UNIVERSIDADE FEDERAL FLUMINENSE

DIEGO NUNES BRANDÃO

Métodos Multinível na Resolução de Equações Diferenciais Parciais com Refinamento Adaptativo de Malhas

NITERÓI

2013

Ficha Catalográfica elaborada pela Biblioteca da Escola de Engenharia e Instituto de Computação da UFF

B819 Brandão, Diego Nunes
Métodos multinível na resolução de equações diferenciais parciais com refinamento adaptativo de malhas / Diego Nunes Brandão. – Niterói, RJ : [s.n.], 2013.
100 f.

Tese (Doutorado em Computação) - Universidade Federal Fluminense, 2013.
Orientador: Mauricio Kischinhevsky.

1. Estrutura de dados (Computação). 2. Método dos volumes finitos. 3. Método multinível. I. Título.

CDD 005.7

UNIVERSIDADE FEDERAL FLUMINENSE

DIEGO NUNES BRANDÃO

Métodos Multinível na Resolução de Equações Diferenciais Parciais com Refinamento Adaptativo de Malhas

Orientador:

Prof. Mauricio Kischinhevsky

NITERÓI

2013

Métodos Multinível na Resolução de Equações Diferenciais Parciais com Refinamento Adaptativo de Malhas

Diego Nunes Brandão

Tese de doutorado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para obtenção do Grau de Doutor em Computação. Área de concentração: Computação Científica e Sistemas de Potência.

Aprovada por:

Prof. Mauricio Kischinhevsky / IC-UFF (Presidente)

Prof. Anselmo Antunes Montenegro/ IC-UFF

Prof. Carlos Antonio de Moura / IME-UERJ

Prof. Helena Cristina da Gama Leitão /IC-UFF

Prof. Jorge Stolfi/ IC-UNICAMP

Prof. José Henrique Carneiro de Araújo / IC-UFF

Prof. Uri Michael Ascher/ CS-UBC

Niterói, 12 de dezembro de 2013.

À memória dos meus avós.

"The way to get started is to quit talking and begin doing."
"People often ask me if I know the secret of success and
if I could tell others how to make their dreams come true.
My answer is, you do it by working." (Walt Disney)

Agradecimentos

À minha família, em especial aos meus pais e irmãos pelo amor e dedicação. Muito obrigado por entenderem os momentos de ausência. Amo vocês! Não poderia deixar de citar minhas madrinhas, meu padrinho, meus cunhados (Mari e Deiner) e minha família mineira (Kennedy, Andréia e Eduarda) pelo incentivo.

À família Figueiredo, meu agradecimento pelas orações, especialmente a Carol pelos momentos de descontração.

Ao meu orientador Prof. Mauricio por todo o apoio, amizade, orientação e horas de conversas, nem sempre sobre a tese.

Aproveito para agradecer o apoio financeiro da CAPES durante os primeiros anos de doutoramento e aos membros da banca pelas inúmeras contribuições.

Aos professores e funcionários do IC/UFF, em especial, Prof. José Henrique, Prof. Otton, Prof. Esteban, Profa. Anna, Hélio, Angela, João, Carlinhos, Dul, Rafael, Carlos, Marister e Matheus, por toda a ajuda. A Teresa Cancela por toda atenção dedicada aos alunos da Pós.

Aos professores e amigos da Rural: Profa. Eulina, Prof. Marcos Bacis, Profa. Rosane, Prof. Valdomiro, Ademilton Luiz, Carlos e Vania, Felipe e Josi, Bruno, Marina Vissirini, Fabi, Tata, Valeska, Belleza, Jacque Giori, Augusto, Edgar e o Play.

Aos professores e alunos do Cefet/RJ unidade de Nova Iguaçu pela ajuda durante os últimos anos do doutorado.

A Natália e Letícia pela amizade, broncas e incentivo nos momentos mais complicados. Aos amigos que também foram professores, Prof. Anselmo e Prof. Christiano pelo apoio e ideias.

Aos amigos que me ajudaram e muito durante o doutorado, João Thompson e Sanderson, especialmente ao Facada por ter virado um grande amigo-irmão.

Não poderia deixar de agradecer aos inúmeros amigos da UFF: Jacques, Família Sena, Coppeti, Edcarllos, Idalmis, Luciana Brugiollo, Anand, Renathinha, Alexandre Antunes,

Petrucci, Matheus, Helena, Monica, Sérgio, Gleiph, Mel, Guerini, Marcos Melo, Gustavo Cabeça, Bahia, Rafael Gaucho e Danubia, Eyder, Puca e Temis, Luciene e Flavio. Meu agradecimento a todos aqueles que não foram citados, mas participaram de alguma forma nessa caminhada.

Dentre os amigos do IC/UFF, um obrigado especial para Vera e Viviane pelo apoio, incentivo, carinho e os inúmeros conselhos que vocês me deram.

Acima de todas as coisas e pessoas, a Deus, pelo Seu amor, por me guiar, ensinar e colocar tantas pessoas especiais em meu caminho, afim de que eu não desistisse. Muito Obrigado!

Resumo

Métodos multinível, ou multigrid, utilizam vários níveis de refinamento de malhas computacionais sobrepostas, e que cobrem uma região, o domínio do problema, para acelerar a convergência dos métodos iterativos empregados na solução de sistemas de equações lineares. Estes surgem da discretização de equações diferenciais parciais que descrevem os problemas abordados. O método dos Volumes Finitos é um dos mais empregados na discretização, especialmente quando se tratam de problemas envolvendo leis de conservação. O refinamento adaptativo de malhas produz a obrigatoriedade de se resolverem sistemas que mesclam várias escalas de comprimento, o que deteriora a taxa de convergência dos resolutores de sistemas. Este trabalho apresenta duas abordagens multinível na resolução de equações diferenciais com refinamento adaptativo da malha geométrica discretizada por volumes finitos. A primeira utiliza um método multigrid geométrico para refinamento adaptativo de malhas. Para tanto, uma estrutura de dados adequada foi desenvolvida. Essa abordagem permite que a diferença de nível de refinamento entre elementos vizinhos seja maior do que um, uma regra imposta a diversas estratégias encontradas na bibliografia. A segunda abordagem utiliza um método multinível algébrico na resolução do sistema linear obtido pela estratégia definida pelo Autonomous Leaves Graph (ALG). Um estudo sobre os autovalores das matrizes geradas pela estratégia definida pelo ALG também foi realizado. Resultados numéricos demonstram um ganho de eficiência das abordagens aqui propostas quando comparadas ao ALG implementado com o método dos gradientes conjugados para resolução dos sistemas de equações lineares.

Abstract

Multigrid methods employ multiple coexisting computational meshes for the same region, each with its level of refinement, and help improve the convergence of iterative methods used to solve systems of equations. These systems often arise from the discretization of differential equations used to study scientific problems on continuum domains, by means of a mathematical model. The method of finite volumes is often used for the discretization of such models, in particular for those dealing with conservation laws. This work aims to present an adaptive scheme for the multigrid method on non-uniform meshes arising from the discretization by the finite volume method and linked to adaptive mesh refinement. One of the contributions of this work is the development of a geometric multigrid algorithm for problems with adaptive refinement, coupled with an infrastructure based on the autonomous leaves graph (ALG). Another new contribution is the coupling of ALG infrastructure with the algebraic multigrid method - AMG. Results show a gain of efficiency when both approaches are compared to the first versions of ALG, for which the numerical solution employs the conjugated gradient method.

Palavras-chave

1. Métodos Multinível
2. Método dos Volumes Finitos
3. Estrutura de Dados
4. Refinamento Adaptativo de Malhas

Glossário

ALG	:	Grafo de Folhas Autonômas
EDP	:	Equações Diferenciais Parciais
MGC	:	Método dos Gradientes Conjugados
MGCM	:	Método dos Gradientes Conjugados Modificado
CHM	:	Curva de Hilbert Modificada
AMG	:	Multigrid Algébrico
MG	:	Multigrid Geométrico
SFP	:	Curva de Preenchimento de Espaço
AMR	:	Refinamento Adaptativo de Malhas
MGA	:	Multigrid Geométrico Adaptativo
GS	:	Método de Gauss-Seidel
FAC	:	Fast Adaptive Composite
MLAT	:	Multilevel Adaptive Technique
FTT	:	Fully Threaded Tree
MVF	:	Método dos Volumes Finitos
cond(A)	:	Número de Condicionamento da Matriz A
FMG	:	Full Multigrid
GJ	:	Método de Gauss-Jacobi
SSOR	:	Método das Relaxações Sucessivas

Conteúdo

Lista de Algoritmos

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A simulação computacional tem papel fundamental na compreensão de fenômenos físicos de problemas reais oriundos das mais diversas áreas. Quando comparada com métodos analíticos e experimentais ela apresenta vantagens significativas, pois permite que diferentes cenários e variáveis do problema sejam avaliados, sem um alto custo e sem comprometer a região ou material de estudo. A simulação se aplica a problemas de aerodinâmica, transporte de contaminantes, propagação de ondas, entre outros. No contexto de transporte de contaminantes, ela permite que diferentes cenários sejam avaliados sem comprometer o meio ambiente.

Os modelos matemáticos adequados para o estudo de tais problemas podem ser descritos por meio de equações diferenciais parciais (EDP's). Essas EDP's são definidas em um certo domínio contínuo de interesse com condições de contorno e inicial apropriadas. A solução numérica dessas EDP's ocorre pela aplicação de diversos métodos numéricos, por exemplo, o método dos volumes finitos. Nesse método o domínio contínuo é dividido (discretizado) em elementos menores, denominados volumes de controle. A união dos volumes de controle forma a malha discreta do problema.

No domínio discreto a EDP é transformada em uma equação discreta, que é utilizada em cada um dos elementos da malha, gerando um sistema de equações algébricas que, quando solucionado, fornece uma aproximação para o valor da solução do problema original em um número finito de pontos discretos do domínio. Na abordagem utilizada neste trabalho, esses pontos estão geometricamente localizados nos centros dos volumes de controle.

Teoricamente, a precisão dessa aproximação é tão boa quanto a discretização do do-

mínio. Isto significa que quanto menores os volumes de controle, maior a quantidade de pontos e melhor torna-se a solução obtida, a menos de erros de arredondamento e truncamento. Em contrapartida, maior torna-se o sistema algébrico a ser resolvido e maior o custo computacional desse processo. Assim, aumentar a quantidade de pontos de maneira que esses pontos sejam uniformemente distribuídos por todo o domínio e assim todas as regiões possuam a mesma quantidade de pontos implica num esforço computacional muito alto na solução do problema. Todavia, observa-se que os fenômenos físicos, em muitos casos, desenvolvem uma dinâmica natural, em geral dependente do tempo, em que ocorrem mudanças rápidas na solução ou na sua derivada em certas regiões do domínio (Burgarelli et al., 2006). Dessa maneira, torna-se importante o uso de técnicas que considerem esse comportamento na solução do problema. Uma alternativa é o uso de técnicas adaptativas que oferecem certo grau de robustez e eficiência (Burgarelli et al., 2006).

Métodos adaptativos têm como objetivo aumentar a precisão da solução aproximada em certas regiões do domínio espacial ou em certos instantes de tempo do problema. A representação computacional desse domínio discretizado, denominado malha computacional, requer a utilização de estruturas de dados especiais. As estruturas de dados do tipo árvore são as mais utilizadas por esses métodos; entretanto, elas enfrentam uma limitação chamada de regra $2:1$. Nessa regra a diferença entre o nível de refinamento de células vizinhas não pode ser maior do que 1, caso contrário os algoritmos de refinamento e desrefinamento tornam-se de difícil manipulação. Contudo, alguns problemas necessitam de diferenças maiores e, assim, ocorre uma propagação do refinamento para células vizinhas, aumentando o custo computacional.

Mesmo que técnicas adaptativas sejam empregadas, o número de variáveis a serem determinadas no problema ainda é grande e por isso torna-se necessário o uso de métodos eficientes na solução do sistema algébrico. Dentre estes, os iterativos são os mais eficientes para problemas esparsos de grande porte. Todavia, tais métodos têm sua velocidade de convergência comprometida quando a matriz do sistema a ser resolvido é mal-condicionada. Uma alternativa para esses problemas são os chamados métodos Multigrid. Estes são mais competitivos quando comparados aos métodos iterativos pois, apesar de produzir um "overhead" nas operações de prolongamento e restrição (explicadas no Capítulo 2), ele possui custo computacional assimptoticamente ótimo.

O método Multigrid tem relação direta com o comportamento de convergência dos métodos iterativos estacionários. Tais métodos tem características de suavizarem as componentes de alta frequência do erro da aproximação da solução de maneira mais rápida

do que as componentes de baixa frequência. As componentes de baixa frequência do erro em um domínio mais discretizado (malha fina) quando observadas em um domínio menos discretizado (malha grosseira) se comportam como componentes de alta frequência.Tal comportamento permite que os métodos iterativos estacionários sejam novamente aplicados suavizando essas componentes. O método Multigrid consiste na aplicação recursiva desse procedimento.

A característica mais importante do Multigrid é que sua convergência é independente do tamanho do problema. Além disso, o custo computacional de cada etapa do método é proporcional ao número de incógnitas do problema, sendo superior a outros métodos iterativos não-clássicos, por exemplo, o método dos gradientes conjugados. Outra caracter'istica do método Multigrid é ser facilmente paralelizável, com exceção da sua fase de construção. No entanto, tal característica não será abordada no âmbito deste trabalho.

Este trabalho tem como foco a resolução de problemas mal-condicionados oriundos de uma abordagem adaptativa em volumes finitos. O objetivo é reduzir o tempo gasto na fase de resolução do sistema linear que surge dessa abordagem. Para tanto, foram utilizados dois algoritmos Multigrid. O primeiro utilizou uma estrutura de dados baseada no trabalho de (Burgarelli et al., 2006) para desenvolver um método Multigrid Geométrico Adaptativo. Já o segundo utilizou a metodologia desenvolvida em (Burgarelli et al., 2006) acoplando a ela um Multigrid Algébrico na resolução do sistema linear. Os resultados obtidos mostram que ambas as abordagens apresentam uma redução considerável no tempo de computação quando comparados com a estratégia original (Burgarelli et al., 2006). No entanto, a primeira abordagem proposta apresenta um custo maior de memória pois mantém em memória diferentes níveis de refinamento simultaneamente.

1.1 Contribuições

Este trabalho tem como contribuição principal o desenvolvimento de uma abordagem Multigrid para esquemas centrados nas células com refinamento adaptativo. Para tanto, uma estrutura de dados baseada nas ideias da estrutura de dados ALG proposta por Burgarelli et al. (2006) e da *quadtree* (Samet, 1990) foi desenvolvida. Essa abordagem permite que o método Multigrid desenvolvido atinja uma convergência mais rápida do que a obtida pelo ALG. Além disso, essa estrutura de dados gera uma quantidade menor de elementos na malha do que a *quadtree*, pois permite que a diferença entre os níveis de elementos vizinhos seja maior do que 1. Todavia, tal estrutura tem um custo de

armazenamento mais alto do que as outras duas.

Também é apresentada uma análise do custo de memória do armazenamento do ALG, tanto para o caso de refinamentos uniformes quanto para o caso adaptativo. A partir dessa análise uma proposta de modificação na estratégia de simplifacação do ALG é realizada. Com essa modificação o custo de armazenamento é reduzido.

Estudos sobre as matrizes de discretização que surgem pela estratégia empregada pelo ALG também são realizados aqui. A partir desses estudos uma estratégia baseada no Multigrid Algébrico (AMG) é apresentada. Isto significa que o AMG é combinado ao ALG gerando um esquema mais eficiente em relação ao tempo de execução quando comparado ao ALG tradicional. Tal estratégia também permite que problemas de geometria complexa sejam abordados sem modificações significativas no resolutor do sistema linear.

As abordagens apresentadas são avaliadas por meio dos problemas de Laplace e Poisson. Tais avaliações demonstram a viabilidade das mesmas e apresentam uma maior eficiência quando comparados ao ALG tradicional. Por fim, o problema da camada limite aplicado a um aerofólio é solucinado utilizando a combinação ALG+AMG.

1.2 Trabalhos Relacionados

Nesta seção são apresentados alguns trabalhos relacionados a este projeto. Primeiramente são abordados os métodos Multigrid para malhas adaptativas, em particular para esquemas centrados nas células. Em seguida, são apresentados alguns trabalhos sobre estruturas de dados para refinamento adaptativo de malhas; novamente, um enfoque maior é dado às que abordam esquemas centrados e o método Multigrid.

1.2.1 Multigrid para Malhas Adaptativas

Existem duas abordagens possíveis para métodos Multigrid Geométrico em malhas adaptativas: estática e dinâmica. Na primeira abordagem, o refinamento é pré definido, a malha é refinada antes do processo de solução. Na abordagem dinâmica, o refinamento da malha ocorre de acordo com a evolução da solução do problema, em geral, busca-se que o erro da aproximação da solução seja mínimo em alguma norma (Briggs et al., 2000). A combinação dessas duas abordagens também é possível. Os algoritmos existentes para métodos Multigrid adaptativos, em geral, partem da ideia de refinamentos estáticos para refinamentos dinâmicos simplesmente acrescentando controladores de erro ao processo

adaptativo (Trottenberg et al., 2001).

A estratégia mais conhecida para métodos Multigrid em malhas adaptativas é a *Multilevel Adaptive Technique* (MLAT) proposta por Brandt (Brandt, 1977). No esquema MLAT um nível da malha é composto por todas as células de um mesmo tamanho. Por exemplo, a região hachurada na Figura 1.1. Como resultado malhas mais refinadas em cada instante da simulação são compostas por subregiões mais refinadas que cobrem partes do domínio. Quando a suavização é aplicada sobre um nível l, condições de contorno temporárias do tipo Dirichlet são aplicadas na interface de um elemento no nível l e um outro elemento no nível $l-1$, ou seja, os elementos do nível menos refinado vizinhos aos elementos de nível l são considerados como valores de contorno (Brandt, 1977; Briggs et al., 2000; Trottenberg et al., 2001). Isto significa que a suavização é realizada somente sobre subdomínios uniformes. Com essa abordagem torna-se desnecessário realizar mudanças nos procedimentos de suavização. Os operadores de prolongamento e restrição são escolhidos de forma a manterem constantes os pontos de contorno entre os níveis, durante a suavização da malha mais refinada. A solução do problema sobre todo o domínio é obtida de forma implícita sobre todos os níveis da malha. Quando o MLAT é formulado como um método de correção de defeitos, as suavizações da malha mais refinada são realizadas de forma a aumentar a precisão do método de solução na malha menos refinada.

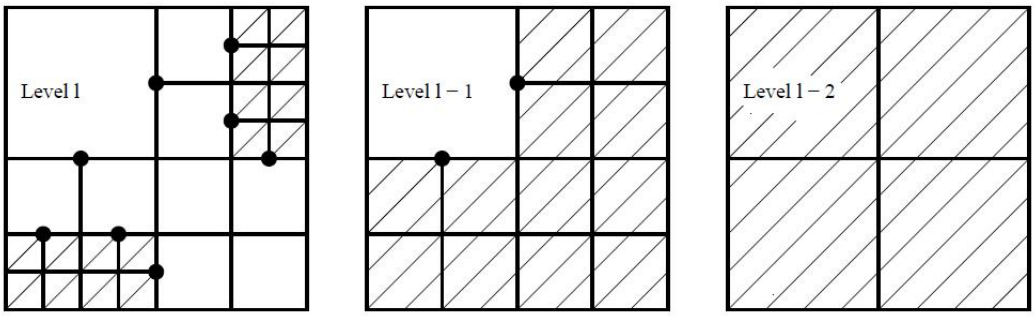


Figura 1.1: Níveis multigrid em uma malha não uniforme. As regiões hachuradas mostram os elementos que fazem parte do nível definido (Jones e Jimack, 2000).

Outra abordagem muito utilizada foi apresentada em (McCormick e Thomas, 1986). Nesse trabalho foi desenvolvido o esquema *Fast Adaptive Composite* (FAC). A teoria de convergência daquele método pode ser encontrada em (McCormick e Rude, 1994), enquanto várias aplicações são apresentadas em (McCormick, 1989; Hart et al., 1986). O FAC define o nível em uma malha hierárquica de forma similar ao MLAT, mas os operadores de discretização são definidos em uma composição do domínio que também inclui o domínio inteiro. Essa composição é composta por células mais refinadas de cada subdomínio sendo portanto não uniforme. Toda a suavização é realizada sobre malhas uniformes e

subdomínios dessas malhas. Apesar disso, todos os resultados são obtidos explicitamente sobre a composição da malha, sendo esses transferidos para os níveis dos subdomínios de maneira a calcular as correções referentes a cada subdomínio. Para evitar a inconsistência nos pontos de contorno interníveis, os operadores de discretização definidos para a composição da malha em tais pontos são utilizados para calcular o resíduo nestes pontos. Os operadores de transferência intergrid são escolhidos de maneira a não interferirem na obtenção do termo residual e na transferência da solução.

Um método adaptativo multinível iterativo foi introduzido em (Rüde, 1994). Ele emprega um método de "refinamento global virtual" no qual, embora todos os pontos de uma malha uniforme estejam presentes, somente um conjunto de pontos ativos é utilizado no processo de suavização. Um ponto é removido do conjunto de pontos ativos se o cálculo da correção para aquele ponto é menor que uma certa tolerância pré fixada. Assim, conforme o processo de suavização aumenta, o conjunto de pontos ativos diminui. Quando é calculada a correção em um ponto, sua vizinhança é adicionada ao conjunto de pontos ativos e o valor residual é atualizado. Quando aplicado como um suavizador de um método Multigrid este processo é algebricamente equivalente ao método FAC. Entretanto, sua diferença em relação ao FAC deve-se ao fato de que, para o método introduzido em (Rüde, 1994), os pontos da malha são divididos em ativos e inativos sobre uma região individual sendo, então, agrupados em grupos (*patches*).

Esse conceito de *patches* é utilizado em diversos trabalhos (Martin e Cartwright, 1996; Thorne, 2003) que utilizam esquemas centrados nas células. Nestes trabalhos são criados elementos temporários nas interfaces de elementos de diferentes níveis de refinamento e seus valores são determinados por interpolação. Assim operadores especiais são desenvolvidos de maneira a atenuar a descontinuidade entre os níveis. Essa abordagem acaba gerando refinamentos em regiões de forma desnecessária, pois busca gerar *patches* mais abrangentes, evitando assim a necessidade excessiva de tratamentos nos contornos dos elementos de níveis diferentes.

Uma abordagem diferente para o método Multigrid foi apresentada por (Hartmann et al., 2008). Neste trabalho os autores desenvolvem o método Multigrid por meio do chamado método de malhas cartesianas. Tal método permite a geração da malha de forma automática, o que, facilita a sua utilização em esquemas adaptativos. O método é aplicado na resolução das equações de Euler e Navier-Stokes.

Uma semelhança nesses trabalhos é a utilização de estruturas de dados baseadas em árvores para armazenamento das informações do problema. Por exemplo, uma árvore

ternária é utilizada no software PLTMG, para o qual o nível de um elemento triangular corresponde ao seu nível na estrutura da árvore. A seguir outras estruturas de dados para métodos adaptativos são discutidas.

Uma outra possibilidade de utilização do Multigrid para malhas adaptativas é a sua versão algébrica (AMG). Esta versão avalia somente as informações da matriz do sistema. Os primeiros trabalhos sobre o AMG foram desenvolvidos em (Brandt et al., 1982; Ruge e Stüben, 1986). São inúmeros trabalhos que utilizam o AMG em problemas adaptativos (Wu e Elman, 2006; Tang e Yuan, 2010; Brezina et al., 2012); todavia, nenhum deles utiliza a mesma abordagem que é apresentada neste trabalho.

1.2.2 Estruturas de Dados para Malhas Adaptativas

O que resulta do refinamento da malha em processos adaptativos são malhas não uniformes que se modificam em cada passo de tempo. Uma estrutura de dados para armazenar esse os dados desse tipo de problema deve facilitar a busca por informações referentes à geometria, conectividade, etc. Existem inúmeras maneiras para essa representação(Samet, 1990). Uma abordagem padrão seria organizar a informação em vetores (Briggs et al., 2000). Outra possibilidade seria a representação da malha de forma hierárquica. Nessa representação são utilizadas as estruturas de dados em forma de árvore, denominadas *quadtree* (ou *octree*) (Samet, 1990; Wirth, 1985).

Uma *quadtree* é uma estrutura de dados hierárquica usada para representar uma malha, onde cada nó da estrutura representa uma célula. Para melhor elucidar essa definição, suponha que o domínio de interesse seja o quadrado unitário. Este quadrado unitário é representado por um elemento da estrutura denominado raíz da mesma. Esta célula pode ser decomposta em 4 novas células iguais, que por sua vez podem ser subdivididas também até que um certo critério de parada seja alcançado. Os nós da estrutura que representam quatro partes de uma célula são denominados filhos do nó que representa esta célula (nodo pai). Os nós filhos são nomeados de acordo com sua posição no elemento pai: nordeste(NE), sudeste(SE), noroeste(NO) e sudoeste(SO)}, veja Figura 1.2.

Um elemento é denominado folha ou terminal se não tem filhos. Dois elementos são chamados de vizinhos ou adjacentes se eles têm uma aresta em comum. O nível de um elemento é igual ao número de decomposições que foram necessárias para obtê-lo. O nível da raiz é zero. Como pode ser visto, as informações relativas a uma *quadtree* constituem uma árvore. Na Figura 1.3 são apresentadas uma malha quadrangular e sua representação em forma de árvore *quadtree*; observe que, internamente, cada nodo possui um ponteiro

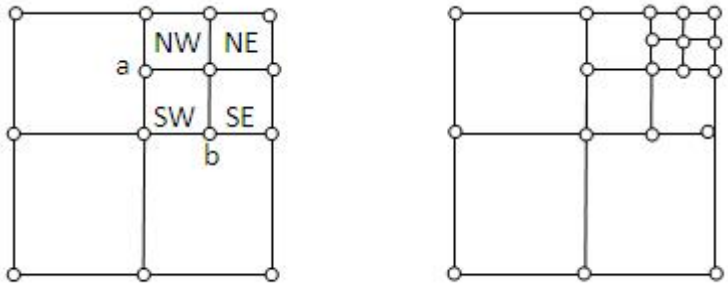


Figura 1.2: Representação de uma *quadtree*.

para seu pai. A altura desta árvore é dada pelo nível máximo de qualquer nó.

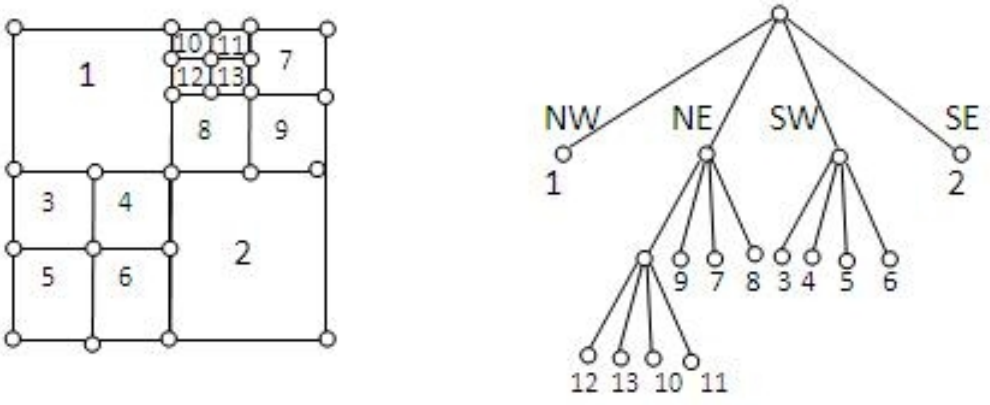


Figura 1.3: Representação de uma *quadtree* e sua representação em árvore.

O emprego de uma estrutura de árvore permite que informações como nível de refinamento, conectividade, ancestral, vizinhos e filhos de um elemento sejam obtidas facilmente. Com estas informações outros dados que forem necessários, como por exemplo o exemplo o vizinho de um dado elemento, podem ser obtidos facilmente. O custo de armazenamento dessa estrutura em memória é, no pior caso, dado por $\sum_{i=0}^{i=n} 4^i = \frac{4^{n+1}-1}{3}$ nodos onde n é a altura da árvore.

Khokhlov (1998) apresentou uma modificação na *quadtree* de forma que elementos de níveis diferentes tenham uma comunicação direta. Ele organizou células provenientes do mesmo pai em uma estrutura denominada FTT (*Fully Threaded Tree*). Entretanto, o gerencimento de uma FTT é complexo quando a diferença entre níveis de refinamento de células vizinhas for maior do que 1 (regra 2:1). Baseado neste trabalho, em (Ji et al., 2010) foi apresentada uma modificação na estrutura de dados FTT de forma a prover um maior paralelismo da mesma. Entretanto a observância à regra $(2:1)$ foi mantida, ou seja, a diferença de nível entre duas células vizinhas pode ser no máximo 1.

Como as árvores apresentam essa estrutura hierárquica e os métodos Multigrid também apresentam essa característica, as árvores são as estruturas de dados mais utilizadas na implementação do Multigrid. São inúmeros os trabalhos envolvendo árvores na implementação do Multigrid (Bank, 1990; Rüde, 1993; Jones e Jimack, 2000; Hartmann et al., 2008; Li et al., 2005; Unfer et al., 2010; Ji et al., 2012).

Entretanto, existem outras estruturas de dados capazes de representar de forma satisfatória malhas adaptativas.

No escopo de métodos Multigrid, Rivara (1984) implementa estrutura de dados baseada em listas para refinamentos adaptativos em malhas de elementos finitos triangulares e conformes. Ela implementa uma lista molecular, onde cada item representa um ponto e armazena o conjunto ordenado de pontos vizinhos, bem como, se ele é um ponto de contorno. A partir disso uma nova lista contendo a sequência de triangulações que surgem de acordo com os refinamentos adaptativos é armazenada.

Todavia o foco do presente trabalho são malhas com elementos quadrangulares e com discretização centrada. Neste escopo, Burgarelli et al. (2006) propuseram uma estrutura de baixo custo computacional para representar refinamento de malhas na solução de equações diferenciais, que empregava o método dos Volumes Finitos com volumes quadrangulares. A estrutura consiste em utilizar um grafo de folhas autônomas, denominado ALG, sendo a ordenação dos volumes da malha feita através de uma Modificação da Curva de Hilbert. Oliveira e Kischinhevsky (2009) propuseram uma modificação nessa estrutura para aplicá-la em volumes finitos triangulares, utilizando a curva de Sierpinski na ordenação dos volumes. Em 2010, Robaina et al. (2010) estenderam o ALG para o caso de visualização científica em $3D$.

Uma simplicação do ALG foi apresentada em Unfer et al. (2010) e aplicada na análise da interação do plasma com fluxo de gás. Já em Oliveira et al. (2011) o problema da camada limite utilizando o ALG é simulado. Oliveira et al. (2012a) utilizaram o ALG na solução de problemas da modelagem elétrica do coração. Já em (Oliveira et al., 2012b) uma versão que implementa o método dos gradientes conjugados em paralelo é associada ao ALG de maneira a aumentar a eficiência da simulação.

A principal vantagem do ALG está na não limitação à regra $(2:1)$. Ele foi construído de maneira a facilitar o refinamento e desrefinamento da malha. Assim diferenças entre os níveis de células vizinhas podem ser de qualquer ordem, sendo especificados pelo problema a ser resolvido. Outra característica importante é que células vizinhas oriundas de pais diferentes são diretamente conectadas. Essa característica permite uma comunicação mais

rápida quando comparada às implementações tradicionais das *quadtrees*. Outra característica importante é a representação somente das células existentes na discretização da malha, enquanto nas *quadtrees* elementos de diferentes níveis de refinamento estão todos presentes mesmo não compondo o domínio computacional do problema naquele instante de tempo. Isto significa que um elemento pai refinado ainda se encontra armazenado em memória.

1.3 Estrutura do Trabalho

O presente trabalho está estruturado da seguinte forma: a estrutura de dados denominada ALG (*Autonomous Leaves Graph*) é revisada no capítulo 2, bem como uma análise do seu custo de armazenamento; uma modificação no processo de simplificação é introduzida, uma discussão sobre o condicionamento da matriz da discretização utilizada por essa abordagem também é apresentada. O capítulo 3 apresenta uma introdução sobre alguns aspectos do método Multigrid, os elementos que constituem este método são discutidos. No capítulo 4 uma estrutura de dados baseada no ALG e na *quadtree* é apresentada, bem como o algoritmo Multigrid Geométrico desenvolvido para malhas adaptativas. No capítulo 5 o método Multigrid Algébrico é apresentado e sua integração com o ALG é discutida. Os resultados das duas estratégias aqui abordadas são apresentados no capítulo 6. Por fim, os trabalhos futuros e conclusões são discutidos no capítulo 7.

Capítulo 2

Revisão sobre o Grafo de Folhas Autonômas

Neste capítulo são revisadas algumas definições sobre a estrutura de dados proposta em (Burgarelli, 1998; Burgarelli et al., 2006) e suas variações (Unfer et al., 2010; Brandão et al., 2009; Robaina et al., 2010). Uma análise do custo de armazenamento dessa estrutura é apresentada, bem como uma discussão sobre o condicionamento da matriz que surge pela estratégia empregada.

2.1 *Grafo de Folhas Autônomas* (ALG)

O ALG é uma estrutura de dados que foi proposta em (Burgarelli, 1998). Ele consiste basicamente de dois tipos de vértices ou nodos: pretos ou nodos células e os brancos ou nodos de transição. Para uma melhor compreensão de tal estrutura, torna-se interessante uma interpretação por meio de uma analogia com o problema geométrico. Assim, considere um domínio bidimensional formado pelo quadrado unitário, $\Omega = [0, 1]^2$. Dividindo esse quadrado em quatro quadrados iguais obtêm-se quatro células quadradas de aresta medindo $\frac{1}{2}$.

Cada um desses quadriláteros é representado por um nodo preto ou nodo célula que pode ser ilustrado como um ponto preto no centro da célula correspondente. As informações referentes à geometria e à física do problema (as grandezas a serem avaliadas) são armazenadas nesses nodos. Esses nodos são os vértices de um grafo orientado (Figura 2.1). Acrescentam-se a essa estrutura os chamados nodos brancos ou nodos de transição. Eles representam tanto a fronteira do domínio quanto uma mudança de nível entre os ele-

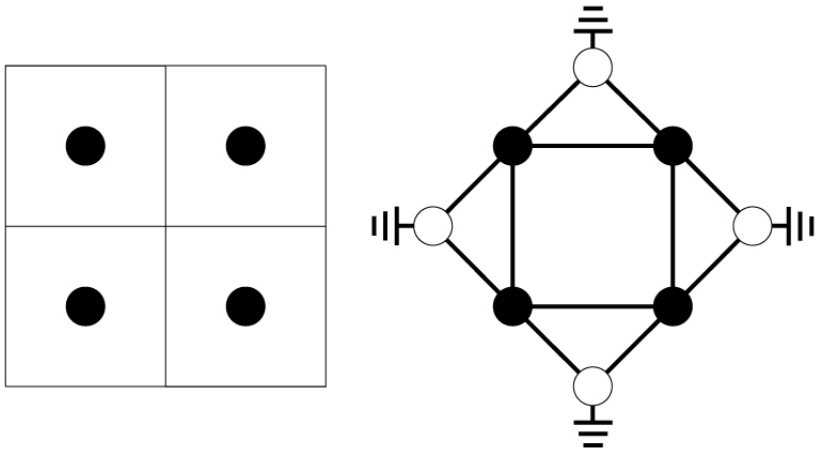


Figura 2.1: Representação geométrica e estrutura de dados do ALG para uma malha quad simples.

mentos do domínio (Burgarelli et al., 2006) . Cada nodo preto possui 4 arestas (ponteiros) para seus vizinhos, tais arestas são nomeadas de acordo com as direções: norte, sul, leste e oeste.

Já os nodos brancos possuem três arestas (*d1, d2 e h*), conforme a Figura 2.2. As duas primeiras apontam para os nodos pretos referentes ao "cacho" que elas pertencem e a última aresta (*singleconnector*) que aponta para um vértice especial, denominado vértice terra que está presente nas fronteiras do domínio. Cada linha na Figura 2.1(direita) representa duas arestas opostas.

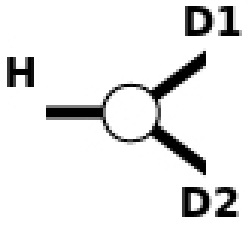


Figura 2.2: Descrição da estrutura do nodo branco com as respectivas arestas.

Assim como na *quadtree*, a definição de nível para esses vértices está ligada à definição de profundidade. Neste caso, a profundidade de uma célula pode ser vista como o número de passos realizados para sua obtenção, sendo convencionado que as células de uma malha composta, no quadrado unitário, por 4 células iguais, estão no nível 1 de profundidade. A partir desse ponto novas células são obtidas com níveis de profundidade sendo acrescido de uma unidade.

A diferença entre os níveis pode ser determinada pelos vértices brancos. Nodos brancos

conectam um nodo preto ou um branco de profundidade n com dois nodos pretos ou brancos de profundidade $n + 1$.

2.1.1 Refinamento e Desrefinamento

Após o refinamento, uma célula de nível n de profundidade é substituída por 4 células de nível $n + 1$ de profundidade. O nível de profundidade de uma célula está diretamente relacionado com o tamanho do lado da célula. Assim, uma célula de lado $\frac{1}{2^n}$ tem nível de profundidade n.

Sempre que uma célula é refinada substitui-se o nodo preto correspondente por uma estrutura similar à Figura 2.1, onde as células da malha inicial são identificadas de acordo com as posições cardeais nordeste, noroeste, sudoeste e sudeste. A Figura 2.3 representa a malha inicial com um refinamento do elemento noroeste (NW). Doravante, os vértices chamados de terra serão omitidos.

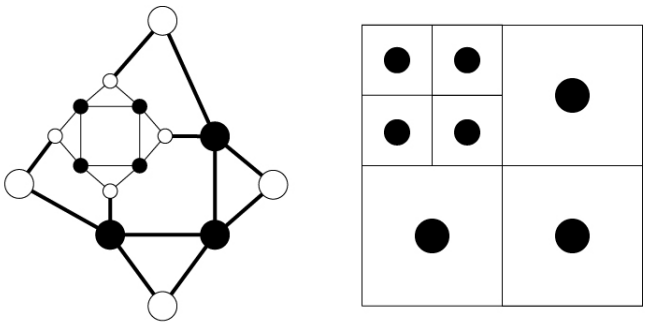


Figura 2.3: Representação da estrutura de dados com o refinamento da célula noroeste.

A estrutura formada por 4 nodos pretos ligados aos nodos brancos representada na Figura 2.1 é denominada *cacho*. Sempre que uma célula for refinada, o nodo preto correspondente será substituído por um cacho. Nesse novo cacho cada nova célula, isto é, cada novo nodo preto armazena algumas informações que permitem determinar quem era o seu pai, sem a necessidade de manter este armazenado. Com essa informação é possível obter o nodo pai no caso de um desrefinamento.

O ALG também realiza algumas simplificações dos chamados vértices de transição. Se, após um refinamento, vértices de transição de mesmo nível estiverem conectados, eles podem ser simplificados (Fig.2.4). Essa simplificação possibilita um funcionamento eficiente do algoritmo de busca dos vizinhos de um nodo preto. Neste caso, nodos pretos que possuem pais diferentes mas estão no mesmo nível e são geometricamente vizinhos

são diretamente conectados (Fig. 2.4). Essa é uma das principais diferenças em relação às *quadtrees* tradicionais.

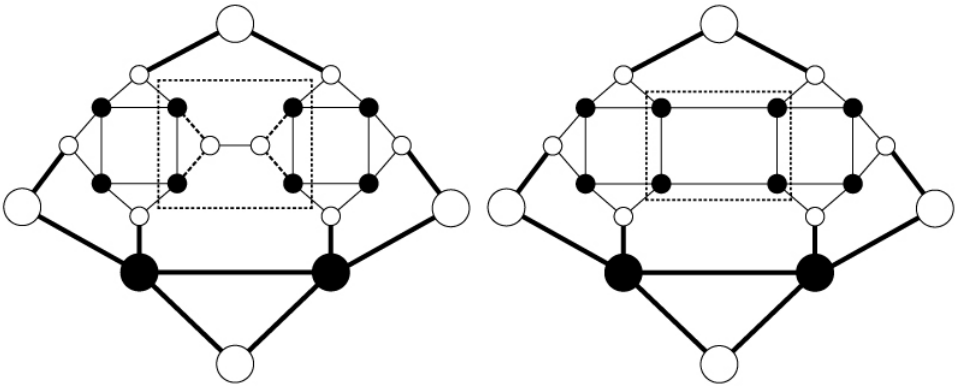


Figura 2.4: Simplificação da estrutura ALG pela eliminação de dois nodos brancos adjacentes de mesmo nível.

O processo de desrefinamento no ALG só pode ser realizado se houver quatro nodos pretos com mesmo nível de profundidade formando um quadrado. Portanto, tais células pertencem ao mesmo *cacho*, a menos dos nodos brancos. Isto significa que quatro células de nível $n+1$ são substituídas por uma única célula de nível n (Figura 2.5). Essas células devem ser oriundas do mesmo pai, o que garante a reversibilidade configuracional da malha sob as operações de refinamento e desrefinamento.

As etapas necessárias ao desrefinamento são as seguintes (Burgarelli et al., 2006):

1. Criação de um novo nodo preto;
2. Atualização das informações (espaciais, profundidade e etc.) do nodo preto criado;
3. Ligação do nodo preto com seus vizinhos;
4. Ligação dos vértices vizinhos com o nodo preto;
5. Ordenação da malha pela curva de Hilbert;
6. Liberação da área de memória dos nodos brancos e pretos que passam a não existir.

O passo (1) se dá pela transformação do vértice noroeste, em relação aos 4 vértices em questão no vértice resultante do desrefinamento. Assim como no processo de refinamento, o processo de desrefinamento pode acarretar a destruição de nodos brancos uma vez que nodos brancos de mesmo nível de profundidade não devem coexistir e nodos pretos filhos de mesmo pai devem ser conectados diretamente. Os casos de simplicação de nodos brancos podem ser verificados em (Burgarelli et al., 2006).

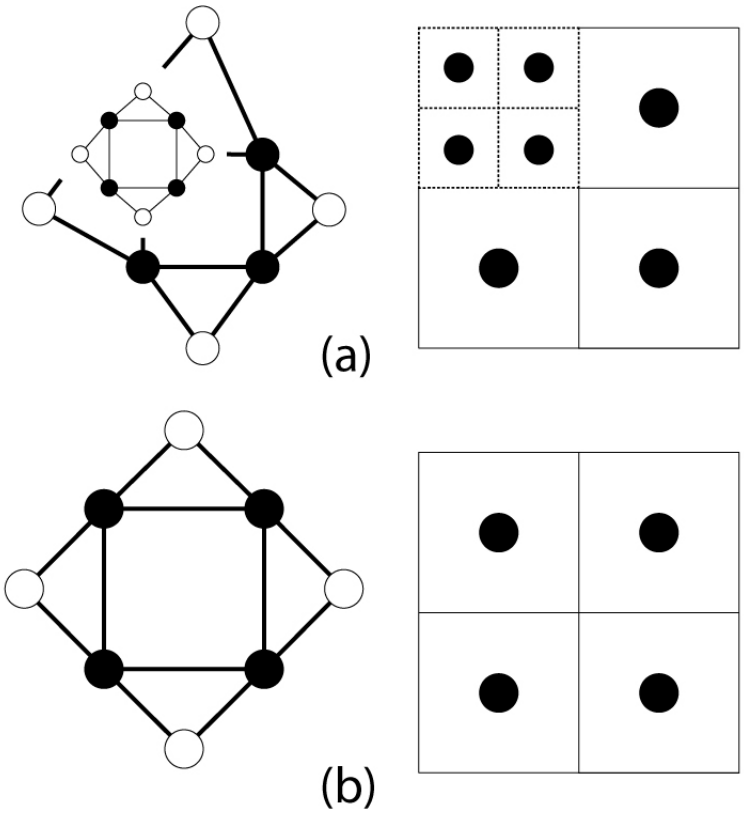


Figura 2.5: Esquema de desrefinamento.

O passo (5) apresentou um conceito ainda não discutido, a ordenação da malha. Em problemas que envolvem uma discretização de um problema contínuo é importante que exista uma forma de ordenar os elementos que surgem. Essa ordenação permite que seja empregado um método computacional para resolução do sistema linear obtido.

2.1.2 Ordenação da Malha

O ALG utiliza uma curva de preenchimento de espaço (*space-filling curve*) para garantir a ordenação da malha.

Uma curva de Hilbert é o limite de uma sequência de curvas $H_1, H_2, H_3, ...$, ilustradas na Figura 2.6, onde são os três primeiros passos da construção. Assim, uma curva de Hilbert de ordem i é dita uma aproximação da curva de Hilbert de preenchimento do espaço, portanto as curvas H_i são aproximações e não a curva em si. O processo de obtenção da curva de Hilbert de ordem i se dá pela substituição dos vértices de H_i por curvas de ordem $i-1$ devidamente rotacionadas ou refletidas. O algoritmo de construção da curva de Hilbert encontra-se disponível em (Burgarelli et al., 2006).

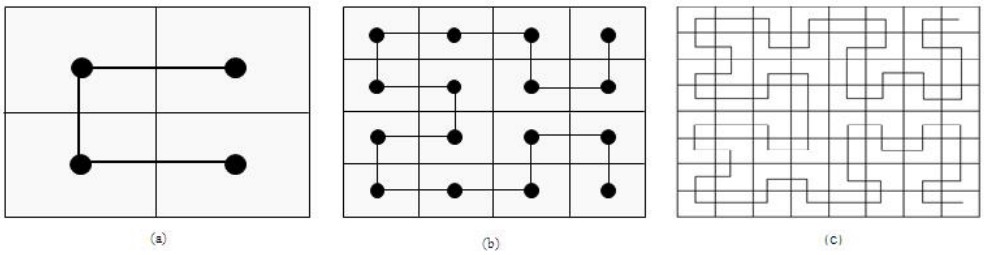


Figura 2.6: Sequência da curva de Hilbert bidimensional.

A estrutura definida anteriormente faz analogia com a curva de Hilbert para realizar uma ordenação da malha. Em cada nodo preto definido anteriormente existem mais dois ponteiros. Uma lista duplamente encadeada é utilizada para ligar todos os vértices pretos da estrutura. Toda vez que uma célula é refinada, um nodo preto é substituído na estrutura. Observe que a curva de Hilbert apresentada na realidade foi modificada, para não se restringir somente a refinamentos uniformes e, por isso, foi denominada curva de Hilbert Modificada (Figura 2.7).

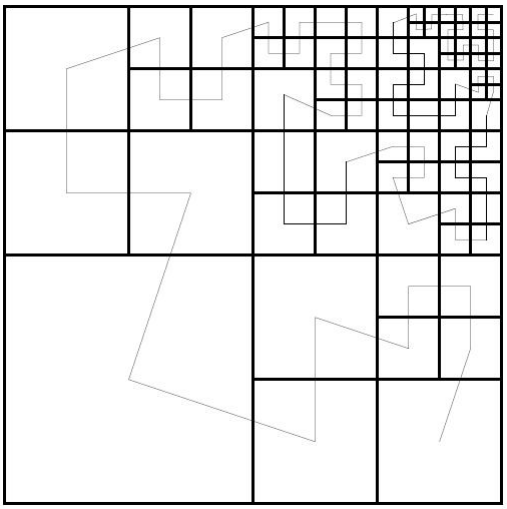


Figura 2.7: Curva de Hilbert Modificada bidimensional.

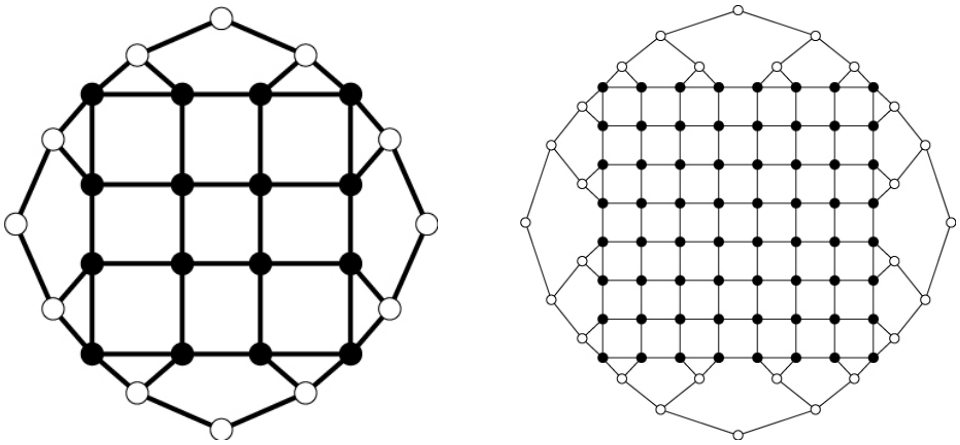
2.2 Análise do Custo de Armazenamento da Malha

A quantidade de memória necessária para o armazenamento da malha discretizada é determinada pela configuração final da malha. Todavia, em um problema adaptativo, a previsão dessa configuração é uma tarefa muito complexa. O processo de refinamento/desrefinamento de uma determinada célula ou região é determinado pelo programa de acordo com algum critério previamente escolhido (fluxo, energia, erro, etc).

Aqui são discutidas duas possibilidades para a configuração final da malha. No melhor caso, admite-se que todo refinamento ocorre de forma uniforme. Isto significa que todos os volumes estão no mesmo nível e consequentemente todos os nodos brancos foram simplificados e não estão presentes na estrutura de dados final, com exceção dos nodos brancos referentes ao contorno. O pior caso será dividido em dois problemas. No primeiro caso nenhum nodo branco será simplificado. No segundo, a partir de alguns refinamentos uniformes aplica-se a restrição do primeiro caso; isto significa que nenhuma simplicação de nodo branco será permitida a partir de um dado refinamento de nível k.

2.2.1 Refinamento Uniforme

Neste caso a discretização inicial da malha contém 4 volumes. A estrutura de dados representativa dessa malha possui 4 nodos pretos e 4 nodos brancos. Admite-se que a cada novo refinamento todos os volumes serão refinados, gerando 4 novos volumes. Isto significa que no nível 2 a estrutura passa a ter 16 nodos pretos e mais 8 nodos brancos, além dos 4 iniciais referentes ao contorno (Fig.2.8.a).



(a) Com 1 refinamento uniforme (b) Com 2 refinamentos uniformes

Figura 2.8: Estrutura do ALG com refinamentos uniformes

No próximo refinamento, referente ao nível 3, a estrutura terá 64 nodos pretos e, além dos 12 nodos brancos referentes ao contorno, mais 16 novos nodos brancos (Fig.2.8.b). Esse processo continua até o nível n conforme pode ser visto na Tabela 2.1.

Dessa forma, a malha terá um total de 2^{2n} nodos pretos e $\sum_{i=0}^{n-1} 2^i = 2^{n+2} - 4$ nodos brancos no n-ésimo nível. Totalizando são $2^{2n}+2^{n+2}-4$ nodos para representar uma malha com 2^{2n} volumes. Neste caso a quantidade de nodos pretos exibe taxa de crescimento (2^{2n}) maior que a de nodos brancos (2^n). O custo computacional do método é determinado pelo

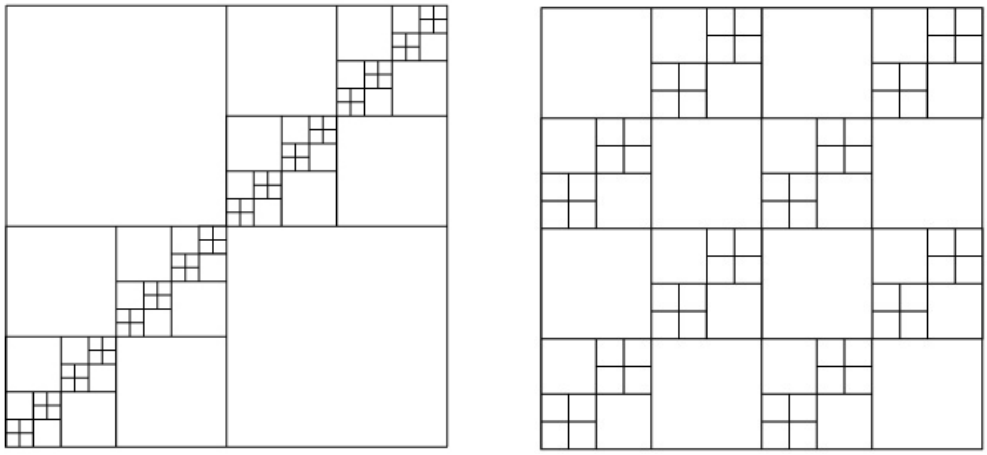
Tabela 2.1: Quantidade de nodos para o caso de refinamentos uniformes no ALG.

Nível	Nodos Pretos	Nodos Brancos
1	$4 = 2^2$	$4 = 2^2$
2	$16 = 2^4$	$12 = 2^2 + 2^3$
3	$64 = 2^6$	$28 = 2^2 + 2^3 + 2^4$
4	$128 = 2^8$	$60 = 2^2 + 2^3 + 2^4 + 2^5$
...	...	...
n	2^{2n}	$\sum_{i=1}^{n} 2^{i+1}$

número de volumes na malha, além do método numérico empregado.

2.2.2 Refinamento Adaptativo

Para que não ocorram simplificações dos nodos brancos em nenhum momento do refinamento seria necessário que o refinamento ocorresse sempre em uma das diagonais do domínio. Considere o refinamento hipotético da Figura 2.9.a.



(a) Malha Refinada sem Simplificação de Nodos Brancos.

(b) Malha Refinada com uma Simplificação dos Nodos Brancos devido a um Refinamento Uniforme.

Figura 2.9: Malhas Refinadas Adaptativamente a partir da diagonal

Para o primeiro caso, como os refinamentos só ocorrem na diagonal principal, a quantidade de nodos pretos para o nível 2 é dada pelos 2 elementos do nível 1 que não foram refinados, mais os 4 novos elementos que surgiram. Já a quantidade de nodos brancos é dada pelos 4 nodos brancos do nível 1 referentes ao contorno, mais 8 que sugiram pela substituição de 2 nodos pretos do nível 1 por dois *cachos* de nível 2. Esse procedimento

continua para o próximo nível e as quantidades de nodos brancos e pretos são apresentadas na Tabela 2.2.

Tabela 2.2: Quantidade de nodos no caso de refinamentos não uniformes para o ALG.

Nível	Nodos Pretos	Nodos Brancos
1	$4 = 2^2$	$4 = 2^2$
2	$2 + 8$	$4 + 8$
3	$2 + 4 + 16$	$4 + 8 + 16$
4	$2 + 4 + 8 + 32$	$4 + 8 + 16 + 32$
...	...	...
n	$2^{n+1} + \sum_{i=1}^{n-1} 2^i$	$\sum_{i=2}^{n+1} 2^i$

Dessa forma, o número total de nodos pretos na malha é dado por $2^{n+1} + 2^n - 2$ enquanto o número de nodos brancos é dado por $2^{n+2} - 3$. Assim, observa-se que o crescimento dos nodos brancos ocorre a uma taxa superior à de nodos pretos.

Esse é um exemplo onde o número de nodos brancos é grande, todavia existem casos onde esse crescimento é ainda maior. Para tanto, deve-se avaliar a configuração inicial do problema. A Figura 2.9.b representa uma situação hipotética em que a malha sofreu 1 refinamento uniforme e posteriormente os refinamentos ocorreram somente nas diagonais do domínio. A Figura 2.10 apresenta a estrutura do ALG referente ao refinamento que acaba de ser descrito.

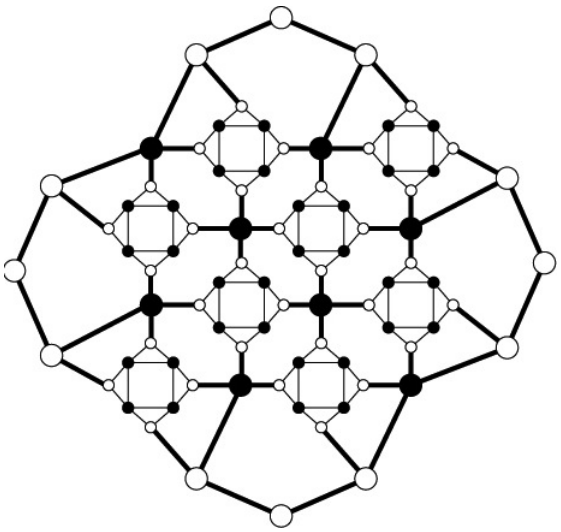


Figura 2.10: Estrutura do ALG com 1 refinamento uniforme seguido por 1 refinamento sem simplificação de nodos brancos.

A Tabela 2.3 apresenta os resultados quando ocorrem refinamentos uniformes seguidos de refinamentos adaptativos.

Tabela 2.3: Quantidade de nodos no caso de k refinamentos uniformes seguidos de $n-k$ refinamentos não uniformes para o ALG.

Nível	Nodos Pretos	Nodos Brancos
1	$4 = 2^2$	$4 = 2^2$
2	16	$4 + 8$
3	64	$4 + 8 + 16$
...	...	...
k	2^{2k}	$\sum_{i=1}^{k} 2^{i+1}$
$k+1$	$2^{2k-1} * 10$	$\sum_{i=1}^{k} 2^{i+1} + 2^{2k+1}$
...	...	...
n	$2^{2k-1} * (5 * 2^{n-k})$	$\sum_{i=1}^{k} 2^{i+1} + 2^{n+k+1} - 2^{2k+1}$

Assim, para representar $2^{2k-1} * (5 * 2^{n-k})$ nodos pretos a malha apresenta $2^{k+2} - 4 + 2^{n+k+1} - 2^{2k+1}$ nodos brancos. Analisando o crescimento dessa função em relação a n percebe-se que o máximo ocorre quando o número de refinamentos uniformes k é igual a $n-1$ e somente o último refinamento é realizado nas diagonais do domínio discretizado. Apesar do número excessivo de nodos brancos, o custo de armazenamento destes é menor do que o custo de armazenamento dos nodos pretos. Esse baixo custo de armazenamento dos nodos brancos é que viabiliza a utilização do ALG. A característrica mais atrativa dele é a simplicidade do algoritmo de busca de vizinhos quando comparado com as *quadtrees* tradicionais. Por isso, uma estratégia de redução da quantidade de nodos brancos é definida a seguir.

2.2.3 Modificação do ALG

Como analisado, o custo de armazenamento dos nodos de transição é elevado. No caso dos nodos brancos referentes ao contorno do domínio, eles acabam perdendo sua finalidade de indicar a transição de nível entre os elementos e passam a simplesmente ocupar memória de maneira desnecessária. A análise apresentada na seção anterior demonstra que o consumo para estes nodos especificamente é de 2^{2n} onde n é o nível do grafo.

Uma simplificação na estrutura pode ser feita substituindo os nodos brancos de con-

torno por um único nó especial, aqui denominado de nodo amarelo. Ele possui 3 informações: o nível, o tipo (amarelo) e um único ponteiro (*singleconnector*) que aponta para NULL. Todos os nodos pretos do contorno apontam para ele. Para utilizar esse nodo basta uma simples modificação na fase de refinamento. Essa fase apresenta um momento de simplificação dos nodos brancos, onde nodos brancos que ligam elementos de mesmo nível são simplificados (ver, por exemplo, Burgarelli et al. (2006)). Nessa fase de simplificação, verifica-se se o nodo branco em questão está ou não no contorno. Em caso afirmativo, ele pode ser apagado da malha e seus vizinhos (nodos pretos) são ligados ao nodo amarelo.

O Algoritmo 1 descreve o procedimento apresentado. Observe que a variável *Con* representa a célula branca que está sendo analisada, enquanto que a variável *V* representa a célula vizinha dela. As propriedades das células brancas *d1 e d2* são os ponteiros já descritos na Figura 2.2.

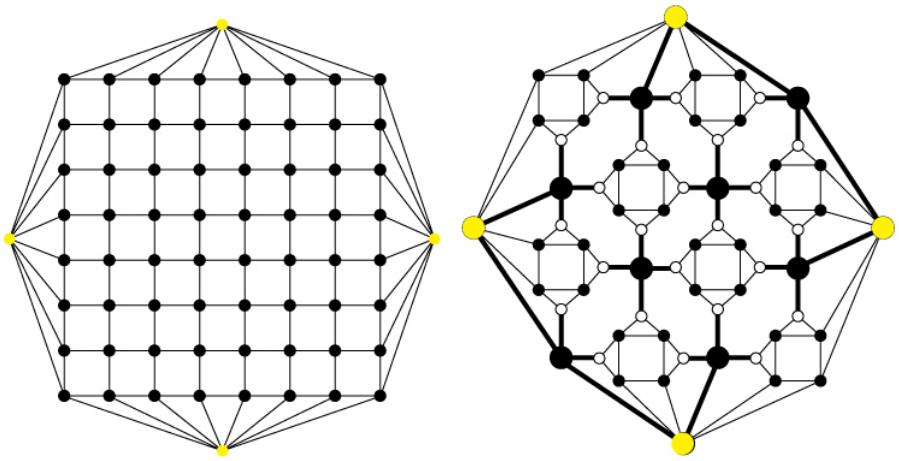
Algoritmo 1: Algoritmo para Simplificação do Contorno

SimplificacaoContorno(Célula *Con, Célula *V, Char Direcao)

```
1: if (Y.tipo =="amarelo") then
2:    if (Direcao=="L") then
3:       Con.d1.leste = V;
4:       Con.d2.leste = V;
5:    end if
6:    if (Direcao=="O") then
7:       Con.d1.oeste = V;
8:       Con.d2.oeste = V;
9:    end if
10:   if (Direcao=="N") then
11:      Con.d1.norte = V;
12:      Con.d2.norte = V;
13:   end if
14:   if (Direcao=="S") then
15:      Con.d1.sul= V;
16:      Con.d2.norte = V;
17:   end if
18:   delete X
19: end if
```

Com essa pequena modificação ao final dos refinamentos a estrutura de dados possuirá

apenas 4 nodos amarelos representando o contorno em lugar dos 2^{n+2} nodos brancos da estrutura original. A Figura 2.11 apresenta a estrutura ao final das simplificações. Essa modificação apresenta um ganho significativo de memória.



(a) Refinamentos Uniformes (b) Refinamentos Adaptativos

Figura 2.11: Estrutura de dados com modificação dos nodos brancos do contorno.

2.3 Algoritmo de resolução utilizando o ALG

O Algoritmo 2 apresenta uma estratégia que utiliza o ALG para resolver um problema. Apesar de ser uma versão simplificada ele representa os passos principais da estratégia utilizada para resolução de problemas diferenciais evolutivos. O algoritmo constrói uma matriz dita esparsa. Por isso, só são armazenados os elementos não nulos, o que permite uma redução consideravelmente do consumo de memória. Já a fase de resolução do sistema linear é descrita na próxima sessão. A fase de refinamento e desrefinamento é baseada nas estratégias já descritas para o ALG.

Algoritmo 2: Algoritmo para Solução Adaptativa utilizando o ALG

```
1: Inicialização da Malha;
2: Definição das condiçõe Iniciais;
3: while (tempo<tempoFinal) do
4:     Definição da Matriz do Problema;
5:     Resolução do Sistema Linear;
6:     Refinamento-Desrefinamento;
7:     Atualização do tempo;
8:     Atualização das Condições Iniciais;
9: end while
```

A Tabela 2.4 apresenta o percentual do tempo gasto pela fase de refinamento e desrefinamento em relação ao tempo total de simulação para o problema de propagação de calor. Observe que o tempo necessário para essa fase é menor quando comparado as fases de definição do problema e resolução do sistema linear. Segundo Burgarelli et al. (2006), a fase do ALG que mais consome tempo é a resolução do sistema linear.

Tabela 2.4: Tempo de execução do ALG para o problema da equação do calor. Adaptado de (Burgarelli et al., 2006)

Condição de Contorno	Nível	Células	Tempo Total	Perc. TempoRef.
$g(x,y) = x^2 - y^2$	6	3529	4.7	2.0
	7	12442	28.6	0.7
	8	55414	169.9	0.8

2.3.1 Resolução do Sistema Linear

A matriz obtida pelo esquema de Volumes Finitos utilizado pelo ALG é simétrica e diagonal dominante no caso de problemas elípticos e parabólicos, conforme demonstrado em (Burgarelli et al., 2006). Além disso, todos os elementos da diagonal principal são positivos e os demais, negativos. Uma matriz com essas propriedades caracteriza-se como uma matriz positiva-definida (Briggs et al., 2000).

Para este tipo de matriz o método dos Gradientes Conjugados (MGC) é considerado um dos mais eficientes (Golub e Van Loan, 1996; Burden e Faires, 2004). Ele realiza uma minimização do funcional quadrático definido pelo problema por meio de uma busca que ocorre nas direções conjugadas do resíduo (Golub e Van Loan, 1996).

O Algoritmo 3 apresenta o MGC de forma detalhada. Os vetores p_i representam as direções conjugadas dos resíduos r_i para cada iteração i realizada pelo algoritmo, ou seja, eles determinam a direção de busca do algoritmo.

Algoritmo 3: Algoritmo do Método dos Gradientes Conjugados

estimativa inicial x_0
cálculo do resíduo inicial $r_0 = b - Ax_0$
$k = 1$
cálculo da direção da busca $p_1 = r_0$
cálculo do passo na direção de busca $\alpha_1 = \frac{(r_0^t r_0)}{p_1^t A p_1}$
próximo valor da solução $x_1 = x_0 + \alpha_1 p_1$
novo valor do resíduo $r_1 = r_0 - \alpha_1 A p_1$
while resíduo $r_k \neq 0$ **do**
    número de iterações $k = k + 1$
    $\beta_k = \left(\frac{r_{k-1}^t r_{k-1}}{r_{k-2}^t A r_{k-2}}\right)$
    cálculo da direção de busca $p_k = r_{k-1} + \beta_k p_{k-1}$
    cálculo do passo na direção de busca $\alpha_k = \left(\frac{r_{k-1}^t r_{k-1}}{r_k^t A r_k}\right)$
    atualização da aproximação $x_k = x_{k-1} + \alpha_k p_k$
    atualização do resíduo $r_k = r_{k-1} - \alpha_k A p_k$
end while
solução $x = x_k$

A convergência do MGC pode ser obtida em uma quantidade fixa de passos. Essa quantidade é definida pelo número de autovalores distintos presentes na matriz de discretização (Golub e Van Loan, 1996). Todavia, os erros de arredondamento influenciam nesse processo e, por isso, a velocidade de convergência é dada por Eq. 2.1.

$$\rho = \frac{2}{(1 + 2(K-1)^{\frac{1}{2}})^K} \tag{2.1}$$

onde K é o número de condicionamento da matriz de discretização, dado por:

$$K = cond(A) = \frac{\max\{|\lambda| : \lambda \in \sigma(A)\}}{\min\{|\lambda| : \lambda \in \sigma(A)\}}$$

com $\sigma(A)$ o conjunto de autovalores da matriz A.

A análise da Eq. 2.1 demonstra que a velocidade de convergência do MGC quando o número de condicionamento da matriz se afasta do valor 1 sofre uma redução. Neste caso o problema matricial é dito ser mal condicionado.

Essa análise precisa ser realizadas nas matrizes de discretização obtidas na estratégia empregada pelo ALG. Caso tais matrizes sejam mal condicionadas, métodos mais eficientes devem ser utilizados no lugar do MGC.

2.3.2 Análise Computacional da Matriz de Discretização

Os problemas abordados neste trabalho consistem em problemas eliptícos e parabólicos. Eles consistem basicamente na discretização do laplaciano pelo método dos Volumes finitos. No caso da Equação do Calor, a informação referente à discretização do termo evolutivo é acrescida à matriz discretizada do laplaciano. Os problemas a seguir apresentam os autovalores e número de condicionamento dessas matrizes.

Problema 1 Considere o problema descrito por:

$$\nabla^2 u = 0 \qquad \Omega = [0,1] \times [0,1] \tag{2.2}$$

com condições de contorno dadas por:

$$u(x,0) = u(0,y) = 0$$

$$u(x,1) = x + 1$$

$$u(1,y) = 1 + y$$

A solução exata deste problema é dada por $u(x,y) = x + y$.

A Figura 2.12 apresenta o comportamento do número de condicionamento das matrizes do problema 1. Após sucessivos refinamentos, as matrizes ficam progressivamente mais mal-condicionadas. Como explicado, tal comportamento prejudica a velocidade de convergência do MGC.

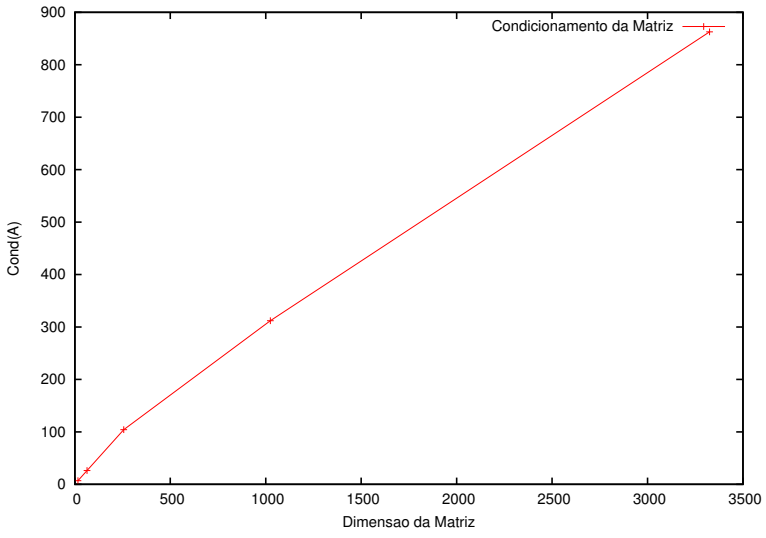


Figura 2.12: Comportamento do número de condicionamento da matriz do Problema 1 com diferentes níveis de refinamento.

A Figura 2.13 apresenta os autovalores deste problema para uma malha com 3328 células e critério de refinamento de 0.00725. Apesar de os autovalores estarem espaçados no intervalo no qual estão contidos, percebe-se a formação de alguns grupos. Esse comportamento tende a reduzir ainda mais a velocidade de convergência do MGC (Forgsgren, 2006; Liu, 2012).

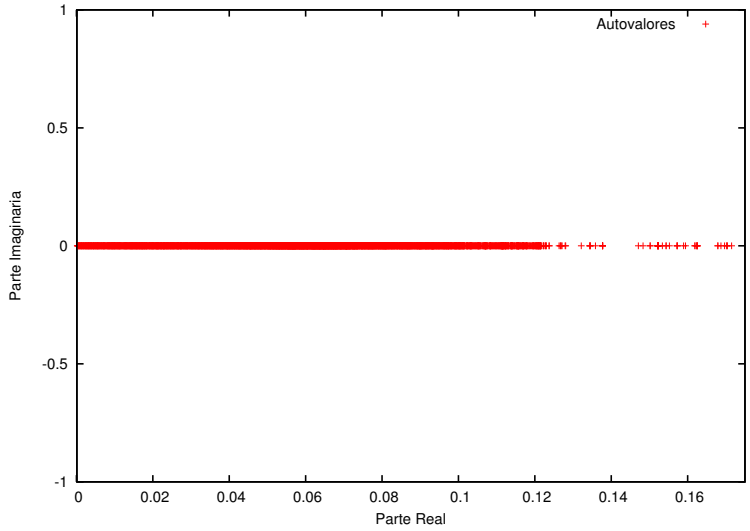


Figura 2.13: Espectro dos autovalores da matriz do Problema 1.

A Figura 2.14 apresenta os autovalores das matrizes dos diferentes níveis de refinamento do Problema 1. Observe que o valor dos autovalores vai diminuindo de acordo com o aumento do número de volumes no domínio. Esse comportamento pode ser analisado devido a suavização das componentes que eram de alta frequência na malha menos refinada e passam a ser de baixa frequência na mais refinada ou como consequência da caracterização minimax para os autovalores do laplaciano (Jost, 2002).

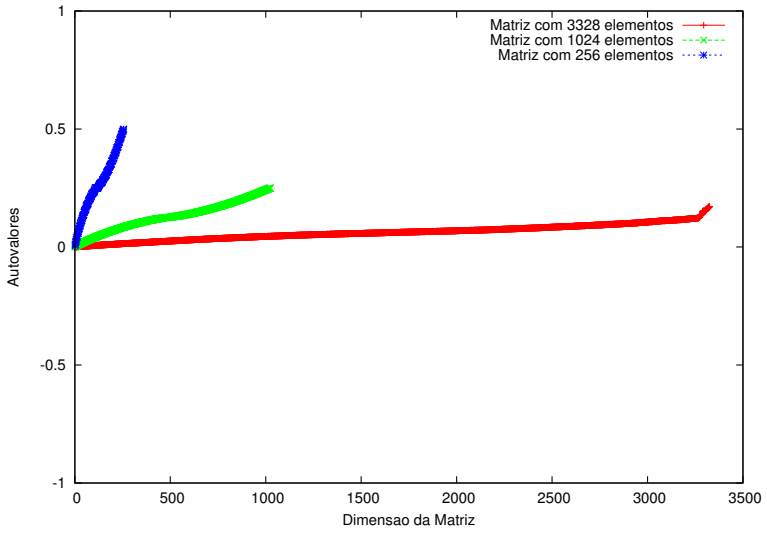


Figura 2.14: Comparação dos autovalores com diferentes níveis de refinamento.

Problema 2 Considere o problema de condução do calor no caso bidimensional:

$$u_t = \nabla^2 u \qquad (x, y) \in \Omega = [0, 1] \times [0, 1] \tag{2.3}$$

Com condições de contorno e inicias dadas por:

$$u(x, y, t) = 10 \qquad \forall (x, y) \in \partial\Omega \quad t > 0$$

$$u(x, y, 0) = 0$$

A Figura 2.17 apresenta o comportamento do condicionamento das matrizes que surgem em cada nível de refinamento do Problema 2. O problema contém 4096 células que foram refinadas de acordo com um critério de 0.00125 com 10 instantes de tempo para $\Delta t = 0.08$. Assim como no problema anterior, a medida que a malha se torna mais refinada maior é o seu número de condicionamento e mais lenta é a convergência do MGC.

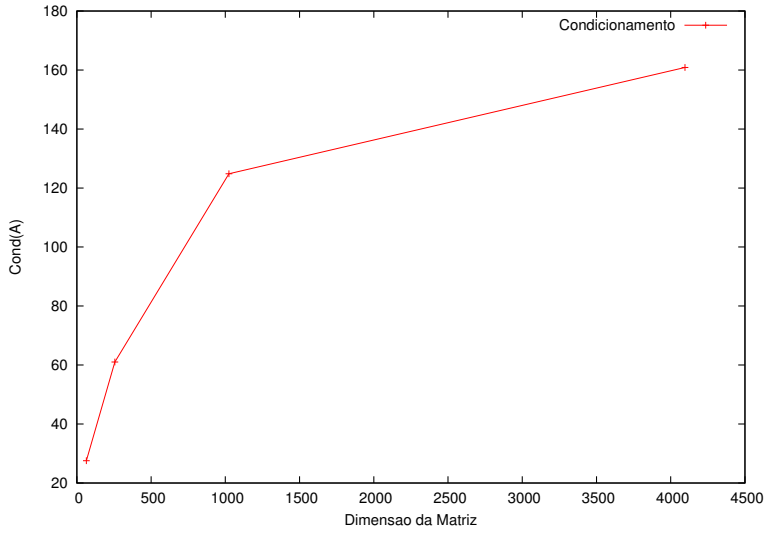


Figura 2.15: Espectro dos autovalores do problema 2 com 4096 elementos.

Os autovalores se comportaram como no problema anterior. À medida que a malha era refinada eles eram atenuados. A Figura 2.16 apresenta esse comportamento. O número de iterações do MGC aumenta consideravelmente. Isso ocorre devido a formação de grupamentos dos autovalores (Forgsgren, 2006; Liu, 2012). A Figura 2.16 apresenta os espectros dos autovalores no caso de 4096 elementos, podem ser observados 3 grupos de autovalores.

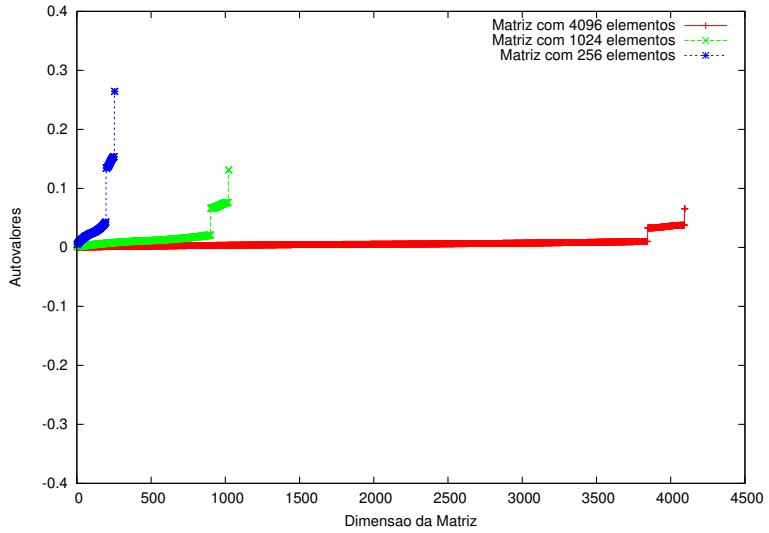


Figura 2.16: Comportamento do número de condicionamento da matriz do Problema 2 com diferentes níveis de refinamento.

Como pode ser observado nos problemas discutidos, a abordagem utilizada pelo ALG por meio do método dos Volumes Finitos apresenta matrizes mal-condicionadas. Tal com-

portamento é comum em matrizes oriundas do MVF (Shi et al., 2004) e prejudica a convergência do MGC. No caso de outras estratégias adaptativas esse mal-condicionamento pode se controlado. Por exemplo, no caso do método dos elementos finitos em malhas triangulares basta que os triângulos sejam equiláteros ou bem próximos disso para que o mal-condicionamento seja atenuado (Roseberg e Stenger, 1975).

No caso do MVF, duas soluções são possíveis para resolver um sistema mal-condicionado. A primeira seria a utilização de pré-condicionadores. O sistema obtido pela aplicação do MVF é modificado por meio de uma matriz de pré-condicionamento gerando um sistema linear equivalente ao primeiro, todavia com um número de condicionamento melhor que o primeiro (Axelsson, 1996).

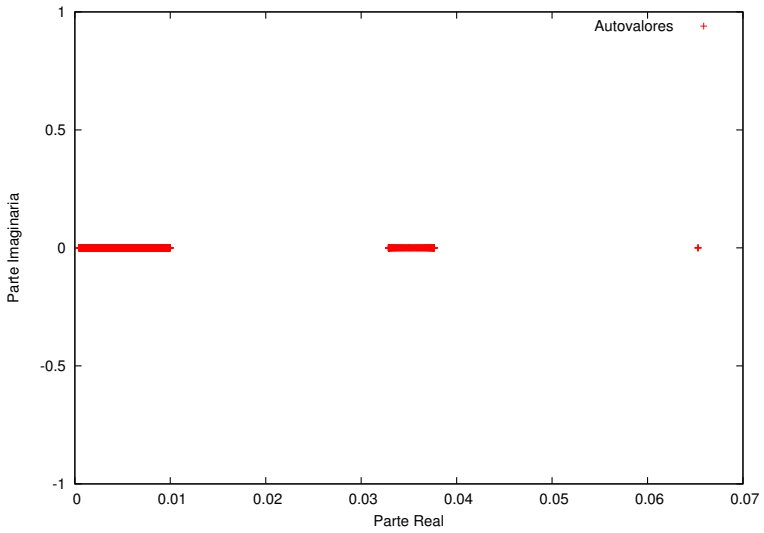


Figura 2.17: Espectro dos autovalores do problema 2 com 4096 elementos.

A segunda seria a utilização de métodos que utilizam estratégias baseadas em múltiplas malhas, os chamados métodos Multigrid. Esse métodos apresentam um comportamento assintótico ótimo. Neste trabalho optou-se pela utilização da segunda, visto sua eficiência e eficácia para resolução de problemas mal-condicionados oriundos da discretização pelo MVF (Shi et al., 2004).

Capítulo 3

Aspectos do Método Multigrid

Neste capítulo são apresentados os conceitos fundamentais para a compreensão do método Multigrid, tanto em sua versão geométrica quanto algébrica. Em ambos os casos, são discutidos alguns dos elementos que os compõem: a equação residual, a propriedade de suavização dos métodos iterativos, os operadores de restrição e prolongamento. Por fim, são apresentados os algoritmos implementados nesta tese para ambas as abordagens.

3.1 Correção de Defeitos

Considere o problema de valor de contorno dado pela Equação 3.1.

$$-U^{''}(x) + \sigma U(x) = f(x), \quad x \in \Omega \subset \Re^2, \sigma \geq 0 \quad U(x) = 0, \quad x \in \partial\Omega \tag{3.1}$$

Apesar de esse problema específico poder ser resolvido analiticamente, aqui ele será tratado por meio de métodos numéricos. Muitas abordagens poderiam ser escolhidas, tendo-se optado aqui pelo método dos volumes finitos, devido à integração de tal método com a estrutura de dados que será utilizada. Ademais, é um dos métodos mais apropriados para a representação de leis de conservação.

A ideia fundamental do método dos volumes finitos é que o problema contínuo seja particionado em intervalos menores denominados volumes de controle, a união de todos os volumes de controle constitui a malha computacional. Em cada um desses volumes de controle são empregadas as leis de conservação. Surge assim um sistema de equações lineares, sendo as incógnitas as grandezas de interesse (pressão, massa, densidade, etc), estipuladas no centro de cada um dos volumes de controle da malha computacional. Para

mais detalhes consulte (Maliska, 2004). Assim, a partir da discretização do problema 3.1 obtém-se um sistema linear dado pela Equação 5.1.

$$Au = f \tag{3.2}$$

Suponha que o Sistema 5.1 tenha solução única e que v seja uma aproximação da solução discreta u. Define-se o erro ou erro algébrico pela Equação 3.3.

$$e = u - v \tag{3.3}$$

Entretanto, pode-se observar que a definição do erro depende da solução do problema, a qual geralmente é desconhecida. Computacionalmente adota-se uma estratégia para estimar o quanto longe a solução encontrada está da solução real do problema. Tal estratégia define o resíduo do problema (Equação 3.4).

$$r = f - Av \tag{3.4}$$

O resíduo pode ser entendido como o quanto a solução v falha ao tentar satisfazer o problema $Au = f$. Pela unicidade da solução tem-se que $r = 0$ se, e somente se, $e = 0$. Pelas definições do erro (Equação 3.3) e do resíduo (Equação 3.4) pode-se definir uma relação entre eles denominada equação residual (Equação 3.5).

$$Ae = r \tag{3.5}$$

Essa relação mostra que o erro satisfaz o mesmo conjunto de equações que a solução u quando o vetor f é substituído pelo resíduo r. De maneira geral, suponha que uma solução aproximada v tenha sido obtida na solução do sistema linear 3.2. A partir dessa aproximação obtém-se o resíduo $r = f - Av$. E assim uma atualização para a solução aproximada pode ser realizada, sendo somente necessário calcular e a partir da equação residual e por fim, calculando $u = v + e$.

Esse processo é denominado processo de correção de defeito, sendo utilizado pelo método Multigrid. Entretanto, torna-se necessário entender um pouco o funcionamento dos métodos iterativos clássicos que também são utilizados pelo Multigrid.

3.2 Suavizadores

O sistema 5.1 que surgiu após a discretização da equação diferencial poderia ser resolvido por um método iterativo estacionário (método de Gauss-Seidel, Gauss-Jacobi, etc). Entretanto, observou-se que a convergência de tais métodos torna-se lenta à medida que a dimensão da matriz aumenta (Briggs et al., 2000). A Figura 3.1 demonstra que a partir de um certo número de iterações do método iterativo clássico a convergência para a solução do sistema linear, definido em (Briggs et al., 2000) pela discretização de um problema de Poisson pelo método das diferenças finitas, torna-se cada vez mais lenta.

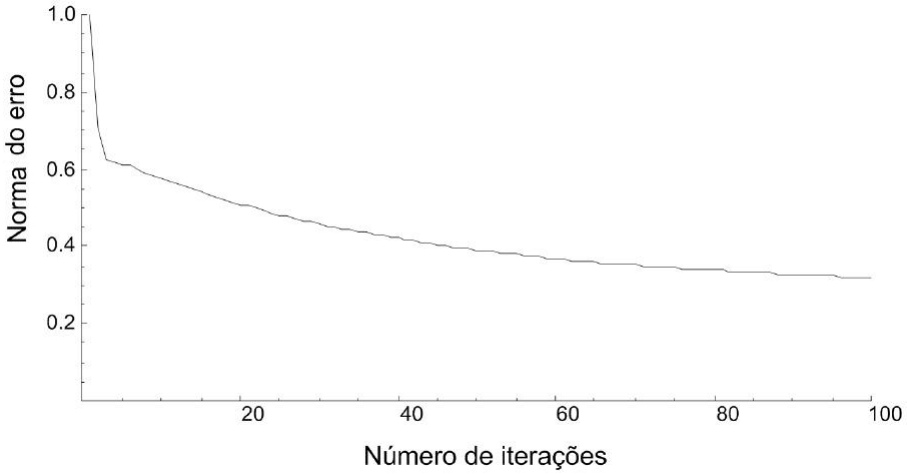


Figura 3.1: Norma do máximo do vetor erro para 100 primeiras iterações do método de Gauss-Jacobi (Briggs et al., 2000).

É possível demonstrar que os autovalores e autovetores da matriz de discretização são os mesmos que os da matriz de iteração do método iterativo estacionário (Briggs et al., 2000). Com isso obtém-se uma expressão para o erro onde as componentes de alta frequência e de baixa-frequência estão bem definidas. Pode-se observar que, conforme o número de iterações do método aumenta, as componentes de alta-frequência têm sua amplitude reduzida consideravelmente, enquanto as de baixa frequência são muito pouco afetadas (Fedorenko, 1964). Assim, os métodos iterativos estacionários são conhecidos como suavizadores, pois eles suavizam o erro eliminando componentes de alta frequência, conforme pode ser observado na Figura 3.2. Uma importante propriedade de tais métodos é que o fator de suavização independe do espaçamento entre os pontos da discretização.

A ordem de convergência dos métodos iterativos estacionários é proporcional ao espaçamento da malha (h^2) do problema discretizado (Tabela 3.1). Dessa forma, torna-se-ia interessante reduzir o tamanho de h para melhorar a convergência de tais métodos. Entretanto isso não é observado de fato, o que acontece é que os autovalores associados

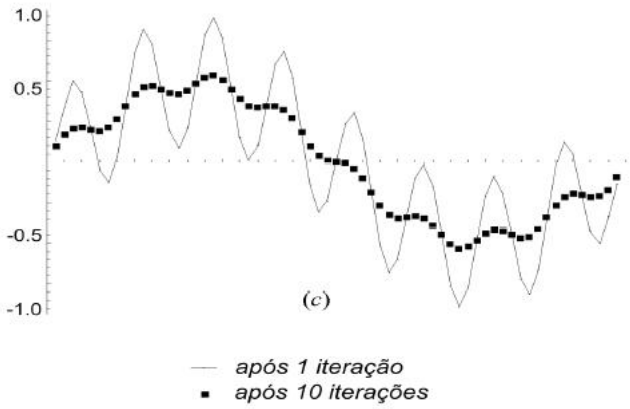


Figura 3.2: Suavização do erro do problema definido em (Briggs et al., 2000) por meio do método de Gauss-Jacobi.

ao erro se aproximam de 1 e a ordem de convergência é degradada. Quando o valor de h é aumentado os valores dos autovalores associados às componentes de baixa frequência do erro são reduzidos permitindo que estes sejam minimizados pelo método iterativo estacionário(Briggs et al., 2000).

Assim, os métodos iterativos estacionários são utilizados de forma fundamental no contexto do método Multigrid. Pode-se sintetizar esta utilização de duas formas:

1. suavização do erro por meio de um método estacionário;
2. modificação do espaçamento da malha, aumentando o valor de h, para novamente aplicar o método iterativo, conseguindo atingir os autovalores desejados.

Esses dois passos são aplicados de forma irrestrita até que o erro seja minimizado da forma desejada. O item 2 torna necessária a definição dos operadores de transferência, isto é, operadores que permitem a reescrita do problema discreto de uma malha menos refinada para uma malha mais refinada e vice-versa.

3.3 Operadores de Transferência

Esta seção define os operadores de transferência: prolongamento e restrição. Para tanto, definem-se duas malhas: Ω_h uma malha refinada com espaçamento h e Ω_{2h} uma malha menos refinada que Ω_h, dita malha mais grosseira, na qual o espaçamento é de $2h$.

O operador de transferência entre a malha mais refinada Ω_h e a malha menos refinada Ω_{2h} é conhecido como operador de restrição e definido de acordo com a função 3.6.

$$I_h^{2h} : \Re(\Omega_h) \rightarrow \Re(\Omega_{2h}) \tag{3.6}$$

O operador de prolongamento, definido como o operador que leva os pontos da malha em Ω_{2h} para a malha Ω_h, é dado por 3.7.

$$I_{2h}^{h} : \Re(\Omega_{2h}) \rightarrow \Re(\Omega_h) \tag{3.7}$$

Os operadores de restrição mais conhecidos e utilizados são os operadores de injeção e o *full weighting*. No primeiro os valores dados em uma malha mais refinada são apenas transferidos para as respectivas posições na malha menos refinada. Já o operador *full weighting* calcula uma média de acordo com os vizinhos na malha mais refinada e leva esse valor para o termo equivalente na malha menos refinada (Trottenberg et al., 2001).

O operador de prolongamento consiste numa interpolação em que, a partir dos valores definidos nos pontos na malha menos refinada são calculados os valores de todos os pontos da malha mais refinada. O operador mais utilizado é o operador de interpolação bilinear, pois ele é uma transposição do operador de restrição *full weighting*. Dessa forma, basta armazenar um dos operadores e, por meio do cálculo da matriz transposta obtém-se o outro operador. Essa manipulação algébrica permite um aproveitamento eficiente do código na implementação do Multigrid, pois requer menos memória no armazenamento dos operadores.

Dispõe-se, portanto, das técnicas necessárias para a construção do método Multigrid.

3.4 Método Multigrid Geométrico

Este método é considerado o mais rápido na solução de sistemas lineares ou não-lineares esparsos oriundos da discretização de equações diferenciais. A complexidade do método denominado FMG (*Full Multigrid*), por exemplo, pode chegar em alguns casos a $O(N)$, conforme demonstrado na Tabela 3.1, onde N é o total de incógnitas do sistema.

Conforme observado anteriormente, os métodos de suavização conseguem atenuar as componentes de alta frequência do erro de maneira rápida enquanto as de baixa frequência são pouco alteradas.

Assim o método Multigrid na sua versão *TwoGrid* usa duas malhas para obter a solução do sistema, uma mais refinada onde define-se uma aproximação para a solução do

Tabela 3.1: Comparação de complexidade de diferentes métodos para resolução de sistemas lineares com precisão ϵ

Método	Número de Operações
Eliminação Gaussiana	O(N^2)
Métodos Iterativos Gauss-Seidel ou Jacobi	O($N^2 log\epsilon$)
Successive Over Relaxation (SOR)	O($N^{\frac{3}{2}} log\epsilon$)
Método dos Gradientes Conjugados	O($N^{\frac{3}{2}} log\epsilon$)
Método Multigrid Iterativo	O($N log\epsilon$)
Multigrid (FMG)	O(N)

problema e o resíduo dessa aproximação. Posteriormente, tal resíduo é transportado para uma malha mais grosseira, em que as componentes que eram de baixa frequência passam a ser de alta frequência e então um esquema de suavização é aplicado até que uma solução seja encontrada. Por fim, a solução é transferida para a malha mais refinada. Neste ponto um operador de prolongamento é aplicado, e a solução aproximada inicialmente definida na malha mais refinada é atualizada com a contribuição oriunda da malha mais grosseira.

O algoritmo *TwoGrid* pode ser observado no Algoritmo 4. Trata-se do método de correção sendo aplicado a duas malhas.

Algoritmo 4: Algoritmo do TwoGrid.

1: Aplique γ_1 passos da relaxação em $A^h u^h = f^h$ com estimativa inicial v_0;
2: Calcule $r^h = f^h - A^h v^h$ e $r^{2h} = I_h^{2h} r^h$;
3: Resolva $A^{2h} e^{2h} = r^{2h}$ em Ω^{2h};
4: Calcule $e^h = I_{2h}^h e^{2h}$
5: Corrija a solução aproximada v^h em Ω^h da forma $v^h \leftarrow v^h + e^h$
6: Aplique γ_2 passos da relaxação em $A^h u^h = f^h$ com estimativa inicial v_h

Conforme pode ser observado no Algoritmo 4 as quantidades de passos de suavização γ_1 e γ_2 não são conhecidas, pois o vetor erro é desconhecido, assim como a solução do problema. Na prática, os valores para γ_1 e γ_2 são 1, 2 ou 3.

A matriz A^h é obtida pela discretização em volumes finitos, enquanto a matriz A^{2h} pode ser obtida aplicando-se os operadores de transferência sobre A^h conforme demonstrado em (Briggs et al., 2000). Os operadores de prolongamento e restrição são de acordo com os definidos na Seção 3.3, e sua escolha deve-se tanto à simplicidade quanto à qualidade da solução produzida por eles (Giraud et al., 2003).

A etapa final, que consistiria na resolução da equação residual em Ω^{2h} pode ser feita de diferentes maneiras. Um solução seria a aplicação de um método direto. Entretanto, isso pode ter um alto custo computacional dependendo da dimensão do problema. Outra solução seria a utilização de um método estacionário, nesse caso a qualidade da correção torna-se dependente tanto da interpolação quanto da precisão obtida para a solução da equação residual em Ω^{2h}.

Algoritmo 5: Algoritmo do Esquema V-Ciclo

$\mathbf{V}^h(v^h, f^h)$

1: Aplique γ_1 passos da relaxação em $A^h u^h = f^h$ com estimativa inicial v_h
2: **if** Ω^h = malha menos refinada **then**
3: Vá para o passo 10
4: **else**
5: $f^{2h} \leftarrow I_h^{2h}(f^h - A^h v^h)$
6: $v^{2h} \leftarrow 0$
7: $v^{2h} \leftarrow V^{2h}(v^{2h}, f^{2h})$
8: **end if**
9: Prolonga $v^h \leftarrow I_{2h}^h v^{2h}$
10: Aplique γ_2 passos da relaxação em $A^h u^h = f^h$ com estimativa inicial v_h;

Uma terceira solução seria levar o problema para uma malha menos refinada, por exemplo, Ω^{4h} e tentar resolver o problema nessa malha por meio do Algoritmo 4. Observe que isso introduz uma nova estratégia denominada grids aninhados. Nesta, novas malhas mais grosseiras são criadas recursivamente até que o problema possa ser resolvido por um método direto em uma malha Ω^{2ph}, com $p > 1$. São esses conjuntos de operações com suavizadores, operadores de transferência e diferentes níveis de malhas que caracterizam o método Multigrid (MG).

O esquema mais conhecido do MG é o denominado V-ciclo, sendo este um caso particular dos algoritmos chamados W-Ciclos (Trottenberg et al., 2001). O V-ciclo é representado pelo Algoritmo 5 em versão recursiva. Essa nomenclatura é devida ao padrão V determinado pelas operações sucessivas de interpolação e restrição.

A Figura 3.3 apresenta um exemplo do Multigrid para o caso bidimensional, as setas no sentido descendente representam a restrição enquanto no sentido ascendente representam o processo de interpolação ou prolongamento. Outros tipos de ciclos também são possíveis, para mais detalhes consultar (Briggs et al., 2000; Trottenberg et al., 2001).

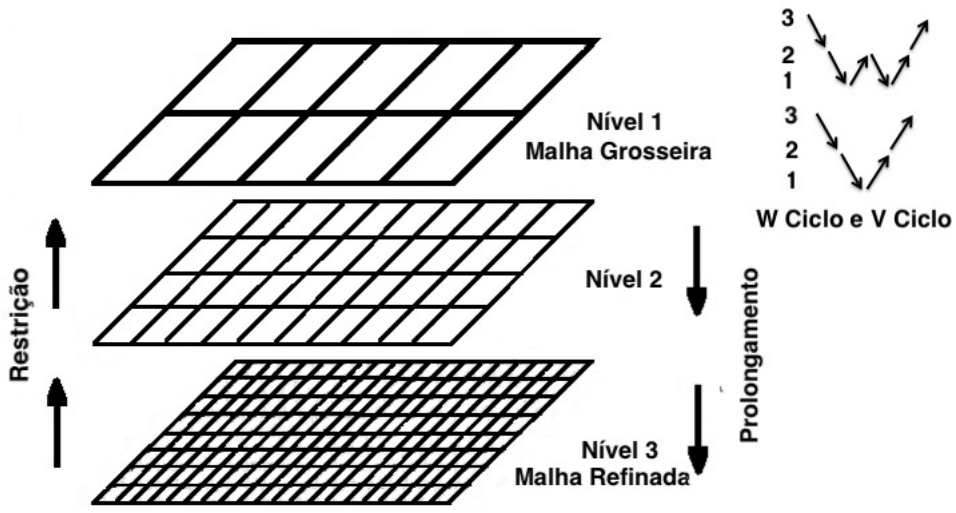


Figura 3.3: Representação do Multigrid para o caso bidimensional com os ciclos V e W.

Capítulo 4

Multigrid Geométrico em Malhas Adaptativas

Este capítulo apresenta uma estrutura de dados baseada no ALG e na *quadtree* que possibilita a utilização do método Multigrid na resolução do sistema linear oriundo da discretização pelo método dos Volumes Finitos. Essa estrutura mantém as facilidades de busca de vizinhos fornecida pelo ALG, permitindo que a diferença entre os níveis de refinamento seja maior do que 1. E ainda, associa as características de uma *quadtree* por manter os nodos refinados de diferentes níveis. Assim, optou-se por reduzir o tempo de execução com um aumento do custo de armazenamento em comparação com as estruturas tradicionais do ALG e da *quadtree*. No decorrer deste capítulo são apresentadas a estrutura de dados utilizada e os algoritmos do método Multigrid Geométrico para Malhas Adaptativas que foram desenvolvidos.

4.1 Estrutura de Dados

Conforme apresentado no Capítulo 2, o ALG é uma estrutura de dados formada por dois tipos de nodos: pretos e brancos ou de transição. Os primeiros representam as células do domínio computacional e os últimos são utilizados para conectar nodos pretos de diferentes níveis de refinamento.

A estrutura aqui apresentada mantém os nodos que acabaram de ser refinados em memória, de modo semelhante ao que ocorre para uma *quadtree*, e acrescenta uma outra funcionalidade aos nodos de transição. Agora, eles também são utilizados para indicar uma mudança de nível em relação aos elementos que estão presentes em malhas computacionais de níveis diferentes.

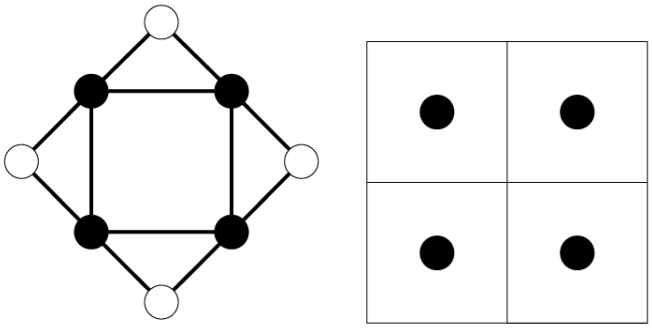


Figura 4.1: Estrutura de dados e malha geométrica inicial.

Considere uma malha inicialmente subdividida em 4 células (Figura 4.1). Um refinamento da célula noroeste é realizado, conforme a Figura 4.2. Observe que os nodos pretos refinados permanecem na malha, sendo representados por $\otimes$. Este nodo preto que foi refinado é agora chamado de nodo pai (nível 1) deu origem a um cacho de nível 2 . Ele é o pai do cacho que acabou de ser criado. Assim, após a atualização dos ponteiros de cada célula do cacho que surgiu com o refinamento a célula pai tem seus ponteiros também atualizados.

Observe que a navegação pelos nodos brancos permite que as informações dos nodos vizinhos de diferentes níveis sejam obtidas de forma simples. O nodo preto refinado passa a se comunicar com seus vizinhos por meio dos nodos brancos, isso porque, apesar de possuir o mesmo nível dos seus vizinhos ele não faz parte da malha computacional que será utilizada na resolução do problema. Aqui será considerado que ele não está no nível da malha computacional que será resolvida. O nível da malha computacional será dado pelo maior nível das células que formam a curva de Hilbert Modificada.

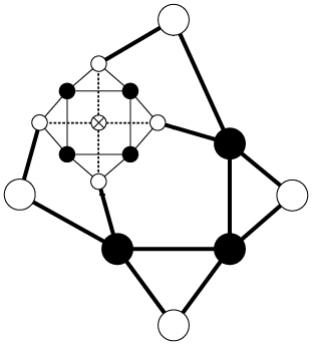
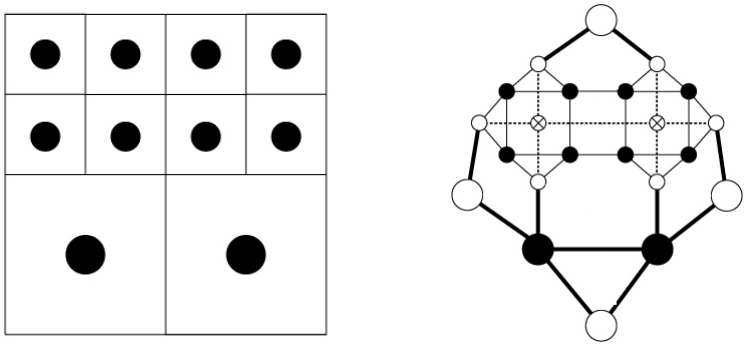


Figura 4.2: Estrutura de dados para o Multigrid Adaptativo com um refinamento da célula noroeste.

Essa nova estrutura pode ser vista como um árvore onde são armazenadas todas as informações referentes ao histórico de refinamentos ou como o ALG no último nível de

refinamento. Na Figura 4.3 a estrutura proposta é simplificada quando ocorre um novo refinamento. Neste caso os elementos de mesmo nível que irão pertencer a mesma malha computacional passam a se comunicar de forma direta.

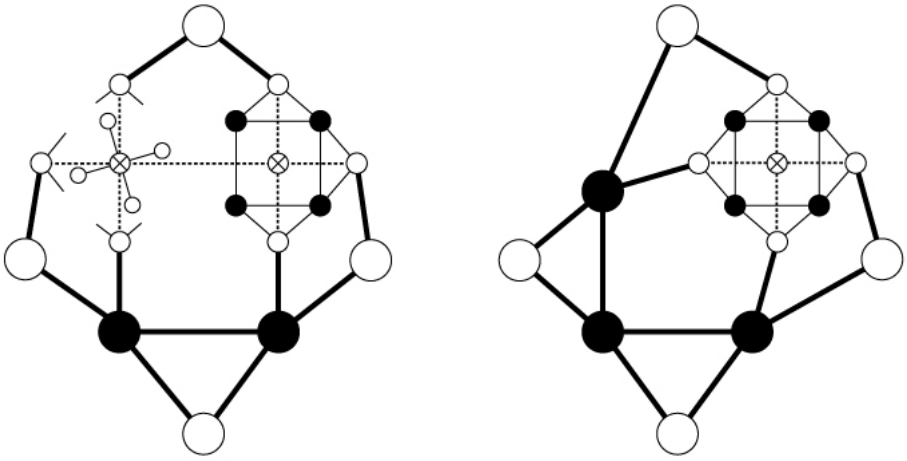


(a) Malha

(b) Estrutura de dados com simplificação de nodo branco.

Figura 4.3: Processo de refinamento e simplificação.

O processo de desrefinamento é similar ao algoritmo do ALG. Determina-se o cacho a ser desrefinado. Em seguida, são criados 4 nodos brancos que apontam para o nodo pai daquele cacho. Os quatro nodos brancos criados têm o mesmo nível do cacho desrefinado. Eles apontam para o nodo pai e para os vizinhos dos nodos pretos que foram desrefinados (Figura 4.4).a. Verifica-se se os vizinhos do nodo pai eram todos brancos, em caso afirmativo o nodo pai é ligado aos 4 nodos brancos criados. Caso contrário, o vizinho preto do nodo pai é ligado a um dos nodos brancos criados, dependendo de sua posição. A Figura 4.4.a ilustra essa situação, onde o vizinho leste do nodo pai deve ser atualizado. Após essa atualização o nodo pai é atualizado. As simplificações dos nodos brancos apresentadas para o ALG também são realizadas. A Figura 4.4.b apresenta a estrutura final após o desrefinamento.



(a) (b)

Figura 4.4: Desrefinamento da estrutura de dados.

A ordenação da malha ainda é realizada com a curva de Hilbert modificada (CHM). A ordenação dos níveis anteriores também é feita pela CHM. Dada uma CHM de nível n a CHM de nível $n-1$ é formada pelos pais das células de nível n e das células de nível menor que n. Esse processo é realizado recursivamente até a CHM de nível 1 ou menor K nível definido pelo ciclo V do Multigrid (Figura 4.5).

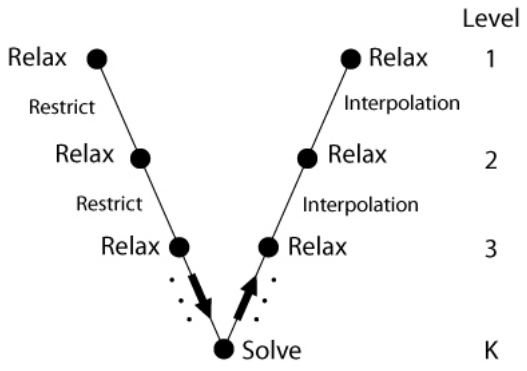


Figura 4.5: Representação dos $k-$níveis de um ciclo V no Multigrid.

4.1.1 Custo de Armazenamento

A estrutura de dados aqui apresentada tem o mesmo comportamento do ALG sendo acrescida dos nodos pretos referentes aos níveis já refinados. Para o caso do refinamentos uniformes uma malha com n células apresentaria um total de $\frac{2^{2n+2}-1}{3}$ células pretas mais os $2^{n+2}-4$ nodos brancos referentes ao contorno, para representar um total de 2^{2n} nodos na malha computacional do último nível. Observe que os nodos brancos podem ser sim-

plificados com a estratégia dada pela utilização dos nodos amarelos descrita no capítulo 2.

Já no caso dos refinamentos adaptativos a quantidade de nodos brancos é a mesma do caso do ALG. Todavia a mudança ocorre na quantidade de nodos pretos presentes em memória. A Tabela 4.1 apresenta a quantidade de nodos presentes em memória quando refinamentos adaptativos ocorrem nas diagonais do domínio. Assim, a estrutura apresenta um custo maior do que uma *quadtree* por armazenar os nodos brancos e um custo maior do que o ALG por armazenar as células que foram refinadas. Como já mencionado, os nodos brancos do contorno podem ser simplificados pelos nodos amarelos e, consequentemente, o custo de armazenamento é reduzido. Ademais, vantagens competitivas de uso desta nova estrutura de dados serão enfatizadas em relação à eficiência adiante.

Tabela 4.1: Quantidade de nodos no caso de k refinamentos uniformes seguidos de $n-k$ refinamentos não uniformes nas diagonais do domínio.

Nível	Nodos Pretos	Nodos Brancos
1	$4 = 2^2$	$4 = 2^2$
2	$4 + 16$	$4 + 8$
3	$4 + 16 + 64$	$4 + 8 + 16$
...	...	...
k	$\sum_{i=1}^{k} 2^{2i}$	$\sum_{i=1}^{k} 2^{i+1}$
$k+1$	$\sum_{i=1}^{k} 2^{2i} + 2^{2k+1}$	$\sum_{i=1}^{k} 2^{i+1} + 2^{2k+1}$
...	...	...
n	$\sum_{i=1}^{k} 2^{2i} + \sum_{j=1}^{n-k} 2^{2k+j}$	$\sum_{i=1}^{k} 2^{i+1} + 2^{n+k+1} - 2^{2k+1}$

Observe que os refinamentos adaptativos só aumentam a quantidade de nodos brancos, mas o pior caso para os nodos pretos ocorre no refinamento uniforme. Além disso, como mencionado a quantidade de memória necessária para os nodos brancos é bem menor quando comparada àquela utilizada para armazenar os nodos pretos.

4.2 Multigrid Geométrico Adaptativo

Esta seção apresenta os principais componentes do método Multigrid Geométrico Adaptativo (MGA) desenvolvido. Uma breve discussão sobre operadores de tranferência para esquemas centrados é realizada. Os passos necessários para determinar as matrizes de cada

nível também são apresentados. Por fim, os algoritmos desenvolvidos para implementar o MGA são apresentados.

4.2.1 Operadores de Transferência

A análise teórica de convergência dos métodos centrados nas células exige que a dimensão dos operadores de transferência seja superior à ordem da equação diferencial a ser resolvida (Wesseling, 1988; Mohr e Wienands, 2004) (Eq. 4.1).

$$m_P^{poli} + m_R^{poli} > M \tag{4.1}$$

onde m_P^{poli} e m_R^{poli} são as ordens dos operadores e M é a ordem da EDP. Aqui, a ordem de um operador é dada pelo grau da interpolação polinomial que ele consegue representar de forma exata, acrescido de 1.

A condição dada pela Eq. 4.1 garante que a convergência do Multigrid Geométrico ocorre independente da dimensão da malha (Wesseling, 1988; Khalil e Wesseling, 1992). Essa condição pode ser enfraquecida para os casos onde é utilizado o operador de Galerkin (Mohr e Wienands, 2004). Segundo Salinas et al. (2013), na prática essa condição pode ser relaxada caso seja utilizado um suavizador robusto. Contudo, no MGA apresentado, a construção das matrizes é realizada sem a utilização deste operador. As matrizes aqui são construídas a cada nível da malha, ou melhor, de acordo com a CHM que percorre aquele nível.

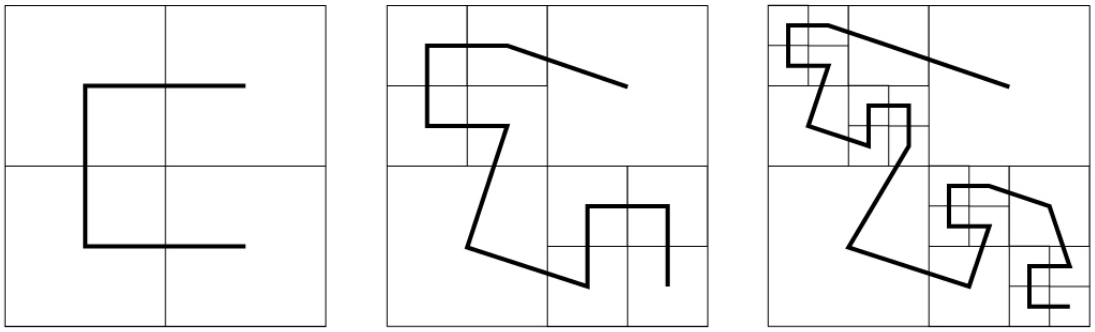
Como já explicado, os operadores de transferência têm o objetivo de mapear os valores das funções em um espaço menos refinado em um mais refinado (prolongamento) e do mais refinado para o menos refinado (restrição). Existem inúmeros operadores de transferência, sendo possível a construção deles até por informações geométricas ou até mesmo um sendo o adjunto (transposto) do outro.

O operador de restrição utilizado é uma média dos volumes que constituem um determinado *cacho* (Eq. 4.2). Este operador é de primeira ordem (Hartmann et al., 2008):

$$I_h^{2h} = \begin{cases} \frac{1}{4}\sum_{i=1}^{4} r_i^l & \text{para } l = n \\ r_i^l & \text{para } l < n \end{cases} \tag{4.2}$$

onde n é o nível do Multigrid no ciclo V que é equivalente ao nível da CHM e l é o nível da célula.

Para uma melhor compreensão desse operador, observe a Figura 4.6.b. Ela representa o nível 2 do MGA, sendo constituída por células que estão nos níveis 1 e 2. O operador descrito pela Eq. 4.2 levaria por mero transporte (cópia) os valores das células de nível 1, enquanto realizaria a média das células de nível 2 para determinar o valor associado ao resíduo da célula pai (célula de nível 1).



(a) Nível 1 (b) Nível 2 (c) Nível 3

Figura 4.6: Hierarquia de malhas gerada pela abordagem multigrid com refinamento adaptativo.

Já para operador de interpolação foram avaliadas diferentes possibilidades. Inicialmente foi utilizado o operador dado pela Eq. 4.3 (Hartmann et al., 2008).

$$I_{2h}^{h} = \frac{1}{16} \begin{bmatrix} 1 & 3 & 3 & 1 \\ 3 & 9 & 9 & 3 \\ 3 & 9 & 9 & 1 \\ 1 & 3 & 3 & 1 \end{bmatrix}_{2h}^{h} \tag{4.3}$$

Observe que esse operador é de segunda ordem (Kwak, 1999.; Trottenberg et al., 2001). Esse operador associado ao operador de restrição definido pela Eq. 4.2 garantiriam a convergência do método. Entretanto, tal operador necessita de informações de células que não são diretamente conectadas, o que implica em um custo maior nas operações de busca.

Para exemplificar esse problema, basta tomar a Figura 4.6. Para determinar os elementos do nível 3 é necessário realizar uma interpolação dos elementos que compõem a malha do nível 2. A Figura 4.7 apresenta este procedimento. Os valores representados por asterisco são utilizados para determinar o valor interpolado do próximo nível da malha. Observe que as informações de 3 células, que serão os valores interpolantes, podem ser obtidas realizando uma visita a célula pai e aos vizinhos do sul e do leste. Todavia, obter as informações referentes a quarta célula é um pouco mais complicado. A busca por essa

informação exige que seja realizada um visita ao vizinho ao leste e depois ao vizinho ao sul deste, ou; uma visita ao vizinho ao sul e depois, ao vizinho ao leste deste. Essa busca acaba tornando o procedimento de interpolação bem mais lento.

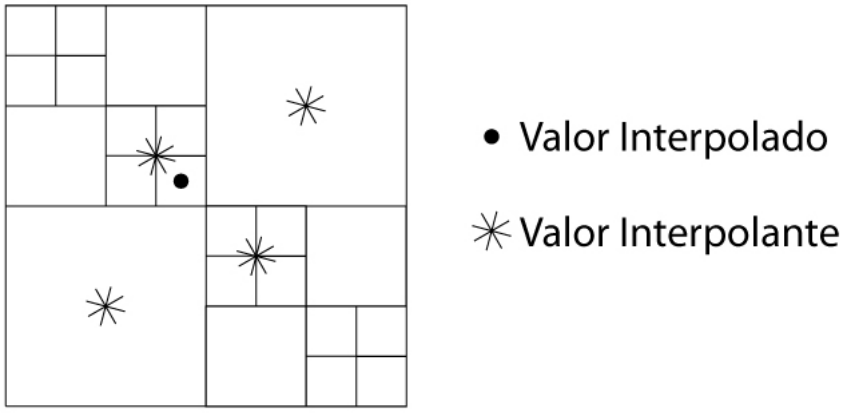


Figura 4.7: Esquema de interpolação para malha refinada no nível 3.

Os operadores apresentados em (Kwak, 1999.; Kwak e Lee, 2004) foram escolhidos. Tais operadores de segunda ordem permitem que só sejam avaliados dois vizinhos de cada célula reduzindo assim os custos de busca na estrutura proposta. Esses operadores são dados pela Eq. 4.4:

$$I_{2h}^{h} = \frac{1}{4} \begin{bmatrix} . & 1 & 1 & . \\ 1 & 2 & 2 & 1 \\ 1 & 2 & 2 & 1 \\ . & 1 & 1 & . \end{bmatrix}_{2h}^{h} \tag{4.4}$$

Observe que a ordem obtida pelos operadores de transferência é maior do que a ordem da EDP, garantindo a convergência do método independente da discretização da malha.

4.2.2 Construção das Matrizes de cada Nível

Antes da execução do algoritmo do Multigrid é necessário realizar a definição das matrizes referentes a cada nível da malha. Para isso um procedimento de busca de vizinhos é necessário.

O Algoritmo 6 descreve a busca dos vizinhos. Ele é utilizado tanto para definir as matrizes de cada nível quanto para realizar a interpolação.

No caso da interpolação ele realiza uma busca em cada direção de uma dada célula. Quando os parâmetros são passados ao algoritmo ele pode receber como vizinho um nodo branco e por isso o algoritmo faz uma busca em uma determinada direção até que seja encontrado um nodo preto que seja vizinho da célula que está sendo avaliada naquele nível da CHM.

Algoritmo 6: Busca de Vizinhos em uma Direção.

Célula BuscaVizinho(Célula Cell, Célula Vizinho, char direcao, int nivelCurva)

```
1: if (Vizinho.tipo==’branco’) then
2:    if (Vizinho.pai == Cell)|| (Vizinho.pai==Cell.pai)
      &&(Vizinho.nivel>=Cell.nível) then
3:       while (Vizinho.d1!=NULL) &&(Vizinho.tipo==’branco’) do
4:          Vizinho = Vizinho.d1;
5:       end while
6:    else
7:       while (Vizinho.d1!=NULL)&&
         (Vizinho.tipo==’branco’)&&(Vizinho.nivel<=nivelCurva) do
8:          Vizinho = Vizinho.d2;
9:       end while
10:   end if
11:   if (Vizinho.tipo==’branco’)&&(Vizinho.nível>nivelCurva) then
12:      Vizinho = Vizinho.Pai;
13:   end if
14: end if
15: return (Vizinho)
```

O último estágio do algoritmo avalia quando é necessário “subir” na estrutura para buscar a informação de vizinhança. A Figura 4.2 representa um exemplo desse caso. O nodo preto de nível 1 (elemento nordeste) precisaria realizar essa operação de subida para buscar a informação do elemento refinado. Esse processo evita uma busca muito demorada por vizinhos, permitindo “saltos” na estrutura.

4.2.3 Algoritmo

Os algoritmos implementados são baseados em (Hartmann et al., 2008). Esta abordagem permite que no mesmo nível do Multigrid existam elementos de diferentes níveis de re-

finamento, realizando pequenas atualizações nas células que estão em um nível menos refinado.

O Algoritmo 7 implementa a estratégia utilizada nesse trabalho. Observe que os operadores de restrição e prolongamento descritos nele são os mesmos definidos na sessão anterior. Os elementos que são do mesmo nível da curva de Hilbert definida são transportados de uma malha para outra de forma automática, isto significa que seus valores não são corrigidos como os outros, eles sofrem um mero transporte (replicação).

Algoritmo 7: Algoritmo Multigrid Geométrico Adaptativo.

MG(l,e^l,r^l, f^l, u^l)

1: Suaviza $A^l u^l = f^l$
2: Calcula resíduo $r^l = f^l - A^l(u^l)$
3: Restringe o resíduo pela Eq.4.2: $r^{l-1} = I_h^{2h} r^l$
4: Restringe a solução pela Eq.4.2: $u^{l-1} = I_h^{2h} u^l$
5: **if** (l==1) **then**
6: Resolve $A^l w^l = f^l$
7: **else**
8: **MG**($l-1$,w^{l-1},r^{l-1}, f^{l-1}, u^{l-1})
9: **end if**
10: Cálculo da correção $v^{l-1} = w^{l-1} - u^{l-1}$
11: Interpolação da Solução pela Eq. 4.4 : $v^l = I_{2h}^h v^{l-1}$
12: Corrige solução: $u^l = w^l + u^l$
13: Suavização da solução u^l.

Já na etapa de interpolação uma atenção especial deve ser dada às células que estão no contorno do domínio (Mohr e Wienands, 2004). O *stencil* dado pela Eq. 4.4 apresenta termos que não fazem parte do domínio. A solução adotada é realizar uma extrapolação linear levando em conta que em um esquema de correção da malha menos refinada a correção interpolada pode ser admitida como nula ao longo do contorno (Wesseling, 1992; Mohr e Wienands, 2004; Martin, 2013). Dessa forma, as contribuições são obtidas a partir do valor obtido da célula pai.

O método dos Gradientes Conjugados (MGC) foi utilizado como suavizador devido à presença de células de diferentes níveis em cada passo do Multigrid. Segundo Bank (1985), o MGC apresenta melhores propriedades de atenuação em malhas adaptativas quando comparado aos métodos clássicos de suavização (Gauss-Seidel, Gauss-Seidel Red Back, Gauss-Jacobi, etc.). Essa característica do MGC ocorre por ele ter um comportamento

adaptativo, onde em cada iteração ocorre um ajuste do passo na direção de busca definida pelo resíduo (Bank, 1985).

A estratégia completa para resolução de uma equação diferencial parcial com refinamento adaptativo utilizando a abordagem multinível é apresentada pelo Algoritmo 8. Na fase de inicialização, são realizados refinamentos uniformes da malha a partir do nível 1 (com 4 células) até que ela consiga representar o fenômeno estudado. Em seguida, as condições inicias e de contorno do problema são armazenadas. A fase seguinte consiste em definir quais os elementos que constituirão cada nível do Multigrid. Para tanto, utiliza-se a curva de Hilbert referente a cada nível conforme apresentado na Figura 4.6. A partir da definição de quais células estão em cada nível do problema é possível construir as matrizes para cada nível do Multigrid, conforme descrito na seção anterior.

Algoritmo 8: Algoritmo de Solução Adaptativa da EDP.

```
1: Inicialização da Malha
2: Definição das Condições Iniciais e de Contorno
3: while (Refina ou Desrefina) do
4:    if (nível>1) then
5:       Definição das curvas de Hilbert para os k níveis do problema.
6:    else
7:       Define curva de Hilbert da malha atual.
8:    end if
9:    Definição das Matrizes das Malhas de todos os k níveis do problema.
10:   Resolve o Sistema Linear usando o Algoritmo 7
11:   Libera Memória
12:   RefinaDesrefina
13: end while
```

Por fim, a fase de refinamento ou desrefinamento. No caso do refinamento, para cada célula de nível k é avaliado o fluxo através dela nas quatro faces (norte, sul, leste e oeste), se esse valor supera um certo limiar pré definido pelo usuário, então a célula é subdivida em 4 novas células de nível $k + 1$. Já o desrefinamento avalia o fluxo entre 4 células do mesmo *cacho*, isto é, 4 células do mesmo pai. Caso o fluxo entre elas seja menor do que um certo valor também pré determinado, então elas são desrefinadas, surgindo uma única célula de nível k. Esse processo se repete até que o fluxo entre todos os elementos da malha obedeça ao critério de refinamento definido pelo usuário.

O Algoritmo 8 apresenta uma solução para um problema estacionário. No caso de

problemas evolutivos, isto é, problemas em que a grandeza física estudada sofre variação no decorrer do tempo, essa abordagem pode ser modificado acrescentando alguns passos (Algoritmo 9).

Algoritmo 9: Algoritmo de Solução Adaptativa da EDP.

```
1: Inicialização da Malha
2: Definição das Condições Iniciais e de Contorno
3: for i = 0, ..., tfinal do
4:    if (nível>1) then
5:       Definição das curvas de Hilbert para k níveis do problema.
6:    else
7:       Define curva de Hilbert da malha atual.
8:    end if
9:    Definição das Matrizes das Malhas de todos os k níveis do problema.
10:   Resolve o Sistema Linear usando o Algoritmo 7
11:   Libera Memória
12:   RefinaDesrefina
13:   t = t + Δt
14: end for
```

A estrutura de dados aqui apresentada possui uma desvantagem no quesito memória quando comparada ao ALG e à *quadtree*. Entretanto, ela aproveita a facilidade que o ALG fornece ao permitir que os vizinhos sejam localizados de forma simples quando comparada à busca de vizinhos em uma quadtree, quando esta restrita pela regra $(2:1)$.

Outra vantagem em relacão aos dois métodos citados, é possibilitar a utilização do método Multigrid geométrico na resolução do sistema linear. Tal associação com o MG não é possível com o ALG, visto que ele não fornece as informações referentes aos níveis anteriores. Estas informações são imprescindíveis para a construção da hierarquia de matrizes utilizada pelo Multigrid.

Os algoritmos aqui propostos não estão preparados para trabalhar com problemas que apresentam geometria complexa, assim como os apresentados em (Hartmann et al., 2008). Neste caso operadores de transferência que avaliem a proporção de contribuição de cada célula seriam necessários, o que encareceria o procedimento. Os operadores utilizados neste trabalho garantem a convergência do método utilizado e oferecem um ganho de eficiência na busca por vizinhos, pois para cada valor interpolado só precisam de buscar dois vizinhos.

O gasto excessivo de memória e as restrições em relação a geometria do problema indicam a possibilidade da utilização de uma nova abordagem. Tal abordagem deve avaliar somente as informações algébricas do problema e apresentar um consumo menor de memória quando comparada ao Multigrid Geométrico Adaptativo (MGA) aqui apresentado.

Capítulo 5

Método Multigrid Algébrico

O capítulo anterior demonstrou que uma abordagem utilizando o método Multigrid Geométrico (MG) requer a construção de uma hierarquia de malhas. Todavia, o domínio computacional do problema pode se tornar extremamente complexo, dificultando ou mesmo inviabilizando a utilização do MG. Neste contexto, surge o conceito do Multigrid Algébrico (Ruge e Stüben, 1986; Briggs et al., 2000; Stüben, 2001; Trottenberg et al., 2001). Esta técnica utiliza os mesmos princípios do MG, contudo as informações são obtidas pela formulação algébrica do problema discreto.

De maneira geral, a aplicação do Multigrid Algébrico (AMG) a um dado problema consiste em duas fases: inicialização e resolução (Trottenberg et al., 2001). A primeira define de forma recursiva a hierarquia de sistemas lineares que representariam, em um contexto geométrico, as malhas menos refinadas do problema. A partir da seleção de um subconjunto de incógnitas do sistema linear é formado o próximo nível menos "refinado", a repetição desse procedimento, recursivamente, forma toda a hierarquia de "malhas". Essa fase está descrita no Algoritmo 10 (Clearly et al., 2000). Nessa fase também são definidos os operadores de transferência (restrição e prolongamento), a maneira de construção de tais operadores será detalhada na Seção 5.2.

Algoritmo 10: Algoritmo da fase de Inicialização do AMG.

1: Partição de Ω^h em dois conjuntos disjuntos C^h e F^h;
2: (a) Faça $\Omega^h = C^h$;
3: (b) Defina a interpolação I_H^h
4: Defina o operador de restrição $I_h^H = (I_H^h)^T$
5: Defina a matriz referente a malha com menos incógnitas A_H
6: **if** $|\Omega^H|$ for pequeno suficiente **then**
7: Fim do Algoritmo.
8: **else**
9: faça $\Omega^h = \Omega^H$
10: volte ao passo 1.
11: **end if**

Na segunda fase os operadores definidos são utilizados de maneira análoga ao MG. Para o caso de dois níveis, essa fase consiste na definição do resíduo, seguido pela transferência deste para uma malha menos refinada onde aplica-se um suavizador (Gauss-Seidel, SOR, Gradiente Conjugado, etc.). O resultado dessa operação é transportado para a malha mais refinada utilizando um operador de prolongamento, onde novamente aplica-se um suavizador. Por fim, a solução inicial é corrigida com o resultado obtido pela última suavização. A generalização para mais níveis ocorre de forma recursiva, conforme apresentado para o MG.

A utilização do AMG em problemas de geometria complexa é possível devido a ele não fixar a hierarquia de malhas como o MG. Na realidade, o AMG fixa o suavizador, geralmente o Gauss-Seidel, e depois define de forma eficiente os operadores de correção e interpolação (Trottenberg et al., 2001). Essa eficiência ocorre devido a ambos os operadores serem determinados a partir da seleção das incógnitas, isto é, da partição do conjuntos de incógnitas.

Durante este capítulo os termos ponto e incógnita serão utilizados sem distinção. Os termos malha fina e malha grossa também serão utilizados. O primeiro para representar os sistemas lineares oriundos da discretização pelo método dos volumes finitos da EDP analisada e o segundo para representar o sistema linear obtido pela redução ou simplificação do primeiro. Analogamente, serão utilizados os termos mais refinado e menos refinado.

Este capítulo apresenta algumas definições e critérios utilizados na fase de inicialização do AMG. Os algoritmos implementados para acoplar ao ALG o multigrid algébrico

também são detalhados. As definições e algoritmos foram adaptados dos trabalhos de (Ruge e Stüben, 1986; Briggs et al., 2000; Clearly et al., 2000; Stüben, 2001; Trottenberg et al., 2001; Iwamura et al., 2003; Pereira, 2007; Suero et al., 2012),

5.1 Conceitos Gerais

Seja um sistema linear dado por 5.1 obtido pela discretização de uma EDP por meio do método dos Volumes Finitos.

$$A_h u^h = f^h \quad ou \quad \sum_{j \in \Omega^h} a_{ij}^h u_j^h = f_i^h \quad (i \in \Omega^h) \tag{5.1}$$

com $\Omega^h = \{1, 2, ..., m\}$ denotando o conjunto de índices das variáveis e, no contexto deste trabalho, A_h sendo uma matriz esparsa.

Para encontrar um sistema linear reduzido do sistema 5.1, isto é, um sistema com menos incógnitas ou, em um contexto geométrico, menos refinado, torna-se necessário subdividir o conjunto Ω^h em dois conjuntos disjuntos $\Omega^h = C^h \cup F^h$. O conjunto C^h representa o conjunto das variáveis que definem o nível menos refinado, por analogia ao caso geométrico, seriam os pontos da malha com discretização $2h$. Já o conjunto F^h é denominado conjunto complementar de C^h, nele estão as incógnitas que não farão parte do nível menos refinado.

A partir desta partição, pode-se definir $\Omega^H = C^h$ como sendo a malha no nível menos refinado. Agora sendo conhecida uma forma de mapeamento dos vetores para o problema em Ω^H em vetores de h por meio de um operador I_H^h, o operador A_H pode ser obtido utilizando o princípio de Galerkin (Eq.5.2).

$$A_H = I_h^H A_h I_H^h \tag{5.2}$$

com $I_h^H = (I_H^h)^T$.

Neste caso, o sistema linear para esse nível é dado pela Eq.5.3.

$$A_H u^H = f^H \quad ou \quad \sum_{j \in \Omega^H} a_{ij}^H u_j^H = f_i^H \quad (k \in \Omega^H) \tag{5.3}$$

Com isso define-se a hierarquia de matrizes necessárias para uma abordagem Multigrid. Os próximos passos consistem em definir os conjuntos C e F em cada nível, os operadores de transferência e um processo de suavização. É fundamental para o desen-

volvimento do AMG um conceito para suavização algébrica, pois este conceito é o que determina a eficiência dos métodos Multigrid (Briggs et al., 2000).

Definição 1. *Um erro é dito suave, ou algebricamente suave, quando ele não é efetivamente reduzido por um método de relaxação, isto é,* $Ae^k \approx 0$.

Esta noção de suavização deve ser observada na definição dos operadores de transferência, reduzindo assim a introdução de ruídos no momento de transferir a solução para a malha mais refinada. Mais detalhes sobre erros algebricamente suaves podem ser obtidos em (Briggs et al., 2000; Trottenberg et al., 2001; Pereira, 2007). A partir dessa definição de erro algebricamente suave obtém-se o operador de suavização linear S_h e a nova solução do problema (Eq.5.4).

$$\hat{u}^h = S_h u^h + (I_h - S_h) A_h^{-1} f^h \tag{5.4}$$

com $\hat{u}^h$ sendo a solução suavizada e I_h o operador identidade do nível h.

Para a definição completa do AMG é necessário escolher quais as incógnitas que definirão as malhas menos refinadas e os operadores de transferência. Para tanto, é necessária a definição de alguns conjuntos que auxiliam nessa escolha. Tais conjuntos determinam o quanto uma variável influencia em outra, permitindo assim que sejam definidos os pontos a serem utilizados nas "malhas" menos refinadas e nos operadores de transferência.

Definição 2. *Um ponto* i *é dito diretamente conectado ao ponto* $j \in \Omega^h$ *se* $a_{ij}^h \neq 0$.

A definição 2 descreve as arestas do grafo de adjacência da matriz A_h. A definição dos níveis menos refinados trata basicamente de como realizar a partição desse grafo. Essa partição requer também informações sobre a vizinhança de um determinado ponto, ou melhor, sobre o quanto um ponto influencia os pontos na sua vizinhança. A definição 3 descreve o conceito de vizinhança de um ponto que será utilizado durante este trabalho.

Definição 3. *A vizinhança para o ponto* i *é dada por:*

$$N_i^h = \{j \in \Omega^h : j \neq i, a_{ij} \neq 0\}$$

A partir da definição de vizinhança é possível determinar o conceito de influência ou dependência de um ponto.

Definição 4. *Uma variável i está fortemente negativamente conectada ou n-conectada a outra variável j, se*

$$-a_{ij} \geq \theta \max_{a_{ik}<0} |a_{ik}| \qquad com \;\; \theta \in (0,1) \; fixo.$$

onde o fator θ é conhecido como fator de redução da malha.

A determinação da conectividade entre as variáveis permite definir o grau de influência entre elas. O conjunto formado por todos as variáveis que influenciam fortemente uma variável i é denotado por S_i.

Definição 5. *O conjunto de pontos que influenciam fortemente i é definido como:*

$$S_i = \{j \in N_i : i \;\; fortemente \; n{-}conectada \; a \; j\}$$

Como consequência da definição 5 surge o conjunto das variáveis que dependem fortemente do ponto i, sendo denotado por $S_i^T = \{j \in \Omega : i \in S_j\}$.

A definição de influência é importante para o processo de interpolação no AMG. Observe que, o valor do coeficiente a_{ij} que multiplica a incógnita u_j, pode ter uma grande influência na equação i. Isto significa que, se o valor para $|a_{ij}|$ for suficientemente grande, qualquer modificação no valor de u_j tem um grande efeito no valor de u_i, ou seja,

$$u_i = \frac{1}{a_{ii}} \sum_{j \neq i} a_{ij} u_j \quad \forall \;\; i = 1, ..., n-1. \tag{5.5}$$

Dessa forma, é importante que no processo de interpolação da incógnita u_i sejam avaliadas as variáveis u_j tais que j influencia fortemente i (Pereira, 2007). A partir da definição desses conjuntos é possível definir os critérios que devem ser obedecidos para determinação das "malhas" menos refinadas.

O primeiro critério é baseado nas conexões diretas. Para cada $i \in F$ os elementos que o influenciam ($j \in S_i$) devem estar no conjunto C e i tem que possuir no mínimo um ponto em C. Essa escolha determina que o erro algebricamente suave varie mais lentamente na direção do engrossamento (Trottenberg et al., 2001).

O segundo critério visa buscar o maior conjunto C, cujos elementos não sejam fortemente n-conectados. Tal critério visa reduzir o custo computacional da fase de suavização devido ao aumento da dimensão da matriz A^H (Briggs et al., 2000).

Obedecer esses dois critérios, em geral, não é simples (Briggs et al., 2000). Dessa forma,

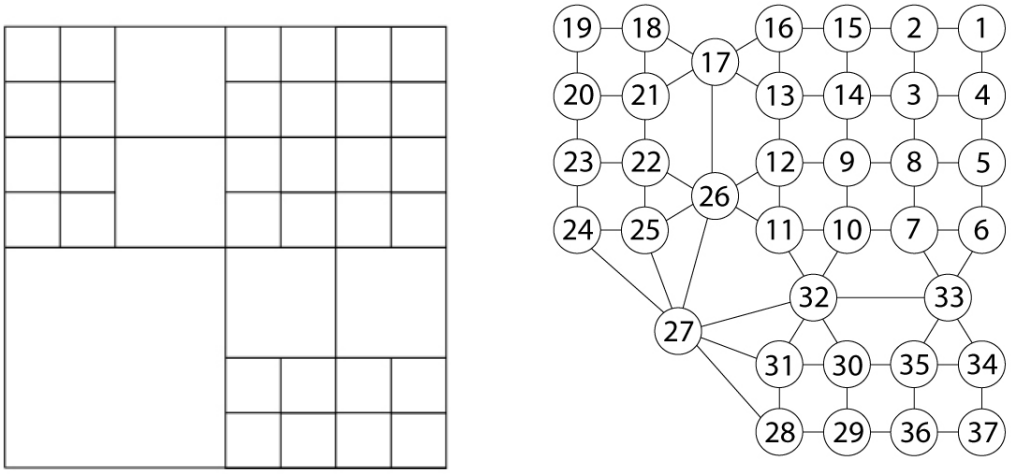
a escolha é obedecer ao primeiro critério e quando possível avaliar o segundo de forma a reduzir o tamanho do conjunto C. A redução do tamanho do conjunto C visa reduzir o esforço computacional na etapa de solução. O Algoritmo 11 realiza uma determinação inicial dos conjuntos. Ele visa realizar uma distribuição quase uniforme dos pontos de C sobre o domínio do problema.

Algoritmo 11: Definição do Conjunto C^h.

1: Sejam $C = \emptyset, F = \emptyset, U = \Omega^h$.
2: Seja $\lambda_i = |S_i^T \cap U| + 2|S_i^T \cap F|$, $\forall i$
3: **while** $(U! = \emptyset)$ **do**
4: Tome $i \in U$ com máximo λ_i. Seja $C = C \cup \{i\}$ e $U = U - \{i\}$
5: **for** $j \in S_i^T \cap U$ **do**
6: $F = F \cup \{j\}$
7: $U = U - \{j\}$
8: **for** $l \in S_j \cap U$ **do**
9: $\lambda_l = \lambda_l + 1$
10: **end for**
11: **end for**
12: **for** $j \in S_i \cap U$ **do**
13: $\lambda_j = \lambda_j - 1$
14: **end for**
15: **end while**

No Algoritmo 11, o parâmetro λ_i diz o quanto uma variável presente no conjunto de restrição realmente influencia nas outras variáveis. Ele funciona como uma métrica de importância, dizendo o quanto uma variável $i \in U$ pertence ao conjunto C, permitindo um acompanhamento do estado atual dos conjuntos C e F.

A Figura 5.1.a apresenta a malha de um problema de Laplace discretizado pelo MVF. Na Figura 5.1.b está o grafo de adjacência da matriz gerada. É importante que as incógnitas estejam ordenadas para garantir que todas as variáveis tenham seu valor de importância (λ_i) determinado (Suero et al., 2012). Observe que a ordenação das incógnitas utilizou a curva de Hilbert Modificada (CHM).



(a) Malha Refinada Adaptativamente.

(b) Grafo de Adjacência da Matriz de Discretização

Figura 5.1: Problema de Laplace com parâmetro de refinamento $\epsilon = 0.7$

A Figura 5.2 apresenta o funcionamento do Algoritmo 11. Primeiro são determinados os valores de λ_i para todos os pontos. Em seguida, o ponto com maior valor de influência é definido como um elemento do conjunto C e os pontos a ele conectado são adicionados ao conjunto F. Os valores de λ_i são recalculados para os pontos ainda não tratados. O processo se repete sempre observando a ordenação pela CHM e novos pontos de C são determinados. A Figura 5.1 apresenta a continuação do procedimento até a partição final da fase inicial de divisão.

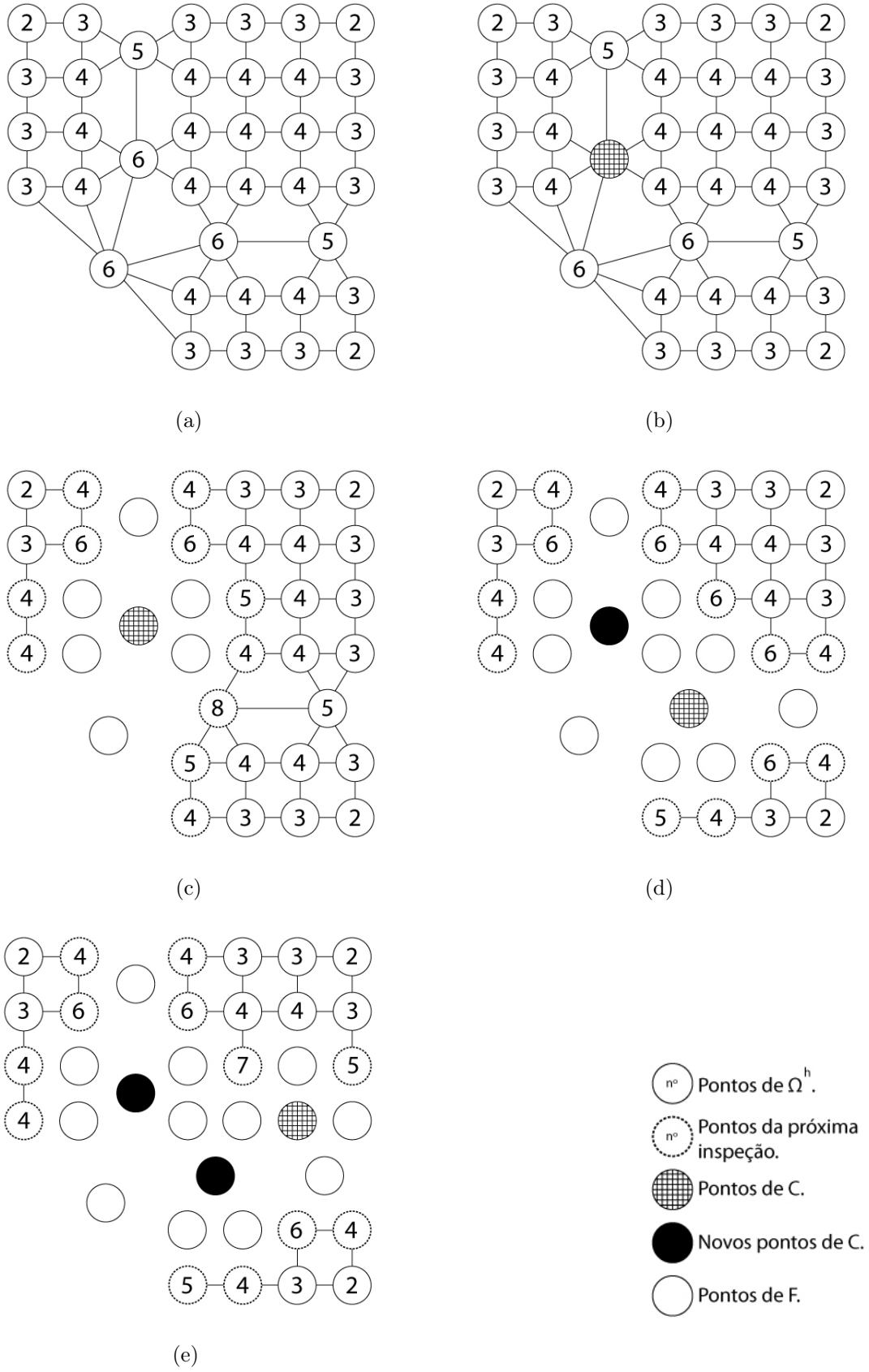


Figura 5.2: Esquema de divisão dos pontos nos conjuntos C e F.

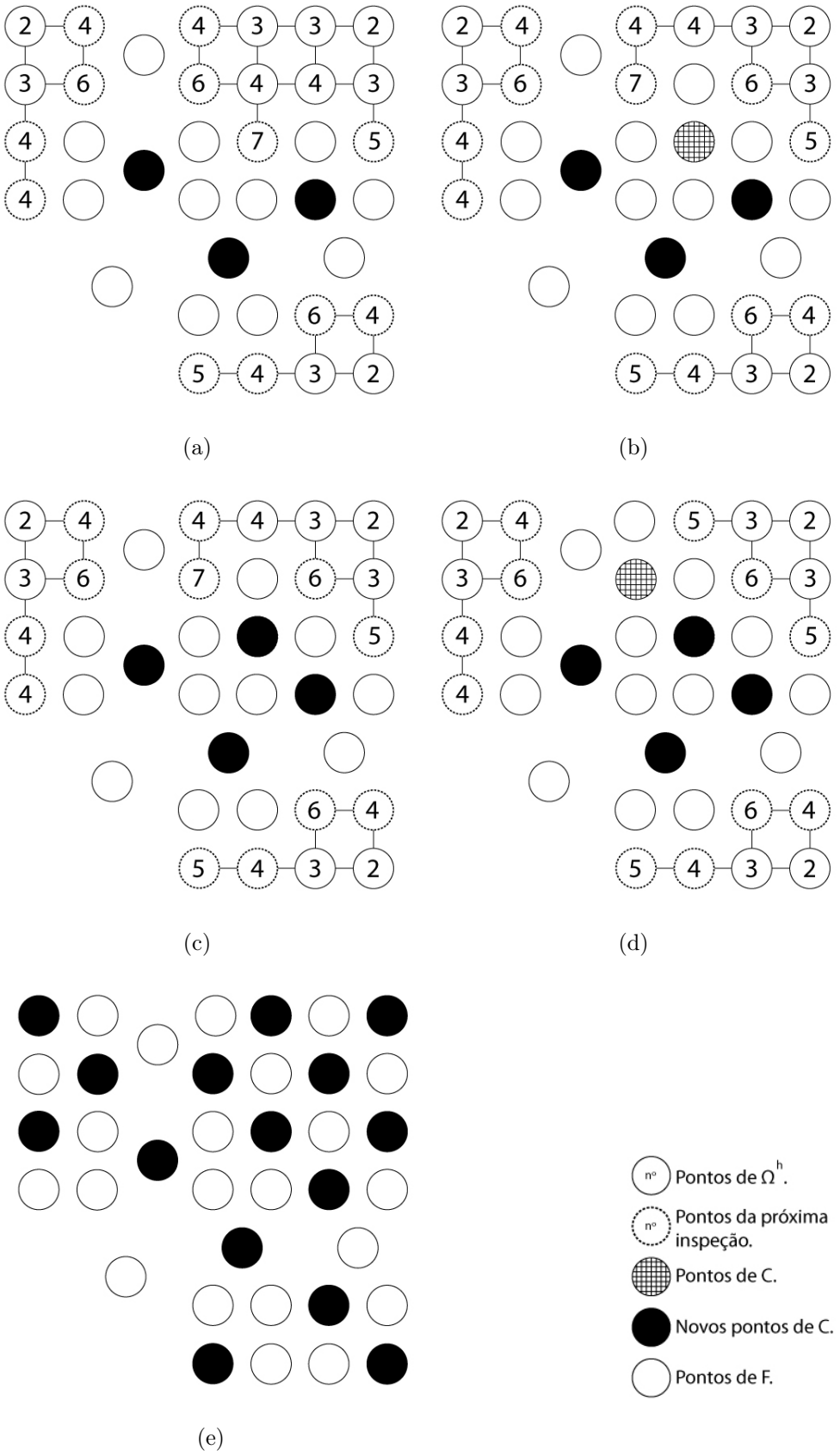


Figura 5.3: Representação final da partição dos conjuntos C/F na fase inicial de seleção de incógnitas.

5.2 Procedimento de Interpolação

Como visto no MG, o operador de interpolação transporta a solução da equação residual da malha menos refinada para a mais refinada $I_H^h : \Omega^H \rightarrow \Omega^h$.

Observando que o erro suave varia lentamente na direção da forte conexão, a Equação 5.5 permite a obtenção do valor de u_i na malha fina por uma interpolação do valor de u_j na malha grossa desde que i dependa fortemente de j (Briggs et al., 2000). Assim, as propriedades de suavidade algébrica do erro e esparsidade da matriz A^h podem ser utilizadas para descrever as componentes da equação do erro (Pereira, 2007):

$$a_{ii}e_i^h = -\sum_{j \in N_i} a_{ij}e_j^h \qquad i = 1, ..., n \tag{5.6}$$

onde N_i representam os vizinhos de i.

A Equação 5.6 apresenta o valor de e_i^h como uma média das contribuições dos seus vizinhos diretos. Todavia, com o particionamento das incógnitas nos conjuntos C/F somente os valores que pertencem ao conjunto C estão disponíveis para a interpolação (Pereira, 2007). Por isso é necessário avaliar melhor a vizinhança de i.

Os pontos em N_i tem algumas características que permitem dividi-los em três subconjuntos (Briggs et al., 2000):

- Conjunto dos pontos que influenciam fortemente i, ou seja, são pontos que estão tanto em C quando em $S_i \subset N_i$, isto é $C_i = C \cap S_i$;
- Conjunto dos pontos que apesar de não pertencerem a C_i, influenciam fortemente o ponto i. Esse conjunto é dado por $D_i^s = (N_i - C_i) \cap S_i = D_i \cap S_i$. Esses são os pontos que pertencem a malha fina e influenciam fortemente i;
- Conjunto D_i^w é aquele que contém os pontos que influenciam fracamente o ponto i. Nele podem estar contidos pontos da malha fina ou da malha grossa. Os elementos desse conjunto são denominados pontos fracamente conectados, isto é, $D_i^w = D_i - S_i$.

A partir da definição desses novos conjuntos, a Eq. 5.6 pode ser reescrita na forma da Eq. 5.7

$$a_{ii}e_i^h = -\sum_{j \in C_i} a_{ij}e_j^h - \sum_{j \in D_i^s} a_{ij}e_j^h - \sum_{j \in D_i^w} a_{ij}e_j^h \tag{5.7}$$

Observe que os pontos conhecidos são somente os pertencentes a C_i. Por isso algumas manipulações algébricas são necessárias para que da Eq. 5.7 o operador de interpolação

possa ser obtido. Em (Pereira, 2007) e (Briggs et al., 2000) todas essas operações são demonstradas originando a Expressão 5.8.

$$(I_H^h e^H)_i = \begin{cases} e_i^H & \text{se } i \in C \\ \sum\limits_{j \in C_i} w_{ij} e_j^H & \text{se } i \in F \end{cases} \tag{5.8}$$

$$\text{onde } w_{ij} = \frac{a_{ij} + \sum\limits_{m \in D_i^s} \left(\frac{a_{im} a_{mj}}{\sum\limits_{k \in C_i} a_{mk}} \right)}{a_{ii} + \sum\limits_{n \in D_i^w} a_{in}}$$

Com a definição do operador de interpolação e recordando que a restrição é dada por $I_h^H = (I_H^h)^T$ (Eq. 5.2), os conceitos necessários para definir o AMG estão completos.

O Algoritmo 12 implementa a construção desses operadores, bem como a divisão final da partição dos conjuntos C/F.

A entrada do algoritmo é uma seleção inicial dos pontos do conjunto C dado pelo Algoritmo 11. Ele verifica quais dos pontos em F podem ser transferidos para o conjunto C, obedecendo ao primeiro critério.

Uma outra questão a ser avaliada é a dimensão do operador de interpolação. Ele afeta diretamente o custo computacional do procedimento. Quanto maior for esse operador maiores são o custo e o gasto de memória na fase de resolução (Iwamura et al., 2003). Por isso, esse operador sofre um truncamento de maneira a desconsiderar as variáveis que possuem uma conexão mais fraca com a variável i, dentre aquelas que possuem uma forte dependência. Com isso surge mais um conjunto, S_i^D, que representa os pontos fortemente dependentes do conjunto C_i (Iwamura et al., 2003; Suero et al., 2012).

$$S_i^D = \left\{ j \in D_i^s : \sum_{l \in C_i} |a_{jl}| > \gamma \left(\frac{|a_{ij}|}{max|a_{ik}|} \right) max|a_{jl}| \right\}$$

O valor de γ define a quantidade de pontos na malha grossa. Quanto maior o seu valor mais pontos estarão presentes na malha grossa e maior será o custo da suavização. Segundo Suero (Suero et al., 2012), a determinação desse parâmetro deve avaliar que cada ponto tenha pelo menos uma conexão fortemente dependente. Na maioria dos trabalhos estudados para este trabalho (Ruge e Stüben, 1986; Briggs et al., 2000; Clearly et al., 2000; Stüben, 2001; Trottenberg et al., 2001; Iwamura et al., 2003; Pereira, 2007) esse valor é fixado

em 0.35. (Suero et al., 2012) apresenta um estudo detalhado sobre diferentes valores para γ e encontra o mesmo valor que nos demais trabalhos.

Algoritmo 12: Define o conjunto C e os pesos da interpolação.

1: Seja $T = \emptyset$.
2: **while** $!(F \subseteq T)$ **do**
3: Tome $i \in F - T$ e $T = T \cup \{i\}$.
4: Sejam C_i, D_i^S, D_i^S, D_i^W, S_i^D e o conjunto $\overline{C_i} = \emptyset$
5: Seja $d_i = a_{ii}$ e para $k \in C_i$, $d_k = a_{ik}$
6: **for** $j \in D_i^w$ **do**
7: $d_i = d_i + a_{ij}$.
8: **end for**
9: **for** $j \in D_i^S$ **do**
10: **if** $(j \in S_i^D)$ **then**
11: $d_k = d_k + \frac{a_{ij}a_{jk}}{\sum_{l \in C_i} a_{jl}}$ para $k \in C_i$
12: **else**
13: **if** $\overline{C_i} \neq \emptyset$ **then**
14: $C = C \cup \{i\}$ e $F = F - \{i\}$
15: Volte ao passo 2.
16: **else**
17: $\overline{C_i} = \{j\}$ e $C_i = C_i \cup \{j\}$ e $D_i^S = D_i^S - \{j\}$
18: Atualize S_i^D
19: Volte ao passo 4.
20: **end if**
21: **end if**
22: $C = C \cup \overline{C_i}$ e $F = F - \overline{C_i}$
23: **for** $k \in C_i$ **do**
24: $w_{ik} = \frac{-d_k}{d_i}$
25: **end for**
26: **end for**
27: **end while**

Por fim, o Algoritmo 13 constrói as matrizes referentes as malhas menos refinadas de acordo com o princípio de Galerkin (Haase e Langer, 2002).

Algoritmo 13: Geração da Matriz de Coeficientes.

```
1: Faça A_H = 0;
2: for k ∈ F ∪ C do
3:    Determine F^k e C^k;
4:    for l ∈ F^k ∪ C^k ∪ {k} do
5:       T = C^k ∪ C^l ∪ ({l} ∩ C);
6:       for i ∈ T do
7:          for j ∈ T do
8:             A_{H,i,j} = A_{H,i,j} + α_{ki} A_{kl} α_{lj}
9:          end for
10:      end for
11:   end for
12: end for
```

Os pesos da interpolação α_{ij} são obtidos a partir da Equação 5.9.

$$\alpha_{ij} = \begin{cases} 1 & i = j \in C \\ \frac{-Aij - c_{ij}}{A_{ii} + c_{ii}} & \text{se } i \in F, j \in C^i \\ 0 & \text{caso contrário} \end{cases} \tag{5.9}$$

com os valores de c_{ij} dados por:

$$c_{ij} = \sum_{k \in F^i} \frac{A_{ik} A_{kj}}{\sum\limits_{l \in C^i} A_{kl} + A_{ki}}$$

Os valores de A_{ij} são referentes a matriz na malha fina, enquanto os conjuntos C^i e F^i são denominados, respectivamente conjuntos dos "vizinhos refinados"e dos "vizinhos finos"(Suero et al., 2012). Esses conjuntos são definidos por:

$$C^i = \{j \in C : a_{ij} \neq 0, i \in F\}$$

e

$$F^i = \{j \in F : a_{ij} \neq 0, i \in F\}$$

5.3 Aspectos Computacionais

Os algoritmos apresentados foram implementados utilizando conceitos de programação genérica, especificamente a biblioteca padrão de gabaritos (STL, *Standard Template Library*) de $C++$ (Drozdek, 2012; Deitel e Deitel, 2005). Essa biblioteca fornece objetos, denominados contêiners que permitem a manipulação de outros objetos. A STL implementa inúmeros tipos de contêiners, cada um com seus recursos e procedimentos que permitem armazenar e manipular os dados.

No presente trabalho foram utilizados os contêiners Set, Vector e List. O primeiro se assemelha a um conjunto matemático, onde os elementos são objetos, os quais não apresentam repetição. Esse contêiner permite a implementação de todos os conjuntos necessários para o AMG. As operações de união, interseção, ordenação, diferença e pertinência são alguns dos recursos fornecidos por este contêiner. O segundo contêiner, consiste em um array unidimensional que permite o acesso dos dados aleatoriamente. O último implementa uma lista encadeada e fornece algoritmos eficientes para inserção, remoção e busca de elementos. Com este são implementados os elementos das matrizes A de todos os níveis do AMG.

5.3.1 Custo Computacional e Consumo de Memória

Idealmente o custo do AMG seria $O(n)$ onde a constante multiplicativa depende somente da natureza do problema e não do tamanho (Ruge e Stüben, 1986). Todavia determinar precisamente o custo e o consumo de memória do AMG é uma tarefa complicada (Briggs et al., 2000). Essa indeterminação ocorre devido à escolha dos pontos que formam a hierarquia de malhas ser realizada de maneira heurística.

Dessa forma, são utilizadas duas grandezas para realizar uma estimativa *a posteriori* dos custos do AMG. A primeira é a Complexidade do Grid. Esta taxa relaciona a quantidade de elementos de todas as malhas com a quantidade de elementos na malha final.

Já a Complexidade do Operador fornece uma relação entre a quantidade de elementos não nulos de todas as matrizes de discretização obtidas nos níveis menos refinados do problema com a matriz da malha fina. Ou seja, esse valor indica o espaço total de memória necessário para armazenar as matrizes de todas as malhas. Observando que a fase de resolução dos algoritmos Multigrid tem seus custos mais elevados dados pelas etapas de suavização e cálculo do resíduo, este operador dá uma boa estimativa do custo

computacional de um algoritmo do tipo ciclo V (Briggs et al., 2000).

Neste trabalho, a implementação do AMG cria para cada nível do AMG uma lista encadeada que armazena os elementos das matrizes de discretização. Dessa forma, são armazenados somente os elementos não nulos da matriz de cada nível. Ao final da execução do AMG toda a memória é desalocada nos moldes do ALG. Além da matriz do problema, o operador de interpolação também é armazenado. São armazenados somente os elementos não nulos deste operador também. O operador de restrição não precisa ser armazenado, suas contribuições são obtidas diretamente do operador de interpolação. Basta observar que o operador de restrição é simplesmente a matriz transposta da matriz que fornece o operador de interpolação. O custo da fase inicial é dominado pelas operações envolvendo os conjuntos (ordenação, união, intersecção, etc.).

No contexto do ALG, o AMG é utilizado como um resolutor de sistema linear no lugar do método dos gradientes conjugados (MGC). Assim, o ALG fornece como parâmetro de entrada para o AMG uma lista encadeada contendo os elementos não-nulos da matriz de discretização obtida do problema adaptativo. O AMG obtém uma aproximação da solução do problema. Por fim, esse valor é retornado como uma lista encadeada para o ALG.

Capítulo 6

Resultados Computacionais

Este capítulo apresenta os resultados computacionais das abordagens apresentadas nos Capítulos 5 e 6. Os algoritmos e estruturas de dados apresentados anteriormente foram implementados em linguagem C++ e compilados com o compilador g++ versão 4.3. As execuções são realizadas em um computador Quad-Core com 2.7 GHz, 4.0 Gigabytes de RAM com um sistema operacional Linux com kernel 3.2.0.

A próxima seção apresenta a descrição sucinta dos problemas abordados: equação de Laplace, equação de Poisson e o problema da camada limite. Todos os problemas apresentados são bem postos no sentido de Hadamard (Garvine, 1968; Mavriplis, 1995; Burgarelli et al., 2006; Oliveira et al., 2011; Suero et al., 2012). As discretizações desses problemas por meio do método dos Volumes Finitos também são apresentadas.

Por fim, os resultados computacionais demonstrando os ganhos de eficiência em relação ao tempo nas abordagens propostas nos capítulos anteriores são apresentados.

6.1 Descrição dos Problemas

6.1.1 Equação de Laplace

O primeiro problema abordado é a conhecida equação de Laplace (Eq.6.1)(Suero et al., 2012; Burgarelli et al., 2006).

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \tag{6.1}$$

$$u(x,0) = u(0,y) = g(x,y)$$
$$u(x,1) = u(x,y) = h(x,y)$$

com condições de contorno dadas por: $g(x,y) = 0$; $h(x,y) = x$ se $y = 1$ e $h(x,y) = y$ se $x = 1$.

A solução analítica desse problema é dada pela Eq. 6.2.

$$u(x,y) = x * y \tag{6.2}$$

A discretização da Eq.6.1 pelo método dos Volumes Finitos é obtido primeiro com a discretização da malha em volumes de controle (células quadrangulares). Em seguida, a equação é integrada sobre cada um desses volumes de controle obtendo a Eq.6.3.

$$\int_V \nabla^2 u dV = 0 \tag{6.3}$$

Por fim, aplica-se o teorema da divergência de Gauss para obter o sistema linear dado pela Eq.6.4.

$$\oint \nabla u.\hat{n}\partial v_i = 0 \tag{6.4}$$

onde $\hat{n}$ é o vetor normal em relação a superfície do volume v_i e ∂v_i representa o contorno do volume de controle.

6.1.2 Equação de Poisson

A equação de Poisson é dada pela equação Eq.6.5.

$$\frac{\partial^2 u}{\partial^2 x} + \frac{\partial^2 u}{\partial^2 y} = S, \quad 0 < x, y < 1$$

$$u(x,0) = u(0,y) = u(1,y) = u(x,1) = 0 \tag{6.5}$$

e o termo fonte é (S) dado por: $S = -2[(1-6x^2)y^2(1-y^2) + (1-6y^2)x^2(1-x^2)]$

A solução analítica desse problema é dada por:

$$u(x,y) = (x^2 - x^4)(y^4 - y^2) \tag{6.6}$$

A discretização em Volumes finitos desse problema é dada pela Eq.6.7

$$\oint \nabla u.\hat{n}\partial v_i = S_i * |V_i| \tag{6.7}$$

onde $|V_i|$ é a área do volume v_i.

6.2 Critério de Refinamento

O critério de refinamento utilizado nesse trabalho é baseado no gradiente do fluxo entre células vizinhas. Caso seja verificado que o gradiente do fluxo de uma determinada célula é maior do que um determinado valor pré-estabelecido pelo usuário, então essa célula é refinada em 4 novas células.

Já para o Multigrid Geométrico Adaptativo (MGA) e para o Multigrid Algébrico (AMG), define-se o critério de parada do ciclo V pela Eq. 6.8 (Miyashita e Yamada, 2005):

$$\frac{|R|_{total}}{|f|_{total}} < \epsilon \tag{6.8}$$

onde $|R|_{total} = \int_\Omega |R| dV$ e $|f|_{total} = \int_\Omega |f| dV$.

Essas integrais são calculadas somando todos os valores sobre o domínio (Ω). Caso esse critério não seja atingido o algoritmo executa um máximo de ciclos pré-definido pelo usuário. Nas simulações realizadas esse valor foi fixado em 30 iterações.

6.3 Resultados Numéricos dos Métodos Multinível

Esta seção apresenta os resultados dos tempos de CPU para os problemas descritos utilizando as abordagens já descritas e o ALG. Resultados que demonstram a convergência dos métodos também são apresentados.

6.3.1 Resultados para Equação de Laplace

A Tabela 6.2 apresenta os resultados do Problema 1. Observe que o MGA e o AMG apresentaram os melhores resultados como esperado.

Tabela 6.1: Tempo de Execução (em segundos) para o Problema de Laplace 1 com refinamento uniforme.

		Métodos		
Nível	Número de Células	ALG	MGA	AMG
7	16384	$0,088$	$0,074$	$0,07$
8	65536	$0,542$	$0,309$	$0,304$
9	262144	$3,224$	$1,22$	$1,12$
10	1048576	$9,376$	$4,77$	$4,5$

No caso de refinamentos não uniformes adotou-se um critério de refinamento de $0,0725$ com nível máximo de refinamento igual a 7. Como critério de parada para os ciclos do MGA adotou-se $\epsilon = 0.01$. Conforme observado por Miyashita e Yamada (2005), esse valor é suficiente para garantir a convergência e não afetar a velocidade do método.

Tabela 6.2: Tempo de Execução (em segundos) para o Problema de Laplace 1 com refinamento não-uniforme.

		Métodos		
Nível	Número de Células	ALG	MGA	AMG
6	3496	0.235	0.025	0.02
7	8695	2.02	0.79	0.05
8	10570	8.95	1.4	0.10

6.3.1.1 Comparação da fases de construção da Matriz e de refinamento da Malha

A Tabela 6.3 apresenta o tempo necessário para criar as matrizes do ALG e da estrutura de dados proposta no capítulo 5. No caso do AMG e da versão paralela do método dos Gradientes Conjugados os resultados são os mesmos obtidos para o ALG. Os resultados apresentados para o AMG incluem a sua fase de inicialização e isso é incluído no tempo total de execução dele.

Os resultados obtidos demonstram que o ALG é mais eficiente nas fases de criação do sistema linear e do refinamento das células. Isso ocorre devido ao ALG só criar 3 células novas e reaproveitar as informações da célula refinada. Outro ponto, é que a estrutura de dados proposta precisa realizar novas simplificações e corrigir os ponteiros dos níveis menos refinados. Além disso, o ALG só realiza a construção da malha mais refinada enquanto o ele não necessita manter as matrizes de diferentes níveis.

Tabela 6.3: Tempo de Execução (em segundos) para o Problema de Laplace 1 das Fases de Refinamento e Determinação das Matrizes

Nível	Número de Células	Métodos			
		ALG		MGA	
		Sistema	*Refinamento*	*Sistema*	*Refinamento*
7	16384	$0,021$	$0,002$	$0,03$	$0,004$
8	65536	$0,091$	$0,007$	$0,119$	$0,017$
9	262144	$0,375$	$0,033$	$0,459$	$0,459$
10	1048576	$1,457$	$0,135$	$2,013$	$0,22$

6.3.2 Resultados para Equação de Poisson

A Tabela 6.4 apresenta os resultados do tempo de CPU para o problema de Poisson com refinamentos adaptativos. Foram utilizados 20 ciclos tanto para o MGA quanto para o AMG. O critério de refinamento adotado foi de 0.0075.

Os resultados apresentados pelo ALG demonstram o quanto o mal-condicionamento da matriz afeta a convergência do método dos Gradientes Conjugados.

No caso do MGA, o tempo superior ao AMG deve-se a fase de interpolação que no caso do AMG é pré-definida na etapa de inicialização do método, enquanto no MGA ela é realizada em cada ciclo do Multigrid. A fase de interpolação no caso do MGA exige 2 operações de busca para cada célula, que no caso de refinamentos adaptativos tem um

Tabela 6.4: Tempo de Execução (em segundos) para o Problema de Poisson com a malha não-uniforme

Nível	Número de Células	Métodos		
		ALG	MGA	AMG
7	4690	$1,104$	$0,287$	$0,128$
8	9892	$2,616$	$0,894$	$0,337$
9	18262	$9,178$	$3,12$	$1,4920$

custo um pouco maior.

O erro encontrado para o nível 9 de refinamento foi de 0.000424, essa precisão deve-se a ordem da discretização utilizada.

A Figura 6.1 apresenta o gráfico do resíduo para cada ciclo no caso do AMG no nível 7 demonstrando a convergência do ciclo.

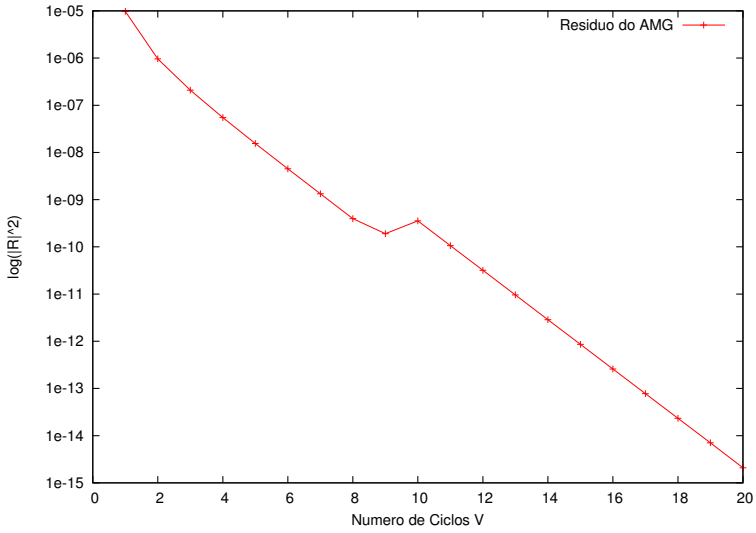


Figura 6.1: Resíduo dos ciclos no AMG para a malha com nível 7 de refinamento.

6.4 Problema da Camada Limite com o AMG

Esse problema descreve o comportamento do fluxo de ar ao redor de uma aerofólio (Garvine, 1968; Mavriplis, 1995; Oliveira et al., 2011). A abordagem descrita aqui é adaptada de (Oliveira et al., 2011). O problema da Camada Limite é descrito para um fluxo bidimensional incomprenssível em regime permanente em coordenadas retangulares pela Eq.6.9.

$$\text{Momento: } u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + v\frac{\partial^2 u}{\partial y^2} \tag{6.9}$$

$$\text{Continuidade: } \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0$$

com ρ é a densidade do fluido, u é a variável dependente da EDP, p é a pressão e v é a viscosidade cinemática.

A discretização da equação do momento pelo método dos Volumes Finitos é dada por(Oliveira et al., 2011):

$$u_P^{n+1} - u_N^{k+1}\frac{v}{hu_P^k} - u_S^{k+1}\frac{v_P^k + \frac{v}{h}}{u_P^k} = u_W^k - 2\frac{v}{h} - v_P^k + u_{ep}\frac{u_{eP}^k - u_{eW}^k}{hu_P^k} \tag{6.10}$$

Os índices W, NeS indicam as células vizinhas ao volume de controle P, respectivamente ao oeste, norte e sul. O termo v indica a viscosidade cinemática e $h = \Delta x = \Delta y$.

A equação discreta que descreve a equação da continuidade obtida pela formulação em Volumes Finitos é definida como:

$$v_P^{k+1} - v_S^{k+1} = u_W^{k+1} - u_P^{k+1} \tag{6.11}$$

Maiores detalhes podem ser obtidos em (Oliveira et al., 2011). Somente o sistema linear que surge da discretização da equação da continuidade é resolvido utilizando o método dos Gradientes Conjugados e o Multigrid Algébrico (AMG). A Tabela 6.5 apresenta os resultados das simulações realizadas para diferente níveis de refinamento. Foi utilizada uma viscosidade de $1,5X10^{-5}$ nos testes realizados.

Tabela 6.5: Tempo de Execução (em segundos) para o Problema da Camada Limite - NACA

		Métodos	
Nível	Número de Células	ALG	AMG
10	6676	0,031	0,004
11	13402	0,059	0,01
12	26848	2,72	0,02
13	53746	5,02	0,0616

A aplicação do Multigrid Geométrico Adaptativo (MGA) a esse problema tornou-se inviável devido ao não tratamento das funções de interpolação de maneira apropriada.

O trabalho de Hartmann et al. (2008) apresenta algumas estratégias que poderão ser integradas ao MGA e aplicadas ao problema aqui descrito.

A partir do nível 12 de refinamento a quantidade de iterações necessárias para a convergência do ALG tem um aumento significativo no tempo,devido ao aumento do mal condicionamento da matriz. Para valores maiores de refinamento, por exemplo, 14 níveis de refinamento a estratégia desenvolvida pelo ALG não converge. Visto que limitamos o número de iterações a quantidade de células da malha. Todavia, o AMG não apresente essa limitação e ainda alcança a convergência.

A Figura 6.2 apresenta a malha para o problema simulado com o AMG representando 53746 células.

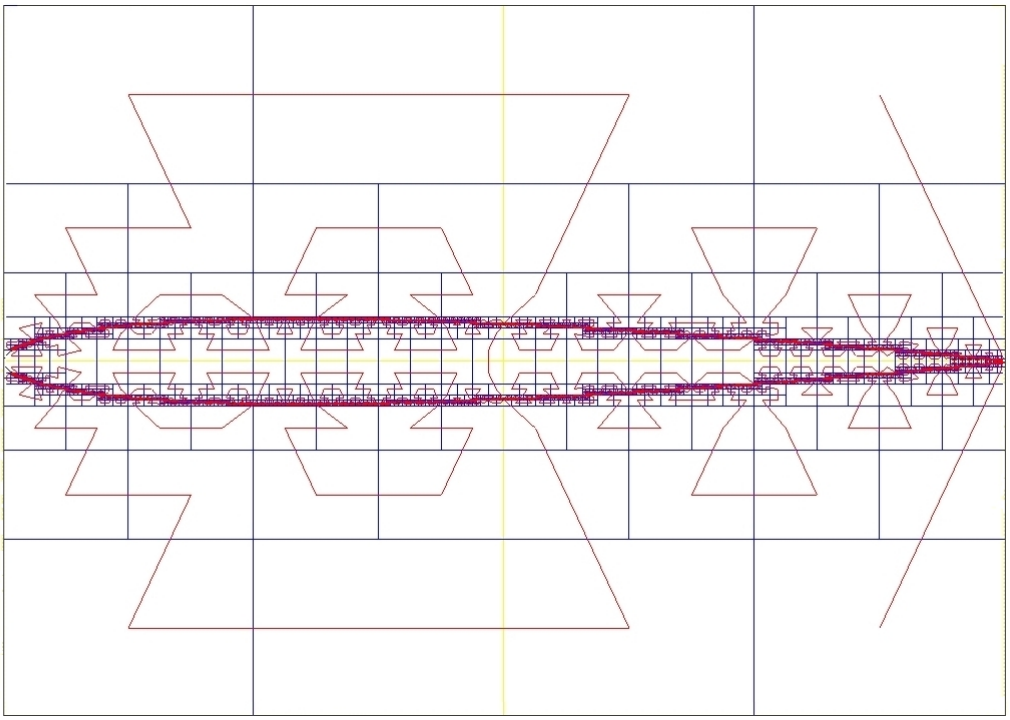


Figura 6.2: Modelo NACA com nível de refinamento 13.

Capítulo 7

Considerações Finais

Esse trabalho apresenta duas estratégias baseadas no método Multigrid para resolução de equações diferenciais parciais com refinamento adaptativo da malha. A primeira abordagem realiza uma modificação na estrutura de dados do ALG. A nova estrutura tem um comportamento híbrido, se comporta como uma árvore mantendo informações de níveis anteriores da malha computacional e mantendo as informações da malha computacional em um grafo de folhas autonômas (ALG). Dessa forma, as informações referentes aos níveis de malha mais grossa podem ser obtidos permitindo assim a aplicação do Multigrid. Cada nível do Multigrid é representado por uma curva de Hilbert modificada.

A segunda abordagem associa ao ALG o método do Multigrid Algébrico. Este método avalia o sistema linear e a partir dele constrói uma hierarquia de sistemas semelhantes a hierarquia de malhas do Multigrid Geométrico. Em seguida são definidos os operadores de transferência baseados nas escolhas dos elementos que fazem parte de cada nível dos sistemas. Por fim, um resolutor de sistemas lineares completa os passos do algoritmo. Essa abordagem foi decorrente de um estudo estudo computacional realizado sobre as matrizes de discretização apresentadas em (Burgarelli et al., 2006). Os resultados encontrados demonstraram que os autovalores dessas matrizes apresentavam um comportamento que prejudica a velocidade de convergência do método dos Gradientes Conjugados.

As abordagens apresentadas demonstram uma melhoria no tempo de processamento do métodos. Todavia, a segunda abordagem apresenta resultados melhores pois seu custo de armazenamento em memória é bem menor. Enquanto a primeira abordagem armazena informações de cada célula, a segunda armazena somente as matrizes do sistema que representariam níveis menos refinados.

A melhora apresentada pode ser ainda maior com a utilização de estratégias de pro-

gramação paralela. O paralelismo dos método dos gradientes conjugados utilizando uma abordagem para ambientes de memória compartilhada já foi apresentado em (Oliveira et al., 2012b). Esta abordagem pode ser utilizada também na fase de suavização do Multigrid, tanto algébrico quanto geométrico.

Uma avaliação mais detalhada dos parâmetros envolvidos na fase de inicialização do AMG também deve ser realizada. Ganhos significativos nessa fase poderão se obtidos conforme os resultados apresentados em (Suero et al., 2012).

Uma abordagem semi-geométrica também é possível. As informações geométricas de posicionamento dos nodos pretos podem ser utilizadas para uma abordagem algébrica. Ou seja, a partir da informação geométrica é determinada uma partição do sistema linear referente à malha mais refinada e a partir disso seguem as etapas do multigrid algébrico.

Ainda nesse contexto, novas estratégias de particionamento de grafos ou mesmo de análise multiresolução deverão ser avaliados de forma a reduzir o tempo necessário na fase de inicialização do AMG.

O desenvolvimento de um algoritmo para o método dos gradientes conjugados em que as direções de busca são definidas a partir de uma hierarquia dos espaços definidos pelo resíduo deverá ser avaliada.

A transferência entre os vetores definidos em cada espaço é realizada pelos operadores de restrição e prolongamento, caracterizando assim como um algoritmo multigrid. Esse procedimento já foi realizado para malhas uniformes (Pflaum, 2008). Entretanto, até o presente momento ainda são desconhecidas pelo autor referências que utilizem essa abordagem no contexto de refinamento adaptativo de malhas.

Um estudo sobre o método multigrid em cascata no contexto de refinamentos adaptativos também deverá ser realizado. Tais métodos apresentam uma grau de eficiência elevado além de uma facilidade de implementação para problemas simétricos como os apresentados aqui.

A utilização do AMG e do MGA como pré−condicionadores para o MGC também deve ser avaliada. Pré−condicionadores melhoram o número de condicionamento da matriz de discretização auxiliando assim na convergência de métodos iterativos como o MGC. Além disso, estimativas teóricas garantem o número máximo de iterações para os métodos iterativos, enquanto que para os métodos Multigrid essa análise é inexistente. Portanto, tal associação MGC + AMG/MGA produz um método muito mais eficiente.

Um estudo comparativo entre o ALG associado ao AMG com a estrutura FTT (Khokh-

lov, 1998) deverá ser realizado bem como com o algoritmo do Multigrid Adaptativo aqui apresentado.

Por fim, a aplicação desses métodos em problemas reais também deverá ser realizada.

Referências

Axelsson, O. **Iterative Solution Methods**. 1st. ed. AustraliaCambridge University Press, 1996. (1, 1).

Bank, R. **PLMTG: A software package for solving elliptic partial differential equations.** 1st. ed. : SIAM - Society for Industrial and Applied Mathematics., 1990.

Bank, R.E.; Douglas, C. **Sharp Estimates for Multigrid Rates of Convergence with General Smoothing and Acceleration**. *Jounal on Numerical Analysis*, v. 22, p. 617–633, 1985.

Brandão, D.; Oliveira, S. e Kischinhevsky, M. **Finite-Element Non-conforming h-adaptive strategy based on autonomous leaves graph**. *Lecture Notes in Computer Science*, v. 5544, p. 570–579, 2009.

Brandt, A. **Multi-level adaptive solutions to boundary-value problems**. *Mathematics of Computation.*, v. 31, n. 138, 1977.

Brandt, A.; McCormick, S. e Ruge, J. **Algebraic Multigrid (AMG) for automatic multigrid solution with application to geodetic computations**. Institute for Computational Studies, Colorado. Technical Report, October 1982.

Brezina, M.; Garcia, J.; Manteuffel, T.; McCormick, S.; Ruge, J. e Tang, L. **Parallel Adaptive Mesh Refinement for First-order system least squares**. *Numerical Linear Algebra with Applications*, v. 19, p. 343–366, January 2012.

Briggs, W.; Henson, V. e Mc Cormick, S. **A Multigrid Tutorial**. 2nd. ed. ColoradoSIAM - Society for Industrial and Applied Mathematics., 2000.

Burden, R. e Faires, J. **Numerical Analysis.** 8th. ed. YoungstownThomson, 2004.

Burgarelli, D. **Modelagem Computacional e Simulação numérica adaptativa de equações diferenciais parciais evolutivas aplicadas a um problema termoacústico.** Tese (Doutorado) — Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 1998.

Burgarelli, D.; Kischinhevsky, M. e Biezuner, R. **A new adaptive mesh refinement strategy for numerically solving evolutionary PDE's**. *Journal of Computational and Applied Mathematics*, v. 196, p. 115–131, 2006.

Clearly, A.; Falgout, R.; Henson, V.; Jones, J.; Manteuffel, T.; Mc Cormick, S.; Miranda, G. e Ruge, J. **Robustness and Scalability of Algebraic Multigrid.** *SIAM Journal on Scientific Computation.*, v. 21, p. 1886–1908, 2000.

Deitel, H. e Deitel, P. **C++ How to Program**. 3th. ed. New Jersey, USABookman, 2005. (1, 1).

Drozdek, A. **Data Structures and Algorithms in C++**. 4th. ed. Boston, USACengage Learning, 2012. (1, 1).

Fedorenko, R. **The Speed of Convergence of one iterative process.** *USSR Comp. Math. Math. Phys.*, v. 3, n. 4, p. 227–235, 1964.

Forgsgren, A. **On The Behavior of the Conjugate-Gradient Method on Ill-Conditioned Problems**. Royal Institute of Technology, Stockholm, Sweden, January 2006.

Garvine, R. **Upstream Influence in Viscous Interaction Problems**. *Physics of Fluids*, v. 11, n. 7, p. 1413–1423, July 1968.

Giraud, L.; Vasquez, F. e Tuminaro, R. **Grid Transfer Operators for Highly Variable Coefficient Problems in two-level Non-overlaping Domain Decomposition Methods.** *Numerical Linear Algebra with Applications*, v. 10, n. 5-6, p. 467–484, July-September 2003.

Golub, G. e Van Loan, C. **Matrix Computations**. 3. ed. : John Hopkins University Press, 1996.

Haase, G. e Langer, U. **Multigrid Methods from Geometrical to Algebraic Version**. In: *Modern Methods in Scientific Computing and Applications*, 1st. ed. : Springer, 2002, (NATO Science Series, 1). p. 103–152.

Hart, L.; McCormick, S. e Gallagher, A. **The fast adaptive composit-grid method algorithms for advanced computers**. *Applied Mathematics and Computation*, v. 19, p. 103–125, 1986.

Hartmann, D.; Meinke, M. e Schröder, W. **An adaptive multilevel multigrid formulation for cartesian hierarchical grid methods.** *Computers and Fluids*, v. 37, p. 1103–1125, December 2008.

Iwamura, C.; Costa, F.; Sbarski, I.; Easton, A. e Li, N. **An efficient Algebraic Multigrid Preconditioned Conjugated Gradient Solver**. *Comput. Methods Appl. Mech. Engrg.*, n. 192, p. 2299–2318, 2003.

Ji, H.; Lien, F. e Yee, E. **A new adaptive mesh refinement data structure with an application to detonation**. *Journal of Computational Physics*, v. 229, p. 8981–8993, August 2010.

Ji, H.; Lien, F. e Yee, E. **Parallel Adaptive Mesh Refinement Combined with Additive Multigrid for the Efficient Solution of the PoissonEquation**. *ISRN Applied Mathematics*, v. 1, n. Article ID 246491, p. 24 pages, 2012.

Jones, A. C. e Jimack, P. K. **An adaptive multigrid tool for elliptic and parabolic systems**. *International Journal for Numerical Methods in Fluids*, v. 00, p. 1–6, September 2000.

Jost, J. **Partial Differential Equations**. 1st. ed. Nova YorkSpringer-Verlag, 2002. (Graduate Texts in Mathematics, 214).

Khalil, M. e Wesseling, P. **Vertex-Centered and Cell-Centered Multigrid for Interface Problems**. *Journal of Computational Physics*, v. 98, n. 1, p. 1–10, 1992.

Khokhlov, A. M. **Fully Threaded Tree Algorithms for Adaptive Refinement Fluid Dynamics Simulations**. *Journal of Computational Physics*, v. 143, n. CP985963, p. 519–543, February 1998.

Kwak, D. **V-Cycle Multigrid for Cell-Centered Finite Differences.** *SIAM Journal on Scientific Computation.*, v. 21, n. 2, p. 552–564, September 1999.

Kwak, D. e Lee, J. **Multigrid Algorithm for the Cell-Centered Finite Difference Method II: Discontinuous Coefficient Case**. *Numerical Methods for Partial Differential Equations*, v. 20, n. 5, p. 742–764, 2004.

Li, M.; Cheng, H. P. e Yeh, G. T. **An adaptive multigrid approach for the simulation of contaminant transport in the 3d subsurface**. *Computers and Geosciences*, v. 31, p. 1028–1041, March 2005.

Liu, C. **Modifications of Steepest Descent Method and Conjugated Gradient Method Against Noise for Ill-posed Linear Systems**. *Communications in Numerical Analysis*, v. 2012, n. 1, p. 24, 2012.

Maliska, C. R. **Transferência de Calor e Mecânica dos Fluidos Computacional**. 2nd. ed. FlorianopolisLTC Editora, 2004.

Martin, D. e Cartwright, K. **Solving Poisson's Equation using Adaptive Mesh Refinement**. U.C. Berkeley, Berkeley. Techincal Report, October 1996.

Martin, H. **Cell-Centered Multigrid with Higher-Order Transfer Operators in waLBerla.** DissertaÁ„o (Bachelor Thesis advisor Ulrich Rüde) — Friedrich-Alexander-Universität Erlagen-Nürnberg, Nürnberg, Germany, March 2013.

Mavriplis, D. **Multigrid Solution Strategies for Adaptive Meshing Problems**. Institute for Computer Applications in Science and Engineering, NASA, Hampton, VA, March 1995.

McCormick, S. **Multilevel adaptive methods for partial differential equations**. 1st. ed. PennsylvaniaSIAM - Society for Industrial and Applied Mathematics., 1989.

McCormick, S. e Rude, U. **A finite volume convergence theory for the fast adaptive composite grid methods.** *Applied Numerical Mathematics*, v. 14, n. 0, p. 91–103, 1994.

McCormick, S. e Thomas, J. **The fast adaptive composite grid (FAC) method for elliptic equations**. *Mathematics of Computation.*, v. 46, n. 174, p. 439–456, 1986.

Miyashita, H. e Yamada, Y. **Practical improvements of multi-grid iteration for adaptive mesh refinement method**. *Fluid Dynamics Research*, v. 26, n. 1, p. 137–152, 2005.

Mohr, M. e Wienands, R. **Cell-Centered Multigrid Revisited.** *Computing and Visualization in Science*, v. 7, n. 3-4, p. 129–140, 2004.

Oliveira, R.; Rocha, B.; Burgarelli, D.; Meira Jr, W. e Santos, R. **An Adaptive Mesh Algorithm for the Numerical Solution of Electrical Models of the Heart An Adaptive Mesh Algorithm for the Numerical Solution of Electrical Models of the Heart An Adaptive Mesh Algorithm for the Numerical Solution of Electrical Models of the Heart.** *Lecture Notes in Computer Science (ICCSA)*, v. 7333, n. 1, p. 649–664, 2012.

Oliveira, R.; Rocha, B.; Burgarelli, D.; Meira Jr, W. e Santos, R. **A parallel accelerated adaptive mesh algorithm for the solution of electrical models of the heart**. *Internation Journal of High Performance Systems Architecture*, v. 4, n. 2, p. 89–100, September 2012.

Oliveira, S. e Kischinhevsky, M. **A graph-based adaptive triangular-mesh refinement applied to classical elliptic and parabolical problems**. In: SBMAC. *Anais do 32o. Congresso de Matemática Aplicada e Computacional*, São Paulo, v. 1, n. 1, p. 607–612, September, 2009.

Oliveira, S.; Kischinhevsky, M. e Burgarelli, D. **Finite Volume Adaptive Mesh Refinement based on graph applied to the boundary layer problem**. *IEEE Latin America Transactions*, v. 9, n. 1, p. 102–108, March 2011.

Pereira, F. **O método Multigrid Algébrico na resolução de sistemas lineares oriundos do método dos elementos finitos.** Tese (Doutorado) — Escola Politécnica da Universidade São Paulo, São Paulo, 2007.

Pflaum, C. **A multigrid conjugated gradient method**. *Applied Numerical Mathematics*, v. 58, p. 1803–1817, December 2008.

Rivara, M. **Design and Data Structure of Fully Adaptive, Multigrid, Finite-Element Software**. *ACM Transactions on Mathematical Software*, v. 10, n. 3, p. 242–264, September 1984.

Robaina, D.; Brandão, D.; Kischinhevsky, M.; Oliveira, S.; Montenegro, A. e Clua, E. **An adaptive graph for volumetric mesh visualization.** *Procedia Computer Science*, v. 1, n. 1, p. 1741–1749, 2010.

Roseberg, I. e Stenger, F. **A lower bound on the angles of triangles constructed by bisecting the longest side**. *Mathematics of Computation*, v. 29, n. 1, p. 390–395, 1975.

Rüde, U. **Fully Adaptive multigrid methods**. *SIAM Journal of Numerical Analysis*, v. 30, n. 1, p. 230–248, 1993.

Rüde, U. **On the multilevel adaptive iterative method**. *SIAM Journal on Scientific Computation.*, v. 15, n. 3, 1994.

Ruge, J. e Stüben, K. **Algebraic Multigrid (AMG)**. In: *Multigrid Methods*. PhiladelphiaSociety for Industrial and Applied Mathematics, 1986, (Frontiers in Applied Mathematics, 1). p. 1–57.

Salinas, P.; Rodrigo, C.; Gaspar, F. e Lisbona, F. **An efficient cell-centered multigrid method for problems with discontinuous coefficients on semi-structured triangular grids.** *Computers and Mathematics with Applications*, v. 65, n. 1, p. 1978–1989, April 2013.

Samet, H. **The Design and Analysis of Spatial Data Structures**. MAAddison-Wesley, 1990.

Shi, Z.; Xu, X. e Man, H. **Cascadic Multigrid for Finite Volume Methods for Elliptic Problems**. *Journal of Computational Mathematics*, v. 22, n. 6, p. 905–920, 2004.

Stüben, K. **A review of algebraic multigrid**. *Journal of Computational and Applied Mathematics*, v. 128, p. 281–309, 2001.

Suero, R.; Pinto, M.; Marchi, C.; Araki, L. e Alves, A. **Analysis of Algebraic Multigrid Parameters for Two-Dimensional Steady-State Heat Diffusion Equations.** *Applied Mathematical Modelling*, v. 36, n. 7, p. 2996–3006, 2012.

Tang, Z. e Yuan, J. In: . Arkadan, A.R.A. editor (Ed.). *14th Biennial IEEE Conference on Electromagnetic Field Computation*, v. 1, n. 1, **A combination of algebraic multigrid method and adaptive mesh refinement for large-scale electromagnetic field calculation**, p. 1, 2010.

Thorne, D. **Multigrid with Cache Optmizations and Adaptive Mesh Refinement Hierarchies**. Tese (Phd Thesis) — University of Kentucky, Kentucky, 2003.

Trottenberg, U.; Oosterlee, C. e Schüller, A. **Multigrid**. 1st. ed. London.Academic Press, 2001.

Unfer, T.; Boeulf, J. e Rogier, F. **Multi-scale gas discharge simulations using asynchronous adaptive mesh refinament**. *Computer Physics Communications*, v. 181, p. 247–258, 2010.

Wesseling, P. **Cell-Centered Multigrid for Interface Problems**. *Journal of Computational Physics*, v. 79, n. 1, p. 85–91, 1988.

Wesseling, P. **An Introduction to Multigrid Methods**. : John Willey & Sons Ltd., 1992. (Reprinted by MGNET, 1).

Wirth, N. **Algorithms + data structure = programs.** 2nd. ed. Palo AltoPrentice-Hall, 1985.

Wu, C. e Elman, H. **Analysis and Comparison of Geometric and Algebraic Multigrid for Convection-Difussion Equations**. *SIAM Journal on Scientific Computation*, v. 28, n. 6, p. 2208–2228, 2006.