PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ
ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

FLORENTINO AUGUSTO FAGUNDES

MÉTODO *MULTIGRID* ALGÉBRICO VIA *WAVELETS* APLICADO A PROBLEMAS 1D E 2D EM ELASTICIDADE E ADVECÇÃO E DIFUSÃO

CURITIBA
2013

FLORENTINO AUGUSTO FAGUNDES

MÉTODO *MULTIGRID* ALGÉBRICO VIA *WAVELETS* APLICADO A PROBLEMAS 1D E 2D EM ELASTICIDADE E ADVECÇÃO E DIFUSÃO

Tese apresentada ao Programa de Pós Graduação em Engenharia Mecânica, área de concentração Mecânica dos Sólidos, da Escola Politécnica da Pontifícia Universidade Católica do Paraná, como requisito parcial à obtenção do título de Doutor em Engenharia Mecânica.

Orientador: Dr. Roberto Dalledone Machado.

Coorientadora: Dra. Viviana Cocco Mariani.

CURITIBA

2013

Termo de Aprovação

Florentino Augusto Fagundes

MÉTODO *MULTIGRID* ALGÉBRICO VIA *WAVELETS* APLICADO A PROBLEMAS 1D E 2D EM ELASTICIDADE E ADVECÇÃO E DIFUSÃO

Tese aprovada como requisito parcial para a obtenção do título de doutor em Engenharia Mecânica no programa de Pós Graduação em Engenharia Mecânica da Pontifícia Universidade Católica do Paraná, PPGEM, na área de Mecânica dos Sólidos pela seguinte banca examinadora:

Prof. Dr. Roberto Dalledone Machado
Pontifícia Universidade Católica do Paraná (PUCPR)

Prof.ª Dra. Viviana Cocco Mariani
Pontifícia Universidade Católica do Paraná (PUCPR)

Prof. Dr. Carlos Henrique Marchi
Universidade Federal do Paraná (UFPR)

Prof. Dr. Fabio Henrique Pereira
Universidade Nove de Julho (UNINOVE)

Prof. Dr. João Elias Abdalla filho
Pontifícia Universidade Católica do Paraná (PUCPR)

Curitiba, 30 de agosto de 2013.

AGRADECIMENTOS

Inicio agradecendo a todas as pessoas que de forma anônima contribuíram para este trabalho.

Agradeço aos professores do PPGEM da PUCPR, em especial aqueles que foram meus professores durante os créditos. Agradecimento que se estende aos coordenadores do Programa e a secretária Jane, bem como aos demais colegas do curso.

Agradecimento aos orientadores, professor Dr. Roberto Dalledone Machado e professora Dra. Viviana Cocco Mariani, pelas contribuições ao desenvolvimento deste trabalho.

Agradecimentos aos professores Dr. Carlos Henrique Marchi, PPGMNE/UFPR; e Dr. Fabio Henrique Pereira, UNINOVE, professores que aceitaram participar de minha banca de qualificação, e também da defesa, e que no primeiro momento contribuíram de forma criteriosa, apontando falhas e propondo soluções, prevendo dúvidas e se colocando a disposição para saná-las; e no segundo momento em que enriqueceram o trabalho propondo ajustes e melhorias. Entre a qualificação e a defesa, o professor Fábio me recebeu na UNINOVE, ocasião inclusive em que tive acesso ao seu código computacional do *multigrid* utilizado como precondicionador. Agradecimento que se estende ao Prof. Dr. João Elias Abdalla Filho que também contribuiu com sugestões na ocasião da defesa.

Agradeço a PUCPR pela bolsa concedida, isentando-me das mensalidades.

Agradecimento especial à família, principalmente à esposa Maria Luiza, à filha Luciana, e ao filho Lucas por aceitarem e compreenderem as ausências exigidas pelo trabalho de titulação concomitante às atividades profissionais do magistério.

E finalmente agradeço ao Criador, por tudo.

RESUMO

Neste documento são avaliadas soluções numéricas para problemas da Engenharia Mecânica por meio do acelerador de convergência *multigrid* algébrico via *wavelets* (WAMG). É uma abordagem recente para o *multigrid* algébrico (AMG), e que utiliza a Transformada Discreta *Wavelet* para obter uma aproximação da matriz do sistema em cada nível do m*ultigrid*, assim como para efetuar as operações de restrição e prolongação. Os resultados são comparados em termos de tempo de processamento e número de ciclos com o método do gradiente conjugado precondicionado (GCP). O WAMG é utilizado também como um precondicionador (PWAMG), e os resultados são comparados com o WAMG e com o GCP. No caso do WAMG, tanto na montagem das matrizes para os diferentes níveis do *multigrid*, quanto para os operadores de restrição e prolongação, a transformada discreta de w*avelets* via banco de filtros é utilizada. Mais precisamente, são utilizados os filtros Haar de comprimento dois e o filtro de Daubechies (1988) de comprimento quatro. Tais problemas da Engenharia Mecânica quando discretizados culminam em matrizes esparsas, que requerem métodos iterativos para resolução, devido a grande dimensão do sistema resultante. Nestes casos, métodos iterativos clássicos, como o Gauss-Seidel (GS), se tornam lentos à medida que se aumenta o tamanho da matriz e se atribui uma tolerância pequena para o erro; e o *multigrid* se torna uma ferramenta eficaz para diminuir o tempo de processamento. Como o WAMG é uma abordagem nova para o AMG, e ainda inexplorada na área da Engenharia Mecânica, optou-se em apresentar resultados satisfatórios e não satisfatórios para os casos investigados.

Palavras-chave: Métodos Numéricos. *Multigrid* Algébrico. *Wavelet.* Precondicionamento. *Multigrid* Algébrico via *Wavelet.*

ABSTRACT

In this document numerical solutions to Mechanical Engineering problems are evaluated through convergence accelerator algebraic multigrid via wavelets (WAMG). It is a new approach for the algebraic multigrid (AMG), which uses the Discrete Wavelet Transform to obtain a matrix approximation system at each multigrid level, as well as to perform the restriction and prolongation operations. The results are compared in terms of processing time and number of cycles with the preconditioned conjugate gradient method (PCG). The WAMG is also used as a preconditioner (PWAMG), and the results are compared with PCG and the WAMG. In the case of WAMG, both in the assembly of the matrices for the different multigrid levels, and for the restriction and prolongation operators, the discrete wavelet transform via filter bank is utilized. More precisely, the Haar filter length two (1988), and Daubechies filter length four (1988) are used. Such problems as discrete Mechanical Engineering culminate in sparse matrices, which require iterative methods for solving due to large size of the resulting system. In these cases, classics iterative methods such as Gauss - Seidel (GS), become slow as it increases the size of the array and assigns a small tolerance for error, and the multigrid becomes an effective tool for reducing the time processing. As WAMG is a new approach for AMG, and unexplored area of Mechanical Engineering, was chosen on the satisfactory results and not for the cases investigated.

Keywords: Numerical Methods. Algebraic Multigrid. Wavelet. Preconditioning. Algebraic Multigrid via Wavelet.

LISTA DE FIGURAS

LISTAS DE TABELAS

LISTA DE ALGORITMOS

LISTA DE SÍMBOLOS E ABREVIATURAS

a_{ij}	Elementos da matriz A
A	Matriz de coeficientes
A^{h}	Matriz de coeficientes na malha fina
A^{2h}	Matriz de coeficientes na malha grossa
AMG	*Multigrid* algébrico, notação em inglês
$A\mathbf{u} = \mathbf{f}$	Sistema de equações lineares
C	Conjunto dos pontos da malha grosseira do AMG
CS	Esquema de correção, notação em inglês
e	Vetor erro
ED	Equação diferencial
EDP	Equação diferencial parcial
F	Conjunto dos pontos da malha refinada, utilizada no AMG
FAS	Esquema de aproximação completa, notação em inglês
FMG	*Multigrid* completo, notação em inglês
GC	Método do Gradiente Conjugado
GCP	Método do Gradiente Conjugado Precondicionado
GMG	*Multigrid* geométrico, notação em inglês
h	Espaçamento entre os pontos de uma discretização
H	Tamanho de um elemento unidimensional genérico
I_h^{2h}	Operador de restrição
I_{2h}^{h}	Operador de prolongação
K_t	Condutividade térmica
max	Máximo elemento de um vetor
MB	Unidade de medida de informação na memória (MegaByte)
MDF	Método das diferenças finitas
MEF	Método dos elementos finitos
N	Número de elementos numa malha
n_x	Tamanho de uma discretização na direção x

n_y	Tamanho de uma discretização na direção y
$\mathbb{N}$	Conjunto dos números naturais
PVC	Problema de valor do contorno
q	Vetor fluxo de calor
r	Vetor resíduo
$\mathbb{R}$	Conjunto dos números reais
RAM	*Random Access Memory*
S(A)	Espectro de uma matriz A
SOR	Método de sobrerrelaxação sucessiva, notação em inglês.
u	Solução exata de um sistema de equações lineares
u_i	i-ésimo elemento do vetor **u**
$\boldsymbol{T}$	Temperaturas
v	Solução inicial ou aproximada de um sistema de equações lineares
V	Subespaço vetorial gerado por translação e dilatações da função *wavelet*
W	Subespaço vetorial gerado por translação e dilatações da função escaladora
$\mathbb{Z}$	Conjunto dos números inteiros

Letras gregas

$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$	Operador Laplaciano, em coordenadas cartesianas
λ	Autovalor
ρ	Raio espectral
ϕ	Função escaladora
$\psi_{a,b}$	Função *wavelet* contínua
$\psi_{i,j}$	Função *wavelet* discreta
ω	Fator de relaxação
ω_{ij}	Pesos da interpolação para o AMG
Ω^h	Região discreta (malha) de um espaço real (malha refinada)

Ω^H	Região discreta (malha) de um espaço real (malha grosseira)
$< , >$	Produto interno
$<<$	Muito menor
$\approx$	Aproximadamente
$\oplus$	Soma direta de espaços vetoriais
$\varnothing$	Conjunto vazio

Subíndices

h	Malha fina
2h	Malha grossa
H	Malha grossa genérica

SUMÁRIO

1 INTRODUÇÃO

Um problema da Engenharia Mecânica pode ser resolvido por meio de modelos experimentais ou numéricos. A abrangência dos problemas não permite uma generalização de que uma abordagem é mais adequada em relação à outra. Em geral, modelos numéricos são preferíveis para solucionar tais problemas em comparação com modelos experimentais porque os casos podem acarretar grandes dificuldades de ordem operacional e financeira. Um modelo matemático, por sua vez, pode ser resolvido analítica ou numericamente. Embora uma precisão teórica seja alcançada na primeira possibilidade, isto não se traduz geralmente em resultados para os problemas específicos e complexos do dia a dia. Soluções numéricas, por sua vez, necessitam que o domínio físico seja discretizado via Método dos Elementos Finitos (MEF), Método das Diferenças Finitas (MDF), Método dos Volumes Finitos, etc.

Uma vez discretizados por um método numérico, os modelos matemáticos são implementados através de códigos computacionais. Assim, graças aos avanços computacionais, os métodos numéricos se tornaram uma das principais ferramentas para análise dos problemas de engenharia com a complexidade que a realidade exige. Mas é preciso ter em mente que um método numérico, por melhor que seja, gera somente resultados aproximados inseridos no modelo utilizado. Por exemplo, se um modelo da Mecânica Estrutural não incluir os esforços cisalhantes, os resultados não trarão esta informação. Além disso, mesmo computadores dotados de processadores rápidos necessitam de algoritmos robustos para melhorar seu desempenho porque o gargalo dos métodos aproximados está na lentidão da convergência.

A técnica *multigrid* já é uma opção consagrada na aceleração de convergência, como será mostrado ao longo deste trabalho.

1.1 CONSIDERAÇÕES INICIAIS SOBRE O *MULTIGRID*

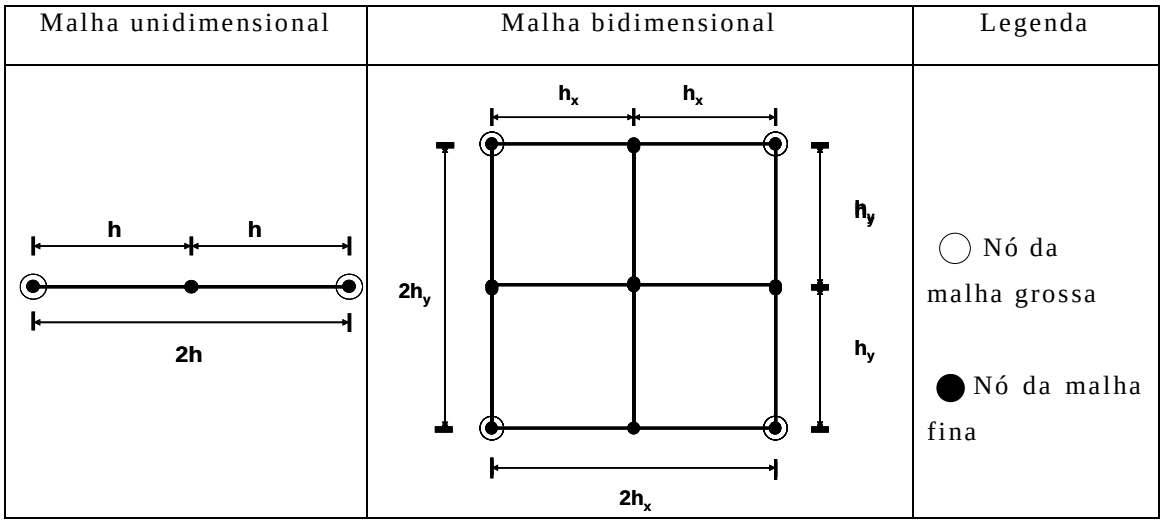
A técnica *multigrid* surgiu como alternativa para se reduzir o tempo de processamento na resolução numérica de modelos matemáticos. Atualmente ela é utilizada na resolução de problemas nas áreas de controle, otimização, reconhecimento de padrões, etc. Algumas aplicações são encontradas na área de Elasticidade (ADAMS e TAYLOR, 2000; MONTERO et al., 2001; GRIEBELet al., 2003; XIAO et al., 2007). Esta técnica pode ser associada a vários métodos numéricos tais como Elementos Finitos, Diferenças Finitas, Volumes Finitos, Elementos de Contorno, entre outros. Trata-se de uma técnica iterativa, que pode ser utilizada tanto em problemas lineares como não lineares, cuja vantagem na resolução de problema de valor de contorno, (PVC) é, em muitos casos, a redução do tempo de processamento (BRIGGS e HENSON, 2000).

A primeira etapa desta técnica é a discretização do domínio do PVC a ser resolvido. O resultado da discretização, independente da técnica empregada (MEF, MDF, etc.), é um sistema de equações lineares. A resolução do sistema de equações pode ser efetuada por meio de métodos diretos como o de Gauss, ou métodos iterativos, como, por exemplo, o método de Gauss-Seidel. O problema é que a resolução de sistemas de equações lineares, por métodos diretos ou iterativos tem um custo computacional elevado. Uma alternativa para isso é a técnica *multigrid*.

A técnica *multigrid* consiste em se trabalhar com várias malhas, passando-se a solução, ou o erro, sucessivamente por cada uma delas. Assim, o tamanho dos elementos para cada nível de uma malha uniforme, ou seja, malha em que todos os elementos tenham a mesma dimensão, pode ser obtido utilizando-se, por exemplo, fatores de 1:2 ou 1:4, até que, se necessário, a malha mais grosseira tenha apenas um nó interno. Adotando-se o fator de 1:2, um elemento numa malha qualquer possui o dobro do tamanho ($2h$) que possuía na malha imediatamente mais refinada (h) para o caso unidimensional. Utilizando-se o mesmo fator para o caso bidimensional, o elemento numa malha imediatamente mais grosseira teria o dobro do comprimento em direção a cada eixo coordenado, conforme mostra a figura 1.2. Observa-se que, para o caso bidimensional, um elemento numa malha

estruturada possui um quarto do tamanho em relação à malha imediatamente mais grosseira.

Figura 1.1: Elementos uni e bidimensionais em uma malha grossa e fina



Fonte: o autor, 2013.

Existem duas famílias de *multigrid*, o *Multigrid* Geométrico (GMG) (WESSELING, 1982; BRIGGS et al., 2000), indicado para problemas em que é possível estabelecer uma sequência de malhas, e o Algébrico (AMG) (BRIGGS et al., 2000; TROTTENBERG et al., 2001), útil quando é difícil ou impossível estabelecer uma hierarquia de discretizações. No *multigrid* algébrico todas as informações são obtidas a partir da matriz do sistema original. A dificuldade inicial do AMG é a escolha dos elementos da matriz do sistema da malha refinada que deverão representar o problema numa malha mais grosseira. Esta dificuldade, que será abordada com mais detalhes no capítulo 3, foi superada pela análise em multirresolução, por meio de funções *wavelets* (BRIGGS; HENSON, 1993; WANG et al., 2000; DE LEON, 2000, AVUDAINAYAGAM e VANI, 2004; PEREIRA, 2006; PEREIRA, 2007; GARCIA et al., 2008).

1.2 MOTIVAÇÃO

Como mencionado anteriormente, um sistema de equações lineares pode ser resolvido via métodos diretos ou iterativos. Num método direto como o da

eliminação de Gauss, a solução do sistema linear é obtida após um número finito de operações. Nos métodos iterativos a solução aproximada da equação é alcançada por meio de um número de operações, até que um determinado critério de parada seja atingido. Um método iterativo é preferível para a resolução de um sistema de equações lineares é de grande porte ou possui muitos elementos nulos (matriz esparsa) (BURDEN; FAIRES, 2003).

Os métodos diretos de solução de um sistema de equações lineares têm custo computacional da ordem de $O(n^3)$, enquanto os métodos iterativos clássicos de Jacobi e de Gauss-Seidel possuem um custo computacional da ordem $O(n^2)$, onde n é o número de incógnitas (BURDEN; FAIRES, 2003). Isto significa que, embora a ordem de convergência para tais métodos iterativos seja inferior aos métodos diretos, ainda assim tal custo cresce de modo não linear com o número de incógnitas do problema. Reside aí uma grande vantagem da técnica *multigrid* cujo custo computacional é proporcional ao número de incógnitas do problema, ou seja, idealmente são da ordem $O(n)$ (FALGOUT, 2006). Considerando que nos problemas reais da engenharia o número n pode chegar à casa dos milhões, a investigação de um método cujo custo computacional cresça linearmente com o número de incógnitas se justifica plenamente. Como o *multigrid* inicialmente proposto depende de um conjunto de malhas, e isto nem sempre é possível de se estabelecer, surgiu então o *multigrid* algébrico, cuja sigla em inglês é AMG, que independe de uma hierarquia de malhas. Dificuldades de ordem operacional para implementação do AMG, como a de escolha dos elementos da matriz, já citada anteriormente, inspiraram novas alternativas, entre elas o AMG via *wavelets*. *Wavelets* são funções originalmente utilizadas em tratamentos de sinais, que mais recentemente passaram a ser empregadas no AMG. Esta abordagem trouxe ganhos e desafios, uma vez que solucionou problemas não convergentes usando o *multigrid* geométrico, mas a um custo computacional muito alto (DE LEON, 2000). Alguns avanços já foram obtidos para baixar estes custos computacionais, entre eles cita-se o WAMG (PEREIRA, 2007).

Neste trabalho o WAMG é aplicado a problemas 1D e 2D em Elasticidade e Advecção e Difusão, implementado com filtros de comprimento dois e quatro.

1.3 OBJETIVOS

O objetivo geral desta tese é implementar o multigrid algébrico via transformada discreta de w*avelets* (WAMG), utilizando banco de filtros de Haar de comprimento 2 e de Daubechies de comprimento quatro (DAUBECHIES, 1988). Para o filtro de Haar a decimação será por dois, e para o filtro de Daubechies serão utilizadas decimação por dois e quatro. O WAMG será aplicado neste trabalho para solucionar os seguintes problemas

Casos unidimensionais

- Aplicar o algoritmo WAMG para o problema de condução de calor com termo fonte;
- Aplicar o algoritmo WAMG para o problema estrutural de barra e de viga.

Casos bidimensionais

- Aplicar o algoritmo WAMG para a equação de Laplace;
- Aplicar o algoritmo WAMG para o problema de difusão advecção;
- Aplicar o algoritmo WAMG para o problema da viga sob estado plano de tensões (EPT);
- Aplicar o algoritmo PWAMG para a equação de Laplace, problema de difusão e advecção, e problema da viga sob EPT.

Os objetivos específicos são comparar o desempenho obtido em termos de tempo de processamento e número de ciclos:

- Entre o WAMG e o GCP para os problemas 1D;
- Entre o WAMG, PWAMG e GCP para a equação de Laplace e da viga sob EPT;
- Entre o WAMG, PWAMG para o problema de difusão advecção;
- Determinar o número ótimo de iterações internas para os algoritmos do WAMG em cada caso;

- Mostrar a memória computacional utilizada pelo aplicativo em cada caso.

1.4 REVISÃO BIBLIOGRÁFICA

A primeira referência encontrada na literatura preocupada em acelerar a convergência dos processos iterativos data do ano de 1950. Trata-se de um texto escrito em russo (ABRAMOV, 1950 *apud* FEDORENKO, 1964). O próprio artigo do Fedorenko que cita Abramov também foi originalmente escrito em russo, e traduzido para o inglês por D. E. Brown; e propunha a solução numérica da equação de Laplace bidimensional discretizada por diferenças finitas no quadrado $0 \leq x, y \leq \pi$, com condições de contorno igual a zero. O termo referenciado em sua formulação matemática é ‘malha auxiliar’ (FEDORENKO, 1964).

O método *multigrid* geométrico foi desenvolvido nos anos de 1970 de forma independente por Brandt e Hackbusch, quando então foram publicados os primeiros trabalhos práticos (BRANDT, 1973; HACKBUSCH, 1976; HACKBUSCH, 1978). Outro pesquisador que deu grande contribuição para o desenvolvimento do método *multigrid* foi Wesseling, ao aplicá-lo em problemas não simétricos (WESSELING, 1982). Nos anos de 1980 foi desenvolvido o método *multigrid* algébrico, cuja sigla em inglês é AMG. O AMG não necessita de uma hierarquia de malhas, sendo as informações necessárias para construção do método obtidas diretamente da matriz original do problema. Sua aplicação é indicada quando é difícil construir uma sequência de discretizações necessárias para o *multigrid* geométrico (MCCORMICK, 1987).

Bank desenvolveu em 1988 o método *multigrid* hierárquico, cuja sigla em inglês é HBM. Nele quando uma malha é refinada, os resultados do nível anterior são preservados. Uma vantagem importante desta abordagem é que as matrizes obtidas em cada refino são melhor condicionadas que as obtidas com a base nodal padrão (BANK et al., 1988). A análise matemática do HBM mostra que ele compartilha das mesmas propriedades básicas do *multigrid*, mas sob condições menos rigorosas (JOUGLARD; COUTINHO, 1998).

Segundo Wesseling (1992), os melhores desempenhos do método *multigrid* são obtidos em problemas totalmente elípticos, ou seja, problemas dominados pela difusão. Já os menores desempenhos foram observados, segundo Ferziger e Peric (1999), em problemas dominados pela advecção.

Em 1993 Briggs e Henson identificaram similaridades entre a técnica *multigrid* e a análise em multirresolução. Isto permitiu conectar o *multigrid* geométrico e a teoria das *wavelets* (BRIGGS; HENSON, 1993).

Em 1996 Chang desenvolveu um algoritmo mais abrangente que o inicialmente teorizado para o AMG, no qual se permite que os elementos a_{ij} fora da diagonal principal da matriz *A* sejam negativos (CHANG et al., 1996).

Briggs estudou o problema de anisotropia geométrica, situação em que é conveniente efetuar engrossamento de uma malha bidimensional apenas na direção de um dos eixos coordenados, mantendo inalterado o tamanho do elemento na direção do outro eixo (BRIGGS et al., 2000). Ainda em 2000 Wang propôs um algoritmo no qual o *multigrid* é implementado utilizando expansão de *wavelets* para solucionar problemas de eletromagnetismo (WANG et al., 2000).

Brezina introduziu um novo algoritmo para o AMG, denominado AMGe, mais eficiente que o AMG, em se tratando de resolver equações diferenciais parciais discretizadas via elementos finitos (BREZINA et al., 2000).

De Leon (2008) introduziu o AMG via *wavelets* para matrizes simétricas. Apesar do grande avanço teórico de sua proposta, a qual permitiu solucionar problemas cuja convergência não acontecia com o uso do *multigrid* geométrico, a implementação de seu algoritmo era de elevado custo computacional, em virtude da necessidade de se obter a inversa da matriz do sistema.

Em alguns problemas como a simulação do voo de uma aeronave, necessita-se de uma maior acurácia dos resultados numéricos em volta das asas, exigência idêntica também ocorre em análise de tensões de um corpo onde sabidamente há uma grande concentração de tensões como, por exemplo, à frente da ponta de uma trinca. Para problemas desta natureza em que se necessita de refinos diferentes em regiões distintas do domínio, desenvolveu-se o método de malha composta adaptativa rápida, cuja sigla em inglês é FAC, *Fast Adaptive Composite grid method* (BRIGGS et al., 2000)

Avudainayagam e Vani exploraram as ideias iniciadas por Briggs e Henson obtendo sucesso em problemas lineares e não lineares (AVUDAINAYAGAM; VANI, 2004).

A questão da anisotropia geométrica investigada por Briggs (2000) inspirou muitas pesquisas, dentre elas Pinto (2006), Pinto e Marchi (2006). Pereira (2006), independente de Wang, propôs o algoritmo WAMG utilizando-o inclusive como precondicionador (PEREIRA et al., 2006b). Pereira implementou o WAMG na programação em paralelo (PEREIRA, 2007).

A grande vantagem do WAMG em comparação com o *multigrid* algébrico é a superação de algumas deficiências inerentes ao AMG, como, por exemplo, a dificuldade em identificar as relações de influência e dependência, e que será melhor detalhada no capítulo 3. O WAMG aproveita a técnica de multirresolução que supera os inconvenientes citados do *multigrid* algébrico. Para isto utiliza Transformada Discreta *Wavelet* (DWT) para construção das matrizes nos diferentes níveis do *multigrid*, assim como nas operações de restrição e prolongação.

Pereira (2007) e Pflaum (2008) utilizam como precondicionador o *multigrid* algébrico via *wavelets* e o *multigrid* geométrico respectivamente.

Garcia et al. (2008) propuseram variações do WAMG, criando os algoritmos WPAMG e WPAMG2, nos quais são considerados os erros de baixa e alta frequência. Os resultados numéricos foram similares aos obtidos por Wang (2000) e Pereira (2006), confirmando assim a competitividade do algoritmo WAMG. Cheng et al. (2012) contribuíram para a melhoria no WAMG ao desenvolver um algoritmo para realizar truncamento para zero dos elementos muito pequenos da matriz dos coeficientes após a aplicação da DWT no sistema de equações original, mantendo assim a característica de matriz esparsa no sistema modificado pela transformada discreta *wavelets*.

Suero et al. (2012) apresentaram um trabalho minucioso sobre a otimização dos parâmetros do AMG para as equações de Laplace e Poisson 2D, discretizadas em malhas triangulares e quadrangulares, a saber: quantidade de malhas, número de iterações internas, fator de redução de malha, e fator de forte dependência. O referido estudo é particularmente interessante em relação ao fator de redução de malha, o qual revelou que este parâmetro ótimo varia de problema para problema, e que para um mesmo

problema o referido valor ótimo também varia com o tipo de malha (triangular ou quadrangular) e com o número de iterações internas. Os resultados apresentados evidenciaram ainda mais as dificuldades inerentes a implementação do AMG, justificando assim a busca por novas abordagens para o *multigrid* algébrico, e que é objetivo desta tese.

Khelifi et al. (2013) desenvolveram um algoritmo para resolver problemas envolvendo convecção e difusão via AMG no qual a convecção e a difusão são tratadas separadamente. De Leon (2013) propôs modificação em seu próprio método defendido em 2000, agora intitulado *multigrid* via *wavelets* usando *wavelets* biortogonais simétricas, avaliando numericamente sua nova proposta em problema de difusão com coeficientes altamente oscilatórios, e em problema de advecção difusão com advecção moderadamente dominante.

A técnica *multigrid* é utilizada na resolução de problemas nas áreas de controle, otimização, reconhecimento de padrões, etc, (BRIGGS et al., 2000), sendo reconhecida pela literatura como uma das mais eficazes técnicas iterativas. Jouglard, (1998) após vários testes conclui que o método *multigrid*, embora muito superior ao método dos gradientes conjugados com precondicionamento diagonal, PCG-D, perde em eficácia para o método *Conjugate Gradients with Hierarchical Preconditioner* PCG-H. Embora Jouglard reconheça que seus testes são insuficientes para uma conclusão decisiva, cita o trabalho de Bank (1988) que chegou a mesma conclusão sobre a maior rapidez do método PCG em comparação com o método *multigrid*. Cabe aqui relembrar que as novas abordagens para o *multigrid* são posteriores as conclusões de Jouglard.

1.5 ORGANIZAÇÃO DO TEXTO

No capítulo 2 será apresentada a fundamentação teórica da técnica *multigrid*, assim como será descrito um resumo dos conceitos matemáticos necessários ao entendimento da técnica. Os métodos iterativos clássicos como Jacobi e Gauss-Seidel também são teorizados, bem como será discutido os modos de erros de Fourier. A seção é finalizada com algoritmos do *multigrid*

geométrico. No capítulo 3 o *multigrid* algébrico é descrito. No capítulo 4 as funções *wavelets* são apresentadas, assim como as similaridades entre elas e a técnica *multigrid*. São também apresentados algoritmos para implementação da prolongação e restrição dos diversos níveis do WAMG. O capítulo 5 mostra os problemas e os parâmetros investigados neste trabalho. No capítulo 6 são apresentados alguns resultados numéricos dos problemas propostos. No capítulo 7 o WAMG é utilizado como precondicionador. No oitavo capítulo são descritas as conclusões do trabalho, as contribuições da tese, assim como as sugestões para trabalhos futuros. Por fim, são apresentadas as referências bibliográficas.

2 FUNDAMENTAÇÃO TEÓRICA DA TÉCNICA MULTIGRID

Neste capítulo é apresentada a técnica *multigrid*, assim como os conceitos matemáticos básicos para seu entendimento e implementação.

A técnica *multigrid* é uma alternativa para se reduzir o tempo de processamento na resolução numérica de equações diferenciais e integrais, via métodos iterativos. Ela consiste basicamente em se trabalhar com várias malhas, passando-se a solução, ou o resíduo, sucessivamente em cada uma delas. O argumento básico é que métodos iterativos clássicos como o Gauss-Seidel suavizam o erro tornando difícil sua eliminação. A ideia então é transferir um vetor suave para uma malha mais grosseira tornando-o oscilatório, podendo assim ser eliminado em poucas iterações.

A primeira etapa desta técnica é a discretização do modelo, que pode ser feita por quaisquer dos métodos disponíveis como diferenças finitas, elementos finitos, etc. Numa etapa seguinte o sistema de equações precisa ser resolvido. Isto pode ser feito por meio de métodos diretos de solução ou por métodos iterativos. No caso do *multigri*d, obtém-se uma solução aproximada do modelo discretizado em algumas poucas iterações, até que o erro se torne suave. A ideia básica é transferir este erro suave para uma dimensão menor, tornando-o oscilatório. A fundamentação teórica do *multigrid* no que diz respeito a trabalhar com várias malhas é válida para todos os tipos, como *multigrid* geométrico que é apresentado neste capítulo, ou o *multigrid* algébrico e o *multigrid* algébrico via *wavelets* apresentados respectivamente nos capítulos 3 e 4. Ainda neste capítulo o *multigrid* geométrico será melhor detalhado. Antes, no entanto, se faz necessário apresentar alguns conceitos matemáticos básicos para um melhor entendimento da técnica *multigrid*, independente do tipo, se é o geométrico, o algébrico ou algébrico via *wavelets*.

A seguir define-se um sistema de equações lineares através da equação (2.1), bem como outros elementos necessários para resolução iterativa do referido sistema.

$$A\mathbf{u} = \mathbf{f}, \tag{2.1}$$

na qual o vetor **u** é a solução exata a ser encontrada do sistema, ou seja, é a incógnita do problema. Logo a matriz *A* e o vetor **f** são conhecidos. Para resolver a equação (2.1) usando métodos iterativos, faz-se necessário a introdução de uma solução aproximada dada pelo vetor **v**. Assim pode-se definir um vetor erro **e** dado por,

$$\mathbf{e} = \mathbf{u} - \mathbf{v} \tag{2.2}$$

Na prática a equação (2.2) é insuficiente, uma vez que não se conhece a solução exata **u**. Isto é resolvido pela determinação de um novo vetor **r** denominado resíduo

$$\mathbf{r} = \mathbf{f} - A\mathbf{v}. \tag{2.3}$$

Se os dois lados da equação (2.2) forem pré-multiplicados pela matriz *A*, tem-se:

$A\mathbf{e} = A(\mathbf{u} - \mathbf{v}) = A\mathbf{u} - A\mathbf{v}$. Como $A\mathbf{u} = \mathbf{f}$. Então $A\mathbf{e} = \mathbf{f} - A\mathbf{v}$, como $\mathbf{f} - A\mathbf{v} = \mathbf{r}$, chega-se a

$$A\mathbf{e} = \mathbf{r}. \tag{2.4}$$

Assim, a equação (2.4) estabelece uma relação entre o resíduo **r** e o erro **e**. Resolvendo esta equação para o erro **e**, uma vez que tanto A quanto **r** são conhecidos, chega-se a uma solução aproximada para o vetor **u**,

$$\mathbf{u} = \mathbf{v} + \mathbf{e}. \tag{2.5}$$

Finalmente, pode-se resolver iterativamente a equação (2.1) para **u** a partir de uma solução inicial **v**, estabelecendo um ou mais critérios de convergência.

2.1 CONCEITOS MATEMÁTICOS BÁSICOS

Na implementação da técnica *multigrid*, assim como dos métodos iterativos, é fundamental estabelecer um critério de parada do processamento quando se atinge algum parâmetro determinado, como por exemplo, a tolerância desejada para o caso. O critério de convergência mais importante nos métodos iterativos é o de se atingir um determinado nível de precisão numérica do problema a ser resolvido. Tal precisão é matematicamente obtida por meio de qualquer uma das normas usuais (BURDEN; FAIRES, 2003) como, por exemplo, a norma 2 também conhecida como norma Euclidiana, ou norma do infinito, definidas respectivamente para um vetor genérico $\mathbf{x} \in R^n$ por

$$\| \mathbf{x} \|_2 = \left(\sum_{j=1}^{n} x_j^2 \right)^{1/2} \tag{2.6}$$

$$\| \mathbf{x} \|_\infty = \max | x_j |, \ 1 \leq j \leq n. \tag{2.7}$$

Uma matriz A, n x n, é dita simétrica se seus elementos são tais que $a_{ij} = a_{ji}$ para $i, j = 1, 2, \ldots, n$. A matriz A^{-1} é chamada inversa da matriz A quando $A^{-1}A = A\,A^{-1} = I_n$, na qual I_n é a matriz identidade, ou seja, $I_n = \delta_{ij}$, com $\delta_{ij} = 1$ se i = j, $\delta_{ij} = 0$ se i ≠ j. A matriz A é definida positiva se satisfaz a expressão

$$\mathbf{x}^T A\mathbf{x} > 0, \text{ para todo } \mathbf{x} \in R^n, \ \mathbf{x} \neq \mathbf{0}. \tag{2.8}$$

Uma matriz simétrica e definida positiva é denominada *M*-matriz quando possui os elementos da diagonal principal positivos, $a_{ii} > 0$ e os elementos fora da diagonal negativos, $a_{ij} < 0$.

A norma matricial natural ou induzida de uma matriz A, n x n, associada com uma norma vetorial descreve como a matriz estende os vetores unitários relacionados com essa norma. A extensão máxima é a norma da matriz que é expressa por

$$\| A \| = \text{Máx } \| A\mathbf{x} \|, \text{ para } \| \mathbf{x} \| = 1, \ \| A \| \geq 0 \tag{2.9}$$

Uma matriz A, $n \times n$, é denominada singular quando não possui inversa. Caso possua inversa, A é dita não singular, ou inversível.

O número de condição de uma matriz não singular A, relativo à norma $\|.\|$ é

$$\text{Cond}(A) = \|A\|.\|A^{-1}\|, \text{Cond}(A) \geq 1. \tag{2.10}$$

Uma matriz A é bem condicionada se cond(A) estiver próximo da unidade. Caso cond(A) seja significativamente maior que 1, A é dita mal condicionada. O termo condição se refere à segurança relativa com que um vetor residual **r** pequeno implica num erro **e** também pequeno. Simbolicamente,

$$\text{Cond}(A) \approx 1 \rightarrow \mathbf{r} \approx \mathbf{0} \text{ e } \mathbf{e} \approx \mathbf{0}. \tag{2.11}$$

Uma função $f(t)$ pertence ao conjunto das funções quadrado integráveis, $L^2(\mathbb{R})$, se verificar as duas propriedades seguintes (CHUI, 1992).

1) A área total sob a curva é zero, ou seja,

$$\int_{-\infty}^{\infty} f(t)dt = 0 \tag{2.12}$$

2) A energia da função é finita, ou seja,

$$\int_{-\infty}^{\infty} |f(t)|^2 dt < \infty \tag{2.13}$$

Uma transformação linear T: $R^n \rightarrow R^m$ é uma aplicação que transforma um vetor do R^n num outro vetor do espaço R^m. Na prática o operador T trata-se de uma matriz T, n x m.

Um vetor $\mathbf{w} \neq \mathbf{0}$ é dito autovetor de uma matriz A, caso a matriz A transforme $\mathbf{w}$ num múltiplo de si mesmo, ou seja

$$A\mathbf{w} = \lambda\mathbf{w}. \tag{2.14}$$

na qual λ é um escalar. O escalar λ da equação acima é denominado autovalor da matriz A e o conjunto de todos os autovalores é o espectro de A, denominado por $S(A)$.

O raio espectral de uma matriz A, $\rho(A)$ é dado pelo máximo autovalor de A em valor absoluto, ou seja,

$$\rho(A) = \text{máx}\,|\lambda(A)|\,. \tag{2.15}$$

A norma 2 de uma matriz A, n x n, é definida pela expressão

$$\|\mathrm{A}\|_2 = \sqrt{\rho(\mathrm{A}^{\mathrm{T}}\mathrm{A})}\,. \tag{2.16}$$

na qual A^{T} é a transposta de A.

Subespaço de Krylov é um subespaço gerado pela sequência de vetores $\{\mathbf{x}, A\mathbf{x}, \ldots, A^{m-1}\mathbf{x}\}$, na qual m é a dimensão do subespaço. A expressão $\{A^{\mathrm{i}}\mathbf{x}\}$ é denominada sequência de Krylov (DATTA, 1995).

Teorema 2.1: A sequência A, A^2,... de potências da matiz A converge para a matriz zero se $|\lambda_{\mathrm{i}}| < 1$ para cada autovalor λ_{i} de A (DATTA, 1995). A reprodução da prova de Datta para este teorema se encontra no anexo B desta tese.

Afirmações 2.1: as afirmações seguintes são equivalentes (BURDEN; FAIRES, 2003).

1) A é uma matriz convergente.
2) $\lim_{n\to\infty}\left\|A^n\right\| = 0$, para alguma norma natural.
3) $\lim_{n\to\infty}\left\|A^n\right\| = 0$, para todas as normas naturais.

4) $\rho(A) < 1$.

5) $\lim_{n\to\infty} A^n \mathbf{x} = \mathbf{0}$, para todo **x**.

Definição (BURDEN; FAIRES, 2003).

Uma sequência $\left\{ \mathbf{X}^{(k)} \right\}_{k=1}^{\infty}$ de vetores em R^n converge para **x** em relação a norma $\|\cdot\|$ se, dado $\varepsilon > 0$, existe um inteiro $N(\varepsilon)$ tal que $\| \mathbf{x}^{(k)} - \mathbf{x} \| < \varepsilon$, para todo $k \geq N(\varepsilon)$.

O método das diferenças finitas (MDF) é uma técnica de discretização de uma Equação Diferencial. Aproxima-se a ED por meio da série de Taylor, truncando a expansão após alguns termos, considerando a ordem de precisão que se deseja. As expressões seguintes ilustram este argumento

$$f(x + h) = f(x) + f'(x)h + o(h^2) \tag{2.17}$$

Assim, a derivada pode ser expressa como uma diferença acrescida de um termo de erro

$$f'(x) = \frac{f(x+h) - f(x)}{h} + o(h) \tag{2.18}$$

Ignorando o termo de erro $o(h)$, obtem-se o operador de diferenças finitas para a primeira derivada de f

$$Df(x) = \frac{f(x+h) - f(x)}{h} \tag{2.19}$$

Há uma literatura abrangente sobre este tema, entre elas cita-se Datta (1995), Bathe (1996) e Burden e Faires (2003).

2.2 MÉTODOS ITERATIVOS: JACOBI E GAUSS-SEIDEL

Como já foi especificado anteriormente, um método iterativo para a resolução do sistema expresso pela equação (2.1) parte de uma solução inicial $\mathbf{v}^0$ que pode ser obtida de várias maneiras, inclusive de forma randômica (BURDEN; FAIRES, 2003). As iterações geram $\mathbf{v}^1$, ..., $\mathbf{v}^n$, ..., que idealmente converge para a solução exata **u**.

Para resolver o sistema (2.1) via métodos iterativos, inicia-se o processo com uma aproximação inicial $\mathbf{x}^{(0)}$ e a partir daí, gera-se uma sequência de vetores $\left\{\mathbf{x}^{(k)}\right\}_{k=1}^{\infty}$, que converge para **x** que idealmente é igual a **u**. Assim, uma técnica iterativa converte o sistema (2.1) ao sistema equivalente $\mathbf{v} = T\mathbf{v} + \mathbf{c}$ para alguma matriz T e algum vetor fixo **c**. A sequência de vetores para aproximar a solução é gerada calculando-se

$$\mathbf{v}^{(k)} = T\mathbf{v}^{(k-1)} + \mathbf{c}, \text{ para } k = 1, 2, 3,... \tag{2.20}$$

O método iterativo de Jacobi é um dos mais simples esquemas de resolução de um sistema de equações lineares, e consiste em resolver a i-ésima equação do sistema (2.1), para a i-ésima incógnita, na iteração k, conforme segue

$$v_i^{(k)} = \frac{\sum_{j=1, j\neq i}^{n}\left(-a_{ij}v_j^{(k-1)}\right) + f_i}{a_{ii}} \tag{2.21}$$

para $a_{ii} \neq 0$, $i = 1, 2, 3,...,n$.

O método iterativo de Gauss-Seidel consiste numa melhoria do método de Jacobi, no sentido de se aproveitar as aproximações das componentes v_1, ...,v_{i-1}, já atualizadas na iteração k. A componente v_i na k-ésima iteração é calculada conforme a equação seguinte

$$v_i^{(k)} = \frac{-\sum_{j=1}^{i-1}(a_{ij}v_j^{(k)}) - \sum_{j=i+1}^{n}\left(a_{ij}v_j^{(k-1)}\right) + f_i}{a_{ii}}, \tag{2.22}$$

para $a_{ii} \neq 0$, $i = 1, 2, 3,\ldots,n$.

Teoricamente é importante expressar os métodos iterativos em termos de matrizes. Para isso define-se uma decomposição da matriz A na forma

$$A = D - L - U, \tag{2.23}$$

na qual D é a diagonal de A, e L e U são, respectivamente as partes estritamente triangular inferior e estritamente triangular superior de A. Assim o sistema (2.1) pode ser expresso por $(D - L - U)\boldsymbol{u} = \boldsymbol{f}$, que pode ser resolvido para **u** conforme segue

$$\boldsymbol{u} = D^{-1}(L + U)\boldsymbol{u} + D^{-1}\boldsymbol{f}. \tag{2.24}$$

Assim, a equação (2.20) na forma matricial, fica

$$\mathbf{v}^{k} = D^{-1}(L + U)\mathbf{v}^{k-1} + D^{-1}\boldsymbol{f}. \tag{2.25}$$

Na equação (2.25), a matriz $S_J = D^{-1}(L + U)$ é chamada matriz de iteração do método Jacobi, e a convergência do método depende dos autovalores de S_J. Calculando a próxima iteração da equação (2.20)

$$\mathbf{v}^{k+1} = D^{-1}(L + U)\mathbf{v}^{k} + D^{-1}\boldsymbol{f}. \tag{2.26}$$

Como pode ser observado nas equações (2.20) e (2.21), a matriz S_J permaneceu inalterada.

Novamente substituindo a expressão (2.23) no sistema (2.1)

$$(D - L - U)\boldsymbol{u} = \boldsymbol{f} \Rightarrow (D - L)\boldsymbol{u} + U\boldsymbol{u} + \boldsymbol{f} \Rightarrow \boldsymbol{u} = (D - L)^{-1}U\boldsymbol{u} + (D - L)^{-1}\boldsymbol{f}$$

Da expressão acima, define-se $S_G = (D - L)^{-1}U$ como a matriz de iteração do método de Gauss-Seidel. Novamente levando a solução acima na equação (2.20):

$$\mathbf{v}^{k} = (D - L)^{-1}U\mathbf{v}^{k-1} + (D - L)^{-1}\boldsymbol{f}. \tag{2.27}$$

Calculando a próxima iteração na equação (2.27)

$$\mathbf{v}^{k+1} = (D - L)^{-1}U\mathbf{v}^{k} + (D - L)^{-1}\boldsymbol{f}. \tag{2.28}$$

Observando as equações (2.27) e (2.28), percebe-se que a matriz $S_G = (D - L)^{-1}U$ permanece inalterada de uma iteração para outra.

Um método iterativo que mantém constante uma matriz S, não necessariamente expressa por S_J ou $S_{G,}$ é denominado método estacionário (BRIGGS et al., 2000). Para uma abordagem teórica mais completa sobre método iterativo estacionário, bem como algoritmos para implementação computacional, ver Burden e Faires (2003). Para entender o comportamento de tais métodos, no que diz respeito à convergência, quando aplicados ao sistema (2.1) é usualmente utilizado o procedimento que segue. A fim de facilitar a análise de convergência, é conveniente trabalhar com o sistema homogêneo $A\mathbf{u} = \mathbf{0}$, uma vez que a solução $\mathbf{u}$ é conhecida, $\mathbf{u} = \mathbf{0}$, e o erro $\mathbf{e}$ produzido por uma aproximação $\mathbf{v}$ é naturalmente $\mathbf{e} = -\mathbf{v}$, pois $\mathbf{v} + \mathbf{e} = \mathbf{u} = \mathbf{0}$.

Para testar o artifício descrito no parágrafo anterior, será utilizada uma matriz tridiagonal, obtida pela discretização por diferenças finitas de segunda ordem da equação de Laplace em um domínio unitário unidimensional

$\{x \in \mathrm{R} \,/\, 0 \leq x \leq 1\}$ (BRIGGS et al., 2000), conforme equação (2.29).

$$\frac{1}{h^2}\begin{pmatrix} 2 & -1 & & & & & \\ -1 & 2 & -1 & & & & \\ & -1 & 2 & -1 & & & \\ & & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & \cdot & \\ & & & & \cdot & \cdot & -1 \\ & & & & & -1 & 2 \end{pmatrix}\begin{pmatrix} v_1 \\ v_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ v_{n-1} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{pmatrix} \tag{2.29}$$

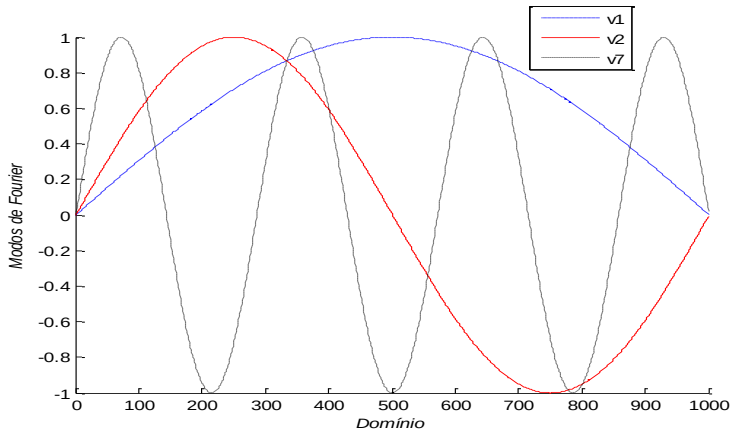
São utilizadas como estimativa inicial vetores **v** compostos de diferentes harmônicas, com baixa, média e alta frequência, dados por BRIGGS et al. (2000)

$$\mathbf{v}_j = sen(jk\pi/n), \qquad 1 \leq k \leq n-1, \qquad 0 \leq j \leq n, \tag{2.30}$$

nos quais k representa a frequência e indica o número de onda do vetor **v**, no domínio do problema. Os vetores **v** são denominados modos de Fourier. A figura (2.1) ilustra os vetores $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_7$ como se **v** fosse uma função contínua na variável j.

A figura 2.1 mostra que para pequenos valores de k, $\mathbf{v}_k$ é suave, tornando-se oscilatório à medida que o índice k aumenta. Ainda sobre o vetor dado pela equação (2.30), é importante adiantar que o mesmo fornece os autovetores da matriz A.

Figura 2.1: Modos de Fourier para uma, duas e sete ondas

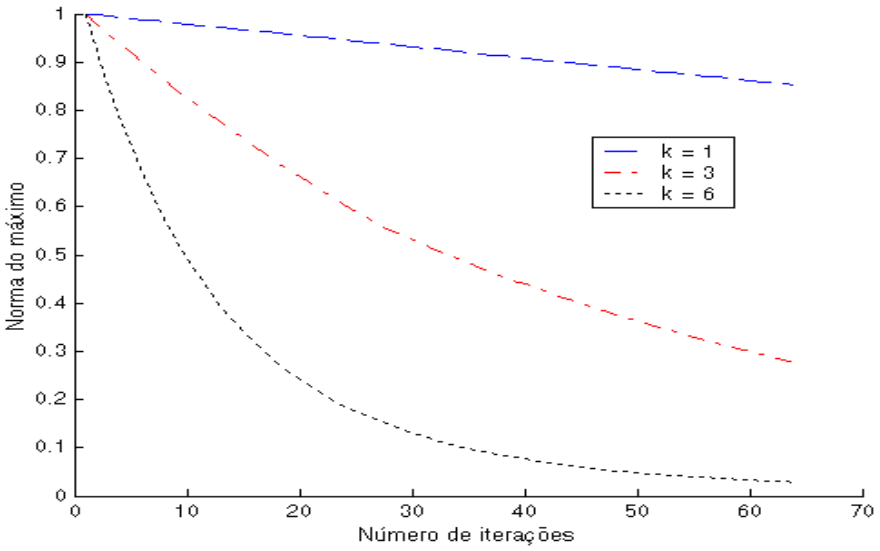


Fonte: o autor, 2013.

Foi utilizado o método Gauss-Seidel com 64 iterações para testar o procedimento descrito acima, numa malha com 64 elementos, utilizando como estimativa inicial os modos de Fourier $\mathbf{v}_1$, $\mathbf{v}_3$ e $\mathbf{v}_6$. Os resultados da norma máxima do erro são mostrados na figura 2.2, os quais são semelhantes com os obtidos por Briggs et al. (2000).

A figura 2.2 mostra que as primeiras iterações promovem uma rápida redução do erro quando este é oscilatório (k = 6), redução que se torna lenta à medida que o número de iterações aumenta.

Figura 2.2: Norma do vetor erro para as 64 primeiras iterações do método GS



Fonte: o autor, 2013.

2.3 PRINCÍPIOS DA TÉCNICA MULTIGRID

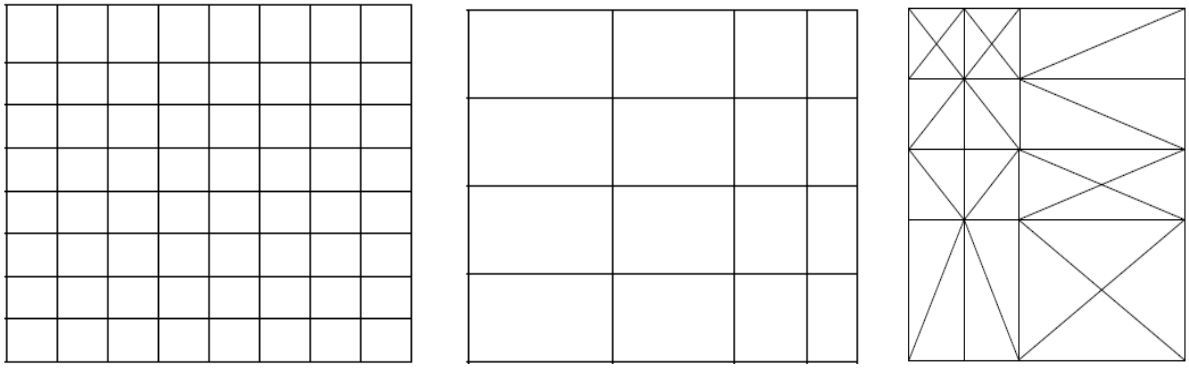
O objetivo desta seção é apresentar a técnica *multigrid*: suas origens, fundamentação teórica, campos de aplicações, mostrando suas vantagens e limitações, sem nenhuma pretensão de revisar toda a literatura sobre o tema. Esta técnica tem se revelado de ótima eficiência, particularmente em problemas que envolvem grandes sistemas de equações. A seguir uma breve descrição da técnica *multigrid*.

Como já citado anteriormente no início deste capítulo, a técnica *multigrid* surgiu como alternativa para se reduzir o tempo de processamento na resolução numérica de equações diferenciais e integrais. Especificamente na área de Elasticidade várias aplicações são encontradas, (JOUGLARD; COUTINHO, 1998; XU, 1999; YOO, 2003; XIAO et al., 2006). Esta técnica pode ser empregada usando quaisquer métodos numéricos tais como elementos finitos, diferenças finitas, volumes finitos e elementos de contorno, entre outros. Trata-se de uma técnica iterativa, que pode ser utilizada tanto em

problemas lineares como não lineares. Alguns estudos mostram que a técnica *multigrid* reduz significativamente o tempo de processamento quando comparado com outros métodos (ADAMS, 2000; MORO, 2004).

A técnica *multigrid* consiste em se trabalhar com várias malhas. O argumento básico é que os métodos iterativos clássicos suavizam o erro tornando muito lenta sua remoção. Assim um erro suave obtido após algumas iterações numa malha refinada é transferido para uma malha mais grosseira, tornando-o oscilatório. Este erro oscilatório é facilmente removido com poucas iterações. No *multigrid* geométrico uma malha pode ser uniforme ou não uniforme, estruturada ou não estruturada. Uma malha é uniforme quando todos os elementos possuem a mesma dimensão (BATHE, 1996). Uma malha é estruturada quando seus elementos podem ser projetados num sistema de eixos coordenados. A figura 2.3, mostra os tipos de malhas para o caso bidimensional.

Figura 2.3: Exemplos de malhas (a) malha estruturada e uniforme, (b) estruturada e não uniforme, (c) não estruturada e não uniforme



(a) malha estruturada e uniforme

(b) estruturada e não-uniforme

(c) não estruturada e não uniforme

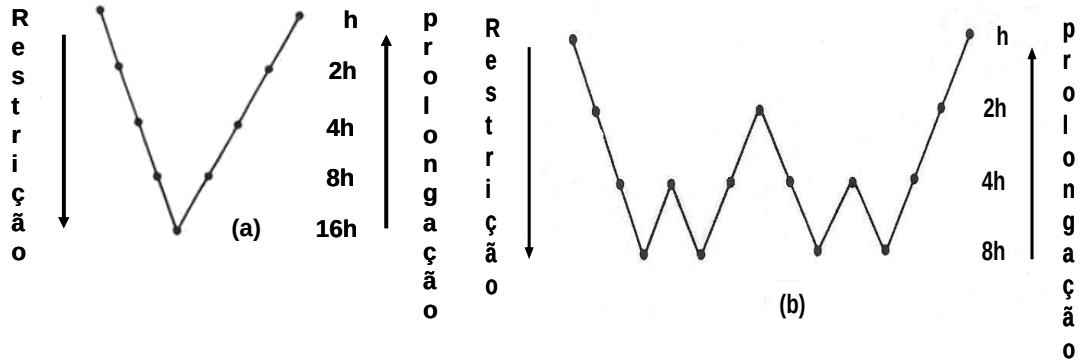
Fonte: PINTO, 2006.

Cada sequência de restrição e prolongação é chamada de ciclo. Os mais utilizados são em forma de V e W. A figura 2.4, baseada em Briggs et al. (2000), mostra os ciclos V e W, com fator de 1:2.

A técnica *multigrid* aceita os seguintes tipos de esquemas, o esquema de correção (*Correction Scheme*, CS), o esquema de aproximação completa (*Full Approximation Scheme*, FAS) e o esquema FMG, *multigrid* completo, (*Full Multigrid*). No esquema CS o problema é resolvido (iterativamente)

apenas na malha mais refinada. Nos níveis mais grosseiros apenas o resíduo e o erro são calculados, (o erro também é calculado iterativamente), sendo que o resíduo obtido num nível é transferido para o próximo nível mais grosseiro, até atingir o nível mais grosseiro possível, caso o ciclo V esteja sendo utilizado, quando então a equação residual é calculada diretamente. Na fase de prolongação apenas os erros são atualizados, utilizando-se os mesmos resíduos obtidos na fase de restrição. Já no esquema FAS, o problema é resolvido em todos os níveis de malha. No esquema FMG uma solução inicial é calculada numa malha grosseira, para ser melhorada nas malhas mais refinadas (BRIGGS et al., 2000). O esquema CS é usualmente utilizado em problemas lineares, enquanto o esquema FAS é aplicado em problemas não lineares (OLIVEIRA et al., 2006).

Figura 2.4: Ciclos *Multigrid*, (a) Ciclo-V, (b) Ciclo-W



Fonte: Adaptado de BRIGGS et al., 2000.

No ciclo V, os cálculos são iniciados na malha mais refinada de nível *h*, passando para a malha vizinha, e assim sucessivamente até que se atinja a malha mais grosseira possível ou desejada. Em cada nível de malha pode ocorrer um número diferente de iterações. Neste caso as informações são transferidas entre as malhas por meio do operador de restrição. Em seguida o processo se inverte, agora indo no sentido da malha fina vizinha, e assim sucessivamente até que se retorne a malha mais refinada de todas ou desejada. Nesta fase, o responsável pela propagação das informações entre as malhas é o operador de prolongação.

O operador que transfere informação de um nível de malha refinada para um nível mais grosseiro é denominado de operador de restrição, cuja notação (para o fator de restrição de 1:2) é I_h^{2h}, onde o subscrito indica o nível de origem (h) e o sobrescrito indica o nível de destino ($2h$). O objetivo da restrição é transformar um erro suave ou de baixa frequência, difícil de ser eliminado, num erro oscilatório ou de alta frequência, cuja remoção acontece em poucas iterações. É importante registrar que a restrição altera a amplitude do erro, não o modo do erro, desde que o erro a ser transferido seja suave. A notação para o operador que transfere informação de uma malha grosseira para uma malha imediatamente mais refinada, operador de prolongação, é I_{2h}^h (para o fator de prolongação de 1:2). Tanto o operador de prolongação (I_{2h}^h) quanto o operador de restrição (I_h^{2h}) podem ser representados matricialmente (BRIGGS et al., 2000).

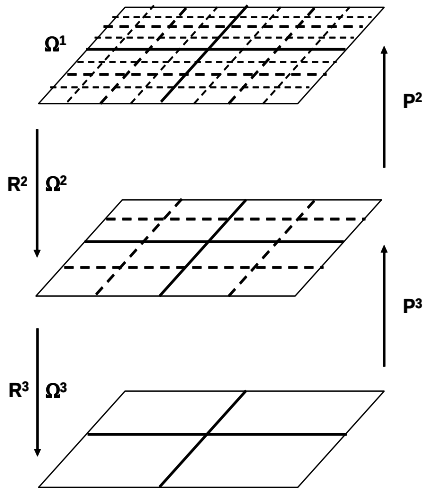
Uma vez definidos os operadores de restrição e prolongação, falta agora definir a matriz A^{2h}, no nível imediatamente mais grosseira $2h$. Como a matriz A^h foi obtida pela discretização com espaçamento regular h entre os pontos, e o conjunto destes pontos é denominado malha, cuja notação é Ω^h, então se o espaçamento entre os pontos fosse de $2h$, o conjunto destes pontos seria Ω^{2h}. Consequentemente, a matriz na malha imediatamente mais grosseira, $2h$ seria obtida pela discretização em Ω^{2h}, e poderia assumir a notação A^{2h}. Outra maneira de se obter a matriz A^{2h} é através da expressão que utiliza propriedades variacionais, (BRIGGS et al., 2000)

$$A^{2h} = I_h^{2h} A^h I_{2h}^h \tag{2.31}$$

A obtenção da matriz A^{2h} em Ω^{2h} por meio da equação (2.31) é conhecida como princípio de Galerkin. Como as três matrizes do lado direito da condição de Galerkin são esparsas, ou seja, possuem muitos elementos nulos, então A^{2h} pode ser calculada explicitamente sem muito esforço computacional. Wesseling (1992) demonstrou o princípio de Galerkin, apresentando algoritmos para sua implementação, bem como traçando comparativos entre as duas maneiras de se obter a matriz A^{2h}.

A figura 2.5 mostra o deslocamento entre malhas em um espaço bidimensional, onde se observa que na fase de restrição (R) o sentido ocorre da malha mais refinada para a mais grosseira. Já na fase de prolongação ou interpolação (P) o sentido acontece da malha mais grosseira para a malha mais refinada (BRIGGS et al., 2000; TROTTENBERG et al., 2001).

Figura 2.5: Transferência entre malhas num espaço bidimensional



Fonte: Adaptado de BRIGGS et al., 2000; TROTTENBERG et al., 2001.

A função do operador de restrição (I_h^{2h}) é transferir um vetor $\mathbf{v}^{h}$ obtido num nível de malha *h*, no vetor $\mathbf{v}^{2h}$ para um nível de malha imediatamente mais grosseiro, 2*h*, ou seja, $I_h^{2h} v^h = v^{2h}$. O operador de prolongação (I_{2h}^{h}) transfere o vetor $\mathbf{v}^{2h}$ no vetor $\mathbf{v}^{h}$, ou seja, $I_{2h}^{h} \mathbf{v}^{2h} = \mathbf{v}^{h}$. Tanto a restrição quanto a prolongação podem ser obtidas de várias maneiras (BRIGGS et al., 2000; TROTTENBERG et al. , 2001). A operação de restrição mais simples é a injeção, definida conforme a equação (2.32) para o caso unidimensional e equação (2.33) para o caso bidimensional

$$v_j^{2h} = v_{2j}^{h},\ 1 \le j \le \frac{N}{2} - 1 \tag{2.32}$$

$$v_{i,j}^{2h} = v_{2i,2j}^{h},\ 1 \le i, j \le \frac{N}{2} - 1 \tag{2.33}$$

na qual N é o número de elementos na malha mais refinada.

Outro operador de restrição bastante utilizado é o operador de restrição com ponderação completa, cuja operação para o caso unidimensional aparece na equação (2.34) e o caso bidimensional é definido pela equação (2.35)

$$v_j^{2h} = \frac{1}{4}\left(v_{2j-1}^{h} + 2v_{2j}^{h} + v_{2j+1}^{h}\right),\ 1 \le j \le \frac{N}{2} - 1 \tag{2.34}$$

$$v_{i,j}^{2h} = \frac{1}{16}[v_{2i-1,2j-1}^{h} + v_{2i-1,2j+1}^{h} + v_{2i+1,2j-1}^{h} + v_{2i+1,2j+1}^{h} + 2(v_{2i,2j-1}^{h} + v_{2i,2j+1}^{h} + v_{2i-1,2j}^{h} + v_{2i+1,2j}^{h}) + 4v_{2i,2j}^{h}] \tag{2.35}$$

para , $1 \le i, j \le \frac{N}{2} - 1$.

A operação de prolongação mais comum é a de interpolação linear expressa através da equação (2.36) para o caso unidimensional e pela equação (2.37) para o caso bidimensional. Outros operadores de prolongação podem ser obtidos nos trabalhos de Trottenberg et al. (2001) e Wesseling (1992).

$$v_{2j}^{h} = v_j^{2h},\ v_{2j+1}^{h} = \frac{1}{2}\left(v_j^{2h} + v_{j+1}^{2h}\right),\ 1 \le j \le \frac{N}{2} - 1 \tag{2.36}$$

$$v_{2i,2j}^{h} = v_{i,j}^{2h},\quad v_{2i+1,2j}^{h} = \frac{1}{2}\left(v_{i,j}^{2h} + v_{i+1,j}^{2h}\right),$$

$$v_{2i,2j+1}^{h} = \frac{1}{2}\left(v_{i,j}^{2h} + v_{i,j+1}^{2h}\right), \tag{2.37}$$

$$v_{2i+1,2j+1}^{h} = \frac{1}{4}\left(v_{i,j}^{2h} + v_{i+1,j}^{2h} + v_{i,j+1}^{2h} + v_{i+1,j+1}^{2h}\right),\ 1 \le \mathrm{i, j} \le \frac{N}{2} - 1$$

2.4 CONCEITOS DO MULTIGRID

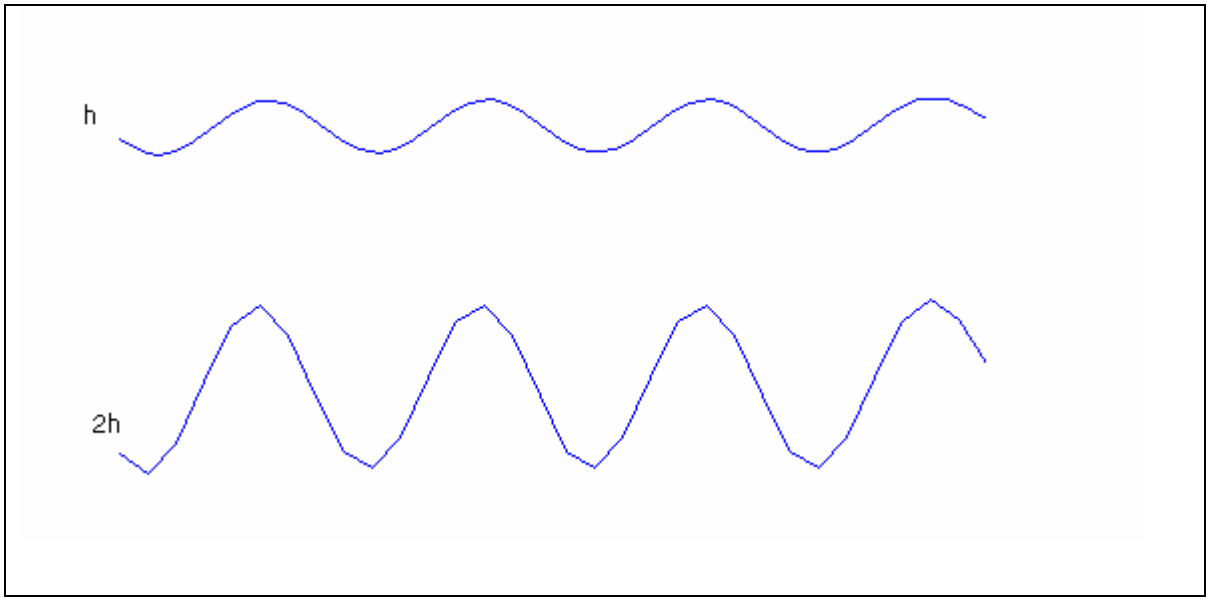
Observando a figura 2.2 é possível verificar que no caso efetuado, as primeiras iterações do método iterativo estacionário proporcionaram uma rápida redução do erro. Essa redução se tornou mais lenta à medida que as

iterações aumentaram. Verifica-se também na figura 2.2 que para o mesmo caso, a redução do erro é mais acentuada para estimativas iniciais **v** com alta frequência, mais especificamente $k = 6$. Já para erros com menor variação, $-\mathbf{v}_1$ e $-\mathbf{v}_3$, a redução do erro foi menos significativa, tornando-se comparativamente muito lenta para o erro $-\mathbf{v}_1$.

Como já visto anteriormente, o custo computacional dos métodos iterativos clássicos, como o Jacobi e Gauss-Seidel, é da ordem de $O(n^2)$, sendo n o número de incógnitas. A quantidade de incógnitas está relacionada com o tamanho da discretização h. Para uma malha unidimensional uniforme, tem-se que $h = 1/N$ na qual N é o número de pontos na discretização do domínio na malha mais refinada. Isto leva a uma suposição imediata: se o tamanho de h for aumentado, gerando uma discretização mais grosseira o custo computacional diminuirá. Para apreciar numericamente esta suposição, aplica-se 10 iterações na discretização com 64 elementos do mesmo sistema utilizado na seção anterior para testar a convergência dos modos de Fourier, com $k = 8$. Como $k = 8 < 64/2$, então o modo de Fourier $\mathbf{v}_8$ para este caso é considerado de baixa frequência, e consequentemente o erro $\mathbf{e}^h = -\mathbf{v}_8$ também. O erro $\mathbf{e}^h$ após 10 iterações é restringido por meio da equação (2.32), transformando-se no vetor $\mathbf{e}^{2h}$. A figura 2.6 apresenta os resultados. O gráfico superior, mais suave apresenta o erro $\mathbf{e}^h$ para a malha com 64 elementos. O gráfico inferior, mais oscilatório, mostra o mesmo erro restringido por meio de injeção para uma malha com 32 elementos no mesmo domínio. Lembrando que $I_h^{2h}\mathbf{e}^h = \mathbf{e}^{2h}$.

A figura 2.6 mostra que um erro suave numa malha com elemento de dimensão h torna-se oscilatório se for restringido para uma malha com elemento de dimensão $2h$. Isto acontece porque a restrição não altera o modo do erro. Nota-se que ambos os gráficos da figura especificada possuem oito ondas. A diferença é que no gráfico inferior as oito ondas estão confinadas entre zero e 32, ao contrário do gráfico superior que está distribuído entre 0 e 64.

Figura 2.6: Vetor erro $\mathbf{e} \in \Omega^h$ superior ($N = 64$), com $k = 8$, projetado em Ω^{2h} inferior, ($N = 32$), $h = 1/N$



Fonte: O autor, 2013.

O objetivo agora é usar conceitos matemáticos mais rigorosos para justificar e generalizar os resultados experimentais, o que será descrito no restante desta seção.

Primeiramente aborda-se a questão do número de onda k, com $1 \leq k \leq N - 1$ do modo de erro de Fourier $\mathbf{v}_k$. A questão proposta é se o modo $\mathbf{v}_k$ continua sendo sempre o mesmo após ser projetado de uma malha refinada Ω^h para uma malha mais grossa Ω^{2h}. Para responder a esta questão considera-se o k-ésimo modo na malha refinada nos pontos pares desta malha. As componentes dos modos suaves na malha Ω^h, para $1 \leq k \leq N/2$ podem ser escritas como

$$w^h_{k,2j} = sen\left(\frac{2jk\pi}{N}\right) = sen\left(\frac{jk\pi}{N/2}\right) = w^{2h}_{k,j} \text{ , para } 1 \leq k \leq N/2 \qquad (2.38)$$

A equação (2.38) mostra que o k-ésimo modo em Ω^h torna-se o k-ésimo modo em Ω^{2h}. Ou seja, o número de onda k em Ω^{2h} continua sendo o mesmo k de Ω^h, apenas tornando-se mais oscilatório. Importante frisar que esta análise somente é válida se o vetor transferido para a malha grossa for suave (BRIGGS et al., 2000).

Faz-se necessário também demonstrar matematicamente a convergência dos métodos iterativos clássicos. Para isso reescreve-se a equação (2.25)

$$\mathbf{v}^{k+1} = S\mathbf{v}^{k} + D^{-1}\mathbf{f} \qquad k = 1, 2, \ldots. \tag{2.39}$$

Como $\mathbf{r}^{k} = \mathbf{f} - A\mathbf{v}^{k}$ e $L + U = D - A$, a equação (2.39) pode ser reescrita em função do vetor residual $\mathbf{r}^{k}$

$$\mathbf{v}^{k+1} = D^{-1}(D - A)\mathbf{v}^{k} + D^{-1}(\mathbf{r} - A\mathbf{v}^{k}).$$

$$\text{Logo, } \mathbf{v}^{k+1} = \mathbf{v}^{k} + S_J\mathbf{r}^{k} \qquad k = 1, 2, \ldots.$$

O método de Gauss-Seidel também pode ser expresso em função do vetor residual $\mathbf{r}$. Neste caso bastaria substituir a expressão S_J na equação acima pela matriz S_G, que num contexto genérico assume a notação S. A literatura utiliza o termo relaxação como sinônimo de suavização (PEREIRA, 2007) o que também será feito nesta tese daqui em diante. Assim, a equação modificada aparece na expressão

$$\mathbf{v}^{k+1} = \mathbf{v}^{k} + S\mathbf{r}^{k} \qquad k = 1, 2, \ldots. \tag{2.40}$$

Teorema 2.2: O método iterativo $\mathbf{x}^{k+1} = S\mathbf{x}^{k} + \mathbf{c}$ é convergente para qualquer aproximação inicial $\mathbf{x}^{(1)}$ se, e somente se, a matriz $S^{k} \to 0$ quando $k \to \infty$.

Prova:

$$\mathbf{x} = S\mathbf{x} + \mathbf{c}. \tag{1}$$

e

$$\mathbf{x}^{k+1} = S\mathbf{x}^{k} + \mathbf{c} \tag{2}$$

Fazendo (1) – (2):

$$\mathbf{x} - \mathbf{x}^{k+1} = S(\mathbf{x} - \mathbf{x}^{k}) \tag{3}$$

Como a expressão (3) é válida para qualquer valor de k, pode-se escrever

$$\mathbf{x} - \mathbf{x}^{k} = S(\mathbf{x} - \mathbf{x}^{(k-1)}) \tag{4}$$

Substituindo (4) em (3):

$$\mathbf{x} - \mathbf{x}^{k+1} = S\,S(\mathbf{x} - \mathbf{x}^{(k-1)}) = S^{2}(\mathbf{x} - \mathbf{x}^{(k-1)}) \tag{5}$$

Como a expressão (5) também é válida para qualquer k, pode-se reescrevê-la:

$$\mathbf{x} - \mathbf{x}^{k} = S^2(\mathbf{x} - \mathbf{x}^{(k-2)}) \qquad (6)$$

Substituindo (6) em (3),

$$\mathbf{x} - \mathbf{x}^{k+1} = S\,S^2(\mathbf{x} - \mathbf{x}^{(k-2)}) = S^3(\mathbf{x} - \mathbf{x}^{(k-2)}) \qquad (7)$$

Continuando este processo k vezes, chega-se na expressão

$$\mathbf{x} - \mathbf{x}^{k+1} = S^k(\mathbf{x} - \mathbf{x}^{(1)}) \qquad (2.41)$$

A expressão (2.41) mostra que $\{\mathbf{x}^{(k)}\}$ converge para a solução $\mathbf{x}$ para qualquer escolha arbitrária de $\mathbf{x}^{(1)}$ se e somente se $S^k \to 0$ quando $k \to \infty$. Isto vale, naturalmente, quando $\mathbf{x}^{(1)}$ não coincide com a solução exata $\mathbf{x}$.

Vale destacar aqui as **Afirmações 2.1**, mais especificamente a (1) e a (4).

Uma vez demonstradas as condições para a convergência dos métodos iterativos estacionários, a equação (2.34) pode ser reescrita em termos da solução exata do problema

$$\mathbf{u} = S\mathbf{u} + \mathbf{g}. \qquad (2.42)$$

na qual

$$\mathbf{g} = \begin{cases} D^{-1}\boldsymbol{f}, \textit{ para Jacobi} \\ \\ (D - L)^{-1}\boldsymbol{f}, \textit{ para Gauss-Seidel} \end{cases}$$

Subtraindo (2.39) de (2.42), obtém-se

$$\mathbf{e}^{k+1} = S\mathbf{e}^{k}. \qquad (2.43)$$

A equação (2.43) aplicada recursivamente resulta em

$$\mathbf{e}^{k+1} = S^{(k)}\mathbf{e}^{0}, \qquad (2.44)$$

na qual o índice (k) representa uma potência.

Da equação (2.44) e do teorema 2.1, se conclui que a convergência dos métodos iterativos estacionários é determinada pelos autovalores da matriz de iteração *S*.

O próximo passo é estabelecer uma relação entre os autovalores da matriz *S* e os autovalores da matriz *A* do problema original, equação (2.1). Para isso será usada a matriz de iteração do método Jacobi com sobrerrelaxação (ω), dada por Briggs et al. (2000).

$$S_\omega = (1 - \omega)I + \omega D^{-1}(L + U). \tag{2.45}$$

Como $L + U = D - A$, resulta

$$S_\omega = I - \omega D^{-1}A. \tag{2.46}$$

Avaliando a equação (2.46) para os dados da equação (2.29), obtém-se

$$S_\omega = I - \frac{\omega}{2}\begin{pmatrix} 2 & -1 & & & & & \\ -1 & 2 & -1 & & & & \\ & -1 & 2 & -1 & & & \\ & & . & . & . & & \\ & & & . & . & . & \\ & & & & . & . & -1 \\ & & & & & -1 & 2 \end{pmatrix} \tag{2.47}$$

Assim, os autovalores de S_ω estão relacionados aos da matriz *A* por

$$\lambda(S_\omega) = 1 - \frac{\omega}{2}\lambda(A). \tag{2.48}$$

A equação (2.48) mostra que os autovalores de S_ω podem ser obtidos calculando os autovalores de *A*. Já os autovalores de *A* são definidos por Briggs et al. (2000)

$$\lambda_k(\mathbf{A}) = 4sen^2(\frac{k\pi}{2N}), \text{ para } 1 \leq k \leq N-1. \tag{2.49}$$

Os autovetores de S_ω são os mesmos da matriz A, e podem ser estabelecidos pela expressão seguinte,

$$w_{k,j} = sen(\frac{jk\pi}{N}), \text{ para } 1 \leq k \leq N-1,\ 0 \leq j \leq N. \tag{2.50}$$

Observando as equações (2.49) e (2.50), percebe-se que os autovetores da matriz A são os modos de Fourier.

Pode-se também aplicar uma relaxação (ω) na matriz S do método Gauss-Seidel, que a passa a ter a expressão $S = \omega(D - \omega L)^{-1}$. Neste caso, o método é denominado SOR.

2.5 ALGORITMOS

Uma vez que os elementos necessários para o *multigrid* geométrico foram estabelecidos, pode-se agora apresentar os algoritmos básicos para sua implementação. Em todos os algoritmos apresentados nesta seção considera-se a razão de engrossamento de malha igual a 2. Em outras palavras, o espaçamento entre os pontos em Ω^{2h} é o dobro do espaçamento em Ω^h. Isto é uma prática comum na literatura, e, segundo Briggs, outras razões não trazem vantagens significativas (BRIGGS et al., 2000). Pesquisas posteriores demonstraram que, ao menos para os casos pesquisados, a razão de engrossamento 2 perde a hegemonia. Moro concluiu que para problemas lineares unidimensionais a relação entre malhas de 1:4 é preferível em comparação com a relação de 1:2 (MORO, 2004). Pinto recomenda a razão entre malhas de 1:3 para o esquema FAS (PINTO, 2006).

A seguir um algoritmo com dois níveis de malha para o *multigrid* geométrico é apresentado.

Algoritmo 2.1. Esquema de correção com dois níveis de malha

1. Aplique υ_1 passos de relaxação em $A^h\mathbf{u}^h = \mathbf{f}^h$ com estimativa inicial $\mathbf{v}^h$
2. Calcule $\mathbf{r}^h = \mathbf{f}^h - A\mathbf{v}^h$
3. Calcule $\mathbf{r}^{2h} = \mathrm{I}_h^{2h}\mathbf{r}^h$
4. Resolva $A^{2h}\mathbf{e}^{2h} = \mathbf{r}^{2h}$, em Ω^{2h}
5. Calcule $\mathbf{e}^h = \mathrm{I}_{2h}^h\mathbf{e}^{2h}$
6. Faça $\mathbf{v}^h \leftarrow \mathbf{v}^h + \mathbf{e}^h$
7. Aplique υ_2 passos de relaxação em $A^h\mathbf{u}^h = \mathbf{f}^h$ com estimativa inicial $\mathbf{v}^h$

Algumas observações a respeito do esquema de correção apresentado no Algoritmo 2.1 (BRIGGS et al., 2000) podem se fazer necessárias para melhor entendimento:

- Teoricamente os números de iterações υ_1 e υ_2 devem ser suficientes para tornar o erro suave. Como na prática o vetor erro é tão desconhecido como a própria solução, é usual fixar υ_1 e υ_2 em 1, 2 ou 3.
- A matriz A^{2h} é a representação da matriz A^h em Ω^{2h}.
- Na prática, o item (4) do algoritmo acima encerra algumas dificuldades, como por exemplo, resolver a equação quando esta for muito grande. Isto sugere uma melhoria no algoritmo, no sentido de se criar malhas mais grosseiras que a Ω^{2h}. Usualmente são criados tantos níveis quanto forem possíveis.

Apresenta-se a seguir um algoritmo, na forma recursiva, para o *multigrid* geométrico.

Algoritmo 2.2: Esquema de correção (CS) ciclo V (forma recursiva)

1. Aplique υ_1 passos de relaxação em $A^h\mathbf{u}^h = \mathbf{f}^h$ com estimativa inicial $\mathbf{v}^h$
2. Calcule o resíduo $\mathbf{r}^h = \mathbf{f}^h - A^h\mathbf{v}^h$
3. Calcule $\mathbf{r}^H \leftarrow \mathrm{I}_h^{2h}\mathbf{r}^h$

Faça até a malha mais grosseira possível

4. Aplique υ_2 passos de relaxação em $A^H\ \mathbf{e}^H = \mathbf{r}^H$ com estimativa inicial $\mathbf{e}^H = \mathbf{0}$
5. Calcule $\mathbf{r}^H = \mathbf{r}^H - A^H\ \mathbf{e}^H$
6. Calcule $\mathbf{r}^{2H} \leftarrow \mathrm{I}_H^{2H}\mathbf{r}^H$
7. Resolva o sistema $\mathbf{e}^{Lh} = (A^{Lh})^{-1}\ \mathbf{r}^{Lh}$
8. Interpole $\mathbf{e}^H \leftarrow \mathrm{I}_{2H}^H\mathbf{e}^{Lh}$

Faça até a malha mais refinada

9. Aplique υ_2 passos de relaxação em $A^{2H}\ \mathbf{e}^{2H} = \mathbf{r}^{2H}$ com estimativa inicial $\mathbf{e}^{2H}$
10. Atualize o erro $\mathbf{e}^{2H} \leftarrow \mathbf{e}^{2H} + \mathrm{I}_{2h}^h\mathbf{e}^{Lh}$
11. Atualize a solução aproximada $\mathbf{v}^h \leftarrow \mathbf{v}^h + \mathbf{e}^h$
12. Retornar ao passo 1.

O termo *Lh* indica a malha mais grosseira (BRIGGS et al., 2000).

3 O MÉTODO MULTIGRID ALGÉBRICO (AMG)

O *multigrid* algébrico (AMG) torna-se útil na resolução de problemas em que é impossível determinar uma sequência de malhas ou então quando as malhas, mesmo existindo, assumem geometrias e ou refinos diferentes ao longo do domínio. Outra situação em que o AMG torna-se vantajoso em comparação com o *multigrid* geométrico é quando há uma descontinuidade no domínio. Há ainda os problemas puramente discretos que não possuem geometria como aqueles que aparecem em geodésia e em problemas econômicos (MCCORMICK, 1987).

Os pioneiros na introdução do *multigrid* algébrico foram Brandt, Ruge e McCormick em 1980 (TROTTENBERG at. al, 2001). A ideia básica do AMG é que as conectividades dos nós são inteiramente determinadas pela matriz A da equação (2.1). Inicialmente o AMG foi teorizado para o caso específico de matriz cujos elementos da diagonal são positivos, $a_{ii} > 0$, e os elementos fora da diagonal são negativos, $a_{ij} < 0$, para $i \neq j$. Atualmente sabe-se que esta é uma condição suficiente, porém não necessária para o AMG.

As conexões dentro da malha são determinadas pelo grafo $G(V, E)$ não direcionado adjacentes da matriz A, onde os elementos a_{ij} de A formam os vértices V do grafo, e E são os segmentos não orientados entre dois vértices para a_{ij} e a_{ji} diferentes de zero. Para outras informações sobre grafos, ver Kreyszig (1999).

Mesmo não havendo uma malha estruturada no *multigrid* algébrico, as notações do *multigrid* geométrico para dois níveis de malhas diferentes, h e $2h$ são mantidas com o mesmo significado, ou seja, h indica o nível mais refinado, $2h$ o nível imediatamente mais grosseiro.

A principal diferença conceitual entre o *multigrid* geométrico e o AMG é que no GMG o parâmetro de engrossamento para qualquer nível de malha é fixado. Ou seja, é possível definir a priori tanto os níveis de malha existentes no ciclo, quanto o número de nós internos terá a malha mais grosseira. Isto deve ser feito de modo que os operadores entre malhas transmitam apuradamente as informações entre dois níveis de malha.

No AMG acontece o contrário, primeiro um esquema de relaxação que permita determinar a natureza do erro suave é fixado. Só então um esquema de engrossamento de malha é selecionado. O termo malha no contexto do AMG é apenas uma analogia com o GMG, pois no *multigrid* algébrico o que se tem é uma sequência de matrizes. Outro aspecto que diferencia o GMG do AMG é que no último não há uma relação fixa entre a quantidade de elementos da matriz A do sistema entre um nível e outro imediatamente mais grosseiro.

Devido à impossibilidade de se estabelecer a priori a razão entre os pontos da malha refinada para uma malha imediatamente mais grosseira, assim como é improvável que esta razão permaneça constante para todos os níveis de malhas, torna-se necessário encontrar outros critérios para estimar os custos computacionais do AMG, sendo eles:

- Complexidade de malha: é o número total de pontos existentes em todas as malhas, dividido pelo número de pontos da malha mais refinada.

- Complexidade do operador: é a quantidade de elementos não nulos de todas as matrizes A^{kh}, dividido pela quantidade de elementos não nulos da matriz da malha mais refinada A^{h}.

No AMG a malha mais grosseira tem a mesma função que no *multigrid* geométrico, ou seja, criar as condições para que a relaxação volte a ser efetiva.

Dois conceitos são fundamentais no AMG. O primeiro é o conceito de suavidade algébrica. Tal suavidade é alcançada quando o tamanho do erro numa determinada iteração ($\mathbf{e}^{i+1}$) não é significativamente menor que a iteração anterior ($\mathbf{e}^{i}$), para um critério de avaliação de erro selecionado. Isto significa que o erro suave no contexto do AMG é aquele que possui uma lenta convergência. Na verdade a palavra “suave”, *smooth* em inglês, deveria ser substituída pelo termo “lento para convergir”, do inglês *slow-to-converge*, mas o próprio autor que afirma isto alega razões históricas para manter o termo “suave” (TROTTENBERG et al., 2001).

Importante salientar que a suavidade algébrica não implica necessariamente em suavidade geométrica, ou seja, um erro pode ser algebricamente suave e sua representação gráfica bastante oscilatória (BRIGGS et al., 2000). Isto significa que um erro num ponto pode ser muito diferente dos erros nos pontos vizinhos (BREZINA et al., 2000). Em geral, no AMG, erro suave corresponde aos autovetores da matriz *A* cujos autovalores associados são pequenos (FALGOUT, 2006).

Uma vantagem do AMG é que o engrossamento ocorre apenas nas direções em que a relaxação suaviza o erro. Por outro lado, o AMG não supera em eficiência o *multigrid* geométrico em problemas em que é possível determinar uma sequência de malhas, uma vez que no AMG há necessidade de uma etapa a mais, denominada fase de preparação que naturalmente tem um custo computacional adicional (TROTTENBERG et al., 2001).

O segundo conceito fundamental no AMG é o de forte dependência ou de forte influência. Para desenvolver tal conceito admite-se que se o coeficiente a_{ij} que multiplica o componente u_j na equação (2.1) é grande em relação aos outros coeficientes da mesma linha, então uma pequena variação no valor de u_j tem mais efeito sobre a componente u_i que uma pequena mudança em outras variáveis desta mesma linha. Isto equivale a dizer que um ponto *i* depende de um ponto *j* se o coeficiente a_{ij} é o maior coeficiente fora da diagonal da equação considerada. Isto pode ser apresentado de outra maneira: se a variável u_i depende da variável u_j, então a variável u_j influencia a variável u_i.

3.1 ENGROSSAMENTO DE MALHA

No *multigrid* algébrico primeiro descobre-se é a natureza do erro suave. Só então, a partir dessa informação sobre a suavidade, é que a malha grosseira é selecionada. No AMG, assim como no *multigrid* geométrico, o erro suave é aquele que não é reduzido pela relaxação.

Para definir os pontos da malha Ω^{2h} a partir da malha Ω^h, voltamos ao conjunto {1, 2,..., *n*} que representa os índices das incógnitas na malha refinada do *multigrid* algébrico. A ideia é particionar o conjunto em dois

subconjuntos C e F de modo que $\{1, 2,..., n\} = C \cup F$, ($C$ união F). Os elementos de C são as variáveis da malha imediatamente mais grosseira Ω^{2h}, enquanto os elementos de F representam as variáveis que pertencem apenas à malha refinada Ω^h. Importante salientar que $C \cap F = \varnothing$ (BRIGGS, et al., 2000). Para uma malha refinada pode-se expressar a igualdade acima da forma $\Omega^h = C^h \cup F^h$. Assim pode-se definir a malha grosseira $\Omega^{2h} = C^h$. A seguir descreve-se o procedimento para iniciar o engrossamento de malha, ou seja, identificar o primeiro elemento do conjunto C.

Definição 3.1 Dado um valor limiar de corte $0 < \alpha \leq 1$, o ponto i depende fortemente do ponto j se,

$$|a_{ij}| \geq \alpha.\max|a_{ik}|,\ k \neq i \tag{3.1}$$

A definição 3.1 já contempla a possibilidade de se utilizar o AMG em problemas cuja matriz A não seja uma M-matriz (CHANG et al., 1996). Por esta definição é possível o ponto i depender fortemente do ponto j, mas o ponto j depender fracamente do ponto i, mesmo no caso da matriz A ser simétrica (FALGOUT, 2006).

Seja S_i o conjunto de pontos que influenciam fortemente o ponto i. Considere S_i^{T} como sendo o conjunto dos pontos que dependem fortemente do ponto i. Há várias maneiras de se escolher o primeiro ponto do conjunto C, inclusive por meio de um critério randômico. Uma das maneiras mais simples, no entanto, é estabelecer uma medida denominada de λ_i que é a quantidade de elementos que contém o conjunto S_i^{T}. Seleciona-se então um ponto com máximo valor λ_i para o primeiro ponto em C.

O processo de escolha do próximo elemento de C é repetido escolhendo-se um novo ponto i com máximo λ_i. Os demais pontos de S_i^{T} são alocados no conjunto F. O processo continua até que todos os pontos estejam devidamente identificados como pertencentes a C ou F.

Para cada ponto i da malha refinada, define-se uma vizinhança de i, N_i, que possui todos os pontos j diferentes de i tal que a_{ij} seja diferente de zero. Estes pontos podem ser divididos em três categorias:

- Conjunto dos pontos da malha grosseira, vizinhos de i que o influenciam fortemente, denominado C_i, $C_i = C \cap N_i$;
- Conjunto dos pontos que influenciam fortemente o ponto i pertencente à malha refinada denominado D_i^s;
- Conjunto dos pontos em N_i que não influenciam fortemente i, denotado por D_i^w. Este conjunto pode conter tanto elementos da malha refinada quanto da malha grosseira e é denominado conjunto dos vizinhos fracamente conectados. Estes conjuntos aparecem na definição de fatores de ponderação de interpolação nas próximas seções.

Falta ainda apresentar os critérios heurísticos para seleção dos elementos do conjunto C.

- $C1$: Para todo $i \in F$ todo ponto $j \in S_i$ deve estar em $C_i = C \cap N_i$, ou depender fortemente de, no mínimo, um ponto em C_i.

- $C2$: C deve ser o máximo subconjunto escolhido de forma que não exista forte conexão entre nenhum de seus pontos.

Com os critérios definidos acima, o processo de particionamento de Ω^h em C e F pode ser finalmente formalizado. Isto é feito com a utilização do algoritmo 3.1, a seguir, no qual o símbolo $|S|$ denota o número de elementos do conjunto S. Como na prática nem sempre é possível assegurar que os dois critérios $C1$ e $C2$ sejam simultaneamente satisfeitos, o usual é garantir primeiramente que $C1$ seja satisfeito, valendo-se de $C2$ como guia para não produzir um conjunto C muito grande (TROTTENBERG et al., 2001). A figura 3.1 mostra este procedimento para uma discretização uniforme.

Algoritmo 3.1. Procedimento de seleção das incógnitas

Dada uma matriz A^h dimensão n x n, faça:

1. $C = \varnothing$, $F = \varnothing$, $U = \Omega^h$, $\lambda_i = |S_i^T|$ para $i = 1, \ldots, n$.
2. Enquanto $U \neq \varnothing$, faça:

i) para $i \in U$ com máximo λ_i

$C \cup i$, $U = U - i$,

ii) para todo $j \in S_i^T \cap U$

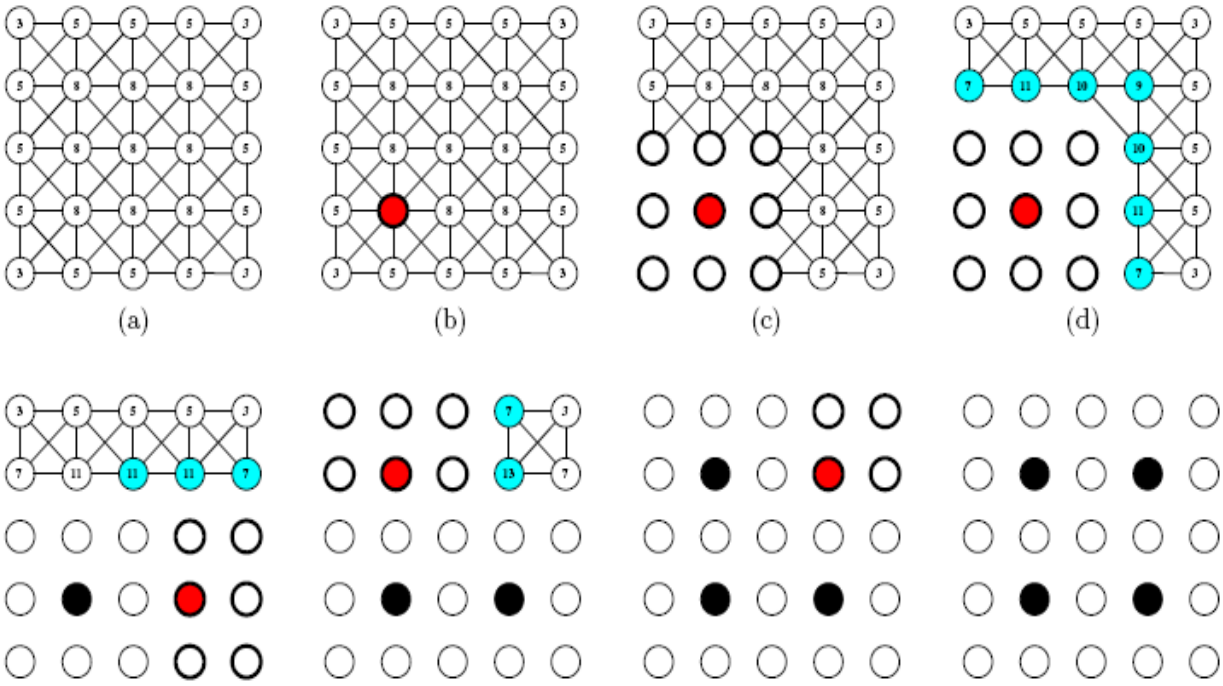
$F = F \cup j$ e $U = U - j$,

$\lambda \ell = \lambda \ell + 1$, para todo $\ell \in S_j \cap U$,

iii) para todo $j \in S_j \cap U$

$\lambda_j = \lambda_j - 1$.

Figura 3.1: Sequência do procedimento de seleção das incógnitas



(a) mostra o conjunto Ω^h, e os pesos que aparecem correspondem aos números das conexões fora da diagonal. (b) ponto com máximo peso é escolhido como primeiro C - ponto. (c) os vizinhos do novo ponto C são marcados como novos F – pontos. (d) para cada novo F – ponto, os pesos de seus vizinhos são incrementados por um. O algoritmo continua até que todos os pontos de Ω^h estejam marcados como C ou F – pontos.
Fonte: FALGOUT, 2006.

Dependendo do problema a ser resolvido torna-se interessante modificar as estratégicas de engrossamento da malha para que os resultados sejam

satisfatórios. A seguir, uma destas modificações, denominada engrossamento agressivo de malha, é apresentada.

O engrossamento de malha agressivo utiliza o conceito de fortes conexões de longo alcance e tem como objetivo reduzir a complexidade do operador. A ideia é estender a definição de forte conectividade para também incluir variáveis que não estão diretamente conectadas. Para isto adota-se o termo *p* que identifica caminho e o termo *l* que significa comprimento. Assim é definido que a variável *i* está fortemente unida a variável *j* se pelo menos *p* caminhos de comprimento menor ou igual a *l* existem tais que *i* está fortemente conectado a *j* ao longo de cada um destes caminhos. Mais detalhes em Trottenberg et al. (2001).

3.2 INTERPOLAÇÃO

A ideia básica para estabelecer um operador de interpolação no ambiente do AMG é a de que o erro suave varia lentamente na direção da forte conexão. Isto justifica o argumento de que a variável u_j da malha grosseira pode ser interpolada para a malha refinada u_i se *i* depende fortemente de *j*.

O operador de interpolação para o AMG é denotado pela mesma expressão e possui o mesmo significado do *multigrid* geométrico, ou seja, I_{2h}^{h}. Apesar de não haver uma malha propriamente dita, $2h$ continua significando o nível grosseiro e *h* o nível refinado. O *i*-ésimo componente de $\mathrm{I}_{2h}^{h}\mathbf{e}$ é dado pela equação (3.2), na qual ω_{ij} são denominados os pesos da interpolação.

$$(\mathrm{I}_{2h}^{h}\mathbf{e})_{\mathrm{i}} = \begin{cases} e_{\mathrm{i}} \text{ se } i \in C \\ \sum_{j \in Ci} \omega_{ij} e_j \quad \text{se } i \in F \end{cases} \tag{3.2}$$

Como a principal característica do erro suave é que o resíduo seja pequeno ($\mathbf{r} \approx \mathbf{0}$) pode-se escrever

$$A\mathbf{e} = \mathbf{r} \approx \mathbf{0}$$

$$\sum_{j_i} a_{ij} e_j \approx 0$$

$$\Rightarrow a_{ii} e_i + \sum_{j \in N_i} a_{ij} e_j \approx 0 \tag{3.3}$$

A partir da equação (3.3) é possível chegar-se na equação (3.4) conforme segue (BRIGGS et al.,2000),

$$\omega_{ij} = -\frac{a_{ij} + \sum_{m \in D_i^S} \left(\frac{a_{im}\, a_{mj}}{\sum_{k \in Ci} a_{mk}} \right)}{a_{ii} + \sum_{n \in D_i^w} a_{in}}, \tag{3.4}$$

na qual, para relembrar, D_i^S representa o conjunto de pontos que influenciam o ponto i mas que não estão em C_i, e D_i^w é o conjunto dos pontos em N_i que não influenciam fortemente o ponto i.

Pode-se agora resumir o procedimento do AMG.

- Seleciona-se uma malha grosseira cujas componentes suaves possam ser representadas apuradamente;
- Define-se um operador de interpolação capaz de transferir apuradamente componentes suaves de uma malha grosseira para uma malha refinada;
- Define-se um operador de restrição e uma versão da matriz A para a malha grosseira usando as propriedades variacionais.

Determinar o valor limiar de corte α definido neste capítulo não é simples. Desde o início, quando o AMG foi proposto, esta dificuldade é identificada (TROTTENBERG et al., 2001). Desde então muitos trabalhos foram publicados propondo otimizar o referido parâmetro para problemas específicos, entre eles Pereira et al. (2006a). Recentemente Suero et al. (2012) estudaram o fator de forte dependência do AMG para as equações bidimensionais de Poisson e Laplace, chegando em números diferentes

daqueles usualmente encontrados na literatura. Estudos estes que reforçam esta dificuldade inerente ao AMG, e que justifica investimento em nova abordagem para o *multigrid* algébrico, como o *multigrid* algébrico via *wavelets* WAMG que será apresentada no próximo capítulo.

4 O MÉTODO MULTIGRID ALGÉBRICO VIA WAVELETS

O objetivo deste capítulo é definir função *wavelet*, além de abordar a análise em multirresolução, justificando a ligação entre as *wavelets* e o *multigrid*.

O termo *wavelets* aparece na literatura em português brasileiro traduzido para ondaletas (MORETTIN, 1999), e significa literalmente pequenas ondas. Neste trabalho adotou-se a grafia em inglês. *Wavelets* são funções que gozam de algumas propriedades comuns. Inicialmente elas eram utilizadas em tratamento de sinais e compressão de dados, mas atualmente são aplicadas em muitas áreas, como será visto adiante.

Wavelets são funções $\psi_{a,b}(t)$ obtidas por meio de translações e dilatações a partir de uma única função $\psi(t)$ denominada *Wavelet* mãe, e sua forma geral é dada pela expressão

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right) \tag{4.1}$$

com $a > 0$, e $b \in \mathrm{R}$.

Na expressão (4.1), a e b são variáveis responsáveis pelas dilatações e translações, respectivamente. O parâmetro b indica uma translação de uma distância b no eixo t da função $\psi(t)$. Já o parâmetro a causa uma mudança de escala, aumentando (se $a > 1$) ou diminuindo (se $a < 1$) a *wavelet* formada pela função $\psi(t)$. O parâmetro a é conhecido como parâmetro de escala. A finalidade do termo $1/\sqrt{a}$ é preservar a norma da função no espaço $\mathrm{L}^2(\mathbb{R})$, e assegura que a energia de $\psi_{a,b}(t)$ seja independente de a e b, de modo que

$$\int_{-\infty}^{\infty} \left|\psi_{a,b}(t)\right|^2 dt = \int_{-\infty}^{\infty} \left|\psi(t)\right|^2 dt. \tag{4.2}$$

Dentre as características importantes das *wavelets* cita-se o fato delas formarem uma base para o espaço $\mathrm{L}^2(\mathbb{R})$, de serem capazes de decompor

outras funções em diferentes escalas; e serem localizadas na variável *t*, anulando-se fora de um determinado intervalo.

A decomposição de uma função *f(t)* com o uso de *wavelets* é conhecida como transformada de *wavelet*, que pode ser operada na forma discreta ou contínua e pode ser dada pelo produto interno definido por:

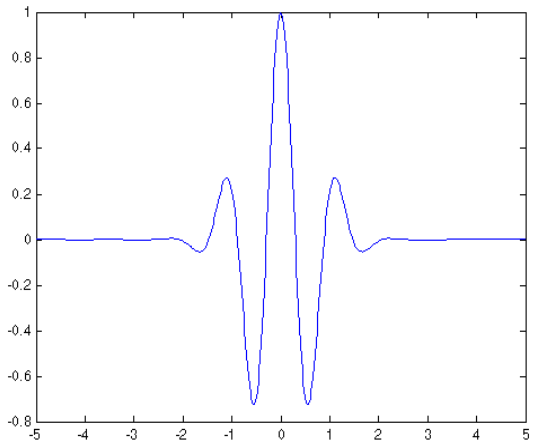
$$W(a,b) = \left\langle f(t), \psi_{a,b}(t) \right\rangle = \int_{-\infty}^{\infty} f(t)\psi_{a,b}(t)dt \tag{4.3}$$

Na transformada discreta *wavelet* (DWT) os parâmetros de translação *b* e de escala *a* da equação (4.1) são discretizados, e usualmente são expressos como $a = 2^j$ e $b = k2^j$, com $j, k \in Z$. Assim, uma família de *wavelets* discretas assume a forma geral

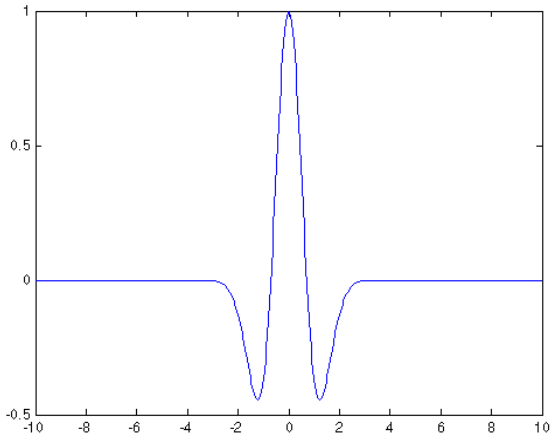
$$\psi_{ij}(x) = 2^{-j/2}\psi(2^{-j}x - k). \tag{4.4}$$

Observa-se que o parâmetro de translação depende da dilatação (CHUI, 1992; KAISER, 1994; MORETTIN, 1999). A figura 4.1 mostra alguns exemplos de funções *wavelets*.

Figura 4.1: Exemplos de *wavelets*



(a) Morlet



(b) Chapéu Mexicano

Fonte: O autor, 2013.

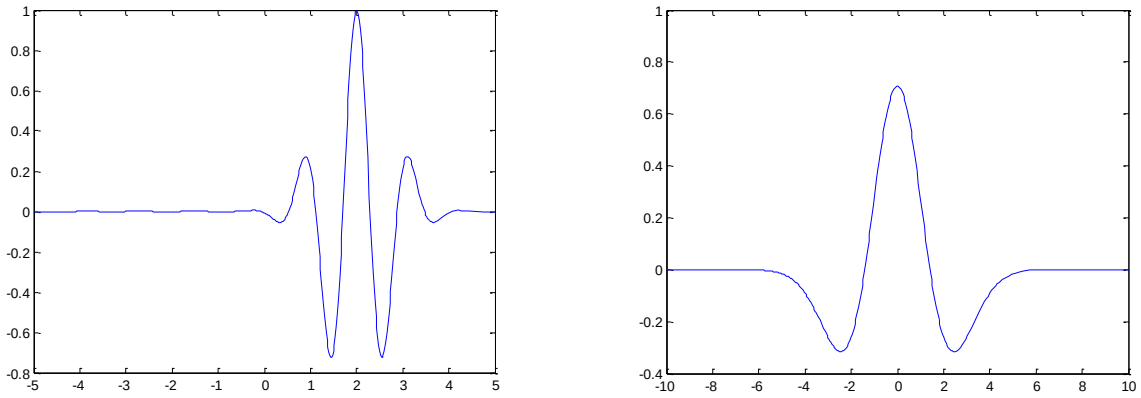
As funções de Morlet e Chapéu Mexicano são, respectivamente, dadas por

$$\psi(t) = e^{-t^2} \cos(\sqrt{(2/\log(2))\pi}\, t), \tag{4.5}$$

$$\psi(t) = (1-2t^2)e^{-t^2}. \tag{4.6}$$

A figura 4.2 mostra a função de Morlet para b = 2, ou seja, $\psi_{1,2}(t)$ e a função Chapéu Mexicano para a = 0,5, ou seja, $\psi_{0,5;0}(t)$ respectivamente. Ou seja, o parâmetro de escala a contrai ou expande verticalmente a figura para $a < 1$ ou $a > 1$ respectivamente. O parâmetro b translada a figuração ao longo do eixo horizontal.

Figura 4.2: (a) Morlet com translação b = 2 e (b) Chapéu Mexicano com fator de escala a = 0,5.



(a) (b)

Fonte: O autor, 2013.

Inicialmente aplicadas em processamento de sinais, as *wavelets* conquistaram a preferência de outras áreas como diversos ramos da física e da medicina, assim como na solução numérica de equações diferenciais e na compressão de dados (DAHMEN et al., 1997).

4.1 ANÁLISE EM MULTIRRESOLUÇÃO

Análise em multirresolução (AMR) é uma sequência V_j, $j \in Z$ de subespaços fechados aninhados, cuja união é densa no espaço das funções quadraticamente integráveis $L^2(\mathbb{R})$. Isto significa que a união ($\bigcup_{-\infty}^{\infty} V_j$) gera o mesmo subespaço das funções $L^2(\mathbb{R})$.

$$\ldots V_{j+1} \subset V_j \ldots \subset V_1 \subset V_0 \subset V_{-1} \ldots \tag{4.7}$$

O conceito de multirresolução foi desenvolvido por Mallat (1989) e consiste em escrever uma função $f \in L^2(\mathbb{R})$ como um limite de sucessivas aproximações, $f = \lim_{m \to \infty} P_m f$, na qual cada aproximação $P_m f$ corresponde a uma versão suavizada de f, com um grau de suavização da ordem de 2^m. Mallat (1989) desenvolveu um algoritmo para decomposição e reconstrução de imagens no qual mostra que a diferença de uma informação entre a aproximação de um sinal nas resoluções 2^{m+1} e 2^m pode ser extraída pela decomposição deste sinal sobre uma base ortonormal de *wavelets* em $L^2(\mathbb{R})$. Importante salientar que o conceito de sinal no âmbito do processamento de sinais é uma amostra discreta de uma função temporal. Neste documento o termo sinal significa vetor.

A análise em multirresolução goza das seguintes propriedades.

a) $\bigcap_{-\infty}^{\infty} V_j = \{0\}$.

b) $\bigcup_{-\infty}^{\infty} V_j$ é densa em $L^2(\mathbb{R})$.

c) Qualquer $f \in L^2(\mathbb{R})$, qualquer $j \in Z$, $f(x) \in V_j \Leftrightarrow f(2x) \in V_{j+1}$.

d) Existe uma função $w_0 \in L^1(\mathbb{R}) \cap L^2(\mathbb{R})$, tal que

d.1) $\{w_0(x-k), k \in Z\}$ é uma base ortonormal de V_0.

d.2) $\{2^{j/2} w_0(2^j x - k)\}_{k \in Z}$ é uma base ortonormal a V_j.

Outras características importantes desses subespaços são:

a) V_0 é gerado por um conjunto ortonormal obtido por meio de translações inteiras de uma única função escaladora $\phi \in L^2(\mathbb{R})$, $\{\phi(x - n)\}_n \in Z$. Para cada inteiro j, V_j é gerado por uma base ortonormal consistindo de translações de escalas da função ϕ:

$$V_j = \text{span}\{\ \phi\ (2^{-j}x - k)\}_k \in Z. \tag{4.8}$$

b) Para cada inteiro j, $\mathbf{V}_j = \mathbf{V}_{j+1} \oplus \mathbf{W}_{j+1}$, onde cada W_{j+1} é gerado por um conjunto ortonormal obtido por meio de translações e escalamentos de uma única função ψ chamada *wavelet*,

$$W_j = \text{span}\{\psi(2^{-j}x - k)\}_k \in Z. \tag{4.9}$$

c) O conjunto obtido da forma $\{\psi(2^{-j}x - k)\}_k \in Z$ é uma base de $L^2(\mathbb{R})$.

Munidos pelas propriedades e características acima, procura-se agora entender o significado prático de algumas de suas ideias, bem como identificar a finalidade da AMR.

Suponha que se deseje obter aproximações de uma função $f \in L^2(\mathbb{R})$ em vários níveis de resoluções. Assim cada subespaço V_j será constituído por funções aproximantes, sendo que a melhor aproximação é obtida considerando-se a projeção ortogonal de f sobre cada V_j. Como V_{j+1} está contido em V_j , então a resolução (j) agrega mais informações que a resolução V_{j+1} (escala 2^{-j}). No limite, ($j \to \infty$) a função aproximada converge para a função original, propriedade (b). Por outro lado, aproximações de f em níveis de resolução cada vez menores, perdem-se informação, de forma que no limite, ($j \to -\infty$) a função f converge para a função nula, propriedade (a). A informação que é perdida de V_j para V_{j+1} pode ser representada pelo espaço W_j, que é o complemento ortogonal de V_j em V_{j+1}, característica (b). Outra conclusão desta característica é que $W_j \perp V_j$.

Por meio da análise em multirresolução é possível analisar uma amostra disponível em várias escalas de resolução. Morettin escreve textualmente: “É

como se pudéssemos “olhar” esses dados por um microscópio e ver seu comportamento em várias magnificações” (MORETTIN, 1999, p. 169). Vetterli e Herley (1992) afirmam que o objetivo da multirresolução é observar a floresta e as árvores.

Após encontrar a função ϕ, a função *wavelet* associada, com todas as propriedades de ortogonalidade exigidas, pode ser encontrada diretamente, (BRIGGS; HENSON, 1993). A implementação da (AMR) pode ser feita por meio de um banco de filtros como será visto mais adiante.

Em questões práticas, como o processamento de imagens, a análise em multirresolução é empregada em problemas discretos definidos ou projetados no subespaço $\mathbf{V}_0$ que representa o mais alto nível de resolução em AMR, ou no *grid* mais refinado no *multigrid*. Assim, dada qualquer função $\mathbf{u} \in \mathbf{V}_0$, os coeficientes $c_{0,k}$ da expansão de **u** em termos da base $\{\phi\ (x - n)\}$ são obtidos usando a ortogonalidade de ϕ e ψ

$$\mathbf{u}(x) = \sum_k c_{0,k}\phi(x-k) \tag{4.10}$$

A expressão acima pode ser considerada como a representação de **u** na malha mais refinada.

Os coeficientes $c_{1,k}$ e $d_{1,k}$ podem ser encontrados para representar o vetor **u** em V_1 e W_1 da forma

$$\mathbf{u}(x) = \sum_k c_{1,k}\phi(\frac{x}{2}-k) + d_{1,k}\psi(\frac{x}{2}-k) \tag{4.11}$$

O processo pode continuar até atingir o nível mais grosseiro possível, sempre decompondo cada representação de **u** em V_j no próximo par de malhas menos refinadas V_{j+1} e W_{j+1} (BRIGGS; HENSON, 1993). Mallat (1989) propôs um algoritmo de complexidade O(N) para processar essa decomposição.

Têm-se agora os elementos que possibilitam a conexão entre a análise em multirresolução e o *multigrid*. Para isso considera-se a equação (2.1), $A\mathbf{u} = \mathbf{f}$, que na malha mais refinada Ω^h tem a forma

$$\sum_k c_{0,k} \langle \phi_{0j}, A\phi_{0j} \rangle = f_{0,j}, \forall j \tag{4.12}$$

Na equação (4.12) $c_{0,k}$ e $f_{0,j}$ representam respectivamente os coeficientes da representação da solução **u** e de **f**, em V_0.

De modo similar, o problema pode ser representado na malha imediatamente mais grosseira, utilizando as representações de **u** e **f** em (V_1, W_1), e as relações de ortogonalidade,

$$\sum_k c_{1,k} \langle \phi_{1j}, A\phi_{1j} \rangle + d_{1,k} \langle \phi_{1j}, A\psi_{1j} \rangle = f_{1,j}, \forall j \tag{4.13}$$

e

$$\sum_k c_{1,k} \langle \psi_{1j}, A\phi_{1j} \rangle + d_{1,k} \langle \psi_{1j}, A\psi_{1j} \rangle = g_{1,j}, \forall j \tag{4.14}$$

Conforme a teoria de multirresolução, a decomposição de V_0 na soma de subespaços V_1 e W_1 para uma função suave **u** é interessante porque a maior parte da variação fica contida em V_1 e a representação de **u** em W_1 é praticamente nula. Desta forma, o problema dado na equação (4.13) pode ser considerado como problema na malha grosseira para os componentes suaves da solução, enquanto a equação (4.14) representa o problema em Ω^{2h} os componentes oscilatórios. Ou seja, as funções ϕ_{1k} geram um mesmo espaço da imagem do operador de prolongação I_{2h}^{h} e ψ_{1k} formam uma base para o espaço do núcleo da restrição I_h^{2h}. Os operadores de restrição e prolongação foram definidos na seção 2.3. Para simplificar a notação, o operador de restrição I_h^{2h} será identificado pela letra R, $I_h^{2h} = R$; e o operador de prolongação I_{2h}^{h} pela letra P, $I_{2h}^{h} = P$. No contexto do AMG, $R = P^T$.

Assim, no contexto da multirresolução, tem-se (BRIGGS; HENSON, 1993)

$$V_0 = \text{span}\{\phi_{1k}\} \oplus \text{span}\{\psi_{1k}\} = V_1 \oplus W_1, \tag{4.15}$$

enquanto em termos do *multigrid* pode-se escrever

$$\Omega^h = \text{span}\{\phi_{1k}\} \oplus \text{span}\{\psi_{1k}\} = \text{Imagem}\{P\} \oplus \text{Núcleo}\{R\}. \tag{4.16}$$

Das expressões acima, conclui-se que o *multigrid* produz a mesma decomposição ortogonal em Ω^h que a multirresolução produz no espaço V_0.

Como no *multigrid* as componentes oscilatórias são removidas pelo processo de relaxação, tanto o segundo termo do lado esquerdo da equação (4.13) quanto da equação (4.14) podem ser suprimidos. Além disso, a matriz gerada pelo produto interno $<\phi_{1j}, A\phi_{1k}>$ é exatamente a matriz da equação residual em Ω^{2h}.

4.2 BANCO DE FILTROS

Uma das maneiras de se gerar *wavelets* é pela função escala ϕ, que é uma solução da equação

$$\phi(t) = \sqrt{2}\sum_k \ell_k \phi(2t-k). \tag{4.17}$$

A função expressa na equação (4.17) gera uma família ortonormal em $L^2(\Re)$,

$$\phi_{j,k}(t) = 2^{\frac{j}{2}}\phi(2^j t - k),\ j,\ k\ \in Z. \tag{4.18}$$

Assim, ψ pode ser obtida de ϕ por

$$\psi(t) = \sqrt{2}\sum_k h_k \phi(2t-k), \tag{4.19}$$

na qual

$$h_k = (-1)^k \ell_{1-k} \tag{4.20}$$

A equação (4.20) define um filtro **h** como um operador linear invariante no tempo. Para o caso discreto, **h** = [$h(0)$, $h(1)$, ..., $h(n)$]. Um banco de filtros é um conjunto de filtros (STRANG; NGUYEN, 1997).

Os termos ℓ_k e h_k são coeficientes de filtros passa baixo (*low pass*) e passa alto (*high pass*) respectivamente, denominados filtro espelho em quadratura, utilizados para calcular a transformada de *wavelet* discreta, DWT e são dados pelas expressões seguintes (MORETTIN, 1999),

$$\ell_k = \sqrt{2} \int_{-\infty}^{\infty} \phi(t)\phi(2t-k)dt, \tag{4.21}$$

$$h_k = \sqrt{2} \int_{-\infty}^{\infty} \psi(t)\phi(2t-k)dt. \tag{4.22}$$

Wavelets como as de Haar e Daubechies dão origem a filtros de resposta impulsiva finita (FIR). A seguir alguns filtros com suporte compacto de Haar e Daubechies são apresentados (DAUBECHIES, 1988; MORETTIN, 1999). Filtro com suporte compacto é sinônimo de filtro de comprimento finito (VETTERLI; HERLEY, 1992). As mínimas exigências para tais filtros FIR são:

1) o comprimento do filtro deve ser par;

2) $\sum_n h_n = \sqrt{2}$;

3) $\sum_n (h_n, h_{n-2k}) = \begin{cases} 1 \ se\, k = 0 \\ 0 \ se\, k \neq 0 \end{cases}$

Coeficientes de Haar ou Daubechies-2: filtro comprimento 2

$$hD_2 = \left[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right] \tag{4.23}$$

Coeficientes Daubechies 4: filtro comprimento 4

$$hD_4 = \left[\frac{1+\sqrt{3}}{4\sqrt{2}}, \frac{3+\sqrt{3}}{4\sqrt{2}}, \frac{3-\sqrt{3}}{4\sqrt{2}}, \frac{1-\sqrt{3}}{4\sqrt{2}}\right] \tag{4.24}$$

Coeficientes Daubechies 6: filtros comprimento 6

$$hD_6 = [h_1, h_2, h_3, h_4. h_5, h_6] \tag{4.25}$$

com,

$$h_1 = \frac{1+\sqrt{10}-\sqrt{5+2\sqrt{10}}}{16\sqrt{2}},$$

$$h_2 = \frac{5+\sqrt{10}-3\sqrt{5+2\sqrt{10}}}{16\sqrt{2}},$$

$$h_3 = \frac{10-2\sqrt{10}-2\sqrt{5+2\sqrt{10}}}{16\sqrt{2}},$$

$$h_4 = \frac{10-2\sqrt{10}+2\sqrt{5+2\sqrt{10}}}{16\sqrt{2}},$$

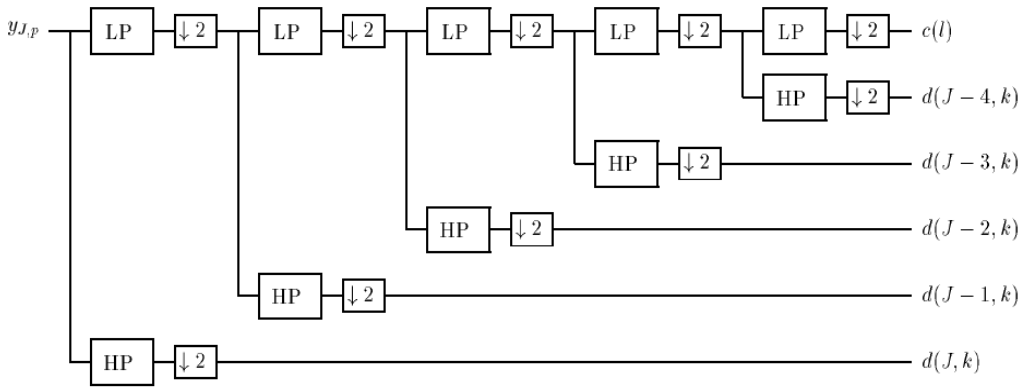
$$h_5 = \frac{5+\sqrt{10}+3\sqrt{5+2\sqrt{10}}}{16\sqrt{2}},$$

$$h_6 = \frac{1+\sqrt{10}+\sqrt{5+2\sqrt{10}}}{16\sqrt{2}}.$$

Este assunto pode ser aprofundado pesquisando a literatura especializada. Dentre as inúmeras publicações podem ser destacados os trabalhos de Chui (1992), Cody (1992), Tewfik et al. (1992) e Kaiser (1994) .

Na prática, a DWT é efetivamente calculada pelo algoritmo denominado por Mallat (1989) de algoritmo piramidal. Uma ilustração deste algoritmo pode ser visualizada na figura 4.3, que utiliza os filtros passa-baixa (LP), e passa-alta (HP).

Figura 4.3: Algoritmo Piramidal com banco de filtros



Fonte: CODY, 1992.

Na figura 4.3 o símbolo (↓2) significa decimação (sub amostragem) por dois. Isto indica que as amostras de índice ímpar do vetor original devem ser eliminadas. Em outras palavras, o processo de decimação por 2 filtra um de cada dois componentes de um sinal discreto. Como pode ser observado, o processo de filtragem do sinal $y_{J,p}$ acontece recursivamente.

No j-ésimo passo, o algoritmo calcula $c_{j,k}$ e $d_{j,k}$ a partir dos coeficientes suaves do nível $j + 1$, $c_{j+1,k}$, pelas expressões seguintes

$$\boldsymbol{c}_{j,k} = \sum_n \ell_{n-2k} \mathbf{c}_{j+1,n} \tag{4.26}$$

$$\boldsymbol{d}_{j,k} = \sum_n h_{n-2k} \mathbf{c}_{j+1,n} \tag{4.27}$$

Assim, um sinal discreto $\boldsymbol{f}$ de comprimento N, com N par, é decomposto em dois subsinais $\mathbf{c}_1$ e $\mathbf{d}_1$, cujos elementos são $c_{1,k}$ e $d_{1,k}$, cada um com a metade do comprimento de $\boldsymbol{f}$, no caso de decimação por dois. O subsinal $\mathbf{c}_1$ é a média ou a tendência de $\boldsymbol{f}$, e $\boldsymbol{d}_1$ representa a flutuação de $\boldsymbol{f}$, e são representados conforme segue,

$$\mathbf{c}_1 = [c_{1,1}, c_{1,2}, \ldots, c_{1,N/2}] \tag{4.28}$$

$$\mathbf{d}_1 = [d_{1,1}, d_{1,2}, \ldots, d_{1,N/2}]. \tag{4.29}$$

Assim, a DWT transforma $\boldsymbol{f}$, em

$$\text{DWT}(\boldsymbol{f}) = [\boldsymbol{c}_1|\boldsymbol{d}_1] = [c_{1,1}, c_{1,2}, \ldots, c_{1,N/2}|d_{1,1}, d_{1,2},\ldots,d_{1,N/2}]. \tag{4.30}$$

Utilizando filtros de comprimento 2 de Haar, tem-se:

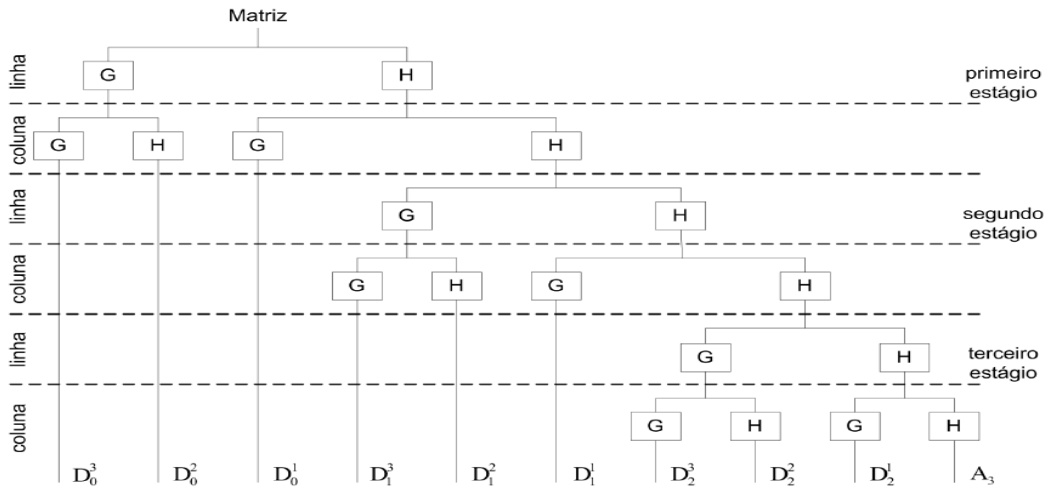
$$\mathbf{c}_{1,\mathrm{k}} = \frac{f_{2k-1} + f_{2k}}{\sqrt{2}} \tag{4.31}$$

$$\mathbf{d}_{1,\mathrm{k}} = \frac{f_{2k-1} - f_{2k}}{\sqrt{2}} \tag{4.32}$$

Caso o sinal $\boldsymbol{f}$ seja suave, os valores de flutuação dados pela equação (4.32) são desprezíveis em comparação com os valores da tendência do vetor $\boldsymbol{f}$ dados pela equação (4.31). Neste caso, os dados de $\mathbf{d}_{1,k}$ podem ser descartados.

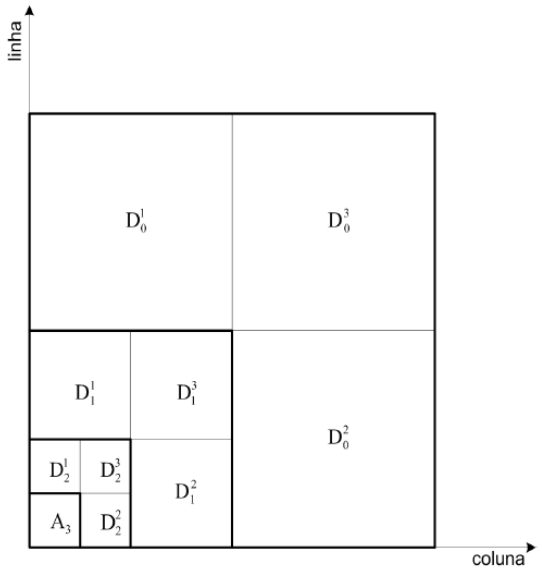
Para o caso bidimensional o sinal discreto é uma matriz e o processo unidimensional é aplicado de forma alternada, nas linhas e colunas da matriz provocando uma transformação conforme mostram as figuras 4.4 e 4.5.

Figura 4.4: Transformada discreta *wavelet* de uma matriz em termos de filtros. *G* e *H* representam, respectivamente, filtros passa-alta e passa-baixa



Fonte: PEREIRA, 2007.

Figura 4.5: Representação da matriz (sinal) $A = A_0$ na transformada *wavelet* bidimensional. Em cada nível *i*, é gerada uma aproximação A_{i+1}, e três classes de coeficientes de detalhes D_i^1, D_i^2, D_i^3.



Fonte: PEREIRA, 2007.

Uma DWT via banco de filtros pode ser representada por meio de uma matriz *R*. Para filtros de comprimento *D* e um processo de decimação por 2, a matriz *R* possui a configuração seguinte (GARCIA et al., 2008)

$$R_{N\,X\,N} = \begin{bmatrix} h_1 & h_2 & \cdots & h_D & & & & \\ & & h_1 & h_2 & \cdots & h_D & & \\ & & & \ddots & \vdots & & & \\ h_3 & h_4 & \cdots & h_D & & & h_1 & h_2 \\ g_1 & g_2 & \cdots & g_D & & & & \\ & & g_1 & g_2 & \cdots & g_D & & \\ & & & & \vdots & & & \\ g_3 & g_4 & \cdots & g_D & & & g_1 & g_2 \end{bmatrix} \tag{4.33}$$

na qual N é o número de amostras do sinal, h_i são os coeficientes dos filtros passa–baixa e g_i os coeficientes dos filtros passa–alta.

Caso não haja necessidade de reconstruir as matrizes e os vetores, os elementos g_i da matriz dada pela equação (4.33) podem ser suprimidos. Assim,

$$R_{\frac{N}{2}\,X\,N} = \begin{bmatrix} h_1 & h_2 & \cdots & h_D & & & & \\ & & h_1 & h_2 & \cdots & h_D & & \\ & & & \ddots & & \vdots & & \\ h_3 & h_4 & \cdots & h_D & & & h_1 & h_2 \end{bmatrix} \tag{4.34}$$

No contexto do *multigrid*, o operador de prolongamento P e a matriz A^{2h} em Ω^{2h} são obtidos conforme segue

$$P = R^T \tag{4.35}$$

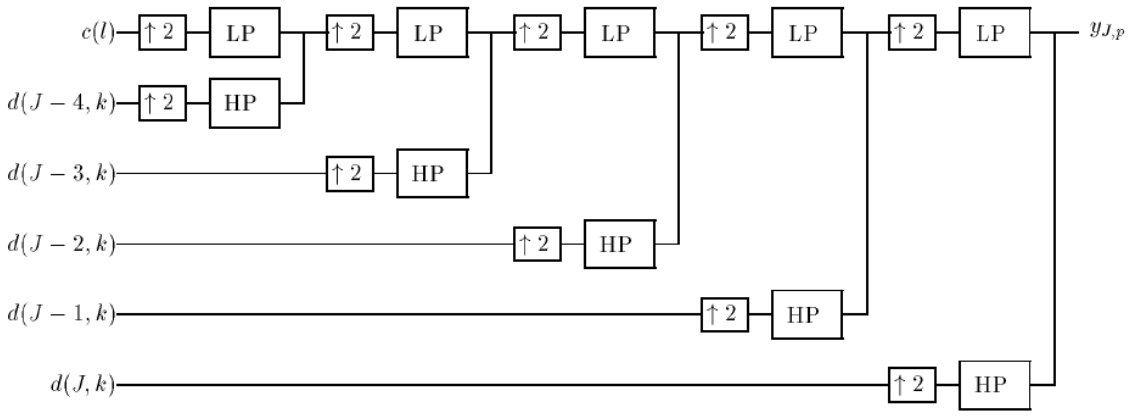
$$A^{2h} = RA^hP \tag{4.36}$$

O sinal $\boldsymbol{f}$ transformado conforme a equação (4.30) pode ser reconstituído, por meio do filtro de Haar, através da expressão

$$\boldsymbol{f} = [\, \frac{c_{1,1}+d_{1,1}}{\sqrt{2}}, \frac{c_{1,1}-d_{1,1}}{\sqrt{2}}, \ldots, \frac{c_{1,N/2}+d_{1,N2}}{\sqrt{2}}, \frac{c_{1,1}-d_{1,1}}{\sqrt{2}} \,] \tag{4.37}$$

A figura 4.6 mostra o algoritmo piramidal inverso, ou seja, o algoritmo que reconstrói o sinal ***f*** submetido a uma DWT.

Figura 4.6: Algoritmo Piramidal Inverso

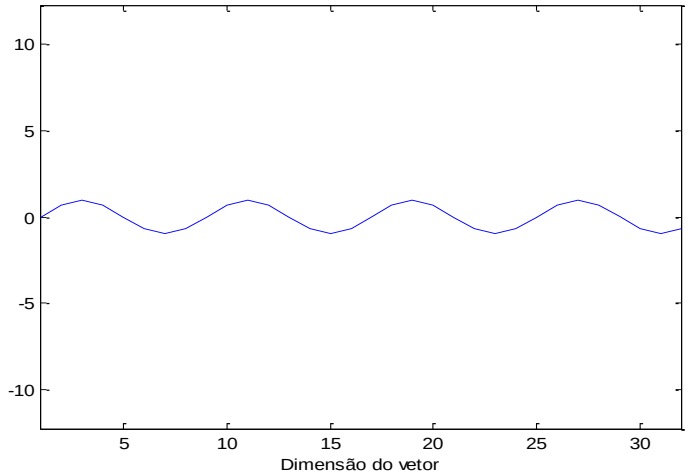


Fonte: CODY, 1992.

A seguir implementa-se a teoria abordada nesta seção por meio de exemplos numéricos e compara-se alguns resultados com a teoria convencional do *multigrid* geométrico. Inicialmente, considera-se um vetor $\mathbf{v}^h$ expresso pela equação (4.38), cuja representação gráfica é suave conforme mostra a figura 4.7.

$$\mathbf{v}^h = v_8(j) = sen(8\pi(j\text{-}1)/n),\ n = 32,\ j = 1, 2, \ldots, 32 \qquad (4.38)$$

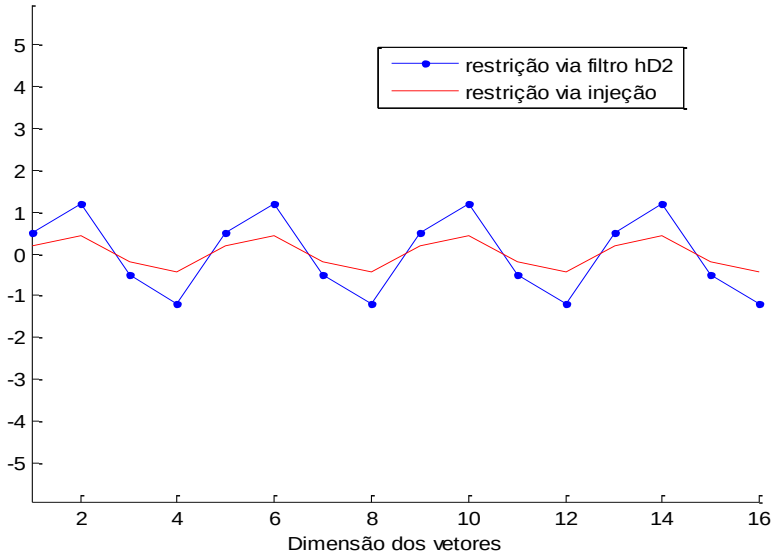
Figura 4.7: Representação gráfica do vetor (modo de Fourier, k = 8), equação (4.38)



Fonte: O autor, 2013.

A figura 4.8 mostra o vetor $\mathbf{v}^h$ expresso graficamente pela figura 4.7 restringido para malha imediatamente mais grosseira, $\mathbf{v}^{2h}$, por meio da equação (4.34), com filtro de Haar de comprimento 2, e por meio da injeção dada pela equação (2.32). Observa-se que a restrição via filtro hD2, de Haar, potencializa a amplitude do erro suave restringido, mantendo a frequência inalterada.

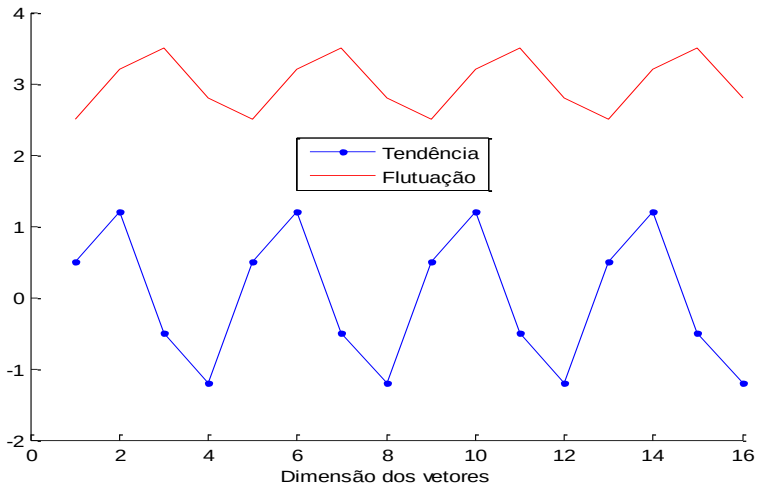
Figura 4.8: Vetor restringido para malha mais grosseira filtro de Haar x Injeção



Fonte: O autor, 2013.

A figura 4.9 mostra a decomposição do vetor dado pela equação (4.38) por meio da equação (4.31), gráfico inferior que representa a tendência do vetor original, e pela equação (4.32), gráfico superior que mostra a flutuação do vetor representado pela figura 4.7. Observa-se que a magnitude do vetor que apresenta a tendência do vetor suave é superior à magnitude daquele que mostra a flutuação do vetor suave original.

Figura 4.9: Decomposição de um vetor suave em $\mathbf{c}_{1,k}$ (tendência) e $\mathbf{d}_{1,k}$ (flutuação)



Fonte: O autor, 2013.

4.3 ALGORITMOS PARA IMPLEMENTAÇÃO DO WAMG

Apresentam-se nesta seção quatro algoritmos, a saber, restrição e prolongação por meio do filtro de Daubechies. Apresenta-se também o algoritmo PWAMG no qual o WAMG é utilizado como um precondicionador, assim como o algoritmo do método dos gradientes conjugados precondicionado.

4.3.1 Algoritmos para restrição e prolongação por meio do filtro de Daubechies

Nesta seção são apresentados quatro algoritmos para executar as operações de restrição e prolongação no WAMG, utilizando-se filtros de Daubechies comprimento 4, com decimação por 2, hD4(2) e decimação 4, hD4(4). Tais algoritmos visam implementar os itens 3, 6, 8 e 10 do Algoritmo 2.2. Vale relembrar que o algoritmo 2.2 é da técnica *multigrid*, independente se é o GMG ou WAMG.

Como tratado na seção 4.2, na decimação por dois, um vetor no nível grosso $2h$ tem a metade da dimensão do nível fino h. Sejam c_i e f_i elementos dos vetores na malha $2h$ e h respectivamente. Ou seja

$$\boldsymbol{c} = [c_1, \ldots, c_i, \ldots c_{L/2}] \tag{4.39}$$

$$\boldsymbol{f} = [f_1, \ldots, f_i, \ldots f_L] \tag{4.40}$$

Assim, utilizando filtro hD4(2), tem-se.

Algoritmo 4.1: Restrição através do filtro de Daubechies, decimação por 2.

$k = 1$

Faça enquanto $k < L/2$

Se $k < L/2 - 1$

$$c_k = \sum_{i=1}^{4} f_{2k-2+i} h_i$$

Senão se $k = L/2$

$$c_{L/2} = \frac{3-\sqrt{3}}{4\sqrt{2}} f_1 + \frac{1-\sqrt{3}}{4\sqrt{2}} f_2 + \frac{1+\sqrt{3}}{4\sqrt{2}} f_{L-1} + \frac{3+\sqrt{3}}{4\sqrt{2}} f_L$$

Fim do Se

Fim do Enquanto.

No Algoritmo 4.1 acima, $k = 1, \ldots L/2$, na qual L é a dimensão do vetor no nível fino, h_i o i-ésimo termo do filtro de Daubechies, conforme equação (4.24). Os quatro primeiros elementos e o último do nível grosso c_k obtidos do nível fino $\boldsymbol{f}_k$, por meio do algoritmo 4.1 são elencados a seguir.

$$c_1 = \frac{1+\sqrt{3}}{4\sqrt{2}} f_1 + \frac{3+\sqrt{3}}{4\sqrt{2}} f_2 + \frac{3-\sqrt{3}}{4\sqrt{2}} f_2 + \frac{1-\sqrt{3}}{4\sqrt{2}} f_4$$

$$c_2 = \frac{1+\sqrt{3}}{4\sqrt{2}} f_3 + \frac{3+\sqrt{3}}{4\sqrt{2}} f_4 + \frac{3-\sqrt{3}}{4\sqrt{2}} f_5 + \frac{1-\sqrt{3}}{4\sqrt{2}} f_6$$

$$c_3 = \frac{1+\sqrt{3}}{4\sqrt{2}} f_5 + \frac{3+\sqrt{3}}{4\sqrt{2}} f_6 + \frac{3-\sqrt{3}}{4\sqrt{2}} f_7 + \frac{1-\sqrt{3}}{4\sqrt{2}} f_8$$

$$c_4 = \frac{1+\sqrt{3}}{4\sqrt{2}} f_7 + \frac{3+\sqrt{3}}{4\sqrt{2}} f_8 + \frac{3-\sqrt{3}}{4\sqrt{2}} f_9 + \frac{1-\sqrt{3}}{4\sqrt{2}} f_{10}$$

$$c_{L/2} = \frac{3-\sqrt{3}}{4\sqrt{2}} f_1 + \frac{1-\sqrt{3}}{4\sqrt{2}} f_2 + \frac{1+\sqrt{3}}{4\sqrt{2}} f_{L-1} + \frac{3+\sqrt{3}}{4\sqrt{2}} f_L .$$

A seguir apresenta-se o algoritmo de prolongação, que transfere um vetor de um nível grosso de dimensão *2h* para o nível fino de dimensão *h*.

Algoritmo 4.2: Prolongação por meio do filtro de Daubechies, decimação por 2.

Faça $f_1 = c_1 h_1 + c_{L/2} h_3$; $f_2 = c_1 h_2 + c_{L/2} h_4$
Para $j = 4, \ldots, L$, sendo L a dimensão do vetor no nível fino
Faça enquanto $j \leq L$
$f_{j-1} = c_{j/2-1}\, h_3 + c_{j/2}\, h_1$
$f_j = c_{j/2-1}\, h_4 + c_{j/2}\, h_2$
$j = j + 2$
Fim do enquanto

No Algoritmo 4.2 c_k é o elemento k do vetor do nível grosso, f_k: elemento k do vetor no nível fino, e h_i: elemento i do filtro de Daubechies de comprimento 4, conforme equação (4.24). Observa-se também que a mudança para k par ou ímpar são apenas os termos do filtro, mantendo-se inalterados os elementos do vetor no nível grosso.

A seguir mais dois algoritmos são apresentados, de Restrição e Prolongação, por meio do mesmo filtro de Daubechies de comprimento 4, agora utilizando-se decimação por 4. Neste caso, a dimensão de um vetor na malha grossa *H* corresponde a um quarto da dimensão do vetor na malha fina *h*.

Algoritmo 4.3: Restrição por meio do filtro de Daubechies, decimação por 4.

Faça enquanto $i \leq L/4$
Faça enquanto $j \leq 4$
$c_i = f_{4i-4+j} h_j$
Fim do enquanto j
Fim do enquanto i.

No algoritmo 4.3, $1 \leq i \leq L/4$, $1 \leq j \leq 4$, Aux_1 é uma variável auxiliar, f_k é o elemento do vetor na malha fina de dimensão h, h_j é elemento j do filtro de Daubechies de comprimento 4, conforme a equação (4.24), e c_i é o elemento do vetor no nível grosso de dimensão H.

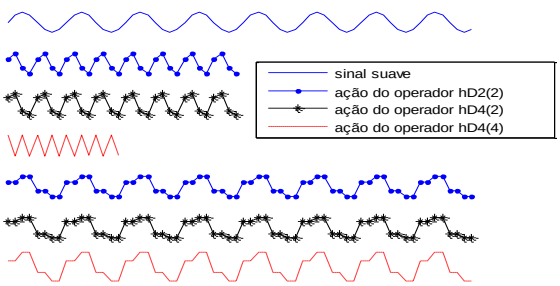
Algoritmo 4.4: Prolongação via filtro de Daubechies, decimação por 4.

Faça $Aux_1 = 1$, $Aux_2 = 1$
Faça enquanto $Aux_1 \leq L$
  Faça enquanto $j \leq 4$
    $f_{Aux1} = h_j c_{Aux2}$
    $Aux_1 = Aux_1 + 1$
  Fim do enquanto j
  $Aux_2 = Aux_2 + 1$
Fim do enquanto Aux_1.

No algoritmo 4.4 acima, Aux_1 e Aux_2 são variáveis auxiliares, $1 \leq i \leq L$, $1 \leq j \leq 4$, f_k e c_k são os elementos do vetor no nível fino de comprimento h e elementos do vetor no nível grosso de comprimento H respectivamente, h_j são os elementos do filtro de Daubechies de comprimento 4, conforme a equação (4.24).

A seguir a figura 4.10 mostra as ações destes algoritmos quando atuando num sinal suave.

Figura 4.10: Ações dos operadores de restrição e prolongação num vetor suave



Fonte: O autor, 2013.

A figura 4.10 mostra no primeiro gráfico de cima para baixo um sinal suave, seguido das atuações dos operadores de restrição sobre este sinal. Logo mais abaixo, a mesma figura mostra a ação dos operadores de prolongação atuando nos sinais restringidos, sendo que transferências utilizando filtros de Daubechies de comprimento 4 foram efetuadas por meio dos algoritmos descritos nesta seção. Observa-se que os algoritmos de prolongação não recompõem o sinal original, embora o torne novamente suave. Isto ocorre porque o operador utilizado no caso é aquele estabelecido genericamente pela equação (4.34) no qual aparecem apenas os componentes de filtros passa-baixa, h_i.

4.3.2 Algoritmo PWAMG no qual o precondicionamento do GCP é efetuado por meio de um ciclo do WAMG

Nesta seção o algoritmo PWAMG é apresentado. Trata-se de uma variante do algoritmo GCP, no qual um ciclo do WAMG é utilizado para efetuar o precondicionamento.

Algoritmo 4.5: PWAMG

(1) $\boldsymbol{x}_0 \leftarrow$ solução inicial
(2) $aux = 1$
(3) $\boldsymbol{x} \leftarrow$ solução aproximada obtida por um ciclo do WAMG, $\boldsymbol{x}^h \leftarrow V^h(\boldsymbol{x}_0{}^h, \boldsymbol{b}^h)$
(4) $\boldsymbol{r} = \boldsymbol{b} - A\boldsymbol{x}$
Faça enquanto verdadeiro
(5) $\boldsymbol{z} \leftarrow$ precondicionamento obtido por um ciclo do WAMG, $\boldsymbol{z}^h \leftarrow V^h(\boldsymbol{x}_0{}^h, \boldsymbol{r}^h)$
(6) $\rho = \boldsymbol{r}^T\boldsymbol{z}$
if $aux = = 1$
(7) $\boldsymbol{p} = \boldsymbol{z}$
else
(8) $\boldsymbol{p} = \boldsymbol{z} + \rho\boldsymbol{p}$
end if
(9) $\boldsymbol{q} = A\boldsymbol{p}$
(10) $\alpha = \rho/(\mathbf{p}^T\boldsymbol{q})$
(11) $\boldsymbol{x} \leftarrow \boldsymbol{x} + \alpha\boldsymbol{p}$
(12) $\boldsymbol{r} = \boldsymbol{r} - \alpha\boldsymbol{q}$
(13) aux = $aux + 1$
Fim do enquanto verdadeiro

4.3.3 Algoritmo do método do gradiente conjugado precondicionado

Nesta seção o algoritmo já bastante conhecido do método dos gradientes conjugados precondicionado é apresentado (HESTENES; STIEFEL, 1952).

Algoritmo 4.6: Método do gradiente conjugado precondicionado - GCP

(1) $\boldsymbol{x} = \boldsymbol{x}_0 \leftarrow$ solução inicial
(2) $\boldsymbol{r} = \boldsymbol{b} - A\boldsymbol{x}$
(3) $\boldsymbol{z} \leftarrow M^{-1}r$
(4) $aux = 1$
Faça enquanto verdadeiro
(5) $\rho = \boldsymbol{r}^{\mathrm{T}}\boldsymbol{z}$
If $aux = = 1$
(6) $\boldsymbol{p} = \boldsymbol{z}$
Else
(7) $\beta = \rho / \rho_0$
(8) $\boldsymbol{p} = \boldsymbol{z} + \beta\boldsymbol{p}$
End if
(9) $\rho_0 = \rho$
(10) $\boldsymbol{q} = A\boldsymbol{p}$
(11) $\alpha = \rho/(\mathbf{p}^{\mathrm{T}}\boldsymbol{q})$
(12) $\boldsymbol{x} \leftarrow \boldsymbol{x} + \alpha\boldsymbol{p}$
(13) $\boldsymbol{r} = \boldsymbol{r} - \alpha\boldsymbol{q}$
(14) $\boldsymbol{z} \leftarrow M^{-1}r$
(15) aux = $aux + 1$
Fim do enquanto verdadeiro

Notas
i) Neste trabalho é utilizado como matriz de precondicionamento M^{-1}, nos passos (3) e (14) do algoritmo 4.6, a matriz $D^{-1/2}$, cujas entradas diagonais são os recíprocos das raízes quadradas das entradas diagonais da matriz A da equação (2.1). Importante relembrar que para o referido algoritmo funcionar, a matriz A do sistema precisa ser definida positiva.
ii) Os passos (3) e (5) do algoritmo 4.5, diferem do Algoritmo 4.6, e neste trabalho, são obtidos por meio do ciclo V do WAMG, ou seja $\boldsymbol{v}^h \leftarrow V^h(\boldsymbol{v}^h, \boldsymbol{b}^h)$.
iii) No algoritmo 4.5 não existe o passo (7) do algoritmo 4.6.

iv) Após o passo (4) o procedimento “While True” deve ser entendido como o critério de parada do programa, seja em número de ciclos, ou em termos de convergência.

v) O vetor $\boldsymbol{r}^T$ é o transposto do vetor $\boldsymbol{r}$.

5 PROBLEMAS INVESTIGADOS E PARÂMETROS AVALIADOS

Neste capítulo são apresentados os problemas investigados, os algoritmos utilizados, os parâmetros avaliados, o relaxador internamente aos ciclos do *multigrid*, o erro considerado, bem como o aplicativo computacional utilizado.

Os testes foram realizados em um computador com processador Intel (R) Core(TM) i7-2600 CPU@, 3,40GHz, 16,0 GB de memória RAM. O código computacional foi escrito em compilador do MATLAB© versão 7.8.

Os algoritmos utilizados são *multigrid* algébrico via wavelets WAMG, gradiente conjugado precondicionado GCP, PWAMG no qual um ciclo do WAMG é utilizado como precondicionador para o GCP, e Gauss-Seidel GS. Os diferentes algoritmos do WAMG são apresentados conforme a nomenclatura: (i) filtro comprimento 2, decimação por 2 - WAMG_hD2(2); e (ii) filtro de comprimento 4 - WAMG_hD4(2) e WAMG_hD4(4), com decimação por 2 e 4 respectivamente.

Os parâmetros avaliados são tempo de processamento e esforço computacional, a saber: memória computacional, número de ciclos para o *multigrid* e número de iterações para o GCP e GS. A memória computacional é obtida por meio da função *memory* do MATLAB.

Todos os casos de *multigrid* desta tese foram processados com esquema de correção CS, ciclo V, e solução inicial nula. A solução da equação residual no nível mais grosseiro do *multigrid* é obtida diretamente via função do MATLAB. O suavizador utilizado internamente em cada nível do *multigrid* é o Gauss-Seidel, exceto quando expresso o contrário. Outros ciclos, além do ciclo V foram pesquisados, mas não apresentaram resultados melhores.

O erro considerado é a norma do máximo para o resíduo, exceto quando se afirmar o contrário.

Quanto a dimensão, os problemas avaliados são uni e bidimensionais, conforme elencados a seguir.

Os problemas unidimensionais avaliados são

- Problema 1: Condução de Calor com Termo Fonte
- Problema 2: Barra Prismática – caso A e caso B
- Problema 3: Viga de Bernoulli – caso A e caso B

Os problemas bidimensionais avaliados são

- Problema 4: Equação de Laplace
- Problema 5: Advecção Difusão
- Problema 6: Viga Bidimensional Sob Estado Plano de Tensões – caso A, caso B e caso C.
- Problema 7: Viga com engaste à esquerda, abaixo, e acima, com carregamento longitudinal.

Os problemas bidimensionais elencados anteriormente são também processados por meio do algoritmo PWAMG, no qual um ciclo do WAMG é utilizado como um precondicionador para o GCP, a saber

- Reanálise do Problema 4: Equação de Laplace
- Reanálise do Problema 5: Advecção Difusão
- Reanálise do Problema 6: Viga Bidimensional Sob Estado Plano de Tensões.

A seguir apresenta-se os sete problemas investigados. Os últimos três problemas, ou seja, reanálise dos problemas 4, 5 e 6, são avaliados por meio do algoritmo 4.5, PWAMG; e são abordados nas seções 7.1, 7.2 e 7.3 respectivamente.

5.1 PROBLEMAS UNIDIMENSIONAIS

Nesta seção são apresentados os problemas unidimensionais investigados neste trabalho, a saber: condução de calor com termo fonte, barra prismática e viga de Bernoulli.

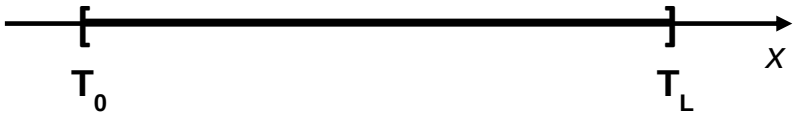
5.1.1 Problema 1: condução de calor unidimensional com termo fonte

O problema de condução de calor unidimensional, com termo fonte *S*, é governado pela equação diferencial de segunda ordem conforme segue

$$\frac{d^2T}{dx^2} + S = 0 \tag{5.1}$$

A equação (5.1) pode ser resolvida analiticamente integrando-a duas vezes em relação a variável *x*, e aplicando em seguida as condições de contorno. O resultado aparece na equação (5.2).

Figura 5.1: Domínio unidimensional, variável espacial e condições de contorno



Fonte: O autor, 2013

$$T = \left[\frac{T_L - T_0}{L} + \frac{q}{2C}(L - x)\right]x + T_0 \tag{5.2}$$

Na Equação (5.1), *T* é a temperatura (ºC), *x* é a coordenada espacial (*m*), e *S* é o termo fonte que está relacionado com a taxa de geração de calor e a condutividade térmica do material. Para geração dos resultados deste problema, a equação (5.1) foi discretizada via elementos finitos. A matriz de rigidez do elemento linear é:

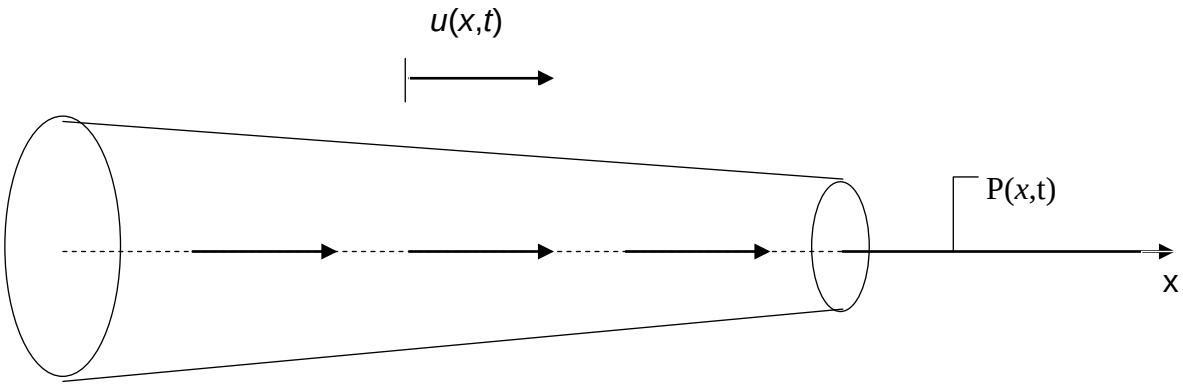
$$\frac{C}{h}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix},$$

na qual *C* é a condutividade térmicas do material e *h* é o tamanho do elemento.

5.1.2 Problema 2: barra prismática

Esta seção apresenta o problema 2, barra prismática unidimensional. Barra é um elemento estrutural de eixo reto de comprimento muito maior que as dimensões de sua seção transversal e que transmite apenas forças axiais de tração ou compressão (ALVES FILHO, 2007), com as seguintes hipóteses básicas adotadas no presente trabalho: (a) a seção transversal que é reta e normal ao eixo da barra antes da deformação permanece reta e normal após a deformação; e (b) o material é elástico e linear (CRAIG JR, 1981). A figura 5.2 mostra uma barra típica.

Figura 5.2: Barra reta



Fonte: Adaptado de CRAIG JR, 1981.

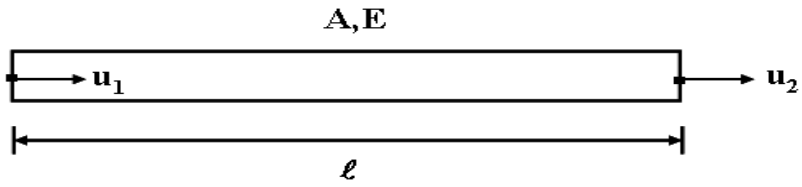
A vibração da barra é um problema dependente do tempo. A equação do movimento que governa este problema é dada pela expressão seguinte

$$\rho A \frac{\partial^2 u}{\partial t^2} - \frac{\partial}{\partial x}\left(EA \frac{\partial u}{\partial x} \right) = p(x,t) \tag{5.3}$$

na qual ρ é a massa específica, $A(x)$ é a área da seção transversal, $E(x)$ é o módulo de elasticidade, p é a força axial externa aplicada por unidade de comprimento e t é o tempo. A equação (5.3) após a introdução das condições de contorno e das condições iniciais, pode ser discretizada e resolvida numericamente, ou seja, o deslocamento $u(x, t)$ encontrado via elementos finitos por exemplo (KWON, 2000).

No presente estudo, o elemento de barra utilizado, de comprimento *l*, possui dois nós com um grau de liberdade em cada nó, a saber, deslocamento axial nos nós inicial e final, conforme figura 5.3.

Figura 5.3: Elemento de barra unidimensional



Fonte: Adaptado de KWON, 2000.

No presente trabalho, o problema da barra foi discretizado via elementos finitos para o caso particular estático, o qual é independente do tempo *t*. Desconsiderando o primeiro termo da Eq. (5.3), a matriz de rigidez do elemento linear é dada por

$$[k]^e = \frac{AE}{l}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{5.4}$$

na qual *A* é a área da seção transversal da barra, e *E* é o módulo de elasticidade do material, *l* é o comprimento do elemento.

5.1.3 Problema 3: viga de bernoulli

Uma viga é uma barra reta de comprimento muito maior que as dimensões de sua seção transversal e que além de forças axiais pode transmitir também momentos fletores, forças cortantes e momentos torçores (ALVES FILHO, 2007).

A teoria clássica de viga, ou viga de Euler-Bernoulli baseia-se em cinco hipóteses básicas:

1) a seção transversal da viga tem um plano longitudinal de simetria, figura 5.4 (a);

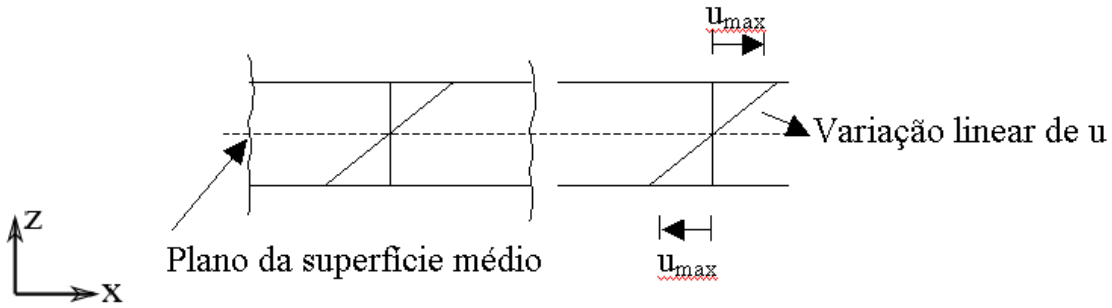
2) o carregamento aplicado atua sobre o plano longitudinal de simetria;

3) uma seção plana originariamente perpendicular ao eixo da viga permanece plana e perpendicular ao eixo da viga após uma deformação, figura 5.4 (b);

4) não ocorre deformação no plano da superfície média, figura 5.4 (a); e

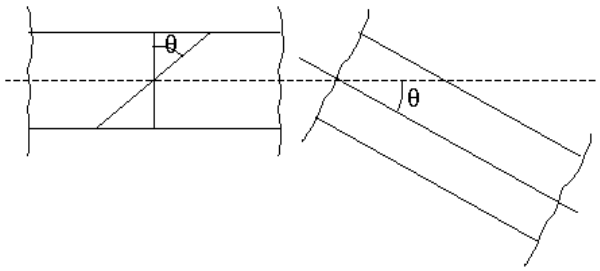
5) um elemento de linha *dx* da viga sofre deslocamentos axial *u*(*x*) na direção *x*, e *w*(*x*) na direção *z*, e sofre uma rotação no plano *xz*, que é a derivada de *w* em relação a *x*, figura 5.4 (c).

Figura 5.4: (a) Viga de Euler-Bernoulli: variação linear do deslocamento normal

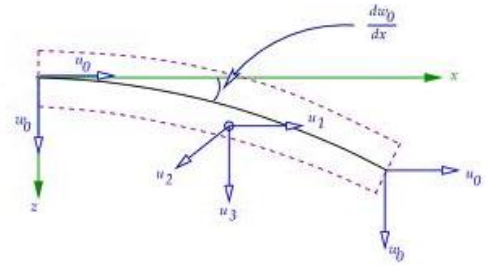


Fonte: Adaptado de GERE e TIMOSHENKO, 1997

Figura 5.4 (b) Viga de Euler-Bernoulli: rotação – (c) Viga de Euler-Bernoulli



(b)



(c)

Fonte: Adaptado de GERE e TIMOSHENKO, 1997.

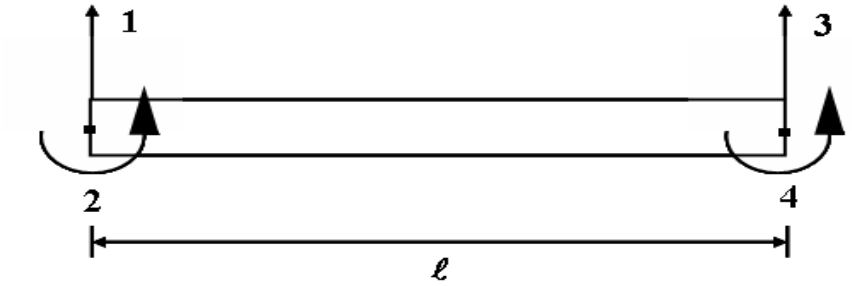
A equação que governa o movimento da viga de Bernoulli no plano *xz* é dada por (KWON, 2000).

$$\rho A \frac{\partial^2 u}{\partial t^2} + \frac{\partial^2}{\partial x^2}\left(EI \frac{\partial^2 u}{\partial x^2} \right) = p(x,t) \tag{5.5}$$

na qual ρ é a massa específica, *A* é a área da seção transversal, *E* é o módulo de elasticidade, *p* é a força externa aplicada por unidade de comprimento e *t* é o tempo, *I* é a inércia. Assim como a equação da barra, a equação da viga de Bernoulli também pode ser resolvida numericamente via elementos finitos (KWON, 2000).

A equação 5.5 foi discretizada via elementos finitos, considerando o problema estático (independente da variável *t*, tempo), e com rigidez a flexão. O elemento de viga utilizado possui dois nós, com dois graus de liberdade por nó, a saber: deslocamento e rotação, conforme mostra a figura (5.5).

Figura 5.5 Elemento de viga de Bernoulli com rigidez a flexão apenas



Fonte: Adaptado de KWON, 2000.

A matriz de rigidez do elemento finito é dada por

$$[k]^e = \frac{EI}{l^3}\begin{bmatrix} 12 & 6l & -12 & 6l \\ 6l & 4l^2 & -6l & 2l^2 \\ -12 & -6l & 12 & -6l \\ 6l & 2l^2 & -6l & 4l^2 \end{bmatrix} \tag{5.6}$$

5.2 PROBLEMAS BIDIMENSIONAIS

Nesta seção são apresentados os problemas bidimensionais investigados, a saber: equação de Laplace, problema envolvendo advecção, e também uma viga sob estado plano de tensões EPT. Os itens avaliados são

tempo de processamento, número de ciclos para o WAMG, número de iterações para o GCP e GS e memória computacional para todos os algoritmos.

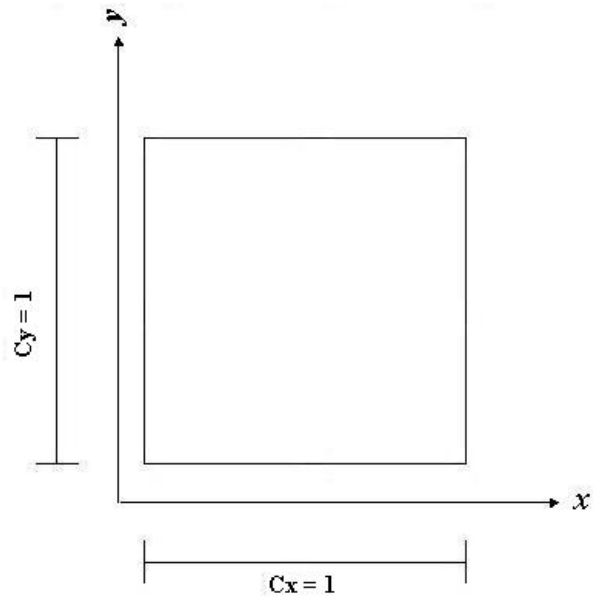
5.2.1 Problema 4: equação de Laplace bidimensional

A equação de Laplace bidimensional analisada neste trabalho é expressa conforme equação (5.7)

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0, \;\; 0 < \mathrm{x}, \mathrm{y} < 1 \tag{5.7}$$

A equação é discretizada via diferenças finitas num quadrado unitário, conforme a figura 5.6, e o estêncil de 5 pontos dado pela equação (5.8).

Figura 5.6: Domínio bidimensional de cálculo para a equação de Laplace



Fonte: PINTO, 2006

$$A = \frac{1}{h^2}\begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix} \tag{5.8}$$

O caso é investigado com as condições de contorno de Dirichlet, ou seja, $T(x, 0) = T(0, y) = T(1, y) = 0$, $T(x,1) = \text{sen}(\pi x)$. A equação (5.7) com as condições de contorno especificadas previamente possui a seguinte solução analítica

$$T(x, y) = sen(\pi x)\frac{senh(\pi y)}{senh(\pi)} \tag{5.9}$$

5.2.2 Problema 5: advecção difusão

O problema de advecção difusão investigado é expresso pela equação seguinte.

$$-\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} + \lambda \frac{\partial u}{\partial x} + \sigma u = f \tag{5.10}$$

Na equação (5.10) λu_x é o termo que representa a advecção, e dependendo dos valores para λ e σ, o problema pode se tornar indefinido ou complexo (BORDNER; SAIED, 1995; WESSELING, 1992). Neste trabalho a equação (5.10) é resolvida numericamente num quadrado unitário, $0 < x, y < 1$, para a condição de contorno $u = 0$, discretizada por diferenças finitas dada pelo estêncil com 5 pontos conforme a equação (5.11)

$$A = \frac{1}{h^2}\begin{bmatrix} & -1 & \\ -1-\lambda h & 4+\lambda h+\sigma h_1 & -1 \\ & -1 & \end{bmatrix} \tag{5.11}$$

na qual $h = 1/(n_x + 1)$; $h_1 = h^2$. O termo independente f discreto é dado pela equação $b_{ij} = sen(\pi x_i)sen(\pi y_j)sen(\sqrt{2}\pi x_i)sen(\sqrt{3}\pi y_j)$.

5.2.3 Problema 6: viga bidimensional sob estado plano de tensões

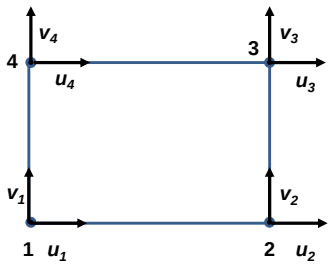
O problema 6 trata-se de uma viga sob estado plano de tensões (EPT). A equação governante de uma viga é dada pela equação (5.12) (GERE e TIMOSHENKO, 1997)

$$\frac{d^2}{dx^2}\left(EI_x \frac{d^2u}{dx^2} \right) = -q \tag{5.12}$$

na qual E é o módulo de elasticidade, q é a força externa aplicada, e I é a inércia. O subscrito x indica que a rigidez a flexão (EI) pode variar em função de x.

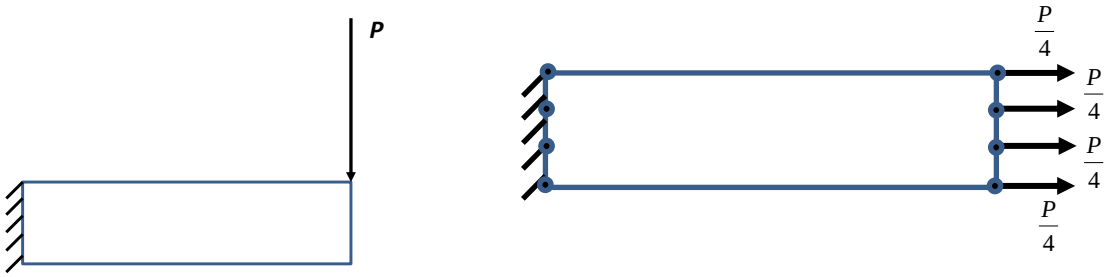
O domínio do problema é discretizado via elementos finitos bidimensionais isoparamétricos, assumindo estado plano de tensões (BORESI; CHONG, 1987) conforme mostra a figura 5.7. Os elementos possuem quatro nós, com dois graus de liberdade cada um, a saber: deslocamento nas direções x e y. A força aplicada P é distribuída igualmente entre os nós, conforme ilustra a figura 5.8, a saber: (a) viga em balanço engastada a esquerda com carregamento vertical na extremidade direita da viga, e (b) viga em balanço engastada a esquerda, com carregamento axial aplicado na extremidade direita da viga.

Figura 5.7: Elemento finito de viga



Fonte: Adaptado de KWON, 2000.

Figura 5.8: Viga sob EPT: (a) carregamento vertical, (b) carregamento axial



(a) carga vertical (b) carga axial

Fonte: o autor, 2013.

A matriz de rigidez do elemento finito é obtida pela equação (5.13) (ALVES FILHO, 2007)

$$[k]^e = \int_{\Omega} [B]^T [D][B] d\Omega \tag{5.13}$$

na qual *B* é a matriz deslocamento deformação, *D* é a matriz de Elasticidade (constitutiva), e Ω é o domínio do elemento.

Após a integração numérica da Equação (5.13) a matriz de rigidez de cada elemento finito é obtida pela equação (5.14) (KWON, 2000)

$$[k]^e = [B]^T [D][B] p_x p_y \det(jacob) \tag{5.14}$$

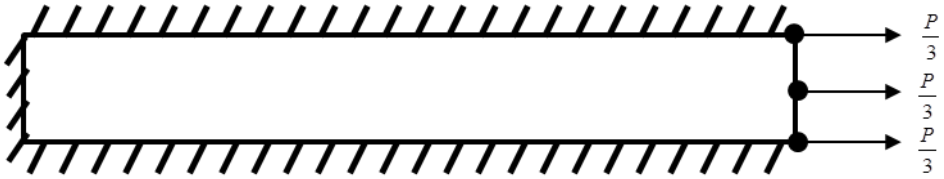
na qual p_x e p_y são os pesos da integração numérica nos eixos *x* e *y* respectivamente, e *det(jacob)* é o determinante do jacobiano da transformação de coordenadas.

5.2.4 Problema 7: viga bidimensional sob estado plano de tensões engastada à esquerda, abaixo, e acima, com carregamento longitudinal

Este problema 7 é na verdade um caso particular do problema 6 no qual a viga sob estado plano de tensões é engastada à esquerda, abaixo e acima, e

aplicada uma força axial *P* distribuída igualmente entre os nós da extremidade direita da viga conforme ilustrado na figura 5.9.

Figura 5.9: Viga engastada à esquerda, abaixo e acima com carregamento axial



6 RESULTADOS NUMÉRICOS PARA PROBLEMAS UNI E BIDIMENSIONAIS

Este capítulo apresenta os resultados numéricos dos sete problemas relacionados no capítulo anterior, cujo objetivo é comparar o desempenho dos algoritmos pesquisados neste trabalho. Os problemas 2 e 3 são desdobrados em casos A e B, enquanto o problema 6 é subdividido nos casos A, B e C.

6.1 RESULTADOS PARA OS PROBLEMAS UNIDIMENSIONAIS

A seguir os problemas unidimensionais investigados neste trabalho conforme descrito na seção 5.1.

6.1.1 Problema 1

O primeiro caso apresentado é um problema de condução de calor unidimensional.

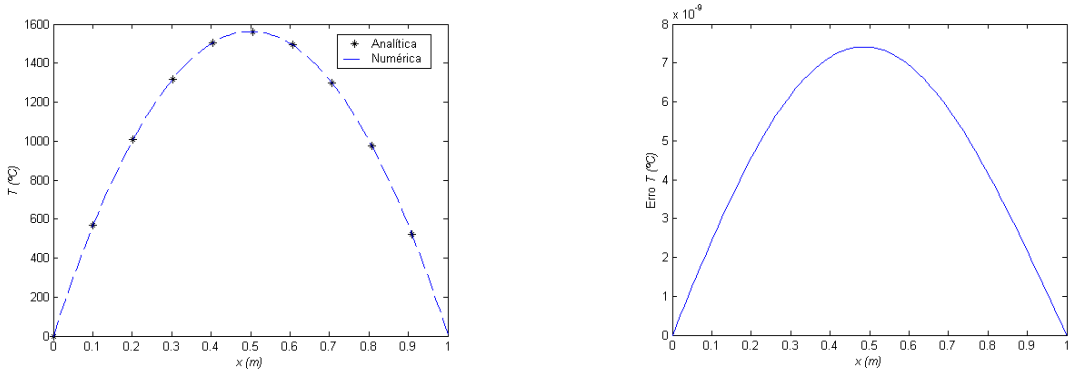
Alguns resultados são mostrados na sequência com os dados especificados a seguir.

$L = 1$	Comprimento do corpo (m)
$T_0 = 0$	Temperatura no contorno esquerdo ($^o\ C$)
$T_L = 8$	Temperatura no contorno direito ($^o\ C$)
$C = 401$	Condutividade térmicas do cobre ($W/m\ ^o\ C$)
$q = 5x10^6$	Taxa de geração de energia por unidade de volume do meio (W/m^3)
$h = L/n$	Comprimento do elemento finito (m)
$S = q/C$	Termo fonte
$tol = 10^{-5}$	Tolerância de erro do resíduo obtida pela norma do máximo

Na figura 6.1a e 6.1b, respectivamente, são mostrados os perfis de temperatura numérica comparada com a solução analítica e a diferença entre a

solução analítica e os dados numéricos, para uma malha de 100 nós e tolerância do erro máximo inferior a 10^{-10}. Como pode ser observado, o erro é pequeno, verificando com isto a discretização do modelo numérico.

Figura 6.1: (a) Perfis de temperatura e (b) diferença entre a solução analítica e iterativa, malha com 100 nós



(a) (b)

Fonte: O autor, 2013.

O problema 6.1.1 foi processado com os seguintes tipos de solução inicial: nula, e randômica. Constatou-se que a solução inicial não afetou o desempenho do *multigrid*, razão pela qual não se apresenta os resultados obtidos, assim como não se explora mais a questão da solução inicial nos próximos casos.

As tabelas 6.1 e 6.2 apresentam o tempo de processamento (*s*), o número de ciclos/iterações e a memória computacional para se obter a tolerância de 10^{-5} durante as simulações considerando a norma do máximo do resíduo, para uma malha com 2.048 e 8.192 graus de liberdade respectivamente, 3 iterações internas para o *multigrid*.

Tabela 6.1: Tempo de processamento (*s*), número de ciclos/iterações e memória computacional (MB), para diferentes algoritmos, malha refinada com 2.048 graus de liberdade com 3 iterações internas para o *multigrid*

Algoritmo	WAMG_hD2(2)	WAMG_hD4(2)	WAMG_hD4(4)	GCP	GS
Tempo	98,1	4,19	786,1	61,9	66.475,2
Memória computacional	1.307	1.303	984	566	560
Ciclos/iterações	1.086	46	14.976	1.637	8.323.127

Tabela 6.2: Tempo de processamento (*s*), número de ciclos/iterações e memória computacional (MB), para diferentes algoritmos, malha refinada com 8.192 graus de liberdade com 3 iterações internas para o *multigrid*

Algoritmo	WAMG_hD2(2)	WAMG_hD4(2)	GCP	WAMG_hD4(4)	GS
Tempo	5.643,6	284,87	3.981,9	*	*
Memória computacional	14.804	14.819	981	-	-
Ciclos/iterações	3.920	70	6.553	-	-

*Resultado não alcançado depois de 72 *h* de processamento

A tabela 6.1 mostra que o WAMG, por meio do algoritmo WAMG_hD4(2), obteve melhor desempenho em termos de tempo de processamento, quando comparado com os demais algoritmos avaliados. A tabela mostra também a significativa diferença de tempo de processamento entre o Gauss-Seidel (GS) e WAMG e GCP. O algoritmo WAMG_hD4(4) apresentou o pior resultado em relação ao esforço computacional, ou seja, tempo de processamento e número de ciclos, embora utilizando a menor quantidade de memória computacional dentre os algoritmos de WAMG.

Observa-se na tabela 6.2 que o WAMG_hD4(2) permanece superior ao WAMG_hD2(2) e ao gradiente conjugado precondicionado (GCP) em termos de tempo de processamento quando do aumento dos graus de liberdade da malha refinada.

Adicionalmente, o caso foi também processado com o método iterativo GS com sobrerrelaxação $\omega = 1,5$ (SOR), o mais rápido dos métodos estacionários investigados neste trabalho, nas mesmas condições especificadas para a tabela 6.2. O resultado, como era de se esperar, ficou muito aquém do *multigrid* e gradiente conjugado precondicionado. Para efeito comparativo do potencial do *multigrid*, após $2,62 \times 10^{5} s$ (mais de 72 horas) de processamento, o erro de iteração do SOR era 0,173, ou seja, muito distante de se obter 10^{-5} alcançado em 284 segundos com o WAMG_hD4(2).

A tabela 6.3 mostra o tempo ótimo para este caso (malha com 8.192 pontos) para os diferentes algoritmos do WAMG.

Tabela 6.3: Tempo de processamento ótimo (*s*), número de ciclos e memória computacional (MB), para diferentes algoritmos, malha refinada com 8.192 graus de liberdade

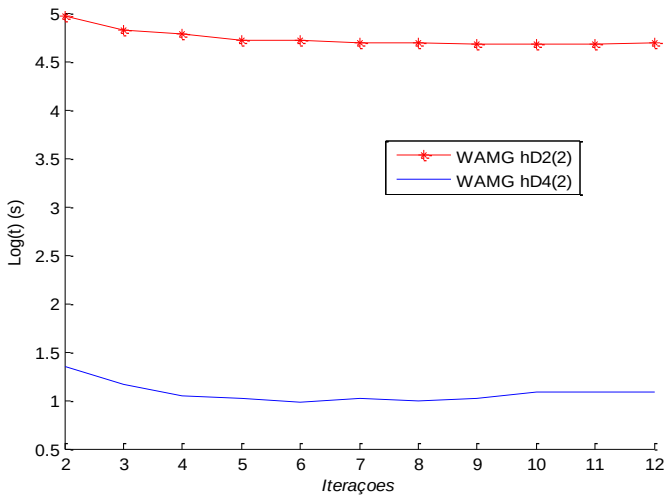
Algoritmo	WAMG_hD2(2) (3 iterações)	WAMG_hD4(2) (4 iterações)	WAMG_hD4(4)
Tempo	5.643,6	175,5	*
Memória computacional	14.804	14.819	-
Ciclos/iterações	3.920	42	-

*Resultado não alcançado depois de 72 h de processamento

Em termos de otimização, verificou-se que para este caso (malha com 8.192 pontos) o algoritmo WAMG_hD4(2) obteve o menor tempo de processamento com quatro iterações internas, 175,5 segundos, 42 ciclos. O melhor resultado para o WAMG_hD2(2) foi alcançado com três iterações internas, tempo de 5.643,6 segundos e 3.920 ciclos.

A figura 6.2 mostra o tempo (em escala logarítmica) de processamento em função das iterações para diferentes algoritmos, para uma malha com 2.048 graus de liberdade na malha mais refinada. Observa-se então que o WAMG_D4(2) obteve o menor tempo de processamento em todo o domínio do caso.

Figura 6.2: Problema da condução de calor: tempo de processamento em função do número de iterações internas para *o multigrid*



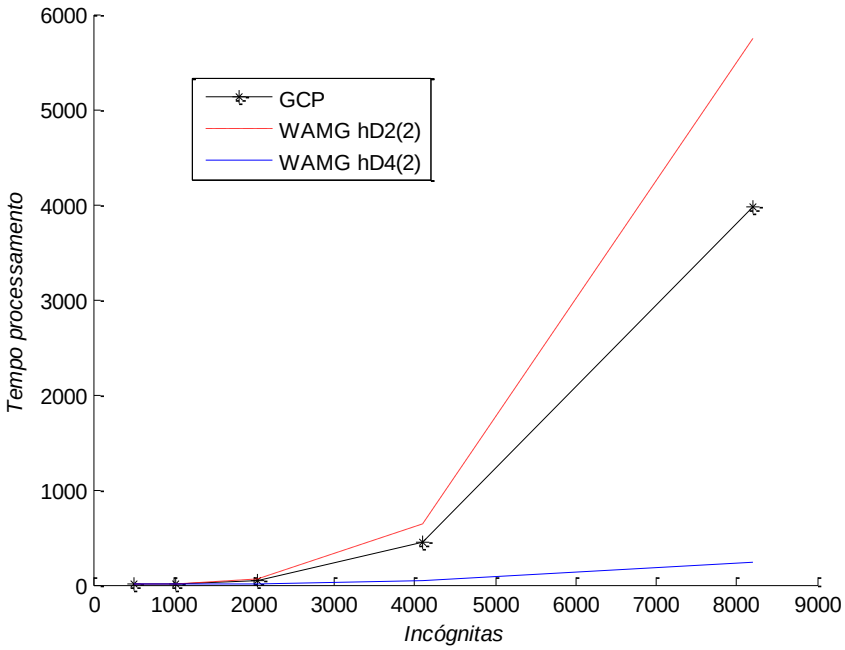
Fonte: O autor, 2013.

A seguir a figura 6.3 mostra o tempo de CPU em função do número de incógnitas para os algoritmos avaliados nesta seção, obtido por meio da equação (6.1) (PINTO; MARCHI, 2006)

$$t_{cpu} = aN^b. \tag{6.1}$$

na qual t_{cpu} é o tempo de CPU em segundos, a e b são coeficientes, este último caracteriza a ordem do método, enquanto N é o número de variáveis no sistema a ser resolvido. Observa-se que o WAMG_hD4(2) apresenta, para a amostra, a menor tendência de tempo de CPU.

Figura 6.3: Tempo de CPU em (*s*) para a equação da condução de calor



Fonte: O autor, 2013

6.1.2 Problema 2

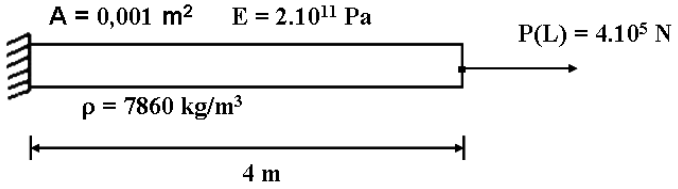
Nesta seção o problema 2, da barra, caso A e caso B são investigados. O problema é resolvido pelos métodos GS, GCP e WAMG. Foi processado considerando diversas malhas, várias tolerâncias para a norma do máximo do resíduo, variando o parâmetro ω do SOR desde 0,5 até 1,99, e explorando-se diferentes números de iterações internas para o *multigrid*. Os melhores resultados para o WAMG foram obtidos com ω = 1,99 para o SOR, conforme

será apresentado adiante. Nos resultados apresentados referentes ao problema 2, da barra, adotou-se para o critério de parada a tolerância de 10^{-5}.

6.1.2.1 Caso A

O caso A da barra é estabelecido conforme os dados da figura 6.4. O número de condição da matriz do sistema foi da ordem de 10^{11}, com autovalores espalhados entre 1 e 10^{11}. Todos os algoritmos avaliados são apresentados na Tabela 6.3 e os resultados coincidiram com o resultado analítico para o deslocamento *u* na extremidade direita da barra: 8,0 *mm*. O resultado analítico foi obtido por meio da expressão $u = (FL)/(EA)$ (ALVES FILHO, 2007). Importante afirmar que este caso é um problema de solução simples, utilizado com o objetivo de testar os algoritmos desta tese.

Figura 6.4: Barra engastada



Fonte: Adaptado de KWON, 2000.

Como no problema anterior, neste caso também o algoritmo WAMG_hD4(4) convergiu com números comparativamente muito ruins. A seguir alguns resultados são apresentados, obtidos de um sistema com 2.048 graus de liberdade na malha mais refinada, com tolerância de 10^{-5}, e três iterações internas para o *multigrid*.

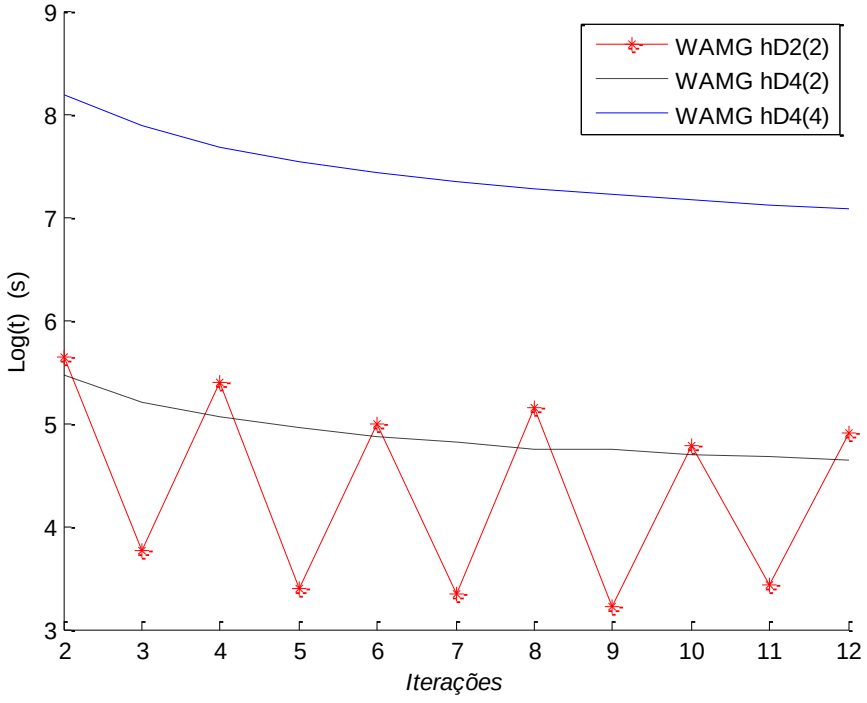
A tabela 6.4 apresenta o esforço computacional para diferentes algoritmos. Os dados mostram que, para o problema da barra à tração, o desempenho do algoritmo WAMG_hD2(2) foi superior ao GCP e a todos os outros.

Tabela 6.4: Problema da barra - Tempo de processamento (*s*), número de ciclos/iterações e memória computacional (MB), para diferentes algoritmos, 3 iterações internas para o *multigrid*

Algoritmo	WAMG_hD2(2)	WAMG_hD4(2)	WAMG_hD4(4)	GCP	GS
Tempo	42,3	177,4	2.554,8	52,3	155.849,2
Memória computacional	1.274	1.274	927	538	538
Ciclos/iterações	541	2.309	62.441	1.637	297.726.491

A figura 6.5 mostra o tempo de processamento em função do número de iterações para os algoritmos de *multigrid*. Observou-se que o tempo de processamento para o algoritmo WAMG_hD2(2) oscila em torno de valores mínimos e máximos para números de iterações ímpares e pares respectivamente.

Figura 6.5: Problema da barra: tempo de processamento em função do número de iterações internas para *o multigrid*

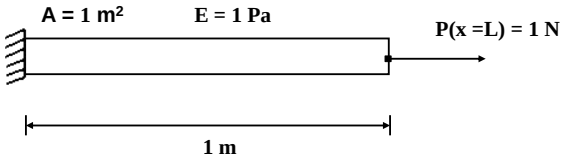


Fonte: O autor, 2013.

6.1.2.2 Caso B

Neste caso B é considerada uma barra hipotética em que suas propriedades geométricas e força aplicada são unitárias, conforme mostra a figura 6.6. O objetivo aqui é investigar o efeito do número de condição da matriz do sistema a ser resolvido, no tempo de processamento, para os algoritmos utilizados nesta seção. Neste caso o número de condição da matriz *A* do sistema está na ordem de 10^6, contra 10^{11} do caso A, item 6.1.2.1. Os dados apresentados nesta seção foram obtidos de um sistema com 2.048 nós na malha mais refinada.

Figura 6.6: Barra hipotética engastada



Fonte: Adaptado de KWON, 2000.

A tabela 6.5 mostra alguns resultados para a barra hipotética, com três iterações internas para o *multigrid*, sobrerrelaxação ω = 1,99 para o WAMG.

Tabela 6.5: Barra hipotética - tempo de processamento (*s*), para diferentes algoritmos.

Algoritmo	WAMG_hD2(2)	WAMG_hD4(2)	WAMG_hD4(4)	GCP
Tempo	16,89	66,1	816,64	50,7
Ciclos/iterações	288	1.130	25.991	1.637
Diferença (%)	46,77	51,06	58,38	0,0

A última linha da tabela 6.5 mostra a diferença percentual entre o custo computacional (número de ciclos para o *multigrid* e iterações para o GCP) do caso A, item 6.1.2.1 (tabela 6.4) e B, item 6.1.2.2, tendo como referência a tabela 6.4. Os resultados desta linha demonstram que o desempenho dos três algoritmos de *multigrid* é afetado pelo número de condição da matriz do sistema a ser resolvido, o mesmo não acontecendo com o GCP. Nota-se ainda

que os algoritmos WAMG_hD4(2) e WAMG_hD4(4) (que utilizam filtros de comprimento quatro) se revelaram mais sensíveis à condição da matriz.

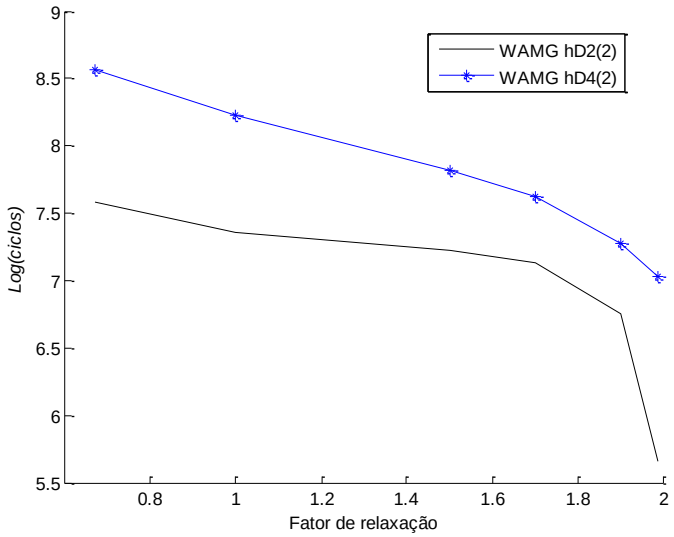
A tabela 6.6 mostra o número de ciclos para o *multigrid* para o mesmo caso B, item 6.1.2.2, apenas variando o fator de relaxação ω. Observa-se que para os dois algoritmos avaliados, o menor número de ciclos foi observado para o ω = 1,99.

Tabela 6.6: Barra hipotética – número de ciclos em função do fator ω.

Relaxação	WAMG_hD2(2)	WAMG_hD4(2)
ω = 0,67	1.972	5.263
ω = 1,00	1.571	3.739
ω = 1,50	1.367	2.493
ω = 1,70	1.255	2.048
ω = 1,90	856	1.446
ω = 1,99	288	1.130
ω = 2,00	4.395	228.651

A figura 6.7 mostra a influência do fator de relaxação ω no custo computacional dos algoritmos de WAMG para o problema da barra hipotética, ou seja, para alguns dados da tabela 6.5.

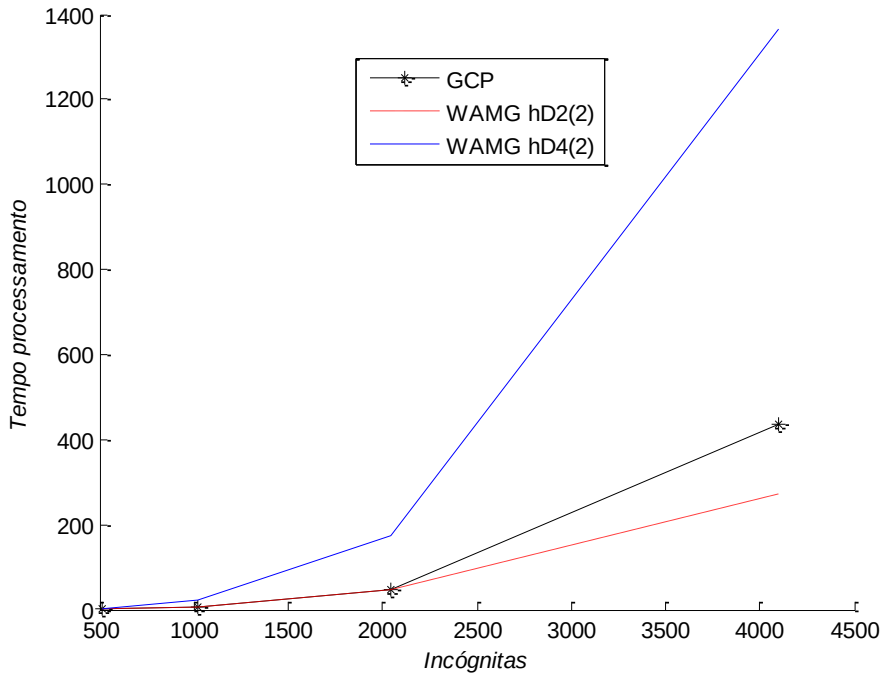
Figura 6.7: Problema da barra hipotética: influência do fator de relaxação nos algoritmos do WAMG



Fonte: O autor, 2013.

A figura 6.8 mostra a relação entre o número de incógnitas e o tempo de processamento dos algoritmos GCP, WAMG_hD2(2) e WAMG_hD4(2) para o caso desta seção. Observa-se que o algoritmo WAMG_hD2(2) obteve a menor tendência de tempo de processamento para a amostra considerada.

Figura 6.8: Tempo de CPU em (s) para o problema da barra hipotética



Fonte: O autor, 2013.

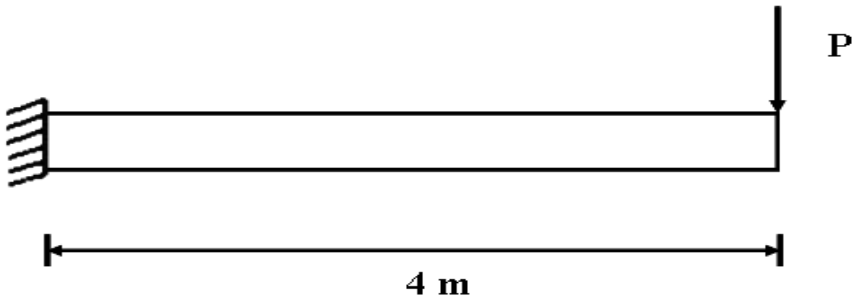
6.1.3 Problema 3

O problema 3, viga de Bernoulli foi resolvido para os casos A e B, conforme segue.

6.1.3.1 Caso A

O caso A para a viga de Bernoulli é resolvido para a condição específica da viga em balanço conforme mostra a figura 6.9 e com as seguintes propriedades: comprimento $L = 4m$, $E = 14{,}5\text{x}10^{10}\ Pa$; largura $b = 0{,}1m$, altura $h = 0{,}2m$, $P = -2.000N$. O número de condição da matriz na malha mais refinada é cond(A) = 3,4 x 10^{11}.

Figura 6.9 Viga em Balanço



Fonte: Adaptado de KWON, 2000.

O caso convergiu para a solução analítica para uma malha com 64 graus de liberdade, e uma tolerância para o erro na ordem de 10^{-1}. Testando vários valores para o suavizador SOR, como suavizador para o WAMG, percebeu-se que a sobrerrelaxação ótima foi $\omega = 2$. Para valores próximos de dois, porém maiores que dois, o resultado piora, sendo que para $\omega = 2{,}1$ os resultados já são divergentes. Os resultados do WAMG, em termos de tempo de processamento foram muito ruins quando comparados com o GCP. Para uma malha com 512 graus de liberdade em que o GCP convergiu em 4,7 segundos, o WAMG_hD4(2) demorou 2.917 segundos.

6.1.3.2 Caso B

Neste caso B o problema da viga de Bernoulli da seção anterior é processado com uma malha de 512 graus de liberdade na malha mais refinada, três iterações para o *multigrid*, e erro do máximo para o resíduo de 10^{-5}. Inicialmente o problema foi processado com as propriedades geométricas unitárias, a saber, $L = 1m$, largura $b = 1m$, altura $h = 1m$, e força aplicada igual a $0{,}1N$, mantendo-se o mesmo $E = 14{,}5\text{x}10^{10}\ Pa$. O algoritmo WAMG_hD4(2) para uma malha com 512 graus de liberdade convergiu após 1.080,6 segundos. O problema é então processado novamente, com as propriedades geométricas unitárias já citadas nesta seção e também admitindo um módulo de elasticidade unitário, $E = 1\ Pa$. A seguir a tabela 6.7 mostra alguns resultados.

Tabela 6.7: Viga hipotética - Tempo de processamento (*s*), número de ciclos/iterações e memória computacional (MB), para diferentes algoritmos

Algoritmo	WAMG_hD2(2)	WAMG_hD4(2)	WAMG_hD4(4)	GCP	GS
Tempo	225,3	141,6	527,4	4,69	12.663,4
Memória computacional	485	485	468	445	449
Ciclos/iterações	49.259	29.531	188.904	369	73.253.101

A tabela 6.7 mostra o desempenho ruim do WAMG em termos de esforço computacional para o problema da viga de Bernoulli, em comparação com o GCP.

6.2 RESULTADOS PARA OS PROBLEMAS BIDIMENSIONAIS

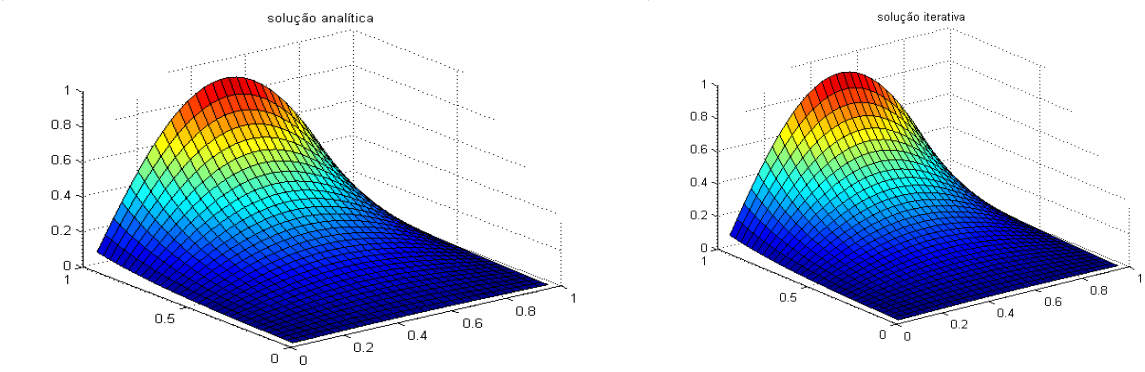
Nesta seção são apresentados os resultados para os problemas bidimensionais, ou seja, os problemas 4, 5, 6 e 7 conforme descrito na seção 5.2.

6.2.1 Problema 4

O problema 4 conforme descrito na seção 5.2.1 é processado com a tolerância do erro de 10^{-10} para o resíduo considerando a norma do erro do máximo.

A figura 6.10 mostra o gráfico da solução analítica (esquerda) e via WAMG (direita) com filtro de comprimento 4 decimação por 2, WAMG_hD4(2), para uma malha com 1.024 nós (32 x 32) e três iterações internas, confirmando qualitativamente a boa aproximação numérica com a solução analítica.

Figura 6.10: Soluções analítica e Iterativa para a equação de Laplace bidimensional com condições de contorno de Dirichlet

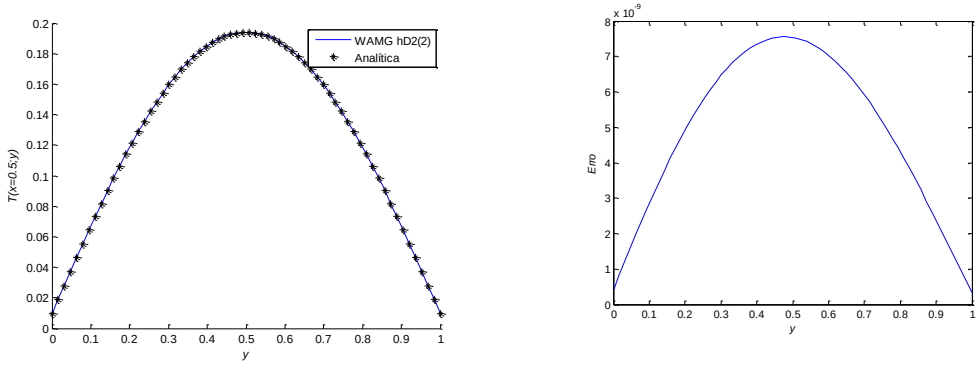


Fonte: O autor, 2013.

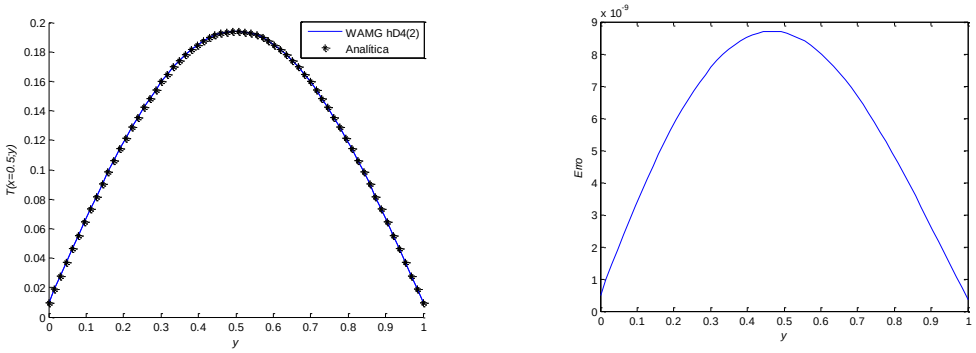
Os resultados da figura 6.11 são apresentados para uma malha com 4.096 pontos e três iterações internas para o *multigrid*.

A figura 6.11 mostra a comparação dos resultados numéricos para os diferentes algoritmos e a solução analítica, para $x = 0{,}5$ e $0 \leq y \leq 1$. As figuras 6(a, c, e) apresentam os perfis de solução analítica e numérica, enquanto as figuras 6(b, d, f) ilustram os perfis do erro, sendo este a diferença entre a solução analítica e a solução numérica obtida pelos algoritmos WAMG_hD2(2), WAMG_D4(2) e WAMG_D4(4), respectivamente.

Figura 6.11: Perfis de Solução e Erro

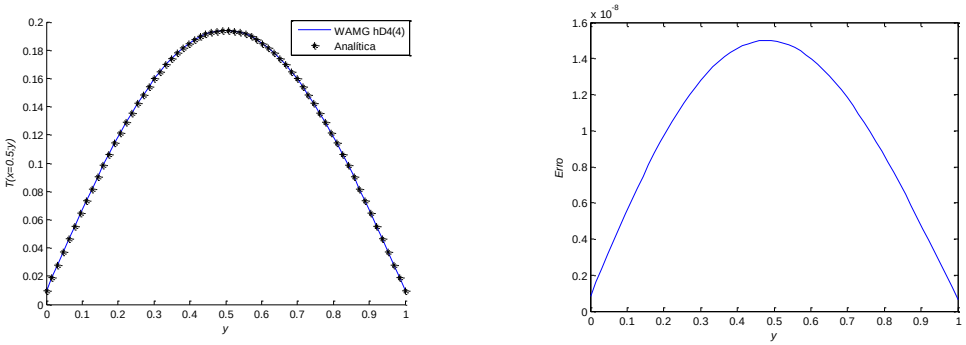


(a) (b)



(c)

(d)



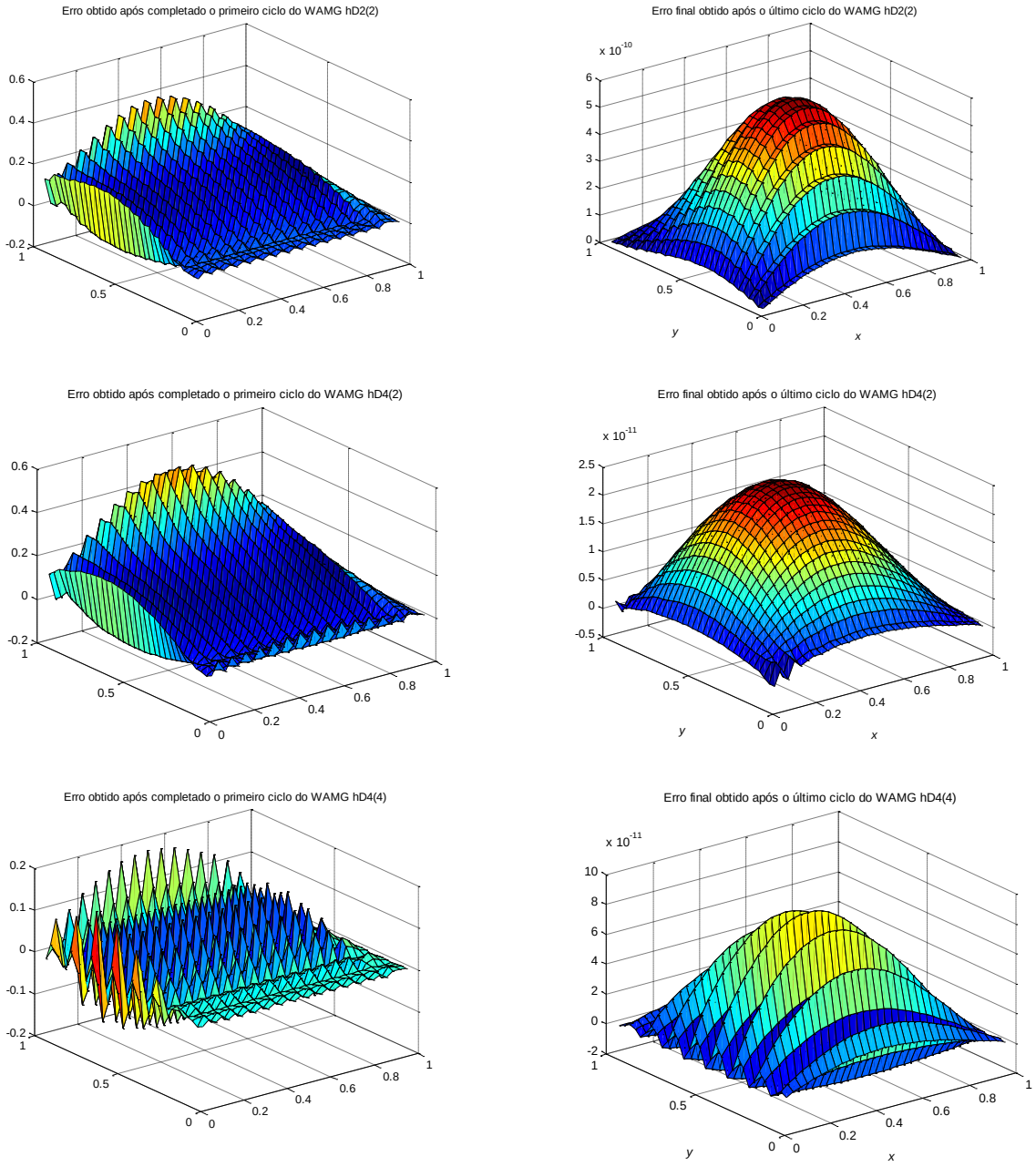
(e)

(f)

Obs.: (a, c, e) Perfis de solução e (b, d, f) Perfis de erro entre a solução analítica e numérica para WAMG_hD2(2), WAMG_D4(2) e WAMG_D4(4), respectivamente.
Fonte: O autor, 2013.

A figura 6.12 mostra o erro de iteração obtido depois de completado o primeiro ciclo, à esquerda; e o último ciclo, à direita; para os algoritmos WAMG_hD2(2), WAMG_hD4(2) e WAMG_hD4(4); para uma malha com 1.024 nós, com três iterações internas. Observa-se que o erro obtido no último ciclo, por meio do WAMG com filtro de comprimento 4 e decimação por 2, é geometricamente mais suave que os outros dois algoritmos.

Figura 6.12: Erros de iteração



Fonte: O autor, 2013

A tabela 6.8 mostra a memória computacional, o tempo de processamento, o número de iterações para o GCP e GS, e ciclos para o WAMG, para atender a tolerância de 10^{-10} para a norma do resíduo, com três iterações internas para os diferentes algoritmos, para uma malha com 8.192 nós, (128 x 64).

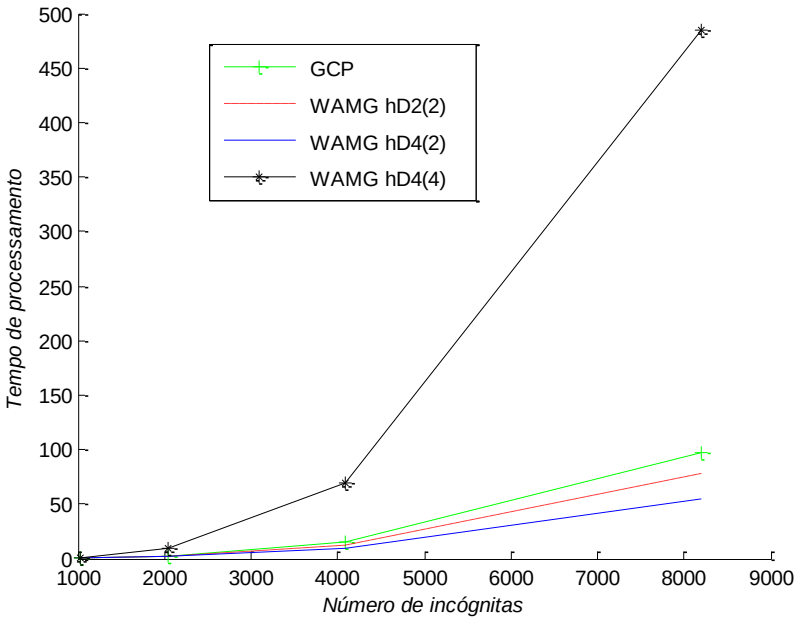
Tabela 6.8: Equação de Laplace bidimensional - Tempo de processamento (*s*), número de ciclos/iterações e memória computacional (MB), malha com 8.192 nós, para diferentes algoritmos

Algoritmo	WAMG_hD2(2)	WAMG_hD4(2)	WAMG_hD4(4)	GCP	GS
Tempo (*s*)	86,2	64,0	472,0	136,4	1.011,0
Memória computacional	14.110	14.077	7.923	540	545
Iterações/ciclos	61	45	1.116	311	11.040

Assim como ocorreu no problema 6.1.1, aqui também o algoritmo WAMG_hD4(2) teve desempenho superior ao WAMG_hD2(2). Observa-se que os algoritmos WAMG_hD2(2) e WAMG_hD4(2) obtiveram menores tempo de processamento comparados com o GCP. Como era de se esperar, a tabela mostra ainda que o WAMG demanda mais memória computacional que o GCP e GS.

A figura 6.13 mostra o tempo de processamento em função do número de incógnitas para os algoritmos de WAMG e GCP para o caso da equação de Laplace. Observa-se que para a amostra considerada, o algoritmo WAMG_hD4(2) apresenta a menor tendência de tempo de CPU.

Figura 6.13: Tempo de CPU em (*s*) para a equação de Laplace bidimensional



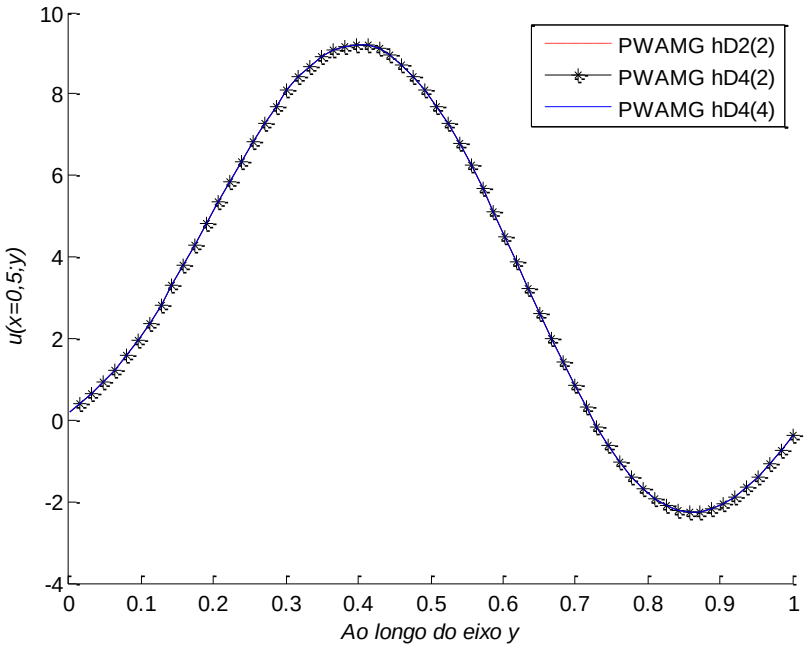
Fonte: O autor, 2013

6.2.2 Problema 5

Nesta seção o problema envolvendo difusão e advecção é avaliado. Como se trata de um problema não simétrico, o método do GCP não se aplica. A seguir apresentam-se resultados para malha com 4.096 nós (64 x 64), λ = 100, σ = 10, tolerância para norma do máximo para o resíduo de 10^{-10}, e três iterações internas para o WAMG.

A figura 6.14 mostra a solução da Equação (5.10) em x = 0,5, para os diferentes algoritmos. Graficamente as três soluções são coincidentes.

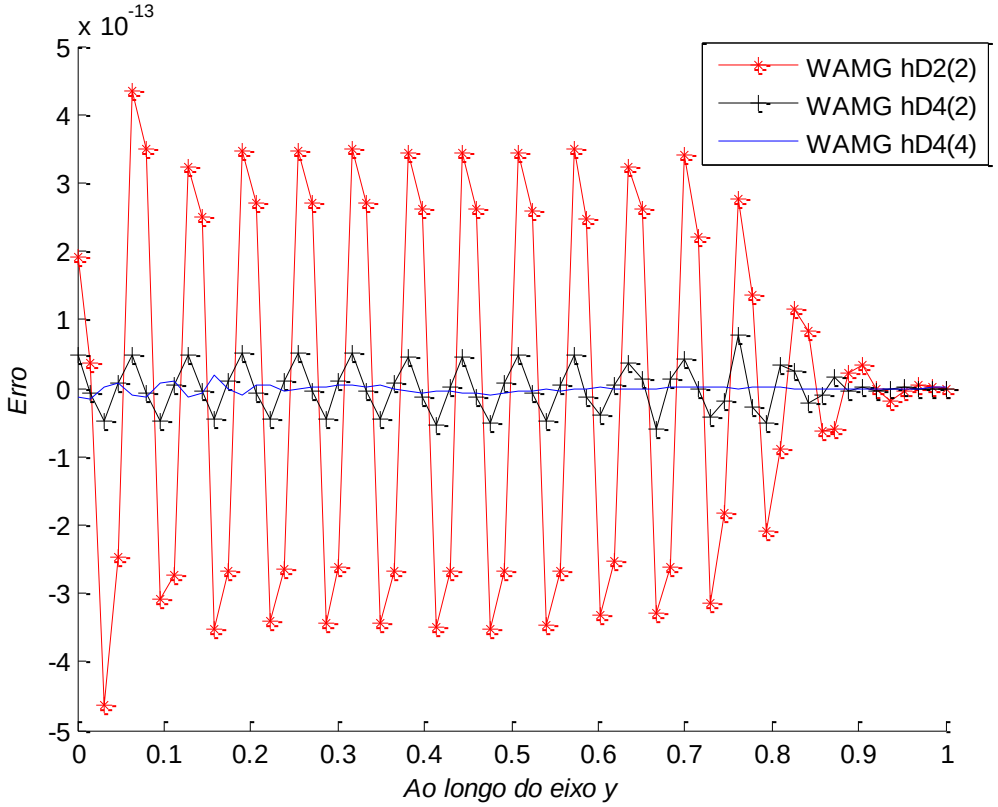
Figura 6.14: Perfil de soluções para diferentes algoritmos



Fonte: O autor, 2013.

A figura 6.15 mostra a diferença entre a solução obtida pelos algoritmos investigados nesta seção e a solução direta do problema via MATLAB, que foi tomada como referência, no perfil da figura 6.14. Observa-se que a variabilidade é maior para o WAMG hD2(2), e está na ordem de 10^{-13}.

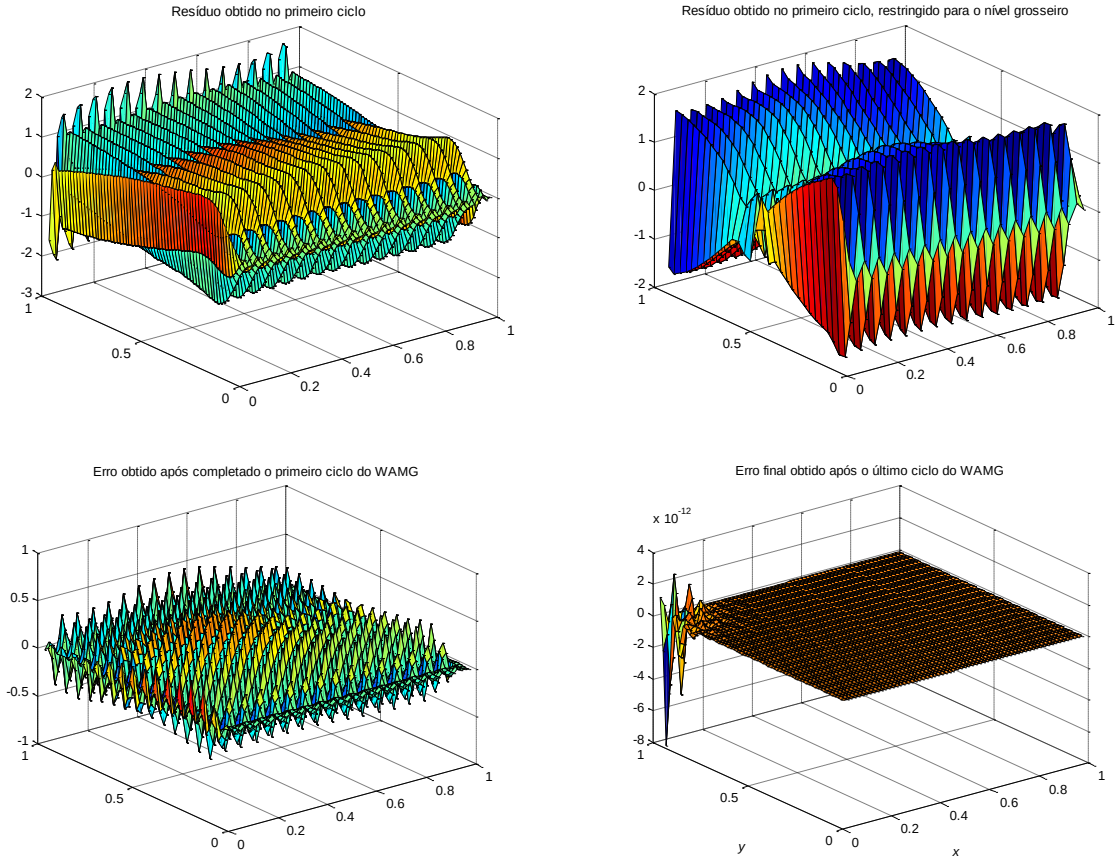
Figura 6.15: Diferença entre solução direta via MATLAB e diferentes algoritmos avaliada no perfil da figura 6.14.



Fonte: O autor, 2013.

A figura 6.16 mostra erros e resíduos para o algoritmo WAMG_hD4(4) para o caso da advecção difusão. Como pode ser observado, o erro no último ciclo é praticamente nulo, exceto próximo a um contorno.

Figura 6.16: Erros e resíduos obtidos com o algoritmo WAMG_hD4(4) para o caso envolvendo advecção.



Fonte: O autor, 2013

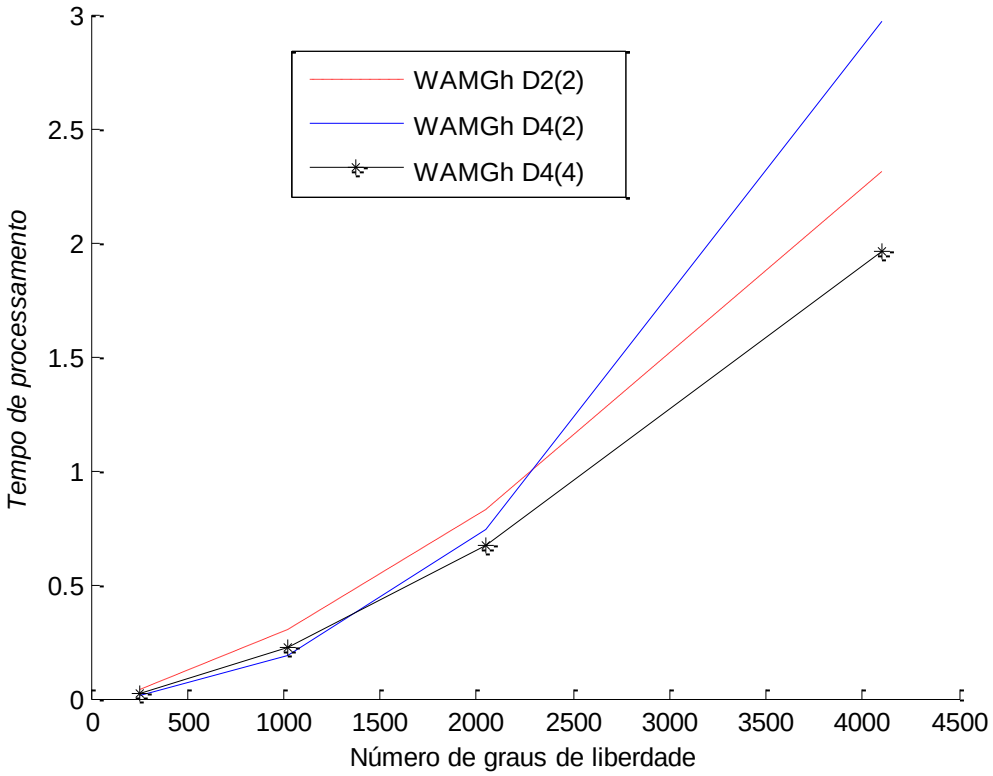
A tabela 6.9 mostra o tempo de processamento, o número de ciclos para o WAMG e o número de iterações para GS, e a memória computacional para se atingir a tolerância especificada no experimento para os diferentes algoritmos. Nota-se que o WAMG implementado com filtro de Daubechies comprimento 4 foi superior ao WAMG via filtro de Haar de comprimento 2 em termos de menor tempo de processamento e menor memória computacional.

Tabela 6.9: Tempo de processamento (s), número de ciclos/iterações e memória computacional (MB), malha com 4.096 nós (64 x 64), para diferentes algoritmos

Algoritmo	WAMG_hD2(2)	WAMG_hD4(2)	WAMG_hD4(4)	GS
Tempo	2,96	2,81	2,68	8,01
Memória computacional	3.737	3.748	2.208	601
Ciclos/iterações	14	13	24	606

A figura 6.17 mostra o tempo de processamento em função do número de incógnitas para os algoritmos do WAMG para o caso da advecção difusão. Observa-se que para a amostra considerada, o algoritmo WAMG_hD4(4) apresenta o menor tempo de CPU.

Figura 6.17: Tempo de CPU em (*s*) para o caso envolvendo advecção difusão



Fonte: O autor, 2013

6.2.3 Problema 6

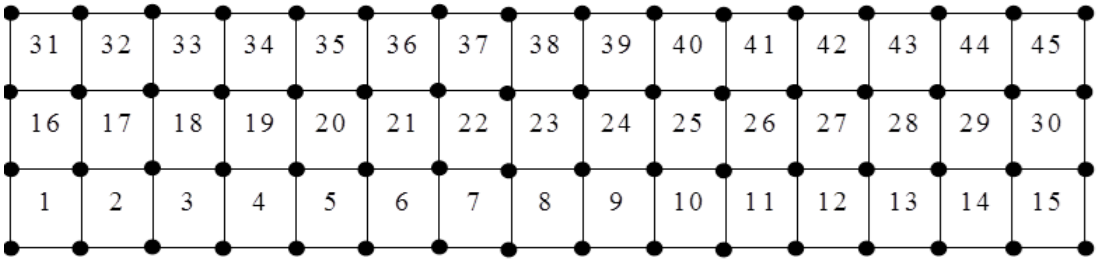
Nesta seção são apresentados os resultados do problema 6, conforme descrito na seção 5.2.3. Cabe ressaltar que este problema foi desdobrado em três casos, A, B e C.

6.2.3.1 Caso A

Neste caso o problema foi resolvido com os seguintes dados: força P = 4.448 N, módulo de elasticidade E = 6,895x10^9 Pa, coeficiente de Poisson ν = 0,3, comprimento L = 101,6 mm, altura h = 25,4 mm, espessura t = 25,4 mm.

O resultado numérico convergiu para o resultado analítico (deslocamento na extremidade direita da viga: 6,5 mm) com uma malha de 128 graus de liberdade (malha 16 x 4 nós conforme mostra a figura 6.18), e uma tolerância para o erro do máximo do resíduo de 10^{-7}. O resultado analítico foi obtido pela expressão: $PL^3/3EI$. Assim como aconteceu no problema 2, para este também os melhores resultados foram obtidos com ω = 1,99 para o SOR utilizado como relaxador no WAMG.

Figura 6.18: Malha bidimensional ilustrativa, com 45 elementos, 64 nós, 128 graus de liberdade



Fonte: o autor, 2013.

A tabela 6.10 apresenta alguns resultados para o caso da viga em balanço sob estado plano de tensões, com erro do máximo de 10^{-5} para o resíduo, para uma malha com 2.048 graus de liberdade (128 x 8 nós), com três iterações internas para o *multigrid*.

Tabela 6.10: Tempo de processamento (*s*), número de ciclos/iterações e memória computacional (MB), problema da viga em balanço, diferentes algoritmos.

Algoritmo	GCP	WAMG_hD2(2)	WAMG_hD4(2)	WAMG_hD4(4)	GS
Tempo	16,27	194,38	230,90	53,27	2.290,6
Memória computacional	1.035	1.357	1.356	1.034	1.035
Ciclos/iterações	617	3.528	4.199	1.902	1.113.729

A tabela 6.10 mostra que o WAMG_hD4(4) apresentou melhor resultado dentre os algoritmos de WAMG, ainda assim, o tempo de processamento foi maior que aquele obtido pelo GCP.

6.2.3.2 Caso B

Nesta seção o problema da viga bidimensional da seção anterior é investigado, mudando-se o carregamento para horizontal. O caso foi inicialmente processado com 2.048 graus de liberdade na malha mais refinada, malha com 128 x 8 nós nas direções do comprimento e largura da viga respectivamente. O WAMG é processado com três iterações internas, relaxação SOR com $\omega = 1,99$.

A tabela 6.11 mostra os resultados para os quais apenas a direção do carregamento é modificada, mantendo-se as demais condições da seção anterior. O número de ciclos e o tempo de processamento para o WAMG diminuíram em comparação com os dados da tabela anterior. A última linha mostra o quociente entre o tempo de processamento da tabela 6.10 e o tempo de processamento da tabela 6.11.

Tabela 6.11: Tempo de processamento (*s*), número de ciclos/iterações e memória computacional (MB), problema da viga com carregamento horizontal

Algoritmo	GCP	WAMG_hD2(2)	WAMG_hD4(2)	WAMG_hD4(4)	GS
Tempo	11,75	142,54	170,90	29,92	2.459,54
Ciclos/ iterações	615	2.627	3.115	1.069	1.149.996
Memória computacional	1.313	1.357	1.356	1.035	1.034
Vantagem	1,39	1,36	1,35	1,78	0,93

6.2.3.3 Caso C

Nesta seção a viga em balanço com carregamento vertical é processada com módulo de elasticidade e força unitária, mantendo-se as demais condições do caso B item 6.2.3.1, inclusive com a malha mais refinada com 2.048 graus de liberdade (128 x 8 nós). A tabela 6.12 mostra alguns resultados. A última linha da referida tabela mostra o quociente entre o tempo de processamento da tabela 6.10 e da tabela 6.12. Os resultados mostram que as alterações já elencadas afetaram o tempo de processamento para todos os algoritmos pesquisados.

Tabela 6.12: Tempo (*s*), número de ciclos/iterações e memória computacional (MB) problema da viga em balanço, carregamento unitário

Algoritmo	GCP	WAMG_hD2(2)	WAMG_hD4(2)	WAMG_hD4(4)	GS
Tempo	9,83	118,47	150,38	35,15	1.031,91
Memória computacional	1.035	1.357	1.358	1.037	1.038
Ciclos/ iterações	489	2.165	2.720	1.245	495.436
Vantagem	1,66	1,64	1,54	1,52	2,22

6.2.4 Problema 7

Este problema avalia o caso da seção 6.2.3.2, mudando apenas as condições de contorno, qual seja, viga bidimensional engastada a esquerda, abaixo e acima e livre no outro lado.

A tabela 6.13 mostra os resultados obtidos para o caso.

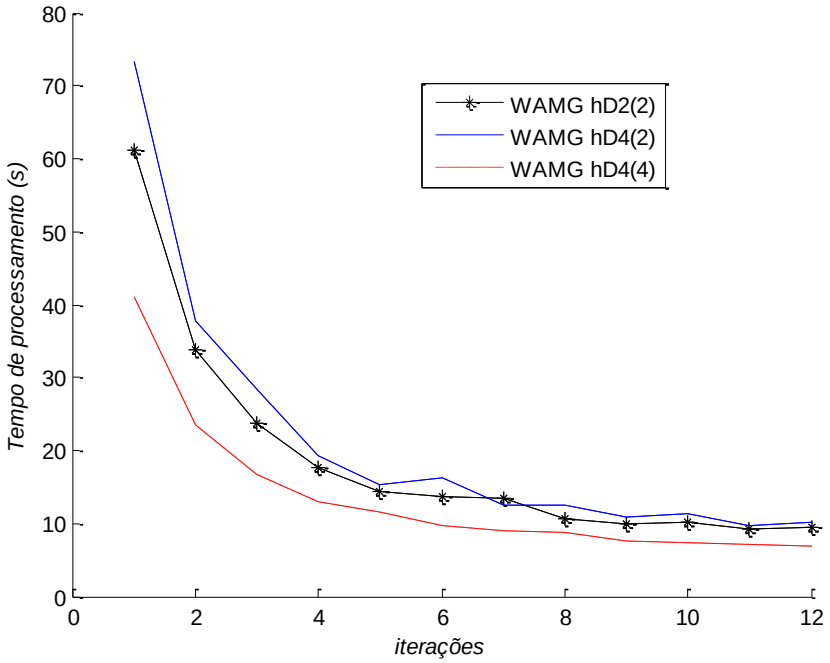
Tabela 6.13: Tempo (*s*), número de ciclos/iterações e memória computacional (MB) problema da viga engastada, carregamento longitudinal

Algoritmo	GCP	WAMG_hD2(2)	WAMG_hD4(2)	WAMG_hD4(4)	GS
Tempo (*s*)	9,01	17,63	21,12	12,59	185,58
Ciclos/ iterações	454	322	387	458	120.597
Memória computacional	1.351	1.347	1.348	1.029	1.029
Vantagem	1,81	11,03	10,93	4,23	12,34

A tabela 6.13 mostra os tempos de processamento, a memória computacional, o número de ciclos para o WAMG e número de iterações para o GCP e GS. A última linha desta tabela mostra o quociente entre o tempo de processamento da tabela 6.10 e o tempo de processamento da tabela 6.13. Como pode ser observado, todos os algoritmos se mostraram sensíveis em relação à direção do carregamento e condições de contorno de uma estrutura mecânica idêntica, com mesma equação governante e mesma discretização. Sendo que para o WAMG o fator foi mais acentuado em comparação com o GCP, chegando a ser mais de dez vezes para os algoritmos WAMG_hD2(2) e WAMG_hD4(2).

A figura 6.19 mostra a relação entre número de iterações internas e o tempo de processamento para os algoritmos de WAMG analisados nesta seção. Esta figura evidencia a tendência no desempenho dos três algoritmos de WAMG para o caso. Observa-se que até cinco iterações internas o tempo de processamento diminui rapidamente para os três algoritmos.

Figura 6.19: Tempo de processamento em função do número de iterações



Fonte: O autor, 2013.

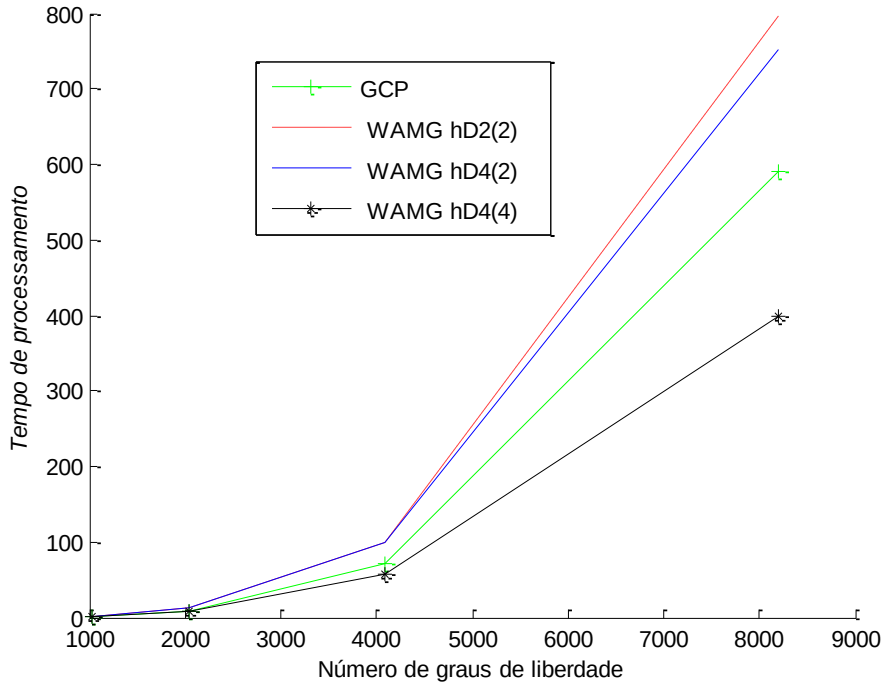
A tabela 6.14 mostra o tempo de processamento, em segundos, com sete iterações internas para o WAMG. Esta tabela mostra que neste caso os resultados do WAMG ficaram mais próximos do GCP, inclusive com o tempo do algoritmo WAMG_hD4(4) inferior ao GCP para algumas malhas.

Tabela 6.14: Tempo (*s*) caso da viga engastada, carregamento longitudinal

Graus de liberdade	GCP	WAMG_hD2(2)	WAMG_hD4(2)	WAMG_hD4(4)
1.024	0,97	1,81	2,40	1,59
2.048	9,1	9,73	9,19	6,76
4.096	72,4	91,90	78,38	32,04
8.192	569,7	901,75	994,42	619,9

A figura 6.20 mostra o tempo de processamento em função do número de graus de liberdade para os dados da tabela 6.14. Observa-se que para a amostra considerada, o algoritmo WAMG_hD4(4) apresenta a menor tendência de tempo CPU, fato já verificado para o problema 5.

Figura 6.20: Tempo de CPU em (*s*) para o caso da viga bidimensional



Fonte: O autor, 2013

6.3 CONCLUSÕES DO CAPÍTULO

Esta seção traz as conclusões do capítulo separadamente para os casos unidimensionais e casos bidimensionais.

A principal conclusão da seção 6.1, a partir dos casos processados, é que o WAMG é sensível ao número de condição da matriz do sistema a ser resolvido, e que utilizando-se filtros de comprimento quatro esta sensibilidade é mais acentuada, em comparação com o WAMG com filtro de Haar, comprimento dois. Os resultados obtidos nesta seção levam também a conclusão de que o *multigrid* algébrico via wavelets WAMG é preferível em comparação com o gradiente conjugado precondicionado GCP quando o problema possui apenas um grau de liberdade por nó, como nos casos da transferência de calor e da barra.

No entanto, quando o elemento possui mais de um grau de liberdade em cada nó e estes graus de liberdade são de naturezas distintas, como nos casos da viga de Bernoulli, em que num mesmo nó tem-se um deslocamento e uma rotação, cujas magnitudes são distintas; o desempenho do WAMG se deteriora. A seção 6.1.3 mostra que alterando a rigidez a flexão da viga artificiosamente, o desempenho do WAMG em termos de tempo de processamento melhora, ainda assim, com resultados muito ruins comparados com o GCP.

A seguir as principais conclusões em relação aos problemas bidimensionais da seção 6.2.

A partir dos resultados do caso 6.2.1, pode-se concluir que o WAMG (algoritmos WAMG_hD2(2) WAMG_hD4(2)), são mais vantajosos quando comparados com o GCP. Para o caso 6.2.2, a vantagem do WAMG é ainda mais significativa, por se tratar de um problema não simétrico e, portanto, não convergente com o GCP.

Quanto aos casos estruturais, observou-se que o WAMG é sensível às propriedades mecânicas da estrutura, à direção do carregamento, bem como às condições de contorno, como pode ser observado pelos resultados do caso 6.2.3.3. No caso da mudança das condições de contorno, a variação no tempo

de processamento foi mais significativa que a mudança ocorrida quando se variou a direção do carregamento ou de propriedade mecânica, tendo como base de comparação o GCP.

Outra conclusão que se chega observando os resultados bidimensionais investigados é que o tempo de processamento utilizando filtro de comprimento 4, é menor que aqueles obtidos com filtro de comprimento 2 para o mesmo caso. Isto contradiz afirmações encontradas na literatura de que filtros de Haar, de menor comprimento seriam mais vantajosos (GARCIA et al., 2008).

7 PRECONDICIONADOR WAMG

Neste capítulo um ciclo do WAMG é utilizado como precondicionador do gradiente conjugado precondicionado GCP, conforme algoritmo 4.5. Resultados numéricos são apresentados para o caso bidimensional, mais precisamente, os problemas 4, 5 e 6, ou seja, equação de Laplace bidimensional, problema da advecção, e viga bidimensional sob estado plano tensões. Os itens avaliados são tempo de processamento, memória computacional, número de ciclos para o PWAMG, e número de iterações para GCP e GS. O tempo de processamento do PWAMG é comparado com o do GCP. A seguir um resumo de como a convergência da solução de um sistema de equações resolvido iterativamente é afetada pelo número de condições da matriz A do sistema a ser resolvido.

A convergência dos métodos iterativos é afetada tanto pelo tamanho quanto pelo mal condicionamento das matrizes. O condicionamento de uma matriz é aferido pela distribuição de seus autovalores, sendo que uma matriz cujos autovalores estão espalhados num intervalo muito grande ou próximo de zero, é dita mal condicionada. O número de condição de uma matriz não singular A relativo à norma $\|.\|$ conforme apresentada no capítulo 2 é dado por

$$\text{Cond}(A) = \|A\| . \|A^{-1}\|, \ \text{Cond}(A) \geq 1. \tag{7.1}$$

Uma matriz A é bem condicionada se cond(A) estiver próximo da unidade. Caso cond(A) seja significativamente maior que 1, A é dita mal condicionada. O termo condição se refere à segurança relativa com que um vetor residual **r** pequeno implica um erro **e** também pequeno. Simbolicamente,

$$\text{Cond}(A) \approx 1 \rightarrow \mathbf{r} \approx \mathbf{0} \text{ e } \mathbf{e} \approx \mathbf{0}. \tag{7.2}$$

Matematicamente a melhora no condicionamento de uma matriz é expressa por meio de uma transformação linear M no sistema determinado pela Eq. 2.1

$$MA\boldsymbol{u} = M\mathbf{b} \tag{7.3}$$

Na qual M é a matriz do precondicionador.

Neste trabalho o precondicionamento da matriz A é efetuado implicitamente por meio de um ciclo V do WAMG, e a solução iterativa do sistema é então encontrada por meio do algoritmo 4.5.

7.1 REANÁLISE DO PROBLEMA 4

Neste caso, o problema é resolvido nas condições especificadas na seção 5.2.1, e já avaliado na seção 6.2.1, sendo que aqui o ciclo do WAMG é utilizado como o precondicionador (PWAMG) do gradiente conjugado. O número de condição da matriz na malha mais refinada é cond(A) = 3,83 x 10^3.

A tabela 7.1 mostra o tempo de processamento para os algoritmos avaliados neste caso com diferentes malhas, tolerância de 10^{-10}, norma do máximo para o resíduo, três iterações internas para o *multigrid*.

Tabela 7.1: Tempo (s) de processamento para diferentes algoritmos

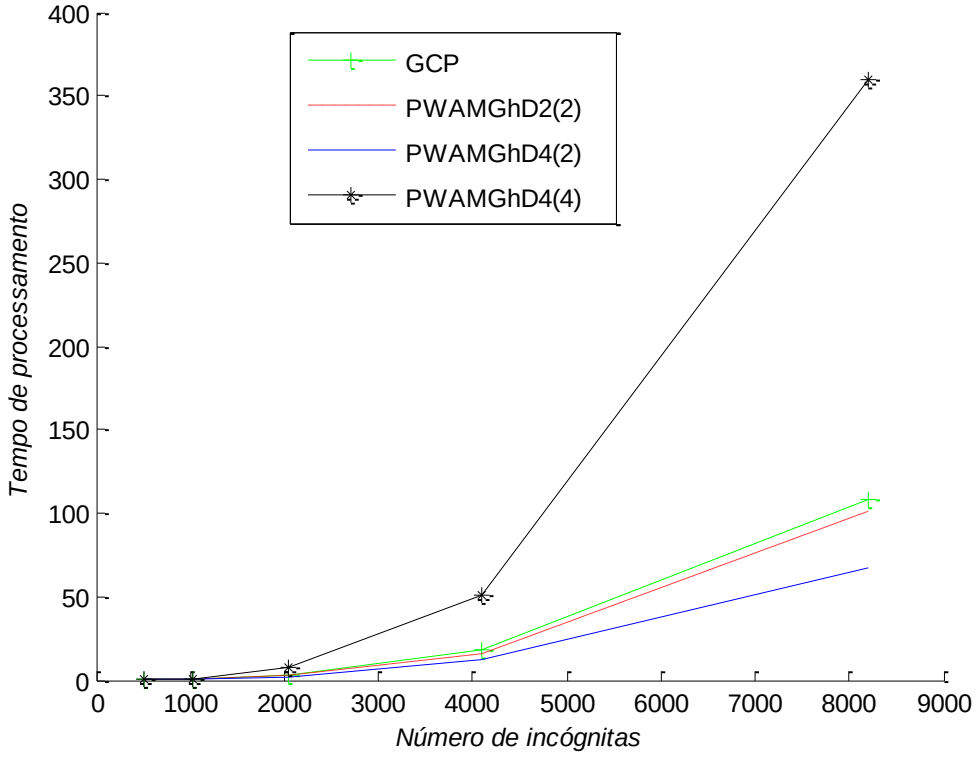
Malha (nós)	GCP	PWAMG_hD2(2)	PWAMG_hD4(2)	PWAMG_hD4(4)
512	0,12	0,08	0,11	0,16
1.024	0,38	0,48	0,30	1,09
2.048	3,29	1,56	1,19	6,20
4.096	14,04	10,81	7,90	55,88
8.192	136,2	167,14	127,14	363,82

A tabela 7.1 mostra o PWAMG com menor tempo de processamento que o GCP para o algoritmo PWAMG_hD4(2) para todas as malhas. Comparando os tempos de processamento da tabela 6.8 e 7.1, observa-se que o PWAMG_hD4(4) obteve uma redução do tempo quando utilizado como

precondicionador para o GCP, ainda assim, com maior tempo dentre os algoritmos avaliados. O PWAMG_hD2(2) teve aumento de tempo em relação ao WAMG_hD2(2).

A figura 7.1 mostra o tempo de processamento em função do número de incógnitas para os algoritmos avaliados neste caso. Observa-se que o algoritmo WAMG_hD4(2) obteve a menor tendência de tempo de processamento.

Figura 7.1: Tempo de CPU para o PWAMG aplicado à equação de Laplace



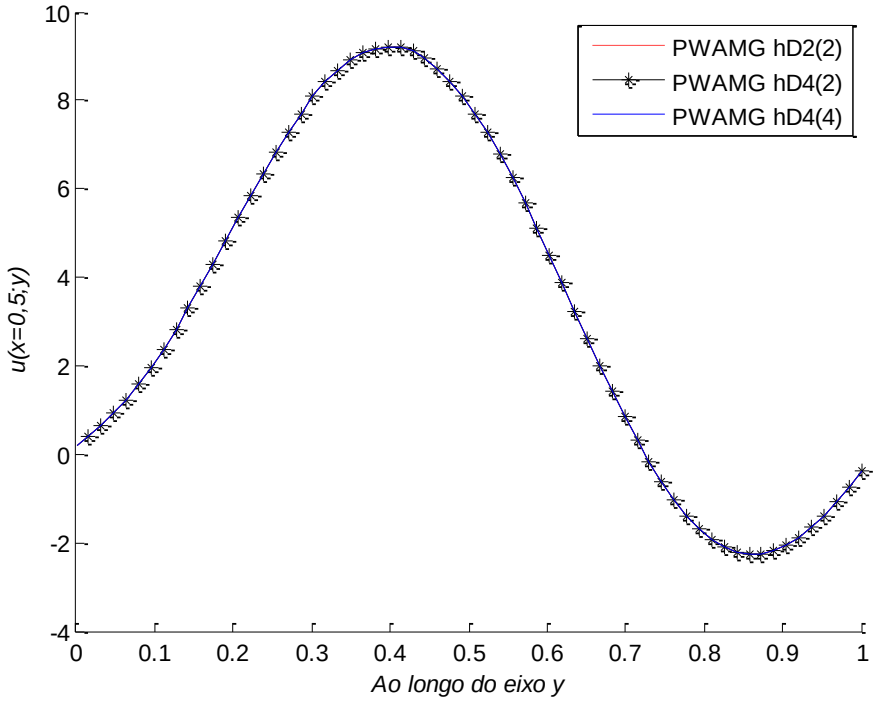
Fonte: O autor, 2013

7.2 REANÁLISE DO PROBLEMA 5

Nesta seção o problema de advecção difusão, descrito na seção 5.2.2 e avaliado na seção 6.2.2 é investigado, mas agora o PWAMG é utilizado. O número de condição da matriz na malha mais refinada é cond(A) = 4,1 x 10^2.

A figura 7.2 mostra a solução do caso em x = 0,5, para os diferentes algoritmos. Assim como ocorreu com a figura 6.14, aqui também visualmente as três soluções são coincidentes.

Figura 7.2: Perfil de soluções para diferentes algoritmos do PWAMG



Fonte: O autor, 2013.

A tabela 7.2 mostra o tempo de processamento, número de ciclos/iterações e memória computacional para se atingir a tolerância especificada no caso com a malha mais refinada contendo 4.096 pontos e três iterações internas para o *multigrid*. Nota-se que o PWAMG_hD4(4) divergiu. Para este caso da advecção, o desempenho do PWAMG foi inferior ao WAMG (tabela 6.9) em relação ao tempo de processamento. Em relação ao PWAMG_hD4(4), o mesmo apresentou convergência a partir de sete iterações internas, com um tempo de 2,92 segundos, e treze ciclos.

Tabela 7.2: Tempo (*s*), número de ciclos/iterações e memória computacional (MB), para diferentes algoritmos com 3 iterações internas para o multigrid com 4.096 nós na malha mais refinada (64 x 64)

Algoritmo	PWAMG_hD2(2)	PWAMG_hD4(2)	PWAMG_hD4(4)	GS
Tempo	4,3	4,3	*	6,93
Memória computacional	3.703	3.715		578
Ciclos/iterações	13	13		906

*divergiu com 3 iterações internas

A tabela 7.3 mostra o número de iterações internas para se obter o menor tempo de processamento do caso para cada um dos três algoritmos, quando o PWAMG é utilizado.

Tabela 7.3: Tempo de processamento (*s*) ótimo, para diferentes algoritmos.

Algoritmo	PWAMG_hD2(2)	PWAMG_hD4(2)	PWAMG_hD4(4)
Tempo Ótimo	2,6	2,8	1,7
Ciclos(iterações)	4(11)	8(5)	6(11)

Nota-se pela tabela 7.3 que apesar do PWAMG_hD4(4) utilizado como precondicionador ter divergido para 3 iterações internas, com 11 iterações internas, obteve menor tempo de processamento que os outros dois algoritmos.

7.3 REANÁLISE DO PROBLEMA 6

Nesta seção o problema da viga bidimensional sob estado plano de tensões conforme descrito na seção 5.2.3, e avaliado na seção 6.2.3 volta a ser analisado, agora com o ciclo V do WAMG utilizado como o precondicionador do método GCP. O número de condição da matriz na malha mais refinada é cond(A) = 2,53 x 10^6. A tabela 7.4 mostra os resultados para as mesmas condições especificadas no caso B, seção 6.2.3.2.

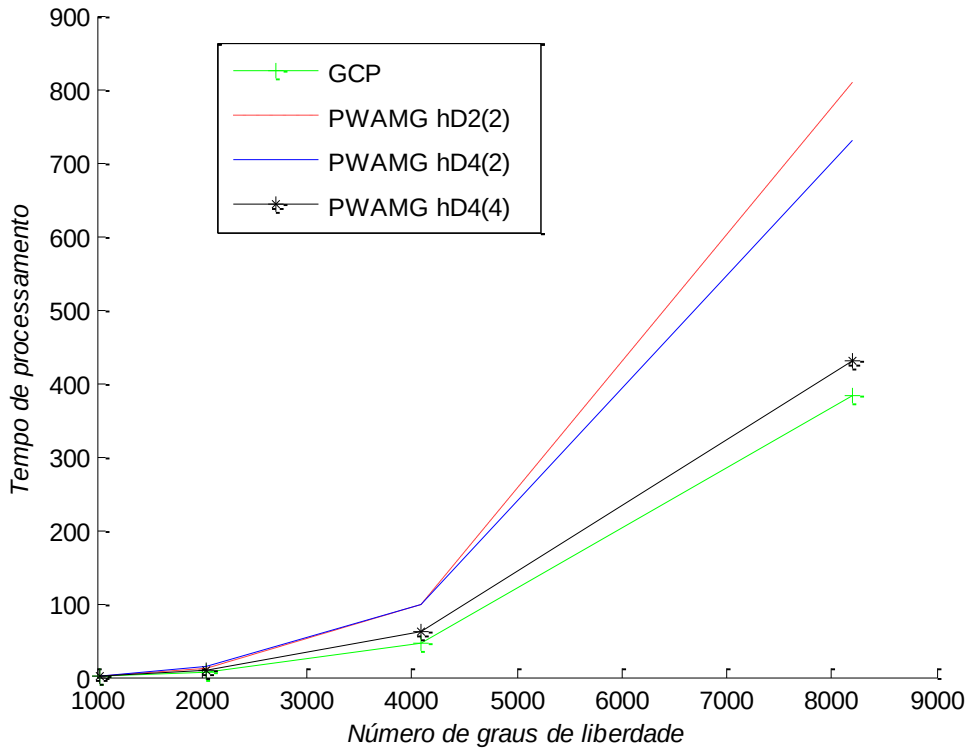
Tabela 7.4: Tempo (*s*), número de ciclos/iterações e memória computacional (MB) PWAMG, viga bidimensional, 3 iterações internas para o *multigrid*

Algoritmo	GCP	PWAMG_hD2(2)	PWAMG_hD4(2)	PWAMG_hD4(4)
Tempo	11,75	194,59	234,55	40,76
Memória computacional	1.313	1.314	1.314	996
Ciclos/iterações	615	2.551	3.061	1.059

Comparando-se os resultados da tabela 7.4 com a 6.11, observa-se que nenhum algoritmo de PWAMG melhorou seu desempenho quando utilizado como precondicionador para o caso da viga bidimensional.

A figura 7.3 mostra o tempo de processamento para os diferentes algoritmos avaliados nesta seção para o problema da viga bidimensional nas mesmas condições do problema 6, caso *C*, seção 6.2.3.3, exceto que aqui um ciclo do WAMG é utilizado como precondicionador. Observa-se que para este caso o GCP obteve a menor tendência de tempo de processamento.

Figura 7.3: Tempo de CPU para o PWAMG aplicado ao problema da viga bidimensional



Fonte: O autor, 2013

7.4 CONCLUSÕES DO CAPÍTULO

Nesta seção apresentam-se as principais conclusões dos casos deste capítulo.

Primeiramente registra-se o fato de que o WAMG utilizado como precondicionador (PWAMG) obteve convergência para todos os casos avaliados neste capítulo, inclusive para o problema da advecção, que não é convergente com o GCP porque a matriz do sistema a ser resolvida não é simétrica. Ou seja, quando um ciclo do WAMG foi utilizado como precondicionador para o GCP, este (PWAMG) obteve convergência para um problema não convergente com o GCP. Ainda em relação ao problema da advecção, seção 7.2, algo intrigante aconteceu, a saber. O algoritmo WAMG_hD4(4) divergiu quando processado com três iterações internas, e convergiu com menor tempo quando processado com onze iterações internas.

Para o caso 7.1, observou-se que o algoritmo WAMG_hD4(2), obteve como precondicionador o menor tempo de processamento dentre os algoritmos avaliados, mantendo o melhor desempenho já obtido para o caso 6.2.1. Quanto ao caso 7.2, o tempo de processamento piorou em comparação com o caso 6.2.2. Em relação ao caso 7.3, apenas o WAMG_hD4(4) melhorou o desempenho em comparação com o problema 6.2.3, ainda assim, o resultado em termos de tempo de processamento foi pior que o GCP.

8 CONCLUSÃO

Este capítulo traz as conclusões do trabalho, as contribuições desta tese e sugestões para trabalhos futuros.

Analisando os resultados obtidos nos capítulos anteriores, conclui-se que os objetivos inicialmente propostos foram atingidos. Códigos computacionais do WAMG e do PWAMG foram implementados e comparados entre si e com o GCP.

Para o WAMG as comparações em termos de tempo de processamento e número de ciclos foram efetuadas utilizando-se filtros de Haar e de Daubechies, este último com decimação por dois e por quatro. O PWAMG foi comparado com o WAMG e com o GCP. Os casos foram direcionados para problemas de elasticidade, difusão e advecção; uni e bidimensionais.

8.1 CONTRIBUIÇÕES DESTA TESE

Os resultados numéricos obtidos geraram contribuições no sentido de se indicar o algoritmo mais adequado para casos específicos, como pode ser observado nas seções 6.3 e 7.4.

Os casos da seção 6.1 demonstraram a superioridade do WAMG para problemas unidimensionais com apenas um grau de liberdade por nó, em comparação com o gradiente conjugado precondicionado (GCP). Os resultados mostraram que o WAMG perde para o GCP em termos de tempo de processamento quando o problema apresenta mais de um grau de liberdade por nó, conforme mostrado na seção 6.1.3. Os resultados mostraram também que o WAMG é sensível ao número de condição da matriz do sistema a ser resolvido.

Para os casos bidimensionais da seção 6.2, o WAMG também se mostrou superior ao GCP, em duas análises, a saber; i) em termos de menor tempo de processamento; e ii) em termos de convergência. O WAMG além de se mostrar superior ao GCP em termos de tempo de processamento, apresentou convergência para o caso da difusão advecção, problema que não é convergente com o GCP. Outras contribuições deste trabalho em relação aos

casos bidimensionais do capítulo 6 são elencadas, a saber. Constatou-se que o WAMG é sensível às propriedades mecânicas da estrutura, à direção do carregamento, e às condições de contorno conforme mostraram os resultados da seção 6.2.4.

No capítulo 7 o WAMG é utilizado como um precondicionador (PWAMG). Os resultados obtidos demonstraram que o algoritmo WAMG_hD4(2) utilizado como precondicionador do gradiente conjugado é mais eficiente que o próprio GCP. Outra constatação interessante é que houve convergência para o problema da advecção quando o WAMG foi utilizado como um precondicionador para o GCP, problema este que não converge com o GCP.

8.2 SUGESTÕES PARA TRABALHOS FUTUROS

No caso do WAMG aplicado em problemas da Engenharia, mesmo com os avanços obtidos neste trabalho e elencados na seção anterior, novos desafios surgiram, e ficam como sugestões para trabalhos futuros. A seguir citam-se algumas questões que merecem ser pesquisadas.

1. Avaliar a sensibilidade dos diferentes filtros em relação ao número de condição da matriz do sistema a ser resolvido.
2. Investigar porque o tempo de processamento oscila para o algoritmo WAMG_hD2(2) em função do número de iterações em alguns casos.
3. Investigar a influência da discretização (diferenças finitas e elementos finitos) no desempenho do WAMG.
4. Aprofundar a investigação da interferência da natureza dos deslocamentos (alongamento, cisalhamento, rotação, etc.) em problemas estruturais da Mecânica, sob a luz do operador diferencial governante do problema, no desempenho do WAMG.
5. Investigar a influência do fator de sobrerrelaxação para o algoritmo WAMG aplicado em problemas estruturais, tendo como ponto de partida os resultados obtidos nesta tese.

6. Investigar o número ótimo de iterações internas para o WAMG para outros casos, confrontando os resultados com aqueles obtidos neste trabalho.

REFERÊNCIAS

ADAMS, M. F. Algebraic multigrid methods for constrained linear systems with applications to contact problems in solid mechanics. **Numerical Linear Algebra With Applications,** v. 00, p. 1-6, 2000.

ADAMS, M. F.; TAYLOR, R. L. Parallel Multigrid solvers for 3D-unstructured large deformation elasticity and plasticity finite element problems**. Finite Element in Analysis and Design**, v. 36, p. 197–214, 2000**.**

ALVES FILHO, A. **Elementos Finitos:** a base da tecnologia CAE. São Paulo: Editora Érica, 2007.

AVUDAINAYAGAM, A. and VANI, C. Wavelet based Multigrid methods for linear and nonlinear elliptic partial differential equations. **Applied Mathematics and Computation**, v. 148, n.2, p. 307 – 320, 2004.

BANK, R. E.; DUPONT, T. F.; YSERENTANT, H. The hierarchical basis Multigrid method. **Numerische Mathematik**, v. 52, p. 427-458, 1988.

BATHE, K. J. **Finite Element Procedures in Engineering Analysis.** New Jersey: Prentice–Hall, 1996.

BORDNER, J.; SAIED, F. MGLab: An Interactive Multigrid Environment. In: **Proceedings** of Cooper Mountain Conference on Multigrid Methods, 7, 1995. Copper Mountain, Colorado, 1995, p. 57-71. Disponível em: http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19970006857_1997005063.pdf#page=74
Acesso em: 01 abr. 2013.

BORESI, A. P. and CHONG, K. P. **Elasticity in Engineering Mechanics**. New Jersey: Prentice Hall, 1987.

BRANDT, A. Multilevel adaptive technique (MLAT) for fast numerical solution to boundary value problems. In **Proceedings** of International Conference On Numerical Methods in Fluid Mechanics, n. 3, 1973, v. 1, Berlin, pp 82-89, 1973.

BREZINA, M.; CLEARY, A. J.; FALGOUT, R. D.; HENSON, V. E.; JONES, J. E.; MANTEUFFEL, T. A.; MCCORMICK, S. F.; RUGE, J. W. Algebraic multigrid based on element interpolation (AMGe). **Society for Industrial and Applied Mathematics**, v. 22, n. 5, p. 1570 – 1592, 2000.

BRIGGS, W. L.; HENSON, V. E. Wavelets and Multigrid. **Society for Industrial and Applied Mathematics.** v. 14, n. 1, p. 506 – 510, 1993.

BRIGGS, W. L.; HENSON, V. E. **A Multigrid Tutorial**. Philadelphia: Society for Industrial and Applied Mathematics, 2000.

BURDEN, R. L.; FAIRES, J. D. **Análise Numérica**. São Paulo: Pioneira Thomson Learning, 2003.

CHANG, Q.; WONG, Y. S.; FU, H. On the algebraic multigrid method. **Journal of Computational Physics**, v. 125, p. 279 – 292, 1996.

CHENG, D. and. LUB, C. and ZENG, T. A fast wavelet block Jacobi method. **Journal of Mathematical Analysis and Applications**. v. 388, n. 2, p. 1080 – 1089, 2012.

CHOPRA, A. K. **Dynamics of Structures Theory and Applications to Earthquake Engineering**. New jersey: Prentice Hall, 1995.

CHUI, C. K. **An introduction to Wavelets.** San Diego: Academic Press, 1992.

CODY, M. A. The fast wavelet transform. **Dr. Dobb's Journal**, v. 17, n. 4, p. 16 – 101, April, 1992.

CRAIG JR, R. R. **Structural Dynamics an introduction to Computer Methods**. New York: John Wiley, 1981.

DAHMEN, W.; KURDILA, A.; OSWALD, P. **Multiscale wavelet methods for partial differential equations**. *In*: Wavelet analysis and its applications, v.6. San Diego: Academic Press, 1997.

DATTA, B. N. **Numerical linear algebra and applications**. Boston: Brooks/Cole Publishing Company, 1995.

DAUBECHIES, I. Orthonormal bases of compactly supported wavelets. **Communications on Pure and Applied Mathematics**, v. 41, p. 909 – 996, 1988.

DE LEON, D. R. N. **Wavelet operators applied to Multigrid Methods**. 2000. (Ph.D. Thesis). Mathematics Department of University of California, Los Angeles, June.

DE LEON, D. A New Wavelet Multigrid Method. **Journal of Computational and Applied Mathematics**, v.220, n.1-2, p.674 – 685, 2008.

DE LEON, D. A Wavelet Multigrid Method Using Symmetric Biorthogonal Wavelets. **American Journal of Computational Mathematics** v. 3, p. 127 – 136, 2013.

FALGOUT, R. D. An introduction to algebraic multigrid**. Computing in Science and Engineering**, v. 12, p. 1-11, June, 2006.

FEDORENKO, R. P. The Speed of convergence of one iterative process**. USSR Computational And Mathemathics and Mathemathics Physics,** v.4, n. 3, p. 227 - 235, 1964.

FERZIGER, J. H.; PERIC, M. **Computational methods for fluid dynamics**. [S.l.]:Springer, 1999.

GARCIA, V. M.; ACEVEDO, L.; VIDAL, A. M. Variants of algebraic wavelet-based multigrid methods: application to shifted linear systems, **Applied Mathematics and Computation**, v. 202, n.1, p. 287 – 299, 2008.

GERE, J. M.; TIMOSHENKO, S. P. **Mechanics of Materials**. Boston: Thomson, 1997.

GRIEBEL, M., OELTZ, D.; SCHWEITZER, M. A. An algebraic multigrid method for linear elasticity. **SIAM Journal on Scientific Computing**, v. 25, n. 2, p. 385 – 407, 2003.

HACKBUSCH, W. **Ein iteratives Verfahren zur schnellen Auflösung elliptischer Randvert probleme,** tech. Rep. 76–12, Institute for Applied Mathematics, University of Cologne, West Germany, 1976.

HACKBUSCH, W. On the multi-grid method applied to difference equations, **Computing**, v. 20, n.4, p. 291-306, 1978.

HESTENES, M. R.; STIEFEL, E. Methods of conjugate gradients for solving linear systems. **Journal of Research of the National Bureau of Standards.** v. 49, p. 409-435, 1952.
Retirado de http://nvlpubs.nist.gov/nistpubs/jres/049/6/V49.N06.A08.pdf, em 08.01.2013.

JOUGLARD, C. E.; COUTINHO, A. L. G. A. A comparison of iterative multi-level finite element solvers. **Computers and Structures**, v. 69, n.5, p. 655 – 670, 1998.

KAISER, G. **A Friendly Guide to Wavelets.** Boston: Birkhauser, 1994.

KHELIFI, S. C.; MÉCHITOUA, N.; HÜLSEMANN, F.; MAGOULÈS, F. A Hybrid Multigrid Method for Convection-Diffusion Problems. **Journal of Computational and Applied Mathematics**, 2013. Artigo in press: http://dx.doi.org/10.1016/j.cam.2013.05.003.

KWON, Y. W. **The Finite Element method using Matlab**. New York: CRC Mechanical Engineering Series, 2000.

KREYSZIG, E. **Advanced Engineering Mathematics.** 8 ed. New York: John Wiley & Sons, 1999.

MALLAT, S. G. A theory for multiresolution signal decomposition: the wavelet representation. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 11, n. 7, p. 674 – 693, July, 1989.

MCCORMICK, S. F. (Ed.). **Multgrid Methods**: frontiers in applied mathematics, v. 3. Philadelphia: SIAM, 1987, Second printing, 1994.

MONTERO, R. S.; Llorente, I. M.; SALAS, M. D. Robust multigrid algorithms for the Navier-Stokes equations. **Journal of Computational Physics.** v. 173, p. 412 – 432, 2001.

MORETTIN, P. A. **Ondas e Ondaletas:** da análise de Fourier à análise de ondaletas. São Paulo: Editora da Universidade de São Paulo, EDUSP, 1999.

MORO FILHO, R. C. **Aplicação da Técnica Multigrid em Transferência de Calor Computacional.** 2004. Dissertação (Mestrado). Métodos Numéricos em Engenharia, Setor de Ciências Exatas e Tecnologia – Universidade Federal do Paraná, Curitiba, 2004.

OLIVEIRA, F.; PINTO, M. A. V.; SANTIAGO, C. D.; MARCHI, C. H. Efeito de parâmetros do Método Multigrid CS e FAS sobre o tempo de cpu em problemas 1d lineares e não-lineares**. Anais** XXVII CILAMCE, 2006.

PEREIRA, F. H.; VERARDI, S. L. L.; NABETA, S. A fast algebraic multigrid preconditioned conjugate gradient solver. **Applied Mathematics and Computation**, v. 179, n. 1, p. 344 - 351, 2006a.

PEREIRA, F. H.; VERARDI, S. L. L.; NABETA, S. A wavelet-based algebraic Multigrid preconditioner for sparse linear systems. **Applied Mathematics and Computation**, v. 182, n.2, p. 1098 - 1107, 2006b.

PEREIRA, F. H. **O método multigrid algébrico na resolução de sistemas lineares oriundos do método dos elementos finitos**. 2007. Tese (Doutorado em Sistemas de Potência) - Escola Politécnica, Universidade de São Paulo, São Paulo, 2007. Disponível em: <http://www.teses.usp.br/teses/disponiveis/3/3143/tde-22072007-212201/>. Acesso em: 2013-10-14.

PFLAUM, C. A multigrid conjugate gradient method. **Applied Numerical Mathematics**. v. 58, n.12, p. 1803 – 1817, 2008.

PINTO, M. A. V.; MARCHI, C. H. Efeito da razão de aspecto de malha sobre o tempo de CPU para a Equação de Laplace 2D resolvida com o Método Multigrid. In: Iberian Latin American Congress on Computational Methods in Engineering, n.XXVII, 2006, Belém. **Anais,** CILAMCE, 2006.

PINTO, M. A. V. **Comportamento do Multigrid Geométrico em Problemas de Transferência de Calor.** 2006. Tese (Doutorado). Métodos Numéricos em Engenharia, Setor de Ciências Exatas e Tecnologia - Universidade Federal do Paraná, Curitiba, 2006.

STRANG, G.; NGUYEN, T. **Wavelets and Filter Banks**. [S.l.]: Wellesley-Cambridge Press, 1997.

SUERO, R.; PINTO, M. A. V.; MARCHI, C. H.; ARAKI, L. K.; ALVES, A. C. Analysis of algebraic multigrid parameters for two-dimensional steady-state heat diffusion equations. **Applied Mathematical Modelling**. v. 36 p 2996 - 3006, 2012.

TEWFIK, A. H.; SINHA, D.; JORGENSEN, P. On the Optimal Choice of a Wavelet for Signal Representation. **IEEE Transactions on Information Theory**, v. 38, n. 2, p. 747 – 765, March, 1992.

TROTTENBERG, U.; OOSTERLEE, C., SCHüLLER, A. **Multigrid**. London: Academic Press, 2001.

VETTERLI, M.; HERLEY, C. Wavelets and Filter Banks: Theory and Design. **IEEE Transactions on Signal Processing,** v. 40, n. 9, p. 2207 – 2232, September, 1992.

WANG, G.; DUTTON, R. W.; HOU, J. A fast wavelet multigrid algorithm for solution of electromagnetic integral equations. **Microwave and Optical Technology Letters**, v. 24, n. 2, p. 86 – 90, 2000.

WESSELING, P. **A robust and efficient Multigrid method.** Multigrid methods, Lecture Notes in Mathematics 960. In: HACKBUSCH, W.; TROTTENBERG, U. (Ed.), p. 614 – 630, springer, Berlin, 1982.

WESSELING, P. **An Introduction to Multigrid Methods**, New York: John Wiley & Sons, 1992.

XIAO, Y-X and ZHANG, P. and SHU, S. An Algebraic Multigrid Method with Interpolation Reproducing Rigid Body Modes for Semi-definite Problems in Two-dimensional Linear Elasticity. **Journal of Computational and Applied Mathematics**, v. 200, n.2, p. 637 – 652, 2007.

XU, X. Multilevel methods for Wilson element approximation of elasticity problem. **Computer Methods in Applied Mechanics and Engineering**, v. 174, n.1, p. 191-201, 1999.

YOO, J. Multigrid for the Galerkin Least Squares method in Linear Elasticity. **Journal Mathematical Analysis and Applications**, v. 286, p. 326 – 339, 2003.

ANEXO A: PUBLICAÇÕES ASSOCIADAS A ESTE TRABALHO

Abaixo, listam-se os trabalhos publicados/submetidos gerados a partir da pesquisa desta tese.

1. FAGUNDES, F. A., MACHADO, R. D. e MARIANI, V. C. **Aplicação dos Métodos Multigrid Geométrico e Algébrico em Sistemas Estruturais de Barras**. XXX Iberian Latin-American Congress on Computational methods in Engineering, CILAMCE. 2009.

2. FAGUNDES, F. A., MACHADO, R. D. e MARIANI, V. C. **A NEW APPROACH FOR ALGEBRAIC MULTIGRID**. 1st International Congress of Mathematics, Engineering and Society, ICMES. 2009.Curitiba.

3. FAGUNDES, F. A., MACHADO, R. D. e MARIANI, V. C. **New Algorithms For Algebraic Wavelet Multigrid Method**. Mathematics and Computers Simulation (submetido)

4.FAGUNDES, F. A., MACHADO, R. D. e MARIANI, V. C **Fast Algorithm for Algebraic Wavelets Multigrid Method as Preconditioner** Mathematics and Computers in Simulation (no prelo)

5. FAGUNDES, F. A., MACHADO, R. D. e MARIANI, V. C. **Fast Algorithms For Algebraic Wavelet Multigrid Method Applied To Two-Dimensional Heat Equation.** 22nd International Congress of Mechanical Engineering, COBEM 2013.

ANEXO B: TEOREMA DA CONVERGÊNCIA DE UMA SEQUÊNCIA DE POTÊNCIAS DE UMA MATRIZ QUADRADA

Este anexo apresenta a reprodução da prova do teorema 2.1 (DATTA, 1995, p. 30).

A sequência A, A^2,... de potências da matiz A converge para a matriz zero se $|\lambda_i| < 1$ para cada autovalor λ_i de A.

Prova:

Para cada matriz A, quadrada de ordem n, existe uma matriz não singular T tal que

$T^{-1}AT = J$, *na qual*

$$J = \begin{pmatrix} J_1 & & 0 \\ & \ddots & \\ 0 & & J_r \end{pmatrix}$$

Na qual cada J_i tem a forma

$$J = \begin{pmatrix} \lambda_i & 1 & & & \\ & \lambda_i & 1 & 0 & \\ & & \ddots & \ddots & \\ & & & \ddots & 1 \\ & 0 & & & \lambda_i \end{pmatrix}$$

É um cálculo simples para mostrar que

$$J_i^k = \begin{pmatrix} \lambda_i^k & k\lambda_i^{k-1} & \binom{k}{2}\lambda_i^{k-2} & \cdots & \binom{k}{n-1}\lambda_i^{k-n+1} \\ & \lambda_i^k & k\lambda_i^{k-1} & \cdots & \vdots \\ & & \ddots & \ddots & \vdots \\ & & & \ddots & k\lambda_i^{k-1} \\ & & & & \lambda_i^k \end{pmatrix}$$

Pode se ver que $J_i^k \rightarrow 0$ se e somente se $|\lambda_i| < 1$.