o domínio e sua fronteira, como visto em (c).

problemas de três dimensões. Dessa forma, a malha do FEM é capaz de se ajustar bem a geometrias complexas com inclinações e discretizar curvas em um conjunto de segmentos ou planos.

O método de elementos finitos é robusto e tem sido extensivamente usado na solução de problemas de Engenharia. Entretanto, o método possui algumas limitações. Uma das principais dificuldades do FEM é gerar uma malha que represente bem o problema e seja de boa qualidade do ponto de vista numérico. A qualidade da forma dos elementos afeta a precisão dos resultados numéricos. Uma malha com boa distribuição de nós e elementos bem formados contribui na geração de sistemas bem condicionados, minimizando erros numéricos e singularidades (Nunes et al., 2007). Em duas dimensões, uma malha de boa qualidade apresenta elementos triangulares próximos ao equilátero. Malhas que apresentam elementos distorcidos, como os mostrados na Figura 1.2a geram sistemas lineares mal condicionados e resultados com erro elevado. Esses elementos se caracterizam por possuir área (ou volume em 3D) próxima de zero. Nesse caso, o fator de qualidade dado pela razão entre base e altura tende a zero ou



(a) Elementos distorcidos (2D)

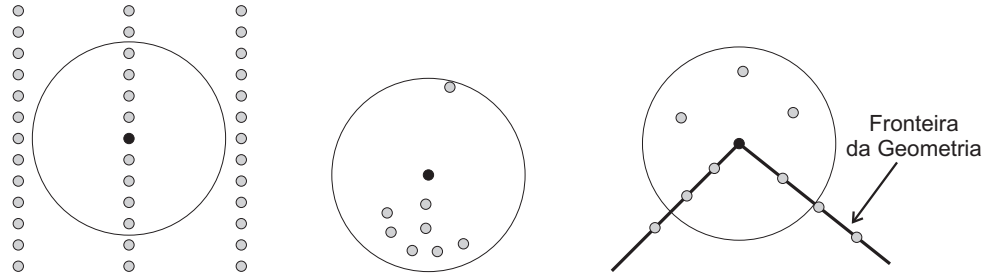
(b) Conformidade

(c) Fronteira entre regiões

Figura 1.2: Principais problemas para geração de malhas para o método de elementos finitos. (a) mostra elementos distorcidos em duas dimensões. (b) mostra uma malha de triângulos não conforme, o elemento da esquerda é definido por quatro segmentos ao invés de somente três. (c) quando o problema envolve mais de uma região, a geometria da interface tem que ser respeitada. Isso pode dificultar a geração de uma malha de boa qualidade (Shewchuk, 1997; Nunes et al., 2007).

para infinito. Elementos distorcidos não representam grande problema para malhas bidimensionais, já que podem ser removidos por meio de refinamento da malha. Entretanto, elementos distorcidos em três dimensões ainda representam grande desafio para a geração automática de malhas com boa qualidade (Idelsohn and Oñate, 2006; Shewchuk, 1997; Nunes et al., 2007). Além de possuir elementos de forma adequada, a malha deve ser conforme e os contornos devem ser respeitados (ver Figura 1.2). Gerar uma malha que satisfaça esses requisitos em 2 dimensões não é um problema, mas, em 3 dimensões os geradores de malhas existentes tipicamente necessitam de muito tempo de máquina e, em muitos casos, é necessário a supervisão humana para se conseguir bons resultados (Idelsohn and Oñate, 2006). Para problemas onde o domínio do problema muda com o tempo, como acontece em estruturas que envolvem movimento, esse problema é agravado pois uma nova malha deve ser gerada a cada configuração do problema.

As dificuldades associadas ao método de elementos finitos são facilmente resolvidas quando se usa um método sem malha. Nos métodos sem malha (*MFree*) um sistema de equações algébricas é estabelecido para todo o domínio do problema sem o uso de uma malha. Métodos



(a) variação direcional de h

(b) não uniformidade de h

(c) Fronteira da geometria

Figura 1.3: Principais problemas para distribuição de nós em métodos sem malha

MFree usam um conjunto de nós espalhados sobre o domínio do problema, bem como um conjunto de nós espalhados sobre suas fronteiras para representar o domínio do problema e suas fronteiras. Esses nós não formam uma malha, já que nenhuma informação sobre conectividade entre eles é necessária, pelo menos para a interpolação de variáveis de campo. A Figura 1.1 compara a maneira que o FEM, o FDM e métodos sem malha discretizam o domínio.

Como os métodos sem malha não precisam de uma conectividade direta entre os nós, o processo de geração da malha é dispensado sendo uma opção bastante atrativa, especialmente para problemas com geometrias tridimensionais complexas ou que envolvem movimento ou deformação da geometria. Entretanto, essa ausência de conectividade introduz algumas dificuldades aos métodos sem malha. Para determinar a conectividade entre os nós, o que normalmente se faz é determinar quais nós estão presentes na vizinhança do nó, como mostra a Figura 1.3, onde a vizinhança é mostrada como uma região circular. A figura mostra também as principais dificuldades para uma boa distribuição de nós em métodos sem malha (Idelsohn and Oñate, 2006); sendo h a distância entre os nós. Algumas vezes a variação de h é maior em uma direção específica (variação direcional de h), como ilustrado na Figura 1.3a ou é não uniforme, formando regiões com muita concentração de nós e regiões com deficiência de nós, como ilustrado na Figura 1.3b. No primeiro caso, uma direção influencia mais a aproximação do que a outra. No exemplo da Figura 1.3a, somente os nós acima e abaixo têm conectividade com o nó em destaque. No segundo caso (Figura 1.3b), apenas a região abaixo do nó em destaque é bem representada enquanto a região acima do nó conta com apenas um nó. O problema das fronteiras presentes nos métodos com malhas é comum aos métodos sem malha, como visto na Figura 1.3c. É necessário forçar a distribuição de pontos sobre as fronteiras. A Tabela 1.1 mostra uma comparação entre as dificuldades entre métodos com malha e sem malha.

Tabela 1.1: Comparação das dificuldades entre malhas e distribuição de nós

	Métodos com malha	Métodos sem malha
Conformidade	difícil	fácil
Elementos degenerados	difícil	fácil
Fronteiras da geometria	difícil	difícil
Variação direcional de h	fácil	difícil
Não uniformidade de h	fácil	difícil

Para tornar métodos sem malha uma opção frente aos métodos tradicionais, é necessário obter implementações eficientes desses métodos. Entretanto, implementações dos métodos sem malha ainda possuem custo computacional elevado quando comparadas aos métodos com malha. Para prover a conectividade entre os nós precisa-se determinar quais nós pertencem à vizinhança de um determinado nó de interesse; para construir as funções de forma necessárias ao cálculo de integrações da forma fraca é necessário fazer inversões de matrizes locais e outros cálculos matriciais. Essa dificuldades criam obstáculos em se gerar implementações eficientes para métodos sem malha, tornando necessário o emprego de técnicas especiais para torná-los mais eficientes. Esse é um dos principais pontos abordados neste trabalho.

Dentre os vários métodos sem malha é possível destacar: *Smooth Particle Hydrodynamics* (SPH) (Lucy, 1977; Gingold and Monaghan, 1977), *Element Free* Galerkin (EFG) (Belytschko et al., 1994), *Meshless Local* Petrov-Galerkin (MLPG) (Atluri and Zhu, 1998), *Point Interpolation Method* (PIM) (Liu, 2002), Radial PIM (RPIM)(Liu, 2002), entre muitos outros.

O SPH foi um dos primeiros métodos sem malhas desenvolvidos, baseando-se na formulação integral de uma função. A função delta de Dirac é aproximada por uma função núcleo (*kernel function*) W e a integral é calculada sobre um conjunto de partículas (Liu and Liu, 2003). Inicialmente, o SPH foi formulado para resolver problemas de astrofísica (Lucy, 1977), mas recentemente vem sendo aplicado principalmente em problemas de dinâmica de fluidos (Liu and Liu, 2003). O SPH foi adaptado para resolver problemas eletromagnéticos no domínio do tempo, gerando um novo método conhecido como Smoothed Particle Electromagnetics (SPEM) (Ala, 2006; Mendes, 2010). No SPEM, as partículas do SPH são divididas em dois grupos: partículas elétricas que armazenam informações sobre o campo elétrico e partículas magnéticas para armazenar as informações do campo magnético. A interpolação do SPH é então aplicada às equações de Maxwell no domínio do tempo (Ala, 2006; Mendes, 2010).

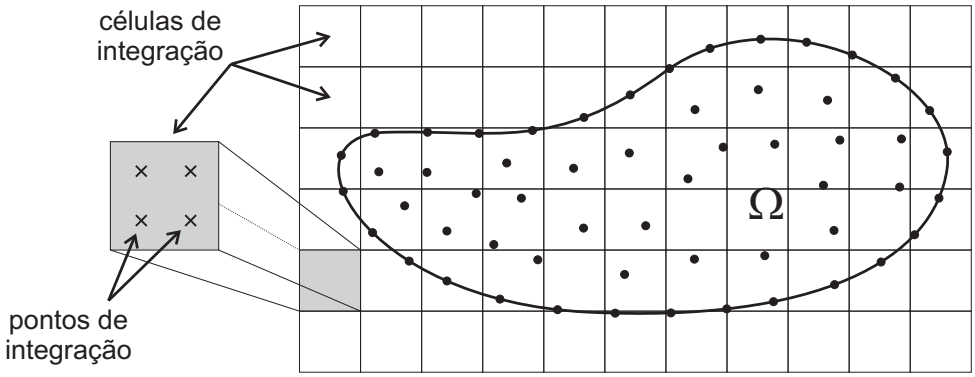


Figura 1.4: Malha de integração para o EFG

O SPH possui algumas vantagens computacionais como simplicidade de implementação, além de se adequar bem a problemas com deformação. Por outro lado, como o método utiliza a função delta de Dirac, apresenta alguns problemas: condições de contorno não são satisfeitas exatamente e o processo de interpolação não é completamente consistente ou completo. Nos últimos anos, muitas técnicas têm sido desenvolvidas para melhorar sua performance e eliminar seus problemas numéricos (Liu and Liu, 2003).

O EFG foi desenvolvido por Belytschko et al. (1994) e se baseia no método de elementos difusos (DEM - Diffuse Elements Method), primeiro método sem malha baseado na técnica de Galerkin. Inicialmente, ambos foram desenvolvidos para resolver problemas mecânicos mas atualmente são aplicados também a problemas eletromagnéticos (Cingoski et al., 1998; Parreira et al., 2006a,b). O EFG é considerado um método sem malha no sentido de não existir uma conectividade direta entre os nós, como acontece no método de elementos finitos através das arestas da malha. No EFG, a conectividade entre os nós é dada pela região de influência exercida por cada nó. Entretanto, há a necessidade de se criar uma malha grade para a integração da forma fraca do problema (Atluri and Shen, 2002). Essa grade (Figura 1.4) não depende da distribuição dos nós sobre o domínio, podendo ser regular. Dessa forma, sua construção é bem mais simples que a malha do FEM.

No EFG as funções de forma e de teste têm que pertencer ao mesmo espaço de funções devido ao uso do método de Galerkin, sendo o método de mínimos quadrados móveis (MLS) comumente usado para gerar tais funções (Liu, 2002).

O MLPG foi originalmente proposto por Atluri and Zhu (1998). Neste método se usa o conceito de forma fraca local, dispensando grades de integração, o que torna o método bastante flexível, considerado por seus autores como um método "verdadeiramente sem malha" (Atluri and Shen, 2002). Por isso, o MLPG é o método escolhido para ser explorado neste trabalho.

Como será visto no Capítulo 3, o MLPG usa uma formulação fraca local. Nessa técnica, cada nó define seu próprio subdomínio que é independente dos demais. Cada subdomínio pode ter qualquer tamanho ou forma geométrica e o conjunto de subdomínios deve cobrir completamente o domínio global. Os subdomínios dispensa a geração de uma grade de integração como a utilizada no EFG, pois a integração é feita localmente em cada subdomínio. Além disso, o MLPG utiliza o método de Petrov-Galerkin que permite que as funções de teste e de forma pertençam a espaços de funções diferentes.

Permutando as possíveis combinações entre funções de teste e de forma, obtêm-se variações do MLPG como (Atluri and Shen, 2002):

- MLPG 1: Usa-se o mesmo tipo de função como função de teste para os subdomínios dos nós e como função de peso para a construção das funções de forma utilizadas pelo método de mínimos quadrados móveis (MLS).
- MLPG 2: A função de teste é a função delta de Dirac o que resulta em um método de colocação. Nesse caso as integrais da forma fraca local se anulam e a forma forte da equação diferencial parcial é discretizada.
- MLPG 3: Escolhe-se como função de teste o resíduo da equação diferencial a ser resolvida. Ou seja, resolve-se um problema de mínimos quadrados.
- MLPG 4: A função de teste é a solução fundamental modificada da equação diferencial. Essa variação do método é conhecida também como *Local Boundary Integration Equation* (LBIE)
- MLPG 5: A função de teste é a função de Heaviside, que é constante e igual a um sobre todo o subdomínio, e igual a zero fora dele.
- MLPG 6: As funções de forma e de teste são idênticas o que torna esse método um caso especial do Bubnov-Galerkin method (Fries and Matthies, 2004). Se esferas são

utilizadas como subdomínios, o método passa a ser conhecido também como método das esferas finitas (De and Bathe, 2000).

Em (Atluri and Shen, 2002) é feita uma análise comparativa do desempenho computacional e da precisão dessas variações e o MLPG 5 é o método que consegue unir boa precisão e o menor custo computacional, uma vez que o uso da função de Heaviside como função de teste elimina a necessidade de integração no interior dos subdomínios. Por isso, essa será a opção utilizada nesse trabalho e daqui para frente a sigla MLPG se refere ao MLPG 5.

1.1 Objetivos

Eficiência computacional e confiabilidade são claramente os requisitos mais importantes para o sucesso de um método numérico, seja ele com malha ou sem malha. As ideias básicas empregadas nos métodos sem malha são simples e bem entendidas. Entretanto, desenvolver um método sem malha eficiente é muito difícil (De and Bathe, 2001). A eficiência depende da escolha apropriada do método de interpolação a ser utilizado para a geração das funções de forma, dos procedimentos utilizados na integração numérica, da determinação da relação de interdependência entre nós, das técnicas utilizadas para impor as condições de contorno, entre outros (De and Bathe, 2001; Idelsohn and Oñate, 2006).

Uma revisão da literatura sobre métodos sem malha revela que a tendência atual está voltada para a aplicação dessas novas técnicas nas diversas áreas da engenharia. Mas está claro que, para aplicações gerais, nenhuma dessas técnicas é tão eficiente computacionalmente quanto os métodos tradicionais baseados em malhas como o método de elementos finitos (De and Bathe, 2001).

Neste trabalho o problema de cálculo de campo eletromagnético será tratado por meio do método MLPG. A principal contribuição do trabalho de doutorado será a proposta de soluções para o problema de alto custo computacional do método em questão, visto que implementações atuais de métodos sem malha são computacionalmente mais caras que implementações de métodos com malha.

O principal objetivo desse trabalho é identificar os principais fatores limitadores de desempenho do método MLPG e propor soluções para torná-lo mais eficiente, do ponto de vista computacional.

1.2 Metodologia

Para a implementação do método MLPG, utiliza-se o paradigma de programação orientada a objetos por suas facilidades em estruturar classes de forma hierárquica, facilitando o desenvolvimento pela reutilização de código através de mecanismos como agrupamento e polimorfismo e por mecanismos de programação genérica através de *templates* e padrões de projeto como *Singleton*, *Factories*, *Observers*, entre outros (Gamma et al., 1994). Entre as possíveis linguagens de programação que suportam a orientação a objetos, escolheu-se o *C++* por questões de eficiência.

Para resolver detalhes de implementações, utilizaram-se conceitos de geometria computacional (de Berg et al., 2000). Como exemplo, uma das principais dificuldades dos métodos sem malhas é determinar a relação de interdependência entre nós, visto que um determinado nó precisa *conhecer* os nós *ao seu redor* (nós vizinhos), os quais exercem influência na integração do subdomínio do nó em questão. Encontrar essa conectividade é um problema que não é tratado em sua real dimensão em muitos artigos (Idelsohn and Oñate, 2006). Achar os vizinhos de um nó usando um método força bruta é computacionalmente caro, podendo tornar a solução, para problemas com muitos nós, inviável. Para contornar esse problema, buscou-se aplicar estruturas de dados mais eficientes, como árvores de busca, para reduzir a ordem de complexidade do método.

Na primeira fase desse trabalho, utilizam-se apenas funções de forma construídas pelo método de mínimos quadrados móveis (MLS) por ser tradicionalmente utilizada em métodos sem malha (Liu, 2002; Atluri and Shen, 2002; Parreira et al., 2006b). Entretanto, as funções de forma MLS não possuem a propriedade do delta de Kronecker o que torna necessário o uso de técnicas especiais para a imposição das condições de contorno de Dirichlet. Aqui, utilizamos o método da penalidade (Liu, 2002; Atluri and Shen, 2002) que nem sempre gera bons resultados (podem ocorrer oscilações nas fronteiras). Como alternativa ao método da penalidade,

passamos a testar funções de forma calculadas utilizando o método de interpolação de pontos com funções RBF e polinômios (RPIMp) (Liu, 2002; Viana et al., 2004). Com esse método, conseguimos funções de forma que possuem a propriedade do delta de Kronecker, o que torna dispensável o método de penalidade para forçar as condições de contorno, e gerar resultados com boa precisão e sem oscilações nas bordas. A desvantagem do uso do RPIMp é sua complexidade computacional elevada. Diante desse cenário, imaginamos a possibilidade de se unir as qualidades das duas funções de forma. O resultado disso é a proposta de um novo método com formulação mista que utiliza as funções de forma RPIMp sobre os nós da fronteira e MLS sobre os nós internos ao domínio (Fonseca et al., 2008c). O método misto apresentou resultados com boa precisão e um tempo de execução intermediário quando comparado à utilização de uma única função de forma.

O método misto apresentado em (Fonseca et al., 2008c) se mostrou capaz de mesclar duas funções de forma diferentes, possibilitando o uso de uma função que possui a propriedade do delta de Kronecker sobre a fronteira de Dirichlet, não necessitando de métodos especiais para impor as condições de contorno. Entretanto, o método se mostrou vantajoso apenas quando o número de nós vizinhos utilizados para a construção das funções de forma é maior que 16 mas, na maioria dos casos, apenas 9 vizinhos geram boas soluções. Para tornar o método misto atrativo, era necessário utilizar uma função de forma para os nós internos que fosse muito rápida de se calcular. Nesse novo contexto, experimentamos funções de forma extremamente simples de serem calculadas, as funções de Shepard.

As funções de Shepard foram propostas por Shepard em 1968 (Fries and Matthies, 2004). Essas funções podem ser interpretadas como um subcaso do método de mínimos quadrados móveis com ordem zero de consistência, com a vantagem de possuírem baixo custo computacional e serem de fácil implementação. Entretanto, a baixa ordem de consistência pode fazer com que as funções de Shepard falhem ao tentar solucionar problemas mais complexos (Fries and Matthies, 2004).

Utilizando apenas as funções de Shepard com o método de penalidades, não obtivemos bons resultados. As oscilações no contorno apresentadas quando usamos MLS são intensificadas quando passamos a usar as funções de Shepard. Por isso, usadas isoladamente, as funções de Shepard não são uma boa opção para o MLPG, mas quando combinadas com funções RPIMp produzem bons resultados e com um custo computacional muito menor. Além do uso de uma função de forma mais simples para o interior do domínio, percebemos que o RPIMp era

necessário apenas para nós sobre a fronteira de Dirichlet, diminuindo um preprocessamento de classificação dos nós. O resultado dessas melhorias foi publicado em (Fonseca et al., 2010).

Além das questões apresentadas acima, apostou-se também na paralelização do método, já que nos últimos anos, ocorreu a disseminação de processadores com vários núcleos de processamento. Um dos principais melhoramentos dos processadores em um futuro próximo consistirá no aumento do número de núcleos e na quantidade de memória cache (Bischof et al., 2007). Portanto, implementações paralelizadas terão um papel fundamental na obtenção de códigos de alto desempenho. Seguindo essa tendência, uma paralelização do método, baseada em uma arquitetura de múltiplos processadores compartilhando a mesma memória, é proposta e implementada (Fonseca et al., 2009).

1.3 Estrutura do texto

No Capítulo 2 são apresentados métodos de interpolações que geram as funções de forma utilizadas pelo MLPG. São discutidos o método dos mínimos quadrados móveis (MLS), as funções de Shepard, o método de interpolação de pontos com funções de base radial (RPIM) e o RPIM associado com termos polinomiais (RPIMp).

No Capítulo 3 apresentamos o método MLPG construído a partir da forma forte para problemas estáticos do eletromagnetismo.

O Capítulo 4 detalha as escolhas e estruturas utilizadas nas implementações: para o problema de se encontrar nós vizinhos é utilizada uma árvore de busca (*kd-tree*); para o problema de imposição das condições de contorno é proposta uma maneira de combinar as funções de forma apresentadas no Capítulo 2 associando as melhores características de cada uma delas: velocidade no processamento no interior do domínio e facilidade de imposição de condições de contorno na fronteira; é apresentada ainda uma forma simples para a paralelização do processo de montagem do sistema linear.

Por fim, no Capítulo 5 são apresentadas as conclusões e propostas para trabalhos futuros.

Capítulo 2

Funções de forma

Neste capítulo, discutem-se métodos de interpolação dado um conjunto de pontos nos quais o valor da função é conhecido. A partir desses métodos são construídas as chamadas funções de forma que são utilizadas em métodos sem malha.

Dado um conjunto de pontos $\mathbf{x}_i$, sendo $i = 1, 2, ..., n$, nos quais se conhece os valores da função V, deseja-se encontrar uma função V^h que se aproxime da função original. Esse problema é ilustrado na Figura 2.1. Existem várias maneiras de se encontrar essa função aproximada (Liu, 2002). Nessa seção serão vistos os métodos dos mínimos quadrados móveis (MLS) e o de interpolação de pontos (*point interpolation method* – PIM) usando funções de base radial (RPIM) e funções de base radial associadas a termos polinomiais (RPIMp).

De forma geral, procura-se representar a aproximação da função V através de um conjunto de funções de forma ϕ_i tais que

$$V^h(\mathbf{x}) = \sum_{i=1}^{n} \phi_i(\mathbf{x}) V(\mathbf{x}_i). \tag{2.1}$$

Usualmente, as funções de forma possuem suporte compacto, influenciando apenas uma pequena porção do domínio.

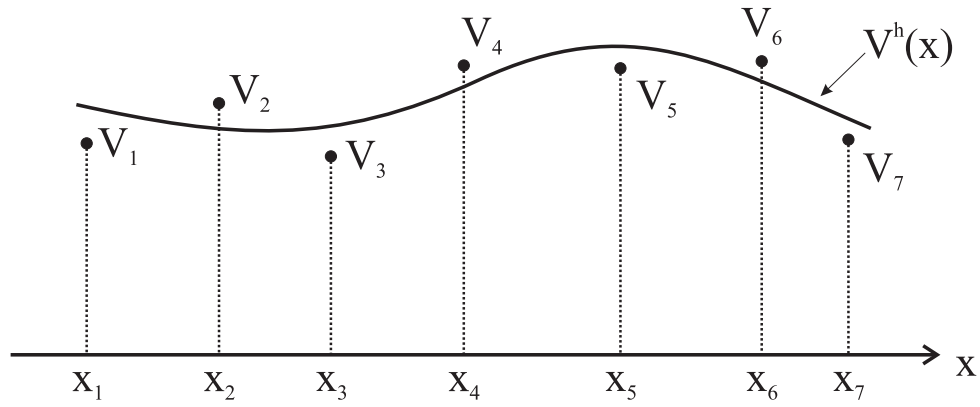


Figura 2.1: Pontos conhecidos de uma função V e sua aproximação $V^h(\mathbf{x})$. No exemplo da figura, V_1, V_2, ..., V_7 são os valores da função V, que se deseja aproximar, nos pontos $\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_7$. $V^h(\mathbf{x})$ é uma função aproximada para V construída a partir dos valores conhecidos.

As funções de forma devem atender a critérios de consistência e reprodutibilidade. O grau de consistência pode ser entendido como o polinômio de mais alta ordem que pode ser representado de forma exata. Uma equação diferencial na forma $\mathcal{L}V = f$ é dita consistente de ordem p se $||\mathcal{L}u - \mathcal{L}^h V|| = O(h^p)$, onde h é uma medida da densidade da distribuição de nós. A consistência é um fator importante para a convergência de métodos sem malha (Belytschko et al., 1998).

Assegurar que um método sem malha com uma distribuição de nós não uniforme seja consistente é relativamente difícil. O que se faz, ao invés disso, é analisar as condições de reprodutibilidade (*reproducing conditions* ou *completeness*) (Belytschko et al., 1998). Uma aproximação $V^h(\mathbf{x})$ é completa de ordem k se qualquer polinômio de ordem menor ou igual a k pode ser representado de maneira exata. Dessa forma, se $V^h(\mathbf{x})$ é dado por

$$V^h(\mathbf{x}) = \sum_{i=1}^{n} \phi_i(\mathbf{x}) V_i, \tag{2.2}$$

onde V_i são os valores nodais, então se os valores de V_i são dados por um polinômio de ordem k, a aproximação $V^h(\mathbf{x})$ reproduzirá o polinômio de forma exata se a aproximação for completa de ordem k (Belytschko et al., 1998). Em uma dimensão, as condições de reprodutibilidade podem ser escritas como se segue: se os valores nodais são dados por um polinômio:

$$V_i = a_0 + a_1 x_i + a_2 {x_i}^2 + \ldots + a_k {x_i}^k, \tag{2.3}$$

então a condição de reprodutibilidade será satisfeita se

$$V^h(x) = \sum_{i=1}^{n} \phi_i(x) V_i = a_0 + a_1 x + a_2 x^2 + \ldots + a_k x^k. \tag{2.4}$$

Como a expressão acima deve ser satisfeita para coeficientes a_i arbitrários, obtém-se as condições abaixo:

$$\sum_{i=1}^{n} \phi_i(x) = 1 \tag{2.5}$$

$$\sum_{i=1}^{n} \phi_i(x) x_i = x \tag{2.6}$$

$$\sum_{i=1}^{n} \phi_i(x) {x_i}^2 = x^2 \tag{2.7}$$

$$\ldots$$

$$\sum_{i=1}^{n} \phi_i(x) {x_i}^k = x^k \tag{2.8}$$

Em duas dimensões, as condições de reprodutibilidade linear podem ser escritas como:

$$\sum_{i=1}^{n} \phi_i(\mathbf{x}) = 1 \tag{2.9}$$

$$\sum_{i=1}^{n} \phi_i(\mathbf{x}) x_i = x \tag{2.10}$$

$$\sum_{i=1}^{n} \phi_i(\mathbf{x}) y_i = y \tag{2.11}$$

onde $\mathbf{x} = (x, y)$.

As Equações 2.5 e 2.9 são conhecidas como condição de partição da unidade (PU) e as Equações 2.6, 2.10 e 2.11 são as condições de reprodutividade linear.

Além de satisfazer a partição da unidade, é desejável, mas não necessário, que as funções de forma satisfaçam outras condições como possuir a propriedade da função delta de Kronecker, dada por

$$\phi_i(\mathbf{x}_j) = \begin{cases} 1 & \text{se} \quad i = j \\ 0 & \text{se} \quad i \neq j \end{cases} \tag{2.12}$$

Como será visto, o método dos mínimos quadrados móveis não possui a propriedade do delta de Kronecker, ou seja, o valor interpolado da função em um ponto conhecido não é necessariamente igual ao valor da função nesse ponto. Em outras palavras, a função interpolada não passa pelos pontos conhecidos.

Usando o PIM construído a partir de RBF's (RPIM), as funções de forma geradas possuem a propriedade do delta de Kronecker e a função interpolada passa pelos pontos conhecidos. Apesar dessa vantagem, o RPIM não têm a capacidade de aproximar todos os tipos de funções. Isso se deve ao fato de que a construção das funções de forma utilizando apenas RBF's não fornecem uma base completa para aproximar tais funções. Para contornar esse tipo de problema, além das RBF's, acrescentam-se termos polinomiais à base do PIM (RPIMp) (Liu, 2002).

2.1 Método dos mínimos quadrados móveis

No método dos mínimos quadrados móveis (Nealen, 2004), usa-se uma aproximação do tipo

$$V^h(\mathbf{x}) = \mathbf{p}^t(\mathbf{x})\mathbf{a}(\mathbf{x}), \tag{2.13}$$

na qual $\mathbf{p}^t(\mathbf{x}) = [p_1(\mathbf{x})\ p_2(\mathbf{x})\ \ldots\ p_m(\mathbf{x})]$ é o vetor contendo os monômios de uma base polinomial completa e $\mathbf{a}(\mathbf{x}) = [a_1(\mathbf{x})\ a_2(\mathbf{x})\ \ldots\ a_m(\mathbf{x})]^t$ é um vetor de coeficientes que dependem de $\mathbf{x}$, ou seja, para cada posição do espaço tem-se um vetor de coeficientes diferente. Se t é a ordem máxima dos monômios e dim é a dimensão do problema, o número de termos m de $\mathbf{p}(\mathbf{x})$ pode ser determinado pela expressão (Liu, 2002):

$$m = \frac{(t+1)(t+2)\ldots(t+dim)}{dim!} \tag{2.14}$$

A Tabela 2.1 mostra alguns exemplos de $\mathbf{p}^t(\mathbf{x})$ variando a dimensão do problema e a ordem máxima dos monômios.

Tabela 2.1: Algumas bases polinomiais

t	dim	m	$\mathbf{p}^T(\mathbf{x})$
1	2	3	$[1\ x_1\ x_2]$
1	3	4	$[1\ x_1\ x_2\ x_3]$
2	2	6	$[1\ x_1\ x_2\ x_1^2\ x_1x_2\ x_2^2]$

Os coeficientes do vetor $\mathbf{a}(\mathbf{x})$ são determinados através da minimização do funcional

$$\begin{aligned} J(\mathbf{a}(\mathbf{x})) &= \sum_{i=1}^{n} w(\|\mathbf{x}-\mathbf{x}_i\|)\left[V^h(\mathbf{x})-V(\mathbf{x})\right]^2 \\ &= \sum_{i=1}^{n} w(\|\mathbf{x}-\mathbf{x}_i\|)\left[\mathbf{p}^t(\mathbf{x}_i)\mathbf{a}(\mathbf{x})-V_i\right]^2 \end{aligned} \tag{2.15}$$

onde $w(\|\mathbf{x}-\mathbf{x}_i\|)$ é uma função de peso para o nó i e V_i é o valor conhecido da função nesse nó. Normalmente, a função de peso para um ponto é uma função de base radial (RBF) com valor máximo no nó i e que diminui quanto maior a distância para esse nó. Existem várias possibilidades para a função w. O Apêndice A mostra várias funções RBF testadas neste trabalho.

Com a utilização da função de peso, dá-se uma importância para um nó em sua vizinhança e, para pontos distantes, esse nó exerce influência pequena ou nula para a aproximação.

Para minimizar o funcional definido na Equação 2.15, faz-se $\frac{\partial J}{\partial a} = 0$, para todo a_i, gerando o conjunto de equações:

$$\begin{aligned} \frac{\partial J}{\partial a_1} &= \sum_{i=1}^{n} 2.w(\|\mathbf{x}-\mathbf{x}_i\|).p_1(\mathbf{x}_i).[\mathbf{p}^t(\mathbf{x}_i).\mathbf{a}(\mathbf{x})-V_i] = 0 \\ \frac{\partial J}{\partial a_2} &= \sum_{i=1}^{n} 2.w(\|\mathbf{x}-\mathbf{x}_i\|).p_2(\mathbf{x}_i).[\mathbf{p}^t(\mathbf{x}_i).\mathbf{a}(\mathbf{x})-V_i] = 0 \\ &\vdots \\ \frac{\partial J}{\partial a_n} &= \sum_{i=1}^{n} 2.w(\|\mathbf{x}-\mathbf{x}_i\|).p_n(\mathbf{x}_i).[\mathbf{p}^t(\mathbf{x}_i).\mathbf{a}(\mathbf{x})-V_i] = 0 \end{aligned} \tag{2.16}$$

Rearranjando as derivadas parciais do funcional, obtém-se a equação matricial

$$\mathbf{Aa} = \mathbf{BV} \tag{2.17}$$

na qual

$$\mathbf{A}(\mathbf{x}) = \sum_{i=1}^{n} w(\|\mathbf{x} - \mathbf{x}_i\|)\mathbf{p}(\mathbf{x}_i)\mathbf{p}^t(\mathbf{x}_i), \tag{2.18}$$

$$\mathbf{B}(\mathbf{x}) = [w(\|\mathbf{x} - \mathbf{x}_1\|)\mathbf{p}(\mathbf{x}_1),\ w(\|\mathbf{x} - \mathbf{x}_2\|)\mathbf{p}(\mathbf{x}_2),\ \ldots,\ w(\|\mathbf{x} - \mathbf{x}_n\|)\mathbf{p}(\mathbf{x}_n)] \tag{2.19}$$

e

$$\mathbf{V} = [V_1,\ V_1,\ V_2,\ \ldots,\ V_n]^t \tag{2.20}$$

Note-se que a matriz $\mathbf{A}$ tem dimensão $(m \times m)$ enquanto a matriz $\mathbf{B}$ tem dimensão $(m \times n)$.

Isolando $\mathbf{a}(\mathbf{x})$ na Equação 2.17 e substituindo na Equação 2.13, obtém-se

$$V^h(\mathbf{x}) = \mathbf{p}^t(\mathbf{x})\mathbf{A}^{-1}\mathbf{B}\mathbf{V} = \Phi(\mathbf{x})\mathbf{V}, \tag{2.21}$$

onde o vetor de funções de forma é dado por

$$\Phi(\mathbf{x}) = \mathbf{p}^t(\mathbf{x})\mathbf{A}^{-1}\mathbf{B} = [\phi_1(\mathbf{x}),\ \phi_2(\mathbf{x}),\ \ldots,\ \phi_n(\mathbf{x})] \tag{2.22}$$

e ϕ_i é a função de forma, ou função de base para o nó i, definida por

$$\phi_i(\mathbf{x}) = \sum_{j=1}^{m} p_j(\mathbf{x}) \left[\mathbf{A}^{-1}\mathbf{B}\right]_{ji} = \mathbf{p}^t(\mathbf{x})[\mathbf{A}^{-1}\mathbf{B}_i], \tag{2.23}$$

na qual $p_j(\mathbf{x})$ é o j-ésimo termo da base polinominal $p(\mathbf{x})$, $\left[\mathbf{A}^{-1}\mathbf{B}\right]_{ji}$ é o elemento (j, i) da matriz $\mathbf{A}^{-1}\mathbf{B}$ e $\mathbf{B}_i$ é a coluna i da matriz $\mathbf{B}$ e pode ser obtida diretamente por

$$\mathbf{B}_i = w_i(\|\mathbf{x} - \mathbf{x}_i\|)\mathbf{p}^t(\mathbf{x}_i) \tag{2.24}$$

Definindo a matriz $\mathbf{P}$ com as linhas dadas pelos vetores $\mathbf{p}$ avaliados em todos os pontos conhecidos e a matriz $\mathbf{W}$ como uma matriz diagonal, na qual os termos da diagonal principal são os valores das funções peso $w(\|\mathbf{x} - \mathbf{x}_i\|)$, tem-se

$$\mathbf{P} = [\mathbf{p}(\mathbf{x}_1),\ \mathbf{p}(\mathbf{x}_2),\ \mathbf{p}(\mathbf{x}_3),\ \ldots,\ \mathbf{p}(\mathbf{x}_n)]^t \tag{2.25}$$

e

$$\mathbf{W} = \begin{bmatrix} w_i(\|\mathbf{x} - \mathbf{x}_1\|) & 0 & \cdots & 0 \\ 0 & w_i(\|\mathbf{x} - \mathbf{x}_2\|) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w_i(\|\mathbf{x} - \mathbf{x}_n\|) \end{bmatrix} \tag{2.26}$$

Com as matrizes **P** e **W** pode-se reescrever as matrizes **A** e **B** de forma compacta:

$$\mathbf{A} = \mathbf{P}^t\mathbf{W}\mathbf{P} = \mathbf{B}\mathbf{P} \tag{2.27}$$

$$\mathbf{B} = \mathbf{P}^t\mathbf{W} \tag{2.28}$$

Fazendo-se $\boldsymbol{\lambda} = \mathbf{p}^t(\mathbf{x})\mathbf{A}^{-1}$, tem-se

$$\boldsymbol{\lambda}_{,k} = \mathbf{p}^t_{,k}(\mathbf{x})\mathbf{A}^{-1} + \mathbf{p}^t(\mathbf{x})\mathbf{A}^{-1}_{,k} \tag{2.29}$$

na qual $\boldsymbol{\lambda}_{,k}$, $\mathbf{p}^t_{,k}$ e $\mathbf{A}^{-1}_{,k}$ são as derivadas parciais de $\boldsymbol{\lambda}$, $\mathbf{p}^t$ e $\mathbf{A}^{-1}$ em relação a x_k, que é a k-ésima dimensão do ponto $\mathbf{x}$. $\mathbf{A}^{-1}_{,k}$, pode ser calculada por

$$\mathbf{A}^{-1}_{,k} = -\mathbf{A}^{-1}\mathbf{A}_{,k}\mathbf{A}^{-1} \tag{2.30}$$

Finalmente, as derivadas parciais de $\boldsymbol{\Phi}$ podem ser calculadas como

$$\boldsymbol{\Phi}_{,k} = \boldsymbol{\lambda}_{,k}\mathbf{B} + \boldsymbol{\lambda}\mathbf{B}_{,k} \tag{2.31}$$

onde

$$\begin{aligned}\mathbf{B}_{,k} &= [w(\|\mathbf{x} - \mathbf{x}_1\|)_{,k}\mathbf{p}(\mathbf{x}_1),\ w(\|\mathbf{x} - \mathbf{x}_2\|)_{,k}\mathbf{p}(\mathbf{x}_2),\ \ldots,\ w(\|\mathbf{x} - \mathbf{x}_n\|)_{,k}\mathbf{p}(\mathbf{x}_n)] \\ &= \mathbf{P}^t\mathbf{W}_{,k}\end{aligned} \tag{2.32}$$

Note que as expressões para o MLS, tanto para o cálculo das funções de forma como de suas derivadas, dependem do valor do ponto de avaliação. Dessa forma, precisamos refazer todos os cálculos a cada novo ponto que queremos avaliar.

A Figura 2.3 mostra a função de forma $\phi(x, y)$ e suas derivadas parciais, para o caso bidimensional construída a partir do método dos mínimos quadrados móveis. Utilizou-se uma distribuição uniforme de 5×5 nós. A função de peso w utilizada foi a função *RBF Cubic Spline* definida pela Equação A.4 com raio suporte $r = 1, 2$, como ilustrado na Figura 2.2.

É importante ressaltar que, para o método dos mínimos quadrados móveis, o que se faz é minimizar um funcional quadrático sem forçar que a função aproximada passe pelos pontos conhecidos. Em outras palavras, o valor de $V^h(\mathbf{x}_i)$ não é necessariamente igual ao valor conhecido (exato) da função V no ponto x_i. Ou seja, $V^h(\mathbf{x}_i) \neq V_i$ e as funções de forma geradas pelo MLS não possuem a propriedade do delta de Kronecker (Liu, 2002). Dessa forma, a aproximação para a função em um nó x_i não depende apenas do valor nodal V_i mas sim

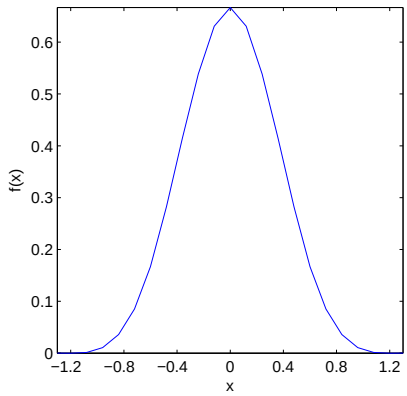


Figura 2.2: Função *RBF Cubic Spline* com raio suporte $r = 1, 2$.

de todos os valores nodais dos pontos dentro do domínio de suporte do nó i. Isso é expresso pelo somatório da Equação 2.2. Essa propriedade faz com que a imposição das condições de contorno usando o MLS não seja tão simples, sendo necessário algum método especial como o método das penalidades.

A consistência da aproximação dada pelo método MLS depende da ordem da base polinomial $p(\mathbf{x})$ utilizada. Se a base polinomial possui monômios de ordem completa k então as funções de forma geradas pelo MLS terão consistência C^k. Para demostrar isso, utilizaremos o argumento dado por Liu (2002) e Kroungauz and Belytschko (1996). Note que o funcional J definido em 2.15 é definido positivo já que escolhemos uma função de peso w positiva. Portanto, o mínimo para o funcional é não negativo. Considere uma função dada por

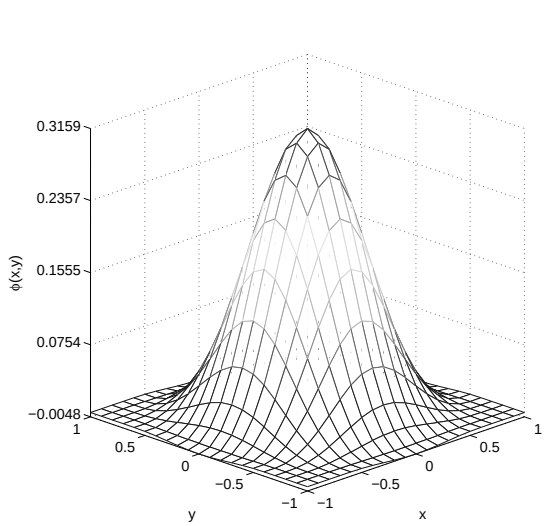
$$V(\mathbf{x}) = \sum_{j=1}^{k} p_j(\mathbf{x})\alpha_j(\mathbf{x}), \quad k \leq m. \tag{2.33}$$

Essa função sempre poderá ser escrita na forma

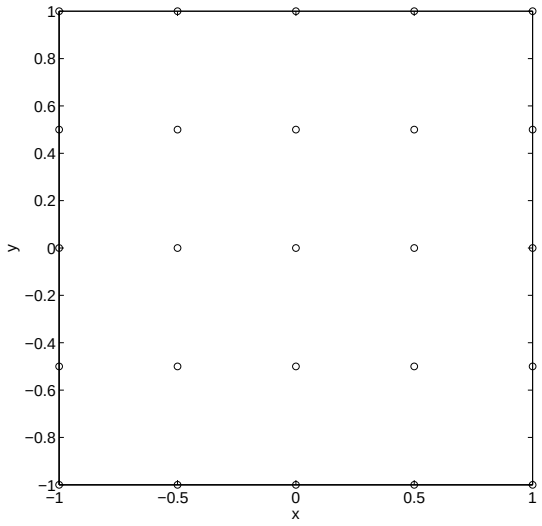
$$V(\mathbf{x}) = \sum_{j=1}^{m} p_j(\mathbf{x})\alpha_j(\mathbf{x}) \tag{2.34}$$

fazendo $\alpha_j(\mathbf{x}) = 0$ para $j > k$. Então, se temos $a_i(\mathbf{x}) = \alpha_i$, J se anula e esse será necessariamente o mínimo para o funcional, o que nos leva a

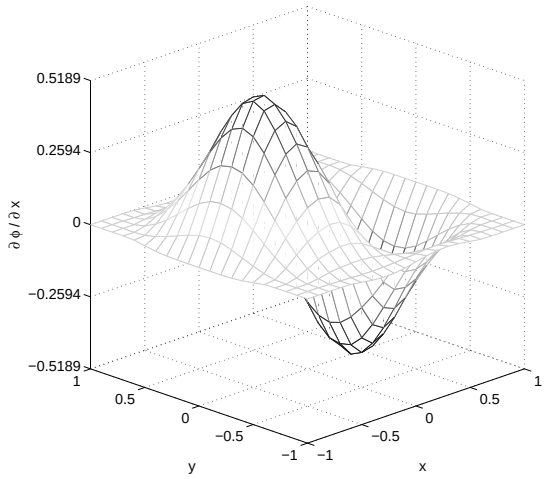
$$V^h(\mathbf{x}) = \sum_{j=1}^{k} p_j(\mathbf{x})\alpha_j(\mathbf{x}) = V(\mathbf{x}) \tag{2.35}$$



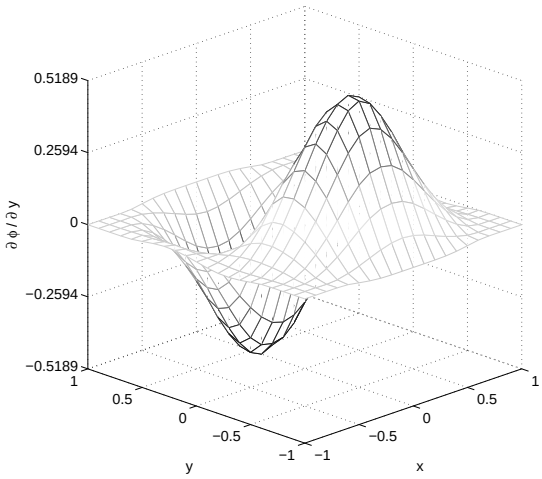
(a) função de forma - MLS



(b) distribuição de nós



(c) derivada parcial em x



(d) derivada parcial em y

Figura 2.3: Função de forma construída utilizando o método de mínimos quadrados móveis. (a) mostra a função de forma $\phi(x, y)$ em duas dimensões para o nó central da distribuição. (b) distribuição de nós utilizados para a construção da função de forma. (c) derivada parcial em relação a x. (d) derivada parcial em relação a y. Para a construção da função de forma foi considerada a base polinomial $\mathbf{p}^t = [1,\ x,\ y]$.

Isso mostra que qualquer função dada por 2.33 será exatamente representada pela aproximação do método MLS. Implica ainda que qualquer função que estiver na base será reproduzida exatamente pelo método.

2.2 Funções de Shepard

As funções de Shepard foram propostas por Shepard em 1968 (Fries and Matthies, 2004). Essas funções podem ser interpretadas como um subcaso do método de mínimos quadrados móveis com ordem zero de consistência, mas com a vantagem de possuírem baixo custo computacional e serem de fácil implementação. Entretanto, a baixa ordem de consistência pode fazer com que as funções de Shepard falhem ao tentar solucionar problemas mais complexos (Fries and Matthies, 2004).

Para definir as funções de Shepard, tomamos a base polinomial como sendo p = [1]. Nesse caso, a matriz $\mathbf{A}(\mathbf{x})$ definida em 2.18 deixa de ser uma matriz $m \times m$ e passa a ser um escalar:

$$\begin{aligned}\mathbf{A}(\mathbf{x}) &= \sum_{i=1}^{n} w(\|\mathbf{x}-\mathbf{x}_i\|) \\ &= w(\|\mathbf{x}-\mathbf{x}_1\|) + w(\|\mathbf{x}-\mathbf{x}_2\|) + \ldots + w(\|\mathbf{x}-\mathbf{x}_n\|)\end{aligned} \tag{2.36}$$

e a matriz $\mathbf{B}$ definida por 2.19 passa a ser um vetor $(1 \times n)$:

$$\mathbf{B}(\mathbf{x}) = [w(\|\mathbf{x}-\mathbf{x}_1\|),\ w(\|\mathbf{x}-\mathbf{x}_2\|),\ \ldots,\ w(\|\mathbf{x}-\mathbf{x}_n\|)] \tag{2.37}$$

Dessa forma, o vetor de funções de forma definido pela Equação 2.22 fica

$$\begin{aligned}\mathbf{\Phi}(\mathbf{x}) &= \mathbf{A}^{-1}\mathbf{B} = [\phi_1,\ \phi_2, \ldots,\ \phi_j, \ldots,\ \phi_n] \\ &= \frac{1}{\sum_{i=1}^{n} w(\|\mathbf{x}-\mathbf{x}_i\|)} [w(\|\mathbf{x}-\mathbf{x}_1\|),\ w(\|\mathbf{x}-\mathbf{x}_2\|),\ \ldots,\ w(\|\mathbf{x}-\mathbf{x}_n\|)]\end{aligned} \tag{2.38}$$

onde

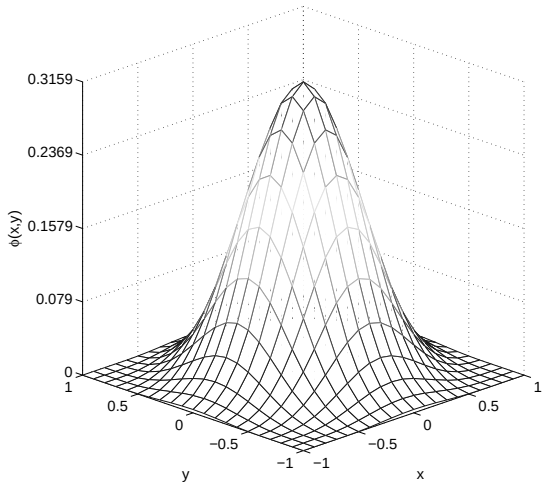
$$\phi_j(\mathbf{x}) = \frac{w(\|\mathbf{x}-\mathbf{x}_j\|)}{\sum_{i=1}^{n} w(\|\mathbf{x}-\mathbf{x}_i\|)} \tag{2.39}$$

Aplicando a derivada da divisão na equação anterior, é fácil ver que as derivadas parciais $\phi_{i,k}$ são dadas por:

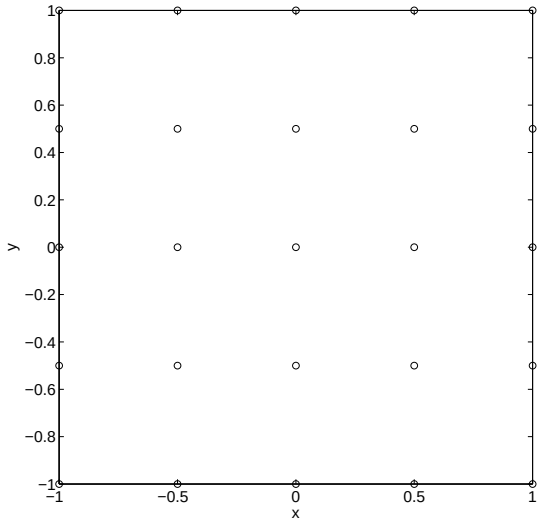
$$\phi_{j,k}(\mathbf{x}) = \frac{w_{,k}(\|\mathbf{x}-\mathbf{x}_j\|)\left[\sum_{i=1}^{n} w(\|\mathbf{x}-\mathbf{x}_i\|)\right] - w(\|\mathbf{x}-\mathbf{x}_j\|)\left[\sum_{i=1}^{n} w_{,k}(\|\mathbf{x}-\mathbf{x}_i\|)\right]}{\left[\sum_{i=1}^{n} w(\|\mathbf{x}-\mathbf{x}_i\|)\right]^2} \tag{2.40}$$

A Figura 2.4 mostra a função de forma $\phi(x, y)$ e suas derivadas parciais, para o caso bidimensional construída a partir do método dos mínimos quadrados móveis. Utilizou-se uma distribuição uniforme de 5×5 nós. A função de peso w utilizada foi a função *RBF Cubic Spline* definida pela Equação A.4 com raio suporte $r = 1, 2$ (Figura 2.2).

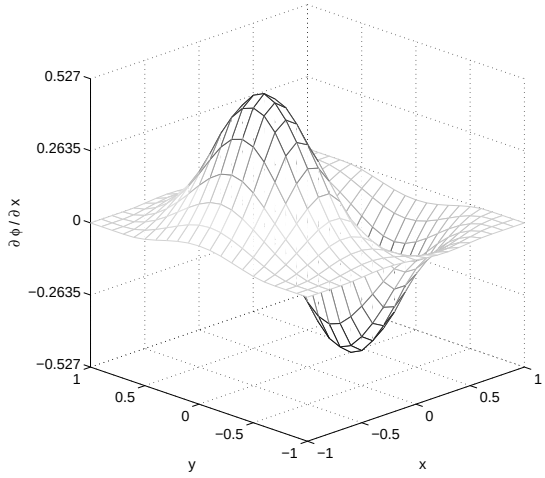
Como discutido anteriormente, a ordem de consistência para o método MLS é dada pela ordem da base polinomial. Como as funções de Shepard são um subcaso do MLS com $p = [1]$, temos que a consistência das funções de Shepard é C^0.



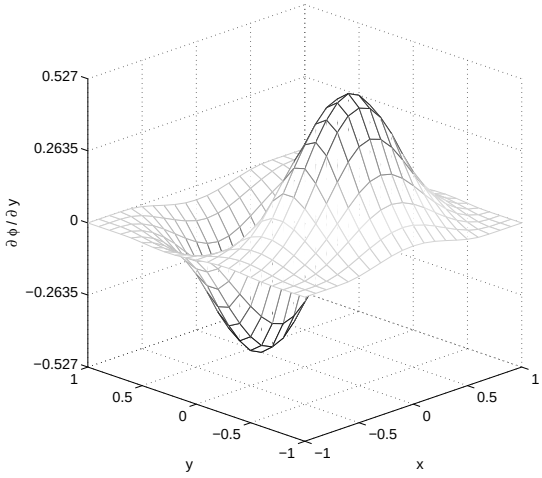
(a) função de forma - Shepard



(b) distribuição de nós



(c) derivada parcial em x



(d) derivada parcial em y

Figura 2.4: Função de forma construída utilizando o método de Shepard. (a) mostra a função de forma $\phi(x, y)$ em duas dimensões para o nó central da distribuição. (b) distribuição de nós utilizados para a construção da função de forma. (c) derivada parcial em relação a x. (d) derivada parcial em relação a y.

2.3 Método de interpolação de pontos (*point interpolation method - PIM*) usando funções de base radial - RPIM

Se para cada um dos n nós $\mathbf{x}_i$ centramos uma RBF, a função $V^h(\mathbf{x})$ pode ser escrita como uma soma ponderada dessas n RBF's:

$$V^h(\mathbf{x}) = a_1 R_1(\mathbf{x}) + a_2 R_2(\mathbf{x}) + ... + a_n R_n(\mathbf{x}) \tag{2.41}$$

ou matricialmente:

$$V^h(\mathbf{x}) = \mathbf{R}^t(\mathbf{x})\mathbf{a} \tag{2.42}$$

onde $\mathbf{R} = [R_1(\mathbf{x}),\ R_2(\mathbf{x}),\ \ldots,\ R_n(\mathbf{x})]^t$ e $\mathbf{a} = [a_1,\ a_2,\ \ldots,\ a_n]^t$.

Para determinar os coeficientes a_i que melhor ajustam a curva, gera-se um sistema de equações lineares nas quais os valores V_i conhecidos da função original devem ser satisfeitos:

$$\begin{aligned} V_1 &= a_1 R_1(\mathbf{x}_1) + a_2 R_2(\mathbf{x}_1) + ... + a_n R_n(\mathbf{x}_1) \\ V_2 &= a_1 R_1(\mathbf{x}_2) + a_2 R_2(\mathbf{x}_2) + ... + a_n R_n(\mathbf{x}_2) \\ &\vdots \\ V_n &= a_1 R_1(\mathbf{x}_n) + a_2 R_2(\mathbf{x}_n) + ... + a_n R_n(\mathbf{x}_n) \end{aligned} \tag{2.43}$$

sendo que V_i é o valor conhecido da função no ponto $\mathbf{x}_i$. Reescrevendo o sistema da Equação 2.43 na forma matricial, tem-se

$$\mathbf{R}_0\mathbf{a} = \mathbf{V} \tag{2.44}$$

na qual

$$\mathbf{V} = [V_1,\ V_2,\ \ldots,\ V_n]^t \tag{2.45}$$

e

$$\mathbf{R}_0 = \begin{bmatrix} R_1(\mathbf{x}_1) & R_2(\mathbf{x}_1) & \cdots & R_n(\mathbf{x}_1) \\ R_1(\mathbf{x}_2) & R_2(\mathbf{x}_2) & \cdots & R_n(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ R_1(\mathbf{x}_n) & R_2(\mathbf{x}_n) & \cdots & R_n(\mathbf{x}_n) \end{bmatrix} \tag{2.46}$$

Como as RBFs são definidas positivas, garante-se que a matriz $\mathbf{R}_0$ é não-singular (Liu, 2002), e, portanto, os coeficientes a_i são obtidos com

$$\mathbf{a} = \mathbf{R}_0^{-1}\mathbf{V}. \tag{2.47}$$

Substituindo 2.47 em 2.42 obtém-se

$$V^h(\mathbf{x}) = \mathbf{R}^t(\mathbf{x})\mathbf{R}_0^{-1}\mathbf{V} = \sum_{i=1}^{n} \phi_i(\mathbf{x})V_i \tag{2.48}$$

onde o vetor com as funções de forma ϕ_i é dado por

$$\mathbf{\Phi}(\mathbf{x}) = \mathbf{R}^t(\mathbf{x})\mathbf{R}_0^{-1} = [\phi_1(\mathbf{x}),\ \phi_2(\mathbf{x}),\ \ldots,\ \phi_i(\mathbf{x}),\ \ldots,\ \phi_n(\mathbf{x})] \tag{2.49}$$

As derivadas parciais de $\mathbf{\Phi}$ podem ser determinadas a partir das derivadas parciais de $\mathbf{R}$ como se segue

$$\Phi(\mathbf{x})_{,k} = [\mathbf{R}^T(\mathbf{x})]_{,k}\mathbf{R}_0^{-1} \tag{2.50}$$

onde

$$[\mathbf{R}(\mathbf{x})]_{,k} = [R_{1,k}(\mathbf{x}),\ R_{2,k}(\mathbf{x}),\ \ldots,\ R_{n,k}(\mathbf{x})]^t \tag{2.51}$$

Como $\mathbf{R}_0$ é não singular, as funções de forma ϕ_i são linearmente independentes. Assim, qualquer vetor de tamanho n será unicamente representado pela combinação linear das n funções de forma. Tomando

$$\mathbf{V} = [V_1, V_2, \ldots, V_i, \ldots, V_n]^T = [0, 0, \ldots, V_i, \ldots, 0]^T \tag{2.52}$$

e substituindo em 2.1, tem-se para $\mathbf{x} = \mathbf{x}_j$, que

$$V^h(\mathbf{x}_j) = \sum_{k=1}^{n} \phi_k(\mathbf{x}_j)V_k = \phi_i(\mathbf{x}_j)V_i. \tag{2.53}$$

Quando $i = j$, tem-se $V_i = \phi_i(\mathbf{x}_i)V_i$ o que implica em $\phi_i(\mathbf{x}_i) = 1$ e $\phi_i(\mathbf{x}_j) = 0$ para $i \neq j$. Dessa forma, as funções de forma RPIM possuem a propriedade do delta de Kronecker.

Como na base do RPIM utilizamos apenas funções de base radial, é de se esperar que o método não consiga reproduzir funções polinomiais. De fato, o RPIM não possui consistência. Para verificar isso, vamos tomar o caso mais simples onde temos uma função constante $V(x) = c$

que desejamos aproximar e nosso conjunto de pontos se reduz a x_1 com uma RBF R_1 centrada em x_1. De acordo com a equação 2.41, temos:

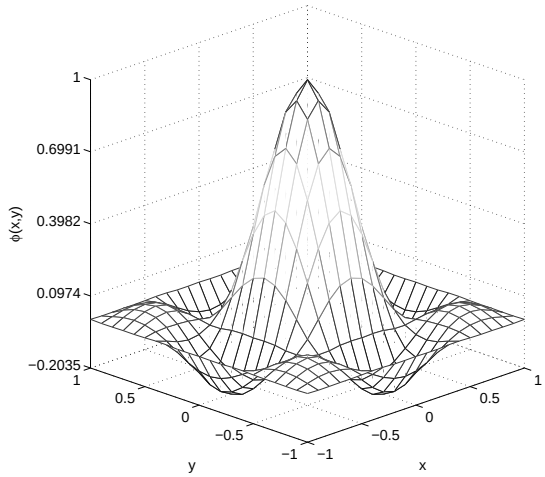
$$V^h(x) = a_1 R_1(x) \tag{2.54}$$

Se o método possuísse ordem de consistência C^0 seriamos capazes de reproduzir a função constante, ou seja, teríamos $V^h(x) = V(x) = c$ e, consequentemente,

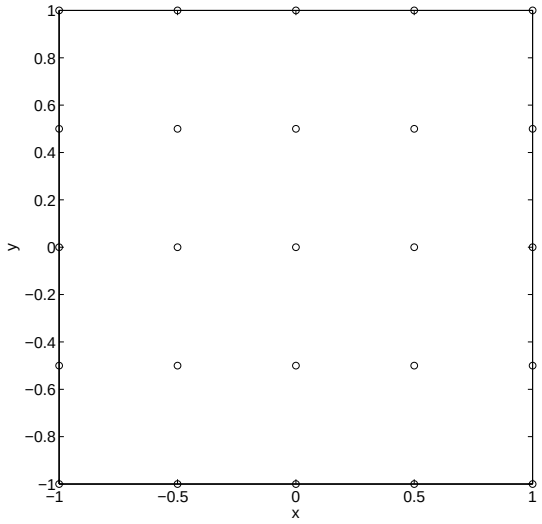
$$a_1 R_1(x) = c \tag{2.55}$$

Isso implicaria que a função de base radial $R_1(x)$ deveria ser constante $R_1(x) = c/a_1$ o que de fato não ocorre. Segundo Liu (2002), qualquer aproximação de funções constantes usando o RPIM converge na medida em que aumentamos o número de nós mas não se consegue reproduzir de forma exata funções polinomiais.

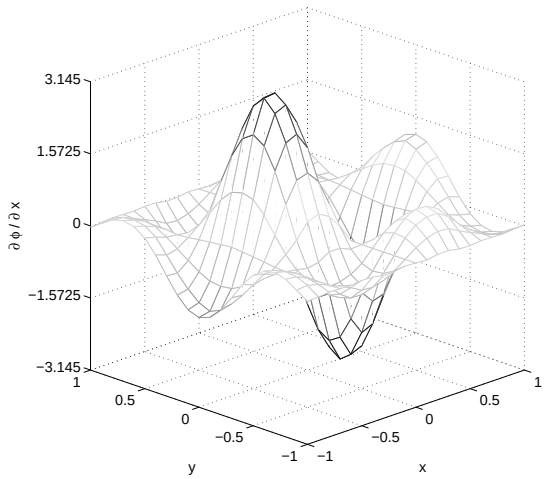
A Figura 2.5 mostra a função de forma $\phi(x, y)$ e suas derivadas parciais, para o caso bidimensional construída a partir do método de interpolação de pontos usando RBFs. Utilizou-se uma distribuição uniforme de 5×5 nós. A função de peso w utilizada foi a função *RBF Cubic Spline* definida pela Equação A.4 com raio suporte $r = 1, 2$ (ver Figura 2.2). A Figura 2.6 mostra a função de forma da Figura 2.5 e sua derivada parcial em relação à x em corte.



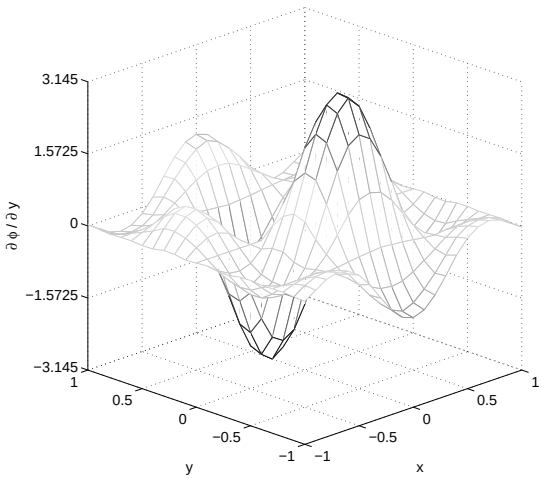
(a) função de forma - RPIM



(b) distribuição de nós

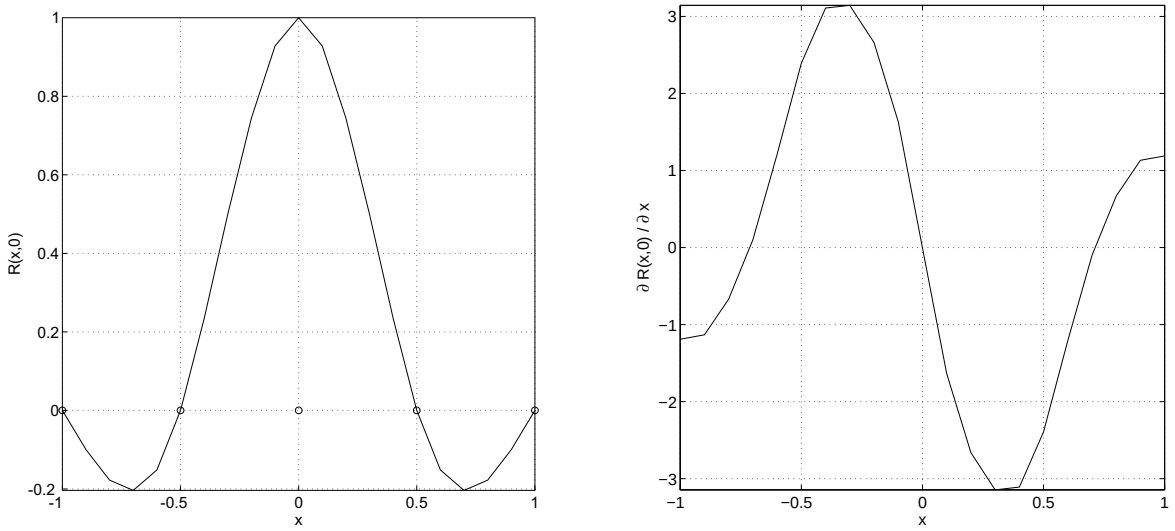


(c) derivada parcial em x



(d) derivada parcial em y

Figura 2.5: Função de forma construída utilizando o método RPIM. (a) mostra a função de forma $\phi(x, y)$ em duas dimensões para o nó central da distribuição. (b) distribuição de nós utilizados para a construção da função de forma. (c) derivada parcial em relação a x. (d) derivada parcial em relação a y.



(a) função de forma em corte

(b) derivada parcial em corte

Figura 2.6: Visão em corte ($y = 0$) da função de forma da Figura 2.5 construída utilizando o método RPIM. (a) mostra a função de forma $\phi(x, y)$ para o corte. Note que a função de forma construída pelo método RPIM possui a propriedade do delta de Kronecker. Os círculos sobre o eixo x mostram as posições dos nós utilizados. A função vale 1 sobre o nó central e zero nos demais. (b) derivada parcial em relação a x.

2.4 Método de interpolação de pontos (*point interpolation method*) usando funções de base radial - RPIMp

O RPIM não forma uma base completa pois utiliza apenas funções de base radial para construir as funções de forma. Dessa forma o método não é consistente, ou seja, não é capaz de reconstruir exatamente componentes lineares. Para solucionar esse problema adicionam-se termos polinomiais à base, obtendo-se o método RPIMp.

Esse tipo de aproximação é uma extensão do método anterior. Para o presente método, além de se associar uma função RBF a cada nó, adicionam-se termos polinomiais. Assim, a Equação 2.42 é modificada tal que

$$V^h(\mathbf{x}) = \mathbf{R}^t(\mathbf{x})\mathbf{a} + \mathbf{p}^t(\mathbf{x})\mathbf{b} \tag{2.56}$$

na qual $\mathbf{R} = [R_1(\mathbf{x}),\ R_2(\mathbf{x}),\ \ldots,\ R_n(\mathbf{x})]^t$ é a matriz das funções RBF ($R(\mathbf{x})$) avaliadas em todos os nós, $\mathbf{a} = [a_1,\ a_2,\ \ldots,\ a_n]^t$ são os coeficientes para as funções RBF, $\mathbf{p}(\mathbf{x})$ é o vetor da base polinomial definido pela Equação 2.13 e $\mathbf{b} = [b_1,\ b_2,\ \ldots,\ b_m]^t$ são os coeficientes para os termos da base polinomial.

Para determinar os coeficientes a_i e b_j que melhor ajustam a curva, gera-se um sistema de equações lineares nas quais os valores conhecidos da função original devem ser satisfeitos:

$$V_k = V^h(\mathbf{x}_k) = \sum_{i=1}^{n} a_i R_i(\mathbf{x}_k) + \sum_{j=1}^{m} b_j p_j(\mathbf{x}_k), \quad k = 1, 2, \ldots, n \tag{2.57}$$

ou na forma matricial:

$$\mathbf{V} = \mathbf{R}_0\mathbf{a} + \mathbf{P}\mathbf{b} \tag{2.58}$$

na qual $\mathbf{R}_0$ é definida pela Equação 2.46, e $\mathbf{P}$ é definida pela Equação 2.25.

Para garantir que a solução seja única, os termos polinomiais devem satisfazer uma condição extra que normalmente é imposta como (Liu, 2002):

$$\sum_{i=1}^{n} p_j(\mathbf{x}) a_i = 0, \quad j = 1, 2, \ldots, m \tag{2.59}$$

ou na forma matricial:

$$\mathbf{P}^t\mathbf{a} = 0 \tag{2.60}$$

Unindo as Expressões 2.58 e 2.60 pode-se obter os coeficientes **a** e **b** pela solução do sistema:

$$\begin{bmatrix} \mathbf{R}_0 & \mathbf{P} \\ \mathbf{P}^t & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix} = \begin{bmatrix} \mathbf{V} \\ \mathbf{0} \end{bmatrix} \tag{2.61}$$

Uma forma mais eficiente de se calcular os coeficientes **a** e **b** é explicitar **a** diretamente a partir da Equação 2.58:

$$\mathbf{a} = \mathbf{R}_0^{-1}\mathbf{V} - \mathbf{R}_0^{-1}\mathbf{P}\mathbf{b} \tag{2.62}$$

Substituindo-se essa expressão em 2.60, obtém-se

$$\mathbf{b} = \mathbf{S}_b\mathbf{V} \tag{2.63}$$

na qual

$$\mathbf{S}_b = \left[\mathbf{P}^t\mathbf{R}_0^{-1}\mathbf{P}\right]^{-1}\mathbf{P}^t\mathbf{R}_0^{-1} \tag{2.64}$$

Substituindo-se 2.63 em 2.62, tem-se

$$\mathbf{a} = \mathbf{S}_a\mathbf{V} \tag{2.65}$$

onde

$$\mathbf{S}_a = \mathbf{R}_0^{-1}\left[1 - \mathbf{P}\mathbf{S}_b\right] = \mathbf{R}_0^{-1} - \mathbf{R}_0^{-1}\mathbf{P}\mathbf{S}_b \tag{2.66}$$

Agora, pode-se expressar a aproximação dada pela expressão como

$$V^h(\mathbf{x}) = \left[\mathbf{R}^t(\mathbf{x})\mathbf{S}_a + \mathbf{p}^t(\mathbf{x})\mathbf{S}_b\right]\mathbf{V} = \Phi(\mathbf{x})\mathbf{V} \tag{2.67}$$

A matriz de funções de forma nodais $\mathbf{\Phi}(\mathbf{x})$ é dada por

$$\mathbf{\Phi}(\mathbf{x}) = \left[\mathbf{R}^t(\mathbf{x})\mathbf{S}_a + \mathbf{p}^t(\mathbf{x})\mathbf{S}_b\right] = \left[\phi_1(\mathbf{x}),\ \phi_2(\mathbf{x}),\ \ldots,\ \phi_k(\mathbf{x}),\ \ldots,\ \phi_n(\mathbf{x})\right] \tag{2.68}$$

na qual $\phi_k(\mathbf{x})$ é a função de forma para o nó k, dada por

$$\phi_k(\mathbf{x}) = \sum_{i=1}^{n} R_i(\mathbf{x})S_{ik}^a + \sum_{j=1}^{m} p_j(\mathbf{x})S_{jk}^b \tag{2.69}$$

onde S^a_{ik} é o elemento (i,k) da matriz $\mathbf{S}_a$ e S^b_{jk} é o elemento (j,k) da matriz $\mathbf{S}_b$. As derivadas de ϕ_k são dadas por

$$\frac{\partial \phi_k}{\partial x} = \sum_{i=1}^{n} \frac{\partial R_i}{\partial x} S^a_{ik} + \sum_{j=1}^{m} \frac{\partial p_j}{\partial x} S^b_{jk} \tag{2.70}$$

Note que, tanto para o cálculo das funções de forma como de suas derivadas, o calculo de $\mathbf{R}_0$ independe do ponto de avaliação. Dessa forma, precisamos calcular as matrizes $\mathbf{S}_a$ e $\mathbf{S}_b$ uma única vez a partir de $\mathbf{R}_0^{-1}$, independentemente do número de pontos que queremos avaliar.

Seguindo um procedimento similar ao apresentado em (Liu, 2002), é possível mostrar que as funções de forma RPIMp possuem a propriedade da função delta de Kronecker. No cálculo das matrizes $\mathbf{S}_a$ (Equação 2.66) e $\mathbf{S}_b$ (Equação 2.64) usa-se a inversa da matriz $\mathbf{R}_0$ (Equação 2.46) que, como visto, é definida positiva e portanto inversível. O termo $\mathbf{P}^T\mathbf{R}_0^{-1}\mathbf{P}$ é uma matriz simétrica e se $n \gg m$, então a matriz terá maior chance de ter posto completo e portanto também será inversível (Liu, 2002). Então, as funções de forma ϕ_i são linearmente independentes e qualquer vetor de tamanho n será unicamente representado pela combinação linear das n funções de forma. Tomando

$$\mathbf{V} = [V_1, V_2, \ldots, V_i, \ldots, V_n]^T = [0, 0, \ldots, V_i, \ldots, 0]^T \tag{2.71}$$

e substituindo em 2.1, tem-se para $\mathbf{x} = \mathbf{x}_j$, que

$$V^h(\mathbf{x}_j) = \sum_{k=1}^{n} \phi_k(\mathbf{x}_j) V_k = \phi_i(\mathbf{x}_j) V_i. \tag{2.72}$$

Quando $i = j$, tem-se $V_i = \phi_i(\mathbf{x}_i)V_i$ o que implica em $\phi_i(\mathbf{x}_i) = 1$ e $\phi_i(\mathbf{x}_j) = 0$ para $i \neq j$. Note-se que a condição $n \gg m$ indica que deve ser utilizado um número mínimo de nós para que o método funcione corretamente.

Considerando uma função dada por

$$V(\mathbf{x}) = \sum_{j=1}^{k} p_j(\mathbf{x})\beta_j, \quad k \leq m, \tag{2.73}$$

sempre poderemos escrevê-la na forma

$$V(\mathbf{x}) = \sum_{i=1}^{n} R(\mathbf{x})\alpha_i + \sum_{j=1}^{m} p_j(\mathbf{x})\beta_j = \mathbf{R}^t(\mathbf{x})\boldsymbol{\alpha} + \mathbf{p}^t(\mathrm{x})\boldsymbol{\beta} \tag{2.74}$$

onde

$$\boldsymbol{\alpha}^t = [0,\ 0, \ldots,\ 0] \tag{2.75}$$

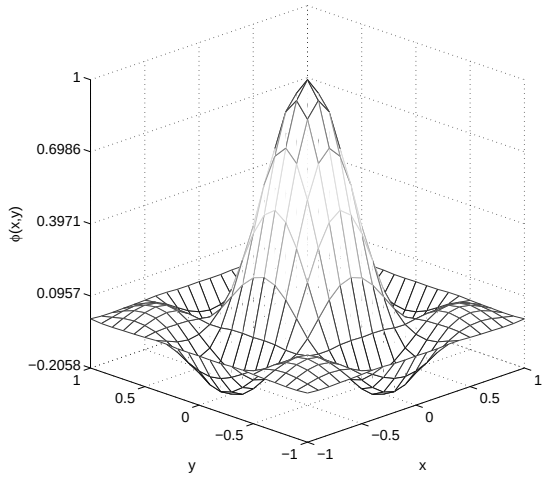
$$\boldsymbol{\beta}^t = [\beta_1,\ \beta_2, \ldots,\ \beta_k,\ 0, \ldots,\ 0] \tag{2.76}$$

Se temos $a_i = \alpha_i$ e $b_j = \beta_j$, então

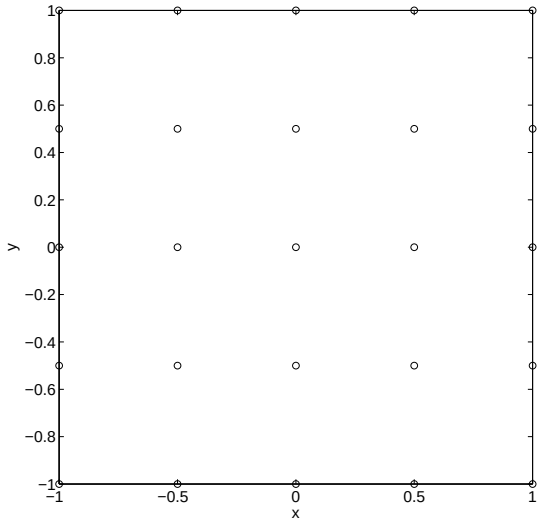
$$V^h(\mathbf{x}) = \sum_{i=1}^{n} R_i(\mathbf{x})\alpha_i \sum_{j=1}^{m} p_j(\mathbf{x})\beta_j = \sum_{j=1}^{k} p_j(\mathbf{x})\beta_j = V(\mathbf{x}) \tag{2.77}$$

Dessa forma, a consistência do RPIMp é garantida pela ordem do polinômio **p** utilizado. Se **p** é uma base polinomial completa de ordem k, então as funções de forma RPIMp terão consistência C^k.

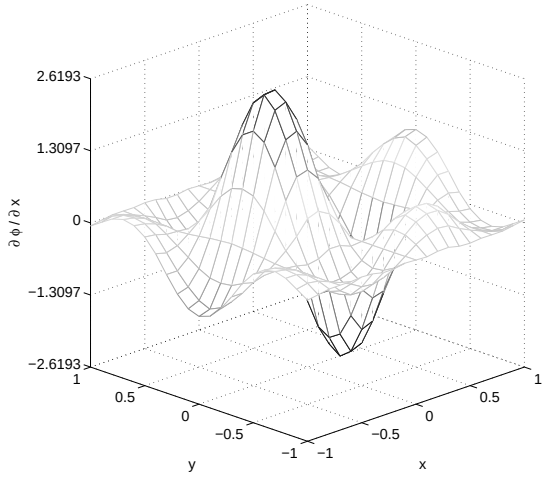
A Figura 2.7 mostra a função de forma $\phi(x, y)$ e suas derivadas parciais, para o caso bidimensional construída a partir do método RPIMp. Utilizou-se uma distribuição uniforme de 5×5 nós. A função de peso w utilizada foi a função *RBF Cubic Spline* definida pela Equação A.4 com raio suporte $r = 1.2$. A Figura 2.8 mostra a função de forma da Figura 2.7 e sua derivada parcial em relação à x em corte.



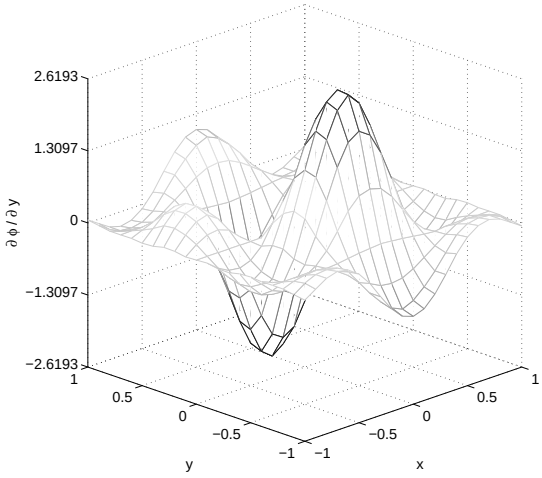
(a) função de forma - RPIMp



(b) distribuição de nós

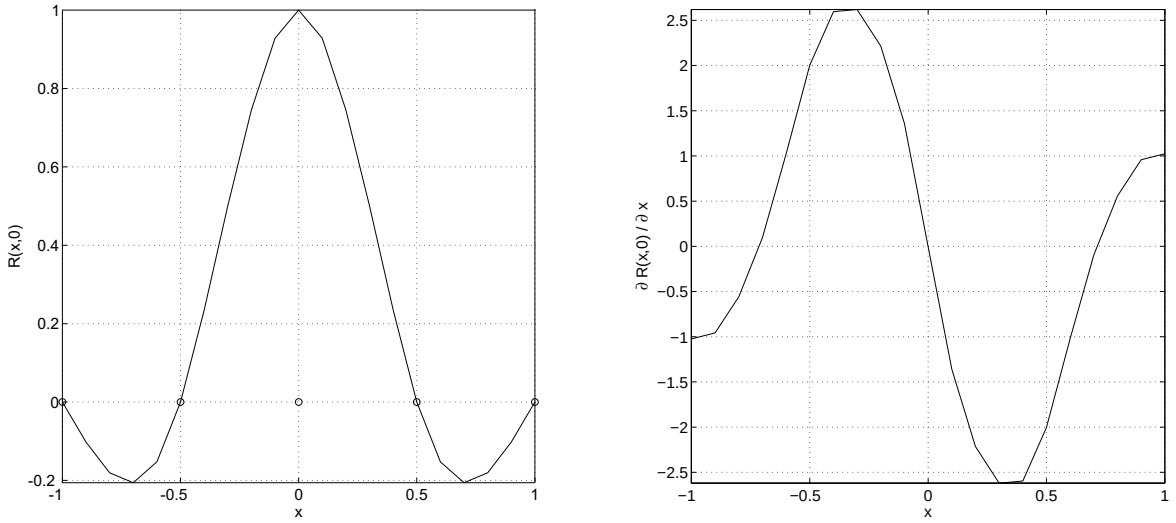


(c) derivada parcial em x



(d) derivada parcial em y

Figura 2.7: Função de forma construída utilizando o método RPIMp. (a) mostra a função de forma $\phi(x, y)$ em duas dimensões para o nó central da distribuição. (b) distribuição de nós utilizados para a construção da função de forma. (c) derivada parcial em relação a x. (d) derivada parcial em relação a y.



(a) função de forma em corte

(b) derivada parcial em corte

Figura 2.8: Visão em corte ($y = 0$) da função de forma da Figura 2.7 construída utilizando o método RPIMp. (a) mostra a função de forma $\phi(x, y)$ em duas dimensões para o nó central da distribuição. Note que a função de forma construída pelo método RPIM possui a propriedade do delta de Kronecker. Os círculos sobre o eixo x mostram as posições dos nós utilizados. A função vale 1 sobre o nó central e zero nos demais. (c) derivada parcial em relação a x.

2.5 Comparação dos métodos de interpolação

A fim de testar a capacidade de aproximação das funções de base geradas pelos métodos apresentados, segue a definição de alguns problemas simples de aproximação em 2D. Nesse momento, consideramos conhecidos os valores da função a ser aproximada em todos os nós. Estamos interessados apenas em verificar experimentalmente a capacidade de interpolação dos métodos.

2.5.1 Função constante

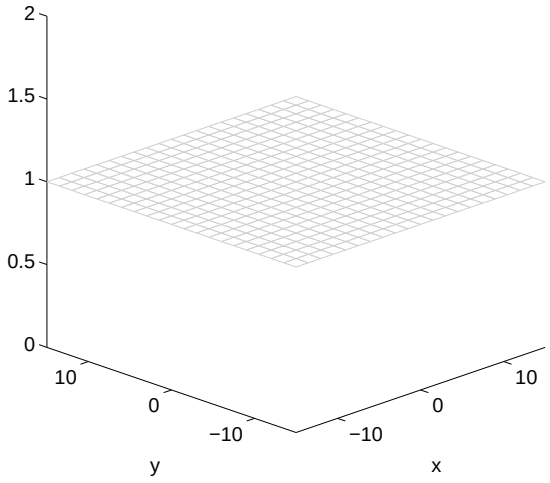
A função constante é um problema bastante simples. O objetivo desse teste é mostrar que o método RPIM, por não possuir uma base completa, não consegue aproximar um componente constante em funções. A função constante é definida por

$$f(x, y) = 1 \tag{2.78}$$

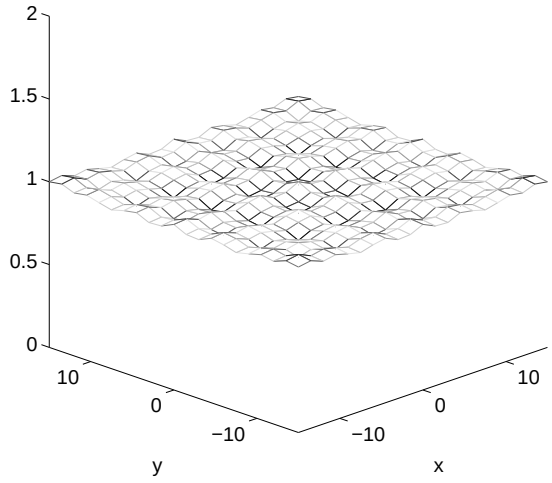
A Figura 2.9a mostra o resultado analítico para a função constante. A Figura 2.9b mostra o resultado obtido pelo método RPIM para a aproximação da função (f^h). As Figuras 2.9c e 2.9d mostram respectivamente o resultado obtido em corte ($y = 0$) e o erro absoluto ($abs(f^h(x, 0) - f(x, 0))$).

Para quantificar erros de aproximação, vamos utilizar o erro quadrático médio percentual definido por

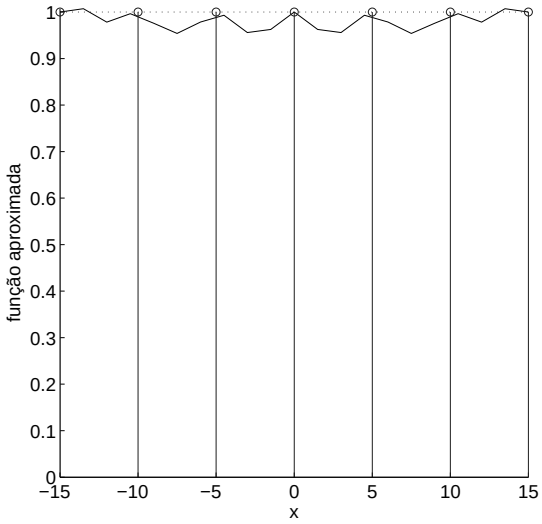
$$Erro_{\%} = 100 \times \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left[\frac{V^h(\mathbf{x}_i) - V(\mathbf{x}_i)}{V(\mathbf{x}_i)}\right]^2} \tag{2.79}$$



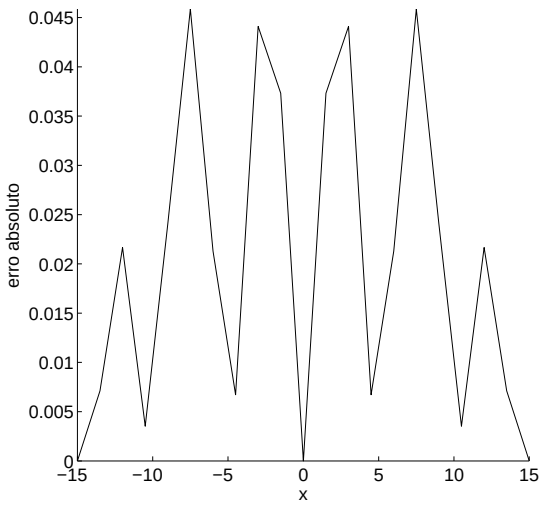
(a) Função constante



(b) Função constante aproximada pelo método RPIM



(c) Corte em $y = 0$



(d) Erro absoluto

Figura 2.9: Aproximação da função constante usando o método RPIM. (a) Solução analítica para a função constante unitária $f(x, y) = 1$. (b) mostra a função aproximada em duas dimensões. (c) mostra a função aproximada em corte ($y = 0$). Os círculos mostram os valores dos nós utilizados. (d) erro absoluto ($abs(f^h(x, 0) - f(x, 0))$). Para essa aproximação foram distribuídos 7×7 nós. Foram distribuídos 15×15 pontos onde se calcularam os valores aproximados da função.

Tabela 2.2: Erro quadrático médio percentual para a aproximação da função constante.

nós	3×3	7×7	29×29	119×119
RPIM	$1,4960\%$	$2,5912\%$	$3,0227\%$	$3,0512\%$
RPIMp	$0,0000\%$	$0,0000\%$	$0,0000\%$	$0,0000\%$
MLS	$0,0000\%$	$0,0000\%$	$0,0000\%$	$0,0000\%$
Shepard	$0,0000\%$	$0,0000\%$	$0,0000\%$	$0,0000\%$

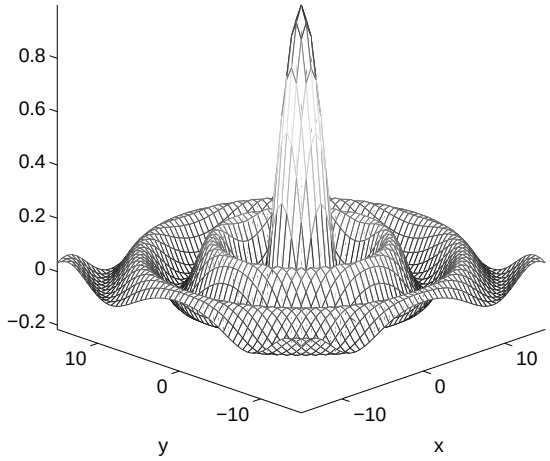
A Tabela 2.2 mostra o erro percentual obtido pelos vários métodos ao se tentar aproximar a função constante. Como era de se esperar, os métodos que possuem grau de consistência maior ou igual a C^0 (RPIMp, MLS e Shepard) conseguem aproximar a função constante de forma exata sem maiores dificuldades. O RPIM por não ter consistência apresenta erro mesmo se utilizando um número elevado de nós, como ilustrado na Figura 2.9. De fato, testes numéricos mostraram ainda que o RPIM não satisfaz a partição da unidade, ou seja, $\sum_{i=1}^{n} \phi_i(x_j) \neq 1$.

2.5.2 Função sinc em duas dimensões

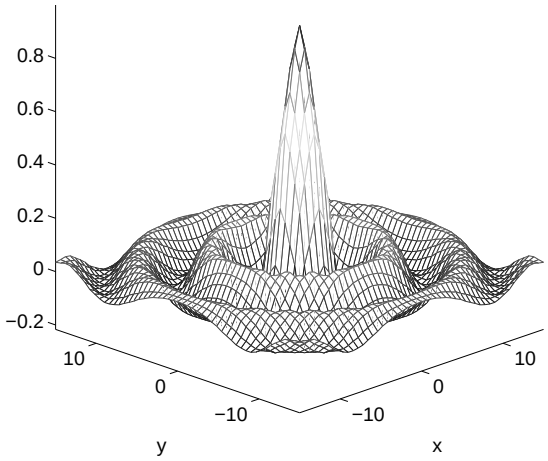
Para testar a capacidade de interpolação das demais funções de forma, usamos a função sinc definida para duas dimensões:

$$f(x,y) = \frac{\operatorname{sen}\left(\sqrt{x^2+y^2}\right)}{\sqrt{x^2+y^2}} \tag{2.80}$$

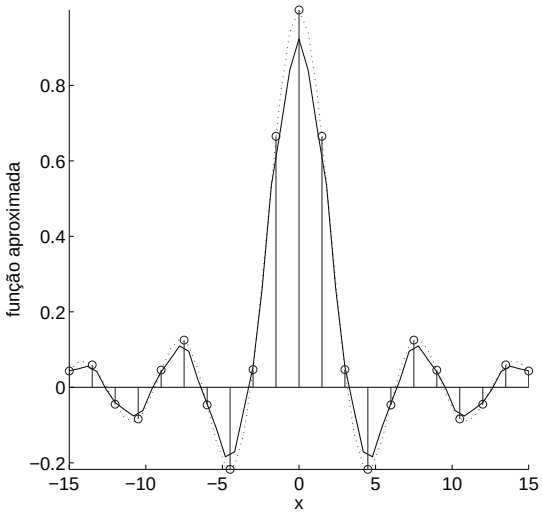
A Figura 2.10a mostra o gráfico da expressão analítica para a função sinc em duas dimensões. Como ilustração, mostramos a solução interpolada para a função usando o método de Shepard na Figura 2.10b. A Figura 2.10c mostra a solução da Figura 2.10b em corte para $y = 0$ e a Figura 2.10d mostra o erro absoluto ponto a ponto para a aproximação.



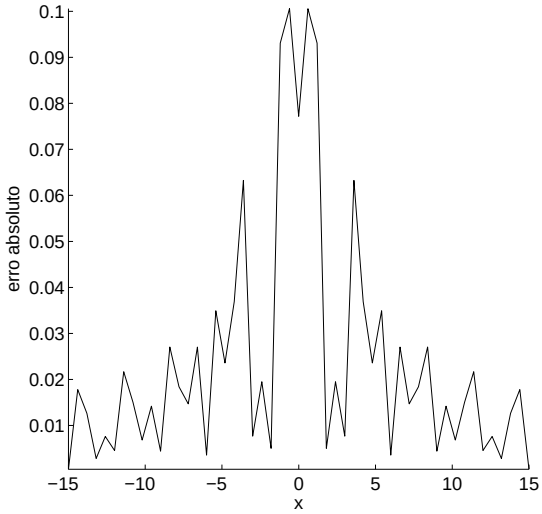
(a) Função sync em 2D



(b) Função sync em 2D aproximada pelo método de Shepard



(c) Corte em $y = 0$



(d) Erro absoluto

Figura 2.10: Aproximação da função sinc usando o método Shepard. (a) Solução analítica para a função sinc em duas dimensões. (b) mostra a função aproximada em duas dimensões. (c) mostra a função aproximada em corte ($y = 0$). Os círculos mostram os valores dos nós utilizados. (d) erro absoluto ($abs(f^h(x, 0) - f(x, 0))$). Para essa aproximação foram distribuídos 21×21 nós. Foram distribuídos 51×51 pontos onde se calcularam os valores aproximados da função.

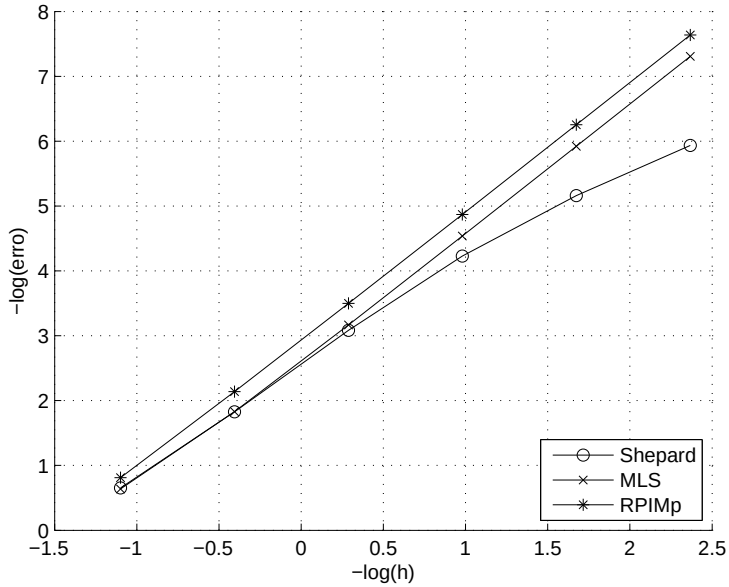


Figura 2.11: Convergência para aproximação da função sinc 2D.

A Figura 2.11 mostra a convergência para os métodos de interpolação vistos nesse capítulo para a função sinc em duas dimensões. Para calcular o erro, utilizou-se a expressão:

$$Erro = \frac{\sqrt{\int_{\Omega} \left[f^h(x,y) - f(x,y)\right]^2 d\Omega}}{\sqrt{\int_{\Omega} \left[f(x,y)\right]^2 d\Omega}} \tag{2.81}$$

Como pode ser visto na figura, a taxa de convergência para os métodos MLS e RPIMp são bastante próximas, sendo que o RPIMp apresenta precisão um pouco maior que o MLS. A taxa de convergência para as funções de Shepard é menor do que a apresentada pelo MLS. Isso já era de se esperar já que elas possuem apenas consistência C^0,

Neste capítulo discutimos alguns tipos de funções de forma que em essência são métodos de interpolação. Como será visto no Capítulo 3 utilizamos as funções de forma para aproximar a solução de um problema regido por equações diferenciais parciais. Como vimos neste capítulo, o método RPIM não possui consistência e por isso não será mencionado nos próximos capítulos. Apesar das equações de Shepard possuírem baixa consistência, seu uso é interessante para diminuir o tempo de processamento. Como será visto no Capítulo 4 unindo as funções

de Shepard com o RPIMp é possível formular um método com boa precisão e baixo custo computacional.

Capítulo 3

Meshless Local Petrov-Galerkin Method

Como discutido na Introdução, o método MLPG (*Meshless Local Petrov-Galerkin Method*) desenvolvido por Atluri and Shen (2002) foi escolhido entre diversos métodos sem malha para ser estudado nessa tese por ser um método bastante flexível e por ser considerado um método verdadeiramente sem malha.

O MLPG difere de outros métodos sem malha baseados no método de Galerkin pois a abordagem Petrov-Galerkin permite que as funções de teste e de forma sejam diferentes (Atluri and Shen, 2002). Esse procedimento torna possível a solução do problema global através de integrações de vários subdomínios locais, ao invés de se resolver uma única integração sobre todo o domínio, como acontece no EFG (Element Free Galerkin) (Parreira et al., 2006b). Cada subdomínio pode ter qualquer tamanho ou forma geométrica* e o conjunto de subdomínios deve cobrir completamente o domínio global.

Nos métodos sem malha, a função de aproximação pode ser definida usando diferentes abordagens, como o método de Mínimos Quadrados Móveis (MLS), Reproducing Kernel Particle Method (RKPM), Funções de Base Radial (RBF), aproximações polinomiais, entre outras. De

* por simplicidade, utilizam-se formas simples como quadrados ou elipses em 2D e cubos e elipsoides em 3D. Neste trabalho utilizamos subdomínios quadrados

acordo com Atluri and Shen (2002), quaisquer desses métodos podem ser usados no MLPG. Atluri sugere também algumas possíveis funções que podem ser usadas para definir a função teste W que veremos a seguir, como a função degrau de Heaviside ou a função MLS. Permutando as possíveis combinações entre funções de teste e de forma, Atluri and Shen (2002) obtêm variações do MLPG que foram denominadas MLPG1, MLPG2, ..., MLPG6. Nesse trabalho, assume-se que W é definida pela função de Heaviside, usada na definição do MLPG5.

A função de Heaviside, como será visto adiante nesse capítulo, permite a simplificação da função da forma fraca local, eliminando a necessidade de integrações no interior dos subdomínios, o que simplifica o método e diminui o custo computacional. Apesar disso, Atluri and Shen (2002) demonstram em seu estudo que o MLPG5 apresenta resultados com precisão equivalente ou melhor às demais variações do método. Como o objetivo principal dessa tese é diminuir o custo computacional de implementações para o método, a variação MLPG5 é a escolha natural a ser estudada.

Nesta tese utilizamos o MLPG para resolver problemas eletromagnéticos. Em especial problemas estáticos (eletrostáticos e magnetostáticos). Iniciamos o capítulo vendo rapidamente as expressões para a forma forte para esse problema e a seguir veremos como resolver o problema a partir da solução de uma forma fraca local além de outros detalhes para o método.

3.1 Modelagem de problemas eletromagnéticos

Problemas eletromagnéticos são descritos matematicamente através das equações de Maxwell, das relações constitutivas do meio e das condições de contorno e de interface entre diferentes materiais (Balanis, 1989; Macedo, 1988). A partir dessas leis particularizamos o caso bidimensional para problemas eletrostáticos e magnetostáticos que serão usados para o estudo do método MLPG.

Esse conjunto de equações simplificadas para os casos eletrostático e magnetostático combinadas com as condições de contorno são denominados forma forte para os problemas eletrostáticos e magnetostáticos.

Problemas estáticos em duas dimensões, tanto elétricos quanto magnéticos, resultam em uma mesma equação diferencial que pode ser generalizada como: dados k, f, g e $\bar{t}$ determinar uma função escalar $u : \Omega \rightarrow R$ que satisfaça

$$\nabla \cdot (k \nabla u) = f \qquad em \quad \Omega \tag{3.1}$$

$$k_1 \frac{\partial u_1}{\partial n} - k_2 \frac{\partial u_2}{\partial n} = c \qquad em \quad \Gamma_{12} \qquad (Interface) \tag{3.2}$$

$$u = g \qquad em \quad \Gamma_u \qquad (Dirichlet) \tag{3.3}$$

$$-k \frac{\partial u}{\partial n} = \bar{t} \qquad em \quad \Gamma_t \qquad (Neumann) \tag{3.4}$$

onde g é a condição de contorno de Dirichlet, $\bar{t}$ é a condição de contorno de Neumann e

$k = \epsilon$ permissividade elétrica do meio $[C^2/Nm^2]$

$u = V$ tensão elétrica $[V]$

$f = \rho$ densidade de carga $[C/m^3]$

$c = \rho_s$ densidade de carga superficial entre os meios $[C/m^2]$

para problemas eletrostáticos e

$k = 1/\mu$ inverso da permeabilidade magnética do meio $[m/H]$

$u = A_z$ componente na direção z do potencial vetor magnético $[Wb/m]$

$f = 0$

$c = 0$

para problemas magnetostáticos.

3.2 Distribuição de subdomínios

Para ilustrar o conceito do MLPG, considere um domínio arbitrário Ω coberto por um conjunto de nós dispostos em seu interior e ao longo de seu contorno Γ, como mostra a Figura 3.1. O contorno Γ é a união do contorno Γ_u – onde se impõe a condição de contorno essencial (Dirichlet) –, e do contorno Γ_t – onde se tem a condição de contorno natural (Neumann). Cada nó distribuído sobre o domínio Ω possui um subdomínio Ω_q. Como exemplo, estão destacados três nós i, j e k, e seus respectivos subdomínios Ω_q^i, Ω_q^j e Ω_q^k. O contorno Γ_q de cada subdomínio é a união dos contornos internos ao domínio global (Γ_{qi}) e da interseção entre o contorno do subdomínio e o contorno global ($\Gamma_{qu} \cup \Gamma_{qt}$).

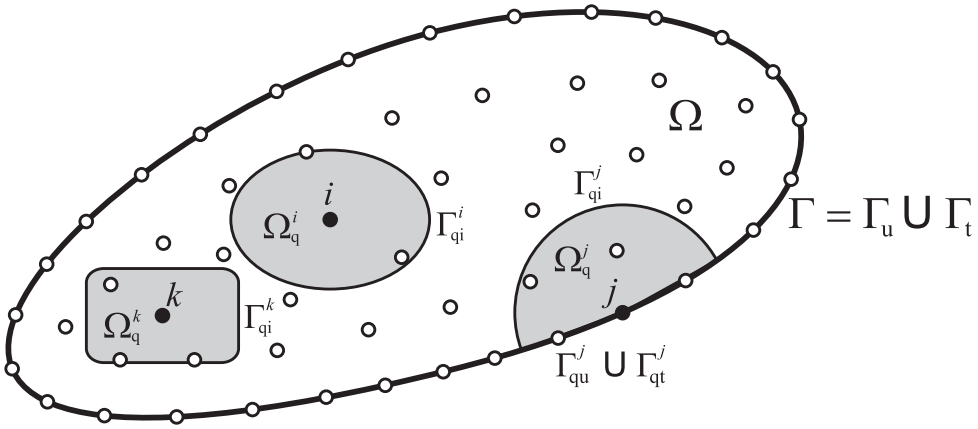


Figura 3.1: Representação do domínio Ω para o MLPG. Em destaque, os subdomínios Ω_q^i, Ω_q^j e Ω_q^k para os nós i, j e k

Os subdomínios podem ter tamanhos e formas diferentes. A única restrição é que todo o domínio global seja coberto pela união dos subdomínios. Apesar de não haver restrição quanto ao formato dos subdomínios, é comum por simplicidade a utilização de quadrados, retângulos, círculos e elipses para o caso bi-dimensional, e de esferas, elipsóides e paralelepípedos para o caso tridimensional.

A seguir serão apresentados alguns conceitos importantes que serão utilizados ao longo do texto: A distância média entre os nós é dada por h. O domínio de influência de um nó é a região na qual esse nó exerce influência no cálculo das funções de forma e na integração dos subdomínios para os nós vizinhos. Este domínio de influência normalizado pela distância média h entre os nós é dada por α_s:

$$\alpha_s = \frac{\text{raio do domínio de influência}}{h} \tag{3.5}$$

O subdomínio Ω_q do nó, também chamado de domínio de quadratura, é a região em torno do nó onde queremos que a forma fraca para o problema seja satisfeita. Para isso, fazemos a integração da forma fraca sobre essa região que gera a contribuição do nó para a solução do problema. Veremos mais detalhes adiante. O raio do subdomínio normalizado pela distância média entre os nós é dada por α_q:

$$\alpha_q = \frac{\text{raio do subdomínio}}{h} \tag{3.6}$$

3.3 Forma fraca para um subdomínio

Para cada subdomínio, reescreve-se o problema a ser resolvido por meio do método dos resíduos ponderados. Dessa forma, utilizando a forma forte para os problemas estáticos definida na Equação 3.1, para um determinado nó i sobre o domínio Ω, obetemos a seguinte expressão:

$$\int_{\Omega_q^i} W_i \left[\nabla \cdot (k\nabla u) - f\right] d\Omega = 0 \tag{3.7}$$

na qual W_i é a função de teste associada ao nó i. Normalmente utilizam-se funções com suporte compacto como funções de teste. Funções de suporte compacto são diferentes de zero em uma região limitada, anulando-se fora dessa região. Isso faz com que a integral seja calculada em uma região limitada, onde a função é não nula. Apesar de não ser uma exigência do método, para simplificar, costuma-se fazer o suporte da função de teste coincidir com o subdomínio do nó.

Separando-se os termos de 3.7, tem-se

$$\int_{\Omega_q^i} W_i \nabla \cdot (k\nabla u)\, d\Omega - \int_{\Omega_q^i} W_i f d\Omega = 0 \tag{3.8}$$

Aplicando-se a identidade vetorial dada por

$$\nabla \cdot (g\mathbf{v}) = \nabla g \cdot \mathbf{v} + g\nabla \cdot \mathbf{v} \quad \Longrightarrow \quad g\nabla \cdot \mathbf{v} = \nabla \cdot (g\mathbf{v}) - \nabla g \cdot \mathbf{v} \tag{3.9}$$

na primeira integral de 3.8, obtém-se

$$\int_{\Omega_q^i} W_i \nabla \cdot (k\nabla u)\, d\Omega = \int_{\Omega_q^i} \nabla \cdot (W_i k\nabla u)\, d\Omega - \int_{\Omega_q^i} \nabla W_i \cdot k\nabla u d\Omega \tag{3.10}$$

Aplicando-se o teorema da divergência, dado por

$$\int_{\Omega} \nabla \cdot \mathbf{A} d\Omega = \int_{\Gamma} \mathbf{A} \cdot \mathbf{n} d\Gamma, \qquad \Gamma = \partial\Omega \tag{3.11}$$

na primeira integral do segundo membro de 3.10, tem-se

$$\int_{\Omega_q^i} \nabla \cdot (W_i k \nabla u)\, d\Omega = \int_{\Gamma_q^i} (W_i k \nabla u) \cdot \mathbf{n} d\Gamma \tag{3.12}$$

Substituindo-se 3.12 e 3.10 em 3.8:

$$\int_{\Gamma_q^i} (W_i k \nabla u) \cdot \mathbf{n} d\Gamma - \int_{\Omega_q^i} \nabla W_i \cdot k \nabla u d\Omega - \int_{\Omega_q^i} W_i f d\Omega = 0 \tag{3.13}$$

Como $\Gamma_q = \Gamma_{qu} \cup \Gamma_{qt} \cup \Gamma_{qi}$ e $k\nabla u \cdot \mathbf{n} = k\frac{\partial u}{\partial n} = \bar{t}$ em Γ_{qt} (condição de fronteira de Neumann), pode-se dividir a primeira integral de 3.13:

$$\int_{\Gamma_q^i} (W_i k \nabla u) \cdot \mathbf{n} d\Gamma = \int_{\Gamma_{qu}^i \cup \Gamma_{qi}^i} (W_i k \nabla u) \cdot \mathbf{n} d\Gamma + \int_{\Gamma_{qt}^i} W_i \bar{t} d\Gamma \tag{3.14}$$

dessa forma, a Equação 3.13 fica

$$\int_{\Gamma_{qu}^i \cup \Gamma_{qi}^i} (W_i k \nabla u) \cdot \mathbf{n} d\Gamma + \int_{\Gamma_{qt}^i} W_i \bar{t} d\Gamma - \int_{\Omega_q^i} \nabla W_i \cdot k \nabla u d\Omega - \int_{\Omega_q^i} W_i f d\Omega = 0 \tag{3.15}$$

Essa equação é chamada de *forma fraca* para o problema estático/magnetostático em duas dimensões, definida localmente para o nó i.

3.4 Discretização da solução

Para resolver computacionalmente o problema dado pela Equação 3.15, aproxima-se a função u por u^h dada por

$$u^h = \sum_{j=1}^{n} \phi_j u_j \tag{3.16}$$

na qual ϕ_j é uma função de aproximação chamada *função de forma*, u_j são os valores nodais e n é o número de nós distribuídos sobre o domínio. As funções de forma podem ser determinadas

por vários métodos. Entre eles o MLS, funções de Shepard, RPIM e RPIMp vistos em detalhe no Capítulo 2 entre outros métodos como o método da Partição da Unidade (Nealen, 2004).

Aplicando-se o operador gradiente em 3.16, obtém-se

$$\nabla u^h = \nabla \sum_{j=1}^{n} \phi_j u_j = \sum_{j=1}^{n} \nabla \phi_j u_j \tag{3.17}$$

Substituindo-se 3.16 e 3.17 em 3.15 e fazendo-se $\nabla \Phi \cdot \mathbf{n} = \frac{\partial \Phi}{\partial n}$, tem-se

$$\sum_{j=1}^{n} \left[\int_{\Gamma^i_{qu} \cup \Gamma^i_{qi}} W_i k \frac{\partial \phi_j}{\partial n} d\Gamma - \int_{\Omega^i_q} \nabla W_i \cdot k \nabla \phi_j d\Omega \right] u_j = \int_{\Omega^i_q} W_i f d\Omega - \int_{\Gamma^i_{qt}} W_i \bar{t} d\Gamma \tag{3.18}$$

ou, na forma matricial,

$$\sum_{j=1}^{n} K_{ij} u_j = F_i \Rightarrow \mathbf{KU} = \mathbf{F} \tag{3.19}$$

na qual **K** é uma matriz $n \times n$ e **F** é um vetor $n \times 1$. O elemento (i, j) da matriz **K** é representado por K_{ij} e a i-ésima posição de **F** é representada por F_i. Observando-se as Equações 3.18 e 3.19, define-se K_{ij} e F_i como

$$\begin{aligned} K_{ij} &= \int_{\Gamma^i_{qu} \cup \Gamma^i_{qi}} W_i k \frac{\partial \phi_j}{\partial n} d\Gamma - \int_{\Omega^i_q} \nabla W_i \cdot k \nabla \phi_j d\Omega \\ F_i &= \int_{\Omega^i_q} W_i f d\Omega - \int_{\Gamma^i_{qt}} W_i \bar{t} d\Gamma \end{aligned} \tag{3.20}$$

3.5 Imposição das condições de contorno essenciais

Quando se constroem as funções de forma a partir de um método que possui a propriedade do delta de Kronecker (como o RPIM ou RPIMp), a imposição das condições de contorno de Dirichlet são impostas de forma direta. Dessa forma, os valores já conhecidos na fronteira de Γ_u podem ser diretamente computados no vetor **F**. Assim, a ordem do sistema passa a ser

igual ao número de nós desconhecidos, isto é, número total de nós menos o número de nós em Γ_u e a Equação 3.20 fica:

$$\begin{aligned} K_{ij} &= \int_{\Gamma^i_{qu}\cup\Gamma^i_{qi}} W_i k \frac{\partial \phi_j}{\partial n} d\Gamma - \int_{\Omega^i_q} \nabla W_i \cdot k \nabla \phi_j d\Omega \\ F_i &= \int_{\Omega^i_q} W_i f d\Omega - \int_{\Gamma^i_{qt}} W_i h d\Gamma - \sum_{j=1}^{m} \left[\int_{\Gamma^i_{qu}\cup\Gamma^i_{qi}} W_i k \frac{\partial \phi_j}{\partial n} d\Gamma - \int_{\Omega^i_q} \nabla W_i \cdot k \nabla \phi_j d\Omega \right] g_j \end{aligned} \tag{3.21}$$

nas quais m é o número de nós em Γ_u e g_k é o valor conhecido de u no nó k em Γ_u.

Entretanto, para funções de forma que não possuem a propriedade do delta de Kronecker como o MLS e as funções de Shepard, necessita-se de algum método para forçar as condições de contorno, como o método das penalidades ou multiplicadores de Lagrange (Liu, 2002; Fries and Matthies, 2004). Nesse trabalho optamos pelo método das penalidades, pois sua implementação é simples e não são geradas novas incógnitas mantendo a ordem do sistema linear.

O método das penalidades consiste em forçar o valor da solução na fronteira de Dirichlet para os valores conhecidos. Ou seja $u = g$ em Γ_u. Para fazer isso introduz-se o termo $\alpha \int_{\Gamma^i_{qu}} W_i(u-g)d\Gamma$ à Equação 3.7 que fica:

$$\int_{\Omega^i_q} W_i \left[\nabla \cdot (k \nabla u) - f\right] d\Omega + \alpha \int_{\Gamma^i_{qu}} W_i (u-g) d\Gamma = 0. \tag{3.22}$$

Aqui, α é uma constante (termo de penalidade) de valor elevado. Como a função de teste W_i é qualquer, a equação acima só será nula se cada integral for nula. Desenvolvendo as equações com o novo termo, passamos a definir os termos K_{ij} e F_i como:

$$\begin{aligned} K_{ij} &= \int_{\Gamma^i_{qu}\cup\Gamma^i_{qi}} W_i k \frac{\partial \phi_j}{\partial n} d\Gamma - \int_{\Omega^i_q} \nabla W_i \cdot k \nabla \phi_j d\Omega + \alpha \int_{\Gamma^i qu} W_i \phi_j d\Gamma \\ F_i &= \int_{\Omega^i_q} W_i f d\Omega - \int_{\Gamma^i_{qt}} W_i h d\Gamma + \alpha \int_{\Gamma^i qu} W_i g d\Gamma \end{aligned} \tag{3.23}$$

Dessa forma, os nós da fronteira fazem parte do sistema matricial, elevando o custo computacional de sua resolução. Além disso, o parâmetro α introduz valores elevados na matriz piorando seu condicionamento, o que piora a precisão da solução. A escolha do parâmetro α que minimiza esses efeitos nem sempre é simples.

Ikuno et al. (2007) estudaram a influência do parâmetro α sobre a precisão dos resultados. Nesse estudo conclui-se que os melhores resultados são encontrados quando $10^4 < \alpha < 10^8$. Dessa forma, neste trabalho escolheu-se $\alpha = 10^6$.

3.6 Escolha da função de teste

Como discutido no capítulo de introdução, a escolha da função de teste no MLPG é independente da escolha das funções de forma. Ou seja, funções de forma e de teste podem pertencer a espaços de funções diferentes (Atluri and Shen, 2002; Liu, 2002). Essa possibilidade cria variações para o MLPG as quais Atluri and Shen (2002) denominou MLPG1, MLPG2, ..., MLPG6.

Como exemplo, no MLPG1 utiliza-se o mesmo tipo de função (MLS) tanto para a função de teste quanto para se construir as funções de forma. No MLPG2, A função de teste é a função delta de Dirac resultando em um método de colocação, etc. Dentre as variações do método, escolhemos trabalhar com o MLPG5, pois a função de teste utilizada (Heaviside) faz com que termos de integração no interior do domínio se anulem tornando o método mais eficiente. Além disso, o trabalho comparativo realizado por Atluri and Shen (2002) mostra que o MLPG5 é a variação do método que alcança os melhores resultados.

A função de Heaviside é definida como sendo constante e igual a 1 dentro de seu domínio suporte Ω_w, e zero fora dele. A Figura 3.2 ilustra a função de Heaviside, definida pela Equação 3.24, para um domínio bidimensional.

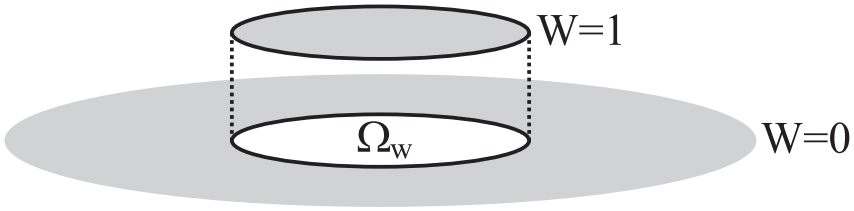


Figura 3.2: Função de Heaviside 2D

$$W(\mathbf{x}) = \begin{cases} 1 & \text{se } \mathbf{x} \in \Omega_w \\ 0 & \text{se } \mathbf{x} \notin \Omega_w \end{cases} \tag{3.24}$$

Escolhendo-se a função de teste W como sendo a função de Heaviside, o termo $\int_{\Omega_q^i} k\nabla W_i \nabla \phi_j d\Omega$ de K_{ij} se anula, já que $\nabla W = 0$. Dessa forma, são necessários apenas os cálculos das integrais na borda do domínio Ω_q (Γ_q). Dessa forma, tem-se

$$\begin{aligned} K_{ij} &= \int_{\Gamma_{qu}^i \cup \Gamma_{qi}^i} k\frac{\partial \phi_j}{\partial n} d\Gamma + \alpha \int_{\Gamma_{qu}^i} \phi_j d\Gamma \\ F_i &= -\int_{\Gamma_{qt}^i} h d\Gamma + \int_{\Omega_q^i} f d\Omega + \alpha \int_{\Gamma_{qu}^i} g d\Gamma \end{aligned} \tag{3.25}$$

quando se utiliza o método das penalidades e

$$\begin{aligned} K_{ij} &= \int_{\Gamma_{qu}^i \cup \Gamma_{qi}^i} k\frac{\partial \phi_j}{\partial n} d\Gamma \\ F_i &= -\int_{\Gamma_{qt}^i} h d\Gamma + \int_{\Omega_q^i} f d\Omega - \sum_{j=1}^{m} \left[\int_{\Gamma_{qu}^i \cup \Gamma_{qi}^i} k\frac{\partial \phi_k}{\partial n} d\Gamma \right] g_k \end{aligned} \tag{3.26}$$

quando a função de forma utilizada possui a propriedade do delta de Kronecker.

Neste capítulo apresentamos o método MLPG. Mostramos a ideia geral do método e como podemos simplificá-lo usando uma variação do método (MLPG5) onde a função de teste utilizada é a função degrau de Heaviside. Foi discutida também a influência das funções de forma. Funções de forma que não possuem a propriedade do Delta de Kronecker tornam necessário o uso de alguma técnica especial para impor as condições de contorno de Dirichlet. No Capítulo 4, discutiremos como implementar de forma eficiente as ideias discutidas até aqui. Como será visto nesse capítulo, apesar das funções de forma sem a propriedade do delta de Kronecker serem mais rápidas, elas introduzem oscilações na fronteira de Dirichlet. Uma formulação mista é então proposta para se obter um método com boa precisão e menor custo computacional.

Capítulo 4

Implementações e resultados

Neste capítulo, serão abordados os procedimentos utilizados na etapa de integração numérica que resultará na montagem do sistema linear (Equação 3.19), bem como os métodos desenvolvidos para agilizar esse processo e para a solução do sistema resultante. Apresentaremos inicialmente uma visão geral das etapas do método e, no decorrer do capítulo, mostramos alguns detalhes importantes de implementação.

Alguns problemas simples são propostos para validar o métodos implementados. Problemas cuja solução analítica é desconhecida terão uma solução calculada usando o método de elementos finitos com uma malha bastante densa. Dessa forma, consideraremos essa solução, que chamaremos de solução densa, como a solução de referência para o problema.

A Figura 4.1 apresenta o algoritmo implementado para o MLPG mostrando em alto nível os principais passos para se resolver um dado problema. Iniciamos lendo do arquivo de entrada as informações sobre a geometria do problema, condições de contorno e os nós distribuídos sobre o domínio (linha 1). A partir daí iniciamos a montagem do sistema matricial calculando a contribuição de cada nó (linhas de 2 a 7). Com o sistema linear em mãos, calculamos sua solução obtendo os valores nodais (linha 8) e, com esses valores calculamos a aproximação da solução em pontos de interesse (linhas de 9 a 12). Por fim, escrevemos a solução no disco (linha 13).

```
1  Leitura do problema;
2  para cada nó sobre o domínio faça
3  |  Calcular a geometria do subdominio e os pontos de integração;
4  |  Achar os nós vizinhos que exercem influência nos pontos de integração;
5  |  Com os nós vizinhos, construir as funções de forma;
6  |  Calcular a contribuição do nó no sistema linear pela integração da forma fraca
   |  no subdomínio;
7  fim para cada
8  Resolver o sistema linear encontrando os valores nodais;
9  para cada ponto onde se deseja calcular a solução faça
10 |  Achar os nós vizinhos que exercem influência sobre o ponto solução;
11 |  Com os nós vizinhos, construir as funções de forma e estimar a solução;
12 fim para cada
13 Escrever a solução;
```

Figura 4.1: Visão geral do algoritmo implementado

Ao longo do capítulo discutimos as estratégias escolhidas para implementar o algoritmo da Figura 4.1. Começamos discutindo a distribuição dos pontos de integração sobre o subdomínio dos nós e das soluções geométricas para representar os subdomínios onde utilizamos a biblioteca de Geometria Computacional CGAL (CGAL, 2007). Na sequencia mostramos a estratégia utilizada para a localização de nós vizinhos utilizando uma árvore de busca também disponibilizada pela CGAL.

Em nossas primeiras implementações, utilizamos o MLS e o RPIMp para construir as funções de forma. Com o MLS utilizamos o método de penalidades para impor as condições de contorno de Dirichlet, enquanto com o RPIMp nós impomos as condições de contorno de maneira direta. Na tentativa de diagnosticar os fatores que mais impactam o tempo de processamento, mediu-se o tempo relativo para os principais passos para essas implementações (os passos para encontrar a interpolação nos pontos de interesse e escrita da solução em disco foram desconsiderados) ao se resolver um problema eletrostático simples.* Os resultados obtidos, apresentados na Tabela 4.1, mostram que o processo de montagem do sistema linear é o principal limitador do método e merece maior atenção.

Para o processo de montagem, verificou-se também que, para as duas formas de construção das funções de forma, o tempo gasto para a busca de nós vizinhos utilizando a *kd-tree* representa

* Trata-se do problema da calha definido mais a frente nesse capítulo.

Tabela 4.1: Tempo relativo para os principais passos da solução do MLPG para implementações sequenciais utilizando funções de forma obtidas por MLS e RPIMp. ($\alpha_s = 1,6569$, $\alpha_q = 3,3137$, ordem de integração $= 4$)

	225 nós (h = 0.8047)		1600 nós (h = 0,3018)		3600 nós (h=0,2012)	
	MLS	RPIMp	MLS	RPIMp	MLS	RPIMp
Leitura do problema	1,09%	0,26%	0,83%	0,27%	0,18%	0,09%
Montagem do sistema	96,72%	99,46%	95,78%	98,88%	96,10%	99,26%
Solução do sistema	2,18%	0,28%	3,37%	0,84%	3,72%	0,65%

menos de 2% do tempo gasto, sendo a construção das funções de forma e a integração dos subdomínios responsáveis por mais de 98% da carga desse passo no método. Portanto, a montagem do sistema é o passo em que mais concentramos nossos esforços para acelerar o método. As solução proposta para diminuir o tempo de processamento desse passo do algoritmo são apresentadas a seguir na Seção 4.5 e na Seção 4.7.

Notamos também que o tempo de processamento das implementações com MLS, em determinadas condições, é menor que o tempo de processamento utilizando RPIMp. Apesar disso, o método de penalidades provoca oscilações próximas ao contorno de Dirichlet, baixando a precisão do método quando se usa MLS. Tendo isso em vista, propomos nesse capítulo um método misto que une as duas funções de forma. Nesse método conseguimos impor as condições de contorno de maneira direta, fugindo das oscilações provocadas pelo método de penalidades e com um tempo de processamento próximo ao MLS. Para acelerar ainda mais o processo propomos a utilização das funções de Shepard em substituição ao MLS.

4.1 Integração dos subdomínios

As equações desenvolvidas no Capítulo 3, em especial as Equações 3.25 e 3.26, indicam que, para um dado nó i, é necessário calcular a integral de linha na fronteira do seu subdomínio correspondente Ω_q^i como mostrado na Figura 4.2. Como a função de peso utilizada é a função de Heaviside, a integração no interior do subdomínio é necessária apenas para calcular a contribuição do termo fonte no vetor F do sistema matricial global. Em cada segmento da fronteira, a integral de linha é calculada através da quadratura de Gauss (Hughes, 2000; Hildebrand, 1987), sendo necessário determinar a ordem de integração que define a quantidade

de pontos a serem utilizados. Na Figura 4.2 exemplifica-se uma quadratura de ordem 3, definindo um total de pontos igual a 12.

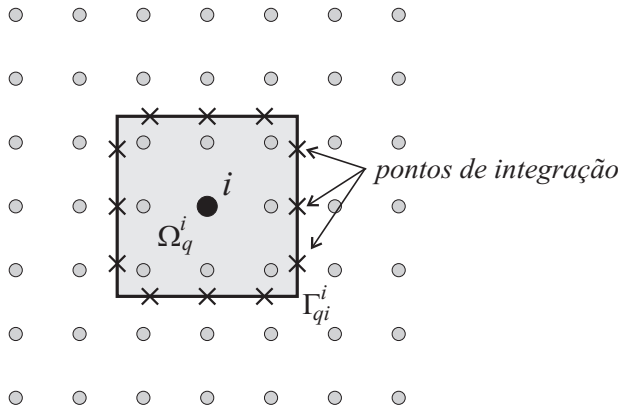


Figura 4.2: Subdomínio de integração definido pelo nó i. A ordem de integração determina o número de pontos de integração sobre cada lado da caixa de integração. Para uma ordem de integração n são usados $4 \times n$ pontos de integração.

Quanto menor a quantidade de pontos de integração, menor é o número de avaliações da função de forma. Para o MLS é vantajoso o menor número de pontos de integração possível, pois é necessário calcular a função de forma para cada ponto (inversão de muitas matrizes). Já para o RPIMp, a inversão da matriz $\mathbf{R}_0$ é feita uma única vez, independentemente do número de pontos de integração a serem avaliados, isso torna o custo computacional do RPIMp menos sensível ao número de pontos de integração em termos de tempo de execução.

A Figura 4.3 mostra como o erro varia quando aumentamos o número de pontos de integração. Veja que o erro estabiliza e apresenta variação muito pequena a partir da ordem de integração 4 (16 pontos de integração). Dessa forma, não se justifica utilizar uma ordem de integração maior, já que não há melhora na solução e o custo computacional aumenta.

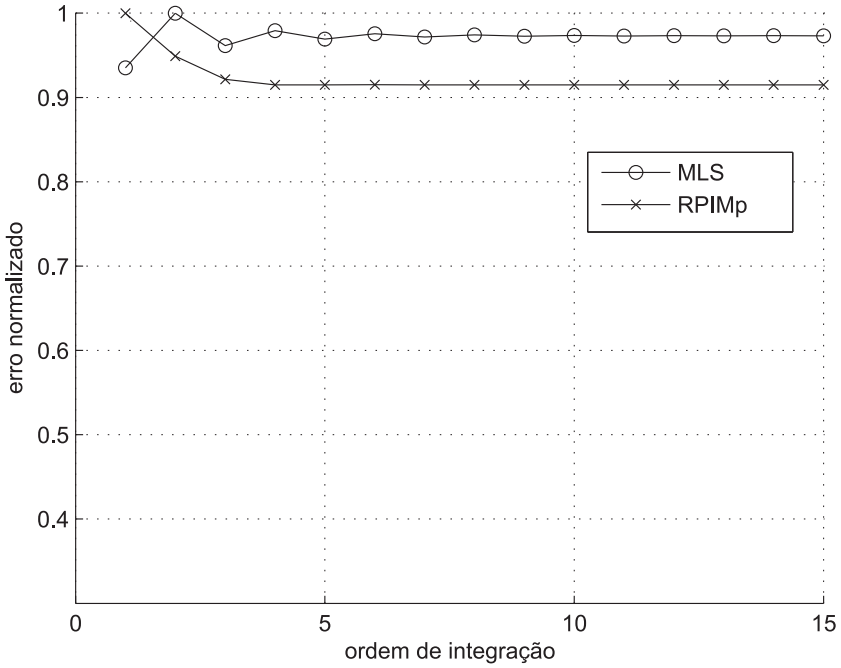


Figura 4.3: Variação do erro devido à ordem de integração. A Figura mostra o erro percentual normalizado pelo maior erro em função da ordem de integração. Os valores do domínio de quadratura e do domínio de influência foram mantidos constantes em $\alpha_q = 0,6$ e $\alpha_s = 1,6$, respectivamente. Utilizou-se um domínio de $10m \times 10m$ com uma distribuição de nós uniforme de 15×15 ($h = 0,667$).

4.2 Generalizando a geometria das regiões

O programa implementado é capaz de trabalhar com domínios quadrados. Para generalizar a geometria dos domínios é necessário realizar operações de interseção entre o domínio global e o domínio de integração local de cada nó para determinar as regiões da caixa de integração que estão fora do domínio global e descartá-las, como ilustrado na Figura 4.4.

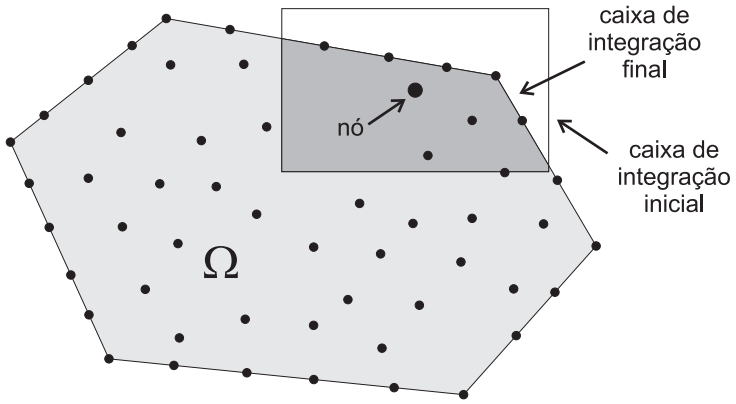


Figura 4.4: Operações de interseção para determinar caixa de integração

Para duas dimensões, adotou-se duas estruturas de dados geométricas denominadas *Nef Polygon* e *Polygon*. Essas estruturas estão disponíveis na biblioteca CGAL (CGAL, 2007).

Um *Nef Polygon* é definido como sendo qualquer conjunto que pode ser obtido a partir de operações de interseção e de complemento sobre um conjunto finito de semi-espaços. Essa estrutura foi a primeira tentativa para resolver o problema, pois apresenta uma extensão para três dimensões (*Nef Polyhedron*), o que tornaria simples a extensão do aplicativo.

Entretanto, *Nef Polygon* é uma estrutura de dados muito mais geral e complexa do que o necessário para resolver o problema descrito na Figura 4.4. O manual da CGAL indica que é necessário que o *Nef Polygon* seja parametrizado com tipos numéricos para representar as coordenadas dos pontos com precisão infinita. Testes com tipos numéricos sem precisão infinita foram feitos e falharam com geometrias simples.

Utilizando o *Polygon*, uma estrutura de dados mais simples representada por uma cadeia fechada de arestas, também disponível na CGAL, obteve-se um processamento mais rápido, pois essa estrutura de dados não necessita trabalhar com precisão infinita.

Outras soluções são possíveis e deverão ser testadas. Por exemplo, poderíamos triangular a geometria resultante quando houver interseção e fazer a integração numérica sobre os triângulos obtidos.

Se a caixa de integração estiver inteiramente dentro do domínio Ω, que é a maior parte dos casos, o resultado da interseção é a própria caixa de integração, não sendo necessário calcular a interseção. Para domínios convexos, poderia-se testar inicialmente se os vértices da caixa estão dentro do domínio, calculando as interseções somente se pelo menos um dos pontos da caixa estiver fora do domínio.

4.3 Localização de nós vizinhos

Ao calcular a integração para o subdomínio de um nó i é preciso determinar quais nós exercem influência nesse subdomínio. A Figura 4.5 mostra o nó j e seu domínio de influência. Como o domínio de influência do nó j possui interseção com o subdomínio do nó i, o nó j influência a integração de Ω_q^i.

Para determinar os nós que participam da integração de um subdomínio Ω_q^i, é necessário determinar todos os nós cujos domínios de influência possuem interseção com o subdomínio Ω_q^i. Assume-se, a princípio, que todos os nós possuem subdomínios de mesmo tamanho e que são quadrados. Basta determinar quais os nós que estão dentro de uma caixa de pesquisa de lado $2(R_\phi + R_i)$, em que R_ϕ é o raio dos domínios de influência dos nós e R_i é a metade do lado que define o subdomínio Ω_q^i, como ilustrado na Figura 4.6.

Os nós k que se encontram a uma distância superior a R_ϕ da fronteira do subdomínio, ou a uma distância $R_\phi + R_i$ do nó, não influenciam na integração, pois as funções RBF desses nós k, que servem de suporte para as funções de forma, são nulas na região de integração, bem como suas derivadas.

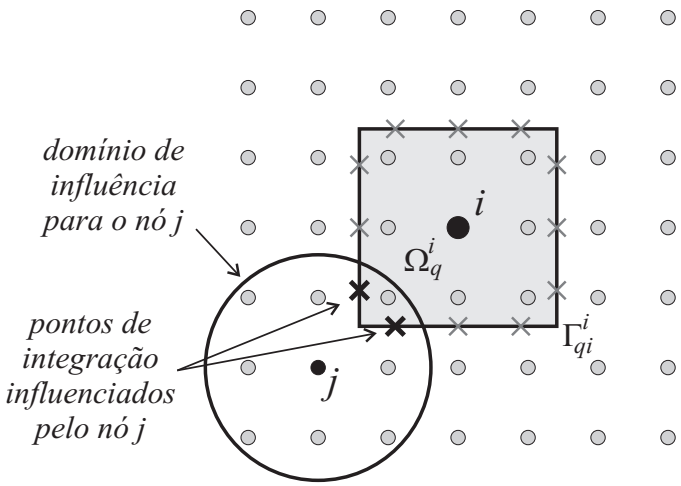


Figura 4.5: Nó j influencia o subdomínio do nó i

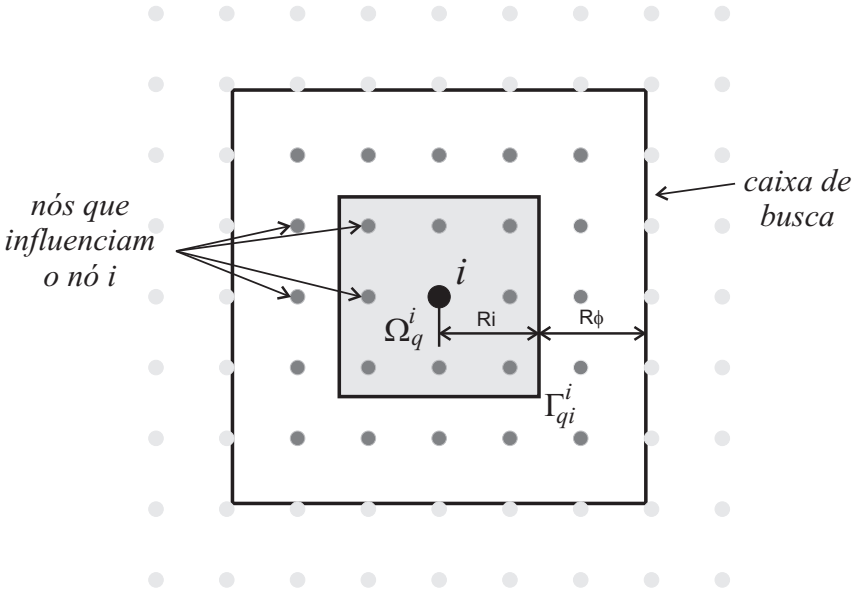


Figura 4.6: Nós que influenciam a integração do subdomínio

4.4 Construção da árvore de busca

Uma forma simples de determinar os nós vizinhos a um ponto p seria varrer todos os nós calculando a distância de p ao nó e excluindo aqueles que tenham uma distância maior que $R_\phi + R_i$. Esse algoritmo teria ordem de complexidade $O(n)$, onde n é o número de nós

distribuídos sobre o domínio. Como é necessário determinar os vizinhos para cada nó, essa busca tem que ser realizada n vezes resultando em uma ordem de complexidade total de $O(n^2)$.

Para agilizar a busca por nós vizinhos a um determinado ponto, utiliza-se uma árvore de busca denominada *kd-tree* (de Berg et al., 2000), disponível na biblioteca CGAL (CGAL, 2007). A construção da *kd-tree* possui ordem de complexidade de $O(n \log n)$ e a consulta aos nós vizinhos é da ordem de $O(\sqrt{n}+k)$, onde k é o número de vizinhos retornados. Como a construção da árvore deve ser feita uma única vez e as consultas por vizinhos devem ser feitas n vezes, a complexidade final é de $O(n \log n) + nO(\sqrt{n} + k)$ que resulta em $O(n\sqrt{n} + nk)$, reduzindo sobremaneira o tempo de execução quando o número de nós aumenta muito. A escolha pela *kd-tree* se deu pela experiência adquirida sobre essa estrutura em trabalhos anteriores. Uma solução mais eficiente seria o uso de uma *range-tree* que possui ordem de complexidade $O(\log^2 n + k)$ para busca. A *range-tree* gasta um pouco mais de memória, $O(n \log n)$, enquanto a *kd-tree* gasta O(n) (de Berg et al., 2000). A CGAL também disponibiliza uma implementação da *range-tree* e pretende-se adotá-la em versões futuras do software.

A *kd-tree* pode ser construída dividindo os conjuntos de pontos. A Figura 4.7 ilustra esse processo para duas dimensões. Cada nodo na árvore está definido em um plano por uma das dimensões das partições do conjunto de pontos em esquerda/direita (ou acima/abaixo), cada um com a metade dos pontos do nodo pai. Os filhos são divididos novamente ao meio, usando planos com dimensão diferente. As divisões param após $log(n)$ níveis, com cada ponto em sua própria folha.

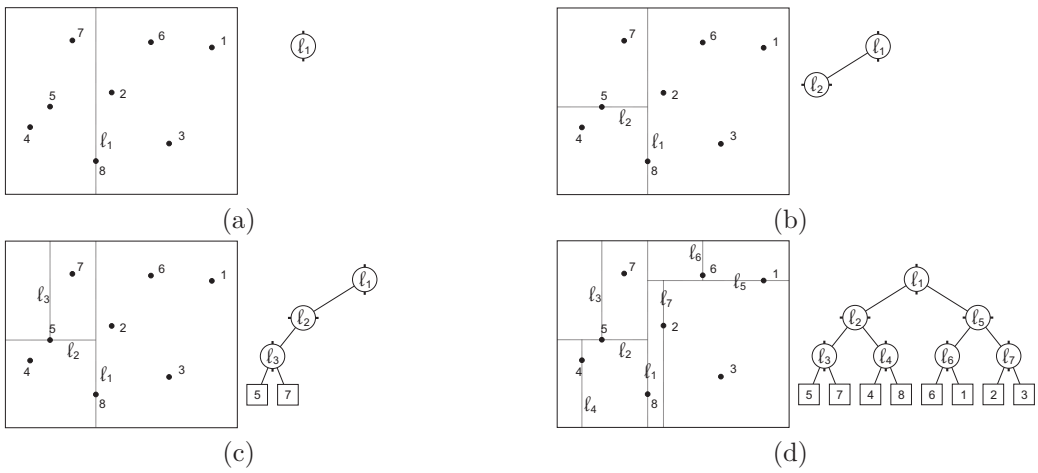
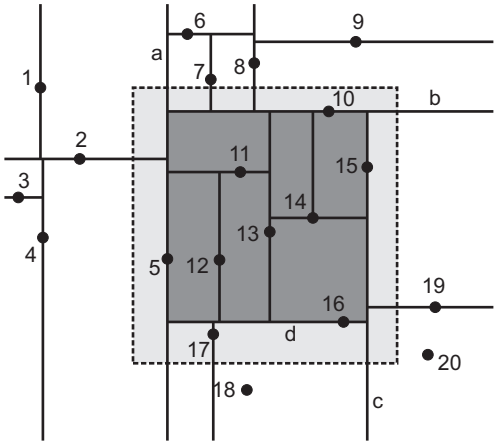
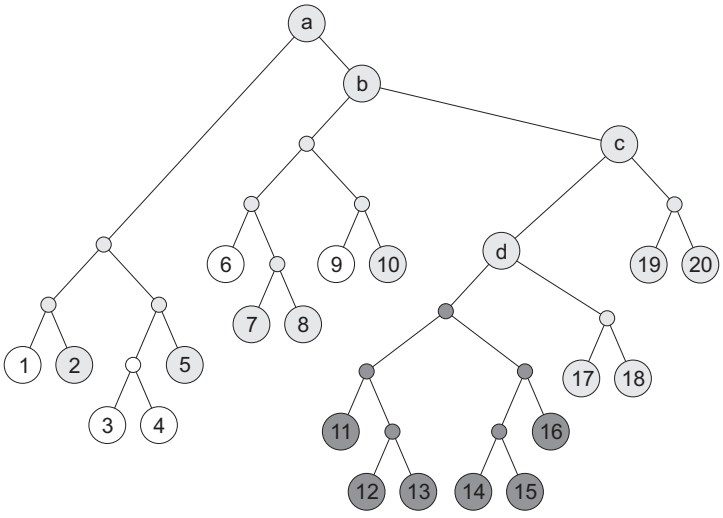


Figura 4.7: Exemplo de construção de uma *kd-tree*

Uma vez construída a árvore de busca, pode-se determinar quais os nós que estão dentro de uma caixa varrendo a árvore pelas linhas que a delimitam. No exemplo da Figura 4.8b é representada uma *kd-tree* não balanceada construída a partir da distribuição de pontos ilustrada na Figura 4.8a. No exemplo da Figura 4.8 pretende-se encontrar os pontos internos à caixa pontilhada em cinza claro. A região destacada em cinza escuro é delimitada pelas linhas a, b, c e d em destaque. Por isso, todos os pontos que pertencem à sub-árvore à esquerda do nodo definido pela linha d estarão dentro da caixa de busca e não precisam ser testados. Além desses pontos, é necessário testar pontos que estão nas regiões cortadas pelos limites da caixa de busca. No exemplo, os pontos 2, 5, 7, 8, 10, 17, 18, 19 e 20 precisam ser testados.



(a) Distribuição de pontos



(b) kd-tree

Figura 4.8: Busca retangular em uma *kd-tree*

4.5 Imposição das condições de contorno

4.5.1 Implementações clássicas

Nas implementações clássicas (Atluri and Shen, 2002; Viana et al., 2004), as funções de forma de todos os nós é determinada através de um único método. A implementação de Atluri and Shen (2002) baseia-se no método dos mínimos quadrados móveis (MLS), enquanto a implementação de Viana et al. (2004) utiliza o RPIMp.

As funções de forma obtidas via MLS não possuem a propriedade do delta de Kronecker. Logo, o valor dos coeficientes que multiplicam cada função obtida pelo MLS não irão necessariamente coincidir com os valores dos nós conhecidos. Com esse tipo de função de forma é necessário utilizar algum método, como o método das penalidades, para forçar o valor dos coeficientes nos nós localizados na fronteira com condição de contorno de Dirichlet.

Funções de forma construídas pelo RPIMp possuem a propriedade do delta de Kronecker, dispensando o uso de técnicas elaboradas para impor as condições de contorno de Dirichlet.

Porém, em testes preliminares, verificou-se que quando o número de nós que influenciam a integração de um subdomínio cresce, a utilização do RPIMp torna a montagem do sistema mais demorada se comparada com o MLS. Por isso, na próxima seção, é proposta uma solução mista, que tenta incorporar as vantagens do uso do RPIMp (maior precisão, facilidade de imposição das condições de contorno de Dirichlet) com as do uso de MLS (menor tempo de processamento).

A imposição das condições de contorno de Dirichlet é um problema em métodos sem malha quando as funções de forma não satisfazem a propriedade do delta de Kronecker. Na última década, alguns autores propuseram o uso de técnicas como multiplicadores de Lagrange, método das penalidades ou a combinar o método sem malha com o método de elementos finitos (Fernández-Méndez and Huerta, 2004).

Entretanto, o uso dessas técnicas implicam em problemas menores, como o crescimento no número de incógnitas no sistema quando se usa multiplicadores de Lagrange e a determininação de um bom valor para o parâmetro α quando se usa o método das penalidades que pode tornar o sistema mal condicionado.

4.5.2 Implementações mistas

Neste trabalho propõe-se a utilização mista de funções de forma (Fonseca et al., 2008d,c). A idéia principal é utilizar o RPIMp para gerar as funções de base para os nós próximos da fronteira onde são impostas as condições de contorno de Dirichlet, eliminando a necessidade de algum método especial como o método das penalidades, e utilizar MLS para nós distantes da fronteira. Como resultado, desenvolveu-se um método com boa precisão no contorno e com tempo de processamento intermediário entre os dois métodos.

O primeiro passo é a classificação dos nós quanto a sua proximidade com relação à fronteira que possui condição de contorno de Dirichlet. A Figura 4.9 mostra uma região onde os nós pretos foram classificados como próximos ao contorno, enquanto os nós brancos foram classificados como distantes do contorno.

Essa classificação leva em conta a caixa de procura por nós vizinhos que influenciam o cálculo da integração em um determinado nó. Se a caixa de busca tem interseção com o contorno de Dirichlet, o nó é classificado como próximo. Logo, nós próximos são aqueles que se encontram a um distância menor ou igual a R_q do contorno de Dirichlet, tal que $R_q = R_i + R_\phi$. O termo caixa de busca utilizado neste trabalho é denotado por autores como Liu (2002) como domínio de suporte. R_q neste trabalho é a metáde do lado da caixa de busca que equivale ao raio do domínio de suporte.

Nas regiões internas ao domínio, onde os nós são classificadas como distantes, não há a preocupação com a imposição de valores de contorno. Nesse caso usa-se MLS para gerar as funções de forma. Para nós próximos ao contorno usa-se RPIMp para gerar as funções de forma.

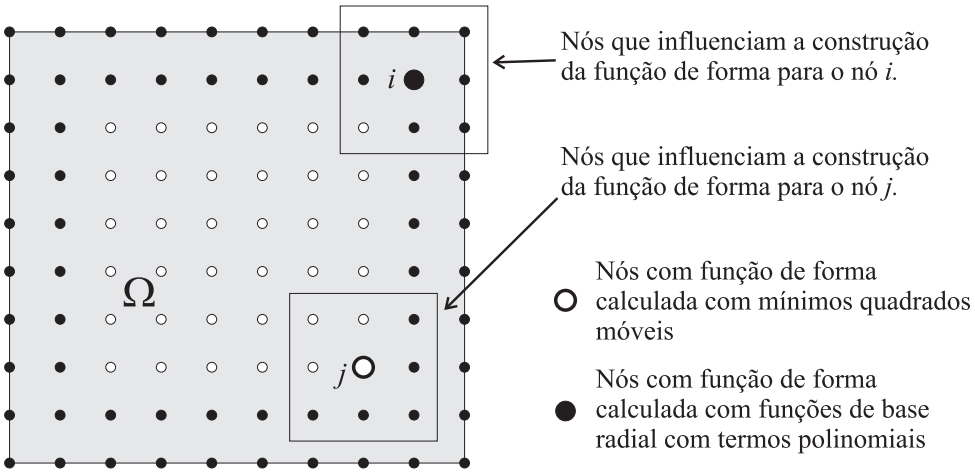


Figura 4.9: Classificação dos nós segundo a proximidade com a fronteira de Dirichlet. O nó i é classificado como próximo da fronteira pois sua caixa de busca faz intercessão com a fronteira. O nó j é classificado como nó interno.

Contudo, atenção deve ser dada à integração de nós que estão sob a influência tanto de nós internos quanto de nós próximos à borda. Duas estratégias foram aplicadas para contornar essa questão.

A primeira implementação considera um tipo fixo de função de forma para cada nó. Nós próximos ao contorno têm uma função de forma gerada pelo RPIMp. Os demais ficam com funções de forma geradas pelo MLS. Durante a integração de subdomínios influenciados pelos dois tipos de nós, dois tipos de funções de forma serão considerados como ilustrado nas Figuras 4.10 e 4.11.

A Figura 4.10 representa os nós i, j e k que influenciam o subdomínio do nó j. O nó j, que determina o subdomínio de integração, é classificado com um nó interno, assim como o nó k, mas o nó i é um nó próximo à fronteira e contribui com uma função de forma gerada pelo RPIMp.

Outra situação é mostrada na Figura 4.11 onde os nós h, i e j influenciam o subdomínio determinado pelo nó i. Nota-se que as funções de forma para os nós i e j não variam de um caso para outro, mesmo se a integração é feita no subdomínio do nó i ou do nó j. A atribuição das funções de forma depende apenas da classificação do nó.

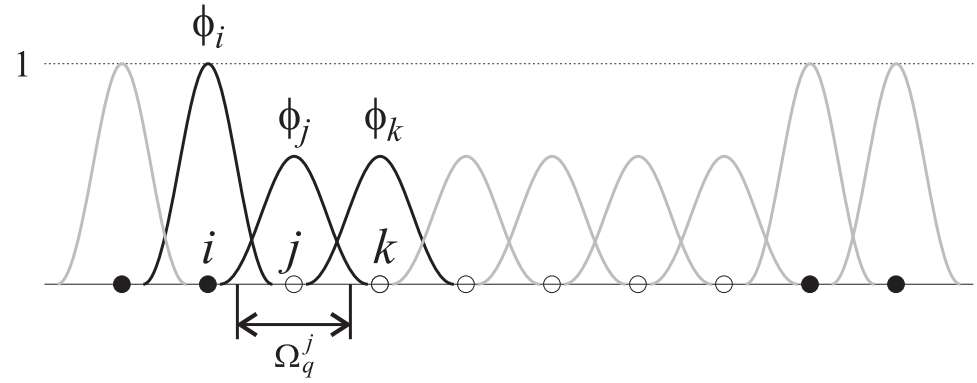


Figura 4.10: Funções de forma que influenciam o nó j - 1ª implementação. Nós pretos são nós próximos da fronteira e nós brancos são nós internos

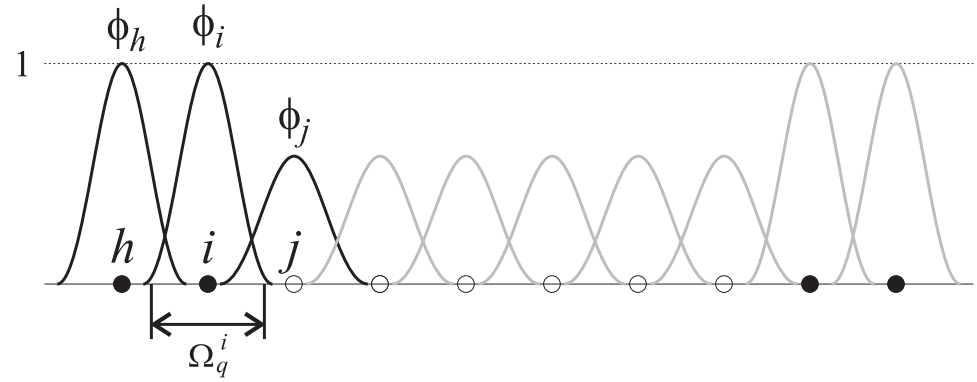


Figura 4.11: Funções de forma que influenciam o nó i - 1ª implementação. Nós pretos são nós próximos da fronteira e nós brancos são nós internos

Na segunda implementação, um nó não possui um tipo fixo de função de forma associada. Sua função de forma pode variar dependendo da classificação do nó em que o subdomínio está sendo integrado.

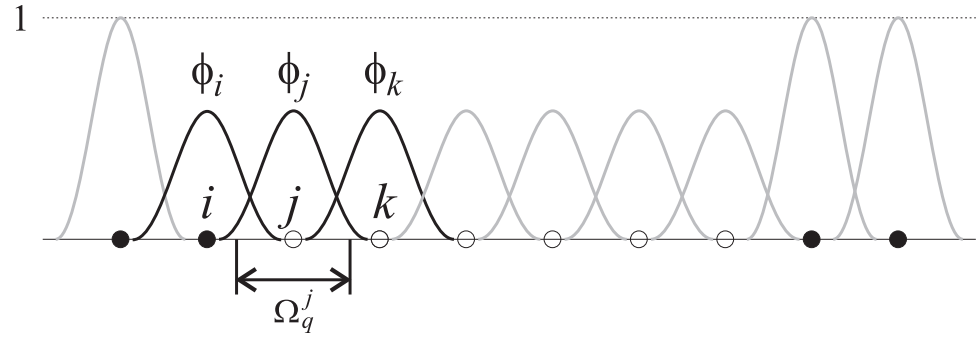


Figura 4.12: Funções de forma que influenciam o nó j - 2ª implementação. Nós pretos são nós próximos da fronteira e nós brancos são nós internos

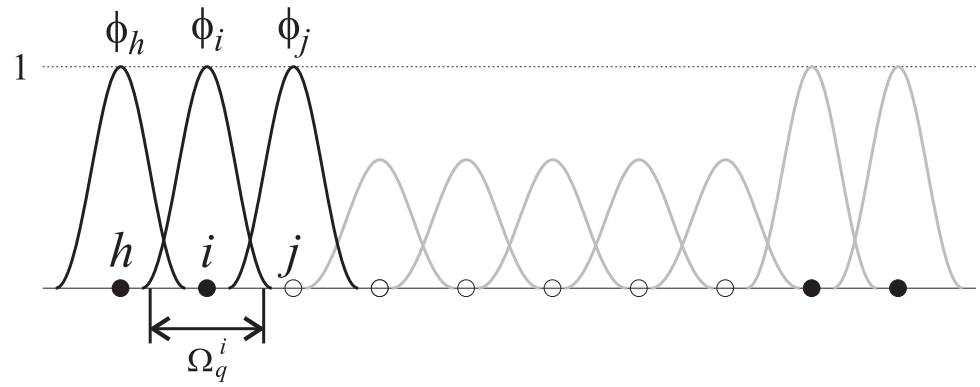


Figura 4.13: Funções de forma que influenciam o nó i - 2ª implementação. Nós pretos são nós próximos da fronteira e nós brancos são nós internos

Na Figura 4.12 a integração é feita no subdomínio definido pelo nó j. Como esse é um nó interno, todos os nós que influenciam sua integração possuirão funções de forma utilizando MLS. Na Figura 4.13 a integração é feita em um subdomínio de um nó próximo à fronteira, determinando que os nós envolvidos na integração possuirão função de forma utilizando RBF's com termos polinomiais. Nesse caso, a atribuição das funções de forma não depende da classificação do próprio nó mas sim da classificação do nó que define o subdomínio de integração.

Para testar a implementação, utilizamos o problema da calha definido na Figura 4.14, onde é aplicada uma tensão V_+ na tampa superior e V_- nos outros lados. Apesar do problema ser bastante simples, os pontos nos cantos superiores apresentam descontinuidade o que representa uma grande dificuldade para qualquer método numérico.

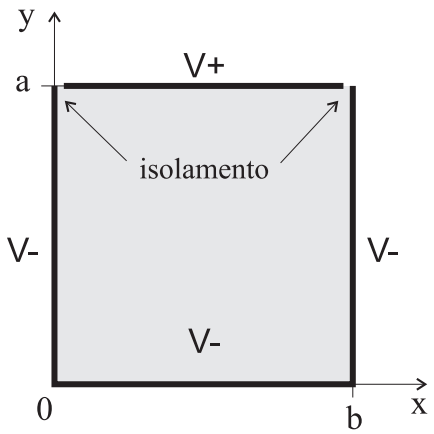


Figura 4.14: Problema da calha

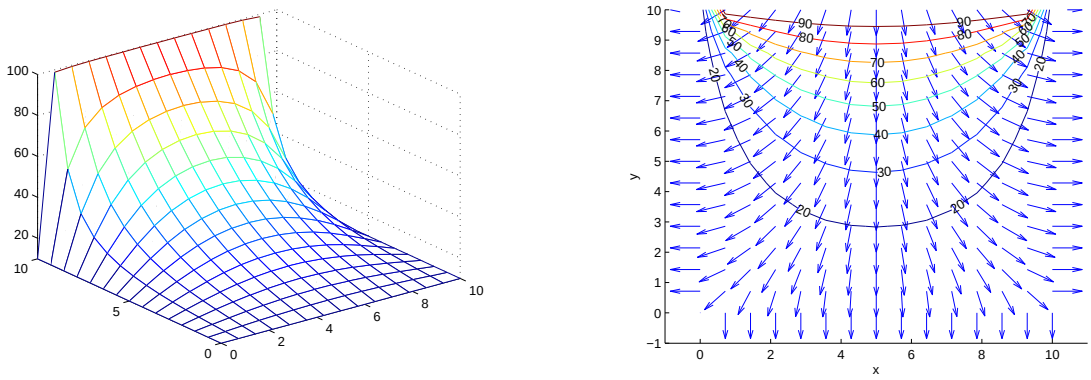
A solução analítica para o problema da calha é mostrada nas Figuras 4.15a e 4.15b. Para esse problema é considerado um valor de $100V$ para a tensão da placa superior V_+ e $10V$ para V_-. A solução analítica pode ser calculada por (Sadiku, 2004):

$$V(x,y) = \frac{4(V_+ - V_-)}{\pi} \sum_{n=1,3,5,\ldots}^{\infty} \frac{\operatorname{sen}\left(\frac{n\pi x}{b}\right) \operatorname{senh}\left(\frac{n\pi y}{b}\right)}{n \operatorname{senh}\left(\frac{n\pi a}{b}\right)} + V_- \tag{4.1}$$

As derivadas parciais para o problema da calha são dadas por

$$\frac{\partial V(x,y)}{\partial x} = \frac{4(V_+ - V_-)}{b} \sum_{n=1,3,5,\ldots}^{\infty} \frac{\cos\left(\frac{n\pi x}{b}\right) \operatorname{senh}\left(\frac{n\pi y}{b}\right)}{\operatorname{senh}\left(\frac{n\pi a}{b}\right)} \tag{4.2}$$

$$\frac{\partial V(x,y)}{\partial y} = \frac{4(V_+ - V_-)}{b} \sum_{n=1,3,5,\ldots}^{\infty} \frac{\operatorname{sen}\left(\frac{n\pi x}{b}\right) \cosh\left(\frac{n\pi y}{b}\right)}{\operatorname{senh}\left(\frac{n\pi a}{b}\right)} \tag{4.3}$$

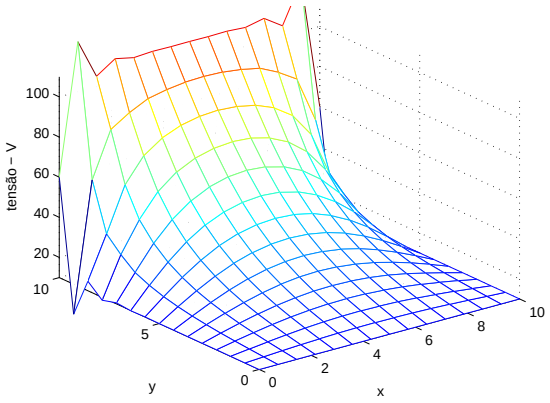


(a) Potencial

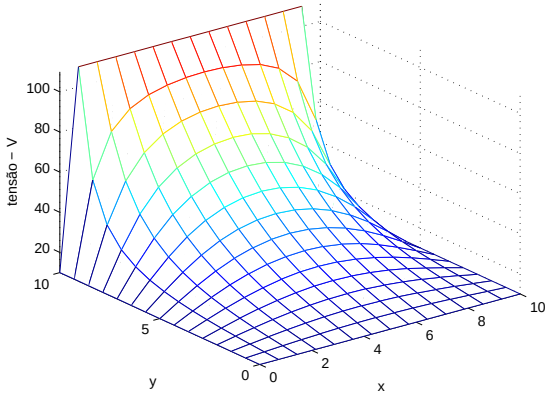
(b) Equipotenciais e campo elétrico

Figura 4.15: Solução para o problema da calha. (a) Superfície definida pelo potencial. (b) Equipotenciais e Campo elétrico normalizado.

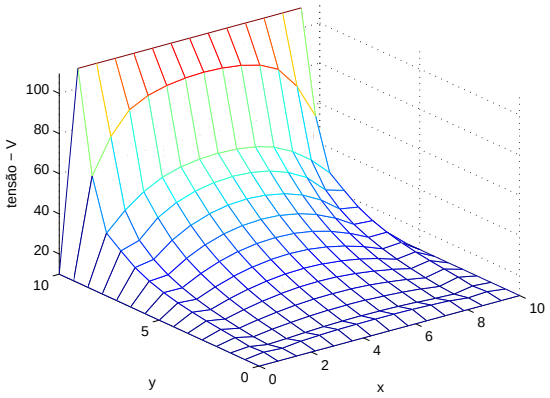
As Figuras 4.16 e 4.17 mostram a solução para o problema da calha usando os métodos estudados. A Figura 4.18 mostra a superfície do erro percentual ponto a ponto para cada método. Nessas superfícies, foi desconsiderada a região próxima aos cantos superiores, onde o problema apresenta descontinuidade. Na Figura 4.16a fica claro que a solução usando apenas funções geradas pelo MLS apresenta problemas no contorno e a Figura 4.18c mostra que a primeira implementação para o método misto apresenta grande erro sobre todo o domínio.



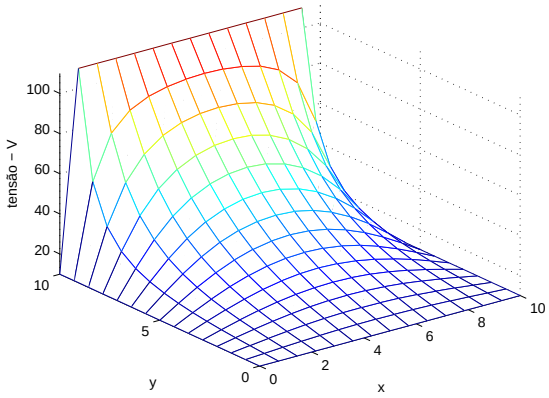
(a) MLS



(b) RPIMp

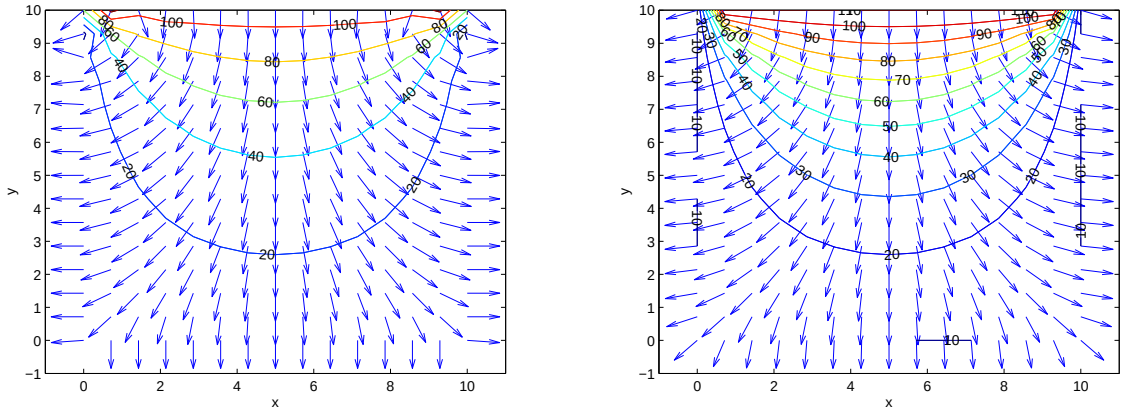


(c) Misto 1



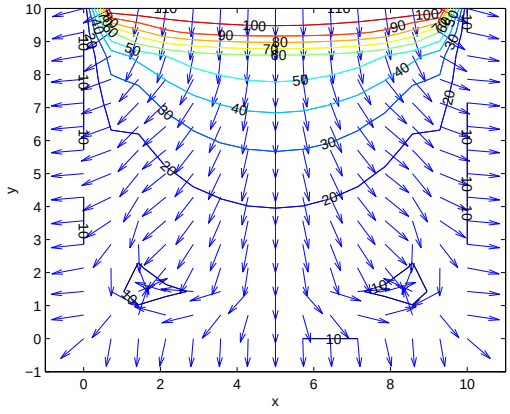
(d) Misto 2

Figura 4.16: Superfícies de tensão geradas pelos métodos para o problema da calha. (a) Usando apenas funções de forma geradas com MLS. Nesse caso a solução apresenta oscilações na borda (método da penalidade) (b) Usando apenas funções de forma geradas com RPIMp. Não apresenta oscilações (função de forma possui propriedade do delta de Kronecker) (c) Usando a primeira implementação do método misto. A área de transição entre as funções de forma apresenta um degrau. (d) Usando a segunda implementação do método misto. Resultado similar ao RPIMp. Parâmetros utilizados: $h = 0,667$, $\alpha_s = 0,75$ e $\alpha_q = 0,75$, ordem de integração = 4.

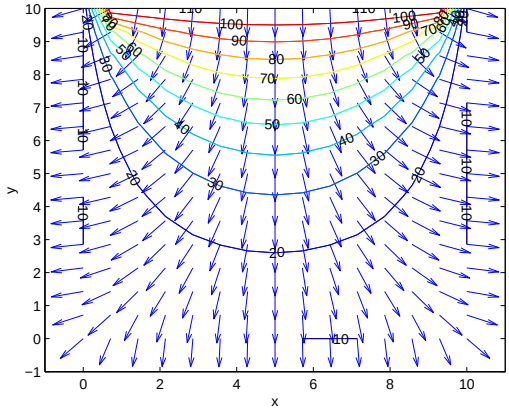


(a) MLS

(b) RPIMp

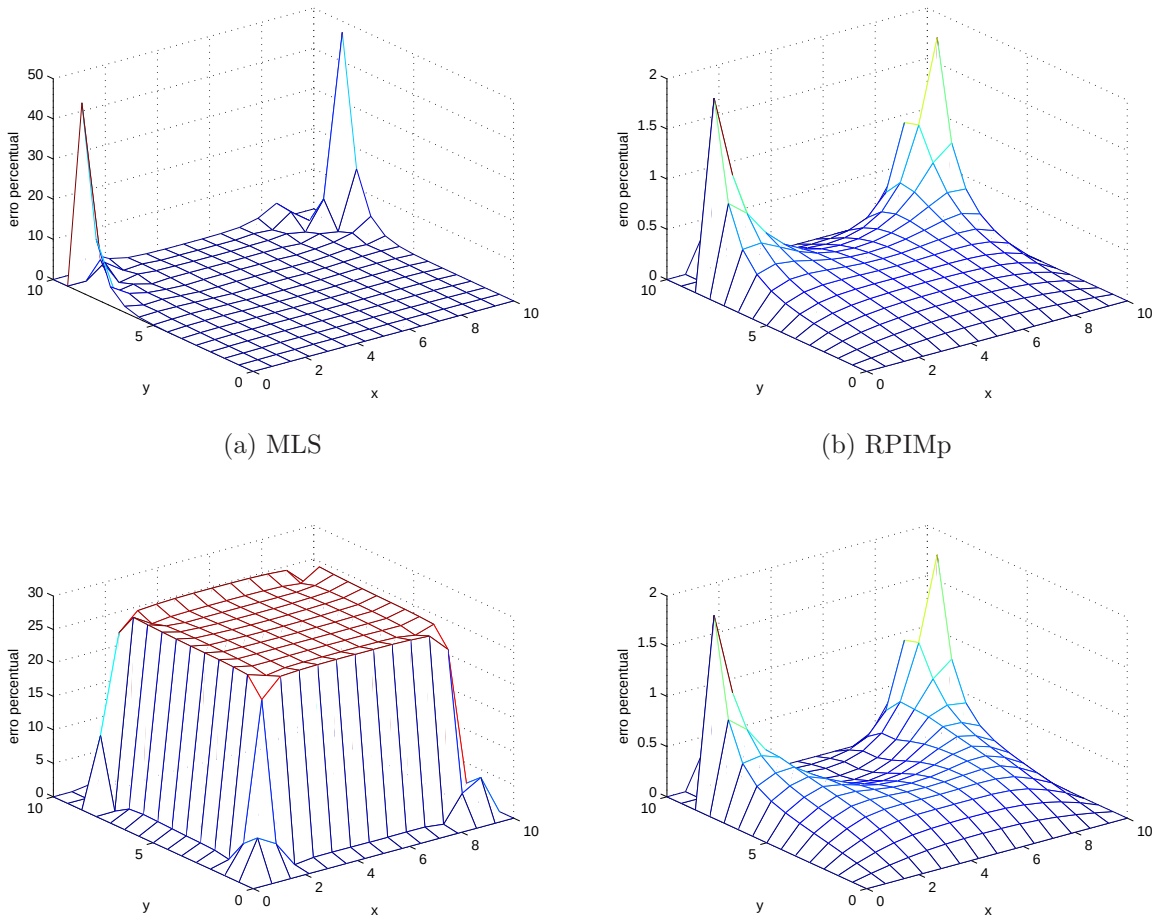


(c) Misto 1



(d) Misto 2

Figura 4.17: Equipotenciais e campo elétrico para as soluções geradas da Figura 4.16. (a) Usando apenas funções de forma geradas com MLS. (b) Usando apenas funções de forma geradas com RPIMp. (c) Usando a primeira implementação do método misto. (d) Usando a segunda implementação do método misto. Parâmetros utilizados: $h = 0,667$, $\alpha_s = 0,75$ e $\alpha_q = 0,75$, ordem de integração = 4.



(a) MLS

(b) RPIMp

(c) Misto 1

(d) Misto 2

Figura 4.18: Superfícies de erro percentual para o valor de tensão V para as soluções geradas da Figura 4.16. (a) Usando apenas funções de forma geradas com MLS. (b) Usando apenas funções de forma geradas com RPIMp. (c) Usando a primeira implementação do método misto. (d) Usando a segunda implementação do método misto. Parâmetros utilizados: $h = 0,667$, $\alpha_s = 0,75$, $\alpha_q = 0,75$ e ordem de integração $= 4$.

A Figura 4.19 mostra o problema da calha em L também utilizado para testar as implementações apresentadas. Trata-se de um problema com gradiente de potencial crescente ao longo da diagonal $\overline{AB}$.

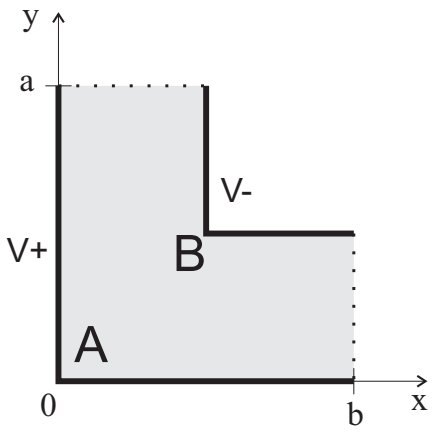
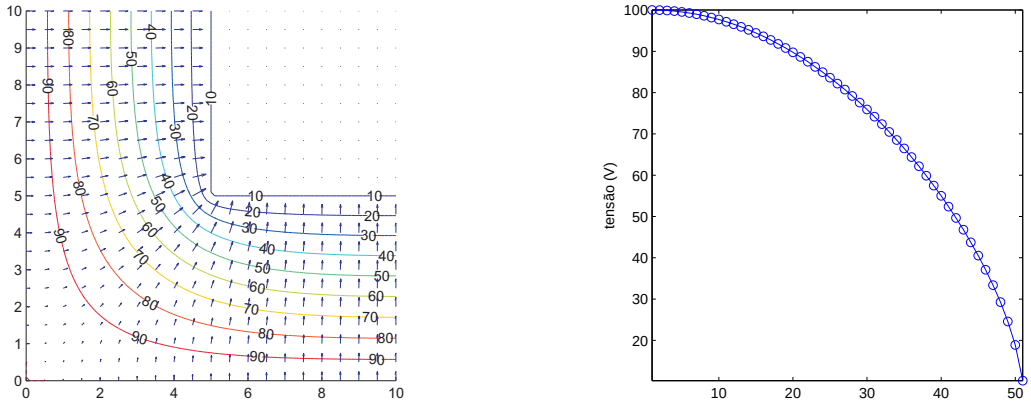


Figura 4.19: Calha em L

A Figura 4.20 mostra o resultado obtido utilizando o FEM. Para o problema resolvido foi considerado $V_+ = 100V$, $V_- = 10V$, $a = b = 10m$ e em $x = b$ e $y = a$ considera-se a condição de Neumann $\partial V/\partial \mathbf{n} = 0$. A malha utilizada foi gerada com uma distância mínima de $h = 0,5m$ entre os vértices.



(a) (b)

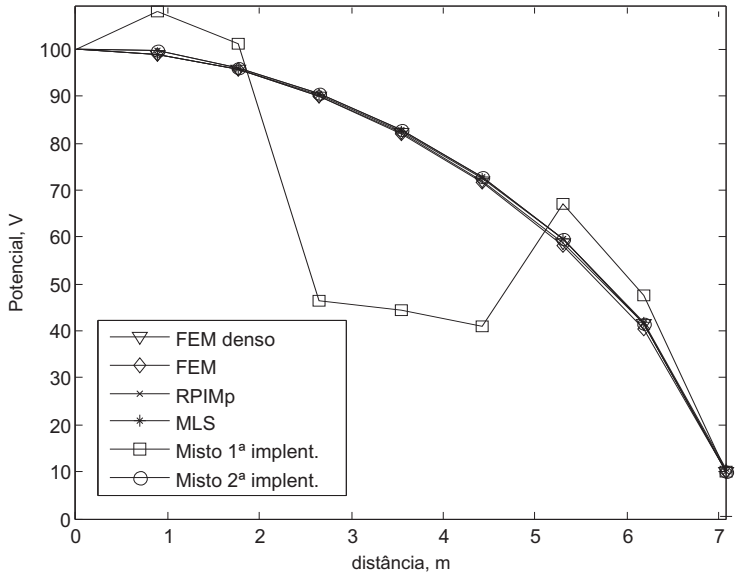
Figura 4.20: Resultado para a calha em L utilizando FEM. (a) Equipotenciais e campo elétrico. (b) Tensão sobre o segmento $\overline{AB}$ definido na Figura 4.19

A Figura 4.21 mostra o teste realizado utilizando o problema da calha em L. O mesmo problema foi resolvido com o MLPG utilizando funções de forma construídas com MLS, RPIMp e pelo método misto. Para o método misto são consideradas as duas implementações apresentadas. Os nós utilizados pelo MLPG são os mesmos vértices da malha utilizada no método de elementos finitos.

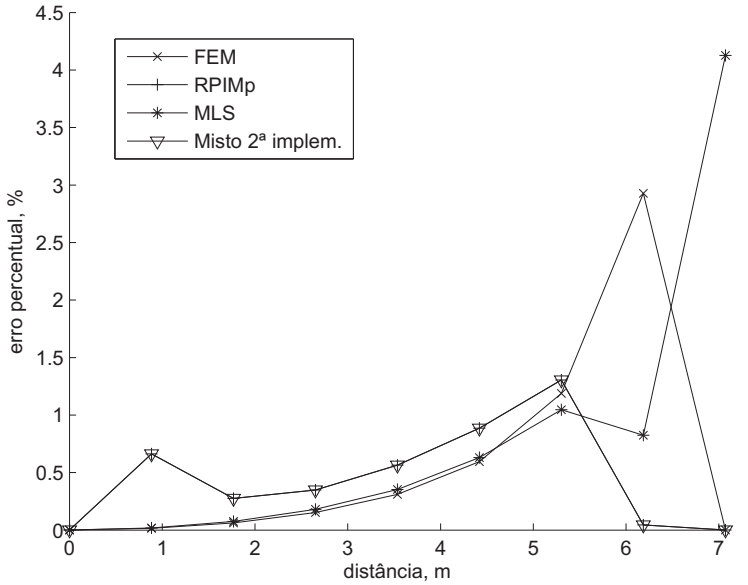
Para comparar os resultados dos métodos, gerou-se uma solução utilizando o FEM com uma malha refinada com $h = 0,05m$, identificado nos gráficos como FEM denso. Essa solução pode ser considerada como uma boa aproximação da solução exata.

Na Figura 4.21a é mostrado o valor de tensão encontrado sobre a diagonal que une os pontos A e B da Figura 4.19. Nota-se que todos os métodos resultaram em soluções muito próximas sendo necessário observar os gráficos de erros dados pela Figura 4.21b para avaliar as diferenças entre os métodos.

A Figura 4.21b mostra o erro percentual ponto a ponto para cada um dos métodos quando comparados ao FEM com malha densa. Nota-se que o MLPG utilizando MLS apresenta uma melhor aproximação no interior do domínio, sendo bem próximo ao FEM com malha menos densa. Entretanto, tanto o FEM quanto o MLPG com MLS apresentam maior dificuldade de aproximar a função nas proximidades do ponto B devido à descontinuidade física que existe neste ponto. Exatamente sobre o ponto B, o MLPG com MLS apresenta um erro significativo, enquanto os demais métodos não apresentam erro já que a condição de contorno é diretamente imposta.



(a) Solução sobre o segmento AB



(b) Erro percentual

Figura 4.21: Problema da calha em L. (a) Comparação dos resultados obtidos pelos métodos. A primeira implementação do método misto apresenta um degrau na região de transição entre funções de forma. Os demais métodos acompanham a curva de tensão. (b) Erro percentual (a primeira implementação do método misto é desconsiderada). Parâmetros utilizados pelos métodos sem malha: $h = 0,625$, $\alpha_s = 0,6$, $\alpha_q = 1,5$, ordem de integração = 4. O MLPG utilizou os vértices da malha do FEM como os nós distribídos sobre o domínio.

Utilizando o primeiro método misto nota-se que existe uma nítida descontinuidade na região de transição entre os nós próximos do contorno para os nós internos ao domínio para todos os problemas testados. Com o segundo método isso não ocorre e a precisão alcançada é similar ao método utilizando apenas RPIMp.

Como discutido no Capítulo 2, funções de forma devem satisfazer o critério da partição da unidade (Equação 2.9). Quando mesclamos funções de forma distintas na integração de um mesmo subdomínio, como foi feito na primeira implementação do método misto, esse critério deixa de ser respeitado e o método não funciona corretamente como mostram os resultados.

Na segunda implementação, a partição da unidade é respeitada, pois apenas um método de interpolação é utilizado para a integração de um mesmo subdomínio. Subdomínios diferentes podem utilizar métodos de interpolação diferentes já que cada subdomínio é independente no MLPG.

Para se ter uma ideia do custo computacional para problemas práticos, onde o número de nós internos é muito maior que o número de nós na fronteira, gerou-se o problema da calha com uma grade regular de 101×101 nós. A Tabela 4.2 mostra o tempo relativo para a montagem do sistema linear. Note-se que quando o número de nós (n) que influenciam a integração de um subdomínio cresce, o método misto torna-se mais atrativo pois possui tempo de processamento intermediário aos métodos tradicionais e mantém boa precisão.

Tabela 4.2: Comparação de tempo de processamento dos métodos. n é o número de nós que influenciam a integração de um subdomínio

	4 pontos de integração			20 pontos de integração		
n	MLS	RPIMp	MLS+RPIMp 2	MLS	RPIMp	MLS+RPIMp 2
16	1,13	1,00	1,10	3,05	1,48	2,86
25	1,32	1,46	1,30	3,77	2,54	3,53
36	1,51	2,35	1,63	4,57	4,30	4,52
49	1,70	4,32	2,33	5,65	7,91	6,15
64	2,00	7,28	3,06	6,84	13,20	8,05

Os resultados indicam que a primeira implementação não funciona corretamente. A segunda implementação respeita o critério da partição da unidade localmente para cada subdomínio e os resultados encontrados possuem precisão similar a quando se utiliza RPIMp porém com um tempo de processamento menor.

4.5.3 Aprimorando a implementação mista

No método misto proposto (Fonseca et al., 2008d,c), utilizamos funções de forma baseados no RPIMp para nós próximos à borda e MLS para nós internos ao domínio. Como as funções RPIMp possuem a propriedade do delta de Kronecker, dispensamos qualquer método especial para forçar as condições de contorno de Dirichlet.

Dessa forma, obtivemos um método mais rápido comparado ao método clássico quando usamos apenas funções de forma RPIMp sobre o domínio inteiro e o número de nós vizinhos é maior que 16. Entretanto, para a maioria dos casos, apenas nove nós vizinhos são suficientes para obter bons resultados. Nesse caso, a formulação mista será atrativa se as funções de forma internas puderem ser calculadas mais rapidamente que as funções de forma utilizando o MLS.

A solução para tornar o método misto mais atrativo é utilizar uma função de forma muito simples e rápida para os nós internos. No caso, utilizamos as funções de Shepard que, como dito no Capítulo 2, são um subcaso do MLS onde o polinômio base se reduz a $p = 1$. Isso faz com que as funções de Shepard fiquem muito simples de implementar e com baixo custo computacional, já que não são necessárias inversões de matrizes. Entretanto, pagamos com uma baixa ordem de consistência (ordem zero). Implementações que usam apenas funções de Shepard apresentam problemas no contorno (assim como o MLS, porém com maior intensidade) quando as condições de contorno não são bem impostas, como no caso das Figuras 4.23a e 4.23b, onde usamos o método da penalidade.

Além de usar uma função de forma mais simples para o interior do domínio, percebemos que poderíamos utilizar funções de forma com a propriedade do delta de Kronecker apenas para os nós sobre o contorno de Dirichlet dispensando um esforço computacional de pré-classificação dos nós. Dessa forma, passamos a maximizar a região interna, aumentando os nós com processamento mais rápido (Figura 4.22).

A Figura 4.23 apresenta o resultado obtido para o problema da calha utilizando o método misto melhorado. A Figura 4.23a mostra a superfície de tensão obtida utilizado apenas funções de forma Shepard. O mesmo resultado é apresentado em curvas de nível na Figura 4.23b. Utilizando apenas funções de Shepard com o método de penalidade para forçar as condições

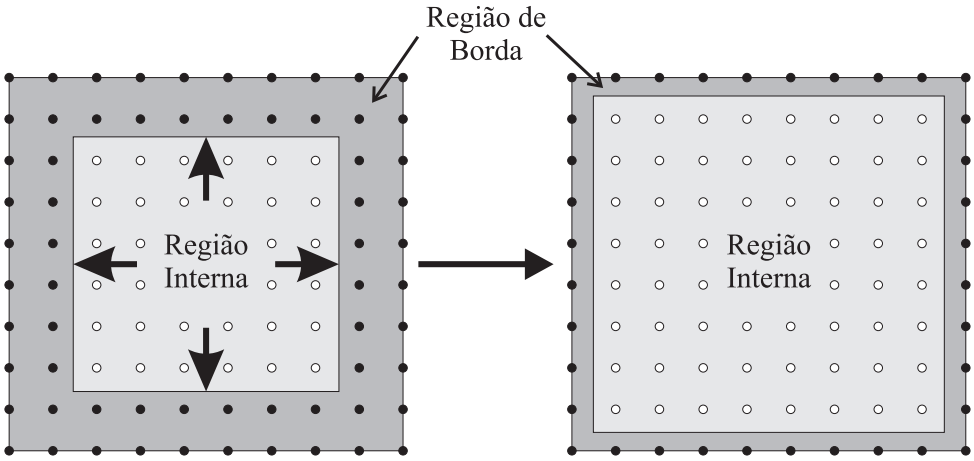
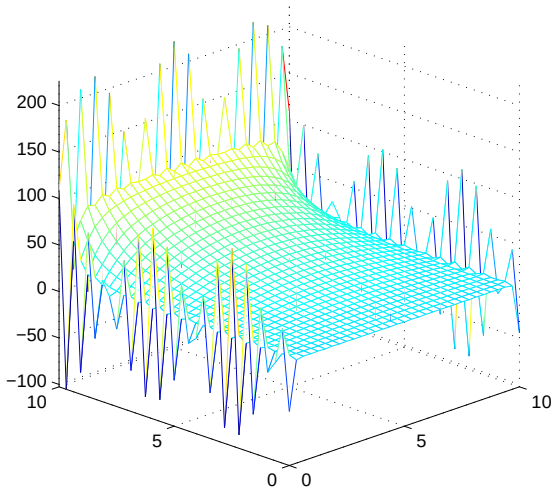


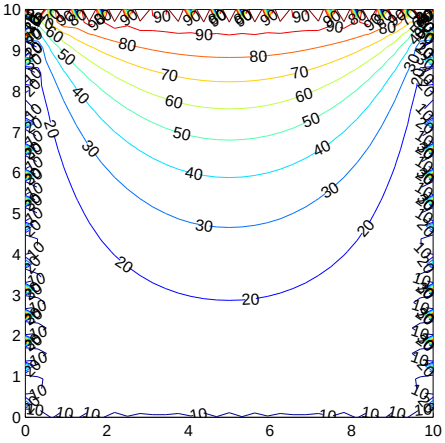
Figura 4.22: Maximizando região interna. Somente os pontos sobre a fronteira de Dirichlet precisam de funções de forma com a propriedade do delta de Kronecker.

de contorno obteve-se um resultado com muita oscilação nas bordas.† Utilizando o método misto associando funções de forma RPIMp para a fronteira de Dirichlet e funções de Shepard apenas para o interior do domínio, nos fornecem uma solução muito melhor, sem oscilações na borda e com um tempo de processamento muito menor do que comparado ao método utilizando apenas RPIMp.

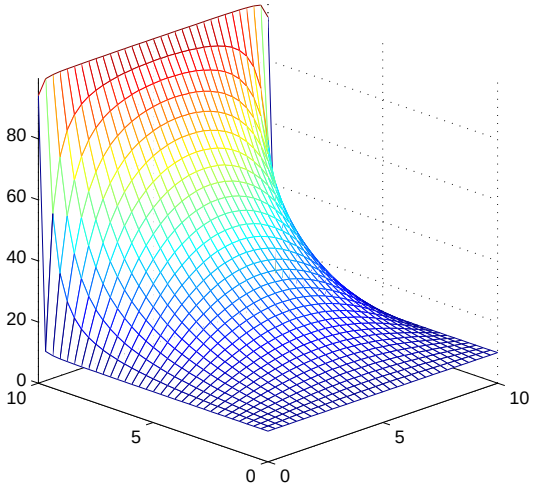
† Os resultados da Figura 4.23 são apresentados no artigo (Fonseca et al., 2010). Entretanto, na época havia um erro de implementação que fez com que as oscilações da borda se propagassem pelo interior do domínio. Com a solução do problema (que é desconhecido) as oscilações se limitam ao contorno do problema.



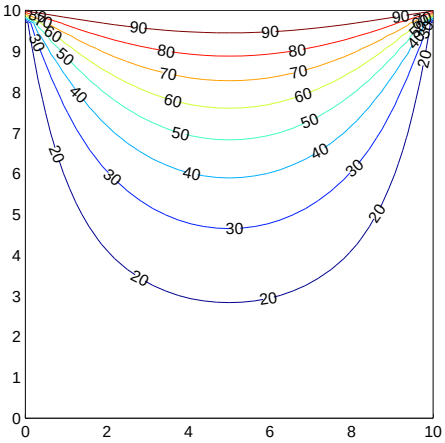
(a) Shepard com penalidade



(b) Shepard com penalidade



(c) Misto



(d) Misto

Figura 4.23: Resultados para o método misto melhorado. (a) e (b) Solução para o problema da calha utilizando funções de Shepard com o método da penalidade. (c) e (d) Solução para o problema da calha utilizando o método misto melhorado combinando funções RPIMp para a borda e funções de Shepard para o interior.

A Tabela 4.3 compara os tempos relativos entre o método misto melhorado e usando apenas as funções de forma Shepard e RPIMp.

Tabela 4.3: Comparação de tempo de processamento dos métodos. n é o número de nós que influenciam a integração de um subdomínio

n	9	16
RPIMp	$1,94$	$7,54$
Shepard	$1,00$	$1,21$
Misto (RPIMp+Shepard)	$1,07$	$2,15$

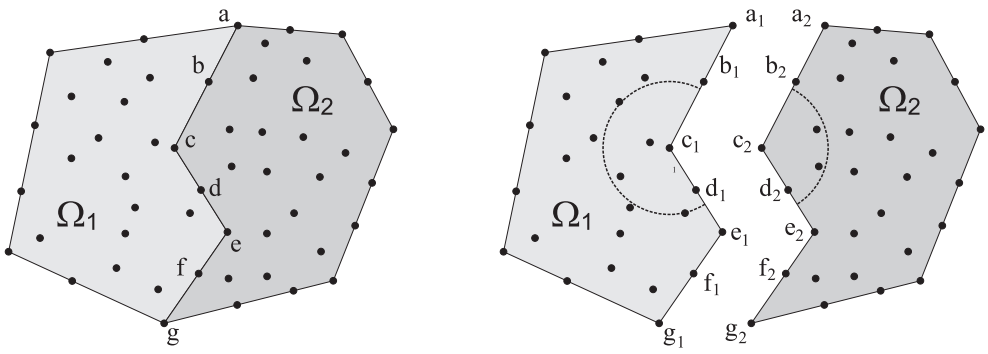
Os resultados obtidos para o método misto mostram que quando usamos apenas as funções de Shepard e o método de penalidade para impor as condições de contorno, ocorrem oscilações na borda que comprometem a qualidade da solução. Entretanto, ao se combinar as funções de Shepard com as funções RPIMp, obtemos um método com precisão aceitável e com um processamento muito próximo ao de se usar apenas funções de Shepard, mesmo quando o número de nós vizinhos utilizados na integração dos subdomínios é reduzido.

4.6 Tratando múltiplos materiais

Até o momento, abordamos apenas o problema com domínios com um único material. Em problemas que envolvem múltiplos materiais surge o problema de como tratar a descontinuidade na interface entre eles. Para isso, usamos o método da visibilidade descrito em (Viana, 2006).

Uma das estratégias sugeridas por Viana (2006) (método da visibilidade) para tratar problemas com vários materiais é considerar que os pontos distribuídos ao longo da interface são pontos duplos, ou seja, existem dois pontos sobrepostos mas um em cada região. A Figura 4.24 mostra esse tipo de situação. A Figura 4.24a mostra o problema original com os pontos sobre a interface sobrepostos (a, b, c, d, e, f e g). Na Figura 4.24b, as mesmas regiões aparecem disjuntas e cada ponto aparece nas duas regiões. Como exemplo, o ponto c_1 é o ponto c na região Ω_1 e c_2 é o ponto c na região Ω_2. Note-se que a região de influência de cada ponto não extrapola o seu domínio de origem.

Na verdade, os nós sobre a interface são únicos, mas têm seu subdomínio repartido em duas partes como mostra a Figura 4.25. Na Figura 4.25, o nó i sobre a fronteira de interface Γ_I



(a) (b)

Figura 4.24: Fronteira entre materiais para o método da visibilidade. Nós sobre a fronteira são considerados nós duplos.

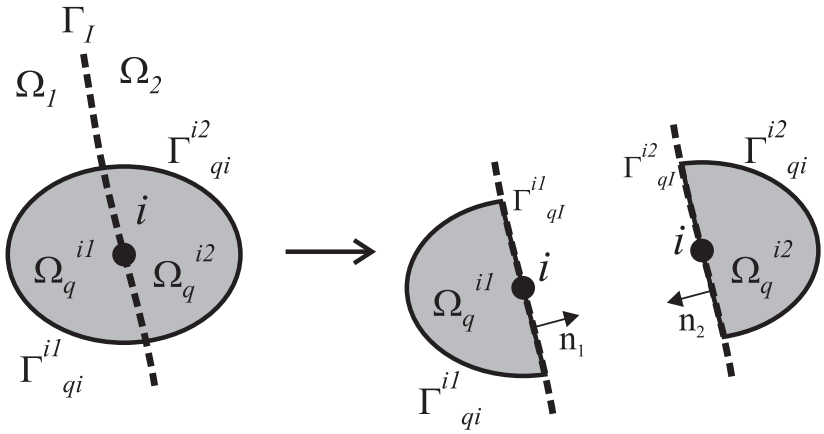


Figura 4.25: Nó i sobre a interface Γ_I entre as regiões Ω_1 e Ω_2. O subdomínio do nó Ω_q^i é dado pela união entre Ω_q^{i1} em Ω_1 e Ω_q^{i2} em Ω_2

entre as regiões Ω_1 e Ω_2, tem que satisfazer a forma fraca local definida por 3.15 tanto em Ω_1 quanto em Ω_2.

Assumindo que a função de peso W é a função de Heaviside, a forma fraca (Equação 3.15) para o nó i da Figura 4.25 pode ser simplificada para

$$\int_{\Gamma^i_{qu}\cup\Gamma^i_{qi}} (W_i k\nabla u)\cdot\mathbf{n}d\Gamma + \int_{\Gamma^i_{qt}} W_i h d\Gamma - \int_{\Omega^i_q} W_i f d\Omega + \alpha\int_{\Gamma^i_{qu}} W_i\,(u-g)\,d\Gamma = 0 \qquad (4.4)$$

Para a porção do subdomínio de i em Ω_1, temos:

$$\int_{\Gamma_{qu}^{i1}\cup\Gamma_{qi}^{i1}} (W_i k_1 \nabla u)\cdot \mathbf{n}_1 d\Gamma + \int_{\Gamma_{qI}^{i1}} (W_i k_1 \nabla u)\cdot \mathbf{n}_1 d\Gamma + \int_{\Gamma_{qt}^{i1}} W_i h d\Gamma - \int_{\Omega_q^{i1}} W_i f d\Omega + \alpha \int_{\Gamma_{qu}^{i1}} W_i (u-g)\, d\Gamma = 0 \quad (4.5)$$

e para a porção do subdomínio de i em Ω_2, temos:

$$\int_{\Gamma_{qu}^{i2}\cup\Gamma_{qi}^{i2}} (W_i k_2 \nabla u)\cdot \mathbf{n}_2 d\Gamma + \int_{\Gamma_{qI}^{i2}} (W_i k_2 \nabla u)\cdot \mathbf{n}_2 d\Gamma + \int_{\Gamma_{qt}^{i2}} W_i h d\Gamma - \int_{\Omega_q^{i2}} W_i f d\Omega + \alpha \int_{\Gamma_{qu}^{i2}} W_i (u-g)\, d\Gamma = 0 \quad (4.6)$$

Somando as Equações 4.5 e 4.6 obtemos a contribuição para o nó i. Note que na interface $\mathbf{n}_1 = -\mathbf{n}_2$ e portanto os termos $\int_{\Gamma_{qI}^{i1}} (W_i k_1 \nabla u)\cdot \mathbf{n}_1 d\Gamma$ e $\int_{\Gamma_{qI}^{i2}} (W_i k_2 \nabla u)\cdot \mathbf{n}_2 d\Gamma$ se anulam, já que pela condição de contorno de interface temos $k_1 \dfrac{\partial u_1}{\partial n} = k_2 \dfrac{\partial u_2}{\partial n}$. Assim, temos:

$$\int_{\Gamma_{qu}^{i1}\cup\Gamma_{qi}^{i1}} (W_i k_1 \nabla u)\cdot \mathbf{n}_1 d\Gamma + \int_{\Gamma_{qu}^{i2}\cup\Gamma_{qi}^{i2}} (W_i k_2 \nabla u)\cdot \mathbf{n}_2 d\Gamma + \int_{\Gamma_{qt}^{i1}\cup\Gamma_{qt}^{i2}} W_i h d\Gamma - \int_{\Omega_q^{i1}\cup\Omega_q^{i2}} W_i f d\Omega + \alpha \int_{\Gamma_{qu}^{i1}\cup\Gamma_{qu}^{i2}} W_i (u-g)\, d\Gamma = 0 \quad (4.7)$$

Nós em que o subdomínio possui interseção com a interface entre dois meios, como ilustrado na Figura 4.26 têm o seu subdomínio truncado para permanecer apenas na região na qual o nó está inserido.

Pode haver casos onde o nó está sobre a fronteira de mais de duas regiões, como ilustra a Figura 4.27. Nesse caso, a contribuição do nó continua sendo de uma única equação dada pelo somatório das contribuições em cada material. Para s materiais, temos:

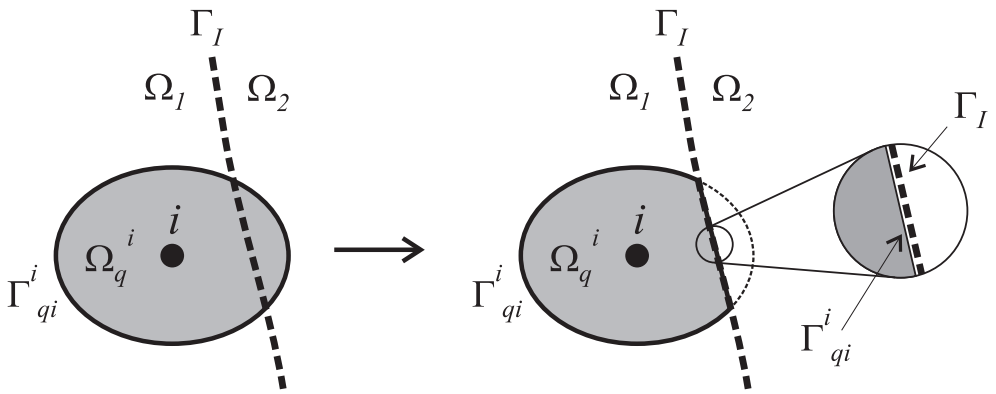


Figura 4.26: Nó i próximo à interface Γ_I entre as regiões Ω_1 e Ω_2. O subdomínio do nó Ω_q^i é truncado pela interface de forma a ficar totalmente contido no interior de Ω_1.

$$\sum_{r=1}^{s}\left[\int_{\Gamma_{qu}^{ir}\cup\Gamma_{qi}^{ir}}(W_i k_r \nabla u)\cdot\mathbf{n}_r d\Gamma+\int_{\Gamma_{qt}^{ir}} W_i h d\Gamma - \int_{\Omega_q^{ir}} W_i f d\Omega+\alpha\int_{\Gamma_{qu}^{ir}} W_i\,(u-g)\,d\Gamma\right]=0 \quad (4.8)$$

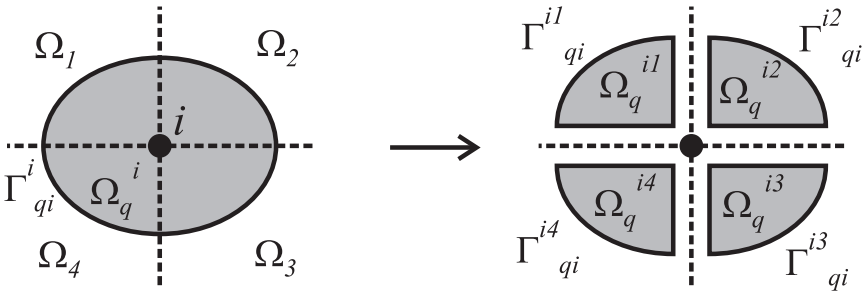


Figura 4.27: Nó i sobre a interface entre as regiões Ω_1, Ω_2, Ω_3 e Ω_4.

Neste trabalho testamos apenas o método da visibilidade. Outra estratégia descrita por Viana (2006), e também implementada em (Nicomedes et al., 2011), é o método da colocação. Nesse método, não fazemos a integração de subdomínios para nós na interface entre materiais mas forçamos que os valores nodais nas várias regiões sejam iguais. No método da visibilidade, nós na fronteira entre materiais geram apenas uma equação no sistema matricial global. Já no método de colocação, um nó que faz divisa com vários materiais contribui com mais de uma equação no sistema. Mais detalhes em (Viana, 2006).

Durante o processo de integração das regiões, cada nó precisa conhecer os nós vizinhos mais próximos que influenciam o cálculo das funções de forma nos pontos de integração. Como

cada nó influencia apenas na sub-região na qual está inserido, para problemas com várias regiões, como ilustrado na Figura 4.28, cria-se uma árvore de pesquisa independente para cada sub-região.

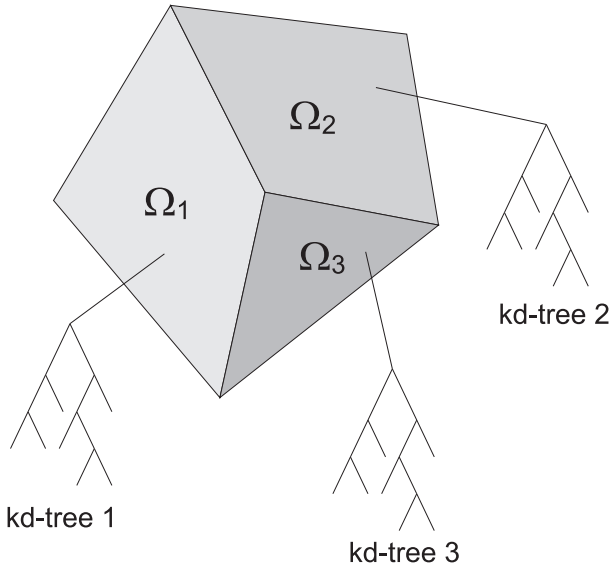
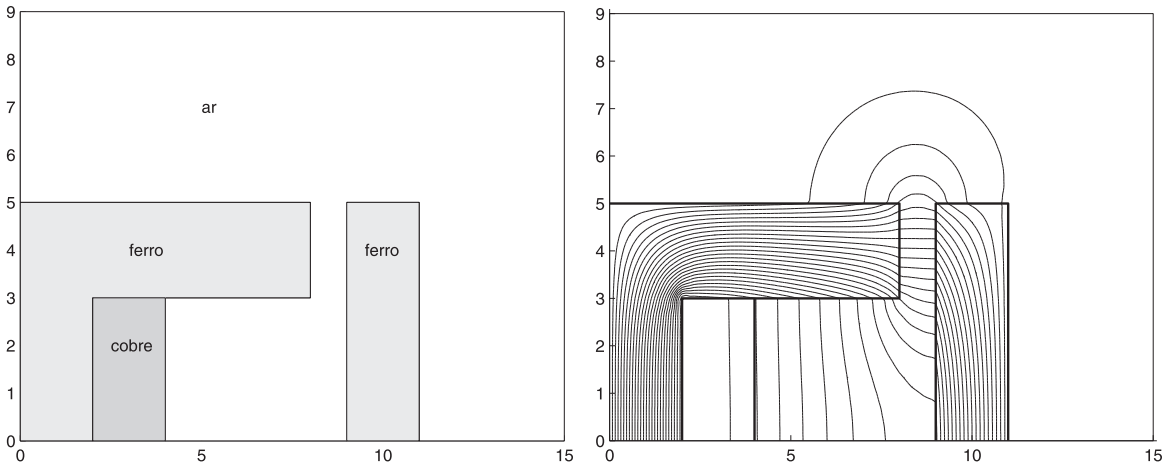


Figura 4.28: Árvores de busca para problemas com mais de uma região

Para se criar uma árvore de pesquisa para cada região, é necessário se conhecer a priori a que região pertence cada nó. Essa informação é gerada no pré-processamento durante a distribuição de nós sobre o domínio. Pontos sobre a interface entre dois (ou mais) materiais pertencem simultaneamente às duas regiões. Dessa forma, esses pontos são inseridos nas árvores de pesquisa das duas regiões.

Para testar a implementação, usamos o problema do eletroímã. Esse problema é um pouco mais complicado comparado aos resolvidos anteriormente pois apresenta três materiais diferentes (ar, ferro e cobre) e várias interfaces entre eles. A Figura 4.29a mostra a definição geométrica do problema. A Figura 4.29b mostra a solução para o problema. Os vários métodos foram testados e a Figura 4.30 mostra a convergência dos métodos. Considera-se como uma aproximação da solução real para o problema o resultado de uma simulação usando elementos finitos com uma malha bastante densa.



(a) Definição do problema (b) Solução para o problema do eletro-imã

Figura 4.29: Problema do eletroimã. (a) Definição da geometria do problema (dimensões em cm). Os valores de permeabilidade magnética usados foram $\mu_{ar} = \mu_{cobre} = 1$, $\mu_{ferro} = 1000$. A densidade de corrente aplicada sobre o cobre é $J_z = 10^{-1} MA/m^2$. (b) Solução obtida utilizando o método misto (RPIMp+MLS) com os parâmetros $\alpha_q = 0,5$, $\alpha_s = 1,5$ e ordem de integração igual a 4.

Pelo gráfico da Figura 4.30 notamos que a convergência do método misto segue a convergência da função de forma interna utilizada. Usando as funções de Shepard, a solução converge mais lentamente, mas com um custo computacional menor.

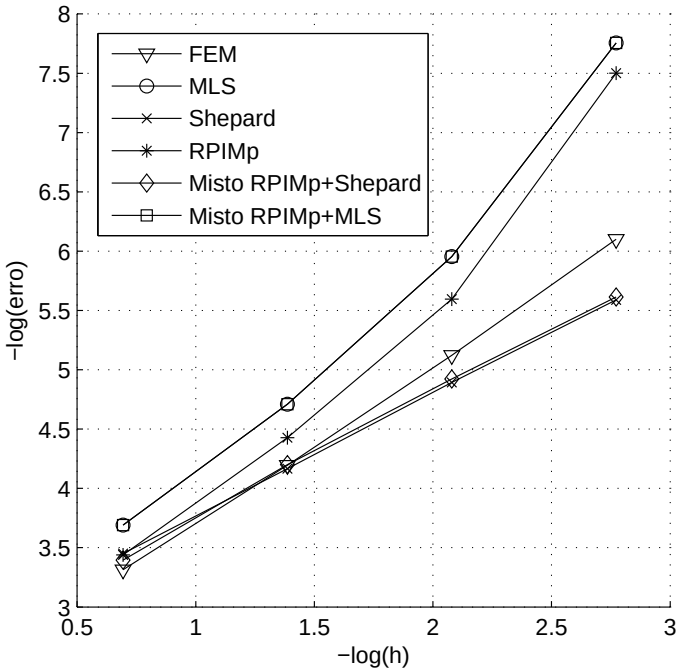


Figura 4.30: Convergência para o problema do eletroima

4.7 Paralelização dos métodos

Em todas as implementações realizadas, gera-se um sistema linear do tipo $\mathbf{Kx} = \mathbf{f}$ a partir da Equação 3.20. A integração de um subdomínio definido por um nó i contribui apenas para a i-ésima linha da matriz $\mathbf{K}$ e para a i-ésima posição do vetor $\mathbf{f}$ (Figura 4.31).

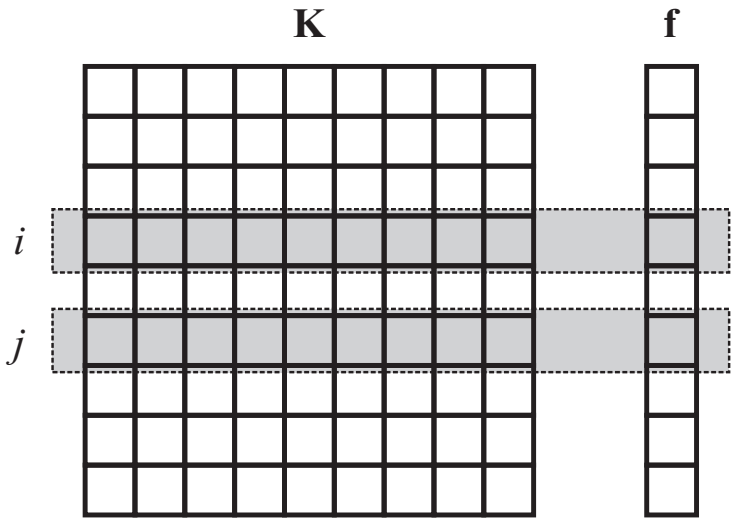


Figura 4.31: Contribuição dos nós para a montagem do sistema

A integração de diferentes nós geram contribuições em partes diferentes da matriz e independem umas das outras. Dessa forma, a integração desses nós pode ser feita de forma independente e em qualquer ordem temporal dispensando inclusive mecanismos de sincronização como *mutexes* ou semáforos.

Logo, a paralelização dos métodos se torna uma maneira natural de torná-los mais rápidos, visto que máquinas com múltiplos núcleos de processamento têm se tornado cada vez mais acessíveis (Bischof et al., 2007; Ikuno et al., 2008; Fonseca et al., 2008b,a). A Figura 4.32 mostra a estrutura do programa implementado no qual o processo de montagem do sistema linear é dividido em n *threads*, ou linhas de execução.

Entre as alternativas para a construção de implementações paralelas destacam-se a OpenMP (Bischof et al., 2007), que é uma API (Application Program Interface), e a Biblioteca Boost Thread (BOOST, 2007).

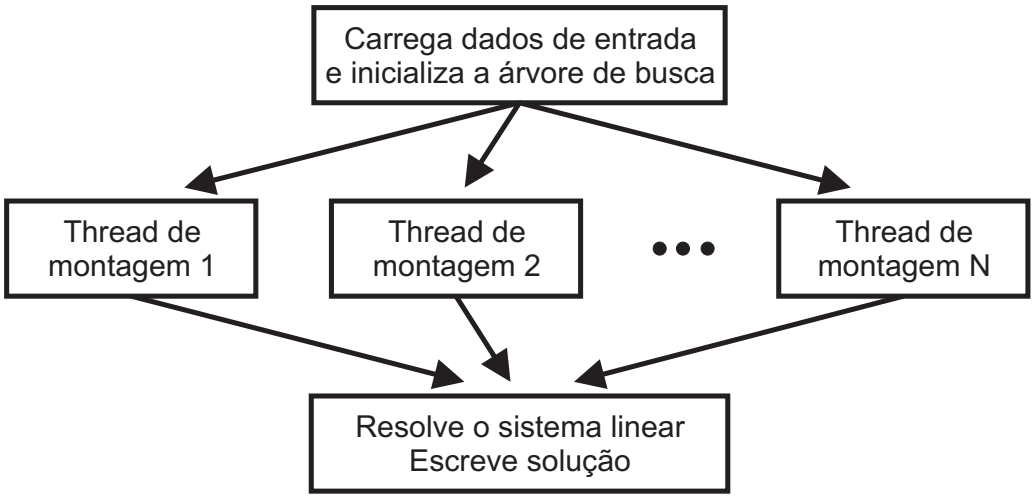


Figura 4.32: Montagem paralela do sistema

A OpenMP usa diretivas do pré-compilador para indicar onde o código deve ser paralelizado. Todo o gerenciamento de quais variáveis devem ser duplicadas e quais os limites de iterações de cada *thread* é feito pela biblioteca, o que permite a paralelização de rotinas quase sem nenhuma intervenção no código. Apesar disso, nas implementações realizadas nesse trabalho, optou-se pela Boost Thread. O mecanismo da Boost Thread consiste em encapsular em classes os métodos que serão disparados em linhas de processamento. A opção pela Boost Thread se deu pelo fato de existir uma maior familiaridade com a biblioteca Boost que já é utilizada inclusive pela biblioteca CGAL. Pretende-se gerar uma versão paralela utilizando a OpenMP para comparar com o desempenho obtido pela Boost.

Para evitar implementações ineficientes, a distribuição de nós entre as *threads* deve ser feita com cuidado. Algumas estratégias simples foram consideradas: a primeira, e mais imediata, simplesmente divide os nós em grupos a partir de seus indexadores. Para um processador com n núcleos e m nós, os nós indexados por 0 a $m/n-1$ serão processados pela primeira *thread*, os nós de m/n a $m/n\times 2-1$ serão processados pela segunda thread e assim sucessivamente. A Figura 4.33 exemplifica essa estratégia dividindo nós igualmente espaçados e sequencialmente indexados entre dois núcleos.

Uma primeira análise da Figura 4.33 talvez indique que essa estratégia é suficiente para se fazer uma igual distribuição de carga entre as *threads*. Entretanto, testes preliminares mostraram que isso não é verdade e a razão para isso é óbvia: nós que estão na fronteira possuem menos vizinhos do que nós interiores ao domínio. Para o exemplo, se o número de processadores

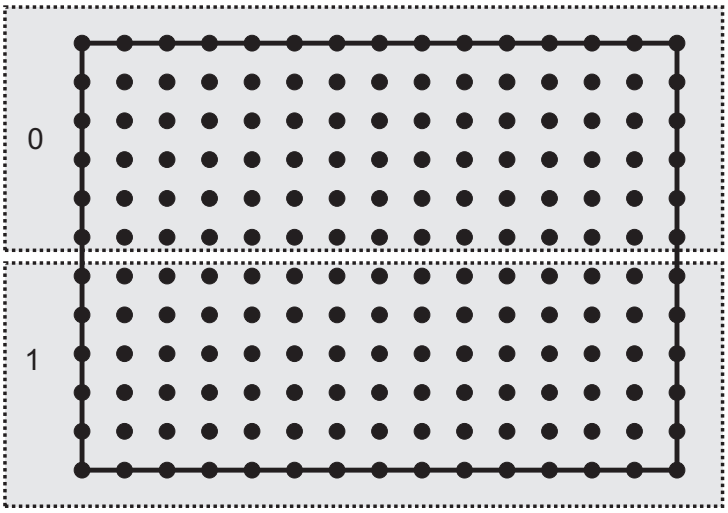


Figura 4.33: Distribuição sequencial

fosse igual a quatro, as *threads* 1 e 2 teriam uma carga maior que as *threads* 0 e 3 já que as últimas processariam mais nós próximos à fronteira. Ou seja, os núcleos 0 e 3 ficariam ociosos aguardando o término das *threads* 1 e 2.

Uma alternativa para a distribuição sequencial é distribuir os nós de maneira alternada entre os núcleos. Todos os nós cujo resto da divisão do seu indexador pelo número de núcleos é zero são processadas pelo primeiro núcleo. Para resto um, os nós são processados pelo segundo núcleo e assim sucessivamente. Para um exemplo com nós igualmente espaçados e sequencialmente indexados obtém-se uma distribuição perfeitamente alternadas dos nós entre os núcleos como mostra a Figura 4.34. Essa é a solução atualmente adotada nas implementações deste trabalho.

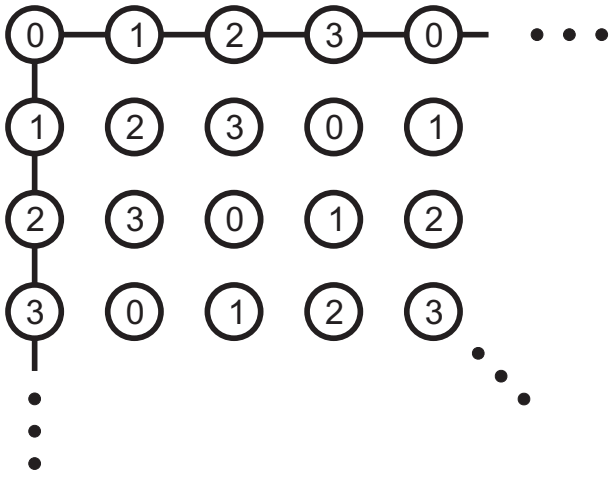


Figura 4.34: Distribuição alternada

Se a indexação dos nós não for sequencial, a distribuição alternada pode não garantir uma distribuição de carga uniforme entre os núcleos. Uma estratégia mais geral seria uma distribuição aleatória dos nós, como mostra a Figura 4.35.

Figura 4.35: Distribuição aleatória

Para testar a eficiência das implementações paralelas foram usados computadores comuns com dois e quatro núcleos de processamento. A Tabela 4.4 mostra os ganhos de desempenho obtidos. A segunda coluna da tabela mostra o ganho de desempenho relativo apenas para montagem do sistema linear. Como as demais partes do programa ainda não foram paralelizadas, o ganho total de performance é menor, como mostrado na terceira coluna da tabela.

Tabela 4.4: Desempenho das implementações paralelas

Número de Núcleos	Montagem	Total
1	1	1
2	1,96	1,65
4	3,78	2,85

Como visto na Tabela 4.4, o processo de montagem do sistema linear nas versões paralelas executaram até $1,96$ vezes mais rapidamente em processadores com dois núcleos e até $3,78$ vezes em computadores com quatro núcleos quando comparados às versões sequenciais.

4.8 Solução do Sistema Linear

Outra dificuldade do método é que a matriz do sistema linear gerado não é simétrica ou definida positiva, o que reduz o número de métodos possíveis para gerar a solução. Além disso, o condicionamento da matriz pode ser ruim, principalmente quando se usa o método das penalidades, pois esse introduz elementos de valor elevado.

Entre os métodos iterativos, tentou-se utilizar o *Preconditioned BiConjugate Gradient method* (BICG) e o *Generalized Minimal Residual method* (GMRES) (Barrett et al., 1994), disponibilizados pela ITL (*Iterative Template Library*) (ITL, 2007). O BICG não convergiu para a maioria dos problemas; o GMRES converge muito lentamente, levando até horas para resolver um problema com um número relativamente pequeno de nós A solução atualmente adotada é a utilização da biblioteca UMFPACK (Davis, 2004a,b; Davis and Duff, 1999). Essa biblioteca utiliza um método direto para a solução do sistema (decomposição LU) pré-ordenando as colunas do sistema original para minimizar o número de inserções de valores não nulos. Sistemas que levavam horas para serem determinados usando o GMRES foram resolvidos em segundos utilizando a UMFPACK.

Seja o sistema $\mathbf{Ax} = \mathbf{b}$, no qual $\mathbf{A}$ é uma matriz esparsa e não simétrica. A UMFPACK fatoriza a matriz $\mathbf{PAQ}$ ou $\mathbf{PRAQ}$ no produto $\mathbf{LU}$, no qual $\mathbf{L}$ e $\mathbf{U}$ são matrizes triangulares inferior e superior, respectivamente. $\mathbf{P}$ e $\mathbf{Q}$ são matrizes de permutação e $\mathbf{R}$ é uma matriz diagonal com fatores de escala para as linhas. $\mathbf{P}$ e $\mathbf{Q}$ são escolhidas de forma a reduzir o número de inserções de novos elementos (não nulos). As permutações de $\mathbf{P}$ tem um papel duplo: reduzir o numero de inserções e manter a precisão numérica, utilizando pivotação parcial com relaxação.

A quantidade de inserções de elementos não nulos em $\mathbf{L}$ e $\mathbf{U}$ que não correspondem a elementos na matriz original **A** é determinada pelas matrizes de permutação $\mathbf{P}$ e $\mathbf{Q}$. Achar a melhor permutação é um problema NP-completo. A biblioteca utiliza uma de três heurísticas para determinar essas permutações. A escolha da heurística é feita após uma análise da matriz **A** (Davis, 2004a).

A biblioteca UMFPACK utiliza a biblioteca BLAS (BLAS, 2008) para implementar suas subrotinas. A BLAS possui uma versão paralelizada, a PBLAS que pode ser utilizada para acelerar ainda mais a solução do sistema linear.

Capítulo 5

Conclusões

Métodos sem malha têm ganho espaço como alternativas para métodos tradicionais como o método de elementos finitos e diferenças finitas. Nesse trabalho, explorou-se o MLPG por sua flexibilidade, visto que a técnica se baseia no método de Petrov-Galerkin que permite que funções de teste e de forma pertençam a espaços de funções diferentes. Além disso, por usar uma formulação local para a forma fraca permitem uma grande independência entre subdomínios que podem assumir formas e tamanhos diferentes, dispensando grades auxiliares para integração caracterizando-o como um método "verdadeiramente sem malha".

A principal desvantagem de métodos sem malha quando comparados a métodos tradicionais é o elevado custo computacional. Nesse aspecto, buscamos utilizar bibliotecas eficientes como a CGAL para cálculos geométricos e busca de nós vizinhos e a UMFPACK para a resolução do sistema matricial. Realizamos testes e a partir dos resultados (Tabela 4.1) verificamos que o processo de montagem do sistema linear é a etapa que mais consome tempo computacional e por isso dedicamos a ela maior atenção.

O custo computacional para a montagem do sistema depende diretamente do tipo de função de forma que escolhemos. O processo será tão mais rápido quanto mais simples for a função de forma escolhida. Entre as funções de forma estudadas, a que apresenta melhor desempenho são as funções de Shepard (caso mais simples do MLS com consistência C^0). Apesar do bom desempenho computacional, as funções de Shepard, assim como o MLS, não possuem a

propriedade do delta de Kronecker exigindo alguma técnica especial para impor as condições de contorno de Dirichlet. Nesse trabalho utilizamos o método de penalidades e percebemos que ao usá-lo o método perde em precisão apresentando oscilações próximo à fronteira.

O método misto proposto permitiu que duas funções de forma fossem usadas ao mesmo tempo: uma função de forma com a propriedade do delta de Kronecker (RPIMp) sobre os nós da fronteira de Dirichlet e outra com baixo custo computacional para os nós internos ao domínio. Dessa forma, o método misto impõe de maneira direta as condições de contorno evitando perda de precisão e ganha em eficiência pois utiliza uma função de forma simples de ser calculada na maior parte dos nós que são os nós internos. Trabalhamos inicialmente com o MLS para os nós internos e obtivemos os resultados mostrados na Tabela 4.2 que nos mostra que o método misto é interessante quando construímos as funções de forma a partir de 16 nós vizinhos e utilizamos muitos pontos de integração. Para melhorar o método, substituímos o MLS pelas funções de Shepard que possuem custo computacional bem menor, pois evita cálculos matriciais complexos. Os resultados obtidos (Tabela 4.3) mostraram que mesmo com um número pequeno de vizinhos (9), os resultados apresentam boa precisão e custo computacional próximo ao das funções de Shepard. No caso, o método misto foi 48% mais eficiente do que quando usamos apenas o RPIMp e apenas 7% a mais quando utilizamos apenas as funções de Shepard.

Com o avanço da tecnologia dos processadores, hoje encontramos no mercado processadores com múltiplos núcleos a preços acessíveis. Como o limite físico para o processamento dos processadores está próximo de ser alcançado pela industria, existe uma tendência que a evolução dos processadores se dê pelo aumento cada vez maior dos núcleos de processamento. Para se tirar proveito desses processadores, é necessário que as aplicações se tornem paralelas para que o esforço computacional possa ser dividido entre os núcleos disponíveis. Nem sempre é simples a paralelização de métodos numéricos, como no caso do FEM, mas percebemos que o MLPG é um candidato natural para a paralelização no seu processo de montagem do sistema linear, etapa de maior custo computacional. A paralelização do método se dá de forma natural pois cada nó sobre o domínio gera uma equação do sistema matricial, ou seja, a influência de cada nó altera uma única linha do sistema e não interfere nas linhas dos demais, dispensando dessa forma técnicas de sincronização e exclusão mútua. Dessa forma, propomos nesse trabalho uma forma simples para a paralelização do método. Os resultados apresentados na Tabela 4.4 mostram que o processo de montagem do sistema linear nas versões paralelas executaram

até $1,96$ vezes mais rapidamente em processadores com dois núcleos e até $3,78$ vezes em computadores com quatro núcleos quando comparados às versões sequenciais.

5.1 Trabalhos futuros

Um longo caminho na pesquisa de métodos sem malha ainda deverá ser percorrido para que esses métodos saiam do mundo acadêmico e se tornem um padrão industrial.

Para diminuir o custo computacional do método, apresentamos uma forma de se paralelizar o processo de montagem do sistema linear que é a etapa do método que demanda maior esforço computacional. Resultados mostraram como o método pode tirar proveito de processadores com múltiplos núcleos sem maiores adaptações do código sequencial, indicando que a paralelização do método é um caminho importante a ser explorado. Indo nessa direção, uma possibilidade interessante é a utilização da placa de vídeo (GPU). As GPUs são processadores massivamente paralelos usados inicialmente para processamento gráfico e jogos mas estão se tornando cada vez mais presentes no cenário da computação de alto desempenho. Além da paralelização em uma mesma máquina pode-se pensar em distribuir o processamento em uma rede ou cluster. Com o aumento de número de núcleos em processadores comuns, espera-se que o MLPG e outros métodos sem malha de fácil paralelização se tornem cada vez mais atrativos.

Melhorar o desempenho no cálculo das funções de forma e suas derivadas também é um problema a ser mais explorado. Esse, de fato, é o principal fator limitador de desempenho identificado para uma implementação eficiente do MLPG, pois a construção das funções de forma envolvem muitos cálculos matriciais como multiplicação e inversão de matrizes que atualmente são implementados de forma direta. Uma forma de otimizar esses cálculos é resolver a inversão de matrizes por meio de solução de sistemas lineares, evitando a multiplicação direta onde for possível. Em (Breitkopf et al., 2000) é proposto um método iterativo para o cálculo das funções de forma e suas derivadas baseadas no MLS, que utiliza apenas produtos de escalares e vetores por vetores e multiplicações de matrizes por vetores. Esse método pode ser implementado para otimizar o método MLS e algo semelhante pode ser proposto para o RPIMp.

Além do custo computacional elevado, existem muitos parâmetros como α_q, α_s que, se não estiverem bem ajustados, fazem com que os métodos não gerem boas soluções. Automatizar a escolha desses parâmetros é um passo importante para tornar o método mais robusto.

Neste trabalho, utilizaram-se distribuições de nós uniformes para discretizar os domínios. O programa aceita distribuições não uniformes de nós, mas o problema é a geração automática desses nós. Para tornar o software mais geral deve-se investigar e implementar soluções mais flexíveis como a abordagem probabilística adotada em (Du et al., 2002).

5.2 Publicações

As publicações abaixo foram geradas durante o período do doutorado, e estão diretamente associadas ao trabalho de tese:

Artigos em periódicos

- Fonseca, Alexandre R.; Correa, Bruno C.; Silva, Elson J.; Mesquita, Renato C. . Improving the Mixed Formulation for Meshless Local Petrov Galerkin Method. IEEE Transactions on Magnetics, v. 46, p. 2907-2910, 2010.
- Fonseca, A. R.; MENDES, Miguel Lima; Mesquita, R. C.; SILVA, Elson José da . Mesh Free Parallel Programming for Electromagnetic Problems. Journal of Microwaves and Optoelectronics, v. 8, p. 101S-113S, 2009.
- Fonseca, A.R.; Viana, S.A.; Silva, E.J.; Mesquita, R.C.. Imposing boundary conditions in the meshless local Petrov Galerkin method. IET Science Measurement & Technology, v. 2, p. 387, 2008.

Anais de eventos

- Fonseca, A. R., CORREA, B. C., Mesquita, R.C., Silva, E.J. Improving the Mixed Formulation for Meshless Local Petrov Galerkin Method In: The 17th Conference on the Computation of Electromagnetic Fields - COMPUMAG 2009, 2009, Florianópolis. Proceedings of the 17th Conference on the Computation of Electromagnetic Fields - COMPUMAG 2009. , 2009. p.380 - 381

- FONSECA, Alexandre Ramos; Viana, S. A.; SILVA, Elson José da; MESQUITA, Renato Cardoso. Imposing Boundary Conditions int the Meshless Local Petrov Galerkin Method. In: The IET 7th International Conference on Computation in Electromagnetics - CEM2008, 2008, Brighton. Proceedings of The IET 7th International Conference on Computation in Electromagnetics - CEM2008, 2008. p. 162-163.

- FONSECA, Alexandre Ramos; MENDES, Miguel Lima; MESQUITA, Renato Cardoso; SILVA, Elson José da. Programação Paralela em Métodos sem Malha Aplicados a Problemas Eletromagnéticos. In: MOMAG 2008 - 13 Simpósio Brasileiro de Microondas e Optoeletrônica - 8 Congresso Brasileiro de Eletromagnetismo, 2008, Florianópolis. Anais do MOMAG 2008 - 13 Simpósio Brasileiro de Microondas e Optoeletrônica - 8 Congresso Brasileiro de Eletromagnetismo., 2008. p. 632-635.

- Fonseca, A. R., Mendes, M. L., Mesquita, R. C., e Silva, E. J. Parallel programming for mesh free methods. In The 13th International IGTE Symposium on Numerical Field Calculation in Electrical Engineering - Abstracts, 2008, p 61–61.

Além delas, foram geradas outras publicações, resultado da colaboração com outros alunos do GOPAC (Grupo de Otimização e Projeto Assistido por Computador), mas que não estão diretamente associadas ao trabalho de tese:

Artigos em periódicos

- CORREA, B. C.; Silva, E.J.; Fonseca, A.R.; OLIVEIRA, D. B.; MESQUITA, Renato Cardoso. Meshless Local Petrov-Galerkin Approach in Solving Microwave Guide Problems. IEEE Transactions on Magnetics, 2011. (aceito para publicação)

- Pereira, G. A. S.; Pimenta, L. C. A.; Fonseca, A. R.; Correa, L. d. Q.; Mesquita, R. C.; Chaimowicz, L.; de Almeida, D. S. C.; Campos, M. F. M. . Robot Navigation in Multi-terrain Outdoor Environments. The International Journal of Robotics Research, v. 28, p. 685-700, 2009.

Anais de eventos

- CORREA, B. C. ; SILVA, Elson José da ; Fonseca, A. R. ; OLIVEIRA, D. B. ; MESQUITA, Renato Cardoso. Meshless Local Petrov-Galerkin in solving microwave guide problems. In: Electromagnetic Field Computation (CEFC), 2010 14th Biennial IEEE Conference on, 2010, Chicago. Electromagnetic Field Computation (CEFC), 2010 14th Biennial IEEE Conference on, 2010.

- BELÉM, F. L. ; Pimenta, L.C.A. ; Fonseca, A. R. ; SALDANHA, Rodney Rezende ; BOSQUE, M. M. ; Mesquita, R. C. ; TAVARES, T. H. B. C . Ferramenta para seleção de corredor de linha aérea de transmissão utilizando geoprocessamento. In: XIV SBSR - Simpósio Brasileiro de Sensoriamento Remoto, 2009, São José dos Campos. Anais do XIV SBSR - Simpósio Brasileiro de Sensoriamento Remoto, 2009. p. 3559-3566.

- PEREIRA, G. A. S., PIMENTA, L. C. A., CHAIMOWICZ, L., FONSECA, A. R., ALMEIDA, D. S. C., CORRÊA, L. Q., MESQUITA, R. C., CAMPOS, M. F. M. Robot navigation in multi-terrain outdoor environments. In: 10th International Symposium on Experimental Robotics, 2006, Rio de Janeiro. Proceedings of the 10th International Symposium on Experimental Robotics. p. 1-10.

- CHAVES, R. D., SILVA, R. P., Nunes, C. R. S., FONSECA, A. R., MESQUITA, R. C. Aplicação de Parâmetros de Qualidade à Visualização de Malhas Volumétricas. In: MOMAG 2006 - 12º SBMO - SIMPÓSIO BRASILEIRO DE MICROONDAS E OPTOELETRÔNICA - 7 CBMag CONGRESSO BRASILEIRO DE ELETROMAGNETISMO, 2006, Belo Horizonte. Anais do MOMAG 2006 - 12o SBMO - SIMPÓSIO BRASILEIRO DE MICROONDAS E OPTOELETRÔNICA - 7 CBMag CONGRESSO BRASILEIRO DE ELETROMAGNETISMO, 2006.

Referências Bibliográficas

Ala, G. (2006). Smoothed particle electromagnetic: A mesh-free solver for transients. Journal of Computational and Applied Mathematics, 191:194–205. [citado na(s) páginas(s) 5]

Atluri, S. N. and Shen, S. (2002). The meshless local Petrov-Galerkin (MLPG) method - A simple and less-costly alternative to the finite element and boundary element methods . CMES - Computer Modeling in Engineering and Sciences, 3(1):11–51. [citado na(s) páginas(s) 6, 7, 9, 40, 41, 48, 60]

Atluri, S. N. and Zhu, T. (1998). A new meshless local Petrov-Galerkin (MLPG) approach in computational mechanics. Computational Mechanics, 22(2):117 – 127. [citado na(s) páginas(s) 5, 6]

Balanis, C. A. (1989). Advanced Engineering Electromagnetics. John Wiley & Sons Inc. [citado na(s) páginas(s) 41]

Barrett, R., Berry, M., Chan, T. F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., and der Vorst, H. V. (1994). Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition. SIAM, Philadelphia, PA. [citado na(s) páginas(s) 87]

Belytschko, T., Krongauz, Y., Dolbow, J., and Gerlach, C. (1998). On the completeness of meshfree particle methods. International Journal for Numerical Methods in Engineering, 43(5):785–819. [citado na(s) páginas(s) 13]

Belytschko, T., Lu, Y. Y., and Gu, L. (1994). Element-free Galerkin methods. International Journal for Numerical Methods in Engineering, 37(2):229 – 256. [citado na(s) páginas(s) 5, 6]

Bischof, C., Mey, D. a., Terboven, C., and Sarholz, S. (2007). Parallel Computers Everywhere. In 16th International Conference on the Computation of Electromagnetic Fields, Compumag 2007, pages 693–700. [citado na(s) páginas(s) 10, 83]

BLAS (Acesso em julho de 2008). BLAS (Basic Linear Algebra Subprograms). http://www.netlib.org/blas/index.html. [citado na(s) páginas(s) 87]

BOOST (2007). Boost.Thread library. http://www.boost.org/doc/html/thread.html. Acesso em abril de 2008. [citado na(s) páginas(s) 83]

Breitkopf, P., Rassineux, A., Touzot, G., and Villon, P. (2000). Explicit form and efficient computation of mls shape functions and their derivatives. International Journal for Numerical Methods in Engineering, 48(3):451–466. [citado na(s) páginas(s) 90]

Buhmann, M. D. (2003). Radial Basis Functions: Theory and Implementations. Cambridge University Press. [citado na(s) páginas(s) 101]

CGAL (2007). The CGAL home page. http://www.cgal.org. Acesso em Setembro de 2007. [citado na(s) páginas(s) 51, 55, 58]

Cingoski, V., Miyamoto, N., , and Yamashita, H. (1998). Element-free Galerkin method for electromagnetic field computations. IEEE Transactions on Magnetics, 34(5):3236—-3239. [citado na(s) páginas(s) 6]

Davis, T. A. (2004a). Algorithm 832: UMFPACK V4.3 - an unsymmetric-pattern multifrontal method. ACM Transactions on Mathematical Software (TOMS), 30(2):196–199. [citado na(s) páginas(s) 87]

Davis, T. A. (2004b). A column pre-ordering strategy for the unsymmetric-pattern multifrontal method. ACM Transactions on Mathematical Software (TOMS), 30(2):165–195. [citado na(s) páginas(s) 87]

Davis, T. A. and Duff, I. S. (1999). A combined unifrontal/multifrontal method for unsymmetric sparse matrices. ACM Transactions on Mathematical Software (TOMS), 25(1):1–19. [citado na(s) páginas(s) 87]

De, S. and Bathe, K.-J. (2000). The method of finite spheres. Computational Mechanics, 25(4):329–345. [citado na(s) páginas(s) 7]

De, S. and Bathe, K.-J. (2001). Towards an efficient meshless computational technique: the method of finite spheres. Engineering Computations, 18(1):170–192. [citado na(s) páginas(s) 8]

de Berg, M., van Kreveld, M., Overmars, M., and Schwarzkopf, O. (2000). Computational Geometry Algorithms and Applications. Springer-Verlag, second edition. [citado na(s) páginas(s) 9, 58]

Du, Q., Gunzburger, M., and Ju, L. (2002). Meshfree, probabilistic determination of point sets and support regions for meshless computing. Computer methods in applied mechanics and engineering, 191(13–14):1349–1366. [citado na(s) páginas(s) 91]

Fernández-Méndez, S. and Huerta, A. (2004). Imposing essential boundary conditions in mesh-free methods. Computer methods in applied mechanics and engineering, 193:1257–1275. [citado na(s) páginas(s) 60]

Fonseca, A. R., Corrêa, B. C., da Silva, E. J., and Mesquita, R. C. (2010). Improving the mixed formulation for meshless local petrov galerkin method. IEEE Transactions on Magnetics, 46:2907–2910. [citado na(s) páginas(s) 10, 74]

Fonseca, A. R., Mendes, M. L., da Silva, E. J., and Mesquita, R. C. (2009). Mesh free parallel programming for electromagnetic problems. Journal of Microwaves and Optoelectronics, 8:101S–113S. [citado na(s) páginas(s) 10]

Fonseca, A. R., Mendes, M. L., Mesquita, R. C., and Silva, E. J. (2008a). Parallel programming for mesh free methods. In The 13th International IGTE Symposium on Numerical Field Calculation in Electrical Engineering - Abstracts, pages 61–61. [citado na(s) páginas(s) 83]

Fonseca, A. R., Mendes, M. L., Silva, E. J., and Mesquita, R. C. (2008b). Programação paralela em métodos sem malha aplicados a problemas eletromagnéticos. In MOMAG 2008 - 13° SBMO – Simpósio Brasileiro de Microondas e Optoeletrônica e o 8° CBMag – Congresso Brasileiro de Eletromagnetismo, pages 632–635. [citado na(s) páginas(s) 83]

Fonseca, A. R., Viana, S. A., Silva, E. J., and Mesquita, R. C. (2008c). Imposing boundary conditions in the meshless local petrov-galerkin method. IET Science, Measurement & Technology, 2(6):387–394. [citado na(s) páginas(s) 9, 10, 61, 73]

Fonseca, A. R., Viana, S. A., Silva, E. J., and Mesquita, R. C. (2008d). Imposing boundary conditions int the meshless local petrov galerkin method. In Proceedings of The IET 7th International Conference on Computation in Electromagnetics - CEM2008. [citado na(s) páginas(s) 61, 73]

Fries, T.-P. and Matthies, H.-G. (2004). Classification and overview of meshfree methods. Technical report, Technical University Braunschweig, Department of Mathematics and Computer Science. [citado na(s) páginas(s) 7, 10, 21, 47]

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. M. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional. [citado na(s) páginas(s) 9]

Gingold, R. A. and Monaghan, J. J. (1977). Smoothed particle hydrodynamics - theory and application to non-spherical stars. Royal Astronomical Society, Monthly Notices, 181:375–389. [citado na(s) páginas(s) 5]

Hildebrand, F. B. (1987). Introduction to Numerical Analysis. Dover Publications. [citado na(s) páginas(s) 52]

Hughes, T. J. R. (2000). The Finite Element Method: Linear Static and Dynamic Finite Element Analysis. Dover Publications. [citado na(s) páginas(s) 1, 52]

Idelsohn, S. R. and Oñate, E. (2006). To mesh or not to mesh. that is the question... Computer methods in applied mechanics and engineering, 195:4681–4696. [citado na(s) páginas(s) 3, 4, 8, 9]

Ikuno, S., Hanawa, T., Takayama, T., and Kamitani, A. (2008). Evaluation of parallelized meshless approach: Application to shielding current analysis in hts. IEEE Transactions on Magnetics, 44(6):1230–1233. [citado na(s) páginas(s) 83]

Ikuno, S., Takakura, K., and Kamitani, A. (2007). Influence of method for imposing essential boundary condition on meshless galerkin/petrov–galerkin approaches. IEEE Transactions on Magnetics, 43(4):1501–1504. [citado na(s) páginas(s) 47]

ITL (Acesso em julho de 2007). The Iterative Template Library. http://osl.iu.edu/research/itl/. [citado na(s) páginas(s) 87]

Kroungauz, Y. and Belytschko, T. (1996). Enforcement of essential boundary conditions in meshless approximations using finite elements. Comput. Methods Appl. Mech. Eng., 131(1–2):133–145. [citado na(s) páginas(s) 20]

Liu, G. R. (2002). Mesh Free Methods: Moving Beyond the Finite Element Method. CRC Press. [citado na(s) páginas(s) 5, 6, 9, 12, 15, 20, 24, 25, 28, 30, 47, 48, 61]

Liu, G. R. and Liu, M. B. (2003). Smoothed Particle Hydrodynamics: A Meshfree Particle Method. World Scientific Publishing Co. Pte. Ltd. [citado na(s) páginas(s) 5, 6]

Lucy, L. B. (1977). A numerical approach to testing the fission hypothesis. The Astronomical Journal, 82:1013–1024. [citado na(s) páginas(s) 5]

Macedo, A. (1988). Eletromagnetismo. LTC. [citado na(s) páginas(s) 41]

Mendes, M. L. (2010). Utilização do método da hidrodinâmica de partículas suavizadas aplicado à resolução de problemas eletromagnéticos. Master's thesis, Universidade Federal de Minas Gerais. [citado na(s) páginas(s) 5]

Nealen, A. (2004). An as-short-as-possible introduction to the least squares, weighted least squares and moving least squares methods for scattered data approximation and interpolation. http://www.nealen.net/projects/mls/asapmls.pdf. Acesso em abril, 2007. [citado na(s) páginas(s) 15, 46]

Nicomedes, W. L., Mesquita, R. C., and Moreira, F. J. S. (2011). A meshless local petrov-galerkin method for three-dimensional scalar problems. IEEE Transactions on Magnetics (accepted for publication). [citado na(s) páginas(s) 79]

Nunes, C. R. S., Mesquita, R. C., and Lowther, D. A. (2007). Remeshing driven by smooth-surface approximation of mesh nodes. In IEEE Transactions on Magnetics, volume 47, pages 1541–1544. [citado na(s) páginas(s) 2, 3]

Parreira, G. F., Fonseca, A. R., Lisboa, A. C., Silva, E. J., and Mesquita, R. C. (2006a). Efficient algorithms and data structures for element-free galerkin method. IEEE Transactions on Magnetics, 42(4):659–662. [citado na(s) páginas(s) 6]

Parreira, G. F., Silva, E. J., Fonseca, A. R., and Mesquita, R. C. (2006b). The element-free galerkin method in threedimensional electromagnetic problems. IEEE Transactions on Magnetics, 42(4):711–714. [citado na(s) páginas(s) 6, 9, 40]

Sadiku, M. N. O. (2004). Elementos de Eletromagnetismo. Bookman, 3ª edition. [citado na(s) páginas(s) 65]

Shewchuk, J. R. (1997). Delaunay Refinement Mesh Generation. PhD thesis, Carnegie Mellon University. [citado na(s) páginas(s) 3]

Taflove, A. (2000). Computational Electrodynamics - The Finite-Difference Time-Domain Method. Artech House, Norwood, MA. [citado na(s) páginas(s) 1]

Viana, S., Rodger, D., and Lai, H. (2004). Meshless local Petrov-Galerkin method with radial basis functions applied to electromagnetics. IEE Proceedings - Science, Measurement and Technology, 6(151):449–451. [citado na(s) páginas(s) 9, 60]

Viana, S. A. (2006). Meshless Methods Applied to Computational Electromagnetics. PhD thesis, University of Bath. [citado na(s) páginas(s) 76, 79]

Wendland, H. (1995). Piecewise polynomial, positive definite and compactly supported radial basis functions of minimal degree. Advances in Computational Mathematics, 4:389–396. [citado na(s) páginas(s) 101]

Wendland, H. (2004). Scattered Data Approximation. Cambridge University Press. [citado na(s) páginas(s) 101]

Wu, Z. (1995). Compactly supported positive definite radial functions. Advances in Computational Mathematics, 4:283–292. [citado na(s) páginas(s) 101]

Apêndice A

Funções RBF

Funções de base radial (RBFs) são aquelas que apresentam simetria radial, ou seja, dependem apenas (além de alguns parâmetros conhecidos) da distância entre o centro da função ($\mathbf{x}_c$) e o um ponto genérico ($\mathbf{x}_i$).

Estas funções podem ser classificadas como globais (diz-se que têm suporte global) ou locais (suporte compacto ou local) quando estão definidas em todo o domínio ou em apenas parte dele. Para facilitar a implementação das funções, definimos r como sendo o raio do suporte (região onde a função é não nula) e normalizamos a distância d entre o centro da função $\mathbf{x}_c$ e o ponto $\mathbf{x}_i$ em função de r. Dessa forma, temos:

$$\begin{aligned} d &= \frac{\|\mathbf{x}_c - \mathbf{x}_i\|}{r} \\ &= \frac{1}{r}\sqrt{(x_1^c - x_1^i)^2 + (x_2^c - x_2^i)^2 + \ldots + (x_k^c - x_k^i)^2 + \ldots + (x_n^c - x_n^i)^2} \end{aligned} \tag{A.1}$$

e as RBFs com suporte compacto são definidas para $0 \leq d \leq 1$. Na equação, x_k^c e x_k^i são os k-ésimos componentes dos pontos $\mathbf{x}_c$ e $\mathbf{x}_i$, respectivamente.

Entre os tipos de funções de suporte global encontram-se (Wu, 1995; Wendland, 1995; Buhmann, 2003; Wendland, 2004):

Multiquádricas	$R(d) = \sqrt{d^2 + c_i^2},$	$c_i > 0$
Multiquádricas recíprocas	$R(d) = \left[d^2 + c_i^2\right]^{-\frac{1}{2}},$	$c_i > 0$
Gaussianas	$R(d) = e^{-cd^2},$	$c_i > 0$
Splines de placas finas	$R(d) = d^{2\beta}\ln(d),$	$\beta \in \mathbb{N}$

onde c e β são parâmetros constantes.

RBFs de suporte compacto são, por exemplo, as de:

Wu e Wendland $\quad R(d) = (1-d)_+^n p(d)$

onde $p(d)$ é um polinômio e $(1-d)_+^n$ é zero para d maior que o raio suporte;

Buhmann $\quad R(d) = \frac{1}{3} + d^2 + \frac{4}{3}d^3 + 2d^2\ln(d)$

além de funções polinomiais definidas por partes.

Para se determinar a derivada parcial de $R(d)$ em relação ao k-ésimo componente de $\mathbf{x}$, x_k, usa-se a regra da cadeia:

$$\frac{\partial R}{\partial x_k} = R_{,k} = \frac{\partial R}{\partial d}\frac{\partial d}{\partial x_k} \tag{A.2}$$

onde $\frac{\partial d}{\partial x_k}$ é dado por

$$\frac{\partial d}{\partial x_k} = \frac{1}{r}\frac{(x_k^i - x_k^c)}{\|\mathbf{x}_c - \mathbf{x}_i\|} \tag{A.3}$$

A seguir, são listadas as RBFs utilizadas nesse trabalho e no final deste apêndice é apresentada uma tabela que compara o tempo de avaliação para as implementações.

A.1 RBF_CubicSpline

$$R(d) = \begin{cases} 4\left(\frac{1}{3} - d + d^2 - \frac{d^3}{3}\right) \\ 4\left(\frac{1}{6} - d^2 + d^3\right) \\ 0 \end{cases} \quad , \quad \frac{\partial R(d)}{\partial d} = \begin{cases} -4 + 8d - 4d^2 & 0,5 < d \leq 1 \\ -8d + 12d^2 & 0 < d \leq 0,5 \\ 0 & d > 1 \end{cases} \tag{A.4}$$

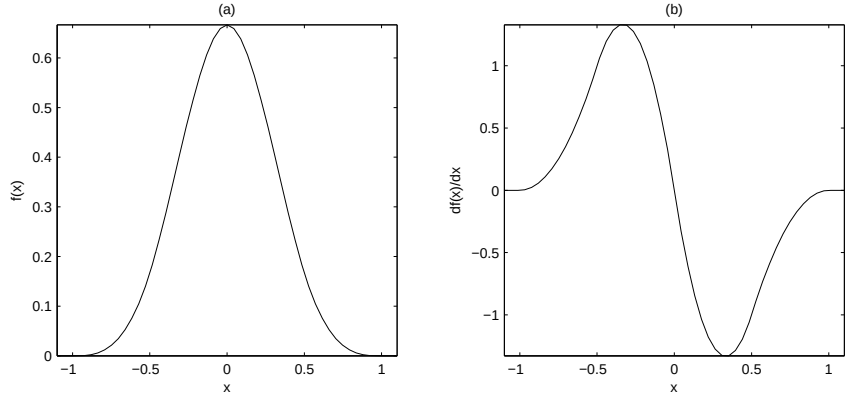


Figura A.1: RBF_CubicSpline

A.2 RBF_Multiquadric

$$R(d) = \sqrt{1 + d^2}, \qquad \frac{\partial R(d)}{\partial d} = \frac{d}{\sqrt{1 + d^2}}$$

A.3 RBF_InverseMultiquadric

$$R(d) = \frac{1}{1 + d^2}, \qquad \frac{\partial R(d)}{\partial d} = -\frac{2d}{(1 + d^2)^2}$$

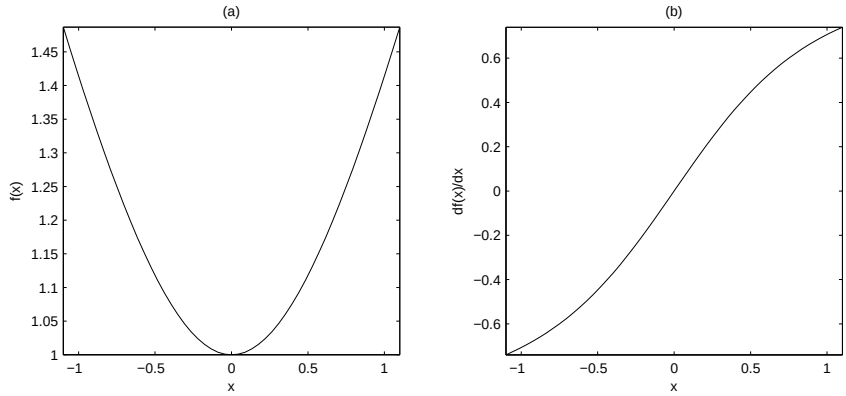


Figura A.2: RBF_Multiquadric

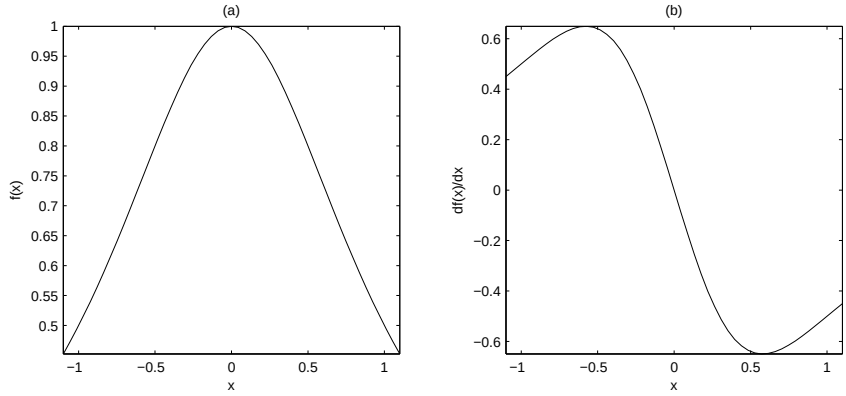


Figura A.3: RBF_InverseMultiquadric

A.4 RBF_Quadratic

$$R(d) = \begin{cases} 2(1-d)^2 \\ 1-2d^2 \\ 0 \end{cases} \quad , \quad \frac{\partial R(d)}{\partial d} = \begin{cases} 4(d-1) & 0,5 < d \leq 1 \\ -4d & 0 < d \leq 0,5 \\ 0 & d > 1 \end{cases}$$

A.5 RBF_ThinPlateSpline

$$R(d) = d^2 \log(|d|), \qquad \frac{\partial R(d)}{\partial d} = d + 2d \log(|d|)$$

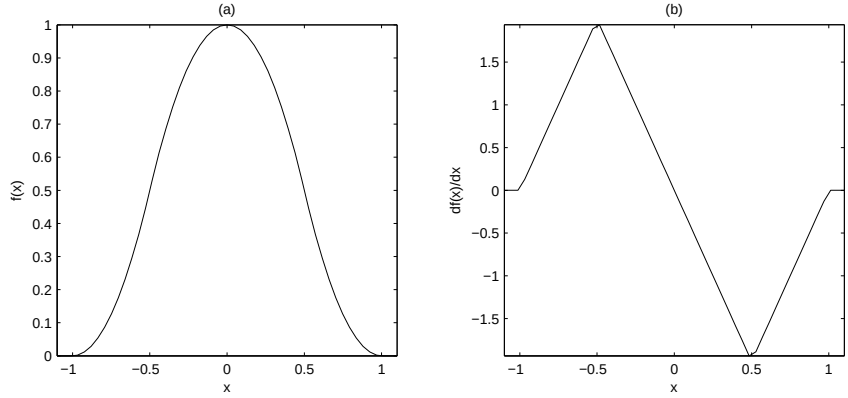


Figura A.4: RBF_Quadratic

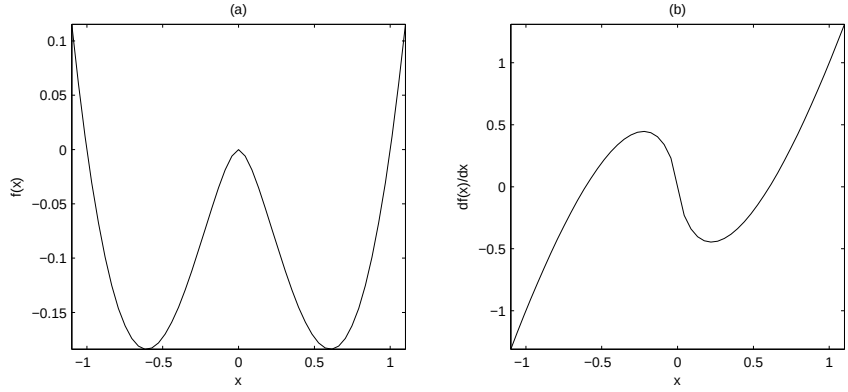


Figura A.5: RBF_ThinPlateSpline

A.6 RBF_Wendland5

$$R(d) = (1-d)^5(8+40d+48d^2+25d^3+5d^4), \qquad 0 \leq d \leq 1$$
$$\frac{\partial R(d)}{\partial d} = -45d^8+189d^6-315d^4+315d^2-144d, \qquad 0 \leq d \leq 1$$

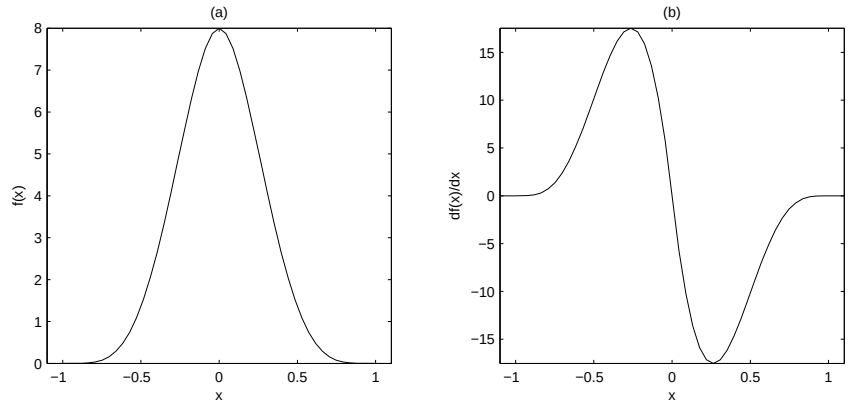


Figura A.6: RBF_Wendland5

A.7 RBF_Wendland6

$$R(d) = (1-d)^6(6 + 36d + 82d^2 + 72d^3 + 30d^4 + 5d^5), \qquad 0 \leq d \leq 1$$
$$\frac{\partial R(d)}{\partial d} = 55d^{10} - 297d^8 + 693d^6 - 1155d^4 + 792d^3 - 88d, \qquad 0 \leq d \leq 1$$

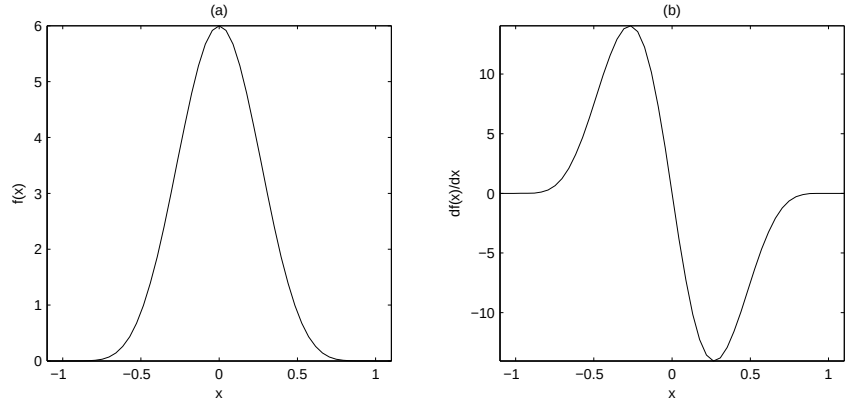


Figura A.7: RBF_Wendland6

A.8 RBF_Quartic

$$R(d) = \begin{cases} 1 - 6d^2 + 8d^3 - 3d^4 \\ 0 \end{cases}, \quad \frac{\partial R(d)}{\partial d} = \begin{cases} -12d + 24d^2 - 12d^4 \\ 0 \end{cases}, \quad \begin{matrix} 0 \leq d \leq 1 \\ d > 1 \end{matrix}$$

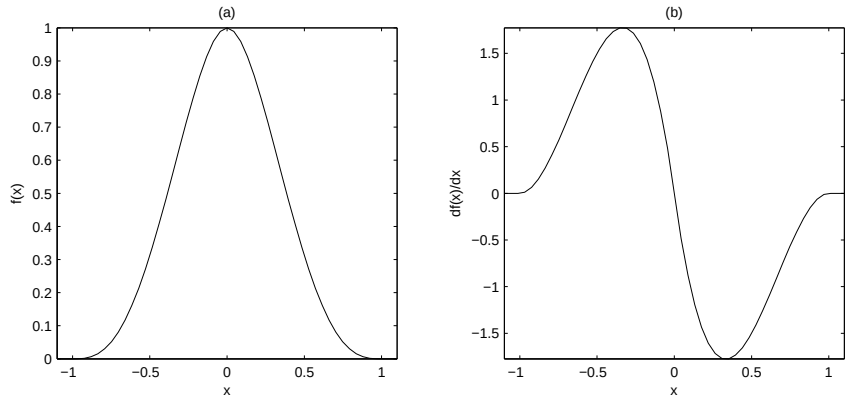


Figura A.8: RBF_Quartic

A tabela abaixo compara os tempos de avaliação das funções e suas derivadas para as várias RBFs implementadas. O tempo médio foi obtido avaliando a função milhares de vezes com d variando aleatoriamente entre 0 e 1 e dividindo a soma desses tempos pelo número de avaliações. Além disso, os tempos são normalizados pela menor média, no caso, o tempo de avaliação da RBF_InverseMultiquadric.

Tabela A.1: Comparação entre os tempos relativos para avaliação das RBFs

	avaliação	derivada
RBF_CubicSpline	1.75142	1.7675
RBF_Multiquadric	2.34356	1.78457
RBF_InverseMultiquadric	1	1.49394
RBF_Quadratic	1.3814	1.65991
RBF_ThinPlateSpline	3.30522	3.95672
RBF_Wendland5	1.71754	2.86569
RBF_Wendland6	1.7957	3.04724
RBF_Quartic	1.25204	2.36359