UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA

MÉTODOS ITERATIVOS E MULTIGRID ADAPTÁVEIS EM MALHAS NÃO-ESTRUTURADAS

Autor: **Marco Lúcio Bittencourt**
Orientador: **Dr. Raúl Antonino Feijóo**
Co-orientador: **Prof. Dr. Hans Ingo Weber**

39/96

ESTE EXEMPLAR CORRESPONDE A REDAÇÃO FINAL DA TESE DEFENDIDA POR Marco Lúcio Bittencourt E APROVADA PELA COMISSÃO JULGADORA EM 12 / 07 / 96.

ORIENTADOR

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

B548m

Bittencourt, Marco Lúcio
Métodos iterativos e multigrid adaptáveis em malhas não-estruturadas / Marco Lúcio Bittencourt.--Campinas, SP: [s.n.], 1996.

Orientador: Raúl Antonino Feijóo.
Tese (doutorado) - Universidade Estadual de Campinas, Faculdade de Engenharia Mecânica.

1. Métodos iterativos. 2. Método dos elementos finitos. 3. Programação orientada a objetos (Computação). 4. C++ (Linguagem de programação de computador). 5. Análise de erros (Matemática). 6. Método de redes múltiplas (Análise numérica). I. Feijóo, Raúl Antonino. II. Universidade Estadual de Campinas. Faculdade de Engenharia Mecânica. iII. Título.

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
DEPARTAMENTO DE PROJETO MECÂNICO

MÉTODOS ITERATIVOS E MULTIGRID ADAPTÁVEIS EM MALHAS NÃO-ESTRUTURADAS

Autor: **Marco Lúcio Bittencourt**
Orientador: **Dr. Raúl Antonino Feijóo**
Co-orientador: **Prof. Dr. Hans Ingo Weber**

Curso: Engenharia Mecânica
Área de Concentração: Mecânica dos Sólidos

Trabalho apresentado à Comissão de Pós-Graduação da Faculdade de Engenharia Mecânica, como parte dos requisitos para obtenção do título de Doutor em Engenharia Mecânica.

Campinas, 1996
S.P. - Brasil

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
DEPARTAMENTO DE PROJETO MECÂNICO

TESE DE DOUTORADO

Métodos Iterativos e Multigrid Adaptáveis em Malhas Não-estruturadas

Autor: **Marco Lúcio Bittencourt**
Orientador: **Dr. Raúl Antonino Feijóo**
Co-orientador: **Prof. Dr. Hans Ingo Weber**

Dr. Raúl Antonino Feijóo, Presidente
LNCC/CNPq

Prof. Dr. Clóvis Raimundo Maliska
DEM/UFSC

Prof. Dr. Paulo de Mattos Pimenta
DEEF/EPUSP

Prof. Dr. Philippe Remy Bernard Devloo
DCC/FEC/UNICAMP

Prof. Dr. Fernando Iguti
DMC/FEM/UNICAMP

Campinas, 12 de julho de 1996.

Agradecimentos

Ao Dr. Raúl Feijóo pelo seu exemplo como pessoa, dedicação, postura profissional, apoio e confiança dispensados.

Ao Prof. Dr. Hans Ingo Weber pelo constante apoio e estímulo às minhas iniciativas.

Aos Profs. Dr. Ilmar Ferreira Santos e Dr. Eduardo Fancello pelo incentivo e confiança na realização do trabalho.

Aos Profs. Dr. Loir Afonso Moreira e Dr. Robson Pederiva pelo apoio dado quando exerceram a chefia do DPM/FEM.

Ao Prof. Dr. Kamal Abel Radi Ismail pela sua colaboração como coordenador da Sub-CPG/FEM.

Ao Dr. Marcelo Venere e ao Dr. Enzo Dari pelo auxílio na implementação dos algoritmos de renumeração e geração das malhas tridimensionais.

Ao Prof. Dr. Craig Douglas pela avaliação e sugestões ao trabalho.

Ao Eng. Antônio Carlos Salgado Guimarães pela ajuda na implementação dos programas.

Ao DPM, FEM, UNICAMP, LNCC, FAPESP, CNPq pelos suportes institucional e financeiro para a realização do trabalho.

Ao CENAPAD/UNICAMP pela colaboração na utilização dos seus computadores.

Conteúdo

Lista de Figuras

Lista de Tabelas

Nomenclatura

Letras Latinas :

$a(\cdot,\cdot)$ – forma bilinear

$a_{ij}(\cdot)$ – funções mensuráveis no sentido de Lebesgue

$\mathbf{a}$ – vetor de incógnitas do *patch*

b – termo do lado direito do PVC

$\mathbf{b}$ – vetor dos termos independentes

$\mathbf{b}^{nh}$ – vetor dos termos independentes na malha Ω^{nh}

$\mathbf{b}_p$ – vetor dos termos independentes do *patch*

$\mathbf{d}^{(n)}$ – vetor de direção na iteração n

e – erro entre as soluções exata e aproximada por elementos finitos

e^* – erro global entre as tensões exata e recuperada

e^*_{ad} – erro admissível nos elementos

e^*_i – erro local entre as tensões exata e recuperada no elemento i

$\mathbf{e}$ – vetor de erro na aproximação da solução do sistema de equações

$\mathbf{e}^{nh}$ – vetor de erro da aproximação na malha Ω^{nh}

$f(\cdot)$ – funcional quadrático

f_b – forças de corpo

$\mathbf{f}_b$ – vetor com as componentes das forças de corpo

$\mathbf{g}^{(n)}$ – vetor gradiente na iteração n

h – tamanho característico da malha

$\mathbf{h}^{(n)}$ – vetor auxiliar no método de GC com pré-condicionamento na iteração n

$\mathbf{k}$ – vetor conhecido de um método iterativo

$l(\cdot)$ – forma linear

m – altura média das colunas (matriz *skyline*) ou número médio de elementos por linha (matriz esparsa)

$m(\cdot)$ – menor autovalor algébrico de uma matriz

$m_E(\cdot)$ – estimativa para o menor autovalor algébrico de uma matriz

m_i – número médio de elementos por linha na matriz do sistema da malha i

m_i^U – altura da coluna i (matriz *skyline*) ou número de elementos na linha i (matriz esparsa)

n_1 – número de vezes que um ciclo multigrid passou pelo nível 1

n_p – número total de pontos de amostragem num *patch*

n_i^{aux} – espaço dos vetores auxiliares do método iterativo na malha i

n_i^p – número total vezes que o operador de pronlogamento foi executado na malha i

n_i^r – número total vezes que o operador de restrição foi executado na malha i

$p_n(\cdot)$ – polinômio característico de um problema de autovalor

$\mathbf{q}_j$ – vetores ortonormais gerados pelo método de Lanczos

$\bar{r}$ – parâmetro da aceleração de Chebyshev

$\mathbf{r}$ – vetor com as componentes do resíduo da aproximação $\mathbf{v}$

$\mathbf{r}^{nh}$ – vetor resíduo na malha Ω^{nh}

$\mathbf{t}$ – vetor das tensões nodais ponderadas pela área dos elementos

u – solução clássica do PVC

$\hat{u}$ – solução fraca do PVC

u_h – solução aproximada do PVC

u_j – j-ésima componente do vetor $\mathbf{u}$

$\mathbf{u}$ – vetor das incógnitas do sistema de equações

$\mathbf{u}_{dir}$ – vetor com a solução do sistema de equações obtida pelo método direto de Gauss

$\mathbf{u}_h$ – vetor com as componentes nodais obtidas pelo MEF

$\mathbf{u}_{ite}$ – vetor com a solução do sistema de equações obtida por um método iterativo

$\mathbf{u}^{(n)}$ – aproximação na iteração n de um método iterativo

$\mathbf{u}^{nh}$ – vetor solução/aproximação na malha Ω^{nh}

$\mathbf{u}^*$ – campo de deslocamentos interpolado com ordem mais alta

$v^{(i)}$ – derivada generalizada

$\mathbf{v}$ – vetor com a aproximação da solução do sistema de equações

$\mathbf{v}_k$ – vetor correspondente ao k-ésimo modo de Fourier

$\mathbf{v}^{nh}$ – vetor com a aproximação do sistema de equações da malha Ω^{nh}

$w(x)$ – transformação linear mapeando $m(\mathbf{G}) \leq x \leq M(\mathbf{G})$ em $-1 \leq w \leq 1$

w_s, w_r, w_b, w_u – funções de ponderação do recuperador EPEC

A – operador diferencial de ordem $2k$ de um PVC

A_i – coordenadas de área

$\mathbf{A}$ – matriz do sistema de equações

$\mathbf{A}^{nh}$ – matriz do sistema de equações na malha Ω^{nh}

$\mathbf{A}_p$ – matriz do sistema de equações do *patch*

$\tilde{\mathbf{A}}$ – matriz pré-condicionada

B – operador diferencial de ordem k de um PVC correspondente às condições de contorno

$\mathbf{B}$ – matriz de iteração do método de Jacobi

$C^{2k}(\Omega)$ – conjunto das funções com derivadas contínuas até a ordem $2k$ em Ω

$\tilde{C}^k(\Omega)$ – conjunto das funções com derivadas generalizadas quadrado-integáveis até a ordem k em Ω

$\mathbf{C}$ – matriz de pré-condicionamento

$\mathbf{C}_L$ – matriz triangular inferior com diagonal nula

$\mathbf{C}_U$ – matriz triangular superior com diagonal nula

D – tensor de elasticidade de Green

$D^i u$ – derivada generaliza de ordem $|i|$ da função u

$\mathbf{D}$ – matriz diagonal

E – módulo de elasticidade longitudinal

$\mathbf{E}$ – matriz não-singular

$F(\cdot)$ – funcional quadrático na forma discreta

$\mathbf{F}$ – matriz triangular inferior usada no método de GC com pré-condicionamento

$\mathbf{G}$ – matriz de iteração do método iterativo

$\mathbf{G}_c$ – matriz diagonal do método GC com pré-condicionamento

$\mathbf{G}_{[\gamma]}$ – matriz de iteração do método de extrapolação

$H^k(\Omega)$ – espaço de Hilbert de ordem k em Ω

I_h^{2h} – operador de restrição

I_{2h}^h – operador de interpolação

$\mathbf{I}$ – matriz identidade

$\mathbf{J}$ – matriz jacobiana

$L_2(\Omega)$ – espaço linear das funções quadrado-integráveis no sentido de Lebesgue

$\mathbf{L}$ – matriz triangular inferior com diagonal unitária

$\mathbf{L}'$ – matriz triangular inferior

$M(\cdot)$ – maior autovalor algébrico de uma matriz

$M_E(\cdot)$ – estimativa para o maior autovalor algébrico de uma matriz

$\mathbf{M}$ – matriz de massa consistente

N – ordem do sistema de equações

N_i – número de variáveis da malha i

N_i^{els} – número de elementos da malha i

$N_i^{\text{nós}}$ – número de nós na malha i

N_m – número de malhas

$N_{\text{nós}}$ – número de nós do elemento

N_p^{els} – número de elementos do *patch*

N_{pi} – número de pontos de integração

$\mathbf{N}$ – matriz com as funções de forma elementares

$\mathbf{P}$ – vetor com os termos polinomiais

$\mathbf{P}_x$ – vetor com as derivadas parciais dos termos polinomiais em relação a x no sistema global do *patch*

$\mathbf{P}_y$ – vetor com as derivadas parciais dos termos polinomiais em relação a y no sistema global do *patch*

$\mathbf{P}_{\bar{x}}$ – vetor com as derivadas parciais dos termos polinomiais em relação a x no sistema local do *patch*

$\mathbf{P}_{\bar{y}}$ – vetor com as derivadas parciais dos termos polinomiais em relação a y no sistema local do *patch*

$\mathbf{P}_{\xi}$ – vetor com as derivadas parciais dos termos polinomiais em relação a coordenada local do elemento ξ

$\mathbf{P}_{\eta}$ – vetor com as derivadas parciais dos termos polinomiais em relação a coordenada local do elemento η

$Q_n(\cdot)$ – sequência de polinômios empregada nos procedimentos de aceleração polinomial

$\mathbf{Q}$ – matriz de partição do método iterativo

$\bar{\mathbf{Q}}^*$ – vetor análogo a $\mathbf{P}$ com os termos da expansão polinomial

$\bar{\mathbf{Q}}$ – matriz diagonal formada pelo vetor $\bar{\mathbf{Q}}^*$

$R(\cdot)$ – coeficiente de Rayleigh

$R_n(\cdot)$ – taxa média de convergência

$\bar{R}_n(\cdot)$ – taxa média virtual de convergência

$R_\infty(\cdot)$ – taxa assimptótica de convergência

$\bar{R}_\infty(\cdot)$ – taxa assimptótica virtual de convergência

$\mathbf{R}_{eq}$ – resíduo na equação de equilíbrio das tensões recuperadas por expansão polinomial

$\mathbf{R}_T$ – resíduo entre as tensões recuperadas por expansão polinomial e por elementos finitos

$\mathbf{R}_{T_b}$ – resíduo entre as tensões calculadas com $\mathbf{u}^*$ e os valores $\mathbf{T}_b$ prescritos no contorno

$\mathbf{R}_u$ – resíduo entre os deslocamentos $\mathbf{u}^*$ e a solução de elementos finitos

$\mathbf{R}_{u_b}$ – resíduo entre os deslocamentos $\mathbf{u}^*$ e os valores $\mathbf{u}_b$ prescritos no contorno

$\bar{S}(\cdot)$ – raio espectral virtual de uma matriz

$S^k(\Omega)$ – espaço linear com as funções de $\tilde{C}^k(\Omega)$

$T_n(\cdot)$ – polinômio de Chebyshev de ordem n

$\mathbf{T}$ – matriz tridiagonal gerada pelo método de Lanczos

$\mathbf{T}(\cdot)$ – tensor de Cauchy

$\mathbf{T}_h$ – tensor de tensões obtido por elementos finitos

$\mathbf{T}_r$ – submatriz dos termos principais de ordem r

$\mathbf{T}^*$ – tensão recuperada

$\bar{\mathbf{T}}^*$ – tensões nodais recuperadas

$\mathbf{T}_p^*$ – tensões recuperadas por expansão polinomial

$\mathbf{U}$ – matriz triangular superior com diagonal unitária

V – conjunto das funções admissíveis

V_i – coordenadas de volume

$\bar{V}$ – conjunto das funções de $H_k(\Omega)$ satisfazendo as condições de contorno

$\bar{V}_h$ – subconjunto de dimensão finita com as funções de aproximação do PVC

$\mathbf{V}$ – matriz formada pelas derivadas do vetor $\mathbf{Q}$

W_l – coeficiente de ponderação na integração numérica

$\mathbf{W}$ – matriz de simetria do método iterativo

Letras Gregas :

α_j – coeficientes da matriz tridiagonal do método de Lanczos

$\alpha_{n,i}$ – coeficientes da combinação linear na aceleração polinomial

β – metade da largura de banda

β_j – coeficientes da matriz tridiagonal do método de Lanczos

β_n – parâmetro empregado no método de GC

β_p – coeficiente de penalização

δ_{il} – delta de Kronecker

η – erro percentual relativo global

$\bar{\eta}$ – erro percentual admissível

γ – fator de extrapolação

γ_i – constante empregada no procedimento de aceleração polinomial

γ_{ot} – fator de extrapolação ótimo

$\bar{\gamma}$ – parâmetro da aceleração de Chebyshev

λ_i – autovalores de uma matriz $(i = 1, 2, \ldots)$

$\kappa(\cdot)$ – número de condição espectral de uma matriz

μ_i – autovalores da matriz de iteração $\mathbf{G}$

ν – coeficiente de Poisson

ν_0 – número de ciclos internos numa estratégia multigrid

ν_1 – número de pré-relaxações

ν_2 – número de pós-relaxações

ω – parâmetro de relaxação

ϕ_i – função de base de $\bar{V}_h$

ρ_i – constante empregada no procedimento de aceleração polinomial

$\bar{\rho}_n$ – parâmetro da aceleração de Chebyshev

$\bar{\sigma}$ – parâmetro da aceleração de Chebyshev

τ_n – escalar na iteração n determinado pelo algoritmo de busca

ξ – precisão

ζ_i – razão entre os erros no elemento e admissível

Γ – contorno do domínio do PVC

Γ_u – parte do contorno com condições essenciais

Γ_b – parte do contorno com condições naturais

Ω – domínio do PVC

Ω_e – domínio do elemento e

Ω^{nh} – malha de elementos com tamanho nh

Ω_p – domínio do *patch* de elementos

$\bar{\Omega}_p$ – domínio local do *patch* de elementos

Símbolos :

$\mathcal{F}$ – funcional

$\mathcal{L}$ – matriz de iteração do método Gauss-Seidel

$\mathcal{L}_\omega$ – matriz de iteração do método SOR

$\mathcal{S}_\omega$ – matriz de iteração do método SSOR

$\mathcal{T}_k$ – triangulação k

∇^s – tensor de deformação

$\Re$ – conjunto dos números reais

CHSS – aceleração de CHebyshev aplicado ao método SSOR

CIT – Custo de uma ITeração de um método iterativo

CMG – Correção de Malha Grossa

EP – Expansão Polinomial

EPE – Expansão Polinomial com Equilíbrio

EPEC – Expansão Polinomial com Equilíbrio e Condições de Contorno

GC – Gradiente Conjugado

GCD – Gradiente Conjugado com matriz de pré-condicionamento Diagonal

GCGS – Gradiente Conjugado com matriz de pré-condicionamento de Gauss-Seidel

GCSS – Gradiente Conjugado com matriz de pré-condicionamento de SSOR

GS – Gauss-Seidel

JAC – Jacobi

MEF – Método de Elementos Finitos

NIT – Número de ITerações

PA – Ponderação pela Área

PVC – Problema de Valor de Contorno

SPD – Simétrico(a) Positivo(a)-Definido(a)

ZZ – Zhu-Zienkiewicz

Resumo

BITTENCOURT, Marco Lúcio, *Métodos Iterativos e Multigrid Adaptáveis em Malhas Não-Estruturadas,* Campinas, Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 1996, 147 p., Tese (Doutorado).

Neste trabalho, apresentam-se métodos numéricos diretos, iterativos e multigrid para a solução de sistemas de equações, provenientes da aplicação do método de elementos finitos a problemas elípticos lineares, tomando-se exemplos de elasticidade bi e tridimensional. Discutem-se também aspectos de análise adaptável, tomando-se um estimador de erro e alguns procedimentos para recuperação de tensões. Todos os algoritmos são implementados empregando o modelo de programação por objetos através da linguagem C++.

Comparações entre os métodos diretos e iterativos em termos do número de operações e espaço de memória são apresentadas, revelando a superioridade das técnicas iterativas quando aceleradas por estratégias multigrid, principalmente em problemas tridimensionais.

Tradicionalmente, os métodos multigrid têm sido utilizados com malhas aninhadas, dificultando o tratamento de problemas de engenharia com contornos complexos. Nesta tese, consideram-se malhas não-estruturadas e não-aninhadas geradas por técnicas frontal e de Delaunay. O acoplamento com procedimentos adaptáveis permite obter uma sequência ótima de malhas para a solução de problemas, dentro de um erro admissível especificado.

Os procedimentos numéricos foram incorporados a uma base de programas já desenvolvida, empregando o paradigma por objetos com C++. Dois ambientes com ferramentas para a definição do contorno geométrico, geração automática de malhas, estimação de erros, refinamento de malhas e visualização de resultados, aplicados a problemas bidimensionais de elasticidade linear, são também apresentados.

Palavras chaves:

- Métodos iterativos
- Método dos elementos finitos
- Programa orientada a objetos (Computação)
- C++ (Linguagem de programação de computador)
- Análise de erros (Matemática)
- Método de redes múltiplas (Análise numérica)

Abstract

BITTENCOURT, Marco Lúcio, *Adaptive Iterative and Multigrid Methods Applied to Non-structured Meshes,* Campinas, Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 1996, 147 p., Thesis (PhD).

This work presents direct, iterative and multigrid methods for solving systems of equations obtained by means of the finite element method applied to linear elliptic problems, considering two and three dimensional elastic examples. Adaptive analysis aspects are also discussed by taking an error estimator and some stress recovery procedures. All the algorithms are implemented using the object-oriented model with the C++ language.

Comparisons between direct and iterative methods concerned to the number of operations and memory space are presented. The superiority of the iterative techniques accelerated by multigrid strategies can be detected, mainly on three dimensional applications.

The multigrid methods have been used at most with nested meshes. In such cases the treatment of engineering problems with complex boundaries becomes difficult. In this thesis, the meshes are non-structured and non-nested and they are generated by frontal and Delaunay techniques. By using adaptive procedures, it is possible to get an optimal sequence of meshes for solving a problem within a specified admissible error.

The numerical procedures were linked with some other developed programs, using the object paradigm in C++. Two environments with tools for defining geometrical boundaries, automatic meshes generation, errors estimation, meshes refinement and visualization of results are also presented, considering two dimensional linear elastic problems.

Keywords:

- Iterative methods
- Finite element method
- Object-oriented programming (Computing)
- C++ (Computer programming language)
- Error analysis (Mathematics)
- Multigrid methods (Numerical analysis)

Capítulo 1

INTRODUÇÃO

A aplicação de métodos numéricos para a simulação computacional de problemas físicos tem apresentado um crescimento considerável. Como justificativas para este fenômeno, verifica-se a maior robustez e confiabilidade das técnicas de análise empregadas, assim como o aumento exponencial na capacidade de processamento dos computadores. Desta forma, é possível, por exemplo, estudar o comportamento mecânico de um componente, desde a fase de concepção, projeto e otimização, permitindo efetivamente implementar o conceito de protótipo eletrônico.

Vários problemas de engenharia são descritos por uma ou mais equações diferenciais e condições de contorno, definindo um *problema de valor de contorno* (PVC). Como na maioria dos casos, não é possível obter a solução analítica u do PVC, aplicam-se procedimentos numéricos, tais como os métodos de diferenças finitas (MDF) e elementos finitos (MEF), dentre outros, visando determinar uma solução aproximada u_h.

Empregam-se os seguintes passos genéricos para a solução numérica de um problema físico:

- definição de um modelo matemático, dado em geral por um PVC, descrevendo o comportamento de interesse do problema físico;
- determinação de um modelo aproximado ou discreto para o modelo matemático;
- aplicação de um método numérico, conduzindo em geral a um sistema de equações ou uma sequência de sistemas em casos transientes e não-lineares;
- solução do(s) sistema(s) de equações através de algoritmos numéricos apropriados, obtendo-se a solução aproximada u_h;
- determinação de estimativas do erro na solução aproximada e utilização de técnicas adaptáveis visando melhorar a qualidade de u_h.

Sob o ponto de vista prático, deve-se representar a geometria do domínio considerado, gerar uma malha de elementos discretos, resolver o respectivo sistema de equações, estimar o erro na aproximação e refinar a malha inicial caso seja necessário. Todas estas etapas contituem-se em áreas de pesquisa onde vários trabalhos estão sendo desenvolvidos. O objetivo principal a ser buscado é simplificar todo o processo de análise e simulação de componentes, facilitando o trabalho do analista. Observa-se que todos os passos anteriores demandam uso intensivo de computadores e apesar do inegável aumento de capacidade dos mesmos, problemas cada vez mais complexos, tanto no que se refere ao modelo matemático quanto ao número de variáveis, têm sido tratados. Desta forma, ferramentas eficientes devem ser utilizadas em todas as fases da análise numérica de problemas reais.

A ênfase principal deste trabalho está relacionada a algoritmos numéricos para a solução de sistemas de equações, tais como métodos diretos, iterativos e multigrid.[1] Consideram-se exemplos de elasticidade linear empregando-se malhas não-estruturadas.

A implementação dos algoritmos é de grande relevância no que se refere a eficiência dos mesmos. Empregou-se o modelo de programação por objetos através da linguagem C++, permitindo disciplinar e organizar o desenvolvimento de programas de análise por elementos finitos.

Nesta introdução, apresentam-se os principais aspectos dos vários temas tratados nos próximos capítulos. Inicialmente, procura-se caracterizar o tipo de problema analisado. Posteriormente, consideram-se métodos numéricos para a solução de sistema de equações, estimadores de erros, métodos multigrid e conceitos do modelo orientado por objetos aplicados em elementos finitos. Finalmente, ressaltam-se as principais contribuições do trabalho.

1.1 Caracterização do Problema

Em geral, o modelo matemático de um sistema físico é descrito por um PVC, no qual procura-se uma função u satisfazendo uma equação diferencial numa região Ω e condições impostas no contorno Γ. O PVC pode ser escrito como,

$$Au = b \quad \text{em} \quad \Omega \qquad Bu = 0 \quad \text{em} \quad \Gamma \tag{1.1}$$

onde A e B são operadores diferenciais de ordens $2k$ e k, respectivamente; b é uma função conhecida em Ω. Por simplicidade, assume-se a existência apenas de condições de contorno homogêneas de Dirichlet $Bu = 0$ em Γ.

A solução clássica de (1.1) consiste numa função u com derivadas contínuas até a ordem $2k$ e satisfazendo as condições de contorno. Logo, a solução u pertence ao conjunto das funções admissíveis $V = \{v \in C^{2k}(\Omega) \mid Bv = 0 \text{ em } \Gamma\}$. No entanto, quando A e B satisfazem certas propriedades, demonstra-se que a solução u também pode ser obtida pela minimização de um funcional quadrático $f : V \rightarrow \Re$ da seguinte forma,

$$f(u) = \frac{1}{2}a(u,v) - l(v) \quad v \in V \tag{1.2}$$

Em (1.2), $a(\cdot,\cdot)$ e $l(\cdot)$ são formas bilinear e linear, respectivamente. Portanto, para $u, v, w \in V$ e $\forall \alpha_l, \beta_l \in \Re$ tem-se que,

$$\begin{aligned} a(\alpha_l u + \beta_l v, w) &= \alpha_l a(u,w) + \beta_l a(v,w) \\ a(w, \alpha_l u + \beta_l v) &= \alpha_l a(w,u) + \beta_l a(w,v) \\ l(\alpha_l u + \beta_l v) &= \alpha_l l(u) + \beta_l l(v) \end{aligned} \tag{1.3}$$

De (1.1), verifica-se que a solução $u \in V$ deve ser suficientemente regular com continuidade até a ordem $2k$. Visando contornar esta restrição, pode-se completar o conjunto das funções admissíveis V para um espaço de Sobolev contendo além das funções em V, outras com suavidade menor que $2k$. Observa-se que o procedimento de completar V é de fundamental importância no MEF, onde a solução do PVC é aproximada por uma combinação linear de funções admissíveis obtidas por este procedimento [2].

[1] Neste texto, será mantida a denominação Multigrid ao invés de traduzi-la como Múltiplas Malhas ou Várias Malhas.

Para isso, seja $L_2(\Omega)$ o espaço linear das funções quadrado-integráveis no sentido de Lebesgue, isto é, $u \in L_2(\Omega)$ se as integrais $\int u\,d\Omega$ e $\int u^2\,d\Omega$ são convergentes e finitas. Observa-se que $L_2(\Omega)$ é um espaço de Hilbert com o seguinte produto interno e respectiva norma,

$$\begin{aligned}(u,v)_{L_2(\Omega)} &= \int_\Omega uv\,d\Omega \quad u,v \in L_2(\Omega)\\ ||u||_{L_2(\Omega)} &= \sqrt{(u,u)}\end{aligned} \tag{1.4}$$

Considere o vetor de índices $i = (i_1, \ldots, i_N)$ e a seguinte notação,

$$D^i u = \frac{\partial^{|i|} u}{\partial x_1^{i_1}, \ldots, \partial x_N^{i_N}} \tag{1.5}$$

onde $|i| = i_1 + \ldots + i_N$.

A partir de (1.5), define-se o seguinte produto interno e norma [88],

$$(u,v)_k = \sum_{|i|\le k} \int_\Omega D^i u\, D^i v\,d\Omega \quad u,v \in \tilde{C}^k(\Omega) \tag{1.6}$$

$$||u||_k = \sqrt{(u,u)_k} = \left(\sum_{|i|\le k} \int_\Omega (D^i u)^2\,d\Omega\right)^{\frac{1}{2}} \quad u \in \tilde{C}^k(\Omega) \tag{1.7}$$

sendo $\tilde{C}^k = \{v \in C^k(\Omega);\ \int_\Omega D^i v\,d\Omega < \infty\ (0 \le |i| \le k);\ Bv = 0 \text{ em } \Gamma\}$. Nas expressões anteriores, $|i| \le k$ significa que se deve considerar todos os diferentes vetores i para os quais $|i| = i_1 + \ldots + i_N \le k$. O espaço constituído pelo conjunto $\tilde{C}^k$ com o produto interno (1.6) e norma (1.7) será denotado por $S^k(\Omega)$.

De (1.4) e (1.7) verifica-se que,

$$||u||_k^2 = \sum_{|i|\le k} \int_\Omega (D^i u)^2\,d\Omega = \sum_{|i|\le k} ||D^i u||_{L_2(\Omega)}^2 \tag{1.8}$$

Logo, o quadrado da norma de $u \in \tilde{C}^k(\Omega)$ é a soma dos quadrados das normas de u e de todas as derivadas parciais até a ordem k em $L_2(\Omega)$.

Dada uma sequência de funções $\{u_n\}_{n=1}^\infty \subset S^k(\Omega)$ convergente na média para $u \in S^k(\Omega)$, ou seja,

$$\lim_{n\to\infty} ||u - u_n||_{S^k(\Omega)} = 0 \ \rightarrow\ \lim_{n\to\infty} u_n = u \tag{1.9}$$

observa-se que,

$$\lim_{n\to\infty} ||u - u_n||_{S^k(\Omega)} = 0 \ \rightarrow\ \lim_{n\to\infty} \sum_{|i|\le k} ||D^i u - D^i u_n||_{L_2(\Omega)} = 0 \tag{1.10}$$

Portanto, a sequência $\{u_n\}$ converge no espaço $S^k(\Omega)$ para o elemento u se e somente se as sequências das funções $\{u_n\}$ e das derivadas correspondentes $\{D^i u_n\}$ $(|i| \le k)$ convergem em $L_2(\Omega)$, respectivamente, para u e suas derivadas $D^i u$ [88]. Assim,

$$\lim_{n\to\infty} u_n = u \text{ em } S^k(\Omega) \iff \lim_{n\to\infty} D^i u_n = D^i u \text{ em } L_2(\Omega)\ \forall\ |i| \le k \tag{1.11}$$

Supondo que a sequência $\{u_n\}$ é fundamental em $S^k(\Omega)$, isto é,

$$\lim_{m,n\to\infty} ||u_m - u_n||_{S^k(\Omega)} = 0 \tag{1.12}$$

verifica-se que a mesma não é necessariamente convergente em $S^k(\Omega)$, visto que este espaço não é completo. No entanto, $\{u_n\}$ é fundamental em $S^k(\Omega)$ se e somente se todas as sequências $\{D^i u_n\}$ ($|i| \leq k$) são fundamentais em $L_2(\Omega)$,

$$\lim_{m,n\to\infty} ||u_m - u_n||_{S^k(\Omega)} = 0 \iff \lim_{m,n\to\infty} ||D^i u_n - D^i u||_{L_2(\Omega)} \ \forall\, |i| \leq k \tag{1.13}$$

Para o caso onde $\{u_n\}$ é fundamental e $u \notin S^k(\Omega)$, verifica-se a partir de (1.13) que cada uma das sequências $\{D^i u_n\}$ é fundamental em $L_2(\Omega)$. Como $L_2(\Omega)$ é completo, as sequências $\{D^i u_n\}$ possuem limites $v^{(i)}$ neste espaço, ou seja,

$$\lim_{n\to\infty} D^i u_n = v^{(i)} \text{ em } L_2(\Omega) \tag{1.14}$$

Para $|i| = 0$, tem-se o elemento limite da sequência $\{u_n\}$,

$$\lim_{n\to\infty} u_n = u \text{ em } L_2(\Omega) \tag{1.15}$$

As funções $v^{(i)}$ são unicamente determinadas por u e denominadas derivadas generalizadas. Assim, se u é suficientemente suave, os conceitos de derivada generalizada e clássica coincidem. O espaço de Hilbert H^k é construído completando-se $S^k(\Omega)$ com as funções obtidas pelos processos de limite (1.14) e (1.15).

Empregando-se a notação (1.5), o PVC (1.1) pode ser reescrito como [88],

$$\sum_{|i|,|j|\leq k} (-1)^{|i|} D^i(a_{ij} D^j u) = b \tag{1.16}$$

onde os coeficientes a_{ij} são funções mensuráveis no sentido de Lebesgue [78, 88].

Multiplicando a equação anterior por $v \in \bar{V} = \{v;\, v \in H^k(\Omega), Bv = 0 \text{ em } \Gamma\}$ em ambos os lados, integrando e aplicando o teorema de Green para o termo do lado esquerdo, chega-se a seguinte expressão,

$$\sum_{|i|,|j|\leq k} \int_\Omega a_{ij} D^i u D^j v \, d\Omega = \int_\Omega bv \, d\Omega \tag{1.17}$$

ou ainda,

$$a(u, v) = l(v) \tag{1.18}$$

A expressão anterior é a condição de mínimo do funcional (1.2). Em (1.17), observam-se as formas bilinear $a(u, v)$ e linear $l(v)$. A equação (1.1) representa a forma forte do PVC, enquanto (1.18) é a forma fraca ou variacional correspondente.

Na equação (1.17), as derivadas podem ser no sentido clássico, caso u e v sejam suficientementes suaves, ou generalizadas. Supondo $b \in L_2(\Omega)$, define-se como solução fraca de (1.1), a função $\hat{u} \in \bar{V}$ satisfazendo (1.17) para todo $v \in \bar{V}$. Observa-se que não se exige suavidade para as funções u, v e b, assim como para os coeficientes a_{ij}, basta que sejam quadrado-integráveis. No entanto, caso estes termos tenham regularidade suficiente, a solução fraca coincide com a solução clássica de (1.1).

Teorema de Lax-Milgran : se além das condições (1.3), as formas $a(u,v)$ e $l(v)$ possuem as seguintes propriedades:

limitada : existem constantes ρ_l e γ_l tal que,

$$|a(u,v)| \leq \rho_l \, ||u||_k \, ||v||_k \quad |l(v)| \leq \gamma_l \, ||v||_k \quad u,v \in \bar{V} \tag{1.19}$$

coerciva ou v-elíptica : existe uma constante $\rho_c > 0$ tal que,

$$a(u,u) \geq \rho_c ||u||_k^2 \tag{1.20}$$

então este teorema [2, 88] garante a existência e a unicidade da solução fraca $\hat{u} \in \bar{V}$ tal que,

$$a(\hat{u},v) = l(v) \quad \forall \, v \in \bar{V} \tag{1.21}$$

Além disso, se a forma $a(\cdot,\cdot)$ é simétrica, ou seja,

$$a(u,v) = a(v,u) \quad \forall u,v \in \bar{V} \tag{1.22}$$

então a função $\hat{u}$ é o único mínimo global do funcional (1.2).

Para o problema de elasticidade linear, a equação diferencial ou forma forte é dada por,

$$\operatorname{div} \mathbf{T}(u) = f_b \tag{1.23}$$

onde $\mathbf{T}(u)$ é o tensor de Cauchy e f_b representa as forças de corpo.

A seguinte relação constitutiva é válida,

$$\mathbf{T}(u) = D\nabla^s u \tag{1.24}$$

sendo D e $\nabla^s(\cdot)$ os tensores de elasticidade de Green e de deformação, respectivamente.

A forma bilinear correspondente a equação (1.23) é dada por,

$$a(u,v) = \int_\Omega D\nabla^s u \, \nabla^s v \, d\Omega = \int_\Omega D^{-1}\mathbf{T}(u) \cdot \mathbf{T}(v) \, d\Omega \tag{1.25}$$

Em geral, torna-se impossível obter a solução clássica u de (1.23), devido a fatores como a forma do domínio, as condições de contorno e propriedades do material. Desta maneira, busca-se uma solução aproximada u_h através de algum tipo de discretização. Supondo que as condições do teorema de Lax-Milgram sejam satisfeitas, no método de Ritz-Galerkin seleciona-se um subespaço $\bar{V}_h \subset \bar{V}$ de dimensão finita N. Como $\bar{V}_h$ é um espaço de Hilbert, todas as condições do teorema são sastisfeitas ao se substituir $\bar{V}$ por $\bar{V}_h$. Logo, a partir de (1.21) tem-se que,

$$a(u_h,v_h) = l(v_h) \leftrightarrow \min_{u \in \bar{V}_h} f(u) = f(u_h) \quad \forall v_h \in \bar{V}_h \tag{1.26}$$

Tomando-se uma base $\{\phi_i\}_{i=1}^N$ para $\bar{V}_h$, ou seja, um conjunto de N funções linearmente independentes, qualquer $w \in \bar{V}_h$ possui a seguinte expansão,

$$w = \sum_{i=1}^{N} w_i \phi_i \quad w_i \in \Re \tag{1.27}$$

Substituindo (1.27) no funcional quadrático (1.2) e utilizando a linearidade de $a(\cdot,\cdot)$ e $l(\cdot)$, obtém-se [2],

$$\begin{aligned} f(u) &= \frac{1}{2}a\left(\sum_{i=1}^{N} u_i\phi_i, \sum_{j=1}^{N} u_j\phi_j\right) - l(\sum_{i=1}^{N} u_i\phi_i) \\ f(u) &= \frac{1}{2}\sum_{i,j=1}^{N} u_i u_j\, a(\phi_i,\phi_j) - \sum_{i=1}^{N} u_i l(\phi_i) \\ F(\mathbf{u}) &= \frac{1}{2}\mathbf{u}^T\mathbf{A}\mathbf{u} - \mathbf{u}^T\mathbf{b} \end{aligned} \tag{1.28}$$

onde,

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} l(\phi_1) \\ l(\phi_2) \\ \vdots \\ l(\phi_N) \end{bmatrix} \quad \mathbf{A} = \begin{bmatrix} a(\phi_1,\phi_1) & a(\phi_1,\phi_2) & \dots & a(\phi_1,\phi_N) \\ a(\phi_2,\phi_1) & a(\phi_2,\phi_2) & \dots & a(\phi_2,\phi_N) \\ \vdots & \vdots & \dots & \vdots \\ a(\phi_N,\phi_1) & a(\phi_N,\phi_2) & \dots & a(\phi_N,\phi_N) \end{bmatrix} \tag{1.29}$$

Observa-se que a matriz $\mathbf{A}$ é simétrica devido a condição (1.22). Como $a(\cdot,\cdot)$ é coerciva tem-se que,

$$\mathbf{u}^T\mathbf{A}\mathbf{u} = a(u,u) \geq \rho_c\,||u||_k^2 \geq 0 \quad \rho_c > 0$$

verificando-se que $\mathbf{A}$ é positiva-definida visto que $\mathbf{u}^T\mathbf{A}\mathbf{u} = 0$ se somente se $\mathbf{u} = \mathbf{0}$.

Como $\min_{u\in\bar{V}_h} f(u) = \min_{\mathbf{u}\in\mathbb{R}^N} F(\mathbf{u})$, a condição de mínimo da expressão (1.28) resume-se ao sistema de equações,

$$\mathbf{A}\mathbf{u} = \mathbf{b} \tag{1.30}$$

Considerando que os domínios dos problemas exato e aproximado coincidem, e como $\bar{V}_h \subset \bar{V}$, a equação (1.18) é válida para todo $v_h \in \bar{V}_h$. Assim, subtraindo (1.26) de (1.18) com $v = v_h$, obtém-se,

$$a(u - u_h, v_h) = 0 \quad \forall v_h \in \bar{V}_h \tag{1.31}$$

ou seja, o erro $e = u - u_h$ é ortogonal, no sentido da forma bilinear $a(\cdot,\cdot)$, à aproximação u_h.

No MEF, o domínio Ω é dividido em subdomínios elementares Ω_e e tomam-se polinômios por partes como funções de base.

Como mencionado anteriormente, neste trabalho, serão apresentados métodos numéricos diretos, iterativos e multigrid para a solução do sistema de equações (1.30). Além disso, visando melhorar a solução aproximada u_h, considera-se um algoritmo para estimar o erro e para problemas coercivos. Acoplam-se então métodos multigrid e estimadores de erro definindo-se técnicas multigrid adaptáveis onde as diversas malhas são geradas aplicando-se um estimador de erro.

1.2 Métodos Numéricos para a Solução de Sistemas de Equações

Para resolver o sistema de equações (1.30), pode-se empregar métodos diretos e iterativos. No primeiro caso, a solução é obtida através de um número determinado de operações, modificando-se os coeficientes da matriz $\mathbf{A}$. Já os métodos iterativos, a partir de uma solução inicial $\mathbf{u}^{(0)}$, realizam

iterações sucessivas até que se atinja a solução dentro de uma precisão especificada. Ambas as técnicas possuem vantagens e desvantagens [2].

Tradicionalmente, os métodos diretos têm sido empregados para sistemas de ordem moderada, devido a maior eficiência computacional dos mesmos. No entanto, a medida que se aumenta o número de equações, os métodos iterativos tornam-se competitivos tanto em relação ao número de operações para a solução de (1.30), quanto ao espaço para armazenamento da matriz **A** e vetores auxiliares.

Assim sendo, no Capítulo 2 apresentam-se resultados da aplicação de métodos diretos e iterativos para a solução de sistemas de equações, obtidos da aplicação do MEF a problemas elásticos lineares bidimensionais. Estes resultados estão na forma do custo computacional e área de memória.

Discutem-se, inicialmente, estruturas de dados em colunas ascendentes e esparsa para a matriz **A**. Posteriormente, apresenta-se uma revisão dos métodos direto de Gauss, iterativos estacionários (Jacobi, Gauss-Seidel, SOR e SSOR) e de aceleração polinomial (Chebyshev e Gradiente Conjugado). Observa-se que uma vasta literatura está disponível na área de métodos diretos e iterativos, podendo-se citar [15, 16, 59, 82, 92, 102]. O Capítulo 2 está baseado principalmente nas referências [15, 16, 59, 82].

A motivação neste caso foi apresentar várias técnicas de solução do sistema (1.30), visando a posterior comparação dos mesmos com as técnicas multigrid discutidas no Capítulo 4.

1.3 Estimadores de Erros

Segundo [116], com a crescente aplicação de softwares de elementos finitos na resolução de problemas de engenharia, três preocupações relacionadas à utilização do método tem-se tornado relevantes:

- conhecimento;
- robustez;
- confiabilidade.

Em relação ao conhecimento, a preocupação dirigi-se ao usuário do pacote. É fundamental que o mesmo conheça claramente o problema físico a ser estudado, tenha um bom domínio das ferramentas disponíveis para efetuar a análise (hardware + software) e noções básicas dos princípios de mecânica. Esses requisitos dirigidos ao usuário reduzem as chances de se cometer erros graves, como por exemplo a definição das condições de contorno e tipo dos elementos, problemas comuns da análise por elementos finitos. Possibilitam ainda ao usuário maior clareza na modelagem e interpretação dos resultados.

A preocupação quanto à robustez dirige-se ao software empregado. O pacote deve garantir a resolução de problemas, sem que ocorram erros fatais e falhas do programa que inviabilizem ou forneçam respostas incoerentes. Também é interessante que o software não exija do usuário excessivo conhecimento numérico, dificultando a sua aplicação.

Finalmente, o aspecto da confiabilidade diz respeito à expressividade dos resultados obtidos. Uma malha de elementos finitos muito refinada pode levar a resultados com uma precisão maior do que a requerida. Neste caso, o tempo de processamento pode ser muito grande, necessitando-se ainda de um espaço de memória considerável. Por outro lado, uma malha grossa pode implicar num erro elevado, tornando os resultados inexpressivos. Daí surge a motivação de se trabalhar com estimadores de erros e malhas adaptáveis. Os estimadores de erros permitem, a partir de uma análise inicial, determinar o erro relacionado aos resultados obtidos. Então, com base nestes dados, a malha é refeita (total ou parcialmente) com o objetivo de reduzir e distribuir uniformemente o erro referente aos resultados. O processo é iterativo até que o erro esteja dentro da precisão requerida ou se atinja um número máximo de refinamentos especificado.

O refinamento da malha pode ser feito de 4 formas:

refinamento r : os nós da malha são simplesmente realocados;

refinamento h : envolve a redefinição no tamanho dos elementos;

refinamento p : mantém-se a mesma malha, aumentando-se, no entanto, o grau das funções de forma dos elementos, geralmente através de polinômios hierárquicos;

refinamentos mistos : são composições dos tipos anteriores, como por exemplos, refinamentos *hp* e *ph*.

O problema básico de uma análise adaptável por elementos finitos pode ser definido a partir da equação (1.31). Dada uma uma tolerância $\bar{\eta} > 0$, deve-se encontrar uma malha com solução u_h, tal que uma certa norma do erro $||e||$ satisfaça a relação,

$$||e|| = ||u - u_h|| \leq \bar{\eta} \tag{1.32}$$

Um algoritmo genérico de análise adaptável pode ser resumido como [103]:

1. para $k = 0$, definir uma malha $\mathcal{T}_k = \mathcal{T}_0$ representando razoavelmente a geometria do problema;
2. resolver o problema discreto para $\mathcal{T}_k$;
3. para todo elemento $T \in \mathcal{T}_k$, calcular a estimativa *a-posteriori* do erro;
4. se o erro global estimado é aceitável, então interrompe-se o processo. Caso contrário, refinam-se os elementos necessários, definindo uma nova malha $\mathcal{T}_{k+1}$. Retorna-se ao passo 2, substituindo k por $k + 1$.

Como é de conhecimento, as singularidades de um problema afetam a precisão da solução aproximada. Assim, torna-se importante não apenas identificar as regiões singulares do domínio, mas também obter um balanço razoável entre as regiões refinadas e as demais áreas visando atingir uma precisão ótima.

Uma outra questão fundamental é a determinação de estimativas confiáveis para a solução aproximada. Para isso, pode-se empregar estimativas de erro *a-priori* provenientes da análise de erros em elementos finitos. Porém, para este caso, tem-se apenas um comportamento assimptótico do erro e assume-se regularidade da solução, impossibilitando o tratamento de singularidades. Desta maneira, aplicam-se os estimadores *a-posteriori* obtidos a partir da solução u_h e dos parâmetros do problema.

Naturalmente, o custo do estimador *a-posteriori* deve ser inferior ao da análise. Além disso, o estimador deve ser local e fornecer limites inferior e superior para a norma do erro. Observa-se que limites superiores globais são suficientes para se obter uma solução numérica com uma precisão menor que uma tolerância especificada. Já os limites inferiores locais asseguram que a malha está corretamente refinada, viabilizando a obtenção de uma solução numérica para uma dada precisão empregando um número mínimo de pontos [103].

Para problemas coercivos, os estimadores *a-posteriori* são, em certo sentido, equivalentes e fornecem limites inferior e superior para o erro de discretização em elementos finitos, sendo classificados como [103]:

métodos de ponderação : a partir da solução aproximada u_h, emprega-se uma técnica local de extrapolação ou ponderação, visando determinar melhores estimativas para as derivadas da solução [112];

técnicas de resíduo : determinam o erro a partir de uma norma adequada de u_h em relação a forma forte do PVC [4, 5];

solução de problemas locais : resolvem problemas discretos locais mais simples que o problema original, estimando o erro a partir de normas apropriadas das soluções locais [5];

bases hierárquicas : calculam o resíduo de u_h com respeito a uma outra base de ordem mais alta ou em relação a uma malha mais refinada [114].

No Capítulo 3, considera-se o estimador de erro Zhu-Zienckiewicz para problemas coercivos, baseado num método de ponderação, sendo aqui denominado ZZ [112]. De forma geral, este estimador determina uma melhor aproximação para a derivada de u_h. A partir daí, calcula-se uma estimativa global pela superposição de erros locais nos elementos, possibilitando identificar aqueles elementos a serem refinados.

Uma análise mais rigorosa deste algoritmo supondo suavidade na solução do PVC está apresentada em [3]. Tem-se ainda uma classe de estimadores de projeção empregando produtos internos com ponderação, permitindo alcançar um comportamento assimptoticamente exato para os estimadores. Dentre algumas aplicações do procedimento ZZ, além de de elasticidade [112], destacam-se problemas de conformação [113], estratégias adaptáveis do tipo *hp* empregando polinômios hierárquicos [114] e flexão em placas [115]. Em [62], apresenta-se uma dedução analítica da matriz de rigidez de triângulos hierárquicos de 3 nós, além de expressões para o estimador de erro baseado em ZZ.

Observa-se que atualmente os procedimentos adaptáveis têm se tornado fundamentais na aplicação do MEF em problemas de engenharia. Assim, torna-se essencial validar os vários estimadores de erro disponíveis, identificando as suas capacidades e limitações. Nas referências [7, 8], apresentam-se metodologias computacionais para esta finalidade no caso de problemas lineares elípticos.

A qualidade e a confiabilidade da estimativa de erro para as técnicas de ponderação dependem fundamentalmente do critério adotado para recuperar as derivadas da solução aproximada. Em [111], apresentam-se algumas estratégias para esta finalidade, considerando um problema unidimensional, sendo deduzidos estimadores baseados em resíduos idênticos aquele dado em [6]. Desenvolveram-se vários outros recuperadores de derivadas ou tensões [44, 105, 106, 112, 117], os quais serão apresentados no Capítulo 3. Neste caso, o objetivo principal deste trabalho é realizar uma comparação de alguns recuperadores e aplicar o estimador ZZ juntamente com algoritmos multigrid.

1.4 Métodos Multigrid

Uma forma de aumentar o desempenho dos métodos iterativos é utilizar uma melhor aproximação inicial. Para isso, pode-se realizar iterações numa malha grossa e transferir o resultado para a malha original adotada na solução do problema considerado. Como a malha grossa possui um número menor de variáveis, o custo de uma iteração é bem inferior comparado com aquele da malha fina. Esta estratégia foi apresentada em [96], sendo denominada relaxação em grupo, tendo-se a sua origem já em 1920.

O procedimento anterior considerava apenas duas malhas, sendo por isso conhecido como método de 2 níveis. A idéia de se trabalhar com vários níveis foi introduzida em [45], sob o ponto de vista teórico, sendo posteriormente generalizada para problemas elípticos de segunda ordem em [9]. No entanto, a eficiência alcançada era ainda inferior em relação a outros algoritmos empregados na mesma época.

Os primeiros procedimentos multigrid completos estão descritos em [28, 75] para o caso de diferenças finitas. Obteve-se um custo para a solução dos sistemas de equações proporcional ao número de

variáveis N, ou seja, com uma ordem ótima $\mathcal{O}(N)$ para N grande. Esta mesma ordem foi demonstrada em [11, 56, 76] para problemas elípticos tratados com elementos finitos, estando alguns aspectos práticos, incluindo exemplos singulares e indefinidos, dados em [77]. Já em [29], há uma síntese de vários aspectos dos métodos multigrid, incluindo uma revisão histórica mais detalhada, tendo-se tornado uma referência padrão.

Desde então, verifica-se um grande número de trabalhos em métodos multigrid, abordando tanto aspectos formais quanto aplicações, não apenas na solução de PVC's, assim como reconhecimento de imagens, controle, otimização, dentre outras áreas. Alguns livros estão disponíveis tais como [31, 57, 58, 69, 70, 79]. Além disso, tem-se a rede MGNet (ftp casper.cs.yale.edu) armazenando artigos, teses, programas, informações gerais, além de um arquivo com uma grande quantidade de referências, constantemente atualizado.

No que se refere a solução de PVC via técnicas multigrid, utilizam-se várias malhas de elementos ao invés de uma única malha como nos métodos numéricos convencionais. O objetivo básico é acelerar a taxa de convergência dos métodos iterativos estacionários, sendo os métodos multigrid classificados como procedimentos de aceleração não-polinomial. Observa-se que entre os algoritmos polinomiais, destacam-se Chebyshev e Gradiente Conjugado.

Os métodos multigrid possuem dois elementos principais, esquematizados na Figura 1.1, ou seja, *iterações aninhadas* e *correção de malha grossa*. Como pode ser observado na Figura 1.1, empregam-se operadores do tipo I_{2h}^{4h} e I_{4h}^{2h} para transferir informações entre as malhas.

Como mencionado anteriormente, uma melhor aproximação inicial para as técnicas iterativas pode ser obtida realizando relaxações preliminares numa malha mais grossa, mapeando o resultado como solução inicial para a malha adotada no estudo do problema. Para uma sequência de malhas, tem-se um processo recursivo, onde o resultado de iterações iniciais numa malha grossa é utilizada como aproximação na malha seguinte. Este procedimento é denominado *iterações aninhadas*.

Por outro lado, dada uma aproximação $\mathbf{v}$ para a solução $\mathbf{u}$ do sistema de equações (1.30), o erro algébrico é dado por,

$$\mathbf{e} = \mathbf{u} - \mathbf{v} \tag{1.33}$$

Em geral, $\mathbf{u}$ não é conhecido, impossibilitando utilizar a norma do erro $||\mathbf{e}||$ como critério de convergência. No entanto, o resíduo

$$\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{v} \tag{1.34}$$

pode ser calculado, indicando o quanto a aproximação $\mathbf{v}$ deixa de satisfazer a equação (1.30).

Verifica-se que o erro algébrico e o resíduo estão diretamente relacionados, pois a partir das expressões (1.33) e (1.34) tem-se que $\mathbf{r} = 0$ se e somente se $\mathbf{e} = 0$. Além disso, reescrevendo (1.34) como $\mathbf{A}\mathbf{v} = \mathbf{b} - \mathbf{r}$ e subtraindo de (1.30) vem que,

$$\mathbf{A}(\mathbf{u} - \mathbf{v}) = \mathbf{r} \;\rightarrow\; \mathbf{A}\mathbf{e} = \mathbf{r} \tag{1.35}$$

A expressão (1.35) é denominada equação do resíduo, indicando que o erro $\mathbf{e}$ satisfaz o mesmo conjunto de equações que a solução $\mathbf{u}$ quando $\mathbf{b}$ é substituído por $\mathbf{r}$.

Assim, dada a aproximação $\mathbf{v}$, obtida por algum método iterativo, pode-se calcular o resíduo $\mathbf{r}$ em (1.34). Para melhorar $\mathbf{v}$, determina-se o erro $\mathbf{e}$ via equação do resíduo (1.35) e corrigi-se a solução $\mathbf{u}$ como $\mathbf{u} = \mathbf{v} + \mathbf{e}$. Observa-se que o custo para a solução exata de (1.35) é da mesma ordem do sistema original (1.30), tornando inviável a aplicação do procedimento anterior. Entretanto, como será visto posteriormente, busca-se a solução na malha fina, empregando-se os demais níveis apenas como esquemas de correção, não havendo a necessidade de resolver de forma exata as respectivas equações de resíduo nestas malhas.

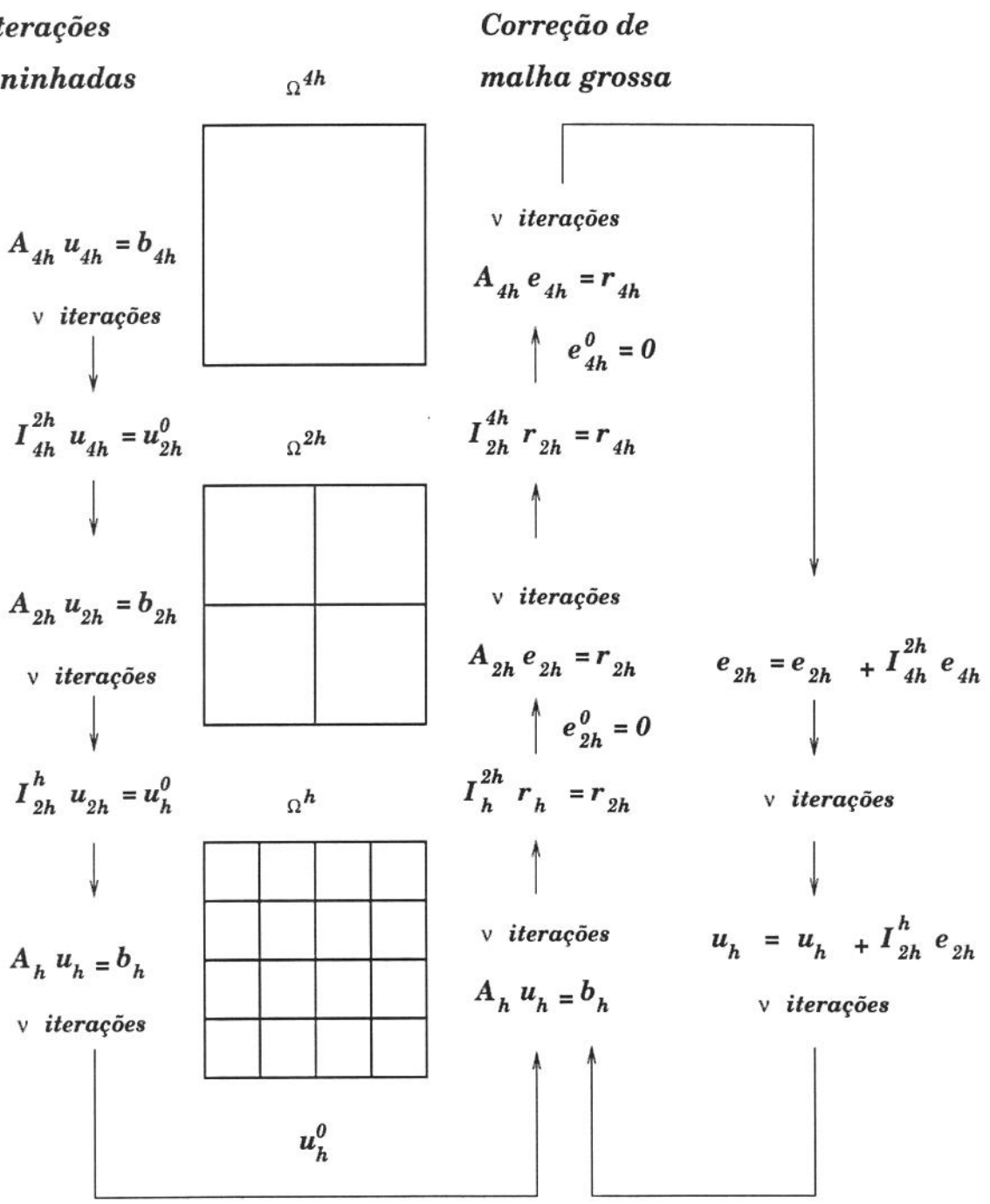
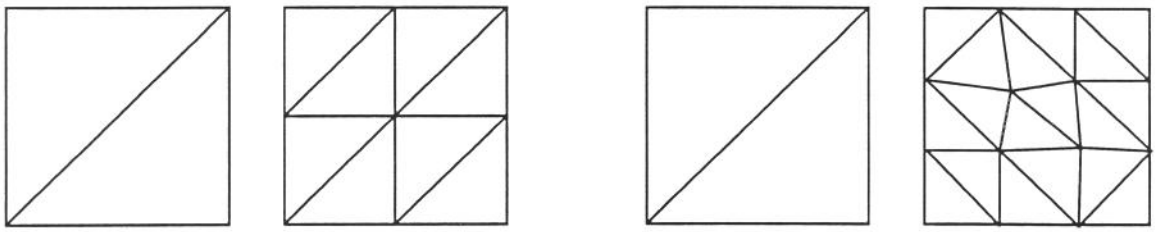


Figura 1.1: Elementos principais dos esquemas multigrid.



malhas aninhadas *malhas não-aninhadas*

Figura 1.2: Refinamentos em malhas aninhadas e não-aninhadas.

Este tipo de procedimento é adotado em métodos multigrid padrão, pois excetuando a malha mais fina, em todas as demais resolve-se uma equação de resíduo análoga a (1.35). Partindo-se da malha mais grossa, o erro **e** é utilizado como correção para o nível seguinte até atingir a malha mais fina. Esta estratégia é conhecida como *correção de malha grossa*.

Todo o desenvolvimento inicial em métodos multigrid e a maioria dos trabalhos até então publicados assumem que as malhas são aninhadas (*nested*). Esta condição está ilustrada na Figura 1.2 para o caso de elementos triangulares. Verifica-se que os nós do elemento sem refinamento estão presentes no triângulo refinado. De forma análoga, as malhas da Figura 1.1 também são aninhadas.

Esta hipótese permite simplificar os operadores de transferência entre malhas e vários teoremas garantindo a convergência dos esquemas multigrid, com ordem ótima de solução, foram demonstrados para alguns tipos de problemas [31, 57, 58, 69, 70, 79]. Além disso, como será visto no Capítulo 4, a partir dos operadores de transferência, tem-se uma relação variacional entre as matrizes dos sistemas de equações de cada nível.

Algoritmos adaptáveis para malhas aninhadas também estão descritos na literatura [12, 28, 29]. Em geral para casos bidimensionais, as técnicas de refinamento subdividem em 4 aqueles elementos nos quais se verificam erros pronunciados, como exemplificado na Figura 1.2. Algumas condições devem ser obedecidas para garantir a compatibilidade da malha e evitar a criação de elementos distorcidos [12, 29]. Para casos tridimensionais, esta tarefa é bem mais complexa, não sendo possível em geral a subdvisão de um tetraedro regular em 8 elementos também regulares [17, 108]. Uma outra linha de algoritmos refere-se aos métodos multigrid algébricos [30], onde se especifica apenas a matriz do sistema de equações a ser resolvido.

Em geral nos métodos multigrid, empregam-se funções de forma lineares nas aproximações por elementos finitos. Para aumentar a ordem de interpolação, utilizam-se algoritmos hierárquicos como em [48, 90]. Particularmente em [90], discute-se um procedimento baseado numa técnica de extrapolação. Partindo-se da malha grossa, aplicam-se refinamentos sucessivos, subdivindo-se cada elemento de forma regular como ilustrado na Figura 1.2. Apenas no caso de elementos com ângulos obtusos, a subdivisão é feita em 6 triângulos [90], permtindo obter espaços de aproximação realmente aninhados. Através do conceito de extrapolação, demonstra-se que a matriz de um elemento quadrático pode ser escrita como uma combinação das matrizes dos elementos lineares. Define-se uma transformação hierárquica, onde devido a forma de construção das malhas, as variáveis grossas estão presentes na malha fina. Além disso, os valores iniciais das incógnitas introduzidas no processo de refinamento, situadas no meio da aresta do elemento, são dadas pela média dos respectivos valores dos vértices do elemento na malha grossa. As relaxações são feitas de forma global em todas as variáveis de cada nível.

Isto permitiu introduzir técnicas multigrid completamente adaptáveis [91], não apenas no refinamento das malhas, mas também efetuando relaxações de forma local nos pontos onde se verifica uma

variação maior no resíduo, assim como no cálculo dos operadores. No entanto, aplicam-se nestes casos estruturas de dados mais complexas [64, 91].

Vários programas têm sido implementados permitindo tratar problemas lineares e não-lineares, considerando ainda estratégias adaptáveis em malhas não-uniformes, podendo-se citar por exemplo PLTMG [13] e KASKADE [36]. No entanto, estes programas trabalham com espaços de aproximação aninhados.

Desta forma, assume-se uma certa estrutura fixa para a malha fina, dificultando a aplicação de técnicas multigrid a problemas práticos de engenharia, os quais possuem contornos complexos. Para evitar esta limitação, pode-se empregar métodos algébricos ou malhas não-aninhadas, como ilustrado na Figura 1.2.

Esta última condição foi aplicada para problemas de escoamento de fluidos em [67, 68, 81]. Nestes trabalhos, dada a malha fina, geram-se automaticamente os níveis intermediários através de algoritmos frontais e de Delaunay. A partir daí, aplicam-se estratégias multigrid para a solução das equações de Poisson e Navier-Stokes. Assume-se que o número de variáveis da malha fina é suficientemente grande, tornando desprezível o custo de geração das malhas comparado com a solução do problema. Além disso, estruturas de dados adequadas são utilizadas para transferir informações entre as malhas.

A convergência dos métodos multigrid em malhas não-aninhadas foi inicialmente discutida em [37] e posteriormente em [27, 94, 108, 109]. Neste trabalho, as formas dos operadores e das malhas empregadas procuram respeitar as condições estabelecidas nestas referências, garantindo assim a convergência dos métodos multigrid, como será apresentado no Capítulo 4.

Algumas estratégias multigrid têm sido aplicadas em problemas de elasticidade [32, 42, 48, 61, 85]. Em [42], apresenta-se um método em dois níveis para elasticidade plana com malhas aninhadas. Discutem-se aspectos relativos a seleção do esquema e do fator de relaxação, a influência da forma e do número de nós do domínio, dos valores das propriedades do material e uma comparação com o método direto de Gauss.

Na referência [61], as estruturas são discretizadas com cubos quadráticos de 20 nós, gerando-se malhas aninhadas através de um critério adaptável. Para isso, toma-se a diferença entre as forças internas e de corpo em cada elemento, desprezando-se a descontinuidade de tensão, devido ao uso de elementos quadráticos. As relaxações são efetuadas de forma local, apenas nas variáveis correspondentes aos elementos refinados, através de um algoritmo de Gauss-Seidel modificado. Esta estratégia foi implementada no programa denominado LAME (*Local Adaptive Multigrid for Elasticity*).

A definição de malha grossa é substituída em [32] por um modelo de agregação baseado em conceitos de corpos rígidos. Dada a malha fina, a qual pode estar constituída por elementos de tipos distintos, efetua-se a divisão dos nós em grupos ou agregados, sem interseção entres os mesmos, estabelecendo-se variáveis independentes para cada agregado. Para o caso de multigrid, grupos de níveis intermediários são tratado como nós. Aplica-se como esquema de relaxação o procedimento de Gradiente Conjugado Quadrado [15, 32] com um pré-condicionador em vários níveis. Um programa computacional, denominado FEAGS (*Finite Element Aggregation Solver*), implementa este algoritmo de agregação em múltiplo níveis [32].

Em [48], apresenta-se uma estratégia multigrid adaptável com refinamento *hp*. Neste caso, gera-se uma malha com tetraedros lineares e resolve-se o respectivo sistema de equações de forma direta. Empregando funções hierárquicas, passa-se para uma malha quadrática incluindo nós no meio da aresta dos elementos lineares. Desta maneira, tem-se um algoritmo aninhado em dois níveis, sendo a malha grossa usada como correção para a quadrática, realizando-se nesta última relaxações do tipo SOR. O critério de parada está baseado no erro relativo entre duas soluções quadráticas sucessivas. Caso não haja convergência, emprega-se um estimador de erro através da energia de distorção da diferença entre as soluções linear e quadrática, refinando-se a malha linear de forma *h* e reinicializando o algoritmo.

Este método foi estendido e utilizado num sistema automático de análise denominado SAFES (*Self-*

Adaptive Finite Element System) [85], baseado numa decomposição espacial recursiva do domínio do problema. Basicamente, tem-se uma única estrutura de dados em árvores octaédricas para a geração da malha, análise e refinamento adaptável. Visto que cada folha da árvore representa um conjunto de elementos da malha, trata-se a mesma como uma subestrutura, permitindo efetuar uma condensação de variáveis nas folhas em cada nível topológico da árvore. Alcançando de forma recursiva o topo, tem-se a estrutura representada apenas em função dos nós de contorno. Este procedimento é usado na solução direta da malha linear. De forma análoga, desenvolveu-se uma variante hierárquica do método SOR aplicada a malha quadrática [84, 85].

Uma outra estratégia usando conceitos de agregação ou aglomeração com volumes finitos, visando a simulação de fluxos através da equação de Poisson com termos convectivos e difusivos, pode ser encontrada em [60]. No caso unidimensional, os coeficientes das matrizes dos vários níveis são obtidos através de um processo de Galerkin, onde duas linhas consecutivas são somadas, identificando-se as duas variáveis finas correspondentes com apenas uma incógnita da malha grossa seguinte. Esta idéia pode ser estendida para casos bi e tridimensionais. No entanto, no caso difusivo tem-se uma inconsistência, a qual é corrigida multiplicando-se os termos viscosos por um fator empírico. Uma aplicação de métodos multigrid em problemas de contato está discutida em [63].

Neste trabalho, consideram-se métodos multigrid aplicados a problemas elásticos com espaços de aproximação lineares e não-aninhados. Em relação às referências [67, 68, 81], tem-se operadores de transferência entre malhas distintos, além de estruturas de dados mais eficientes. Já nos trabalhos [32, 48, 61, 84], formas distintas de espaços aninhados estão presentes. Particularmente em [48, 61, 84], o emprego de malhas quadráticas das maneiras descritas, pode gerar sistemas de alta ordem sem garantias de se gerar uma malha ótima para a solução do problema com uma certa precisão, devido por exemplo ao critério de refinamento adotado em [48].

Observa-se que a presença de procedimentos realmente automáticos para a geração de malhas [35, 43] é de fundamental importância. Foi possível não apenas obter malhas adequadas para as estratégias multigrid, assim como tratar domínios com geometrias complexas. Em relação a algoritmos multigrid adaptáveis, pode-se partir da malha grossa e aplicar estimadores de erros, visando obter uma sequência de malhas mais adequadas para a solução do problema dentro de um certo erro percentual dado, automatizando o processo de análise.

Todos estes conceitos serão discutidos no Capítulo 4. Inicialmente, apresenta-se um estudo espectral do comportamento dos métodos iterativos estacionários. Posteriormente, discutem-se os principais elementos das técnicas multigrid, além de operadores de transferência e a recuperação das informações entre malhas, assim como alguns algoritmos, incluindo casos adaptáveis, e seus custos computacionais. Aspectos de convergência e uma forma variacional dos operadores são também considerados. Finalmente, resultados de aplicações bi e tridimensionais são apresentados e as performances comparadas com métodos direto e iterativos.

1.5 Programação por Objetos em Elementos Finitos

A implementação computacional dos métodos numéricos para a análise de problemas de engenharia é de fundamental importância. Sob o ponto de vista do usuário, o programa deve possuir uma interface simples de utilização, ser confiável e eficiente. No que se refere à implementação, desejam-se características como modulação, extensibilidade, fácil manutenção, dentre outras.

Apesar de vários progressos, a produtividade no desenvolvimento de softwares ainda é pequeno, comparado ao crescimento vertiginoso da indústria de hardware nos últimos tempos. Desta maneira, torna-se essencial disciplinar a implementação de programas aplicando por exemplo conceitos de engenharia de software [86]. No caso do meio científico, estes requisitos são importantes para uma maior qualificação dos programas, evitando a repetição de procedimentos em cada trabalho realizado. Desta

forma, através de uma maior organização, pode-se ter uma base sólida de programas, permitindo a rápida simulação no estudo de novos problemas.

O modelo de programação orientada por objetos [18] tem possibilitado aumentar a produtividade no desenvolvimento de programas. Alguns conceitos deste paradigma estão apresentados no Apêndice A. Basicamente, identificam-se os agentes principais da base de conhecimento a ser reprentada no programa. Estes agentes ou objetos possuem atributos e um conjunto de métodos descrevendo as características e os comportamentos de interesse. Assim, num programa tem-se um conjunto de objetos implementados através de classes, as quais constituem-se em moldes para a definição dos objetos como instâncias de uma classe. Por sua vez, duas instâncias diferem entre si pelos valores atribuídos as suas variáveis. Desta maneira, as classes são a unidade básica de modulação num sistema por objetos.

Os objetos se comunicam através do envio de mensagens, as quais especificam apenas a ação a ser feita, sendo responsabilidade do objeto receptor interpretá-la e executar o conjunto de operações correspondente, retornando o resultado para a instância que emitiu a mensagem. Logo, numa classe tem-se o encapsulamento da informação, a qual pode ser acessada apenas através do envio de uma mensagem para o método desejado. Observa-se ainda que a uma mesma mensagem pode corresponder diferentes procedimentos dependendo da classe receptora. Por exemplo, a operação + para matrizes representa uma adição usual, enquanto para um conjunto de caracteres poderia significar concatenação. Esta característica é denominada polimorfismo.

Além disso, tem-se o conceito de herança permitindo especializar a informação ao longo de uma hierarquia de classes, onde aquelas situadas em níveis mais baixos herdam as características (dados + métodos) daquelas classes situadas acima. Assim, é possível estender ou reutilizar classes em várias aplicações, bastando acrescentar apenas as variáveis e métodos necessários para descrever a especialização desejada.

A principal diferença do paradigma por objetos é a proximidade com os conceitos do mundo real a ser implementado no programa. Ao contrário do modelo por procedimentos, onde se isolam os comportamentos através de subrotinas, tem-se entidades próprias descrevendo as suas principais caracteríticas através de variáveis e métodos.

Estes conceitos têm sido aplicados na análise estrutural de problemas de engenharia, podendo-se citar alguns trabalhos iniciais [18, 47, 49, 50, 53, 73, 120]. Um dos primeiros programas implementados está discutido em [50], usando a linguagem Object NAP baseada em rotinas em C e Pascal. Em [73], apresentam-se conceitos gerais de programação por objetos, incluindo aspectos sobre concorrência, processamento distribuído e banco de dados, além de um programa para análise por elementos finitos implementado com uma extensão da linguagem LISP, denominada Flavors.

Conceitos do modelo por objetos aplicados ao MEF são também discutidos em [120], apresentando-se uma extensão da hierarquia de classes da linguagem Smalltalk para o caso do MEF. Detalhes de implementação deste programa protótipo estão dados em [40]. Devido a baixa eficiência do ambiente Smalltalk, desenvolveu-se em [41] uma versão em C++ [65] do mesmo programa. Uma extensão para o tratamento de problemas de plasticidade está dada em [72].

Aplicações da linguagem C++ com diferentes enfoques também podem ser encontradas em [87, 89, 93]. Em [107], tem-se uma hierarquia de classes para o tratamento de vários tipos de matrizes tais como simétrica, esparsa, coluna, dentre outras, além de uma série de classes para o tratamento de entidades do MEF. Já em [33], tem-se um ambiente em C++ constituído de um interpretador e um módulo de execução. Através de uma linguagem interpretada, pode-se especificar interativamente os parâmetros do modelo de elementos finitos e da solução. Exemplos de problemas lineares e não-lineares são resolvidos.

Em [1, 104] apresentam-se aspectos da programação por objetos aplicados a análise numérica, representação gráfica das classes, bancos de dados, incluindo ainda a aplicação destes conceitos na implementação de um programa para o tratamento de laminas de materiais compostos. Além disso,

aspectos de inteligência artificial são também abordados em conjunto com a análise estrutural. Esta temática também é discutida em [100, 101], apresentando uma série de ferramentas computacionais para automatizar o processo de análise, fornecendo uma base de conhecimentos e sistemas especialistas para auxiliar a especificação de parâmetros da análise, interpretação dos resultados, procedimentos adaptáveis, tratamento de erros, dentre outros. O objetivo básico é acumular um conjunto de conhecimentos de um especialista, auxiliando o usuário comum na simulação computacional de problemas mecânicos, tornando o ambiente de análise mais *inteligente*.

Um grande esforço no desenvolvimento de programas para engenharia estrutural tem sido realizado por pesquisadores do LNCC, começando com a linguagem Fortran [46], passando para C [53] e finalmente utilizando o modelo de programação orientada por objetos [35, 43, 47, 54] através da linguagem C++. De forma geral, tem-se bibliotecas de classes e procedimentos para bancos de dados, tratamento de erros, estruturas de dados, manipulação de matrizes e vetores, rotinas matemáticas, além de um conjunto de classes para análise linear por elementos finitos

A contribuição deste trabalho está relacionada a uma nova proposta de organização das classes para elementos finitos baseada no conceito de tipos parametrizados disponíveis em C++; a inclusão de matrizes esparsas juntamente com métodos de Gauss e iterativos; procedimentos multigrid para problemas bi e tridimensionais, além de alguns recuperadores de tensão. No Capítulo 5, apresentam-se as principais características das classes implementadas. Consideram-se ainda os sistemas SAFE [54] e SAT [51, 80, 98] para análise de problemas elásticos bidimensionais.

1.6 Contribuições do Trabalho

A partir do exposto nas seções anteriores, consideram-se os seguintes aspectos como contribuições principais deste trabalho:

- a comparação de métodos diretos e iterativos para sistemas de equações, tomando-se como exemplos problemas elásticos lineares bi e tridimensionais;
- a revisão de recuperadores de tensão aplicados em conjunto com o estimador ZZ;
- métodos multigrid em malhas não-estruturadas e não-aninhadas, possibilitando a solução de problemas com contornos complexos, alcançando uma ordem ótima de solução $\mathcal{O}(N)$ para problemas tridensionais;
- a definição de estratégias multigrid adaptáveis, onde partindo-se de uma malha grossa, torna-se possível obter uma solução aproximada com um erro percentual dado, sendo o custo de solução dos sistemas de equações inferior em relação aos métodos diretos e iterativos considerados neste trabalho;
- a aplicação de conceitos de programação por objetos em elementos finitos, permitindo desenvolver programas eficientes e de boa qualidade;
- a apresentação de ambientes de análise para problemas elásticos bidimensionais, integrando de forma efetiva ferramentas para a definição de contornos, geração automática de malhas, métodos de solução eficientes, análise adaptável e visualização de resultados.

Observa-se que todas as conclusões apresentadas ao longo do trabalho, referem-se a problemas lineares e elípticos, em particular o caso de elasticidade. Apesar dos algoritmos discutidos poderem ser aplicados a outros tipos de problemas, as conclusões não são necessariamente extensíveis de forma direta para outros casos, tornando-se importante realizar uma análise mais próxima do problema considerado.

Capítulo 2

MÉTODOS NUMÉRICOS PARA A SOLUÇÃO DE SISTEMAS DE EQUAÇÕES LINEARES

O objetivo principal deste capítulo é realizar uma análise comparativa entre alguns métodos diretos e iterativos para a solução do sistema de equações (1.30), tomando-se o número de operações e o espaço de memória. Como estes parâmetros dependem do tipo de armazenamento da matriz **A**, discutem-se, inicialmente, estruturas de dados para esta finalidade [2, 15, 16].

Nas seções 2.2 a 2.7, tem-se uma revisão dos métodos direto de Gauss, algoritmos iterativos estacionários (Jacobi, Gauss-Seidel, SOR, SSOR), procedimentos de aceleração polinomial (Chebyshev e Gradiente Conjugado) e critérios de convergência [2, 15, 16, 59, 82]. Para todos os métodos, consideram-se algoritmos simbólicos com a implementação dos mesmos, seguindo as notações dadas em [2, 15].

Apresentam-se resultados da aplicação destes métodos para 3 problemas planos. Foram geradas 4 malhas para cada um destes exemplos, tomando-se um número crescente de equações. Finalmente, considera-se um problema de fratura com forte singularidade discretizado por malhas com até mais de 55.000 variáveis.

2.1 Estruturas de Dados para a Matriz do Sistema de Equações

Visando alcançar eficiência na solução do sistema de equações (1.30), torna-se essencial empregar estruturas de dados adequadas para a matriz **A**. Deve-se procurar armazenar o número mínimo de elementos necessários para a resolução de (1.30), através de um método direto ou iterativo, reduzindo assim não apenas o espaço de memória, mas também o número de operações.

Como a matriz **A** é simétrica, armazena-se apenas a sua parte triangular superior ou inferior. No entanto, em geral, os elementos não-nulos de **A** estão confinados no interior de uma banda formada pelas diagonais principal e secundárias. Para uma matriz simétrica, denomina-se semibanda o conjunto de elementos na parte superior, ou seja, os termos a_{ij} tais que $0 < j - i < \beta$ onde β é a metade da largura de banda [82]. Desta maneira, no armazenamento em banda cada linha da matriz possui β elementos. Torna-se importante empregar algoritmos para a renumeração ótima dos nós, visando reduzir β e por conseguinte a demanda computacional em termos de memória e número de operações [2, 34, 52, 66, 82, 97].

A maioria dos elementos na banda da matriz são iguais a zero. Para contornar esta deficiência, utiliza-se o armazenamento em colunas ascendentes (*skyline*), onde a largura de banda é variável

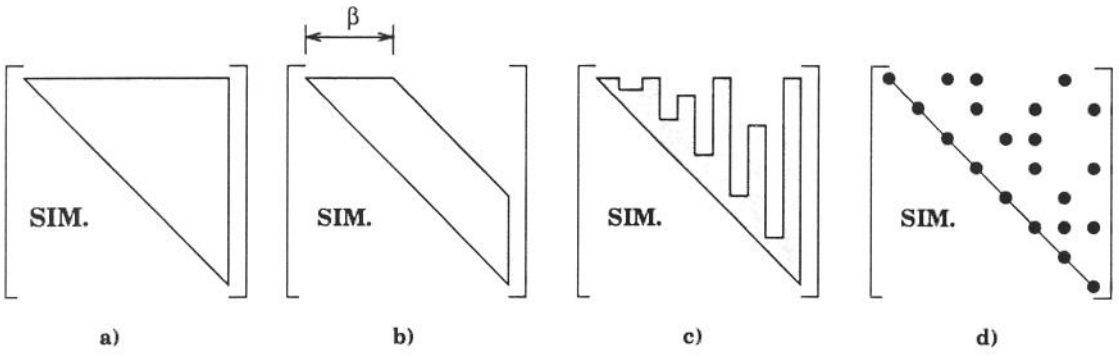


Figura 2.1: Armazenamento da matriz **A**: a) simétrica; b) banda; c) colunas ascendentes; d) esparsa.

A=

$$\begin{bmatrix} 10 & 30 & 50 & 0 & 90 \\ & 20 & 0 & 70 & 0 \\ & & 40 & 0 & 0 \\ & \text{SIM.} & & 60 & 0 \\ & & & & 80 \end{bmatrix}$$

colunas ascendentes:

maxa = [1 2 4 7 10 15]

a = [10 20 30 40 0 50 60 0 70 80 0 0 0 90]

esparsa:

diag = [1 5 7 8 9 10]

a = [10 30 50 90 20 70 40 60 80]

col = [1 2 3 5 2 4 3 4 5]

Figura 2.2: Estruturas de dados para matrizes em colunas ascendentes e esparsa.

[15, 16]. Entretanto, ainda a maior parte dos elementos são nulos. Devido a esparsidade das matrizes provenientes da aplicação do MEF, obtém-se um ganho significativo em eficiência utilizando-se estruturas de dados para matrizes esparsas, onde apenas os elementos não-nulos são considerados. Para isso, tem-se várias estruturas esparsas, cujas finalidades dependem das características do problema em estudo [15, 82].

A Figura 2.1 mostra todos os tipos de armazenamento da matriz **A** discutidos anteriormente. Neste trabalho, foram adotados os armazenamentos em colunas ascendentes e esparso, estando as estruturas de dados ilustradas na Figura 2.2 para um exemplo simples.

No caso de colunas ascendentes, tem-se os vetores **a** para os elementos da matriz e **maxa** com os indíces de **a** correspondentes aos termos da diagonal principal [16]. Este último vetor contém $N+1$ posições e a diferença entre os elementos $i+1$ e i ($i = 1, \ldots, N$) fornece o número de elementos m_i da coluna i. Por sua vez, a altura da coluna i, denotada por m_i^U, é dada por $m_i^U = m_i - 1$, ou seja, não se considera o elemento da diagonal principal.

Para matriz esparsa, utiliza-se o Armazenamento Comprimido por Linhas [2, 15]. Este esquema é bastante simples e armazena estritamente apenas os elementos não-nulos da matriz. São necessários um vetor **a** para os coeficientes, outro denominado **col** com os índices das colunas, além do vetor **diag**, análogo ao **maxa**, com os indíces de **a** correspondentes aos termos da diagonal principal da matriz. Uma das desvantagens deste formato é o endereçamento indireto para o acesso aos elementos de uma linha. Uma estrutura de dados análoga, denominada Armazenamento Comprimido por Colunas [15], é recomendada caso se deseje manipular as colunas ao invés das linhas da matriz.

Observa-se que o uso de métodos iterativos com matrizes esparsas não requer a renumeração ótima das equações. No entanto, para colunas ascendentes de forma geral e métodos diretos em matrizes esparsas é fundamental determinar uma numeração ótima, visando reduzir a demanda de memória e

processamento. No caso de colunas ascendentes, empregou-se o algoritmo de Cuthill-Mckee [34, 66]. Já para matrizes esparsas, utilizou-se a técnica de grau mínimo [52, 97].

Denota-se por m a altura média das colunas para a estrutura em colunas ascendentes ou o número de médio de elementos por linha na parte superior da matriz esparsa. Da mesma maneira, m_i^U denota a altura da coluna i ou o número de elementos na matriz esparsa desconsiderando o termo da diagonal principal. Estas constantes estão relacionadas por $m = \sum_{i=1}^{N} m_i^U/N + 1$.

Verifica-se que para as duas estruturas de dados consideradas, necessitam-se, respectivamente, $N + \sum_{i=1}^{N} m_i^U = Nm$ e $(N+1)/2$ posições de memória com precisão dupla para os vetores **a** e **maxa**/**diag**. Para matriz esparsa, tem-se ainda o vetor **col** de números inteiros ocupando a metade do armazenamento de **a**, ou seja, $Nm/2$ posições.

Neste trabalho, uma operação de ponto flutuante (*flop - floating operation*) será entendida como uma multiplicação/divisão seguida geralmente de uma adição/subtração.

2.2 Métodos Diretos

Os métos diretos para a solução de (1.30) estão baseados na eliminação de Gauss. Pode-se citar outras técnicas derivadas deste procedimento, tais como a decomposição de Cholesky e o algoritmo de Gauss-Jordan [16, 82].

No método de Gauss, as equações do sistema são modificadas em passos sucessivos até que a solução seja encontrada [82]. Para isso, a matriz **A** é fatorada como,

$$\mathbf{A} = \mathbf{L}'\mathbf{U} \tag{2.1}$$

onde $\mathbf{L}'$ e **U** são matrizes triangulares inferior e superior, respectivamente, sendo a diagonal de **U** unitária.

Frequentemente, a matriz $\mathbf{L}'$ é escrita como $\mathbf{L}' = \mathbf{LD}$, sendo **L** triangular inferior com diagonal unitária e **D** uma matriz diagonal. Além disso, como **A** é simétrica, tem-se que $\mathbf{U} = \mathbf{L}^T$. Logo, a partir de (2.1),

$$\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^T \tag{2.2}$$

Substituindo (2.2) em (1.30), obtém-se o sistema fatorado,

$$\mathbf{L}\mathbf{D}\mathbf{L}^T\mathbf{u} = \mathbf{b} \tag{2.3}$$

Denotando $\mathbf{w} = \mathbf{D}\mathbf{L}^T\mathbf{u}$, chega-se ao sistema triangular inferior,

$$\mathbf{L}\mathbf{w} = \mathbf{b} \tag{2.4}$$

cuja solução é obtida por um processo de substituição a frente.

Conhecido o vetor **w**, observa-se que a solução **u** de (1.30) é determinada por uma substituição para trás, ou seja,

$$\mathbf{L}^T\mathbf{u} = \mathbf{D}^{-1}\mathbf{w} \tag{2.5}$$

A fatoração de **A** pode ser realizada em N passos através das colunas ou linhas, ambas conduzindo ao mesmo resultado. No entanto, a primeira forma é mais conveniente para armazenamento em colunas ascendentes, enquanto a decomposição por linhas é empregada para matrizes esparsas. Observa-se que os elementos da diagonal são denominados pivôs na eliminação de Gauss. Como todos os termos de uma linha serão divididos pelo seu respectivo pivô, este elemento não deve ser nulo ou inferior a uma

certa precisão ξ especificada. Caso esta condição seja violada, tem-se uma singularidade e o algoritmo deve ser interrompido.

Na decomposição por colunas, o objetivo do passo k é eliminar todos os elementos não-nulos da coluna k situados abaixo da diagonal. No primeiro passo, inicialmente normaliza-se a linha 1 dividindo-se os termos não-nulos pelo pivô a_{11}. A eliminação é efetuada subtraindo múltiplos convevientes desta linha de todas as demais contendo um elemento não-nulo na coluna 1. Ao final obtém-se a matriz $\mathbf{A}$ com $a_{i1} = 0 \ \ (i > 1)$ e $a_{11} = 1$.

Assim, no inicío do passo k, os termos nas primeiras $k-1$ colunas abaixo da diagonal são nulos e iguais a 1 ao longo das mesmas posições da diagonal. Por exemplo, para $N = 5$, $k = 3$ e com x indicando um elemento não-nulo, tem-se,

$$\mathbf{A} = \begin{bmatrix} 1 & x & x & x & x \\ 0 & 1 & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & x & x & x \end{bmatrix}$$

Novamente, no passo k toma-se a_{kk} como pivô, normaliza-se a linha k e eliminam-se os elementos não-nulos abaixo da diagonal na coluna k. Continua-se este processo até o passo N, obtendo-se ao final uma matriz triangular superior com diagonal unitária.

O passo k da eliminação de Gauss é equivalente a multiplicar $\mathbf{A}$ por $\mathbf{L}_k^{-1}\mathbf{D}_k^{-1}$, onde,

$$\mathbf{L}_k = \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & \ddots & & & \\ & & & 1 & & \\ & & & l_{ik} & \ddots & \\ & & & l_{i+1,k} & & 1 \end{bmatrix} \qquad \mathbf{D}_k = \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & \ddots & & & \\ & & & d_{kk} & & \\ & & & & \ddots & \\ & & & & & 1 \end{bmatrix}$$

com $d_{kk} = a_{kk}$ e $l_{ik} = a_{ik} \ \ (i > k)$. Os termos l_{ik} são denominados fatores de multiplicação de Gauss. Para se obter $\mathbf{L}_k^{-1}$ basta inverter o sinal dos elementos fora da diagonal principal.

Assim, no passo N multiplicando-se $\mathbf{D}_N^{-1}\mathbf{A}$ por $\mathbf{L}_N^{-1} \equiv \mathbf{I}$ obtém-se,

$$\mathbf{L}_N^{-1}\mathbf{D}_N^{-1} \ldots \mathbf{L}_2^{-1}\mathbf{D}_2^{-1}\mathbf{L}_1^{-1}\mathbf{D}_1^{-1}\mathbf{A} = \mathbf{U} \tag{2.6}$$

A partir daí, recupera-se a decomposição (2.1) com $\mathbf{L}' = \mathbf{D}_1\mathbf{L}_1\mathbf{D}_2\mathbf{L}_2 \ldots \mathbf{D}_N\mathbf{L}_N$. Na implementação deste algoritmo, os elementos l_{ij} e d_{ii} são armazenados nas posições originais dos termos a_{ij} e a_{ii}, respectivamente. Além disso, no passo k os elementos a_{ij} são modificados da seguinte forma,

$$a_{ij} \leftarrow a_{ij} - l_{ik}u_{kj} \quad i,j > k \tag{2.7}$$

Portanto, a expressão completa para os termos a_{ij} é dada por,

$$a_{ij} \leftarrow a_{ij} - \sum_{m=1}^{k} l_{im}u_{mj} \quad i,j > k \tag{2.8}$$

Como já mencionado, a eliminação por linhas é mais eficiente para o caso de matrizes esparsas armazenadas por linhas. No início do passo k, os elementos da diagonal nas linhas 1 a $k-1$ são unitários, com coeficiente nulos à esquerda da diagonal. Para $N = 5$ e $k = 3$, tem-se por exemplo,

$$\mathbf{A} = \begin{bmatrix} 1 & x & x & x & x \\ 0 & 1 & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{bmatrix}$$

O passo k consiste na eliminação dos elementos não-nulos da linha k à esquerda da diagonal, subtraindo múltiplos convenientes de cada uma das linhas 1 a $k-1$. Normaliza-se então a linha k dividindo-se todos os elementos pelo valor da diagonal a_{kk}.

Uma forma precisa para estimar o número de operações na fatoração de **A** e nos processos de substituição está dada em [82], considerando o parâmetro m_i^U definido anteriormente. Tem-se, então,

Fatoração de A :

- multiplicações: $\sum m_i^U(m_i^U + 3)/2$
- adições: $\sum m_i^U(m_i^U + 1)/2$

Processos de substituição :

- multiplicações: $N + 2\sum m_i^U$
- adições: $2\sum m_i^U$.

Devido a definição de operação de ponto flutuante adotada, tem-se um total de $N + \sum m_i^U(m_i^U + 7)/2$ operações para a solução de (1.30) via método de Gauss. No que se refere ao espaço de memória, além da matriz **A**, necessitam-se mais N posições para o vetor **b**.

Na fatoração de **A**, ocorre em geral preenchimento (*fill in*), ou seja, elementos nulos tornam-se diferentes de zero devido à eliminação de termos nas linhas e colunas como apresentado anteriormente. Em colunas ascendentes, este fenômeno está confinado no perfil da matriz. Logo, todos os elementos estão alocados no início da fatoração.

Já para matrizes esparsas, a fatoração inclui novos termos os quais não estavam previstos inicialmente. Para isso, aplica-se, antes da fatoração, o processo denominado Gauss simbólico, o qual determina os termos que se tornarão não-nulos, permitindo então alocar espaço para estes novos elementos [82]. Este fenômeno pode ser visto no exemplo a seguir, onde ao se eliminar o elemento a_{21}, subtraindo a linha 1 da 2, o termo a_{24} torna-se não-nulo,

$$\begin{bmatrix} x & x & & x & \\ x & x & x & & x \\ & x & x & x & \\ x & & x & x & x \\ & x & & x & x \end{bmatrix} \rightarrow \begin{bmatrix} x & x & & x & \\ o & x & x & * & x \\ & o & x & x & * \\ o & + & o & x & x \\ & o & + & o & x \end{bmatrix}$$

tendo sido adotada a seguinte notação: x – elemento não-nulo; o – elemento na parte inferior a ser eliminado; $*$ – elemento introduzido pela eliminação; $+$ – elementos novos introduzidos na parte inferior devendo ser eliminados.

Assim de forma geral, dada uma linha i, se existe um elemento $a_{ij} \neq 0 \ \ (j > i)$, deve-se eliminar a_{ji} na parte triangular inferior de **A**. Caso exista na mesma linha i, o termo $a_{ik} \neq 0 \ \ (k > j)$ e por sua

vez $a_{jk} = 0$, então ocorrerá preenchimento e o espaço para o elemento a_{jk} deverá ser alocado, mesmo que resulte igual a zero, antes de se iniciar a eliminação de Gauss. O valor assumido por a_{jk} é dado por,

$$a_{jk} = -\frac{a_{ij}}{a_{ii}} a_{ik} \tag{2.9}$$

O algoritmo abaixo apresenta a decomposição de Gauss por linhas numa matriz esparsa, supondo já realizado o procedimento simbólico. Observa-se a presença do vetor **Eqs** com N posições, armazenando para as colunas presentes na linha i, a posição correspondente no vetor **col**. Assim, por exemplo, para a linha 10, o elemento $a_{10,10}$ é o primeiro elemento, sendo armazenado o valor 1 na posição 10 de **Eqs**. Isto facilita o acesso aos elementos por coluna, evitando um procedimento de busca.

```
for i = 1, N
    NumElemLineI = diag(i + 1) - diag(i)
    d_i = diag(i)
    a_ii = a(d_i)
    if (a_ii < ξ)
      return(i)  →  pivô invalido
    for m = 2, NumElemLineI
        d_ij = d_i + m - 1
        j =col(d_ij)
        a_ij =a(d_ij)
        cte = a_ij/a_ii
        d_j =diag(d_ij)
        a_jj =a(d_j)
        a_jj = a_jj - a_ij*cte
        if (j + 1 ≤ NumElemLineI)
          NumElemLineJ = diag(d_j + 1) - diag(d_j)
          for k = 2, NumElemLineJ
              col_kj = col(d_j + k - 1)
              Eqs(col_kj)= k
          end
          for l = 2, NumElemLineI-1
              k =col(d_i + l)
              d_k =Eqs(k)
              a_ik =a(d_i + l)
              a_jk =a(d_j + d_k - 1)
              a_jk = a_jk - a_ij*cte
          end
          a_ij =cte
        end
    end
end
```

Em geral no método de Gauss, a tarefa de triangulação da matriz **A** é responsável pelo maior esforço computacional. No entanto, para problemas onde o número de vetores independentes **b** é superior a cerca de $\frac{1}{4}m$, o número de operações para o processo de substituição é superior aquela encontrada na fatoração de **A**. Isto acontece por exemplo em problemas dependentes do tempo com coeficientes constantes [2].

Para estes casos, as técnicas iterativas apresentam a vantagem que a solução no passo anterior, pode ser empregada como solução inicial no próximo passo de tempo. Da mesma maneira, para problemas resolvidos ao longo de uma sequência de malhas, como em análises adaptáveis e métodos multigrid, toma-se a solução anterior como aproximação inicial para uma nova malha.

2.3 Métodos Iterativos Básicos

Dada uma matriz de partição não-singular $\mathbf{Q}$, pode-se reescrever o sistema de equações (1.30) como,

$$(\mathbf{A}+\mathbf{Q}-\mathbf{Q})\mathbf{u}=\mathbf{b} \rightarrow \mathbf{u}=\mathbf{Q}^{-1}\mathbf{b}+(\mathbf{I}-\mathbf{Q}^{-1}\mathbf{A})\mathbf{u} \tag{2.10}$$

sendo $\mathbf{I}$ a matriz identidade de ordem N.

A partir de (2.10), tem-se uma expressão geral para os métodos iterativos básicos [59],

$$\mathbf{u}^{(n+1)}=\mathbf{G}\mathbf{u}^{(n)}+\mathbf{k} \quad n=0,1,2,\ldots \tag{2.11}$$

onde $\mathbf{G}$ e $\mathbf{k}$ são, respectivamente, a matriz de iteração do método e um vetor conhecido dados por,

$$\mathbf{G}=\mathbf{I}-\mathbf{Q}^{-1}\mathbf{A} \qquad \mathbf{k}=\mathbf{Q}^{-1}\mathbf{b} \tag{2.12}$$

Os métodos definidos por (2.11) são também conhecidos como métodos estacionários lineares de primeiro grau. Esta denominação se deve aos seguintes fatos:

estacionário : $\mathbf{G}$ e $\mathbf{k}$ não dependem de n.

linear : $\mathbf{G}$ e $\mathbf{k}$ não dependem de $\mathbf{u}$;

primeiro grau : $\mathbf{u}^{(n+1)}$ depende explicitamente apenas de $\mathbf{u}^{(n)}$ e não das demais aproximações $\mathbf{u}^{(n-1)},\ldots,\mathbf{u}^{(1)}$;

Assumindo que $\mathbf{A}$ é não-singular, tem-se que $\bar{\mathbf{u}}$ é solução do sistema associado,

$$(\mathbf{I}-\mathbf{G})\mathbf{u}=\mathbf{k} \tag{2.13}$$

se e somente se $\bar{\mathbf{u}}$ é a única solução de (1.30), ou seja, $\bar{\mathbf{u}}=\mathbf{A}^{-1}\mathbf{b}$.

Um método iterativo na forma (2.11), cujo sistema relacionado (2.13) tem solução única $\bar{\mathbf{u}}$ coincidente com a solução de (1.30), é denominado *completamente consistente*. Sendo $\{\mathbf{u}^{(n)}\}$ a sequência de iterações determinada por (2.11), esta propriedade implica que se $\mathbf{u}^{(n)}=\bar{\mathbf{u}}$ para algum valor de n, tem-se que $\mathbf{u}^{(n+1)}=\mathbf{u}^{(n+2)}=\ldots=\bar{\mathbf{u}}$. Além disso, se $\{\mathbf{u}^{(n)}\}$ converge para algum vetor $\hat{\mathbf{u}}$ então $\hat{\mathbf{u}}=\bar{\mathbf{u}}$.

Por outro lado, um método iterativo é *convergente* se para qualquer vetor inicial $\mathbf{u}^{(0)}$, a sequência de aproximações $\mathbf{u}^{(1)},\mathbf{u}^{(2)},\ldots$ definida por (2.11) converge para $\bar{\mathbf{u}}$. Uma condição necessária e suficiente para a convergência é dada por [15, 59],

$$S(\mathbf{G})<1 \tag{2.14}$$

Na expressão anterior, $S(\mathbf{G})$ é o raio espectral da matriz de iteração $\mathbf{G}$, definido como o maior valor em módulo dos autovalores $\{\lambda_i\}_{i=1}^{N}$ de $\mathbf{G}$,

$$S(\mathbf{G})=\max_{1\leq i\leq N}|\lambda_i| \tag{2.15}$$

A convergência na iteração n é medida a partir do vetor de erro $\mathbf{e}^{(n)}$,

$$\mathbf{e}^{(n)} = \mathbf{u}^{(n)} - \bar{\mathbf{u}} \tag{2.16}$$

Substituindo o vetor $\mathbf{k}$ dado em (2.13), com $\mathbf{u} = \bar{\mathbf{u}}$, na expressão (2.11), verifica-se que,

$$\mathbf{e}^{(n)} = \mathbf{G}\mathbf{e}^{(n-1)} = \ldots = \mathbf{G}^n\mathbf{e}^{(0)} \tag{2.17}$$

Tomando-se uma norma adequada $||\cdot||$, obtém-se,

$$||\mathbf{e}^{(n)}|| \leq ||\mathbf{G}^n||\,||\mathbf{e}^{(0)}|| \tag{2.18}$$

Assim, $||\mathbf{G}^n||$ fornece uma medida da redução da norma do erro após n iterações. As taxas média e assimptótica de convergência são definidas, respectivamente, como [2, 15],

$$R_n(\mathbf{G}) = -n^{-1}\log||\mathbf{G}^n|| \tag{2.19}$$

$$R_\infty(\mathbf{G}) = \lim_{n\to\infty} R_n(\mathbf{G}) \tag{2.20}$$

Além disso, demonstra-se que se $S(\mathbf{G}) < \mathbf{1}$,

$$R_\infty(\mathbf{G}) = -\log S(\mathbf{G}) \tag{2.21}$$

Observa-se que $R_n(\mathbf{G})$ depende da norma empregada, enquanto $R_\infty(\mathbf{G})$, geralmente denominada taxa de convergência, não utiliza nenhum tipo de norma.

No entanto, não é necessário que os métodos definidos por (2.11) sejam convergentes de acordo com a definição anterior. Exige-se apenas que os mesmos sejam simétricos. Para isso, deve existir uma matriz não-singular $\mathbf{W}$ tal que a matriz $\mathbf{W}(\mathbf{I} - \mathbf{G})\mathbf{W}^{-1}$ seja SPD.

Se o método iterativo (2.11) é simétrico, então as seguintes propriedades são válidas [59]:

- os autovalores de $\mathbf{G}$ são reais;
- o maior autovalor algébrico de $\mathbf{G}$ é menor que 1;
- os autovetores de $\mathbf{G}$ definem uma base para o espaço vetorial associado.

A simetria do método não implica na sua convergência, pois os autovalores não são necessariamente inferiores a 1 em valor absoluto, violando assim a condição (2.14).

Porém, tem-se o método extrapolado baseado em (2.11), o qual é sempre convergente caso (2.11) seja simétrico. Este método é definido como [59],

$$\mathbf{u}^{(n+1)} = \gamma(\mathbf{G}\mathbf{u}^{(n)} + \mathbf{k}) + (1-\gamma)\mathbf{u}^{(n)} = \mathbf{G}_{[\gamma]}\mathbf{u}^{(n)} + \gamma\mathbf{k} \tag{2.22}$$

e

$$\mathbf{G}_{[\gamma]} = \gamma[\mathbf{G} + (1-\gamma)\mathbf{I}] \tag{2.23}$$

O parâmetro γ é conhecido como fator de extrapolação. Se o método é simétrico, então o valor ótimo γ_{ot} é dado por,

$$\gamma_{ot} = \frac{2}{2 - M(\mathbf{G}) - m(\mathbf{G})} \tag{2.24}$$

onde $M(\mathbf{G})$ e $m(\mathbf{G})$ são, respectivamente, o maior e o menor autovalores algébricos de $\mathbf{G}$.

Além disso, esta variante do método (2.11) é convergente visto que,

$$S(\mathbf{G}_{[\gamma]}) = \frac{M(\mathbf{G}) - m(\mathbf{G})}{2 - M(\mathbf{G}) - m(\mathbf{G})} < 1$$

A seguir serão apresentados os métodos iterativos de Jacobi, Gauss-Seidel, SOR e SSOR. Para isso, expande-se a equação (1.30) como,

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1i} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2i} & \dots & a_{2N} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{i1} & a_{i2} & \dots & a_{ii} & \dots & a_{iN} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{N1} & a_{N2} & \dots & a_{Ni} & \dots & a_{NN} \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ \vdots \\ u_i \\ \vdots \\ u_N \end{Bmatrix} = \begin{Bmatrix} b_1 \\ b_2 \\ \vdots \\ b_i \\ \vdots \\ b_N \end{Bmatrix} \tag{2.25}$$

com $a_{ij} = a_{ji}$ $(i, j = 1, \dots, N)$ devido a simetria de $\mathbf{A}$.

Os elementos a_{ij} são números reais definindo uma partição pontual de $\mathbf{A}$. Podem-se considerar ainda partições por linha, coluna ou do tipo *red-black*, onde os elementos a_{ij} constituem-se em submatrizes de ordem $n_i \times n_j$ [59]; analogamente para os elementos u_i e b_i dos vetores $\mathbf{u}$ e $\mathbf{b}$.

A partir de (2.25), a matriz do sistema $\mathbf{A}$ é expressa como,

$$\mathbf{A} = \mathbf{D} - \mathbf{C}_L - \mathbf{C}_U \tag{2.26}$$

sendo,

$$\mathbf{D} = \begin{bmatrix} d_{11} & & & 0 \\ & d_{22} & & \\ & & \ddots & \\ 0 & & & d_{NN} \end{bmatrix} \quad \mathbf{C}_U = -\begin{bmatrix} 0 & a_{12} & a_{13} & \cdots & a_{1N} \\ 0 & 0 & a_{23} & \cdots & a_{2N} \\ 0 & 0 & 0 & \cdots & a_{3N} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}$$

$$\mathbf{C}_L = -\begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ a_{21} & 0 & 0 & \cdots & 0 \\ a_{31} & a_{32} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ a_{N1} & a_{N2} & a_{N3} & \cdots & 0 \end{bmatrix} \tag{2.27}$$

2.3.1 Método de Jacobi

Dada uma solução inicial $\mathbf{u}^{(0)}$, o método de Jacobi (JAC) consiste em obter para cada passo i a respectiva incógnita u_i $(i = 1, \dots, N)$. Assim, para a primeira iteração e tomando-se a i-ésima equação, tem-se a partir de (2.25),

$$a_{ii}u_i^{(1)} = \left(b_1 - a_{i1}u_1^{(0)} - a_{i2}u_2^{(0)} - \dots - a_{i,i-1}u_{i-1}^{(0)} - a_{i,i+1}u_{i+1}^{(0)} - \dots - a_{iN}u_N^{(0)}\right) \tag{2.28}$$

De forma geral, para a iteração $(n+1)$, vem que,

$$a_{ii}u_i^{(n+1)} = b_i - \sum_{j=1\,(j\neq i)}^{N} a_{ij}\,u_j^{(n)} \quad i = 1, \dots, N \tag{2.29}$$

Observa-se que a ordem na qual as incógnitas são atualizadas é irrelevante. Por isso, este algoritmo é denominado como *método de deslocamentos simultâneos*, estando a sua implementação apresentada a seguir [15].

Dada uma aproximação inicial $\mathbf{u}^{(0)}$
for $n = 1, 2, \ldots$
  for $i = 1, N$
    $\sigma = 0$
    for $j = 1, i-1$
      $\sigma = \sigma + a_{ij} u_j^{(n-1)}$
    end
    for $j = i+1, N$
      $\sigma = \sigma + a_{ij} u_j^{(n-1)}$
    end
    $u_i^{(n)} = (b_i - \sigma)/a_{ii}$
  end
  Verificar convergência
end

Em notação matricial, este procedimento é expresso como,

$$\mathbf{u}^{(n+1)} = \mathbf{B}\mathbf{u}^{(n)} + \mathbf{k} \tag{2.30}$$

Na relação anterior, a matriz de iteração $\mathbf{B}$ e o vetor $\mathbf{k}$ são definidos por,

$$\mathbf{B} = \mathbf{D}^{-1}(\mathbf{C}_L + \mathbf{C}_U) = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A} \qquad \mathbf{k} = \mathbf{D}^{-1}\mathbf{b} \tag{2.31}$$

O método de Jacobi é simétrico tomando-se $\mathbf{W} = \mathbf{D}^{\frac{1}{2}}$, $\mathbf{W} = \mathbf{A}^{\frac{1}{2}}$ ou $\mathbf{W} = \mathbf{S}$, sendo $\mathbf{S}$ uma matriz tal que $\mathbf{S}^T\mathbf{S} = \mathbf{D}$. Verifica-se a convergência se e somente se $S(\mathbf{B}) < 1$. O método de extrapolação ótimo (2.22) baseado no método de Jacobi é sempre convergente, possuindo a seguinte forma,

$$\mathbf{u}^{(n+1)} = \gamma_{ot}(\mathbf{B}\mathbf{u}^{(n)} + \mathbf{k} - \mathbf{u}^{(n)}) + \mathbf{u}^{(n)} \tag{2.32}$$

Em geral, tanto para o método de Jacobi como para Gauss-Seidel, o raio espectral é próximo de $1 - \mathcal{O}(h^2)$. Observa-se que $h = N^{-d}$ é o tamanho característico da malha, sendo N o número de variáveis e d a dimensão do problema. Estes métodos são efetivos na redução das amplitudes das componentes de alta frequência da aproximação $\mathbf{u}^{(n)}$, sendo portanto adequados em estratégias multigrid, como será visto mais adiante. De forma geral, a convergência do método de Jacobi só é alcançada para casos onde a matriz $\mathbf{A}$ é fortemente diagonal dominante [15].

Para colunas ascendentes, uma implementação eficiente do método de Jacobi é efetuada considerando o algoritmo ao longo das colunas de $\mathbf{A}$. No entanto, independente do tipo de armazenamento, são necessários, além de $\mathbf{A}$, os vetores $\mathbf{u}^{(n)}$, $\mathbf{u}^{(n+1)}$ e $\mathbf{b}$, resultando em mais $3N$ posições de memória. Por sua vez, o número de operações por iteração é $N(2m-1)$.

2.3.2 Método de Gauss-Seidel

O método de Gauss-Seidel (GS) é análogo ao anterior, diferenciando-se pelo fato que as componentes já atualizadas da aproximação $\mathbf{u}^{(n)}$ são empregadas a medida que são calculadas. Por exemplo,

tomando-se a iteração $(n+1)$, no cálculo de $u_i^{(n+1)}$ utilizam-se as componentes $u_1^{(n+1)}, \ldots, u_{i-1}^{(n+1)}$ determinadas nesta mesma iteração, ao invés dos valores $u_1^{(n)}, \ldots, u_{i-1}^{(n)}$ da iteração n como no método de Jacobi. Portanto,

$$a_{ii}u_i^{(n+1)} = \left(b_i - a_{i1}u_1^{(n+1)} - a_{i2}u_2^{(n+1)} - \ldots - a_{i,i-1}u_{i-1}^{(n+1)} - a_{i,i+1}u_{i+1}^{(n)} - \ldots - a_{iN}u_N^{(n)}\right)$$

De forma geral,

$$a_{ii}u_i^{(n+1)} = b_i - \sum_{j=1}^{i-1} a_{ij}u_j^{(n+1)} - \sum_{j=i+1}^{N} a_{ij}u_j^{(n)} \tag{2.33}$$

Neste caso, cada componente $u_i^{(n+1)}$ depende daquelas calculadas anteriormente. Além disso, a aproximação $\mathbf{u}^{(n+1)}$ é função da ordem na qual as equações são examinadas. Desta maneira, o algoritmo GS é conhecido como *método de deslocamentos sucessivos*, enfatizando a dependência no ordenamento das equações.

Reescrevendo (2.33) matricialmente, tem-se,

$$(\mathbf{D} - \mathbf{C}_L)\mathbf{u}^{(n+1)} = \mathbf{C}_U\mathbf{u}^{(n)} + \mathbf{b} \rightarrow \mathbf{u}^{(n+1)} = \mathcal{L}\mathbf{u}^{(n)} + \mathbf{k} \tag{2.34}$$

onde,

$$\mathcal{L} = (\mathbf{I} - \mathbf{C}_{DL})^{-1}\mathbf{C}_{DU} \qquad \mathbf{k} = (\mathbf{I} - \mathbf{L})^{-1}\mathbf{D}^{-1}\mathbf{b} \qquad \mathbf{C}_{DL} = \mathbf{D}^{-1}\mathbf{C}_L \qquad \mathbf{C}_{DU} = \mathbf{D}^{-1}\mathbf{C}_U \tag{2.35}$$

Observa-se que $\mathcal{L}$ é a matriz de iteração de Gauss-Seidel. Por sua vez, a matriz de partição $(\mathbf{D} - \mathbf{C}_L)$ não é SPD e em geral o método não é simétrico. Como neste caso, $\mathbf{A}$ e $\mathbf{D}$ são SPD, demonstra-se que o método de Gauss-Seidel sempre converge. Em geral, os autovalores de $\mathcal{L}$, embora menores que a unidade em módulo, podem ser complexos, enquanto os autovetores não constituem necessariamente uma base para o espaço vetorial associado. Neste casos, o método de extrapolação não pode ser empregado.

No que se refere à implementação, utiliza-se apenas um vetor solução $\mathbf{u}^{(n)}$ como apresentado no algoritmo abaixo [15].

Dada uma aproximação inicial $\mathbf{u}^{(0)}$
for $n = 1, 2, \ldots$
  for $i = 1, N$
    $\sigma = 0$
    for $j = 1, i-1$
      $\sigma = \sigma + a_{ij}u_j^{(n)}$
    end
    for $j = i+1, N$
      $\sigma = \sigma + a_{ij}u_j^{(n-1)}$
    end
    $u_i^{(n)} = (b_i - \sigma)/a_{ii}$
  end
  Verificar convergência
end

Geralmente, para o critério de convergência deve-se calcular um resíduo, sendo necessário mais um vetor. Desta forma, o espaço de memória e o custo por iteração são os mesmos do método de Jacobi. Verifica-se ainda que o número de iterações necessárias para a convergência é inferior quando comparado ao método de Jacobi.

2.3.3 Método de sobre-relaxação

No método de sobre-relaxação (SOR), emprega-se um parâmetro de relaxação ω, visando acelerar a taxa de convergência das aproximações. De maneira geral tem-se,

$$a_{ii}u_i^{(n+1)} = \omega\left(b_i - \sum_{j=1}^{i-1} a_{ij}u_j^{(n+1)} - \sum_{j=i+1}^{N} a_{ij}u_j^{(n)}\right) + (1-\omega)a_{ii}u_i^{(n)} \quad i = 1, \ldots, N \tag{2.36}$$

Para $\omega = 1$, obtém-se o método de Gauss-Seidel; para $\omega < 1$ e $\omega > 1$ tem-se, respectivamente, sub e sobre-relaxação.

Em forma matricial, (2.36) reduz-se a,

$$\mathbf{D}\mathbf{u}^{(n+1)} = \omega(\mathbf{b} + \mathbf{C}_L\mathbf{u}^{(n+1)} + \mathbf{C}_U\mathbf{u}^{(n)}) + (1-\omega)\mathbf{D}\mathbf{u}^{(n)} \rightarrow \mathbf{u}^{(n+1)} = \mathcal{L}_\omega\mathbf{u}^{(n)} + \mathbf{k}_\omega^{(F)} \tag{2.37}$$

com,

$$\mathcal{L}_\omega = (\mathbf{I} - \omega\mathbf{C}_{DL})^{-1}(\omega\mathbf{C}_{DU} + (1-\omega)\mathbf{I}) \qquad \mathbf{k}_\omega^{(F)} = (\mathbf{I} - \omega\mathbf{C}_{DL})^{-1}\omega\mathbf{D}^{-1}\mathbf{b} \tag{2.38}$$

Por sua vez, $\mathcal{L}_\omega$ é a matriz de iteração do método; $\mathbf{C}_{DL}$ e $\mathbf{C}_{DU}$ são dadas em (2.35). A matriz de partição dada por $(\omega^{-1}\mathbf{D} - \mathbf{C}_L)$ não é SPD. Para $\omega > 1$, este método não é simétrico e a matriz de iteração $\mathcal{L}_\omega$ possui autovalores complexos, impossibilitando a aplicação do método com extrapolação.

Como $\mathbf{A}$ e $\mathbf{D}$ são SPD, o método SOR converge para $0 < \omega < 2$. É possível ainda selecionar ω de tal forma que a convergência seja sensivelmente superior àquela obtida nos métodos de Jacobi e GS. Para problemas elípticos, pode-se tomar $\omega \approx 1,8$ como valor ótimo. Além disso, tem-se algoritmos adaptáveis para determinar ω, utilizando estimativas para a taxa na qual a iteração corrente está convergindo [15, 59].

A implementação deste método é análoga ao caso anterior, como pode ser observado abaixo, devendo-se considerar apenas as multiplicações por ω a cada passo [15]. O custo de cada iteração é $2Nm$ e a memória necessária é a mesma do método GS.

Dada uma aproximação inicial $\mathbf{u}^{(0)}$
for $n = 1, 2, \ldots$
  for $i = 1, N$
    $\sigma = 0$
    for $j = 1, i-1$
      $\sigma = \sigma + a_{ij}u_j^{(n)}$
    end
    for $j = i+1, N$
      $\sigma = \sigma + a_{ij}u_j^{(n-1)}$
    end
    $\sigma = (b_i - \sigma)/a_{ii}$
    $u_i^{(n)} = u_i^{(n-1)} + \omega(\sigma - u_i^{(n-1)})$
  end
  Verificar convergência
end

2.3.4 Método de sobre-relaxação simétrico

O método de sobre-relaxação simétrico (SSOR) é definido pelas seguintes equações:

$$a_{ii}u_i^{(n+1/2)} = \omega\left(b_i - \sum_{j=1}^{i-1} a_{ij}u_j^{(n+1/2)} - \sum_{j=i+1}^{N} a_{ij}u_j^{(n)}\right) + (1-\omega)a_{ii}u_i^{(n)} \quad i = 1,\ldots,N \tag{2.39}$$

$$a_{ii}u_i^{(n+1)} = \omega\left(b_i - \sum_{j=1}^{i-1} a_{ij}u_j^{(n+1/2)} - \sum_{j=i+1}^{N} a_{ij}u_j^{(n+1)}\right) + (1-\omega)a_{ii}u_i^{(n+1/2)} \quad i = N,\ldots,1 \tag{2.40}$$

Inicialmente, calculam-se $u_1^{(n+1/2)},\ldots,u_N^{(n+1/2)}$ aplicando-se o método SOR a (2.39). Posteriormente, determinam-se $u_N^{(n+1)},\ldots,u_1^{(n+1)}$ usando SOR de modo reverso.

Em forma matricial, (2.39) e (2.40) reduzem-se a,

$$\mathbf{D}\mathbf{u}^{(n+1/2)} = \omega(\mathbf{b} + \mathbf{C}_L\mathbf{u}^{(n+1/2)} + \mathbf{C}_U\mathbf{u}^{(n)}) + (1-\omega)\mathbf{D}\mathbf{u}^{(n)}$$

$$\mathbf{D}\mathbf{u}^{(n+1)} = \omega(\mathbf{b} + \mathbf{C}_L\mathbf{u}^{(n+1/2)} + \mathbf{C}_U\mathbf{u}^{(n+1)}) + (1-\omega)\mathbf{D}\mathbf{u}^{(n+1/2)}$$

ou ainda,

$$\mathbf{u}^{(n+1/2)} = \mathcal{L}_\omega\mathbf{u}^{(n)} + \mathbf{k}_\omega^{(F)} \tag{2.41}$$

$$\mathbf{u}^{(n+1)} = \mathcal{U}_\omega\mathbf{u}^{(n+1/2)} + \mathbf{k}_\omega^{(B)} \tag{2.42}$$

Em (2.41), $\mathcal{L}_\omega$ e $\mathbf{k}_\omega^{(F)}$ são dados por (2.38). Por sua vez, tem-se,

$$\mathcal{U}_\omega = (\mathbf{I} - \omega\mathbf{C}_{DU})^{-1}(\omega\mathbf{C}_{DL} + (1-\omega)\mathbf{I}) \qquad \mathbf{k}_\omega^{(B)} = (\mathbf{I} - \omega\mathbf{C}_{DU})^{-1}\omega\mathbf{D}^{-1}\mathbf{b} \tag{2.43}$$

Substituindo (2.41) em (2.42) obtém-se,

$$\mathbf{u}^{(n+1)} = \mathcal{S}_\omega\mathbf{u}^{(n)} + \mathbf{k}_\omega \tag{2.44}$$

A matriz de iteração $\mathcal{S}_\omega$ e o vetor $\mathbf{k}_\omega$ são expressos como,

$$\mathcal{S}_\omega = \mathcal{U}_\omega\mathcal{L}_\omega \qquad \mathbf{k}_\omega = \omega(2-\omega)(\mathbf{I} - \omega\mathbf{C}_{DU})^{-1}(\mathbf{I} - \omega\mathbf{C}_{DL})^{-1}\mathbf{D}^{-1}\mathbf{b} \tag{2.45}$$

A matriz de partição é dada por,

$$\mathbf{Q} = \frac{\omega}{2-\omega}\left(\frac{1}{\omega}\mathbf{D} - \mathbf{C}_L\right)\mathbf{D}^{-1}\left(\frac{1}{\omega}\mathbf{D} - \mathbf{C}_U\right) \tag{2.46}$$

Como $(\omega^{-1}\mathbf{D} - \mathbf{C}_L)^T = \omega^{-1}\mathbf{D} - \mathbf{C}_U$, a matriz $\mathbf{Q}$ é SPD e, portanto, o procedimento SSOR é simétrico, tornando viável a aplicação do método de extrapolação (2.22).

Para $\mathbf{A}$ SPD, demonstra-se que o método SSOR converge para todo $0 < \omega < 2$. Além disso, os autovalores de $\mathcal{L}_\omega$ são reais e não-negativos, implicando numa convergência duas vezes mais rápida do respectivo método de extrapolação ótimo em relação ao SSOR original. Observa-se ainda que a convergência é relativamente insensível ao valor de ω.

Demonstra-se que a matriz de iteração $\mathcal{S}_\omega$ definida em (2.45) é similar a uma matriz simétrica [59]. Desta maneira, a principal aplicação do algoritmo SSOR é como pré-condicionador para métodos iterativos com matrizes simétricas. Além disso, tomando-se um mesmo valor ótimo para ω, a taxa de convergência do SSOR é inferior ao SOR [15].

Finalmente, a implementação é análoga ao método SOR, como apresentado abaixo [15]. No entanto, o custo de cada iteração é duas vezes superior, ou seja, $4Nm$.

Dada uma aproximação inicial $\mathbf{u}^{(0)}$
for $n = 1, 2, \ldots$
  for $i = 1, N$
    $\sigma = 0$
    for $j = 1, i-1$
      $\sigma = \sigma + a_{ij} u_j^{(n+1/2)}$
    end
    for $j = i+1, N$
      $\sigma = \sigma + a_{ij} u_j^{(n-1)}$
    end
    $\sigma = (b_i - \sigma)/a_{ii}$
    $u_i^{(n+1/2)} = u_i^{(n-1)} + \omega(\sigma - u_i^{(n-1)})$
  end
  for $i = N, 1$
    $\sigma = 0$
    for $j = 1, i-1$
      $\sigma = \sigma + a_{ij} u_j^{(n+1/2)}$
    end
    for $j = i+1, N$
      $\sigma = \sigma + a_{ij} u_j^{(n)}$
    end
    $\sigma = (b_i - \sigma)/a_{ii}$
    $u_i^{(n)} = u_i^{(n+1/2)} + \omega(\sigma - u_i^{(n+1/2)})$
  end
  Verificar convergência
end

2.4 Aceleração Polinomial

Os procedimentos de aceleração polinomial visam aumentar as taxas de convergência dos algoritmos iterativos básicos discutidos na seção anterior. Para isso, determina-se uma nova sequência de aproximações, tomando-se combinações lineares dos vetores obtidos nas iterações do método considerado.

Suponha que o método básico seja completamente consistente, simétrico e descrito por,

$$\mathbf{w}^{(n)} = \mathbf{G}\mathbf{w}^{(n-1)} + \mathbf{k} \quad n = 1, 2, \ldots \tag{2.47}$$

Neste caso, o vetor de erro $\tilde{\mathbf{e}}^{(n)} = \mathbf{w}^{(n)} - \bar{\mathbf{u}}$ da n-ésima iteração satisfaz a relação,

$$\tilde{\mathbf{e}}^{(n)} = \mathbf{G}^n \tilde{\mathbf{e}}^{(0)} \tag{2.48}$$

Para acelerar a convergência das aproximações $\mathbf{w}^{(n)}$, toma-se uma nova sequência $\mathbf{u}^{(n)}$ dada pela seguinte combinação linear,

$$\mathbf{u}^{(n)} = \sum_{i=0}^{n} \alpha_{n,i}\, \mathbf{w}^{(i)} \quad n = 0, 1, \ldots \tag{2.49}$$

tal que,

$$\sum_{i=0}^{n} \alpha_{n,i} = 1 \quad n = 0, 1, \ldots \tag{2.50}$$

Subtraindo a solução exata $\bar{\mathbf{u}}$ de ambos os lados de (2.49) e utilizando (2.48), obtém-se,

$$\mathbf{e}^{(n)} = \sum_{i=0}^{n} \alpha_{n,i}\tilde{\mathbf{e}}^{(i)} = (\sum_{i=0}^{n} \alpha_{n,i}\mathbf{G}^{(i)})\tilde{\mathbf{e}}^{(0)} = Q_n(\mathbf{G})\tilde{\mathbf{e}}^{(0)} \tag{2.51}$$

onde $Q_n(\mathbf{G}) = \alpha_{n,0}\mathbf{I} + \alpha_{n,1}\mathbf{G} + \ldots + \alpha_{n,n}\mathbf{G}^n$ é um polinômio matricial. Tomando-se $Q_n(x) = \alpha_{n,0} + \alpha_{n,1}x + \ldots + \alpha_{n,n}x^n$ como o polinômio algébrico associado a $Q_n(\mathbf{G})$ observa-se que $Q_n(1) = 1$.

A partir de (2.49) e (2.50) verifica-se que $\mathbf{e}^0 = \tilde{\mathbf{e}}^0$. Desta forma, a expressão para o erro (2.51) pode ser reescrita como,

$$\mathbf{e}^{(n)} = Q_n(\mathbf{G})\mathbf{e}^{(0)} \tag{2.52}$$

Devido a forma da equação (2.52), o procedimento definido por (2.47) e (2.49) é denominado *método de aceleração polinomial*. No entanto, através de (2.49), verifica-se um alto custo computacional em termos de processamento e armazenamento.

Para contornar este problema, basta tomar os polinômios $Q_n(x)$ dados pela seguinte relação de recorrência [59],

$$\begin{aligned} Q_0(x) &= 1 \\ Q_1(x) &= \gamma_1 x - \gamma_1 + 1 \\ Q_{n+1}(x) &= \rho_{n+1}(\gamma_{n+1}x + 1 - \gamma_{n+1})Q_n(x) + (1 - \rho_{n+1})Q_{n-1}(x) \quad n \geq 1 \end{aligned} \tag{2.53}$$

onde γ_i e ρ_i são números reais. Além disso, observa-se que $Q_n(1) = 1 \ \forall n \geq 0$.

Como (2.47) é completamente consistente e sendo $Q_n(x)$ dado por (2.53), demonstra-se que as aproximações $\mathbf{u}^{(n)}$ podem ser determinadas através da relação com 3 termos [59],

$$\begin{aligned} \mathbf{u}^{(1)} &= \gamma_1(\mathbf{G}\mathbf{u}^{(0)} + \mathbf{k}) + (1 - \gamma_1)\mathbf{u}^{(0)} \\ \mathbf{u}^{(n+1)} &= \rho_{n+1}\{\gamma_{n+1}(\mathbf{G}\mathbf{u}^{(n)} + \mathbf{k}) + (1 - \gamma_{n+1})\mathbf{u}^{(n)}\} + (1 - \rho_{n-1})\mathbf{u}^{(n-1)} \quad n \geq 1 \end{aligned} \tag{2.54}$$

Existem várias sequências de polinômios $\{Q_n(x)\}$ satisfazendo (2.53), como por exemplo aquelas associadas aos métodos de Chebyshev e Gradiente Conjugado.

Sendo (2.47) simétrico com matriz $\mathbf{W}$, tem-se a partir de (2.52),

$$||\mathbf{e}^{(n)}||_{\mathbf{W}} = ||Q_n(\mathbf{G})\mathbf{e}^{(0)}||_{\mathbf{W}} \tag{2.55}$$

onde $||\mathbf{a}||_{\mathbf{W}} = (\mathbf{a}^T, \mathbf{W}\mathbf{a})^{\frac{1}{2}}$.

No método de Gradiente Conjugado, procura-se determinar a sequência $\{Q_n(x)\}$ tal que a norma do erro $||\mathbf{e}^{(n)}||_{\mathbf{W}}$ seja minimizada.

Devido a simetria de (2.47), verifica-se que $\mathbf{W}\mathbf{G}\mathbf{W}^{-1}$ e por conseguinte $\mathbf{W}Q_n(\mathbf{G})\mathbf{W}^{-1}$ são simétricas, podendo-se escrever [59],

$$||\mathbf{e}^{(n)}||_{\mathbf{W}} \leq ||Q_n(\mathbf{G})||_{\mathbf{W}}\,||\mathbf{e}^{(0)}||_{\mathbf{W}} = S(Q_n(\mathbf{G}))||\mathbf{e}^{(0)}||_{\mathbf{W}} \tag{2.56}$$

No método de Chebyshev, toma-se $\{Q_n(x)\}$ tal que o raio espectral,

$$S(Q_n(\mathbf{G})) = \max_{1\leq i\leq N} |Q_n(\mu_i)| \tag{2.57}$$

seja pequeno, onde μ_i $(i = 1, \ldots, N)$ são os autovalores de $\mathbf{G}$.

Como todo espectro de $\mathbf{G}$ é raramente conhecido, trabalha-se com o raio espectral virtual dado por,

$$\bar{S}(Q_n(\mathbf{G})) = \max_{m(\mathbf{G})\leq x\leq M(\mathbf{G})} |Q_n(x)| \tag{2.58}$$

Como $\mu_i \in [m(\mathbf{G}), M(\mathbf{G})]$ tem-se que,

$$S(Q_n(\mathbf{G})) \leq \bar{S}(Q_n(\mathbf{G})) \tag{2.59}$$

Logo, no método de Chebyshev procura-se minimizar $\bar{S}(Q_n(\mathbf{G}))$ através da seleção adequada da sequência $\{Q_n(\mathbf{G})\}$.

A partir de (2.58), definem-se as taxas de convergência virtuais média e assimptótica, respectivamente, por,

$$\bar{R}_n(Q_n(\mathbf{G})) = -n^{-1} \log \bar{S}(Q_n(\mathbf{G})) \tag{2.60}$$

$$\bar{R}_\infty(Q_n(\mathbf{G})) = \lim_{n\to\infty} \bar{R}_n(Q_n(\mathbf{G})) \tag{2.61}$$

Observa-se que outras técnicas de aceleração não-polinomial dos métodos básicos podem ser empregadas, como por exemplo os métodos multigrid. Nas seções seguintes, apresentam-se os métodos de Chebyshev e Gradiente Conjugado.

2.5 Aceleração de Chebyshev

Como mencionado anteriormente, supondo que o método iterativo básico seja simétrico e completamente consistente, a aceleração de Chebyshev procura minimizar o raio espectral virtual $\bar{S}(Q_n(\mathbf{G}))$ dado em (2.58). Para isso, definem-se parâmetros de iteração depedentes dos autovalores extremos $m(\mathbf{G})$ e $M(\mathbf{G})$ da matriz de iteração $\mathbf{G}$.

O polinômio de Chebyshev de ordem n é dado pela expressão recursiva [55, 59],

$$\begin{aligned} T_0(w) &= 1 \\ T_1(w) &= w \\ T_{n+1}(w) &= 2wT_n(w) - T_{n-1}(w) \end{aligned} \tag{2.62}$$

Visando minimizar (2.58), deve-se determinar um polinômio $P_n(x)$ tal que $P_n(1) = 1$ e além disso,

$$\max_{m(\mathbf{G})\leq x\leq M(\mathbf{G})} |P_n(x)| \leq \max_{m(\mathbf{G})\leq x\leq M(\mathbf{G})} |Q_n(x)| \tag{2.63}$$

onde $Q_n(x)$ é qualquer polinômio de grau menor ou igual a n satisfazendo $Q_n(1) = 1$.

Toma-se,

$$w(x) = \frac{2x - m(\mathbf{G}) - M(\mathbf{G})}{m(\mathbf{G}) - M(\mathbf{G})} \tag{2.64}$$

como a transformação linear mapeando o intervalo $m(\mathbf{G}) \leq x \leq M(\mathbf{G})$ em $-1 \leq w \leq 1$.

A partir daí, demonstra-se que o único polinômio $P_n(x)$ satisfazendo (2.63) é dado por [59],

$$P_n(x) = \frac{T_n\left(\frac{2x-m(\mathbf{G})-M(\mathbf{G})}{M(\mathbf{G})-m(\mathbf{G})}\right)}{T_n\left(\frac{2-m(\mathbf{G})-M(\mathbf{G})}{M(\mathbf{G})-m(\mathbf{G})}\right)} = \frac{T_n(w(x))}{T_n(w(1))} \tag{2.65}$$

Observa-se a partir de (2.62) e (2.65) que $P_n(x)$ satisfaz uma relação de recorrência análoga a (2.53), ou seja,

$$\begin{aligned} P_0(x) &= 1 \\ P_1(x) &= \bar{\gamma}x - \bar{\gamma} + 1 \\ P_{n+1}(x) &= P_n(x) + (1 - \bar{\rho}_{n+1})P_{n-1}(x) \end{aligned} \tag{2.66}$$

com,

$$\bar{\gamma} = \frac{2}{2 - m(\mathbf{G}) - M(\mathbf{G})} \qquad \bar{\rho}_{n+1} = 2w(1)\frac{T_n(w(1))}{T_{n+1}(w(1))} \tag{2.67}$$

Da definição (2.62) dos polinômios de Chebyshev, verifica-se que o parâmetro $\bar{\rho}_{n+1}$ satisfaz as relações,

$$\begin{aligned} \bar{\rho}_1 &= 1 \\ \bar{\rho}_2 &= \left(1 - \frac{1}{2}\bar{\sigma}^2\right)^{-1} \\ \bar{\rho}_{n+1} &= \left(1 - \frac{1}{4}\bar{\sigma}^2\bar{\rho}_n\right)^{-1} \qquad n \geq 2 \end{aligned} \tag{2.68}$$

onde,

$$\bar{\sigma} = \frac{1}{w(1)} = \frac{M(\mathbf{G}) - m(\mathbf{G})}{2 - m(\mathbf{G}) - M(\mathbf{G})} \tag{2.69}$$

No procedimento ótimo de aceleração de Chebyshev, as aproximações são determinadas pela seguinte relação de recorrência,

$$\mathbf{u}^{(n+1)} = \bar{\rho}_{n+1}\{\bar{\gamma}(\mathbf{G}\mathbf{u}^{(n)} + \mathbf{k}) + (1 - \bar{\gamma})\mathbf{u}^{(n)}\} + (1 - \bar{\rho}_{n+1})\mathbf{u}^{(n-1)} \tag{2.70}$$

A expressão anterior pode ser reescrita como,

$$\mathbf{u}^{(n+1)} = \bar{\rho}_{n+1}\{\bar{\gamma}\delta^{(n)} + \mathbf{u}^{(n)} + \mathbf{u}^{(n-1)}\} + \mathbf{u}^{(n-1)} \tag{2.71}$$

onde $\delta^{(n)} = \mathbf{G}\mathbf{u}^{(n)} + \mathbf{k} - \mathbf{u}^{(n)} = \mathbf{u}^{(n+1)} - \mathbf{u}^{(n)}$ é denominado vetor de pseudo-resíduo.

A partir de (2.11), obtém-se $\delta^{(n)}$ resolvendo-se o sistema,

$$\mathbf{Q}\delta^{(n)} = \mathbf{b} - \mathbf{A}\mathbf{u}^{(n)} \tag{2.72}$$

No que se refere à convergência, o raio espectral virtual de $P_n(\mathbf{G})$ é dado por [59],

$$\bar{S}(P_n(\mathbf{G})) = \frac{1}{T_n[w(1)]} = \frac{1}{T_n(1/\bar{\sigma})} \tag{2.73}$$

O termo $T_n(1/\bar{\sigma})$ pode ainda ser expresso como,

$$T_n\left(\frac{1}{\bar{\sigma}}\right) = \frac{1+\bar{r}^n}{2\bar{r}^{n/2}} \tag{2.74}$$

sendo,

$$\bar{r} = \frac{1-\sqrt{1-\bar{\sigma}^2}}{1+\sqrt{1-\bar{\sigma}^2}} \tag{2.75}$$

Portanto, a partir de (2.73) e (2.74) tem-se,

$$\bar{S}(P_n(\mathbf{G})) = \frac{2\bar{r}^{n/2}}{1+\bar{r}^n} \tag{2.76}$$

Utilizando as definições (2.60) e (2.61) juntamente com (2.76), obtém-se as taxas virtuais de convergência média e assimptótica do método ótimo (2.70). Logo,

$$\bar{R}_n(P_n(\mathbf{G})) = -\frac{1}{2}\log\bar{r} - \frac{1}{n}\log\left(\frac{2}{1+\bar{r}^n}\right) \tag{2.77}$$

$$\bar{R}_\infty(P_n(\mathbf{G})) = -\frac{1}{2}\log\bar{r} \tag{2.78}$$

Verifica-se então que $\bar{R}_n(P_n(\mathbf{G})) < \bar{R}_\infty(P_n(\mathbf{G}))$ para todo n finito. Além disso, $\bar{R}_n(P_n(\mathbf{G}))$ é uma função crescente de n. Em geral, várias iterações são necessárias até que se atinja a taxa assimptótica de convergência [59].

Os seguintes passos podem ser empregados na implementação do procedimento ótimo de Chebyshev.

Dados $\mathbf{u}^{(n-1)} = \mathbf{0}$, $\mathbf{u}^{(n)} = \mathbf{0}$ e $\mathbf{u}^{(n+1)} = \mathbf{0}$
$\gamma = 2/(2 - m(\mathbf{G}) - M(\mathbf{G}))$
$\sigma = (M(\mathbf{G}) - m(\mathbf{G}))/(2 - m(\mathbf{G}) - M(\mathbf{G}))$
for $n = 1, 2, \ldots$
  if$(n == 1)$
    $\rho = 1$
  elseif $(n == 2)$
    $\rho = 1/(1 - \sigma^2/2)$
  else
    $\rho = 1/(1 - \rho\sigma^2/4)$
  end
  Resolver $\mathbf{Q}\delta^{(n)} = \mathbf{b} - \mathbf{A}\mathbf{u}^{(n)}$
  $\mathbf{u}^{(n+1)} = \rho(\gamma\delta^{(n)} + \mathbf{u}^{(n)} - \mathbf{u}^{(n-1)}) + \mathbf{u}^{(n-1)}$
  Verificar convergência
  $\mathbf{u}^{(n-1)} = \mathbf{u}^{(n)}$
  $\mathbf{u}^{(n)} = \mathbf{u}^{(n+1)}$
end

O termo ótimo é empregado devido a utilização dos autovalores $m(\mathbf{G})$ e $M(\mathbf{G})$ da matriz de iteração $\mathbf{G}$. Como em geral, estas constantes não estão disponíveis, utilizam-se as respectivas estimativas $m_E(\mathbf{G})$ e $M_E(\mathbf{G})$. Para isso, pode-se aplicar um algoritmo adaptável visando melhorar estimativas

iniciais para os autovalores extremos de $\mathbf{G}$ [59]. No entanto, este procedimento dependente sensivelmente dos valores iniciais de $m_E(\mathbf{G})$ e $M_E(\mathbf{G})$. Neste trabalho, calculam-se estas estimativas através do algoritmo de Lanczos [55], discutido na próxima seção.

Observa-se que nenhum procedimento de aceleração polinomial é aplicado aos métodos GS e SOR. Por exemplo, ao se tomar a matriz de iteração SOR com valor ótimo para ω como pré-condicionador, os autovalores de $\mathbf{A}$ são mapeados para um círculo no plano complexo. Assim, qualquer procedimento de aceleração recupera o algoritmo SOR original [15].

Considerou-se a aceleração de Chebyshev aplicada ao método SSOR, denotado por CHSS. A implementação está baseada em [95], necessitando $5N$ posições de memória para os vetores auxiliares e $N(4m+1)$ operações por iteração.

2.5.1 Método de Lanczos

Emprega-se o método de Lanczos na solução de problemas de autovalor do tipo $\mathbf{Au} = \lambda\mathbf{u}$, para os casos onde $\mathbf{A}$ é uma matriz simétrica, de alta ordem e esparsa. O algoritmo gera uma sequência de matrizes tridiagonais $\mathbf{T}_j$, cujos autovalores extremos são progressivamente melhores estimativas para os respectivos autovalores de $\mathbf{A}$. Este procedimento é particularmente útil nos casos onde se deseja apenas alguns dos menores e maiores autovalores extremos de $\mathbf{A}$.

Verifica-se uma correspondência entre os métodos de Lanczos e Gradiente Conjugado, no sentido que se pode derivar um a partir do outro [15, 55]. No entanto, a dedução aqui adotada está baseada na otimização do quociente de Rayleigh dado por [16, 55],

$$R(\mathbf{u}) = \frac{\mathbf{u}^T\mathbf{Au}}{\mathbf{u}^T\mathbf{u}} \qquad \mathbf{u} \neq \mathbf{0} \tag{2.79}$$

É conhecido que os valores mínimo e máximo de $R(\mathbf{u})$ são iguais ao menor e maior autovalores de $\mathbf{A}$, indicados respectivamente por λ_1 e λ_N. Suponha que $\{\mathbf{q}_i\} \subset \Re^n$ é uma sequência de vetores ortonormais e sejam os escalares m_j e M_j definidos por,

$$m_j = \lambda_1(\mathbf{Q}_j^T\mathbf{AQ}_j) = \min_{\mathbf{v}\neq\mathbf{0}} \frac{\mathbf{v}^T(\mathbf{Q}_j^T\mathbf{AQ}_j)\mathbf{v}}{\mathbf{v}^T\mathbf{v}} = \min_{\|\mathbf{v}\|_2=1} R(\mathbf{Q}_j\mathbf{v}) \geq \lambda_1(\mathbf{A}) \tag{2.80}$$

$$M_j = \lambda_N(\mathbf{Q}_j^T\mathbf{AQ}_j) = \max_{\mathbf{v}\neq\mathbf{0}} \frac{\mathbf{v}^T(\mathbf{Q}_j^T\mathbf{AQ}_j)\mathbf{v}}{\mathbf{v}^T\mathbf{v}} = \max_{\|\mathbf{v}\|_2=1} R(\mathbf{Q}_j\mathbf{v}) \leq \lambda_N(\mathbf{A}) \tag{2.81}$$

onde $\mathbf{Q}_j = [\mathbf{q}_1 \ldots \mathbf{q}_j]$. Deriva-se o algoritmo de Lanczos, gerando-se os vetores $\mathbf{q}_j$ tal que m_j e M_j sejam progressivamente melhores estimativas para os autovalores extremos λ_1 e λ_N.

Para isso, seja o vetor $\mathbf{u}_j \in \text{span}\{\mathbf{q}_1, \ldots, \mathbf{q}_j\}$ com $M_j = R(\mathbf{u}_j)$. Como $R(\mathbf{u}_j)$ cresce mais rapidamente na direção do gradiente,

$$\nabla R(\mathbf{u}) = \frac{2}{\mathbf{u}^T\mathbf{u}}(\mathbf{Au} - R(\mathbf{u})\mathbf{u}) \tag{2.82}$$

pode-se afirmar que $M_{j+1} > M_j$ se $\mathbf{q}_{j+1}$ é determinado de forma que $\nabla R(\mathbf{u}_j) \in \text{span}\{\mathbf{q}_1, \ldots, \mathbf{q}_{j+1}\}$. Analogamente, se $\mathbf{v}_j \in \{\mathbf{q}_1, \ldots, \mathbf{q}_j\}$ satisfaz $m_j = R(\mathbf{u}_j)$, então deseja-se que $\nabla R(\mathbf{v}_j) \in \{\mathbf{q}_1, \ldots, \mathbf{q}_{j+1}\}$, pois $R(\mathbf{u})$ decresce mais rapidamente na direção de $-\nabla R(\mathbf{u})$.

Para satisfazer as duas condições anteriores, observa-se que $\nabla R(\mathbf{u}) \in \text{span}\{\mathbf{u}, \mathbf{Au}\}$. Logo, basta impor a condição,

$$\text{span}\{\mathbf{q}_1, \ldots, \mathbf{q}_j\} = \text{span}\{\mathbf{q}_1, \mathbf{Aq}_1, \ldots, \mathbf{A}^{j-1}\mathbf{q}_1\} = \mathcal{K}(\mathbf{A}, \mathbf{q}_1, j)$$

e selecionar $\mathbf{q}_{j+1} \in \mathcal{K}(\mathbf{A}, \mathbf{q}_1, j+1) = \text{span}\{\mathbf{q}_1, \ldots, \mathbf{q}_{j+1}\}$. Assim, o problema se reduz a determinação de bases ortonormais para os subespaços de Krylov $\mathcal{K}(\mathbf{A}, \mathbf{q}_1, j)$.

Obtém-se esta base de forma eficiente, considerando a correspondência entre a tridiagonalização de $\mathbf{A}$ e a fatoração QR da matriz de Krylov,

$$\mathcal{K}(\mathbf{A}, \mathbf{q}_1, n) = [\mathbf{q}_1, \mathbf{A}\mathbf{q}_1, \ldots, \mathbf{A}^{n-1}\mathbf{q}_1] \tag{2.83}$$

Logo, se $\mathbf{Q}^T\mathbf{A}\mathbf{Q} = \mathbf{T}$ é tridiagonal com $\mathbf{Q}\mathbf{e}_1 = \mathbf{q}_1$, então a fatoração QR da matriz $\mathcal{K}(\mathbf{A}, \mathbf{q}_1, n)$ é dada por,

$$\mathcal{K}(\mathbf{A}, \mathbf{q}_1, n) = \mathbf{Q}[\mathbf{e}_1, \mathbf{T}\mathbf{e}_1, \mathbf{T}^2\mathbf{e}_1, \ldots, \mathbf{T}^{n-1}\mathbf{e}_1] \tag{2.84}$$

Observa-se que o vetor $\mathbf{q}_j$ pode ser efetivamente obtido pela tridiagonalização de $\mathbf{A}$ com uma matriz cuja primeira coluna é $\mathbf{q}_1$. Segundo [55], os procedimentos de Householder e Givens devem ser evitados para preservar a esparsidade de $\mathbf{A}$.

Calcula-se, então, os elementos de $\mathbf{T} = \mathbf{Q}^T\mathbf{A}\mathbf{Q}$ diretamente. Considere $\mathbf{Q} = [\mathbf{q}_1 \ldots \mathbf{q}_n]$ e

$$\mathbf{T} = \begin{bmatrix} \alpha_1 & \beta_1 & \ldots & \ldots & 0 \\ \beta_1 & \alpha_2 & \ldots & \ldots & \vdots \\ & \ddots & & & \vdots \\ & & \ddots & \beta_{n-1} & \\ & & \beta_{n-1} & \alpha_n & \end{bmatrix} \tag{2.85}$$

Equacionando as colunas em $\mathbf{A}\mathbf{Q} = \mathbf{Q}\mathbf{T}$ vem que,

$$\mathbf{A}\mathbf{q}_j = \beta_{j-1}\mathbf{q}_{j-1} + \alpha_j\mathbf{q}_j + \beta_j\mathbf{q}_{j+1} \qquad \beta_0\mathbf{q}_0 = \mathbf{0} \qquad j = 1, \ldots, n-1 \tag{2.86}$$

A ortonormalidade de $\mathbf{q}_i$ implica que $\alpha_j = \mathbf{q}_j^T\mathbf{A}\mathbf{q}_j$. Além disso, se $\mathbf{r}_j = (\mathbf{A} - \alpha_j\mathbf{I})\mathbf{q}_j - \beta_{j-1}\mathbf{q}_j$ é não-nulo, então $\mathbf{q}_{j+1} = \mathbf{r}_j/\beta_j$ para $\beta_j = \pm\|\mathbf{r}_j\|_2$. Se $\beta_j = 0$ então deve-se interromper o processo. Na prática, compara-se β_j com uma precisão ξ. A implementação do algoritmo de Lanczos é feita pelos passos a seguir [55].

$\mathbf{r}_0 = \mathbf{q}_1$, $\beta_0 = 1$, $\mathbf{q}_0 = \mathbf{0}$, $j = 0$
while $(\beta_j > \xi)$
    $\mathbf{q}_{j+1} = \mathbf{r}_j/\beta_j$
    $j = j + 1$
    $\alpha_j = \mathbf{q}_j^T\mathbf{A}\mathbf{q}_j$
    $\mathbf{r}_j = (\mathbf{A} - \alpha_j\mathbf{I})\mathbf{q}_j - \beta_{j-1}\mathbf{q}_j$
    $\beta_j = \|\mathbf{r}_j\|_2$
end

A partir daí, os autovalores extremos de $\mathbf{T}$ são determinados pelo método da bisseção. Para isso, seja $\mathbf{T}_r$ a submatriz dos termos principais de ordem r, obtida eliminando-se as r últimas linhas e colunas de $\mathbf{T}$. Definindo os polinômios $p_n(x) = \det(\mathbf{T}_r - x\mathbf{I})$ $(r = 1, \ldots, N)$, a seguinte relação é válida,

$$p_r(x) = (\alpha_r - x)p_{r-1}(x) - \beta_{r-1}^2 p_{r-2}(x) \qquad r = 2, \ldots, N \qquad p_0(x) = 1 \tag{2.87}$$

Para $p_n(y)p_n(z) < 0$, o método da bisseção é implementado como mostrado a seguir [55].

while $(|y - z| > \xi(|y| + |z|))$
  $x = (y + z)/2$
  if $p_n(x)p_n(y) < 0$
    $z = x$
  else
    $y = x$
  end
end

Logo, o algoritmo termina com uma aproximação igual a $(y + z)/2$ para uma raiz de $p_n(x)$, ou seja, para um autovalor de $\mathbf{T}$. Observa-se que a iteração converge linearmente e o erro é reduzido pela metade a cada iteração.

Em alguns casos, é necessário calcular o k-ésimo maior autovalor de $\mathbf{T}$. O teorema de Gershgorin [55] estabelece que $\lambda_k(\mathbf{T}) \in [y, z]$ com,

$$y = \min_{1 \le i \le N} \alpha_i - |\beta_i| - |\beta_{i-1}| \qquad z = \min_{1 \le i \le N} \alpha_i + |\beta_i| + |\beta_{i-1}| \tag{2.88}$$

A propriedade da sequência de Sturm indica que os autovalores de $\mathbf{T}_{r-1}$ separam os autovalores de $\mathbf{T}_r$ da seguinte maneira [16, 55],

$$\lambda_r(\mathbf{T}_r) < \lambda_{r-1}(\mathbf{T}_{r-1}) < \lambda_{r-1}(\mathbf{T}_r) < \ldots < \lambda_2(\mathbf{T}_r) < \lambda_1(\mathbf{T}_{r-1}) < \lambda_1(\mathbf{T}_r) \tag{2.89}$$

Além disso, se $a(\lambda)$ denota o número de trocas de sinal na sequência $\{p_0(\lambda), p_1(\lambda), \ldots, p_N(\lambda)\}$, então $a(\lambda)$ contém o número de autovalores de $\mathbf{T}$ menores que λ.

Empregando esta propriedade, as estimativas (2.88) e o método da bisseção, o seguinte procedimento determina uma sequência de subintervalos contendo $\lambda_k(\mathbf{T})$ [55].

while $(|y - z| > \xi(|y| + |z|))$
  $x = (y + z)/2$
  if $a(x) \ge k$
    $z = x$
  else
    $y = x$
  end
end

Os algoritmos apresentados nesta seção foram empregados em conjunto para o cálculo das estimativas $m_E(\mathbf{G})$ e $M_E(\mathbf{G})$ dos autovalores extremos da matriz de iteração $\mathbf{G}$ no método CHSS.

2.6 Método de Gradiente Conjugado

Considere o funcional quadrático F dado em (1.28). Como a matriz $\mathbf{A}$ é SPD, observa-se que o problema de minimizar F é análogo à solução do sistema de equações (1.30). Assim, tem-se uma classe de métodos iterativos cujo objetivo é determinar um mínimo local de um funcional quadrático. A solução iterativa deste problema de mínimo possui a seguinte forma geral [2],

$$\mathbf{u}^{(n+1)} = \mathbf{u}^{(n)} + \tau_n \mathbf{d}^{(n)} \tag{2.90}$$

sendo $\mathbf{d}^{(n)}$ uma direção de busca e τ_n um escalar selecionado de tal maneira a minimizar $F(\mathbf{u})$ ao longo da linha passando por $\mathbf{u}^{(n)}$ na direção $\mathbf{d}^{(n)}$.

Supondo que $F \in C^1(\Re^n)$, demonstra-se que dentre todas as direções $\mathbf{d}$ num ponto $\mathbf{u}$, aquela onde F decresce mais rapidamente na vizinhaça de $\mathbf{u}$ é dada pelo gradiente de F, ou seja [2],

$$\mathbf{d} = \mathbf{g}(\mathbf{u}) = -\nabla F(\mathbf{u}) = \mathbf{b} - \mathbf{A}\mathbf{u} \tag{2.91}$$

Os procedimentos para a determinação de τ_n são conhecidos como algoritmos de busca unidimensional. No caso de um funcional quadrático, τ_n pode ser obtido de forma simples. Para isso, considera-se,

$$F(\mathbf{u} + \tau\mathbf{d}) = \frac{1}{2}\tau^2(\mathbf{d}, \mathbf{A}\mathbf{d}) + \tau(\mathbf{d}, \mathbf{g}) \tag{2.92}$$

Tomando-se o mínimo de (2.92), obtém-se,

$$\tau = -\frac{(\mathbf{d}, \mathbf{g})}{(\mathbf{d}, \mathbf{A}\mathbf{d})} \tag{2.93}$$

A partir das expressões (2.90), (2.91) e (2.93), define-se o método de máximo decréscimo para a minimização do funcional (1.28) como,

$$\begin{aligned} \mathbf{u}^{(0)} &\quad \text{arbitrário} \\ \mathbf{g}^{(n)} &= \mathbf{A}\mathbf{u}^{(n)} - \mathbf{b} \\ \tau_n &= \frac{(\mathbf{g}^{(n)}, \mathbf{g}^{(n)})}{(\mathbf{g}^{(n)}, \mathbf{A}\mathbf{g}^{(n)})} \\ \mathbf{u}^{(n+1)} &= \mathbf{u}^{(n)} - \tau_n \mathbf{g}^{(n)} \end{aligned} \tag{2.94}$$

Em geral, este método possui uma taxa de convergência baixa. Para contornar esta limitação, o método de Gradiente Conjugado (GC) define a direção de busca como,

$$\mathbf{d}^{(n+1)} = -\mathbf{g}^{(n+1)} + \beta_n \mathbf{d}^{(n)} \quad n = 0, 1, 2 \ldots \tag{2.95}$$

sendo $\mathbf{d}^{(0)} = -\mathbf{g}^{(0)}$ e $\beta_0, \beta_1, \ldots$ constantes determinadas de tal forma a minimizar o erro $||\mathbf{e}^{(n)}||_A$ a cada iteração. Demonstra-se que os parâmetros β_n são dados por [2],

$$\beta_n = \frac{(\mathbf{g}^{(n+1)}, \mathbf{A}\mathbf{d}^{(n)})}{(\mathbf{d}^{(n)}, \mathbf{A}\mathbf{d}^{(n)})} \tag{2.96}$$

Além disso, os gradientes $\mathbf{g}^{(n)}$ e as direções $\mathbf{d}^{(n)}$ satisfazem as seguintes relações [2, 59],

$$\begin{aligned} (\mathbf{g}^{(i)}, \mathbf{g}^{(j)}) &= 0 \quad i \neq j \\ (\mathbf{d}^{(i)}, \mathbf{A}\mathbf{d}^{(j)}) &= 0 \quad i \neq j \\ (\mathbf{g}^{(i)}, \mathbf{A}\mathbf{d}^{(j)}) &= 0 \quad i \neq j \quad i \neq j+1 \end{aligned} \tag{2.97}$$

Multiplicando (2.90) por $\mathbf{A}$, subtraindo $\mathbf{b}$ em ambos os lados da equação e substituindo (2.91) no resultado, obtém-se a relação,

$$\mathbf{g}^{(n+1)} = \mathbf{g}^{(n)} + \tau_n \mathbf{A}\mathbf{d}^{(n)} \tag{2.98}$$

Os parâmetros β_n e τ_n podem ainda ser reescritos como [2, 59],

$$\beta_n = \frac{(\mathbf{g}^{(n+1)}, \mathbf{g}^{(n+1)})}{(\mathbf{g}^{(n)}, \mathbf{g}^{(n)})} \tag{2.99}$$

$$\tau_n = \frac{(\mathbf{g}^{(n)}, \mathbf{g}^{(n)})}{(\mathbf{d}^{(n)}, \mathbf{A}\mathbf{d}^{(n)})} \tag{2.100}$$

A partir destas equações, a forma computacionalmente mais conveniente do método de gradiente conjugado pode ser resumida em [2]:

$$\begin{aligned} \tau_n &= \frac{(\mathbf{g}^{(n)}, \mathbf{g}^{(n)})}{(\mathbf{d}^{(n)}, \mathbf{A}\mathbf{d}^{(n)})} \\ \mathbf{u}^{(n+1)} &= \mathbf{u}^{(n)} + \tau_n \mathbf{d}^{(n)} \\ \mathbf{g}^{(n+1)} &= \mathbf{g}^{(n)} + \tau_n \mathbf{A}\mathbf{d}^{(n)} \\ \beta_n &= \frac{(\mathbf{g}^{(n+1)}, \mathbf{g}^{(n+1)})}{(\mathbf{g}^{(n)}, \mathbf{g}^{(n)})} \\ \mathbf{d}^{(n+1)} &= -\mathbf{g}^{(n+1)} + \beta_n \mathbf{d}^{(n)} \end{aligned} \tag{2.101}$$

com $\mathbf{u}^{(0)}$ dado e $\mathbf{d}^{(0)} = -\mathbf{g}^{(0)}$.

A implementação computacional deste algoritmo é executada pelos seguintes passos [2].

Dada uma aproximação inicial $\mathbf{u}^{(0)}$
$\mathbf{u} = \mathbf{u}^{(0)}$
$\mathbf{g} = \mathbf{A}\mathbf{u} - \mathbf{b}$
$\delta_0 = \mathbf{g}^T\mathbf{g}$
if $\delta_0 \leq \xi$ **then stop**
$\mathbf{d} = -\mathbf{g}$
R: $\mathbf{h} = \mathbf{A}\mathbf{d}$
$\tau = \delta_0/\mathbf{d}^T\mathbf{h}$
$\mathbf{u} = \mathbf{u} + \tau\mathbf{d}$
$\mathbf{g} = \mathbf{g} + \tau\mathbf{d}$
$\delta_1 = \mathbf{g}^T\mathbf{g}$
if $\delta_1 \leq \xi$ **then stop**
$\beta = \delta_1/\delta_0$
$\delta_0 = \delta_1$
$\mathbf{d} = -\mathbf{g} + \beta\mathbf{d}$
goto R

Em relação ao espaço de memória, deve-se armazenar um total de 4 vetores, equivalentes a 4N posições de memória. No que se refere ao número de operações, tem-se o produto $\mathbf{Ad}$, 2 produtos internos e 3 fórmulas de recorrência, respectivamente, com $(2m-1)N$, $2N$ e $3N$ operações, totalizando $N(2m+4)$ por iteração.

A partir da expressão (2.93), a constante τ_n pode ser reescrita como,

$$\tau_n = -\frac{(\mathbf{d}^{(n)}, \mathbf{g}^{(n)})}{(\mathbf{d}^{(n)}, \mathbf{A}\mathbf{d}^{(n)})} \tag{2.102}$$

Esta expressão para τ_n faz com que o gradiente $\mathbf{g}^{(n+1)}$ da iteração $n+1$ seja ortogonal a $\mathbf{d}^{(n)}$. Isto pode ser visto pré-multiplicando (2.98) por $\mathbf{d}^{(n)^T}$ e substituindo (2.102). Multiplicando agora (2.95) por $\mathbf{g}^{(n+1)^T}$ tem-se que,

$$\mathbf{g}^{(n+1)^T}\mathbf{d}^{(n+1)} = -||\mathbf{g}^{(n+1)}||_2^2 + \beta_n \mathbf{g}^{(n+1)^T}\mathbf{d}^{(n)} \tag{2.103}$$

Como $\mathbf{g}^{(n+1)^T}\mathbf{d}^{(n)} = 0$, se $\mathbf{d}^{(n+1)} = \mathbf{0}$ então $||\mathbf{g}^{(n+1)}||_2 = 0$ implicando que $\mathbf{u} = \bar{\mathbf{u}}$. Portanto, uma direção nula só é determinada se o mínimo do funcional for encontrado. Assim, a convergência do algoritmo anterior é verificada comparando-se $\mathbf{g}^T\mathbf{g}$ com a precisão ξ. Isto é equivalente a se comparar a norma do resíduo $||\mathbf{Au} - \mathbf{b}||_2$ com ξ.

Uma das vantagens deste método é o fato que na ausência de erros de arrendondamento, verifica-se a convergência em no máximo N iterações.

Dada uma precisão $\xi > 0$ e definindo $p(\xi)$ como o menor número inteiro n tal que,

$$||\mathbf{u}^{(n)} - \bar{\mathbf{u}}||_A \leq \xi ||\mathbf{u}^{(0)} - \bar{\mathbf{u}}||_A \tag{2.104}$$

então verifica-se que [2],

$$p(\xi) \leq \frac{1}{2}\sqrt{\kappa(\mathbf{A})}\,\ln(\frac{2}{\xi}) + 1 \tag{2.105}$$

sendo $\kappa(\mathbf{A})$ o número de condição espectral de $\mathbf{A}$, dado pela razão entre o maior e o menor autovalores de $\mathbf{A}$, ou seja,

$$\kappa(\mathbf{A}) = \frac{\lambda_N}{\lambda_1} \tag{2.106}$$

2.6.1 Método de gradiente com pré-condicionamento

Como se pode observar a partir de (2.105), a taxa de convergência do método de Gradiente Conjugado depende do número de condição $\kappa(\mathbf{A})$. O objetivo da técnica de pré-condicionamento é minimizar, ao invés de (1.28), o funcional quadrático $\tilde{F}$ dado por,

$$\tilde{F}(\mathbf{v}) = \frac{1}{2}(\mathbf{v}, \tilde{\mathbf{A}}\mathbf{v}) - (\tilde{\mathbf{b}}, \mathbf{v}) \quad \mathbf{v} \in \Re^n \tag{2.107}$$

de tal forma que $\kappa(\tilde{\mathbf{A}}) < \kappa(\mathbf{A})$, aumentando assim a taxa de convergência do método.

O funcional $\tilde{F}$ é obtido tomando-se uma matriz não-singular $\mathbf{E}$ e a transformação $\mathbf{v} = \mathbf{E}^T\mathbf{u}$, a qual substituída em (1.28) resulta em (2.107) com,

$$\tilde{\mathbf{A}} = \mathbf{E}^{-1}\mathbf{A}\mathbf{E}^{-T} \quad \tilde{\mathbf{b}} = \mathbf{E}^{-1}\mathbf{b} \tag{2.108}$$

Como $\mathbf{v}^T\tilde{\mathbf{A}}\mathbf{v} = \mathbf{u}^T\mathbf{A}\mathbf{u}$ e $\mathbf{A}$ é SPD, logo $\tilde{\mathbf{A}}$ também é positiva-definida. Por sua vez, a transformação de similaridade

$$\mathbf{E}^{-T}\tilde{\mathbf{A}}\mathbf{E}^T = \mathbf{E}^{-T}\mathbf{E}^{-1}\mathbf{A} = \mathbf{C}^{-1}\mathbf{A} \tag{2.109}$$

revela que $\tilde{\mathbf{A}}$ e $\mathbf{C}^{-1}\mathbf{A}$ possuem os mesmos autovalores.

Observa-se que $\mathbf{C} = \mathbf{E}\mathbf{E}^T$ é positiva-definida, sendo denominada *matriz de pré-condicionamento*. Da mesma forma, $\tilde{\mathbf{A}}$ é a *matriz pré-condicionada*.

Pode-se aplicar o algoritmo de GC ao funcional (2.107), definindo o método de GC com pré-condicionamento transformado, pois a sequência de aproximações $\mathbf{v}^{(n)}$ converge para $\bar{\mathbf{v}} = \tilde{\mathbf{A}}^{-1}\tilde{\mathbf{b}}$. Para

se obter a solução do problema original, efetua-se a transformação $\mathbf{u} = \mathbf{E}^{-T}\mathbf{v}$. Esta variante será empregada na seção 2.6.3 com o intuito de reduzir o número de operações.

Logo, a partir de (2.101), a versão pré-condicionada do método de gradiente não-transformado é dada por [2],

$$\begin{aligned}
\tau_n &= \frac{(\mathbf{g}^{(n)}, \mathbf{g}^{(n)})}{(\mathbf{d}^{(n)}, \mathbf{A}\mathbf{d}^{(n)})} \\
\mathbf{u}^{(n+1)} &= \mathbf{u}^{(n)} + \tau_n \mathbf{d}^{(n)} \\
\mathbf{g}^{(n+1)} &= \mathbf{g}^{(n)} + \tau_n \mathbf{A}\mathbf{d}^{(n)} \\
\mathbf{h}^{(n+1)} &= \mathbf{C}^{-1}\mathbf{g}^{(n+1)} \\
\beta_n &= \frac{(\mathbf{g}^{(n+1)}, \mathbf{h}^{(n+1)})}{(\mathbf{g}^{(n)}, \mathbf{h}^{(n)})} \\
\mathbf{d}^{(n+1)} &= -\mathbf{h}^{(n+1)} + \beta_n \mathbf{d}^{(n)}
\end{aligned} \tag{2.110}$$

Os seguintes passos são adotados na implementação da versão pré-condicionada (2.110) [2].

Dada uma aproximação inicial $\mathbf{u}^{(0)}$
$\mathbf{u} = \mathbf{u}^{(0)}$
$\mathbf{g} = \mathbf{A}\mathbf{u} - \mathbf{b}$
$\mathbf{h} = \mathbf{C}^{-1}\mathbf{g}$
$\mathbf{d} = -\mathbf{h}$
$\delta_0 = \mathbf{g}^T\mathbf{h}$
if $\delta_0 \leq \xi$ **then stop**
R: $\mathbf{h} = \mathbf{A}\mathbf{d}$
$\tau = \delta_0/\mathbf{d}^T\mathbf{h}$
$\mathbf{u} = \mathbf{u} + \tau\mathbf{d}$
$\mathbf{g} = \mathbf{g} + \tau\mathbf{h}$
$\mathbf{h} = \mathbf{C}^{-1}\mathbf{g}$
$\delta_1 = \mathbf{g}^T\mathbf{g}$
if $\delta_1 \leq \xi$ **then stop**
$\beta = \delta_1/\delta_0$
$\delta_0 = \delta_1$
$\mathbf{d} = -\mathbf{h} + \beta\mathbf{d}$
goto R

O número de iterações $p(\xi)$ do algoritmo anterior passa a depender de $\kappa(\tilde{\mathbf{A}})$, ou seja,

$$p(\xi) \leq \frac{1}{2}\sqrt{\kappa(\tilde{\mathbf{A}})}\,\ln(\frac{2}{\xi}) + 1 \tag{2.111}$$

Portanto, se $\mathbf{C}$ possui a propriedade tal que $\kappa(\tilde{\mathbf{A}}) < \kappa(\mathbf{A})$, logo a versão pré-condicionada apresenta uma taxa de convergência maior que a versão original (2.101). Observa-se ainda em (2.110) a presença da matriz $\mathbf{C}$ e não de $\mathbf{E}$. Assim, como qualquer matriz positiva-definida $\mathbf{C}$ possui várias fatorações do tipo $\mathbf{C} = \mathbf{E}\mathbf{E}^T$, pode-se tomar a matriz de pré-condicionamento na classe de matrizes positivas-definidas.

Em (2.111), o número de condição $\kappa(\tilde{\mathbf{A}})$ é dado por,

$$\kappa(\tilde{\mathbf{A}}) = \frac{\tilde{\lambda}_N}{\tilde{\lambda}_1} \tag{2.112}$$

onde $\tilde{\lambda}_1$ e $\tilde{\lambda}_N$ indicam o menor e o maior autovalores de $\mathbf{C}^{-1}\mathbf{A}$. Assim, $\kappa(\tilde{\mathbf{A}})$ é independente da forma empregada para fatorar $\mathbf{C}$.

Uma matriz de pré-condicionamento adequada deve possuir as seguintes propriedades [2]:

- $\kappa(\tilde{\mathbf{A}})$ é significativamente menor que $\kappa(\mathbf{A})$;
- os termos de $\mathbf{C}$ devem ser determinados de forma rápida;
- a resolução do sistema $\mathbf{Ch}^{(n)} = \mathbf{g}^{(n)}$ deve ser mais eficiente que a solução de (1.30).

2.6.2 Aceleração de métodos iterativos

Os métodos iterativos estacionários constituem-se numa fonte para a obtenção de matrizes de pré-condicionamento para o método de gradiente conjugado. Para ilustrar este fato, considere o sistema relacionado (2.13) e multiplique o mesmo por $\mathbf{W}$, ou seja,

$$\mathbf{W}(\mathbf{I} - \mathbf{G})\mathbf{u} = \mathbf{Wk}$$

Como em geral, a matriz $\mathbf{W}(\mathbf{I} - \mathbf{G})$ não é simétrica, introduz-se o vetor $\hat{\mathbf{u}} = \mathbf{Wu}$. Logo, chega-se ao sistema de equações,

$$\hat{\mathbf{A}}\hat{\mathbf{u}} = \hat{\mathbf{b}} \tag{2.113}$$

onde,

$$\hat{\mathbf{A}} = \mathbf{W}(\mathbf{I} - \mathbf{G})\mathbf{W}^{-1} \qquad \hat{\mathbf{b}} = \mathbf{Wk} \tag{2.114}$$

Substituindo $\mathbf{G}$ e $\mathbf{k}$ dados em (2.12) obtém-se,

$$\hat{\mathbf{A}} = \mathbf{WQ}^{-1}\mathbf{AW}^{-1} \quad \hat{\mathbf{b}} = \mathbf{WQ}^{-1}\mathbf{b} \tag{2.115}$$

O sistema (2.113) é também denominado pré-condicionado. Esta definição é mais geral que aquela apresentada na seção anterior. No entanto, assumindo que a matriz de partição do método iterativo pode ser decomposta em $\mathbf{Q} = \mathbf{W}^T\mathbf{W}$ e substituindo em (2.115) vem que,

$$\hat{\mathbf{A}} = \mathbf{W}^{-T}\mathbf{AW}^{-1} \quad \hat{\mathbf{b}} = \mathbf{W}^{-T}\mathbf{k} \tag{2.116}$$

Tomando-se $\mathbf{W}^T = \mathbf{E}$ tem-se a partir de (2.108) e (2.116) que $\tilde{\mathbf{A}} = \hat{\mathbf{A}}$ e $\tilde{\mathbf{b}} = \hat{\mathbf{b}}$, recuperando a definição inicial de sistema pré-condicionado. Assim, por exemplo, a matriz de partição $\mathbf{Q}$ do método SSOR dada em (2.46) pode ser considerada como matriz de pré-condicionamento para o método de gradiente em (2.110).

O método de gradiente com pré-condicionamento pode ser visto como um procedimento de aceleração, de forma análoga ao algoritmo de Chebyshev. Para isso, tem-se a seguinte forma com 3 termos para o método de GC [59],

$$\mathbf{u}^{(n+1)} = \rho_{n+1}\{\gamma_{n+1}\mathbf{g}^{(n)} + \mathbf{u}^{(n)}\} + (1 - \rho_{n+1})\mathbf{u}^{(n-1)} \tag{2.117}$$

sendo,

$$\begin{aligned}\gamma_{n+1} &= \left[1 - \frac{(\mathbf{W}\delta^{(n)}, \mathbf{W}\mathbf{G}\delta^{(n)})}{(\mathbf{W}\delta^{(n)}, \mathbf{W}\delta^{(n)})}\right]^{-1} \\ \rho_1 &= 1 \\ \rho_{n+1} &= \left[1 - \frac{\gamma_{n+1}}{\gamma_n}\frac{(\mathbf{W}\delta^{(n)}, \mathbf{W}\delta^{(n)})}{(\mathbf{W}\delta^{(n-1)}, \mathbf{W}\delta^{(n-1)})}\frac{1}{\rho_n}\right]^{-1}\end{aligned}$$

Demonstra-se que o método de gradiente minimiza a norma $[\mathbf{W}^T\mathbf{W}(\mathbf{I}-\mathbf{G})]^{1/2}$ do erro comparando-se com qualquer outro procedimento de aceleração aplicado a (2.11). Além disso, denotando por $\mathbf{e}^{(n)}$ e $\tilde{\mathbf{e}}^{(n)}$, respectivamente, os vetores de erro associados aos métodos de aceleração de gradiente e Chebyshev, e assumindo que $\mathbf{e}^{(0)} = \tilde{\mathbf{e}}^{(0)}$, então [59],

$$||\mathbf{e}^{(n)}||_{[\mathbf{W}^T\mathbf{W}(\mathbf{I}-\mathbf{G})]^{1/2}} \leq ||\tilde{\mathbf{e}}^{(n)}||_{[\mathbf{W}^T\mathbf{W}(\mathbf{I}-\mathbf{G})]^{1/2}} \leq \frac{2\bar{r}^{n/2}}{1+\bar{r}^n}||\mathbf{e}^{(0)}||_{[\mathbf{W}^T\mathbf{W}(\mathbf{I}-\mathbf{G})]^{1/2}} \tag{2.118}$$

sendo $\bar{r}$ dado por (2.75). Assim, a norma $[\mathbf{W}^T\mathbf{W}(\mathbf{I}-\mathbf{G})]^{1/2}$ do erro no método de gradiente possui a mesma ordem de grandeza, comparando-se com aquela obtida através da aceleração de Chebyshev.

2.6.3 Matrizes de pré-condicionamento

Como mencionado anteriormente, as matrizes de pré-condicionamento do método de gradiente conjugado podem ser obtidas a partir dos algoritmos iterativos estacionários.

Denotam-se as matrizes de pré-condicionamento como,

$$\mathbf{C} = \mathbf{F}\mathbf{G}_c^{-1}\mathbf{F}^T \tag{2.119}$$

onde $\mathbf{F}$ e $\mathbf{G}_c$ são matrizes triangular inferior e diagonal, respectivamente.

Na iteração n, o sistema $\mathbf{C}\mathbf{h}^{(n)} = \mathbf{g}^{(n)}$ do algoritmo (2.110) é resolvido através dos seguintes processos de substituição,

$$\mathbf{F}\mathbf{h}^{(n)} = \mathbf{g}^{(n)} \tag{2.120}$$

$$\mathbf{F}^T\mathbf{h}^{(n)} = \mathbf{G}_c\mathbf{h}^{(n)} \tag{2.121}$$

Neste trabalho, consideram-se as matrizes de partição dos métodos SSOR e Gauss-Seidel simétrico [74] como pré-condicionadoras para o método GC, estando as mesmas dadas respectivamente por,

$$\mathbf{C} = \left(\frac{1}{\omega}\mathbf{D} - \mathbf{C_L}\right)\left(\frac{1}{\omega}\mathbf{D}\right)^{-1}\left(\frac{1}{\omega}\mathbf{D} - \mathbf{C_L}\right)^T \tag{2.122}$$

$$\mathbf{C} = (2\mathbf{D} - \mathbf{C_L})(2\mathbf{D})^{-1}(2\mathbf{D} - \mathbf{C_L})^T \tag{2.123}$$

Seguindo a notação (2.119) para as matrizes anteriores tem-se,

$$\text{SSOR} \rightarrow \mathbf{F} = \left(\frac{1}{\omega}\mathbf{D} - \mathbf{C_L}\right) \quad \mathbf{G}_c^{-1} = \left(\frac{1}{\omega}\mathbf{D}\right)^{-1}$$

$$\text{GS} \rightarrow \mathbf{F} = (2\mathbf{D} - \mathbf{C_L}) \quad \mathbf{G}_c^{-1} = (2\mathbf{D})^{-1}$$

Independente da escolha da matriz $\mathbf{C}$ entre (2.122) e (2.123), a solução do sistema $\mathbf{Ch}^{(n)} = \mathbf{g}^{(n)}$ requer cerca de $(2m+1)N$ operações. Além disso, a versão pré-condicionada (2.110) necessita 5N operações para os produtos internos e fórmulas de recorrência, mais $(2m-1)N$ operações para o produto $\mathbf{h} = \mathbf{Ad}$. Ao final, tem-se um total de $(4m+5)N$ operações para cada iteração do método de gradiente com pré-condicionamento (2.110).

No entanto, este número de operações pode ser reduzido, considerando a versão transformada do método e aplicando (2.101) ao sistema $\tilde{\mathbf{A}}\mathbf{v} = \tilde{\mathbf{b}}$, onde $\tilde{\mathbf{A}}$ e $\tilde{\mathbf{b}}$ estão dados em (2.108). Se $\mathbf{C}$ é decomposta na forma $\mathbf{C} = \mathbf{EE^T}$, então a partir de (2.119) tem-se $\mathbf{E} = \mathbf{FG_c^{-1/2}}$.

Como $\mathbf{G}_c$ é uma matriz diagonal, pode-se definir uma terceira variante do método, tomando-se $\mathbf{C} = \mathbf{G}_c^{-1}$ como matriz de pré-condicionamento do seguinte sistema,

$$\bar{\mathbf{A}}\mathbf{w} = \bar{\mathbf{b}} \tag{2.124}$$

onde,

$$\bar{\mathbf{A}} = \mathbf{F}^{-1}\mathbf{A}\mathbf{F}^{-T} \quad \bar{\mathbf{b}} = \mathbf{F}^{-1}\mathbf{b} \quad \mathbf{w} = \mathbf{F}^T\mathbf{u}$$

Todas as variantes do método são equivalentes visto que,

$$\mathbf{v}^{(n)} = \mathbf{G}_c^{-1/2}\mathbf{w}^{(n)} = (\mathbf{FG_c^{-1/2}})^T\mathbf{u}^{(n)} \quad n = 0, 1, 2, \ldots \tag{2.125}$$

A aplicação de (2.110) ao sistema (2.124) resulta em,

$$\begin{aligned}
\bar{\tau}_n &= \frac{(\bar{\mathbf{g}}^{(n)}, \bar{\mathbf{g}}^{(n)})}{(\bar{\mathbf{d}}^{(n)}, \bar{\mathbf{A}}\bar{\mathbf{d}}^{(n)})} \\
\bar{\mathbf{w}}^{(n+1)} &= \bar{\mathbf{w}}^{(n)} + \bar{\tau}_n\bar{\mathbf{d}}^{(n)} \\
\bar{\mathbf{g}}^{(n+1)} &= \bar{\mathbf{g}}^{(n)} + \bar{\tau}_n\bar{\mathbf{A}}\bar{\mathbf{d}}^{(n)} \\
\bar{\mathbf{h}}^{(n+1)} &= \mathbf{G}_c\bar{\mathbf{g}}^{(n+1)} \\
\bar{\beta}_n &= \frac{(\bar{\mathbf{g}}^{(n+1)}, \bar{\mathbf{h}}^{(n+1)})}{(\bar{\mathbf{g}}^{(n)}, \bar{\mathbf{h}}^{(n)})} \\
\bar{\mathbf{d}}^{(n+1)} &= -\bar{\mathbf{h}}^{(n+1)} + \bar{\beta}_n\bar{\mathbf{d}}^{(n)}
\end{aligned} \tag{2.126}$$

Inicialmente, toma-se $\mathbf{u}^{(0)}$ e calcula-se $\mathbf{w}^{(0)} = \mathbf{F}^T\mathbf{u}^{(0)}$, $\mathbf{g}^{(0)} = \mathbf{F}^{-1}(\mathbf{Au}^{(0)} - \mathbf{b})$, $\mathbf{d}^{(0)} = -\mathbf{h}$. No entanto, este procedimento converge para $\bar{\mathbf{w}}$ e não para $\bar{\mathbf{u}}$, determinado por $\mathbf{u}^{(n)} = \mathbf{F}^{-T}\mathbf{w}^{(n)}$.

Em alguns casos, a variante (2.126) possui vantagens em relação a (2.110) no cálculo de $\bar{\mathbf{A}}\bar{\mathbf{d}}^{(n)}$. Tomando como pré-condicionador o método SSOR, verifica-se que,

$$\mathbf{A} = \mathbf{D} - \mathbf{C}_L - \mathbf{C}_U = \mathbf{H} + \mathbf{F} + \mathbf{F}^T$$

sendo,

$$\mathbf{F} = (1/\omega)\mathbf{D} + \mathbf{L} \quad \mathbf{H} = (1 - 2/\omega)\mathbf{D}$$

Desta forma, observa-se que [2],

$$\bar{\mathbf{A}}\bar{\mathbf{d}}^{(n)} = \mathbf{F}^{-1}\mathbf{A}\mathbf{F}^{-T}\bar{\mathbf{d}}^{(n)} = \mathbf{F}^{-1}(\mathbf{H} + \mathbf{F} + \mathbf{F}^T)\mathbf{F}^{-T}\bar{\mathbf{d}}^{(n)} = \mathbf{F}^{-1}(\mathbf{H}\bar{\mathbf{f}}^{(n)} + \bar{\mathbf{d}}^{(n)}) + \bar{\mathbf{f}}^{(n)}$$

com $\bar{\mathbf{f}}^{(n)} = \mathbf{F}^{-T}\bar{\mathbf{d}}^{(n)}$.

A implementação desta variante pode ser feita através dos seguintes passos.

Dada uma aproximação inicial $\mathbf{u}^{(0)}$
$\mathbf{u} = \mathbf{u}^{(0)}$
$\mathbf{g} = \mathbf{F}^{-1}(\mathbf{A}\mathbf{u} - \mathbf{b})$
$\mathbf{u} = \mathbf{F}^{T}\mathbf{u}$
$\mathbf{h} = \mathbf{G}_c\mathbf{g}$
$\mathbf{d} = -\mathbf{h}$
$\delta_0 = \mathbf{g}^{T}\mathbf{h}$
if $\delta_0 \leq \xi$ **goto** T
S: $\mathbf{f} = \mathbf{F}^{-T}\mathbf{d}$
$\mathbf{h} = \mathbf{H}\mathbf{f} + \mathbf{d}$
$\mathbf{h} = \mathbf{F}^{-1}\mathbf{h}$
$\mathbf{h} = \mathbf{h} + \mathbf{f}$
$\tau = \delta_0/\mathbf{d}^{T}\mathbf{h}$
$\mathbf{u} = \mathbf{u} + \tau\mathbf{d}$
$\mathbf{g} = \mathbf{g} + \tau\mathbf{h}$
$\mathbf{h} = \mathbf{G}_c\mathbf{g}$
$\delta_1 = \mathbf{g}^{T}\mathbf{h}$
if $\delta_1 \leq \xi$ **return**
$\beta = \delta_1/\delta_0$
$\delta_0 = \delta_1$
$\mathbf{d} = -\mathbf{h} + \beta\mathbf{d}$
goto S
T: $\mathbf{u} = \mathbf{F}^{-T}\mathbf{u}$

Em (2.110), tem-se a solução de 2 sistemas triangulares e a multiplicação por $\mathbf{A}$ enquanto em (2.126) tem-se apenas um sistema resultando em $(2m+7)N$ operações por iteração. Porém, deve-se armazenar o vetor $\mathbf{f}$, aumentando o espaço de memória dos vetores auxiliares para $5N$. Procedimento análogo pode ser efetuado tomando-se a matriz $\mathbf{C}$ em (2.123).

Finalmente, observa-se que os métodos com pré-condicionamento serão indicados por GCD, GCSS e GCGS respectivamente para os casos de matriz de pré-condicionamento diagonal, SSOR e GS simétrico.

2.7 Critérios de Convergência

Como visto anteriormente, os métodos iterativos produzem uma sequência de aproximações $\{\mathbf{u}^{(n)}\}$ convergindo para a solução $\bar{\mathbf{u}}$ de (1.30). Assim, torna-se fundamental estabelecer critérios de convergência visando encerrar o processo iterativo. Um bom critério de parada deve possuir as seguintes características [15]:

- identificar quando o erro $\mathbf{e}^{(n)}$ é suficientemente pequeno;
- parar se o erro não está decrescendo ou se a taxa de decaimento é muito baixa;
- limitar a quantidade máxima de iterações.

Deseja-se parar as iterações quando a magnitude do vetor $\mathbf{e}^{(n)}$ for pequena. No entanto, é impossível empregar $\mathbf{e}^{(n)}$ como critério de parada, procurando-se limitá-lo em função do resíduo $\mathbf{r}^{(n)}$. As

magnitudes destes vetores são determinadas através de uma norma adequada, tais como as normas euclidiana $||\cdot||_2$ e infinita $||\cdot||_\infty$.

Denomina-se como erro posterior a norma $||\mathbf{e}^{(n)}||$. Por sua vez, o erro anterior é definido como a menor variação entre $\max\{||\delta\mathbf{A}||/||\mathbf{A}||, ||\delta\mathbf{b}||/||\mathbf{b}||\}$, de tal forma a fazer $\mathbf{u}^{(i)}$ a solução exata de $(\mathbf{A}+\delta\mathbf{A})\mathbf{u}^{(i)} = (\mathbf{b}+\delta\mathbf{b})$.

Se os valores em $\mathbf{A}$ e $\mathbf{b}$ possuem erros provenientes de outros cálculos, não é interessante iterar até que $\delta\mathbf{A}$ e $\delta\mathbf{b}$ sejam inferiores a estes erros. Por exemplo, se ξ é a precisão da máquina, não vale a pena atingir a condição $||\delta\mathbf{A}|| < \xi||\mathbf{A}||$, pois os erros de arrendondamento nos elementos de $\mathbf{A}$ para se adequar ao valor da precisão ξ serão superiores.

É possível limitar o erro posterior em função do anterior através da equação do resíduo,

$$\mathbf{e}^{(i)} = \mathbf{u}^{(i)} - \mathbf{u} = \mathbf{A}^{-1}(\mathbf{A}\mathbf{u}^{(i)} - \mathbf{b}) = \mathbf{A}^{-1}\mathbf{r}^{(i)} \tag{2.127}$$

Logo, conhecendo-se uma estimativa para $||\mathbf{A}^{-1}||$ vem que,

$$||\mathbf{e}^{(i)}|| \leq ||\mathbf{A}^{-1}||\,||\mathbf{r}^{(i)}|| \tag{2.128}$$

Assim, pode-se empregar o erro anterior para a definição dos seguintes critérios de convergência [15]:

Critério 1 : $||\mathbf{r}^{(i)}|| \leq \xi$

Tem-se, neste caso, um limite superior para o erro posterior da forma $||\mathbf{e}^{(i)}|| \leq \xi||\mathbf{A}^{-1}||$;

Critério 2 : $||\mathbf{r}^{(i)}|| \leq \xi(||\mathbf{A}||\,||\mathbf{u}^{(i)}|| + ||\mathbf{b}||)$

Este critério é equivalente a requerer que $||\delta\mathbf{A}|| \leq \xi||\mathbf{A}||$ e $||\delta\mathbf{b}|| \leq \xi||\mathbf{b}||$, implicando no seguinte limite para o erro posterior,

$$||\mathbf{e}^{(i)}|| \leq \xi||\mathbf{A}^{-1}||\,(||\mathbf{A}||\,||\mathbf{u}^{(i)}|| + ||\mathbf{b}||)$$

Critério 3 : $||\mathbf{r}^{(i)}|| \leq \xi\,||\mathbf{b}||$

Novamente, os termos do erro anterior satisfazem $\delta\mathbf{A} = \mathbf{0}$ e $||\delta\mathbf{b}|| \leq \xi||\mathbf{b}||$. Este critério não é recomendado para matrizes mal-condicionadas. No entanto, esta deficiência não está presente nos problemas analisados adiante, desde que os elementos da malha tenham uma boa qualidade em relação à forma.

Critério 4 : $||\mathbf{r}^{(i)}|| \leq \xi\,||\mathbf{u}^{(i)}||/||\mathbf{A}^{-1}||$

É possível especificar uma precisão relativa ξ para a solução $\mathbf{u}^{(i)}$, ou seja,

$$\frac{||\mathbf{e}^{(i)}||}{||\mathbf{u}^{(i)}||} \leq \frac{||\mathbf{A}^{-1}||\,||\mathbf{r}^{(i)}||}{||\mathbf{u}^{(i)}||} \leq \xi$$

Para comparar a performance dos métodos iterativos abordados, adotou-se o critério de convergência 3 baseado na norma relativa do resíduo $||\mathbf{r}^{(n)}||_l$, ou seja,

$$||\mathbf{r}^{(n)}||_l = \frac{||\mathbf{A}\mathbf{u}^{(n)} - \mathbf{b}||_l}{||\mathbf{b}||_l} < \xi \quad l = 2, \infty \tag{2.129}$$

Visando uniformizar os resultados, selecionou-se a precisão ξ, de forma que o erro relativo entre as soluções direta de Gauss $\mathbf{u}_{dir}$ e iterativa $\mathbf{u}_{ite}$ fosse no máximo $0,1\%$, ou seja,

$$||e_r^{dir/ite}||_l = \frac{||\mathbf{u}_{dir} - \mathbf{u}_{ite}||_l}{||\mathbf{u}_{dir}||_l} < 0,1\% \tag{2.130}$$

Observa-se que são necessárias $2Nm$ operações para o cálculo do critério de convergência (2.129), sendo equivalente a uma iteração do método. Para reduzir este custo, pode-se verificar a convergência a cada 5 ou 10 iterações. No entanto, para os métodos de gradiente conjugado (GC, GCD, GCSS, GCGS), o gradiente em cada iteração corresponde ao próprio resíduo. Assim, tem-se o custo do critério apenas para os métodos de Jacobi, GS, SOR e CHSS.

De forma geral, seleciona-se ξ observando o grau de incerteza nos elementos de $\mathbf{A}$ e $\mathbf{b}$ em relação a $||\mathbf{A}||$ e $||\mathbf{b}||$, respectivamente. Por exemplo, $\xi = 10^{-6}$ significa que $\mathbf{A}$ e $\mathbf{b}$ possuem erros na faixa de $\pm 10^{-6}\,||\mathbf{A}||$ e $\pm 10^{-6}\,||\mathbf{b}||$. A precisão com que o algoritmo irá calcular a solução $\mathbf{u}$ não será maior que as incertezas nos termos do sistema de equações.

Um parâmetro importante é o número máximo de iterações visando garantir o término do algoritmo, caso o critério de convergência não seja satisfeito. Outra estratégia de parada é observar quando há estagnação na taxa de convergência, encerrando então o processo. No entanto, esta possibilidade é depedente do método empregado e do problema em estudo.

2.8 Estudo de Casos

A Tabela 2.1 resume o número de operações e o espaço de memória dos vetores auxiliares para todos os métodos discutidos anteriormente. O parâmetro N_{it} indica o número de iterações para a convergência no caso dos algoritmos iterativos.

Método	Operações	Memória
Gauss	$N + \sum m_i^U(m_i^U + 7)/2$	N
Jacobi	$N_{it}N(2m-1)$	$3N$
GS	$N_{it}N(2m-1)$	$3N$
SOR	$2N_{it}Nm$	$3N$
SSOR	$4N_{it}Nm$	$3N$
CHSS	$N_{it}N(4m+1)$	$5N$
GC	$N_{it}N(2m+4)$	$4N$
GCD	$N_{it}N(2m+5)$	$5N$
GCSS	$N_{it}N(2m+7)$	$5N$
GCGS	$N_{it}N(2m+7)$	$5N$

Tabela 2.1: Número de operações e espaço de memória dos vetores auxiliares para os métodos direto e iterativos considerados.

Observa-se que as expressões na Tabela 2.1 consideram apenas o núcleo principal dos métodos. A demanda computacional real é superior àquela indicada devido a fatores como gerenciamento de memória, expressões condicionais, sincronismos, dentre outros. No entanto, estas relações são significativas para efeito de uma análise comparativa dos métodos, além do que o cálculo do número de operações a partir destas expressões independe da implementação adotada para os vários métodos. Deve-se ressaltar ainda que não estão contabilizados os custos de renumeração de nós, do procedimento de Gauss simbólico em matrizes esparsas e do cálculo das estimativas dos autovalores extremos através

do algoritmo de Lanczos no método CHSS. Nos casos dos procedimentos SOR e GCSS, empregou-se um valor fixo $\omega = 1,81$ como parâmetro de sobrerelaxação.

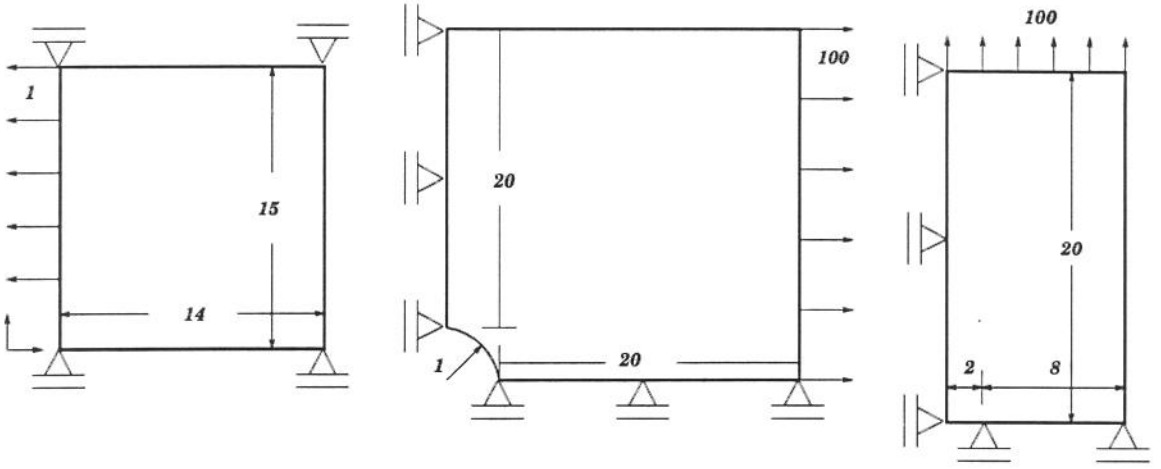


Figura 2.3: Problemas planos analisados: cilindro vertical, placa com furo e fratura ($E = 21 \times 10^5$, $\nu = 0,3$).

Para avaliar o desempenho dos vários algoritmos apresentados, considerou-se inicialmente os 3 problemas planos ilustrados na Figura 2.3. Foram geradas 4 malhas de elementos triangulares de 6 nós para cada um destes exemplos, como mostrado nas Figuras 2.4 a 2.6. No cilindro vertical, tem-se uma estrutura axissimétrica sem singularidades; na placa com furo tem-se um problema de estado plano de tensão com singularidade fraca; e finalmente o exemplo de fratura constitui-se novamente num caso de estado plano mas com uma forte singularidade. A Tabela 2.2 apresenta as principais características de cada uma das malhas, ou seja,

- número de nós;
- número de elementos;
- número de equações;
- m_{sky}, NElems_{sky} = altura média das colunas e número de coeficientes da matriz em colunas ascendentes com renumeração pelo algoritmo de Cuthill-Mckee [34, 66];
- m_{esp}^{dir}, $\text{NElems}_{esp}^{dir}$ = número médio de elementos por linha e número total de coeficientes na matriz esparsa com renumeração pelo algoritmo de grau mínimo [52, 97];
- m_{esp}^{ite}, $\text{NElems}_{esp}^{ite}$ = número médio de elementos por linha e número total de coeficientes não-nulos na matriz esparsa para os métodos iterativos;
- $||b||_2$, $||b||_\infty$ = normas euclidiana e infinita do vetor de carregamento.

Observa-se que na geração das malhas procurou-se apenas aumentar o número de equações. Assim, não se aplicou nenhum procedimento adaptável visando obter uma malha ótima para a análise de cada um dos problemas. Para os métodos iterativos, tomou-se sempre como solução inicial $\mathbf{u}^{(0)} = \mathbf{0}$.

Aplicaram-se os seguintes métodos na análise destes exemplos: Gauss, GS, SOR, CHSS, GC, GCD, GCSS e GCGS. Em todos os casos, verificou-se a convergência através de (2.129) nas normas euclidiana e infinita, com armazenamento da matriz $\mathbf{A}$ em colunas ascendentes e esparso. Para uniformizar

Cilindro vertical										
Nós	Elementos	Equações	m_{sky}	NElems_{sky}	m_{esp}^{dir}	$\text{NElems}_{esp}^{dir}$	m_{esp}^{ite}	$\text{NElems}_{esp}^{ite}$	$\|\|b\|\|_2$	$\|\|b\|\|_\infty$
531	244	1020	69,4	70739	30,7	31327	11,2	11397	2108.41	571.20
1595	760	3116	134,1	417803	47,9	149306	11,5	35931	1607.36	330.69
3069	1482	6036	183,3	1106335	57,7	348535	11,7	70395	1349.42	232.71
5293	2578	10452	242,7	2536176	67,9	709156	11,7	122733	1185.71	179.52

Placa com furo										
Nós	Elementos	Equações	m_{sky}	NElems_{sky}	m_{esp}^{dir}	$\text{NElems}_{esp}^{dir}$	m_{esp}^{ite}	$\text{NElems}_{esp}^{ite}$	$\|\|b\|\|_2$	$\|\|b\|\|_\infty$
533	248	1020	62,3	63581	34,4	35073	11,3	11481	603.49	222.22
1558	745	3034	111,5	338319	45,9	139123	11,5	35034	428.54	111.11
3172	1537	6226	153,1	952947	55,6	345996	11,7	72714	360.49	78.43
5078	2477	10014	214,1	2143473	65,3	654376	11,8	117678	303.66	55.56

Fratura										
Nós	Elementos	Equações	m_{sky}	NElems_{sky}	m_{esp}^{dir}	$\text{NElems}_{esp}^{dir}$	m_{esp}^{ite}	$\text{NElems}_{esp}^{ite}$	$\|\|b\|\|_2$	$\|\|b\|\|_\infty$
563	260	1078	70,4	75838	31,8	34297	11,2	12060	423.10	222.22
1538	735	3002	112,3	337056	44,1	132369	11,5	34650	329.98	133.33
3202	1547	6284	160,0	1005527	59,3	372732	11,7	73209	261.87	83.33
5216	2539	10282	211,0	2169416	66,2	680468	11,7	120600	214.27	55.56

Tabela 2.2: Atributos das malhas dos 3 problemas planos analisados.

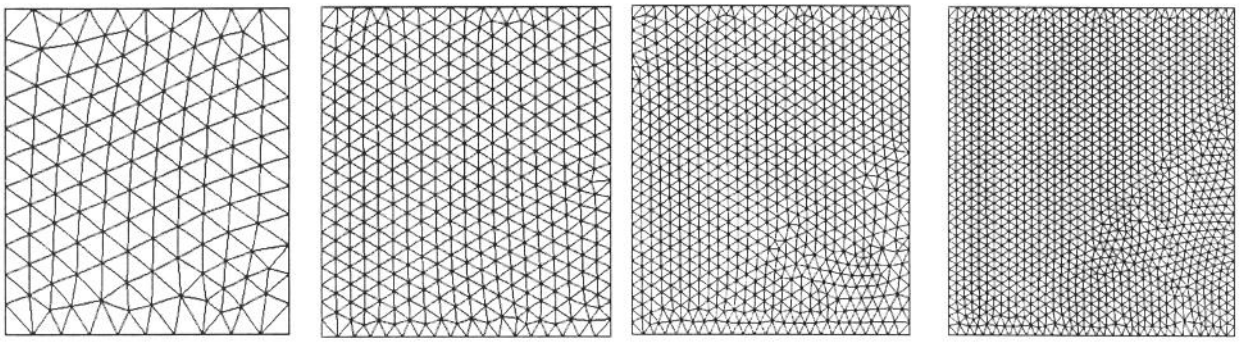


Figura 2.4: Malhas geradas para o cilindro vertical.

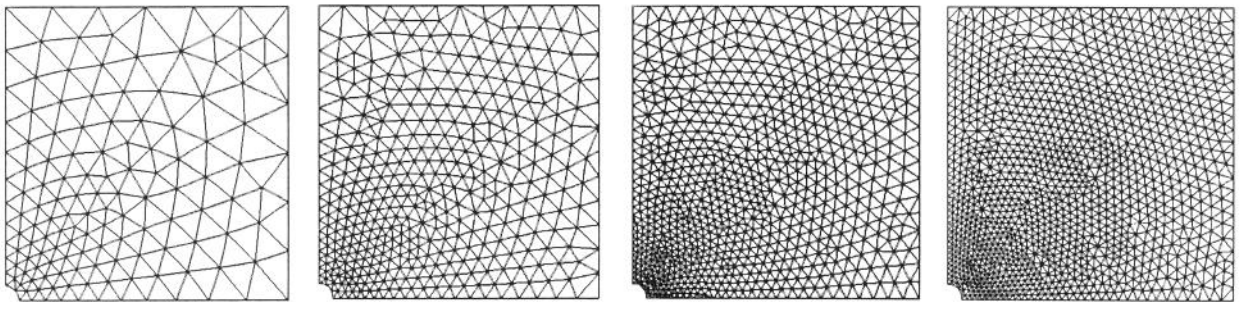


Figura 2.5: Malhas geradas para a placa com furo.

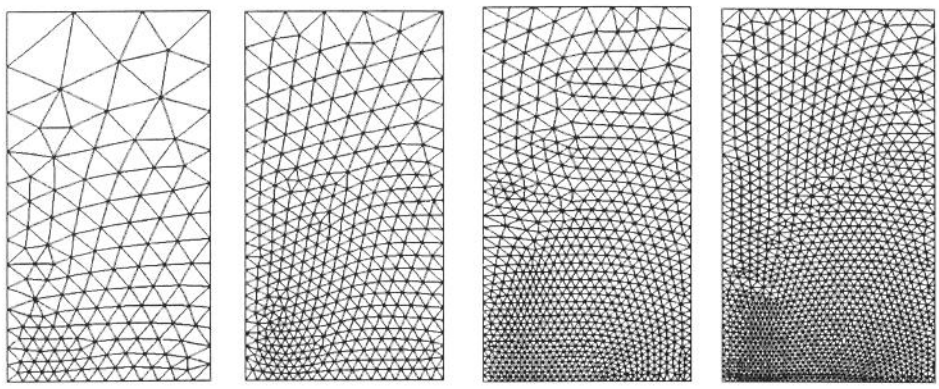


Figura 2.6: Malhas geradas para o problema de fratura.

os resultados, considerou-se a condição (2.130). Com este procedimento, obteve-se uma grande quantidade de resultados, apresentados no Apêndice B, tais como número de iterações para convergência, norma do resíduo e erro relativo. Neste capítulo, faz-se um resumo de todas as simulações, procurando ressaltar os aspectos mais relevantes.

A Tabela 2.3 contém os números de iterações, tomados do Apêndice B, em cada uma das quatro malhas dos 3 problemas analisados, com armazenamento em matriz esparsa. Não se considera o custo para o cálculo do critério de convergência, segundo a norma euclidiana, nos métodos SOR e CHSS.

Nas Figuras 2.7 a 2.9, tem-se gráficos em escala logarítmica com o número total de operações em MFlop calculado a partir das Tabelas 2.1 a 2.3. Foram ajustadas retas da forma $y = Cx^{\alpha}$ para cada um dos métodos, inclusive Gauss em colunas ascendentes e esparso, estando os coeficientes angulares α indicados na Tabela 2.3 e entre colchetes nas Figuras 2.7 a 2.9. As Figuras 2.10 a 2.12 mostram o comportamento do resíduo do critério de convergência na norma euclidiana para a primeira malha dos 3 problemas planos.

2.9 Discussão dos Resultados

A partir dos resultados obtidos observa-se que:

- os termos da matriz do sistema e dos vetores foram tratados com precisão dupla e portanto ocupam 8 bytes. Assim, as expressões para o espaço de memória devem ser multiplicadas pelo fator 8/1024 resultando valores em Kbytes.

- o método GS foi considerado apenas para a primeira malha dos 3 problemas, pois o número de iterações para a convergência nas demais malhas é muito elevado. Os resultados estão relacionados no Apêndice B.

- para a primeira malha dos 3 problemas, com cerca de 1000 equações, o método SOR apresenta uma performance comparável, ao até mesmo superior, aos métodos CHSS, GC e GCD. A medida que o número de equações aumenta, verifica-se uma queda progressiva na eficiência deste método. Procedimentos adaptáveis visando melhorar o valor de ω, e por consequência o comportamento do método, estão dados em [59].

- o número de iterações para a convergência com o método CHSS é inferior ao GC na maioria dos resultados obtidos. No entanto, o seu custo computacional é superior devido ao fator $4m$ presente na expressão para o cálculo do número de operações. Este fato pode ser comprovado

Cilindro vertical					
Método	1	2	3	4	α
Gauss - esparsa	–	–	–	–	1,83
Gauss - *skyline*	–	–	–	–	2,08
SOR	116	337	677	1121	2,00
CHSS	101	175	257	348	1,55
GC	124	193	265	355	1,47
GCD	102	162	187	245	1,38
GCSS	75	111	165	220	1,48
GCGS	57	91	102	132	1,36

Placa com furo					
Método	1	2	3	4	α
Gauss - esparsa	–	–	–	–	1,68
Gauss - *skyline*	–	–	–	–	2,07
SOR	287	1045	1783	3040	2,03
CHSS	127	228	317	481	1,58
GC	138	247	351	448	1,53
GCD	129	203	285	429	1,52
GCSS	64	142	204	257	1,62
GCGS	73	109	157	236	1,51

Fratura					
Método	1	2	3	4	α
Gauss - esparsa	–	–	–	–	1,78
Gauss - *skyline*	–	–	–	–	1,96
SOR	369	1139	2242	3704	2,04
CHSS	147	252	373	476	1,54
GC	153	250	378	477	1,53
GCD	140	237	349	447	1,53
GCSS	86	139	207	251	1,50
GCGS	79	131	191	244	1,52

Tabela 2.3: Número de iterações para os métodos adotados e coeficientes das retas ajustadas.

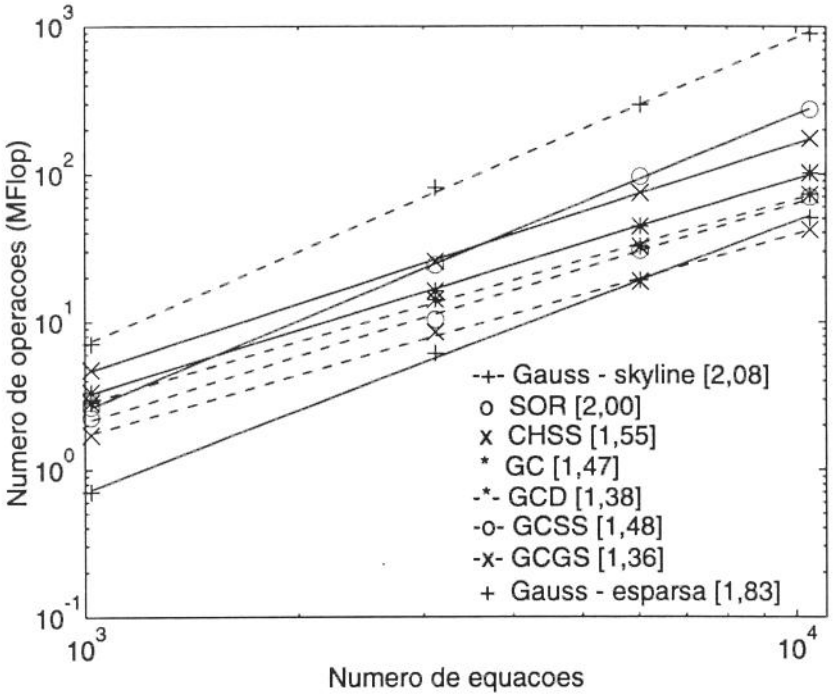


Figura 2.7: Cilindro vertical: número de operações para os métodos de Gauss e iterativos.

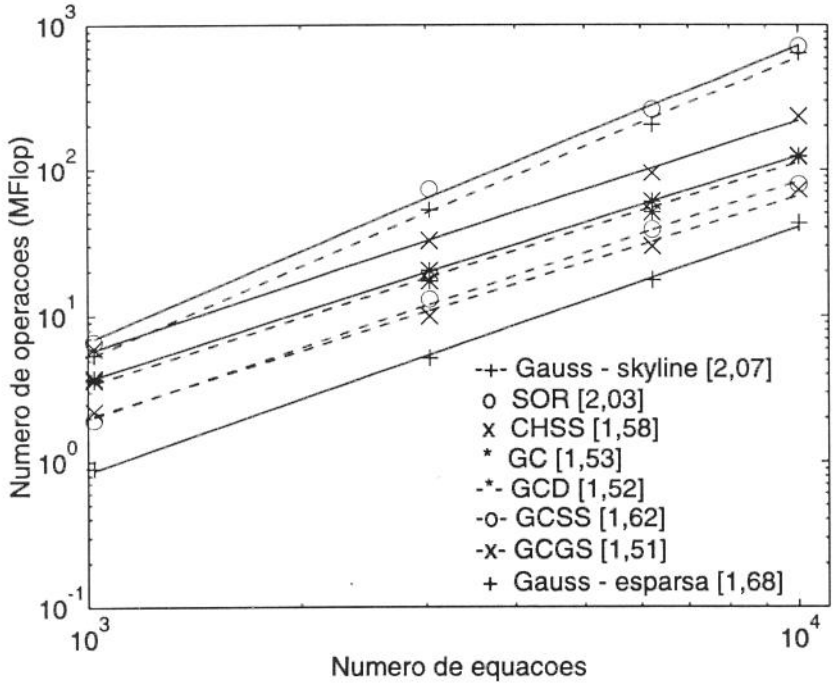


Figura 2.8: Placa com furo: número de operações para os métodos de Gauss e iterativos.

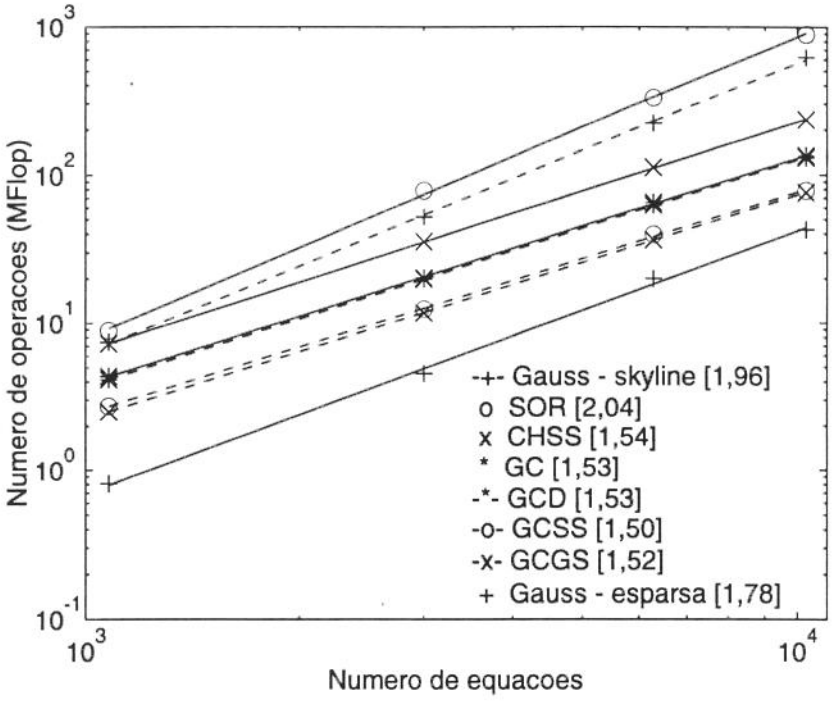


Figura 2.9: Fratura: número de operações para os métodos de Gauss e iterativos.

nas Figuras 2.7 a 2.9, pois as retas correspondentes ao CHSS estão sempre acima daquelas do GC. Como indicado em [59], a performance do algoritmo GC com pré-condicionadores é superior ao CHSS.

As estimativas para os autovalores obtidas pelo método de Lanczos permitiram uma redução significativa no número de iterações em relação ao procedimento adaptável dado em [59].

Uma das vantagens da aceleração de Chebyshev em relação ao método de Gradiente Conjugado está em não efetuar produtos internos, facilitando a sua paralelização. Observa-se que os produtos internos constituem-se em pontos de sincronismo em versões paralelas do algoritmo de GC [15].

- no método GCD, o número de iterações é inferior em relação aos procedimentos CHSS e GC. No entanto, a medida que a singularidade torna-se mais forte, verifica-se um equilíbrio nestes procedimentos. Assim, tomando-se a malha 4 dos 3 problemas analisados, no cilindro vertical a performance do GCD é bem superior ao CHSS e GC, enquanto para os outros exemplos tem-se um equilíbrio.

  Este fato também pode ser observado comparando-se o número de operações dos métodos GC e GCD nas Figuras 2.7 a 2.9. No primeiro caso, as retas encontram-se suficientemente espaçadas, tornando-se mais próximas nos outros dois casos, não se observando praticamente nenhuma diferença no problema de fratura da Figura 2.9.

- as versões pré-condicionadas do método de Gradiente Conjugado GCSS e GCGS permitiram reduzir significativamente o número de iterações em relação aos demais métodos. Na maioria dos casos, o algoritmo GCGS foi superior ao GCSS. No entanto, empregando $\omega = 0,5$ na equação (2.122), tem-se o mesmo comportmaento para os dois métodos.

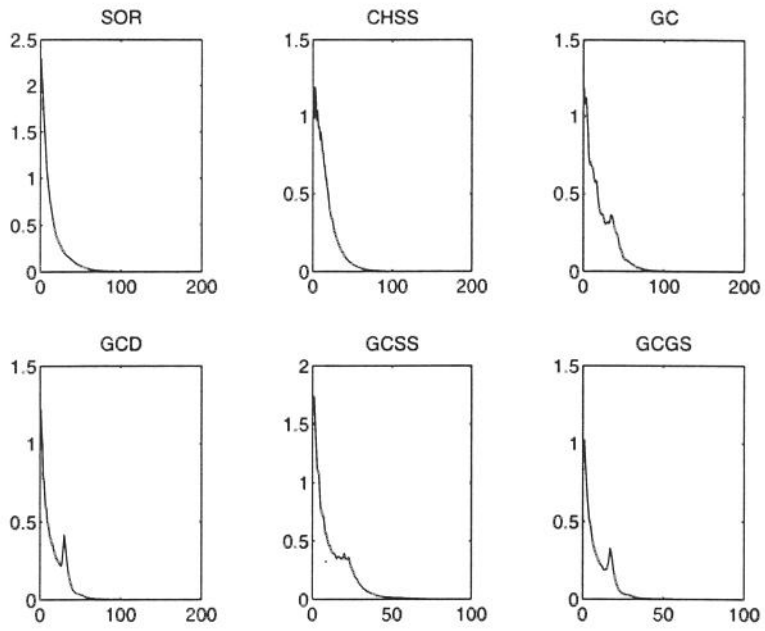


Figura 2.10: Cilindro vertical: comportamento da norma do resíduo $||\mathbf{r}||_2$ na primeira malha.

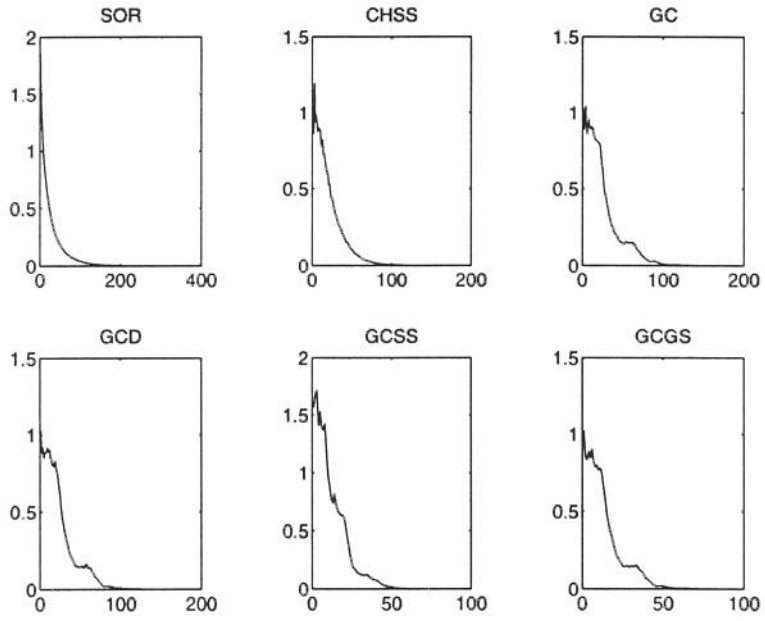


Figura 2.11: Placa com furo: comportamento da norma do resíduo $||\mathbf{r}||_2$ na primeira malha.

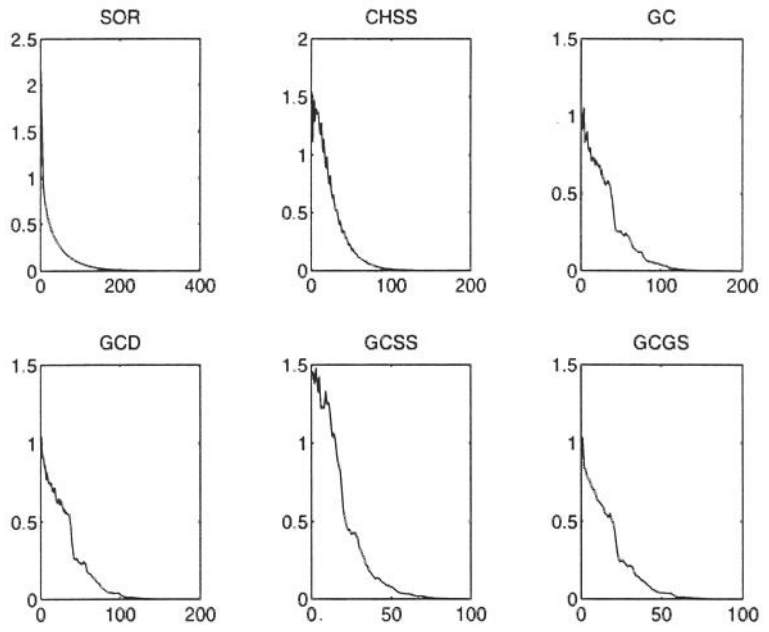


Figura 2.12: Fratura: comportamento da norma do resíduo $||\mathbf{r}||_2$ na primeira malha.

Novamente, acontece um comportamento semelhante ao método GCD, ou seja, a medida que se aumenta a singularidade, a diferença entre os algoritmos GCSS e GCGS torna-se progressivamente menor, conforme apresentado nas Figuras 2.7 a 2.9.

- em geral, para todas as simulações efetuadas, visando obter um erro relativo $||e_r||_l$ com a mesma ordem de grandeza, tomou-se uma precisão ξ menor ao se empregar a norma infinita.

- no caso de métodos iterativos, torna-se essencial utilizar um armazenamento em matrizes esparsas. Como pode ser visto na Tabela 2.2, no caso das 3 malhas mais finas, apenas cerca de 5% dos elementos são não-nulos. No caso do método de Gauss, também se verifica uma redução no número de elementos, mas não tão substancial devido ao processo de fatoração, onde são gerados novos elementos não-nulos. Além disso, como já mencionado, para os métodos iterativos com matrizes esparsas não é necessário renumerar de forma ótima as equações.

- em geral, os métodos iterativos baseados em gradiente conjugado são superiores em relação ao procedimento de Gauss em colunas ascendentes. No entanto, considerando apenas o número de operações, o método de Gauss esparso é superior até um certo número de equações dependente do problema em estudo. Nas Figuras 2.7 a 2.9, verificam-se os cruzamentos entre as retas dos métodos de Gauss e GCGS, indicando um mesmo custo computacional, respectivamente para 6438, 182310 e 81181 equações.

- o problema de fratura apresenta uma forte singularidade, implicando num maior número de iterações para a convergência em relação a placa com furo. Por sua vez, este problema com singularidade moderada demanda mais iterações que o caso de cilindro vertical. Portanto, como esperado, a presença de singularidades afeta diretamente a performance dos métodos iterativos.

- em relação ao comportamento do resíduo, ilustrado nas Figuras 2.10 a 2.12, observa-se um decaimento suave para o método SOR. Para as demais técnicas, o resíduo decresce em patamares

ocorrendo maiores instabilidades nas iterações iniciais. Nota-se em alguns casos um crescimento pronunciado da norma resíduo em algumas iterações, como por exemplo nos métodos GCD e GCGS na Figura 2.10. De forma geral, tem-se uma queda mais instável do resíduo a medida que a singularidade aumenta.

Das simulações numéricas anteriores, verifica-se que dentre os métodos iterativos, o algoritmo GCGS é superior em relação aos demais. Tomou-se o problema de fratura com dimensões 100 vezes maiores que aquelas indicadas na Figura 2.3, com o objetivo de estudar o comportamento dos métodos de Gauss e GCGS a medida que se aumenta o número de equações. Foram geradas 7 malhas com elementos triangulares de 6 nós, estando as características destas malhas indicadas na Tabela 2.4.

Nós	Elementos	Equações	m_{sky}	NElems_{sky}	m_{esp}^{dir}	$\text{NElems}_{esp}^{dir}$	m_{esp}^{ite}	$\text{NElems}_{esp}^{ite}$
654	303	1254	73,2	91798	34,3	43016	11,2	14076
1817	868	3540	129,0	456687	46,7	165205	11,5	40821
3006	1449	5888	169,6	998440	50,7	298805	11,6	68451
5190	2527	10226	213,1	2179279	60,6	619395	11,7	119958
10304	5055	20390	298,2	6079571	72,8	1483760	11,8	240756
16272	8013	32272	376,1	12136114	81,2	2620189	11,9	382305
27765	13722	55376	503,7	27895204	93,0	5148300	11,9	658151

Tabela 2.4: Atributos das malhas do problema de fratura modificado para matrizes em colunas ascendentes (*sky*) e esparsa (*esp*); métodos direto (*dir*) e iterativo (*ite*).

A Tabela 2.5 apresenta o número de iterações para o método GCGS em cada uma das malhas, utilizando precisões $\xi = 10^{-2}$ e $\xi = 10^{-3}$. As Figuras 2.13 a 2.15 ilustram o número de operações, o espaço de memória (matriz + vetores auxiliares) e o número médio de elementos por linha/coluna em função do número de equações. Os coeficientes angulares das retas das Figuras 2.13 e 2.14 estão indicados entre colchetes e também na Tabela 2.6.

Malha	1	2	3	4	5	6	7
NIT ($\xi = 10^{-2}$)	71	122	159	206	293	374	493
NIT ($\xi = 10^{-3}$)	87	155	189	251	358	459	676

Tabela 2.5: Número de iterações para GCGS nas 7 malhas do problema de fratura modificado.

Método	Operações	Memória
Gauss - esparsa	1,70	1,26
Gauss - *skyline*	2,00	1,50
GCGS ($\xi = 10^{-2}$)	1,52	1,01
GCGS ($\xi = 10^{-3}$)	1,54	1,01

Tabela 2.6: Coeficientes angulares das retas do problema de fratura modificado.

A partir dos resultados para o problema de fratura modificado, acrescentam-se as seguintes considerações:

- para casos bidimensionais, embora os coeficientes angulares das retas de custo computacional dos algoritmos tipo gradiente sejam menores em relação ao método de Gauss em matriz esparsa, este último apresenta melhor desempenho. Como será visto no Capítulo 4, esta conclusão não é válida para problemas tridimensionais.

A precisão ξ afeta diretamente a performance da técnica GCGS. Em particular para o caso de fratura, o erro relativo (2.130) foi inferior a 1% e 0.1% para $\xi = 10^{-2}$ e $\xi = 10^{-3}$, respectivamente. Para o critério de convergência adotado, $\xi = 10^{-3}$ parece mais adequado.

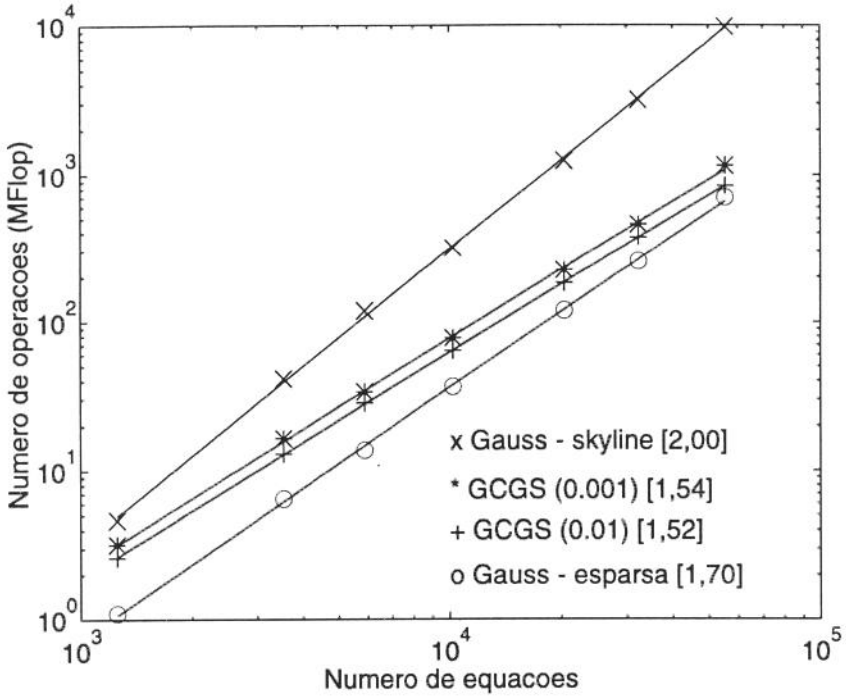


Figura 2.13: Fratura modificado: número de operações (MFlop) para o método de Gauss em matrizes esparsa e colunas ascendentes; método iterativo GCGS com precisões de $\xi = 10^{-2}$ e $\xi = 10^{-3}$.

- o espaço de memória indicado na Figura 2.14 considera apenas a matriz e os vetores auxiliares. Não se incluem, por exemplo, estruturas de dados aplicadas ao procedimento de Gauss simbólico em matrizes esparsas, as quais passam a ocupar um espaço considerável a medida que se aumenta o número de equações. Logicamente, este último aspecto depende da implementação do procedimento.

  Como esperado, a demanda em termos de memória para matriz esparsa é inferior para GCGS em relação ao método de Gauss. Como esta demanda é função não apenas do número de equações, mas também do número médio de elementos por linha/coluna da matriz, pode-se observar como este parâmetro varia ao longo das malhas na Figura 2.15. Para o caso de colunas ascendentes, ocorre um crescimento pronunciado. No caso do método de Gauss em matriz esparsa, observa-se uma variação quasi-linear, enquanto que para os métodos iterativos tem-se um valor praticamente constante.

- de forma geral, o número de equações a partir do qual tem-se um mesmo custo computacional para os procedimentos de Gauss e GCGS é dependente das características do problema.

  Um aspecto fundamental a ser considerado é a demanda em termos de memória, a qual cresce de forma mais pronunciada para o algoritmo de Gauss, não apenas para os elementos do sistema de equações, mas também no procedimento simbólico. Assim, para números crescentes de equações, o método de Gauss necessitará preliminarmente de memória em disco (*swap file*) fazendo cair a sua performance.

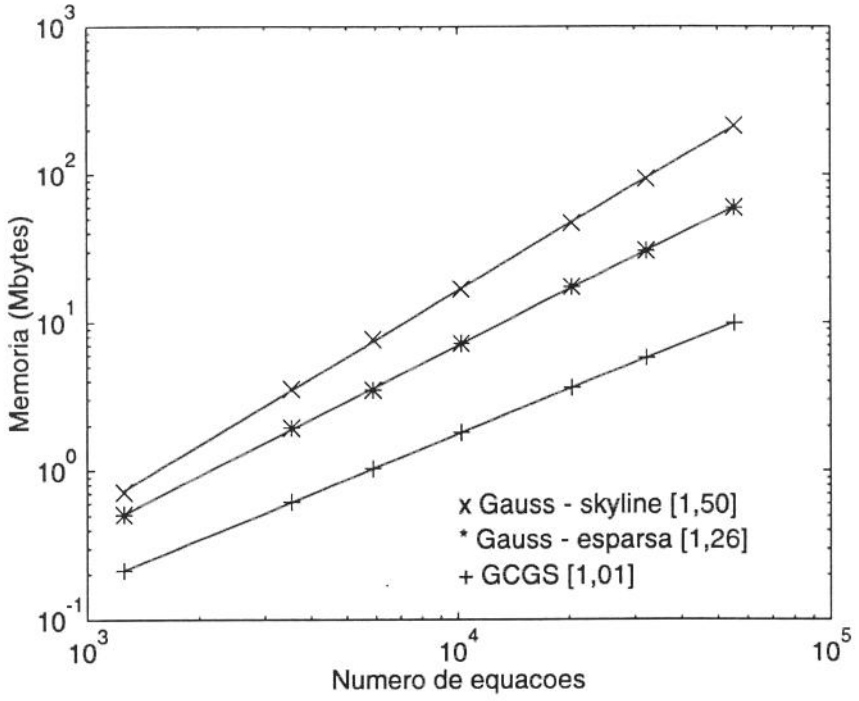


Figura 2.14: Fratura modificado: espaço de memória (Mbytes) para armazenamento em colunas ascendentes e esparso para o método de Gauss e esparso para o algoritmo GCGS.

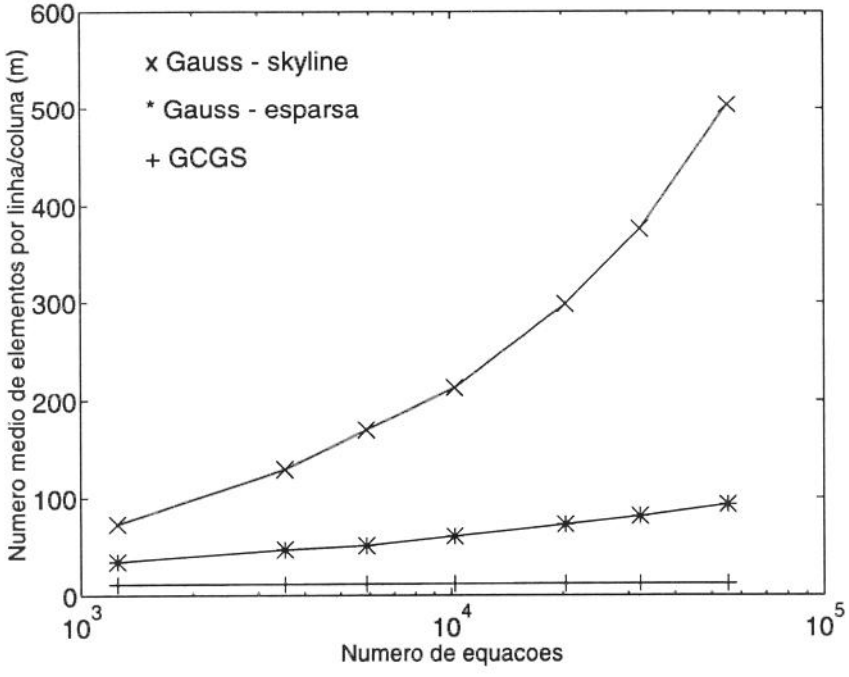


Figura 2.15: Fratura modificado: número médio de elementos por linha ou coluna para o método de Gauss e técnicas iterativas.

Desta forma, a utilização da técnica GCGS mostra-se competitiva em relação ao método de Gauss, principalmente para sistemas de alta ordem pois demanda um menor espaço de memória. Para casos tridimensionais, pode-se tornar inviável armazenar toda a matriz **A** para o método de Gauss, sendo necessário, então, aplicar procedimentos elemento-elemento ou recorrer a uma técnica iterativa.

Capítulo 3

ESTIMADOR DE ERRO E RECUPERADORES DE TENSÃO

Neste capítulo, apresenta-se o estimador de erro ZZ [112] baseado numa técnica de ponderação. Este estimador requer um procedimento para recuperar as tensões de forma mais precisa a partir da solução aproximada u_h obtida pelo MEF. Discutem-se alguns algoritmos para esta finalidade, incluindo a proposta original presente em [44, 112], além de outros onde se faz uma expansão polinomial das tensões ao longo de um conjunto de elementos [117, 119], permitindo incluir equações de equilíbrio [105] e condições de contorno [106].

3.1 Estimador Zhu-Zienkiewicz (ZZ)

Considerando o problema elástico, a forma bilinear (1.25) permite definir a norma de energia,

$$||u||_a = \sqrt{a(u,u)} = \left(\int_\Omega \mathbf{D}^{-1}\mathbf{T}(u)\cdot\mathbf{T}(u)\,d\Omega\right)^{\frac{1}{2}} \tag{3.1}$$

Dadas as soluções exata u e aproximada u_h, assim como as respectivas tensões $\mathbf{T}$ e $\mathbf{T}_h$, emprega-se (3.1) para obter as normas $||u||_a$ e $||u_h||_a$. Subtraindo $||u_h||_a^2$ de $||u||_a^2$, tem-se a norma de energia do erro, ou seja,

$$||e||_a = ||u-u_h||_a = (a(u-u_h,u-u_h))^{\frac{1}{2}} = \left(\int_\Omega \mathbf{D}^{-1}(\mathbf{T}-\mathbf{T}_h)\cdot(\mathbf{T}-\mathbf{T}_h)\,d\Omega\right)^{\frac{1}{2}} \tag{3.2}$$

Como a solução exata $(u,\mathbf{T})$ não é conhecida, calcula-se uma estimativa do erro empregando um recuperador de tensões para uma obter uma melhor aproximação $\mathbf{T}^*$ para $\mathbf{T}$ em relação a $\mathbf{T}_h$.

Desta maneira, a norma da estimativa do erro é obtida a partir de (3.2). Logo,

$$||e^*||_a = \left(\int_\Omega \mathbf{D}^{-1}(\mathbf{T}^*-\mathbf{T}_h)\cdot(\mathbf{T}^*-\mathbf{T}_h)\,d\Omega\right)^{\frac{1}{2}} \tag{3.3}$$

Define-se o erro percentual relativo global como,

$$\eta = \frac{||e||_a}{||u||_a}\times 100\% \tag{3.4}$$

Portanto, dado um erro percentual admissível $\bar{\eta}$, ao final da análise deseja-se que $\eta \leq \bar{\eta}$. Como $||e||_a$ não pode ser calculado, pois u não é conhecido, utiliza-se a estimativa (3.3) em (3.4) obtendo-se,

$$\eta \approx \frac{||e^*||_a}{(||u_h||_a^2 + ||e^*||_a^2)^{\frac{1}{2}}} \leq \bar{\eta} \tag{3.5}$$

Nos algoritmos adaptáveis, deseja-se alcançar a uniformidade do erro, ou seja, pretende-se que em todos os elementos da malha, o erro não supere um certo valor estabelecido. Por outro lado, o quadrado da norma do erro pode ser calculado pela contribuição dos N^{els} elementos da malha, ou seja,

$$||e^*||_a^2 = \sum_{i=1}^{N^{els}} ||e_i^*||_a^2 \tag{3.6}$$

Da condição de uniformidade do erro,

$$||e^*||_a^2 = N^{els}\, ||e_{ad}^*||_a^2 \tag{3.7}$$

sendo $||e_{ad}^*||_a$ a norma de energia do erro admissível em cada elemento.

Substituindo (3.7) em (3.5), determina-se a seguinte relação,

$$\begin{aligned} \eta &= \frac{(N^{els}||e_{ad}^*||_a^2)^{\frac{1}{2}}}{(||u_h||_a^2 + ||e^*||_a^2)^{\frac{1}{2}}} \leq \bar{\eta} \\ &= \frac{\sqrt{N^{els}}||e_{ad}^*||_a}{(||u_h||_a^2 + ||e^*||_a^2)^{\frac{1}{2}}} \leq \bar{\eta} \\ ||e_{ad}^*||_a &\leq \bar{\eta}\frac{(||u_h||_a^2 + ||e^*||_a^2)^{\frac{1}{2}}}{\sqrt{N^{els}}} \end{aligned} \tag{3.8}$$

Definindo para cada elemento i ($i = 1, \ldots, N^{els}$) a razão ξ_i entre os erros no elemento $||e_i^*||_a$ e o admissível $||e_{ad}^*||_a$,

$$\zeta_i = \frac{||e_i^*||_a}{||e_{ad}^*||_a} \tag{3.9}$$

tem-se que,

- se $\zeta_i > 1$ deve-se refinar o elemento;
- se $\zeta_i < 1$ aumenta-se o tamanho do elemento ou mantém-se o mesmo inalterado.

Observa-se que em [112], aplica-se um fator empírico de correção em ζ_i, dependente do tipo do elemento, sendo 1,3 e 1,4, respectivamente, para triângulos linear e quadrático.

A partir da teoria matemática do MEF, tem-se que a norma do erro da aproximação, assumindo ausência de descontinuidades, está limitada por,

$$||e|| \leq \alpha h^p \tag{3.10}$$

sendo h o tamanho do elemento e p a ordem das funções de interpolação. A partir desta expressão e da equação (3.9), determina-se o novo tamanho h_i' do elemento i como,

$$\zeta_i = \frac{\alpha(h_i)^p}{\alpha(h_i')^p} \rightarrow h_i' = \frac{h_i}{\zeta_i^{1/p}} \tag{3.11}$$

Desta forma, para $\zeta_i > 1$ o novo tamanho do elemento h_i' será menor que h_i; para $\zeta_i < 1$, tem-se $h_i' < h_i$. Para evitar crescimento excessivo no tamanho dos elementos, podendo ocasionar malhas de baixa qualidade, introduz-se um fator de amortecimento no crescimento dos elementos. Logo, para $\zeta_i < 1$, adota-se,

$$\zeta_i \leftarrow \zeta_i + \frac{(1-\zeta_i)}{2}$$

implicando que o elemento não crescerá mais que $2^{1/p}$ vezes o seu tamanho original a cada refinamento.

Calculado os novos tamanhos dos elementos, transferem-se os valores para os nós através de um processo análogo ao recuperador PA, discutido na próxima seção. Obtém-se uma função contínua no domínio de análise, definida pelos valores nodais da malha. Assim, o tamanho previsto do novo elemento num ponto qualquer do domínio é determinado, aplicando-se uma interpolação dos valores nodais do elemento contendo este ponto. Esta função é denominada *malha de parâmetros*, sendo empregada no refinamento de uma malha com o gerador ARANHA [43].

Como mencionado, um ponto fundamental é o algoritmo empregado para o cálculo de $\mathbf{T}^*$ a partir da aproximação $\mathbf{T}_h$, de tal forma que não se tenha um custo equivalente ao da análise. Nas próximas seções, apresentam-se 4 recuperadores de tensão para esta finalidade [44, 105, 106, 112, 117].

3.2 Recuperadores de Tensão

3.2.1 Ponderação pela área dos elementos (PA)

Originalmente em [112], as tensões $\mathbf{T}^*$ são obtidas empregando-se as mesmas funções de interpolação dos deslocamentos. Portanto,

$$\mathbf{T}^* = \mathbf{N}\bar{\mathbf{T}}^* \tag{3.12}$$

As tensões nodais $\bar{\mathbf{T}}^*$ são determinadas pela minimização do seguinte funcional,

$$\mathcal{F}(\bar{\mathbf{T}}^*) = \int_\Omega (\mathbf{T}^* - \mathbf{T}_h)^2 \, d\Omega \tag{3.13}$$

Por sua vez, a condição de mínimo deste funcional é dada por,

$$\int_\Omega \mathbf{N}^T(\mathbf{N}\bar{\mathbf{T}}^* - \mathbf{T}_h)\, d\Omega = 0 \tag{3.14}$$

A expressão (3.14) pode ser escrita como,

$$\mathbf{M}\bar{\mathbf{T}}^* = \mathbf{t} \tag{3.15}$$

onde,

$$\mathbf{M} = \int_\Omega \mathbf{N}^T\mathbf{N}\, d\Omega \qquad \mathbf{t} = \int_\Omega \mathbf{N}^T\mathbf{T}_h\, d\Omega \tag{3.16}$$

Devido a forma de $\mathbf{M}$, o custo da solução do sistema de equações anterior é da mesma ordem de grandeza daquele da análise. Para contornar esta limitação, emprega-se uma matriz diagonal, ao invés da matriz de massa consistente $\mathbf{M}$ [44]. Para isso, tomam-se os pontos de integração nas coordenadas locais dos nós, resultando numa matriz de massa diagonal, pois para um certo nó i a única função de forma não-nula, no caso igual a 1, é aquela correspondente a i. Assim, para cada elemento e empregando integração numérica vem que,

$$M_{ij}^e = \int_{\Omega^e} \phi_i \phi_j \, d\Omega \approx \sum_{l=1}^{N_{\text{nós}}} \phi_i(l)\phi_j(l)\, W_l \det \mathbf{J}_l \tag{3.17}$$

onde $N_{\text{nós}}$ é o número de nós do elemento, W é a ponderação e $\mathbf{J}$ é a matriz jacobiana. Como $\phi_i(l) = \delta_{il}$ (δ_{il} = delta de Kronecker), verifica-se que,

$$\begin{cases} M_{ii}^e = \sum_{l=1}^{N_{\text{nós}}} W_l \det \mathbf{J}_l \\ M_{ij} = 0 \end{cases} \tag{3.18}$$

Da mesma maneira, as forças nodais equivalentes são dadas por,

$$t_i^e = \sum_{l=1}^{N_{\text{nós}}} \mathbf{T}_h(l) W_l \det \mathbf{J}_l \tag{3.19}$$

Para elementos lineares, o jacobiano é igual a duas vezes a área do elemento. Logo, os termos da diagonal de $\mathbf{M}$ são proporcionais, em cada grau de liberdade de um nó l, ao somatório das áreas dos elementos compartilhando l. Da mesma maneira, os coeficientes do termo independente são iguais as tensões nodais ponderadas pelas respectivas áreas dos elementos.

No caso quadrático, o jacobiano não é constante ao longo do elemento. Porém, supondo que os elementos tenham boa qualidade em relação a forma, pode-se aplicar o mesmo critério de aproximação [44].

3.2.2 Expansão polinomial (EP)

É conhecido que as derivadas da solução aproximada u_h são superconvergentes em alguns pontos dos elementos da malha [14], ou seja, tem-se uma ordem de convergência maior que aquela prevista pela teoria matemática do MEF. Por sua vez, o estimador ZZ será assimptoticamente exato na norma de energia, caso a técnica de recuperação seja superconvergente [117].

Baseado nestes fatos, desenvolveu-se uma técnica de recuperação [117, 118] tomando-se uma expansão polinomial da função descrevendo as derivadas. Considera-se para cada nó, os elementos que o compartilham e realiza-se a expansão ao longo dos pontos superconvergentes dos elementos.

Assim neste procedimento, as tensões nodais $\bar{\mathbf{T}}^*$ pertencem a uma expansão polinomial $\mathbf{T}_p^*$, com a mesma ordem completa p que as funções de forma, ao longo do conjunto de elementos (*patch*) compartilhando o nó considerado, como ilustrado na Figura 3.1.

A expansão polinomial será empregada para cada componente de tensão ou derivada da seguinte maneira,

$$\mathbf{T}_p^* = \mathbf{P}\mathbf{a} \tag{3.20}$$

onde $\mathbf{P}$ contém os termos polinomiais apropriados e $\mathbf{a}$ são parâmetros desconhecidos. Para elementos unidimensionais de ordem p tem-se,

$$\mathbf{P} = \begin{bmatrix} 1 & x & x^2 & \dots & x^p \end{bmatrix} \quad \mathbf{a} = \begin{bmatrix} a_1 & a_2 & a_3 & \dots & a_{p+1} \end{bmatrix}^T \tag{3.21}$$

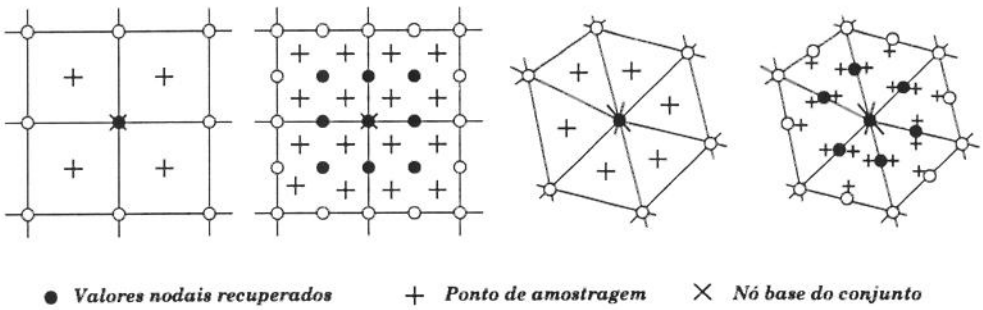


Figura 3.1: Conjuntos de elementos (*patch*) para um nó da malha [117].

Para casos bi e tridimensionais, consideram-se em $\mathbf{P}$ apenas os termos polinomiais completos, com exceção dos quadriláteros onde se recomenda empregar os mesmos termos das funções de forma do elemento [117]. Para expansões linear e quadrática em triângulos, tem-se, respectivamente,

$$\mathbf{P} = \begin{bmatrix} 1 & x & y \end{bmatrix} \quad \mathbf{P} = \begin{bmatrix} 1 & x & y & x^2 & xy & y^2 \end{bmatrix} \tag{3.22}$$

enquanto que para o quadrado linear inclui-se ainda o termo xy.

Os parâmetros $\mathbf{a}$ são determinados através de um ajuste por mínimos quadrados pelos n_p pontos superconvergentes do conjunto de elementos de um nó. Para isso, minimiza-se o funcional [117],

$$\begin{aligned} \mathcal{F}(\mathbf{a}) &= \sum_{i=1}^{n_p} \left(\mathbf{T}_h(x_i^s, y_i^s) - \bar{\mathbf{T}}_p^*(x_i^s, y_i^s) \right)^2 \\ &= \sum_{i=1}^{n_p} \left(\mathbf{T}_h(x_i^s, y_i^s) - \mathbf{P}(x_i^s, y_i^s)\mathbf{a} \right)^2 \end{aligned} \tag{3.23}$$

onde (x_i^s, y_i^s) $(i = 1, \ldots, n_p)$ são as coordenadas dos pontos de amostragem; $n_p = kN_p^{els}$ é o número total de pontos; k é o número de pontos em cada um dos N_p^{els} elementos do conjunto.

A condição de mínimo do funcional (3.23) é dada por,

$$\sum_{i=1}^{n_p} \mathbf{P}^T(x_i^s, y_i^s)\mathbf{P}(x_i^s, y_i^s)\mathbf{a} = \sum_{i=1}^{n_p} \mathbf{P}^T(x_i^s, y_i^s)\mathbf{T}_h(x_i^s, y_i^s) \tag{3.24}$$

Em forma matricial,

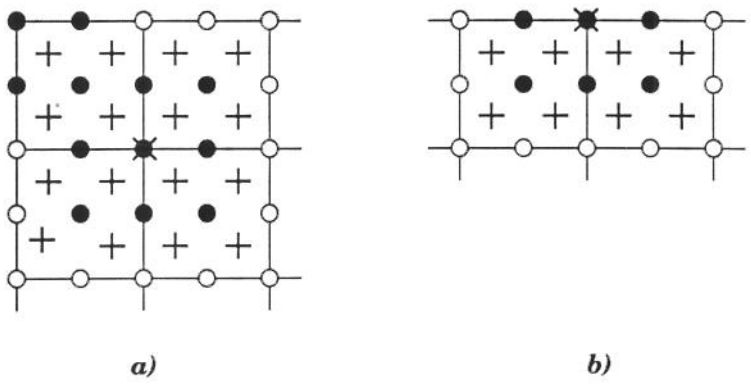
$$\mathbf{A}_p\mathbf{a} = \mathbf{b}_p \tag{3.25}$$

com

$$\mathbf{A}_p = \sum_{i=1}^{n} \mathbf{P}^T(x_i^s, y_i^s)\mathbf{P}(x_i^s, y_i^s) \quad \mathbf{b}_p = \sum_{i=1}^{n} \mathbf{P}^T(x_i^s, y_i^s)\mathbf{T}_h(x_i^s, y_i^s) \tag{3.26}$$

Verifica-se que o número de equações no sistema (3.25) é bastante pequeno, sendo igual ao número de coeficientes presentes em $\mathbf{P}$. Além disso, a matriz $\mathbf{A}_p$ é simétrica, sendo calculada uma única vez e utilizada para todas as componentes de tensão.

Uma vez que os parâmetros $\mathbf{a}$ estejam disponíveis, os valores nodais recuperados $\bar{\mathbf{T}}^*$ são determinados empregando-se (3.20) com as respectivas coordenadas locais dos nós (x_i^l, y_i^l). Observa-se que



a) *b)*

Figura 3.2: Recuperação das derivadas para os nós de contorno: a) vértice; b) face [117].

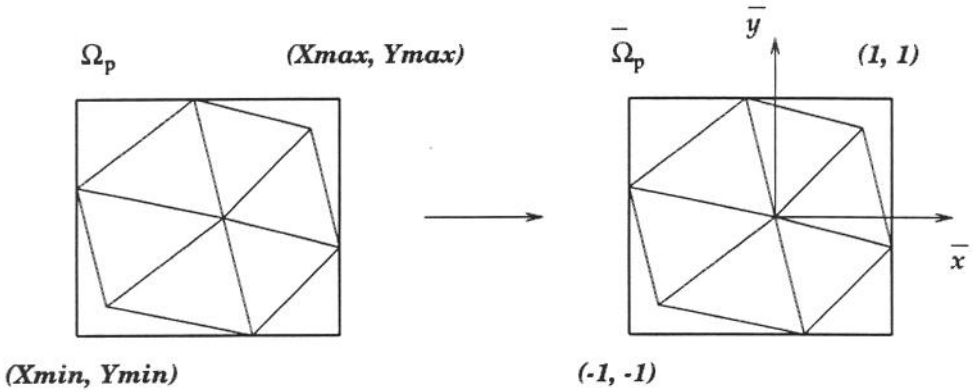


Figura 3.3: Transformação de domínio para o conjunto de elementos.

apenas os vértices do conjunto de elementos são considerados. Para os nós do meio de uma face ou internos do elemento, os valores recuperados são determinados por dois ou mais conjuntos, como pode ser visto na Figura 3.1, devendo-se efetuar uma média dos valores obtidos.

Para o contorno do domínio, tem-se as situações ilustradas na Figura 3.2, onde apenas um ou dois elementos compartilham um nó. Para um nó de vértice, o número de elementos do conjunto é insuficiente para determinar os parâmetros a, devendo-se utilizar o conjunto de um nó interno. No outro caso, o número de elementos é suficiente, recomendando-se, no entanto, também usar os coeficientes determinados por um conjunto interno [117].

Na Figura 3.1, pode-se observar os pontos de amostragem. Para os quadrados as derivadas superconvergentes ocorrem nos pontos de Gauss. No caso de triângulos, a existência de pontos superconvergentes não está totalmente comprovada [117]. Para elementos lineares, tomam-se as coordenadas do centróide, enquanto para os quadráticos consideram-se o centro das arestas do elemento. Neste último caso, experimentos numéricos apresentaram *ultraconvergência* para as tensões recuperadas com ordem $\mathcal{O}(h^4)$ [117].

Para elementos de ordem mais alta, a matriz $\mathbf{A}_p$ pode se tornar mal-condicionada, devido a presença de termos polinomiais com alto grau em função das coordenadas globais dos nós [119]. Aplica-se, então, uma transformação do domínio original Ω_p do conjunto de elementos para um outro normalizado $\bar{\Omega}_p$, onde todas as coordenadas estão no intervalo $-1 \leq x, y \leq 1$, como ilustrado na Figura 3.3. Para isso, define-se a janela contendo os elementos a partir dos pontos (x_{min}, y_{min}) e (x_{max}, y_{max}). A partir daí, tem-se uma transformação linear para cada uma das coordenadas x e y

da seguinte forma,

$$x = \frac{1}{2}(1+\bar{x})x_{max} + \frac{1}{2}(1-\bar{x})x_{max} = \frac{1}{2}(x_{max}+x_{min}) + \frac{1}{2}(x_{max}-x_{min})\bar{x} \quad = x_m + \frac{l_x}{2}\bar{x} \tag{3.27}$$

$$y = \frac{1}{2}(1+\bar{y})y_{max} + \frac{1}{2}(1-\bar{y})y_{max} = \frac{1}{2}(y_{max}+y_{min}) + \frac{1}{2}(y_{max}-y_{min})\bar{y} = y_m + \frac{l_y}{2}\bar{y} \tag{3.28}$$

Pode-se expressar, então, $\bar{x}$ e $\bar{y}$ em função de x e y como,

$$\bar{x} = \frac{2}{l_x}(x - x_m) \qquad \bar{y} = \frac{2}{l_y}(y - y_m) \tag{3.29}$$

3.2.3 Expansão polinomial com equações de equilíbrio (EPE)

Ao se aplicar a técnica de recuperação anterior, verifica-se uma taxa de convergência superior para os nós internos em relação àquela encontrada para os nós de contorno [105, 117]. Além disso, as tensões recuperadas não satisfazem a equação de equilíbrio do problema. Neste procedimento as componentes de tensão são acopladas através das equações de equilíbrio [105].

As tensões recuperadas nos pontos superconvergentes (x_i^s, y_i^s) pela expressão (3.20) são mais precisas que os valores $\mathbf{T}_h$ calculados nos mesmos pontos. Portanto, tem-se um resíduo $\mathbf{R}_T$ da seguinte forma,

$$\mathbf{R}_T = \mathbf{T}_p^*(x_i^s, y_i^s) - \mathbf{T}_h(x_i^s, y_i^s) = \mathbf{P}(x_i^s, y_i^s)\mathbf{a} - \mathbf{T}_h(x_i^s, y_i^s) \tag{3.30}$$

Como as tensões $\mathbf{T}_p^*$ não satisfazem a equação de equilíbrio, determina-se um outro resíduo dado por,

$$\mathbf{R}_{eq} = \nabla^T \mathbf{T}_p^* + \mathbf{f}_b \tag{3.31}$$

sendo $\mathbf{f}_b$ o vetor com as componentes das forças de corpo.

Para um problema de estado plano, o termo $\nabla^T \mathbf{T}_p^*$ é dado por,

$$\begin{aligned} \nabla^T \mathbf{T}_p^* &= \begin{bmatrix} \partial/\partial x & 0 & \partial/\partial y \\ 0 & \partial/\partial y & \partial/\partial x \end{bmatrix} \begin{Bmatrix} T_x^* \\ T_y^* \\ T_{xy}^* \end{Bmatrix} \\ &= \begin{bmatrix} (T_x^*)_x + (T_{xy}^*)_y \\ (T_y^*)_y + (T_{xy})_x \end{bmatrix} = \begin{bmatrix} \mathbf{P}_x & 0 & \mathbf{P}_y \\ 0 & \mathbf{P}_y & \mathbf{P}_x \end{bmatrix} \begin{Bmatrix} a_x \\ a_y \\ a_{xy} \end{Bmatrix} \end{aligned} \tag{3.32}$$

sendo $(T_x^*)_x = \frac{\partial T_x^*}{\partial x}$.

Os parâmetros $\mathbf{a}$ são determinados aplicando-se mínimos quadrados ao funcional [105],

$$\mathcal{F}(\mathbf{a}) = \sum_{i=1}^{n_p} \mathbf{R}_T^T \mathbf{R}_T + \int_{\Omega_p} \beta_p \mathbf{R}_{eq}^T \mathbf{R}_{eq}\, d\Omega_p \tag{3.33}$$

onde o parâmetro β_p é um número de penalização, caso se considere a equação de equilíbrio como uma restrição ao problema.

Substituindo as expressões (3.30) a (3.32) em (3.33) e aplicando a condição de mínimo de $\mathcal{F}(\mathbf{a})$, determina-se o sistema de equações,

$$\frac{\partial \mathcal{F}(\mathbf{a})}{\partial \mathbf{a}} = 0 :$$

$$\begin{bmatrix} \sum_{i=1}^{n_p} \mathbf{P}^T\mathbf{P} + \int \beta_p \mathbf{P}_x^T \mathbf{P}_x \, d\Omega_p & \mathbf{0} & \int \beta_p \mathbf{P}_x^T \mathbf{P}_y \, d\Omega_p \\ \mathbf{0} & \sum_{i=1}^{n_p} \mathbf{P}^T\mathbf{P} + \int \beta_p \mathbf{P}_y^T \mathbf{P}_y \, d\Omega_p & \int \beta_p \mathbf{P}_x^T \mathbf{P}_y \, d\Omega_p \\ \int \beta_p \mathbf{P}_x^T \mathbf{P}_y \, d\Omega_p & \int \beta_p \mathbf{P}_x^T \mathbf{P}_y \, d\Omega_p & \sum_{i=1}^{n_p} \mathbf{P}^T\mathbf{P} + \int \beta_p (\mathbf{P}_x^T \mathbf{P}_x + \mathbf{P}_y^T \mathbf{P}_y) \, d\Omega_p \end{bmatrix}$$

$$\begin{Bmatrix} \mathbf{a}_x \\ \mathbf{a}_y \\ \mathbf{a}_{xy} \end{Bmatrix} = \begin{Bmatrix} \sum_{i=1}^{n_p} \mathbf{P}^T\mathbf{T}_x^s - \int \beta_p \mathbf{P}_x^T f_b \, d\Omega_p \\ \sum_{i=1}^{n_p} \mathbf{P}^T\mathbf{T}_y^s - \int \beta_p \mathbf{P}_y^T f_b \, d\Omega_p \\ \sum_{i=1}^{n_p} \mathbf{P}^T\mathbf{T}_{xy}^s - \int \beta_p (\mathbf{P}_x^T + \mathbf{P}_y^T) f_b \, d\Omega_p \end{Bmatrix} \tag{3.34}$$

Pode-se utilizar a presença das equações de equilíbrio e empregar polinômios com uma ordem mais alta na expansão polinomial. Portanto, se as funções de forma para os deslocamentos têm ordem p, é possível interpolar as tensões com grau $p+1$ [105].

A solução do sistema (3.34) é possível apenas se o número de parâmetros em $\mathbf{a}$ é menor ou igual ao número de equações. Assim, a quantidade de termos na expansão polinomial deve ser limitada, de tal maneira que a ordem do vetor $\mathbf{a}$ não exceda o número de equações independentes para um certo conjunto de elementos.

Tal fato é de fundamental importância para os nós de contorno, cujas tensões devem ser recuperadas empregando-se conjuntos de contorno, exceto quando o número de termos em $\mathbf{a}$ é insuficiente. Nestes casos, utiliza-se um conjunto de um nó interno ou reduz-se a ordem dos polinômios em $\mathbf{P}$. Para problemas bidimensionais, conjuntos internos são usados para os nós de contorno apenas quando o conjunto correspondente a este nó possuir menos que 3 elementos [105]. Observa-se que para $\beta_p = 0$, tem-se o recuperador EP. Consideram-se ainda os casos degenerados: $(p, \beta_p \neq 0)$ e $(p+1, \beta_p = 0)$.

Novamente, aplica-se a transformação de domínios para o conjunto de elementos. Os termos envolvendo o produto $\mathbf{P}^T\mathbf{P}$ são tratados de forma análoga ao recuperador anterior. Entretanto, tem-se as expressões provenientes das equações de equilíbrio envolvendo as derivadas $\mathbf{P}_x$ e $\mathbf{P}_y$.

A partir das expressões (3.27) e (3.28), verifica-se que,

$$dx = \frac{l_x}{2} d\bar{x} \qquad dy = \frac{l_y}{2} d\bar{y} \tag{3.35}$$

Aplicando a regra da cadeia, tem-se as seguintes relações,

$$\mathbf{P}_x = \frac{2}{l_x} \mathbf{P}_{\bar{x}} \qquad \mathbf{P}_y = \frac{2}{l_y} \mathbf{P}_{\bar{y}} \tag{3.36}$$

Utilizando as expressões anteriores (3.35) e (3.36) nos termos contendo derivadas de $\mathbf{P}$ da expressão (3.34) obtém-se,

$$\beta_p \int_{\Omega_p} \mathbf{P}_x^T \mathbf{P}_x \, dx \, dy = \beta_p \int_{\bar{\Omega}_p} \left(\frac{2}{l_x}\right)^2 \mathbf{P}_{\bar{x}}^T \mathbf{P}_{\bar{x}} \frac{l_x}{2} \frac{l_y}{2} d\bar{x} \, d\bar{y} = \frac{l_y}{l_x} \beta_p \int_{\bar{\Omega}_p} \mathbf{P}_{\bar{x}}^T \mathbf{P}_{\bar{x}} \, d\bar{x} \, d\bar{y} \tag{3.37}$$

$$\beta_p \int_{\Omega_p} \mathbf{P}_y^T \mathbf{P}_y \, dx \, dy = \beta_p \int_{\bar{\Omega}_p} \left(\frac{2}{l_y}\right)^2 \mathbf{P}_{\bar{y}}^T \mathbf{P}_{\bar{y}} \frac{l_x}{2} \frac{l_y}{2} d\bar{x} \, d\bar{y} = \frac{l_x}{l_y} \beta_p \int_{\bar{\Omega}_p} \mathbf{P}_{\bar{y}}^T \mathbf{P}_{\bar{y}} \, d\bar{x} \, d\bar{y} \tag{3.38}$$

$$\beta_p \int_{\Omega_p} \mathbf{P}_x^T \mathbf{P}_y \, dx \, dy = \beta_p \int_{\bar{\Omega}_p} \frac{2}{l_x} \frac{2}{l_y} \mathbf{P}_{\bar{y}}^T \mathbf{P}_{\bar{y}} \frac{l_x}{2} \frac{l_y}{2} d\bar{x} \, d\bar{y} = \beta_p \int_{\bar{\Omega}_p} \mathbf{P}_{\bar{x}}^T \mathbf{P}_{\bar{y}} \, d\bar{x} \, d\bar{y} \tag{3.39}$$

Estes termos devem ser integrados numericamente considerando os pontos de Gauss para cada elemento. Por exemplo para a equação (3.39) tem-se,

$$\beta_p \int_{\bar{\Omega}_p} \mathbf{P}_{\bar{x}}^T \mathbf{P}_{\bar{y}} \, d\bar{x} \, d\bar{y} = \sum_{e=1}^{N_p^{els}} \sum_{l=1}^{N_{pi}} \mathbf{P}_{\xi}^T \mathbf{P}_{\eta} W_l \det \mathbf{J} \tag{3.40}$$

onde N_p^{els} é o número de elementos do conjunto; N_{pi} é o número de pontos de integração; (ξ, η) são as variáveis locais do elemento.

3.2.4 Expansão polinomial com equações de equilíbrio e condições de contorno (EPEC)

Os recuperadores anteriores não consideram as condições de contorno tais como deslocamentos e trações prescritos. Além disso, como já observado, a qualidade das derivadas recuperadas no contorno é inferior em relação aos pontos internos do domínio. Em geral, para problemas com deslocamentos impostos, os gradientes da solução aproximada são frequentemente imprecisos em pontos próximos ou sob o contorno.

O procedimento dado em [106] incorpora as condições de contorno através de um ajuste por mínimos quadrados com ponderações, numa tentativa de garantir que as derivadas recuperadas satisfaçam as condições presentes no contorno. Observa-se que este procedimento considera não apenas os pontos superconvergentes em tensão, mas também pontos no contorno e diferentes funções de ponderação.

Assume-se que o contorno Γ do domínio Ω está subdividido em $\Gamma = \Gamma_u \cup \Gamma_b$ com $\Gamma_u \cap \Gamma_b = \emptyset$. Logo, Γ_u denota a parte de Γ onde se tem as condições essenciais, neste caso, deslocamentos prescritos $u = u_b$ em Γ_u. Analogamente, em Γ_b aplicam-se as condições de contorno naturais, ou seja, trações prescritas $\nabla_n^T \mathbf{u} + \mathbf{t}_b = \mathbf{T}(\mathbf{u}^*)\mathbf{n} + \mathbf{t}_b = 0$ em Γ_b.

Definem-se os seguintes resíduos [106]:

$$\begin{aligned}
\mathbf{R}_T &= [w(\mathbf{T}^* - \mathbf{T}_h)]_s \\
\mathbf{R}_{eq} &= [\nabla^T \mathbf{T}^* + \mathbf{f}_b]_s \\
\mathbf{R}_u &= [w(\mathbf{u}^* - \mathbf{u}_h)]_r \\
\mathbf{R}_{t_b} &= [w(\nabla_n^T \mathbf{u}^* + \mathbf{t}_b)]_b \\
\mathbf{R}_{u_b} &= [w(\mathbf{u}^* - \mathbf{u}_b)]_u
\end{aligned} \tag{3.41}$$

sendo $\mathbf{R}_T$ e $\mathbf{R}_{eq}$ os resíduos nos pontos superconvergentes em tensão (s); $\mathbf{R}_u$ é o resíduo em deslocamentos para os pontos de amostragem (r); $\mathbf{R}_{u_b}$ e $\mathbf{R}_{t_b}$ representam, respectivamente, uma violação nas condições de contorno essenciais e naturais pela solução recuperada; w é uma função de ponderação; $\mathbf{u}^*$ é um campo de deslocamento interpolado com uma ordem mais alta que aquela presente para $\mathbf{u}_h$.

Os parâmetros $\mathbf{a}$ são determinados minimizando o seguinte funcional [106],

$$\begin{aligned}
\mathcal{F}(\mathbf{a}) &= F_1 + F_2 + F_3 + F_4 + F_5 \\
&= \sum_s \mathbf{R}_T^T \mathbf{D}^{-1} \mathbf{R}_T + \beta_p \int \mathbf{R}_{eq}^T \mathbf{D}^{-1} \mathbf{R}_{eq} \, d\Omega_p + \sum_r \mathbf{R}_u^T \mathbf{D}^{-1} \mathbf{R}_u + \\
&+ \sum_b \mathbf{R}_{t_b}^T \mathbf{D}^{-1} \mathbf{R}_{T_b} + \sum_u \mathbf{R}_{u_b}^T \mathbf{D}^{-1} \mathbf{R}_{u_b}
\end{aligned} \tag{3.42}$$

onde a matriz de elasticidade $\mathbf{D}$ pode ser substituída por $E\mathbf{I}$, sendo E o módulo de elasticidade longitudinal e $\mathbf{I}$ a matriz identidade.

Neste recuperador, emprega-se um campo de deslocamentos de ordem mais alta $\mathbf{u}^*$ com correspondentes tensões $\mathbf{T}^*$. Logo, para um problema de estado plano e utilizando notação matricial vem que,

$$\mathbf{u}_i^* = \bar{\mathbf{Q}}^* \mathbf{a}_i \;\; (i = x, y) \;\rightarrow\; \left\{ \begin{array}{c} \mathbf{u}_x^* \\ \mathbf{u}_y^* \end{array} \right\} = \left[\begin{array}{cc} \bar{\mathbf{Q}}^* & \mathbf{0} \\ \mathbf{0} & \bar{\mathbf{Q}}^* \end{array} \right] \left\{ \begin{array}{c} \mathbf{a}_x^* \\ \mathbf{a}_y^* \end{array} \right\} \;\rightarrow\; \mathbf{u}^* = \bar{\mathbf{Q}} \mathbf{a}^* \tag{3.43}$$

$$\begin{aligned} \mathbf{T}^* &= \mathbf{D} \nabla \mathbf{u}^* = \mathbf{D} \nabla \bar{\mathbf{Q}} \mathbf{a}^* \\ &= \mathbf{D} \left[\begin{array}{cc} \partial/\partial x & 0 \\ 0 & \partial/\partial y \\ \partial/\partial y & \partial/\partial x \end{array} \right] \left[\begin{array}{cc} \bar{\mathbf{Q}}^* & \mathbf{0} \\ \mathbf{0} & \bar{\mathbf{Q}}^* \end{array} \right] \left\{ \begin{array}{c} \mathbf{a}_x^* \\ \mathbf{a}_y^* \end{array} \right\} = E \left[\begin{array}{cc} \bar{\mathbf{Q}}_x^* & \mathbf{0} \\ \mathbf{0} & \bar{\mathbf{Q}}_y^* \\ \bar{\mathbf{Q}}_y^* & \bar{\mathbf{Q}}_x^* \end{array} \right] \left\{ \begin{array}{c} \mathbf{a}_x^* \\ \mathbf{a}_y^* \end{array} \right\} = \mathbf{V} \mathbf{a}^* \end{aligned} \tag{3.44}$$

sendo $\bar{\mathbf{Q}}^*$ uma matriz similar a $\mathbf{P}$ da expressão (3.20).

O gradiente transposto de $\mathbf{V}$ é dado por,

$$\nabla^T \mathbf{V} = \left[\begin{array}{ccc} \partial/\partial x & 0 & \partial/\partial y \\ 0 & \partial/\partial y & \partial/\partial x \end{array} \right] E \left[\begin{array}{cc} \bar{\mathbf{Q}}_x^* & \mathbf{0} \\ \mathbf{0} & \bar{\mathbf{Q}}_y^* \\ \bar{\mathbf{Q}}_y^* & \bar{\mathbf{Q}}_x^* \end{array} \right] = E \left[\begin{array}{cc} \bar{\mathbf{Q}}_{xx}^* + \bar{\mathbf{Q}}_{yy}^* & \bar{\mathbf{Q}}_{xy}^* \\ \bar{\mathbf{Q}}_{yx}^* & \bar{\mathbf{Q}}_{xx}^* + \bar{\mathbf{Q}}_{yy}^* \end{array} \right] \tag{3.45}$$

Para se obter os coeficientes $\mathbf{a}$, substituem-se os resíduos (3.41) em (3.42) e utiliza-se a condição de mínimo deste funcional. Para simplificar, considera-se cada termo F_i $(i = 1, \ldots, 5)$ em separado. Portanto,

tensões :

$$\begin{aligned} F_1 &= E^{-1} \sum_s [w_s(\mathbf{T}^* - \mathbf{T}_h^s)]^T [w_s(\mathbf{T}^* - \mathbf{T}_h^s)] \\ &= E^{-1} \sum_s w_s^2 [\mathbf{a}^T \mathbf{V}^T \mathbf{V} \mathbf{a} - 2\mathbf{a}^T \mathbf{V}^T \mathbf{T}_h^s + (\mathbf{T}_h^s)^T \mathbf{T}_h^s] \\ \frac{\partial F_1(\mathbf{a})}{\partial \mathbf{a}} &= 2E^{-1} \sum_s w_s^2 (\mathbf{V}^T \mathbf{V} \mathbf{a} - \mathbf{V}^T \mathbf{T}_h^s) \end{aligned} \tag{3.46}$$

equilíbrio :

$$\begin{aligned} F_2 &= E^{-1} \beta_p \int (\nabla^T \mathbf{T}^* + \mathbf{f}_b)^T (\nabla^T \mathbf{T}^* + \mathbf{f}_b) \, d\Omega_p \\ &= E^{-1} \beta_p \int [\mathbf{a}^T (\nabla^T \mathbf{V})^T \nabla^T \mathbf{V} \mathbf{a} + 2\mathbf{a}^T (\nabla^T \mathbf{V})^T \mathbf{f}_b + \mathbf{f}_b^T \mathbf{f}_b] \, d\Omega_p \\ \frac{\partial F_2(\mathbf{a})}{\partial \mathbf{a}} &= 2E^{-1} \beta_p \int [(\nabla^T \mathbf{V})^T \nabla^T \mathbf{V} \mathbf{a} + (\nabla^T \mathbf{V})^T \mathbf{f}_b] \, d\Omega_p \end{aligned} \tag{3.47}$$

deslocamentos :

$$\begin{aligned} F_3 &= E\sum_r [w_r(\mathbf{u}^* - \mathbf{u}_h)]^T[w_r(\mathbf{u}^* - \mathbf{u}_h)] \\ &= E\sum_r w_r^2(\mathbf{a}^T\bar{\mathbf{Q}}^T\bar{\mathbf{Q}}\mathbf{a} - 2\mathbf{a}^T\bar{\mathbf{Q}}^T\mathbf{u}_h + \mathbf{u}_h^T\mathbf{u}_h) \\ \frac{\partial F_3(\mathbf{a})}{\partial \mathbf{a}} &= 2E\sum_r w_r^2(\bar{\mathbf{Q}}^T\bar{\mathbf{Q}}\mathbf{a} - \bar{\mathbf{Q}}^T\mathbf{u}_h) \end{aligned} \tag{3.48}$$

tensões prescritas :

$$\begin{aligned} F_4 &= E^{-1}\sum_b [w_b(\nabla_n^T\mathbf{u}^* + \mathbf{t}_b)]^T[w_b(\nabla_n^T\mathbf{u}^* + \mathbf{t}_b)] \\ &= E^{-1}\sum_b w_b^2[\mathbf{a}^T(\nabla_n^T\bar{\mathbf{Q}})^T\nabla_n^T\bar{\mathbf{Q}}\mathbf{a} - 2\mathbf{a}^T(\nabla_n^T\bar{\mathbf{Q}})^T\mathbf{t}_b + \mathbf{t}_b^T\mathbf{t}_b] \\ \frac{\partial F_4(\mathbf{a})}{\partial \mathbf{a}} &= 2E^{-1}\sum_b w_b^2[(\nabla_n^T\bar{\mathbf{Q}})^T\nabla_n^T\bar{\mathbf{Q}}\mathbf{a} + (\nabla_n^T\bar{\mathbf{Q}})^T\mathbf{t}_b] \end{aligned} \tag{3.49}$$

deslocamentos prescritos :

$$\begin{aligned} F_5 &= E\sum_u [w_u(\mathbf{u}^* - \mathbf{u}_b)]^T[w_r(\mathbf{u}^* - \mathbf{u}_b)] \\ &= E\sum_u w_u^2(\mathbf{a}^T\bar{\mathbf{Q}}^T\bar{\mathbf{Q}}\mathbf{a} - 2\mathbf{a}^T\bar{\mathbf{Q}}^T\mathbf{u}_b + \mathbf{u}_b^T\mathbf{u}_b) \\ \frac{\partial F_5(\mathbf{a})}{\partial \mathbf{a}} &= 2E\sum_u w_u^2(\bar{\mathbf{Q}}^T\bar{\mathbf{Q}}\mathbf{a} - \bar{\mathbf{Q}}^T\mathbf{u}_b) \end{aligned} \tag{3.50}$$

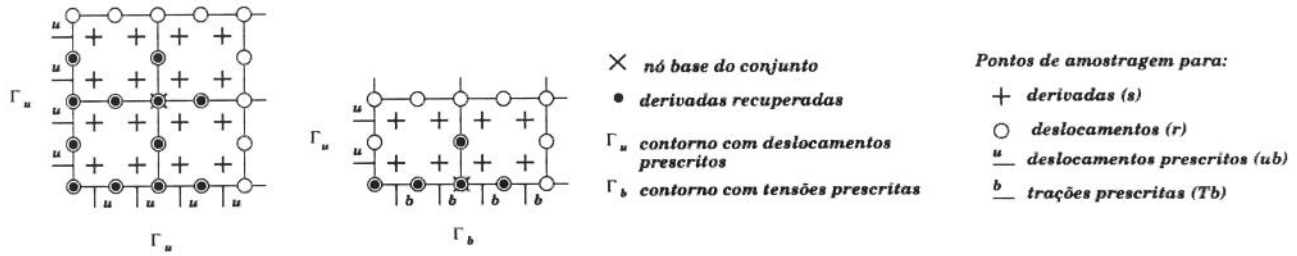


Figura 3.4: Conjuntos de elementos quadráticos com condições de contorno a) essenciais; b) gerais [106].

Somando as expressões (3.46) a (3.50) e simplicando, obtém-se um sistema análogo a (3.25) com $\mathbf{A}_p$ e $\mathbf{b}_p$ dados por,

$$\mathbf{A}_p = \sum_s w_s^2\mathbf{V}^T\mathbf{V} + \beta_p\int(\nabla^T\mathbf{V})^T\nabla^T\mathbf{V}\,d\Omega_p + \sum_r E^2w_r^2\bar{\mathbf{Q}}^T\bar{\mathbf{Q}} + \sum_b w_b^2(\nabla_n^T\bar{\mathbf{Q}})^T\nabla_n^T\bar{\mathbf{Q}} + \sum_u E^2w_u^2\bar{\mathbf{Q}}^T\bar{\mathbf{Q}} \tag{3.51}$$

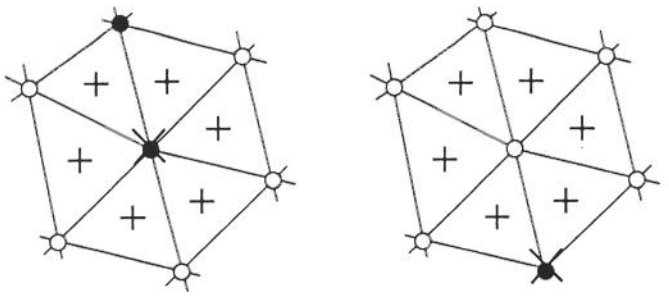


Figura 3.5: Conjuntos de elementos para condições de contorno a) essenciais; b) naturais [106].

$$\mathbf{b}_p = \sum_s w_s^2 \mathbf{V}^T \mathbf{T}_h^s - \beta_p \int (\nabla^T \mathbf{V})^T \mathbf{f}_b \, d\Omega_p + \sum_r E^2 w_r^2 \bar{\mathbf{Q}}^T \mathbf{u}_h + \sum_b w_b^2 (\nabla_n^T \bar{\mathbf{Q}})^T \mathbf{t}_b + \sum_u E^2 w_u^2 \bar{\mathbf{Q}}^T \mathbf{u}_b \quad (3.52)$$

Observa-se que os dois últimos termos em $\mathbf{A}_p$ e $\mathbf{b}_p$ são determinados em pontos do contorno. Já o para o terceiro termo, utilizam-se as coordenadas nodais, indicadas pelo índice r, as quais usualmente constituem-se em pontos superconvergentes para os deslocamentos. Os termos w_s, w_r, w_u e w_b são, respectivamente, as funções de ponderação para as tensões, deslocamentos nodais, deslocamentos e trações prescritos. Empregam-se $w_s = w_r = 1$ e valores mais altos para w_{u_b} e w_{T_b} visando garantir que a solução recuperada satisfaça as condições de contorno [106]. A Figura 3.4 ilustra os pontos de amostragem considerando condições de contorno essenciais e gerais.

Para recuperar os valores no contorno, utilizam-se conjuntos de elementos no contorno. No entanto, para casos onde se tem apenas condições essenciais a configuração ilustrada na Figura 3.5a) é mais adequada, pois tem-se mais informação do campo de tensões nos pontos interiores. Já para as condições naturais, aplicam-se conjuntos de contorno como mostrado na Figura 3.5b).

3.3 Estudo de Caso

A Figura 3.6a) ilustra um cilindro, com raios r_i e r_e, submetido a pressões interna (P_i) e externa (P_e), tratado como um problema de estado plano de deformação. As componentes de tensão radial (T_r) e circunferencial (T_θ) são dadas respectivamente por [83],

$$T_r = C_1 - \frac{C_2}{r^2} \quad (3.53)$$

$$T_\theta = C_1 + \frac{C_2}{r^2} \quad (3.54)$$

sendo,

$$C_1 = \frac{P_i r_i^2 - P_e r_e^2}{r_e^2 - r_i^2} \qquad C_2 = \frac{(P_i - P_e) r_i^2 r_e^2}{r_e^2 - r_i^2} \quad (3.55)$$

Na análise deste problema foram geradas 6 malhas com elementos triangulares lineares e quadráticos, diminuindo progressivamente o tamanho do elemento. A Tabela 3.1 apresenta os número de nós, elementos e equações para cada uma desta malhas. A Figura 3.7 ilustra as malhas geradas com elementos de 3 nós.

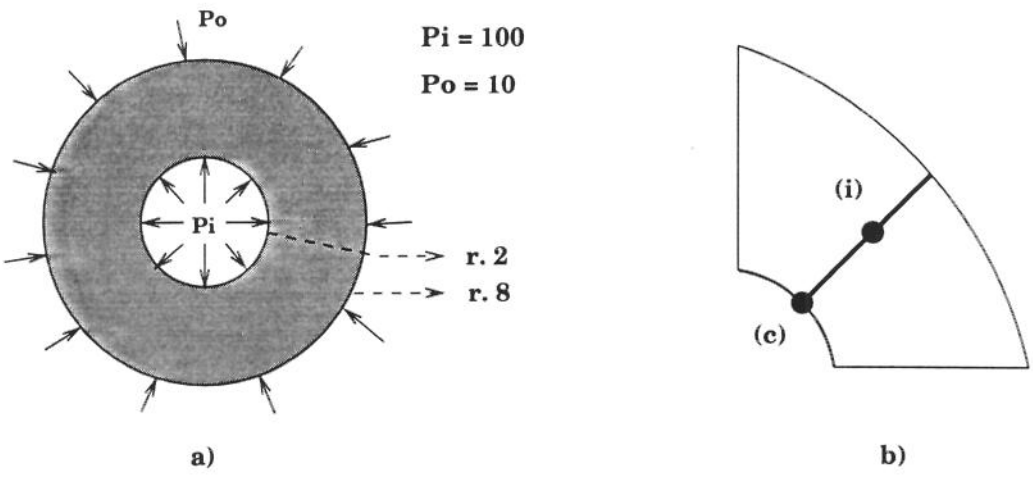


Figura 3.6: Cilindro com pressões interna e externa modelado como estado plano de deformação.

Malha	Linear			Quadrático		
	Nós	Elementos	Equações	Nós	Elementos	Equações
1	65	100	106	229	100	432
2	101	165	185	366	165	700
3	161	274	300	595	274	1148
4	357	642	682	1355	642	2648
5	1375	2610	2688	5359	2610	10596
6	5500	10720	10878	21719	10720	43196

Tabela 3.1: Características das malhas com elementos linear e quadrático.

Para verificar a taxa de convergência das tensões nodais calculadas pelo MEF e recuperadas pelos algoritmos PA e EP, consideraram-se os pontos interno (i) e de contorno (c) ilustrados na Figura 3.6b). O ponto (i) possui coordenadas $(r, \theta) = (5, 45^o)$ e os valores analíticos das tensões são $\sigma_r = -19.36$ e $\sigma_\theta = 11.36$. Já o ponto (c) tem coordenadas $(r, \theta) = (2, 45^o)$ e as tensões calculadas pelas expressões (3.53) e (3.54) são $\sigma_r = -100$ e $\sigma_\theta = 92$.

Toma-se como erro absoluto, o módulo da diferença entre a componente de tensão analítica (T^a) e o valor recuperado (T^r) pelas técnicas indicadas, ou seja,

$$e_i^T = |T_i^a - T_i^r| \tag{3.56}$$

sendo o superescrito r substituído por MEF, PA ou EP; o indíce i indica a componente de tensão, neste caso $i = r, \theta$.

O erro absoluto foi calculado para os nós interno e de contorno indicados na Figura 3.6b), tomando-se cada uma das malhas geradas com valores determinados pelo MEF e pelas técnicas de recuperação PA e EP. A partir daí, ajustaram-se retas do tipo $y = Cx^\alpha$ em escala logarítmica. As Figuras 3.8 e 3.9 ilustram os gráficos obtidos para as tensões radial e circunferencial dos nós interno e de contorno das malhas lineares. Os mesmos resultados estão apresentados nas Figuras 3.10 a 3.11 para as malhas quadráticas. A Tabela 3.2 contém os coeficientes angulares α das retas das Figuras 3.8 a 3.11.

Considerou-se, então, a convergência na norma de energia, estando o erro dado por,

$$||e||_a^2 = \int_\Omega \mathbf{e}^T \mathbf{D}^{-1} \mathbf{e}\, d\Omega = \sum_{i=1}^{N^{els}} \int_{\Omega_e} \mathbf{e}_i^T \mathbf{D}^{-1} \mathbf{e}_i\, d\Omega_e \tag{3.57}$$

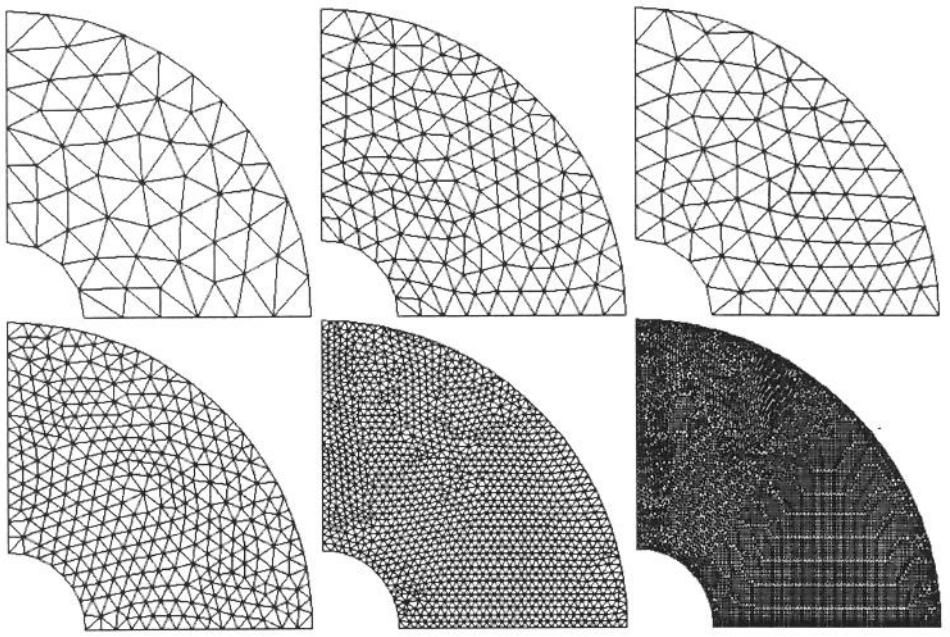


Figura 3.7: Malhas com elementos lineares para um quarto do cilindro.

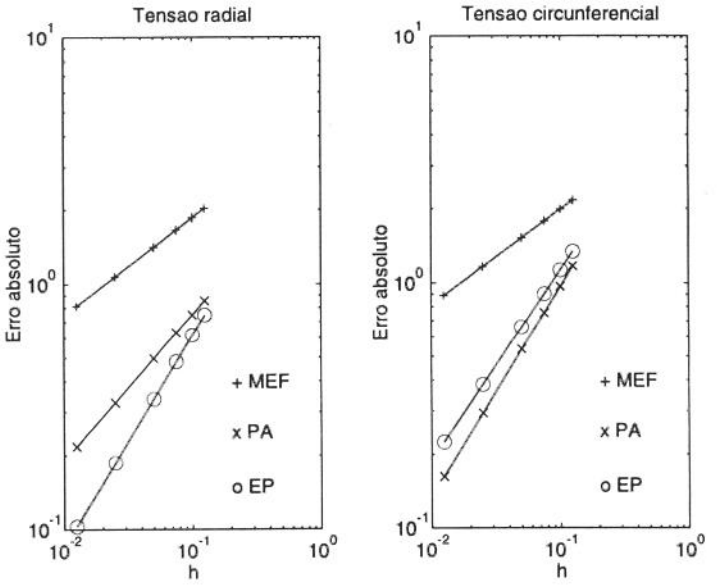


Figura 3.8: Comportamento das componentes de tensão radial e circunferencial para o nó interno determinados por MEF, PA e EP nas malhas lineares.

Nó interno				
Técnica	Linear		Quadrático	
	T_r	T_θ	T_r	T_θ
MEF	0,92	0,89	2,15	1,83
PA	1,37	1,98	3,00	1,55
EP	1,98	1,79	2,22	2,08

Nó de contorno				
Técnica	Linear		Quadrático	
	T_r	T_θ	T_r	T_θ
MEF	0,92	0,79	1,53	1,65
PA	0,97	1,08	2,00	1,71
EP	2,22	1,61	2,34	2,28

Tabela 3.2: Coeficientes angulares das retas de erro absoluto.

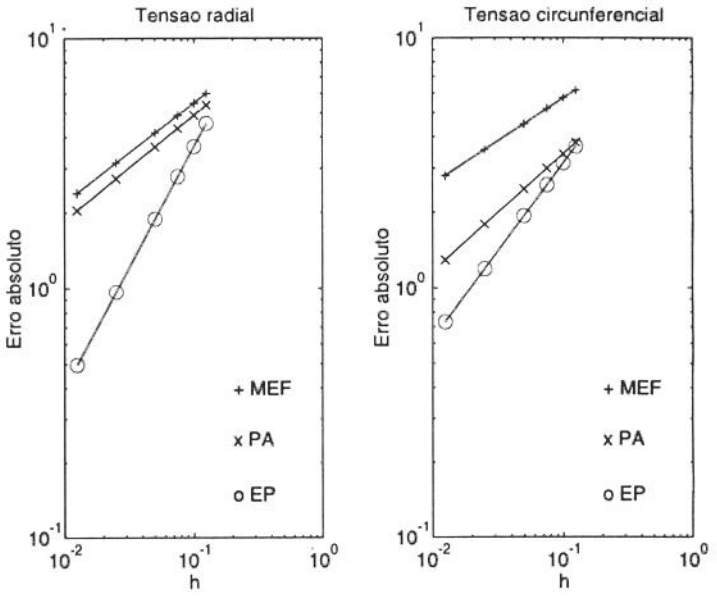


Figura 3.9: Comportamento das componentes de tensão radial e circunferencial para o nó de contorno determinados por MEF, PA e EP nas malhas lineares.

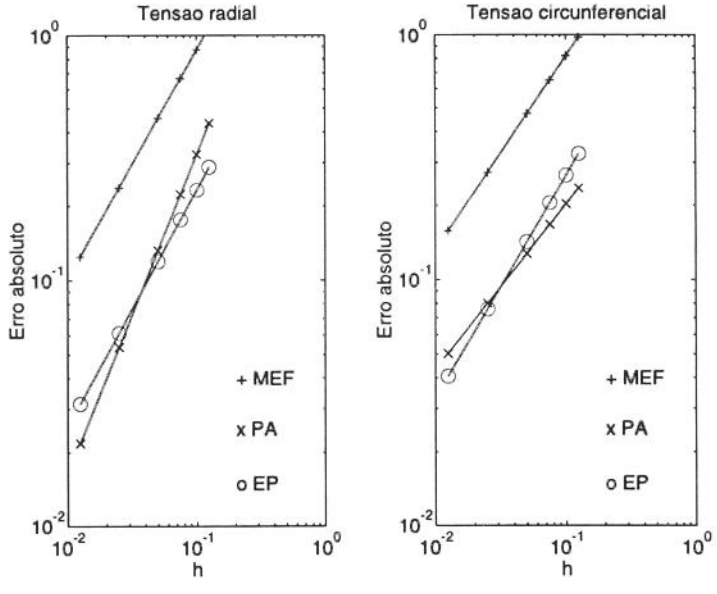


Figura 3.10: Comportamento das componentes de tensão radial e circunferencial para o nó interno determinados por MEF, PA e EP nas malhas quadráticas.

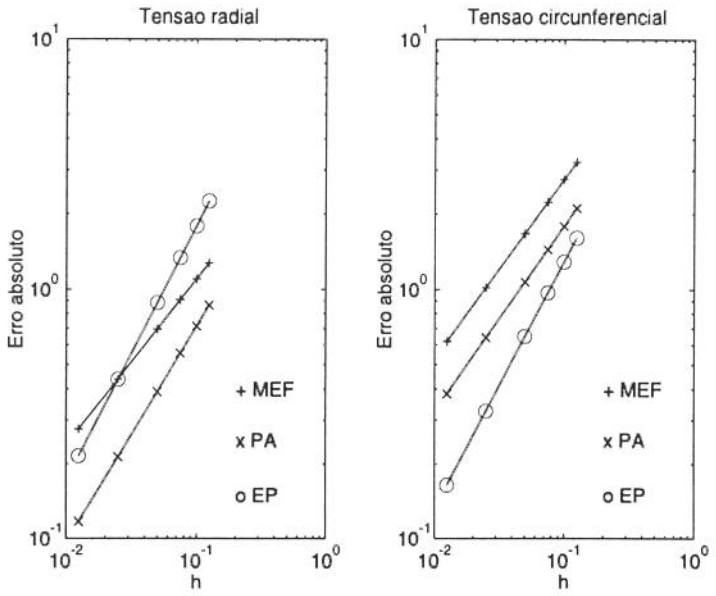


Figura 3.11: Comportamento das componentes de tensão radial e circunferencial para o nó de contorno determinados por MEF, PA e EP nas malhas quadráticas.

Técnica	Linear	Quadrático
MEF	1,02	1,87
PA	1,50	1,90
EP	1,93	2,66

Tabela 3.3: Coeficientes angulares das retas para a norma de energia do erro.

onde Ω_e é o domínio do elemento e o erro $\mathbf{e}_i$ é dado pela diferença entre as soluções teórica e aquelas determinadas por MEF, PA e EP.

Calculou-se a norma de energia do erro para cada uma das malhas segundo as técnicas consideradas. A Figura 3.12 ilustra as retas obtidas para as malhas com elementos lineares e quadráticos. A Tabela 3.3 apresenta os coeficientes angulares das retas da Figura 3.12.

3.4 Discussão dos Resultados

A partir dos resultados obtidos observa-se:

- para o elemento linear, as taxas de convergência dos erros obtidas pelo MEF para os nós interno e de contorno considerados estão inferiores ao valor esperado igual a 1. No entanto, a convergência do erro na norma de energia é mais significativa, sendo a taxa obtida dentro da ordem $\mathcal{O}(h)$ prevista por estimativas *a-priori* do erro no MEF. Um comportamento semelhante foi observado com o elemento quadrático, sendo os resultados para o nó interno superiores ao de contorno.

- para a técnica PA, os valores obtidos para o nó de contorno apresentam uma ordem de convergência padrão. Para o nó interno, as componentes T_θ do caso linear e T_r do elemento quadrático comportam-se com $\mathcal{O}(h^{p+1})$ sendo superconvergentes, onde p é a ordem das funções de interpolação. Por sua vez, a tensão T_r do elemento linear possui um comportamento acima

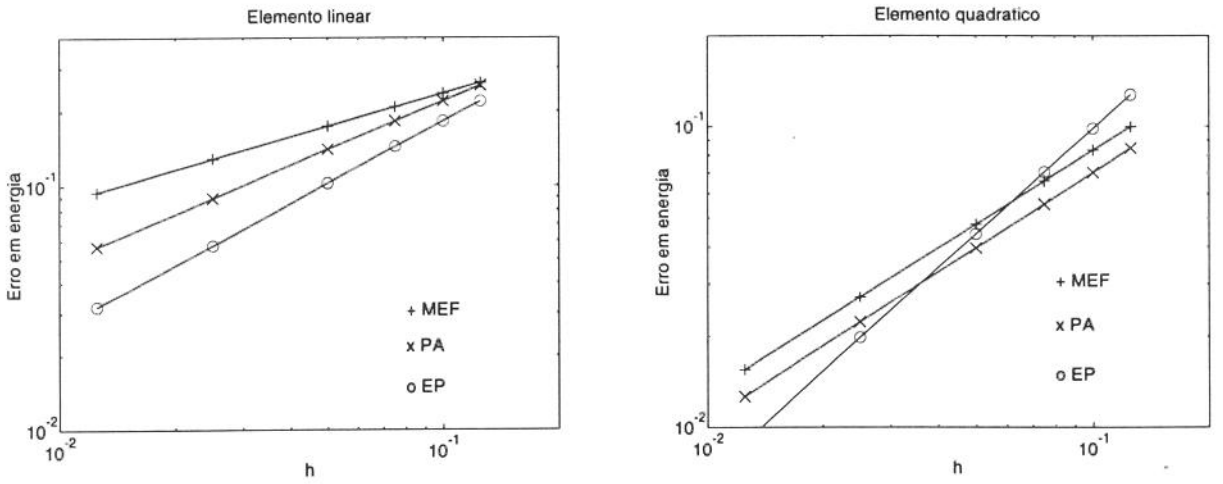


Figura 3.12: Norma em energia do erro para as malhas linear e quadrática.

do esperado, enquanto T_θ do caso quadrático está abaixo do previsto pela teoria matemática do MEF.

Finalmente no erro em energia, o resultado para o elemento linear pode ser considerado superconvergente, enquanto que no quadrático tem-se uma ordem padrão $\mathcal{O}(h^2)$. Este mesmo comportamento foi observado em [3], sendo colocado que o estimador ZZ converge mais rápido para funções de base lineares quando comparado com o caso quadrático.

- para o recuperador EP, os valores para os nós interno e de contorno do elemento linear são superconvergentes, enquanto àqueles do elemento quadrático estão dentro da ordem prevista $\mathcal{O}(h^2)$. O mesmo comportamento é obtido na norma de energia.

  Em [117], alguns resultados para elementos quadráticos apresentaram ultraconvergência com ordem $\mathcal{O}(h^4)$ para os nós internos em malhas regulares. No entanto, para a norma de energia, a ordem indicada é de $\mathcal{O}(h^{p+\alpha})$ com $\alpha \geq 0,5$ para elementos triangulares. Desta forma, os resultados obtidos são semelhantes àqueles dados em [105, 117].

- os resultados obtidos em [105] para o recuperador EPE são semelhantes ao caso EP para os nós internos e ligeiramente superiores para o contorno. Observa-se que a ordem do sistema de equações do conjunto de elementos do procedimento EPE é maior do que no recuperador EP, devido ao acoplamento das equações pelo termo de resíduo da equação de equilíbrio. Por sua vez, o recuperador EPEC é bem mais complexo, devendo ser empregado apenas quando for necessário obter valores mais precisos ao longo do contorno.

Capítulo 4

MÉTODOS MULTIGRID

Neste capítulo, abordam-se os métodos multigrid e aplicações dos mesmos a problemas elásticos. Inicialmente, apresenta-se um estudo espectral do comportamento dos métodos iterativos estacionários, procurando identificar as suas limitações e como as mesmas podem ser contornadas através dos elementos principais das técnicas multigrid discutidas na seção 4.2.

No caso de malhas não-aninhadas, os operadores de interpolação e restrição são fundamentais, demandando estruturas de dados robustas e eficientes, como será visto na seção 4.3. Posteriormente, consideram-se estratégias multigrid padrão e critérios adaptáveis para o refinamento das malhas, além de expressões para o cálculo do número de operações e espaço de memória. Uma forma variacional geral para os operadores de transferência e aspectos de convergência estão apresentados, respectivamente, nas seções 4.5 e 4.6

Finalmente, aplicam-se os algoritmos discutidos na análise de problemas elásticos bi e tridimensionais, comparando-se os resultados com os métodos de Gauss e iterativo GCGS.

4.1 Comportamento dos Métodos Iterativos Básicos

Visando estudar o comportamento dos métodos iterativos básicos, toma-se o seguinte PVC unidimensional descrito pela equação diferencial ordinária de segunda ordem com condições de contorno homogêneas,

$$\begin{aligned} &-u''(x) + \sigma u(x) = b(x) \quad 0 < x < 1 \quad \sigma \geq 0 \\ &u(0) = u(1) = 0 \end{aligned} \tag{4.1}$$

Empregando o método de diferenças finitas ao PVC anterior, o domínio do problema é particionado em N subintervalos pelas coordenadas $x_j = jh$, definindo a malha Ω^h ilustrada na Figura 4.1, com o tamanho do elemento $h = \frac{1}{N}$. Em cada um dos $N-1$ pontos interiores, introduz-se uma aproximação de segunda ordem dada por,

$$\begin{aligned} &\frac{-u_{j-1} + 2u_j - u_{j+1}}{h^2} + \sigma u_j = b(x_j) \qquad 1 \leq j \leq N-1 \\ &u_0 = u_N = 0 \end{aligned} \tag{4.2}$$

onde u_j é uma aproximação para solução exata $u(x_j)$. As expressões em (4.2) podem ser dispostas num sistema da forma (1.30) [31].

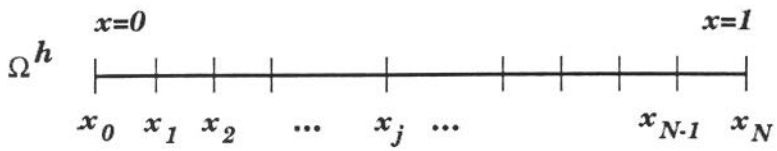


Figura 4.1: Malha unidimensional Ω^h para o intervalo $[0, 1]$ com $h = \frac{1}{N}$ [31].

No entanto, para analisar os métodos estacionários, basta considerar o sistema homogêneo $\mathbf{Au} = \mathbf{0}$ tomando uma solução inicial arbitrária $\mathbf{v}$. Neste caso, a solução exata $\mathbf{u} = \mathbf{0}$ é conhecida e o erro algébrico é simplesmente $\mathbf{e} = -\mathbf{v}$. Assim em (4.2), tomando-se $b(x_j) = 0$ e $\sigma = 0$, determinam-se as seguintes expressões,

$$\begin{aligned} &-u_{j-1} + 2u_j - u_{j+1} = 0 \quad 1 \leq j \leq N-1 \\ &u_0 = u_N = 0 \end{aligned} \tag{4.3}$$

Como solução inicial $\mathbf{v}$ para o sistema homogêneo com equações dadas por (4.3), empregam-se os vetores ou modos de Fourier,

$$v_j = \sin\left(\frac{jk\pi}{N}\right) \quad 0 \leq j \leq N \quad 1 \leq k \leq N-1 \tag{4.4}$$

O índice j denota a componente do vetor $\mathbf{v}_k$, onde k é o número de onda representando a quantidade de meio-senos presentes em $\mathbf{v}_k$, como ilustrado na Figura 4.2 para $k = 1, 3, 6$. Observa-se que valores pequenos de k correspondem a ondas longas e suaves, enquanto valores altos estão relacionados a modos altamente oscilatórios.

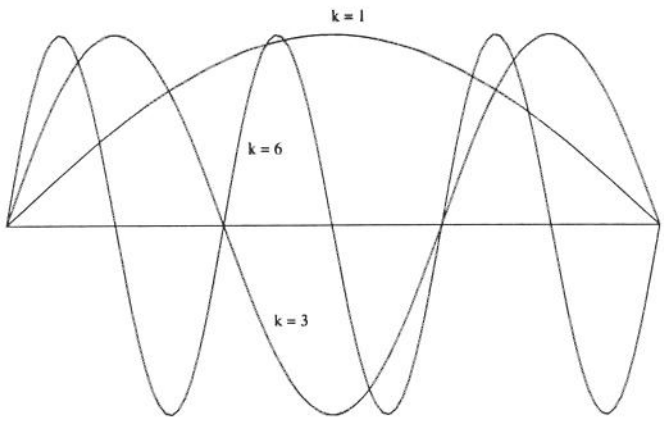


Figura 4.2: Modos de Fourier para números de onda $k = 1, 3, 6$ [31].

Aplica-se ao problema (4.3), o método de Jacobi ponderado, análogo ao algoritmo SOR, com coeficiente de relaxação $\omega = 2/3$ numa malha com $N = 64$ [31]. Empregam-se as aproximações iniciais $\mathbf{v}_1$, $\mathbf{v}_3$ e $\mathbf{v}_6$ executando 100 iterações. Em cada passo, calcula-se a norma infinita do erro $||\mathbf{e}||_\infty$. A Figura 4.3a) ilustra o logaritmo do erro em função do número de iterações. Verifica-se um decrescimento linear do erro, sendo a taxa mais pronunciada para números de onda maiores.

Em geral, a aproximação inicial contém componentes de vários vetores de Fourier (4.4). Assim na Figura 4.3b), tem-se a norma do erro $||e||_\infty$ para uma solução inicial dada pela soma dos modos de baixa ($k = 1$), média ($k = 6$) e alta ($k = 32$) frequências. Neste caso, verifica-se uma queda acentuada do erro nas 5 primeiras relaxações e uma estagnação na taxa de decrescimento nas demais iterações.

Os modos de Fourier podem ser classificados como *modos suaves ou de baixa frequência* para $1 \leq k < N/2$ ou como *modos oscilatórios ou de alta frequência* para $N/2 \leq k \leq N-1$. Assim, na Figura 4.3b), o decrescimento inicial corresponde a rápida eliminação dos modos oscilatórios, enquanto os modos suaves implicam num decrescimento menor da norma do erro.

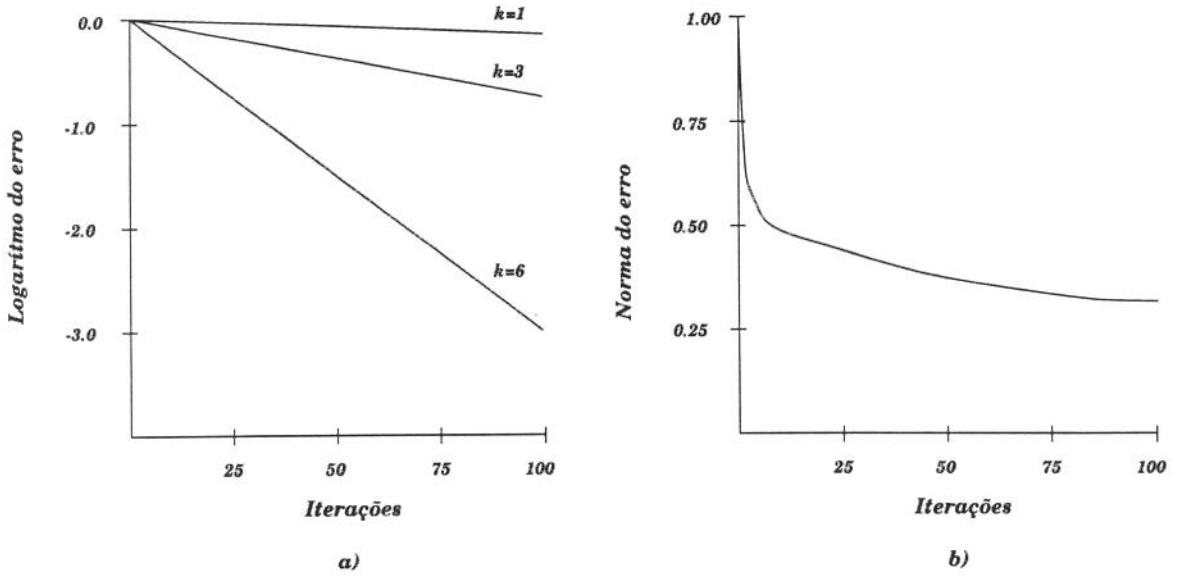


Figura 4.3: Norma infinita do erro para cada iteração com soluções iniciais: a) $\mathbf{v} = \mathbf{v}_1$, $\mathbf{v} = \mathbf{v}_3$, $\mathbf{v} = \mathbf{v}_6$; b) $\mathbf{v} = \frac{1}{3}(\mathbf{v}_1 + \mathbf{v}_6 + \mathbf{v}_{32})$ [31].

Esta é uma propriedade típica dos métodos estacionários, ou seja, observa-se uma rápida convergência quando o erro possui componentes de alta frequência. Da mesma maneira, tem-se uma baixa performance na eliminação das componentes suaves degradando a performance destes métodos. Denomina-se esta característica como *propriedade de suavização* [31].

As técnicas multigrid procuram evitar esta limitação dos algoritmos estacionários, empregando os elementos discutidos na próxima seção. Como será visto, a função dos métodos iterativos é eliminar em cada malha as componentes oscilatórias do erro através de algumas poucas iterações. Desta forma, aplicam-se esquemas de relaxação simples como Jacobi e Gauss-Seidel, pois não se procura alcançar a convergência em cada malha, mas sim a solução do problema proposto utilizando várias malhas.

4.2 Elementos Principais dos Métodos Multigrid

4.2.1 Iterações aninhadas

Como mencionado anteriormente, uma forma de aumentar a performance de um esquema de relaxação é empregar uma melhor aproximação inicial, obtida por exemplo através de iterações numa malha grossa. Como esta malha possui um número menor de variáveis, tem-se um menor custo computacional em relação a uma iteração realizada na malha mais fina.

Dada uma malha Ω^h com $N = 12$, ilustrada na Figura 4.4, verifica-se que o modo suave $k = 4$ torna-se oscilatório ao se projetá-lo numa malha Ω^{2h} com $N = 6$. Portanto, para $1 \leq k < N/2$, o

k-ésimo modo em Ω^h, transforma-se no k-ésimo modo em Ω^{2h}, ou seja, ao se passar da malha fina para a grossa, um modo suave transforma-se num modo oscilatório. Para $k = N/2$ em Ω^h corresponde o modo nulo em Ω^{2h}. Finalmente, os modos oscilatórios em Ω^h aparecem relativamente suaves em Ω^{2h}.

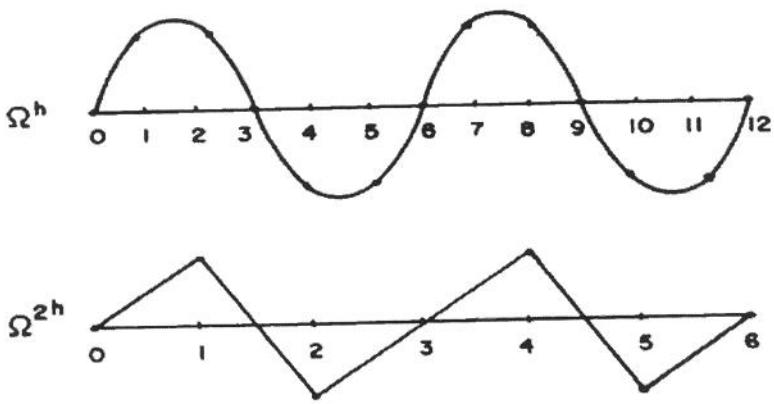


Figura 4.4: Modo suave $k = 4$ em Ω^h com $N = 12$ transforma-se no modo oscilatório $k = 4$ em Ω^{2h} com $N = 6$ [31].

Portanto, ao se verificar uma estagnação no algoritmo de relaxação na malha fina, indicando que as componentes oscilatórias foram eliminadas, restando apenas os modos suaves, deve-se passar para a malha grossa. Desta forma, os modos suaves projetados da malha fina tornam-se oscilatórios, sendo eficientemente eliminados através de relaxações.

Assim, a seguinte estratégia, denominada *iterações aninhadas*, permite obter uma melhor aproximação inicial para a malha mais fina [31]:

Iterar em $\mathbf{Au} = \mathbf{b}$ na malha mais grossa
$\vdots$
Iterar em $\mathbf{Au} = \mathbf{b}$ na malha Ω^{4h} para obter uma aproximação inicial para Ω^{2h}
Iterar em $\mathbf{Au} = \mathbf{b}$ na malha Ω^{2h} para obter uma aproximação inicial para Ω^h
Iterar em $\mathbf{Au} = \mathbf{b}$ na malha mais fina para obter uma aproximação final para a solução $\mathbf{u}$.

Esta estratégia não garante que ao final, a solução em Ω^h não contenha componentes suaves. A possibilidade de empregar a correção do erro proveniente das malhas grossas evita esta limitação.

4.2.2 Correção de malha grossa

A expressão (1.35) estabelece uma relação entre o erro $\mathbf{e}$ e o resíduo $\mathbf{r}$, permitindo iterar diretamente sobre o erro $\mathbf{e}$. Além disso, verifica-se que relaxar na equação original $\mathbf{Au} = \mathbf{b}$ com uma aproximação $\mathbf{v}$ é equivalente a relaxar na equação do resíduo $\mathbf{Ae} = \mathbf{r}$ com aproximação inicial $\mathbf{e} = \mathbf{0}$.

A partir deste fato, propõe-se incorporar a equação de resíduo para relaxar sobre o erro, permitindo definir o procedimento de *correção de malha grossa* da seguinte maneira:

Iterar em $\mathbf{Au} = \mathbf{b}$ na malha Ω^h para obter uma aproximação $\mathbf{v}^h$
Calcular $\mathbf{r} = \mathbf{b} - \mathbf{Av}^h$
Iterar na equação $\mathbf{Ae} = \mathbf{r}$ em Ω^{2h} determinando uma aproximação $\mathbf{e}^{2h}$
Corrigir a aproximação obtida em Ω^h com o erro estimado em Ω^{2h} : $\mathbf{v}^h \leftarrow \mathbf{v}^h + \mathbf{e}^{2h}$.

Portanto, após relaxar na malha fina até que a convergência se deteriore, passa-se a iterar na equação de resíduo numa malha mais grossa, obtendo-se uma aproximação para o erro, a qual corrige a solução na malha fina.

4.2.3 Operadores de transferência entre malhas

Os elementos principais discutidos nas seções anteriores revelam a necessidade de operadores para transferir informações entre as malhas. Inicialmente, assume-se que a malha grossa possui a metade do número de incógnitas da malha fina ou ainda que o tamanho dos elementos grossos é o dobro daqueles da malha fina. Assim, a malha grossa está contida na fina, ou seja, estão aninhadas (*nested meshes*).

O primeiro operador transfere funções da malha grossa Ω^{2h} para a malha fina Ω^h, sendo denominado *operador de interpolação ou prolongamento* e denotado por I_{2h}^h. É empregado para se passar o erro $\mathbf{e}^{2h}$ ou a aproximação inicial $\mathbf{v}^{2h}$ da malha grossa para a fina. Logo,

$$\begin{aligned} I_{2h}^h \; &: \; \Omega^{2h} \rightarrow \Omega^h \\ &\quad \mathbf{v}^{2h} \rightarrow \mathbf{v}^h = I_{2h}^h \mathbf{v}^{2h} \end{aligned} \tag{4.5}$$

A Figura 4.5 mostra o esquema de interpolação linear. Para o caso unidimensional tem-se,

$$\begin{cases} v_{2j}^h = v_j^{2h} & 0 \leq j \leq \frac{N}{2} - 1 \\ v_{2j+1}^h = \frac{1}{2}(v_j^{2h} + v_{j+1}^{2h}) \end{cases} \tag{4.6}$$

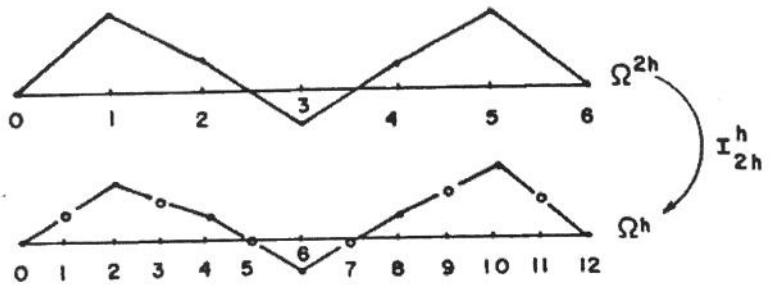


Figura 4.5: Interpolação linear de uma função definida na malha grossa Ω^{2h} para a malha fina Ω^h [31].

O segundo operador, I_h^{2h}, transforma funções da malha fina Ω^h para a grossa Ω^{2h}, como ao se projetar o resíduo $\mathbf{r}^h$ em Ω^{2h}, sendo conhecido como *operador de restrição* e denotado por:

$$\begin{aligned} I_h^{2h} \; &: \; \Omega^h \rightarrow \Omega^{2h} \\ &\quad \mathbf{v}^h \rightarrow \mathbf{v}^{2h} = I_h^{2h} \mathbf{v}^h \end{aligned} \tag{4.7}$$

O tipo de restrição mais simples em malhas aninhadas é o operador injetivo,

$$v_j^{2h} = v_{2j}^h \tag{4.8}$$

ou seja, os pontos da malha grossa assumem os valores correspondentes da malha fina.

Uma outra forma de restrição emprega uma ponderação, como ilustrado na Figura 4.6, sendo dado por,

$$v_j^{2h} = \frac{1}{4}(v_{2j-1}^h + 2v_{2j}^h + v_{2j+1}^h) \quad 1 \leq j \leq \frac{N}{2} - 1 \tag{4.9}$$

No entanto, neste trabalho, considera-se que uma malha grossa não está necessariamente contida na malha fina seguinte, ou seja, não se tem uma relação fixa de 2^{-d} entre os tamanhos dos elementos, onde d é a dimensão do problema. Esta condição define o que é denominado na literatura como malhas não-aninhadas (*non-nested meshes*).

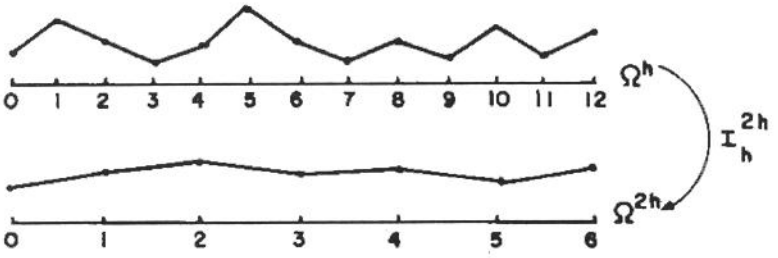


Figura 4.6: Restrição ponderada de uma função da malha fina Ω^h para a malha grossa Ω^{2h} [31].

4.2.4 Extensão dos operadores para malhas não-aninhadas

Supondo que um problema bidimensional seja resolvido pelo MEF empregando triângulos lineares, definem-se os operadores de restrição e pronlogamento utilizando as funções de forma do elemento, dadas em coordenadas de área $A_i\,(i = 1, 2, 3)$. A Figura 4.7 apresenta duas malhas triangulares permitindo observar a forma dos operadores. De maneira geral, basta identificar para cada nó fino I, o triângulo grosso T_g contendo I. A partir daí, determinam-se as coordenadas de área do nó I, tomando-se o elemento T_g.

Inicialmente, deseja-se projetar o resíduo da malha fina para a grossa. Considerando a notação utilizada na Figura 4.7, o operador de restrição determina o resíduo em cada nó do triângulo T_g, a partir da contribuição dos resíduos dos nós finos presentes em T_g. Logo,

$$\mathbf{r}^i = I_h^{2h}\mathbf{r}^h = \sum_{l=1}^{N_f} A_i^I \mathbf{r}_l^I \quad i = 1, 2, 3 \tag{4.10}$$

onde N_f é o número de nós finos I, com respectivo resíduo $\mathbf{r}^I$ e coordenadas A^I, contidos no triângulo da malha grossa T_g.

Analogamente, para mapear o erro da malha grossa para a fina basta utilizar uma interpolação via as funções de forma de T_g. Logo,

$$\mathbf{e}^I = I_h^{2h}\mathbf{e}^{2h} = \sum_{l=1}^{3} A_l^I \mathbf{e}^l \tag{4.11}$$

Observa-se que a extensão destes operadores para problemas tridimensionais empregando tetraedros lineares é análoga, devendo-se usar as coordenadas de volume $V_i\,(i =, 1, \ldots, 4)$.

A possibilidade de utilizar malhas não-estruturadas permite uma maior flexibilidade na aplicação dos algoritmos multigrid, principalmente para domínios complexos onde a geração da malha torna-se trabalhosa ao se impor uma fator fixo de 2^{-d} no tamanho dos elementos entre malhas consecutivas. Porém, no caso de malhas não-estruturadas este fator deve estar próximo de 2^{-d} visando garantir a convergência do algoritmo. Pode-se empregar estimadores de erro permitindo a definição de técnicas multigrid adaptáveis como será visto posteriormente.

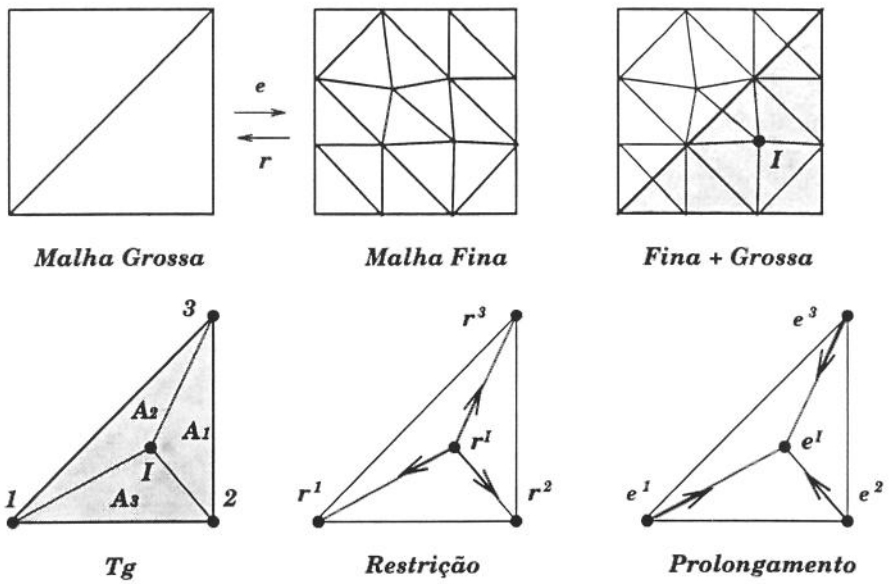


Figura 4.7: Operadores de restrição e interpolação para malhas 2D não-aninhadas.

4.3 Recuperação de Infomações entre Malhas

Para aplicar os operadores de interpolação e prolongamento em malhas não-aninhadas definidos anteriormente, deve-se conhecer, para cada nó de uma malha fina, o elemento da malha grossa onde está contido e os respectivos valores das funções de forma do nó fino neste elemento. Desta maneira, torna-se essencial utilizar algoritmos e estruturas de dados adequadas para recuperar de forma eficiente as informações necessárias para os operadores entre duas malhas.

Este tipo de problema também é encontrado com frequência em áreas como geração de malhas e modelagem sólida, sendo denominado *busca geométrica* [26, 35]. De forma geral, dadas n entidades, deseja-se saber quais delas contém um certo ponto. A solução direta ou trivial consiste em varrer a lista de entidades, verificando se uma delas contém o ponto especificado. O tempo computacional deste procedimento é proporcional a n, ou seja, tem-se um algoritmo de ordem $\mathcal{O}(n)$. Alguns métodos permitem reduzir este custo linear com o número de entidades. Nesta seção, apresenta-se um breve resumo destes procedimentos, tomando-se o caso bidimensional, a partir da referência [35]. A extensão para problemas tridimensionais é direta.

O princípio básico para reduzir a ordem do problema de busca consiste em dividir a região de interesse num conjunto de células com formas simples, fazendo uma correspondência das mesmas com os elementos considerados. Inicialmente, determina-se a célula onde se localiza o ponto e a partir daí quais elementos associados a esta célula contém o ponto dado.

Uma etapa de pré-classificação é necessária, onde a região é dividida em células, determinando-se para cada uma delas os seus elementos associados. O custo desta etapa é de pelo menos $\mathcal{O}(n)$ sendo portanto comparável ao método direto de busca. Logo, a aplicação de um algoritmo para reduzir esta ordem é interessante apenas em casos onde o número total de buscas implica num custo superior a $\mathcal{O}(n)$.

Deve-se levar em consideração ainda a demanda de memória das estruturas de dados auxiliares para armazenar as células e seus elementos, procurando compatibilizar o custo do procedimento de busca e o espaço de memória.

A seguir apresentam-se os principais aspectos de 4 métodos discutidos em [35], onde se encontram procedimentos detalhados para estes algoritmos. São eles:

malha auxiliar uniforme : define-se um retângulo contendo a região de interesse. Divide-se o mesmo num conjunto de $n_x \times n_y$ células iguais denominado malha auxiliar, como ilustrado na Figura 4.8. As coordenadas extremas (X_{min}, Y_{min}) e (X_{max}, Y_{max}) do retângulo são determinadas a partir dos vértices dos elementos. Por sua vez, o tamanho das células é igual a $s = t_m f_m$, onde t_m é o tamanho médio dos elementos e f_m é um parâmetro a ser ajustado conforme se queira melhorar a perfomance em termos de memória ou processamento.

A partir daí, calcula-se o número de divisões como,

$$n_x = (X_{max} - X_{min})/s + 1 \qquad n_y = (Y_{max} - Y_{min})/s + 1$$

e o tamanho das células em cada direção, ou seja,

$$d_x = (X_{max} - X_{min})/n_x \qquad d_y = (Y_{max} - Y_{min})/n_y$$

Criam-se, então, as $n_x \times n_y$ células e associa-se a cada uma delas uma lista de elementos inicialmente vazia. Toma-se cada elemento, determina-se o retângulo $(x_{min}, y_{min}) \times (x_{max}, y_{max})$ que o contém e calculam-se os índices das células que podem interceptá-lo, ou seja,

$$i_{min} = (x_{min} - X_{min})/d_x \qquad i_{max} = (x_{max} - X_{min})/d_x$$

$$j_{min} = (y_{min} - Y_{min})/d_y \qquad j_{max} = (y_{max} - Y_{min})/d_y$$

Para cada célula $C(i, j)$ com índices em $[i_{min}, i_{max}] \times [j_{min}, j_{max}]$, agrega-se o elemento caso se verifique a interseção entre os mesmos. Esta é a etapa de pré-classificação deste método.

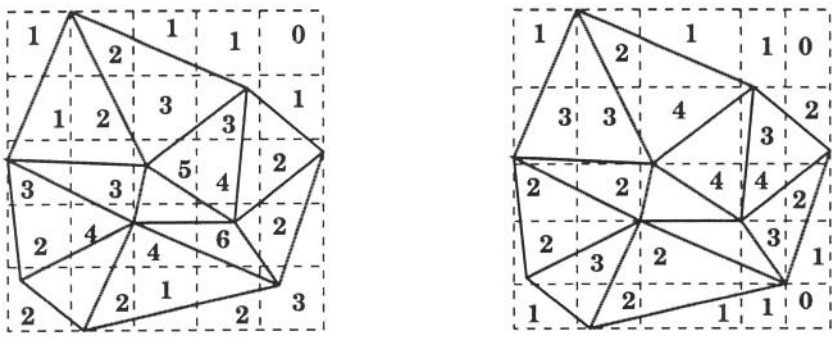
Dado um ponto com coordenadas (x, y), determina-se em qual célula está localizado empregando as relações,

$$i = (x - X_{min})/d_x \qquad j = (y - Y_{min})/d_y$$

Após encontrar a célula, basta verificar em qual elemento associado está localizado o ponto (x, y).

A principal desvantagem deste método está relacionada ao tratamento de regiões com elementos de tamanhos muito variados. Ao se considerar o tamanho das células igual aos elementos menores, tem-se uma maior demanda de memória. Por outro lado, tomando-se para as células o tamanho médio dos elementos, verifica-se em algumas delas um grande número de elementos, degradando a performance do procedimento de busca.

malha auxiliar não-uniforme : para contornar a limitação anterior, pode-se construir células com tamanhos não-uniformes. Para isso, tomam-se divisões com intervalos irregulares em cada eixo cartesiano, reduzindo o número médio de elementos associados a cada célula. Na Figura 4.8, tem-se malhas auxiliares uniforme e não-uniforme indicando-se o número médio de elementos em cada célula. No caso não-uniforme, as divisões em cada eixo coincidem com os vértices dos elementos, a partir da definição de um tamanho mínimo para as células. Apesar de reduzir o número mínimo de elementos em cada célula, tem-se um custo maior na etapa de pré-classificação e no algoritmo de busca [35].



Uniforme **Não-uniforme**

Figura 4.8: Malhas auxiliares uniforme e não-uniforme com o número de elementos por célula [35].

árvore quaternária : a desvantagem dos métodos anteriores ocorre quando se tem uma distribuição arbitrária no tamanho dos elementos. A estrutura de dados denominada árvore quaternária, adapta-se facilmente a qualquer conjunto de elementos, limitando o número máximo de elementos por célula. Basicamente, é uma série de quadrados superpostos na região considerada, os quais dividem-se de forma recursiva, procurando ajustar os tamanhos das células e dos elementos em cada parte do domínio, como ilustrado na Figura 4.9. O quadrado maior é denominado raiz da árvore, enquanto aqueles sem subdivisões chamam-se folhas e os demais são ramos. Algoritmos para a construção da estrutura de dados e busca estão apresentados em [35].

malha uniforme de árvores quaternárias : este método procura aliar as vantagens dos procedimentos de malha uniforme e árvores quaternárias, permitindo o tratamento de regiões tanto com elementos de tamanho uniformes, como regiões com distribuição arbitrária de tamanho dos elementos.

Para isso, gera-se uma malha auxiliar uniforme, onde as células são raízes de árvores quaternárias. Neste caso, no processo de busca, determina-se inicialmente a célula da malha uniforme que contém o ponto dado. Posteriormente, chega-se a folha da árvore correspondente ao ponto, varrendo-se então a lista de elementos associados a esta folha.

Empregou-se a árvore quaternária neste trabalho para determinar o elemento da malha grossa contendo um nó fino e os respectivos valores das funções de forma. Estas informações são geradas antes da aplicação de uma técnica multigrid e armazenadas numa estrutura de dados, a qual é acessada sempre que os operadores de interpolação e restrição são usados. No caso tridimensional, tem-se uma estrutura análoga utilizando, no entanto, árvores octaédricas.

4.4 Técnicas Multigrid

A partir da definição dos operadores de transferência, pode-se reescrever a estratégia de correção de malha grossa (CMG) como [31]:

$\mathbf{v}^h \leftarrow CMG(\mathbf{v}^h, \mathbf{b}^h)$
Iterar ν_1 vezes em $\mathbf{A}^h\mathbf{u}^h = \mathbf{b}^h$ na malha Ω^h com aproximação inicial $\mathbf{v}^h$
Calcular $\mathbf{r}^{2h} = I_h^{2h}(\mathbf{b}^h - \mathbf{A}^h\mathbf{v}^h)$
    Resolver a equação $\mathbf{A}^{2h}\mathbf{e}^{2h} = \mathbf{r}^{2h}$ em Ω^{2h}
Corrigir a aproximação da malha fina: $\mathbf{v}^h \leftarrow \mathbf{v}^h + I_{2h}^h\mathbf{e}^{2h}$
Iterar ν_2 vezes em $\mathbf{A}^h\mathbf{u}^h = \mathbf{b}^h$ na malha Ω^h com aproximação inicial $\mathbf{v}^h$.

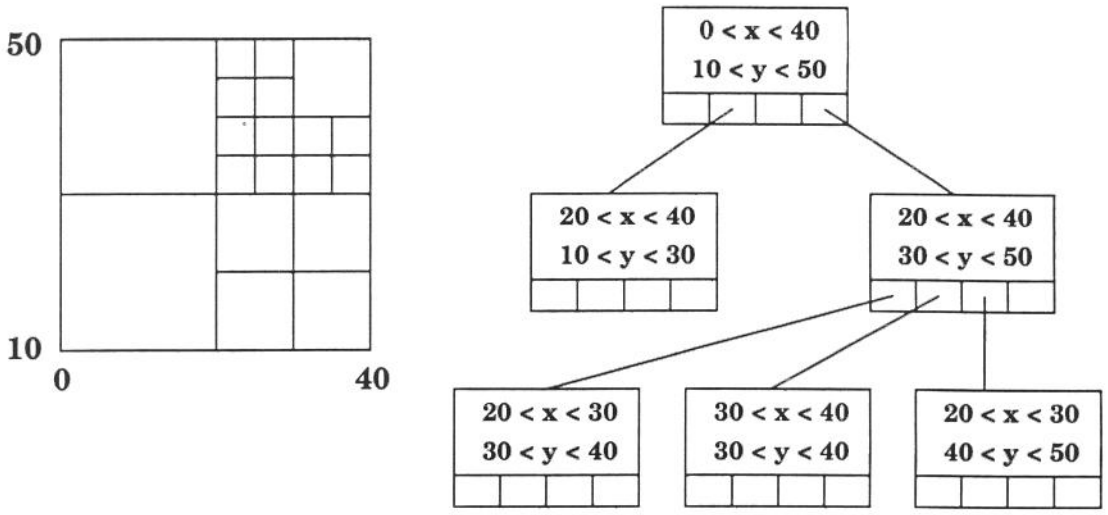


Figura 4.9: Exemplo de uma estrutura auxiliar baseada em árvores quaternárias [35].

Portanto, itera-se ν_1 vezes na malha fina até que a taxa de decrescimento do erro se estabilize. Calcula-se o resíduo na aproximação $\mathbf{v}^h$ em Ω^h e transfere-se o mesmo, através do operador de restrição I_h^{2h}, para a malha grossa. Determina-se a solução $\mathbf{e}^{2h}$ da equação do resíduo, aplicando-se o operador de pronlogamento I_{2h}^h para corrigir a solução corrente $\mathbf{v}^h$ em Ω^h. Executam-se então ν_2 iterações para eliminar as componentes oscilatórias de $\mathbf{v}^h$.

Alguns comentários podem ser feitos sobre a estratégia de correção definida:

- em relação ao número de pré e pós-relaxações ν_1 e ν_2, pode-se estabelecer valores fixos para os mesmos no início do algoritmo. Outra possibilidade é utilizar um critério adaptável onde se calcula a norma do resíduo após cada iteração. Caso entre duas iterações sucessivas não se verificar uma queda em torno de 60% na norma de $\mathbf{r}^{nh}$ em Ω^{nh} deve-se passar para a próxima malha [29];

- o termo $\mathbf{A}^{2h}$ indica a representação na malha Ω^{2h} da matriz $\mathbf{A}^h$. Para malhas aninhadas, tem-se uma propriedade variacional relacionando estas matrizes [31],

$$\begin{aligned} \mathbf{A}^{2h} &= I_h^{2h}\mathbf{A}^h I_{2h}^h \\ I_h^{2h} &= c(I_{2h}^h)^T \quad c \in \Re \end{aligned} \tag{4.12}$$

  Para malhas não-aninhadas, deve-se calcular para cada malha Ω^{nh} a matriz $\mathbf{A}^{nh}$ correspondente através do método usado na solução do problema.

- a equação de resíduo na malha grossa $\mathbf{A}^{2h}\mathbf{e}^{2h} = \mathbf{r}^{2h}$ é análoga ao sistema $\mathbf{A}^h\mathbf{u}^h = \mathbf{b}^h$ em Ω^h. Assim, torna-se viável aplicar o mesmo esquema de correção anterior utilizando agora uma malha Ω^{4h}. Este procedimento pode ser aplicado sucessivamente chegando-se a um algoritmo recursivo onde a malha mais grossa, possuindo um número menor de variáveis, é resolvida de forma direta.

A recursividade na estratégia de correção de malha grossa permite definir uma família de métodos multigrid, denominada Ciclos μ, dada por [31]:

$\mathbf{v}^h \leftarrow \mathrm{M}\mu^h(\mathbf{v}^h, \mathbf{b}^h)$

1. Iterar ν_1 vezes em $\mathbf{A}^h\mathbf{u}^h = \mathbf{b}^h$ com aproximação inicial $\mathbf{v}^h$
2. Se Ω^h = malha mais grossa $\rightarrow$ passo 4
   Senão $\mathbf{b}^{2h} \leftarrow I_h^{2h}(\mathbf{b}^h - \mathbf{A}^h\mathbf{v}^h)$
   $\mathbf{v}^{2h} \leftarrow \mathbf{0}$
   $\mathbf{v}^{2h} \leftarrow \mathrm{M}\mu^{2h}(\mathbf{v}^{2h}, \mathbf{b}^{2h})$ ν_0 vezes
3. Corrigir $\mathbf{v}^h \leftarrow \mathbf{v}^h + I_{2h}^h\mathbf{v}^{2h}$
4. Iterar ν_2 vezes em $\mathbf{A}^h\mathbf{u}^h = \mathbf{b}^h$ com aproximação inicial $\mathbf{v}^h$

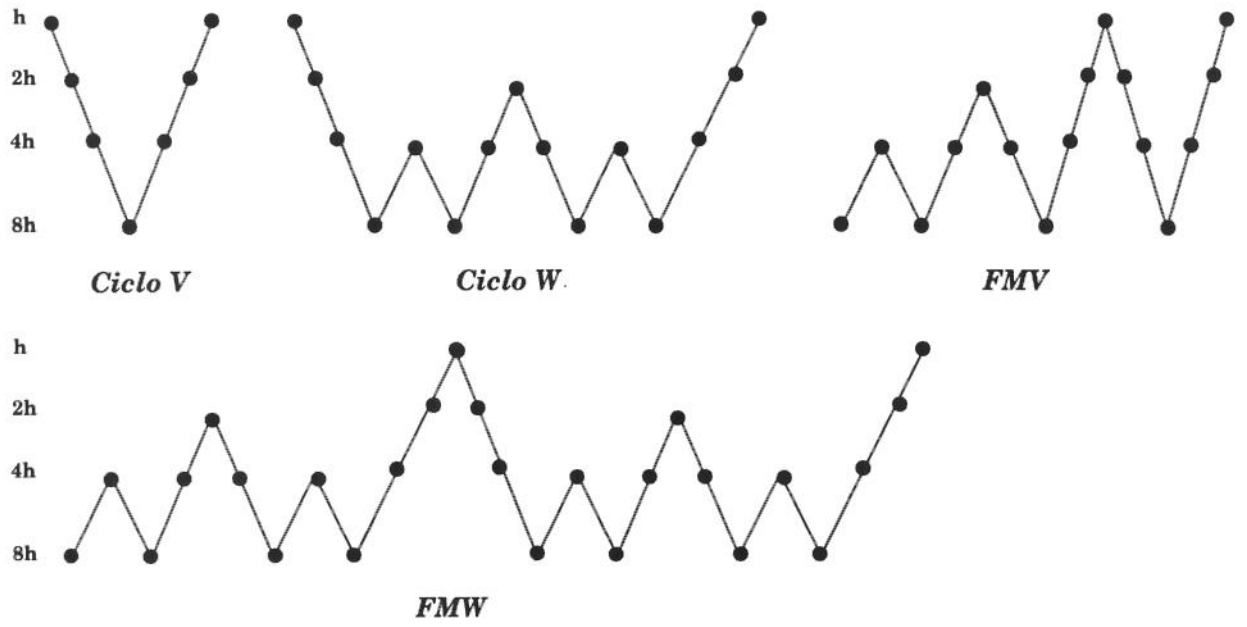


Figura 4.10: Estratégias multigrid.

Na notação usada, $\mathbf{v}^{2h}$ indica a solução da equação do resíduo $\mathbf{e}^{2h}$; da mesma maneira $\mathbf{b}^{2h}$ é utilizado ao invés de $\mathbf{r}^{2h}$, pois os mesmos são termos independentes dos sistemas de equações envolvidos.

Para $\nu_0 = 1$, tem-se o Ciclo $\mu = V$, o qual partindo da malha mais fina alcança a malha mais grossa mapeando o resíduo entre as malhas, retornando para a malha mais fina aplicando as correções em cada nível. Para $\nu_0 = 2$, define-se o Ciclo $\mu = W$ que juntamente com o esquema V estão ilustrados na Figura 4.10.

No entanto, as estratégias multigrid apresentadas não incorporam o conceito de iterações aninhadas. Para isso, tem-se a estratégia FMV, onde cada ciclo V é precedido por um ou mais ciclos V menores, conforme o valor do parâmetro ν_0, com o objetivo de prover uma melhor aproximação inicial em cada nível. Este esquema, também ilustrado na Figura 4.10 com $\nu_0 = 1$, pode ser resumido de forma recursiva como [31]:

$\mathbf{v}^h \leftarrow \mathrm{FMV}^h(\mathbf{v}^h, \mathbf{b}^h)$

1. Se Ω^h = malha mais grossa $\rightarrow$ passo 3
   Senão $\mathbf{b}^{2h} \leftarrow I_h^{2h}(\mathbf{b}^h - \mathbf{A}^h\mathbf{v}^h)$
   $\mathbf{v}^{2h} \leftarrow \mathbf{0}$
   $\mathbf{v}^{2h} \leftarrow \mathrm{FMV}^{2h}(\mathbf{v}^{2h}, \mathbf{b}^{2h})$ μ vezes
2. Corrigir $\mathbf{v}^h \leftarrow \mathbf{v}^h + I_{2h}^h\mathbf{v}^{2h}$
3. $\mathbf{v}^h \leftarrow \mathrm{MV}^h(\mathbf{v}^h, \mathbf{b}^h)$ ν_0 vezes

O primeiro passo deste algoritmo, basicamente, mapeia o termo independente da malha fina consecutivamente até atingir a malha mais grossa, na qual aplicam-se relaxações ou resolve-se de forma direta obtendo-se a solução exata. A partir daí, o vetor obtido é prolongado e usado como solução inicial para um ou mais ciclos V no próximo nível, dependendo do valor de ν_0, e recursivamente para os demais níveis.

Uma outra estratégia denominada FMW, utiliza ciclos W menores para obter uma melhor aproximação inicial para um próximo nível, como apresentado novamente na Figura 4.10 para $\nu_0 = 1$.

Algumas variantes destes algoritmos podem ser empregados. Por exemplo, ao final de um ciclo FMV, concatenam-se vários ciclos V, sendo este esquema denominado FMVV. De forma análoga, define-se o algoritmo FMWV, estando estas duas variantes ilustradas na Figura 4.11.

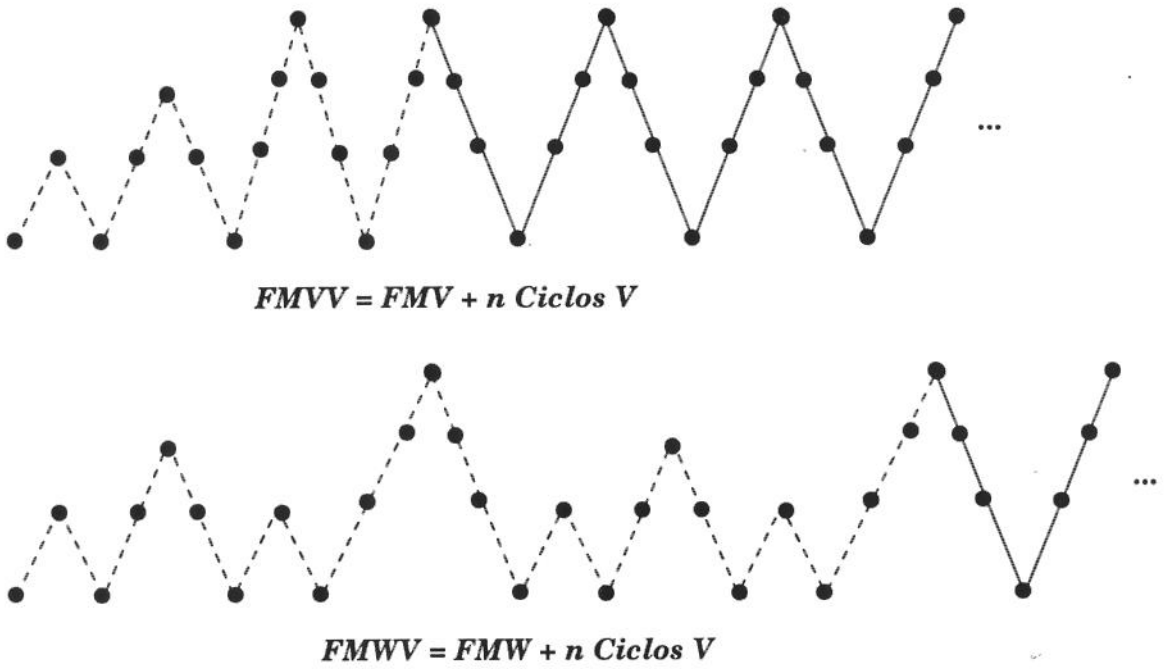


Figura 4.11: Estratégias multigrid FMVV e FMWV.

Em [29] foi introduzido a variante FAS (*Full Aproximation Storage*) para os algoritmos multigrid. Ao invés de se armazenar em cada nível k a correção $\mathbf{e}^{kh}$, considera-se a aproximação $\mathbf{v}^{kh}$ dada pela soma da correção $\mathbf{e}^{kh}$ e da aproximação da malha fina $\mathbf{v}^{(k+1)h}$, ou seja,

$$\mathbf{v}^{kh} = I^{kh}_{(k+1)h}\mathbf{v}^{(k+1)h} + \mathbf{e}^{kh} \quad k = 1, \ldots, N_m - 1 \tag{4.13}$$

onde N_m é o número de malhas.

Desta maneira, denotam-se as equações de correção como,

$$\mathbf{A}^{kh}\mathbf{e}^{kh} = \bar{\mathbf{b}}^{kh} \tag{4.14}$$

sendo, $\bar{\mathbf{b}}^{kh} = \mathbf{A}^{kh}(I^{kh}_{(k+1)h}\mathbf{v}^{kh}) + I^{kh}_{(k+1)h}(\bar{\mathbf{b}}^{(k+1)h} - \mathbf{A}^{(k+1)h}\mathbf{v}^{(k+1)h})$ e $\bar{\mathbf{b}}^{h} = \mathbf{b}^{h}$ na malha mais fina.

Para casos lineares elípticos, a variante FAS não introduz nenhuma diferença em relação aos algoritmos anteriores. A grande vantagem é o tratamento de problemas não-lineares, empregando os mesmos elementos do caso linear. Para isso, suponha que $\mathbf{A}^h\mathbf{u}^h = \mathbf{b}^h$ seja uma equação não-linear, sendo $\mathbf{v}^h$ a aproximação inicial obtida por um esquema de relaxação.

Novamente, tem-se o resíduo $\mathbf{r}^h = \mathbf{b}^h - \mathbf{A}^h\mathbf{v}^h$ e a correção $\mathbf{e}^h = \mathbf{u}^h - \mathbf{v}^h$. Neste caso, a equação do resíduo é dada por $\mathbf{A}^h\mathbf{u}^h - \mathbf{A}^h\mathbf{v}^h = \mathbf{b}^h$, com a seguinte aproximação na malha grossa,

$$\mathbf{A}^{2h}\mathbf{u}^{2h} - \mathbf{A}^{2h}I_h^{2h}\mathbf{v}^h = I_h^{2h}\mathbf{b}^h \tag{4.15}$$

Ao se interpolar a aproximação $\mathbf{v}^{2h}$ de $\mathbf{u}^{2h}$ para a malha fina Ω^h, considera-se a correção $\mathbf{e}^{2h} = \mathbf{v}^{2h} - I_h^{2h}\mathbf{u}^h$. Assim, o operador de interpolação é dado de forma geral como [29],

$$\mathbf{u}^{(k+1)h} \leftarrow \mathbf{u}^{(k+1)h} + I_{kh}^{(k+1)h}(\mathbf{u}^{kh} - I_{(k+1)h}^{kh}\mathbf{u}^{(k+1)h}) \tag{4.16}$$

Geralmente, tem-se que $I_{kh}^{(k+1)h}I_{(k+1)h}^{kh}\mathbf{u}^{(k+1)h} \neq \mathbf{u}^{(k+1)h}$. A principal característica do FAS é que a função armazenada em cada Ω^{kh} $(k = 1, \ldots, N_m - 1)$ é uma aproximação para a solução da malha fina Ω^h, permitindo manipular aproximações mais precisas nas malhas grossas. Uma desvantagem é o custo de cerca de duas vezes maior para o cálculo do termo independente nas malhas grossas.

Um outro algoritmo denominado FAC (*Fast Adaptive Composite Grid*) está apresentado em [71] para o caso de diferenças finitas. Definem-se discretizações aninhadas, utilizando-se várias malhas regulares correspondentes a refinamentos sucessivos. Num processo de vários níveis, a malha fina é composta da união destas regiões refinadas, sendo as relaxações efetuadas de forma local apenas nas variáveis introduzidas em cada nível.

4.4.1 Algoritmos adaptáveis

Para evitar a necessidade de especificar todas as malhas empregadas na solução por métodos multigrid, pode-se acoplar algumas estratégias definidas anteriormente com um estimador de erros, como por exemplo o algoritmo ZZ discutido no Capítulo 3. Definem-se aproximações do tipo *top-down* e *bottom-up*.

No primeiro caso, partindo-se de uma malha inicial e considerando por exemplo o método de Gauss para a solução do respectivo sistema de equações, aplica-se sucessivamente o estimador e uma técnica de refinamento visando atingir uma solução com erro percentual menor que $\bar{\eta}$. Neste processo, pode-se obter ao final uma malha com um número elevado de variáveis, principalmente em problemas com fortes singularidades. Como a solução desta malha fina, pode não ser viável via método direto, aplicam-se esquemas multigrid.

No processo de refinamento, o novo tamanho médio do elemento h' é determinado divindo-se o tamanho corrente h por 2, ou seja, $h' \approx h/2$. As malhas obtidas desta maneira podem não ser adequadas para métodos multigrid, pois pode ocorrer uma penetração muito forte dos elementos entre malhas sucessivas, inviabilizando a convergência dos esquemas empregados. Para contornar esta limitação, toma-se a malha mais fina e aplica-se o gerador duplicando o tamanho médio dos elementos até atingir por exemplo um total de 4 malhas. Ao final, emprega-se um esquema multigrid para obter a solução do problema.

Apesar da demanda computacional para a solução da malha fina via multigrid ser em geral inferior ao método de Gauss, como mencionado acima outras malhas podem ser necessárias, implicando num custo de geração das mesmas e montagem das matrizes dos respectivos sistemas de equações. Desta forma, este procedimento é interessante apenas em situações onde a utilização de métodos diretos e iterativos torna-se inviável, como por exemplo em problemas singulares ou em casos tridimensionais.

Em [67, 68, 81], tem-se um procedimento parecido, onde parte-se da malha mais fina e geram-se os demais níveis através de procedimentos frontais ou de Delaunay, não se aplicando no entanto nenhum refinamento adaptável.

Na variante *bottom-up*, dados um erro percentual $\bar{\eta}$ e uma precisão ξ, a idéia geral é partindo da malha grossa, resolver o problema aplicando estratégias multigrid, sendo as várias malhas geradas a partir da distribuição de erro dada pelo estimador ZZ.

Optou-se pelo esquema FMV com $\nu_0 = 2$ ilustrado na Figura 4.12. Assim, resolve-se a malha grossa de forma direta, aplica-se o estimador ZZ e uma estratégia de refinamento, obtendo-se a segunda malha. A partir daí, consideram-se 2 ciclos V intermediários e chega-se a terceira malha de forma análoga, após aplicar o algoritmo ZZ. Este processo é repetido até atingir o número máximo de malhas especificadas ou quando o erro estimado for próximo ou inferior ao valor $\bar{\eta}$. Neste caso, no refinamento das malhas, visando garantir a convergência dos métodos multigrid, impõe-se a seguinte restrição no novo tamanho h' dos elementos,

$$\frac{1}{3}h \leq h' \leq \frac{3}{2}h \tag{4.17}$$

Observa-se que nas malhas intermediárias, apenas 2 ciclos V internos não garantem a solução com a precisão ξ. Como estes níveis são empregados apenas como esquemas de correção, não havendo necessidade de obter uma solução exata, esta estratégia mostra-se viável. No entanto, ao se atingir a malha mais fina, aplica-se uma estratégia multigrid, não necessariamente aquela empregada na geração das malhas, até alcançar uma solução com precisão ξ. Aplicou-se este procedimento a casos planos como será apresentado posteriormente.

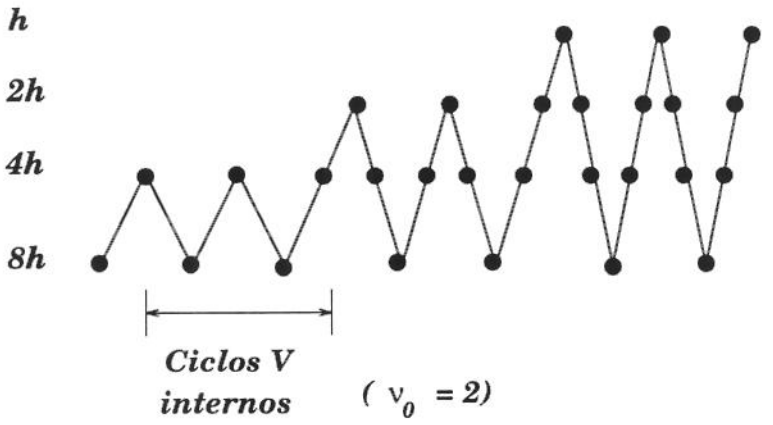


Figura 4.12: Esquema FMV adaptável para a geração das malhas.

4.4.2 Custos dos esquemas multigrid

Torna-se fundamental determinar o custo computacional e o espaço de memória dos algoritmos multigrid discutidos na seção anterior. Em geral, o custo é representado como o número equivalente de iterações do algoritmo de relaxação na malha fina.

Supondo que sejam empregadas N_m malhas para a solução de um problema por multigrid, uma forma mais precisa para estimar o número de operações deve considerar os seguintes ítens:

iterações : excetuando a malha mais grossa, na qual aplica-se método direto de Gauss, a demanda computacional em cada nível i será igual ao número total de iterações realizado (NIT_i) em todos os ciclos vezes o custo da iteração (CIT_i) do método numérico empregado. Logo,

$$\sum_{i=2}^{N_m} (\mathrm{NIT}_i)\,(\mathrm{CIT}_i) \tag{4.18}$$

Por exemplo, utilizando como esquema de relaxação o algoritmo de Gauss-Seidel, tem-se a partir da Tabela 2.1 que $\mathrm{CIT}_i = N_i(2m_i - 1)$.

solução direta na malha grossa : neste caso, deve-se fatorar a matriz $\mathbf{A}_1$ uma única vez e empregar processos de substituição sempre que a malha mais grossa for visitada. Portanto, a partir do Capítulo 2, o custo computacional destes procedimentos é dado por,

$$\left[\sum_{i=1}^{N_1} m_i^U(m_i^U + 3)\right] + 2N_1(m_1 - 1)n_1 \tag{4.19}$$

onde n_1 representa o número total de vezes que a técnica multigrid utilizada passou pelo nível 1. Por exemplo, para um ciclo W, tem-se $n_1 = 4$. Se na malha grossa for empregado um esquema de relaxação, despreza-se o custo indicado em (4.19) e considera-se a expressão (4.18) com o índice $i = 1, \ldots, N_m$. Caso seja empregado um método iterativo distinto das demais malhas, o custo será $n_1\mathrm{CIT}_1$, onde CIT_1 é o número de operações para uma iteração do algoritmo usado.

operadores : a partir da Figura 4.7, observa-se que para cada equação de um nó fino I são necessárias 3 multiplicações para mapear o resíduo ou a correção nos respectivos operadores. Portanto, tem-se um custo de $3N_i$ operações em cada nível i sempre que uma restrição ou um prolongamento é executado. Para casos tridimensionais, como os tetraedros lineares possuem 4 nós, o custo passa a ser de $4N_i$.

Denotando n_i^r e n_i^p como o número total de operações de restrição e pronlogamento executados nos ciclos da estratégia multigrid, o custo dos operadores é dado por,

$$N_{\text{nós}} \sum_{i=2}^{N_m} N_i(n_i^r + n_i^p) \tag{4.20}$$

sendo $N_{\text{nós}}$ o número de nós do elemento; 3 para triângulos e 4 para tetraedros.

cálculo do resíduo : ao se aplicar o operador de restrição entre as malhas i e $i-1$, deve-se calcular o resíduo $\mathbf{r}_i$ demandando $N_i(2m_i - 1)$ operações. Assim, para todos os níveis, tem-se o seguinte custo,

$$\sum_{i=2}^{N_m} N_i(2m_i - 1)n_i^r \tag{4.21}$$

Observa-se que o custo do cálculo do resíduo é igual a uma iteração do método de Gauss-Seidel no mesmo nível i. Neste sentido, a estratégia de correção adaptável discutida na seção 4.5 não é interessante, pois o cálculo do resíduo a cada iteração do esquema de relaxação demanda um esforço equivalente ao da própria iteração.

O número de operações total para a solução de um sistema de equações via técnicas multigrid em malhas não-aninhadas é igual a soma das expressões (4.18) a (4.21). Para se obter o número equivalente de iterações na malha fina, basta dividir o valor obtido pelo custo de uma iteração do esquema de relaxação utilizado.

No que se refere ao espaço de memória, tem-se uma demanda computacional superior em relação aos métodos iterativos do Capítulo 2, não apenas pelo fato de se trabalhar com várias malhas, mas também pela necessidade de se armazenar informações auxiliares utilizadas nos operadores de restrição e pronlogamento, tais como incidência e numeração das variáveis. Desta forma, deve-se contabilizar o espaço de memória para os seguintes parâmetros:

matrizes + vetores : neste caso, consideram-se as matrizes e os vetores dos sistemas de equações de cada malha. Para um nível i, tem-se um total de $N_i(m_i + 2)$ posições de memória, além do espaço n_i^{aux} dos vetores auxiliares do esquema de relaxação empregado. Por exemplo, para o método GS em matrizes esparsas é necessário mais um vetor fazendo $n_i^{aux} = N_i$. Logo, de forma geral o número de posições de ponto flutuante necessário é dado por,

$$\sum_{i=1}^{N_m} [N_i(m_i + 2) + n_i^{aux}] \tag{4.22}$$

tabelas de mapeamento : como discutido na seção 4.2.4, para aplicar os operadores de transferência, deve-se conhecer para cada nó fino I, o triângulo da malha grossa que o contém e neste elemento o valor das coordenadas de área (A_1, A_2, A_3) do nó I. Por exemplo, supondo que o nó fino 100 está contido no triângulo 32 e possui coordenadas de área (0,1; 0,5; 0,4) armazenam-se os seguintes dados:

$$32 \quad 0,1 \quad 0,5$$

onde emprega-se a relação $A_3 = 1 - A_1 - A_2$.

Observa-se que o número do elemento armazenado como um inteiro requer a metade do número de *bytes* de cada coordenada, considerada como um termo de precisão dupla. Logo, no total são necessárias $2,5$ posições de ponto flutuante. No caso tridimensional, tem-se 4 coordenadas de volume V_i $(i = 1, \ldots, 4)$, resultando um total de $3,5$ posições.

Portanto, estas tabelas necessitam do seguinte espaço de precisão dupla,

$$n_t \sum_{i=2}^{N_m} N_i^{\text{nós}} \qquad n_t = 2,5 \text{ ou } n_t = 3,5 \tag{4.23}$$

onde $N_i^{\text{nós}}$ é o número total de nós da malha i.

equações : armazena-se para cada nó, a numeração das suas variáveis. Para problemas elásticos bi e tridimensionais, tem-se, respectivamente, 2 e 3 variáveis em cada nó, resultando em,

$$n_{eq} \sum_{i=1}^{N_m} N_i^{\text{nós}} \tag{4.24}$$

posições de precisão dupla com $n_{eq} = 1$ e $n_{eq} = 1,5$, respectivamente para problemas planos e espaciais.

incidência : da mesma forma, armazena-se a incidência de cada elemento necessitando o seguinte número de posições de precisão dupla,

$$n_{inc} \sum_{i=1}^{N_m} N_i^{els} \tag{4.25}$$

sendo N_i^{els} o número de elementos da malha i; com $n_{inc} = 1,5$ e $n_{inc} = 2$, respectivamente para problemas bi e tridimensionais.

O espaço de memória total é dado pela soma das expressões (4.22) a (4.25), lembrando-se que o valor obtido deve ser multiplicado pelo fator $\frac{8}{1024}$ para se ter os resultados em Kbytes.

4.5 Dedução Variacional dos Operadores

Nesta seção, apresenta-se de maneira mais formal a obtenção dos operadores de restrição e prolongamento, aplicados para a transferência de informação entre as malhas. A abordagem empregada tem a vantagem adicional de ser válida tanto para malhas aninhadas como para não-aninhadas. Além disso, permite deduzir os operadores de transferência para problemas envolvendo graus de liberdade generalizados, tal como ocorre em problemas de flexão de placas, cascas e modelos mistos em geral.

Para simplificar a apresentação, impõe-se que o problema variacional a ser resolvido com métodos multigrid está dado por: *determinar* $u \in H_0^1(\Omega)$ *tal que*,

$$a(u,v) = b(v) \quad \forall v \in H_0^1(\Omega) \tag{4.26}$$

onde $H_0^1(\Omega)$ é o espaço de Hilbert cujos elementos são funções (classes) com valor nulo no contorno, quadrado-integráveis com derivadas primeiras também quadrado-integráveis.

A aproximação por elementos finitos e técnicas multigrid implica na solução dos seguintes problemas,

$$a(u_k,v) = b(v) \quad \forall v \in V_k \tag{4.27}$$

onde $V_k = \{v \in C_0(\Omega) \mid v|_K \text{ é linear } \forall K \in \mathcal{T}_k\} \subset H_0^1(\Omega)$; $\{\mathcal{T}_k, k = 1,2,\ldots\}$ é uma família de triangulações do domínio Ω, sendo em geral não-aninhadas, ou seja, $\mathcal{T}_k \not\subset \mathcal{T}_{k+1}$. A condição anterior implica também que $V_k(\Omega) \not\subset V_{k+1}(\Omega)$.

Seja I_k o operador usual de projeção em V_k dado por,

$$\begin{array}{rccl} I_k: & C_0(\Omega) & \rightarrow & V_k(\Omega) \\ & u(x) & \rightarrow & I_k u(x) = \sum_{n_i \in \mathcal{N}_k} u(n_i)\phi_{k,i}(x) \end{array} \tag{4.28}$$

sendo $C_0(\Omega)$ o espaço de funções contínuas em Ω e nulas (homogêneas) no contorno; $\mathcal{N}_k$ é o conjunto de pontos nodais correspondentes à partição $\mathcal{T}_k$; $\phi_{k,i}(x)$ é a função de base associada ao nó n_i, correspondente ao tipo de elemento finito definindo a triangulação $\mathcal{T}_k$.

Emprega-se abaixo a seguinte notação,

$$I_k u(x) = \sum_{n_i \in \mathcal{N}_k} u(n_i)\phi_{k,i}(x) = \Phi_k \cdot \mathbf{u}_k \tag{4.29}$$

onde Φ e $\mathbf{u}$ são, respectivamente, os vetores de funções bases de interpolação e de valores nodais.

Pela própria definição do interpolante, segue-se que,

$$I_k u(x) = u(x) \quad \forall u(x) \in V_k(\Omega) \quad \forall k = 1,2,\ldots \tag{4.30}$$

Designando-se como $\tilde{u}_k \in V_k$ uma aproximação da equação (4.27), o resíduo associado a mesma está dado por,

$$\tilde{b}(v) = b(v) - a(\tilde{u}_k, v) = (r_k, v) = \int_\Omega r_k \Phi_k \, d\Omega \cdot \mathbf{v}_k = \mathbf{r}_k \cdot \mathbf{v}_k \quad \forall v \in V_k \tag{4.31}$$

onde $\mathbf{v}_k$ é o vetor de valores nodais associado a $v \in V_k$.

O problema de transferir o resíduo, associado a $\tilde{u}_k$, da malha $\mathcal{T}_k$ para a malha $\mathcal{T}_{k-1}$ pode ser efetuado de diferentes maneiras, tais como:

1. $$\tilde{b}(v) = b(I_k v) - a(\tilde{u}_k, I_k v) = (r_k, I_k v) = \int_\Omega r_k I_k v \, d\Omega \quad \forall v \in V_{k-1} \tag{4.32}$$

Para todo $v \in V_{k-1}$ resulta,

$$\begin{aligned} I_k v &= \sum_{n_i \in \mathcal{N}_k} v_{k-1}(n_i)\phi_{k,i}(x) = \sum_{n_i \in \mathcal{N}_k} \left[\sum_{n_j \in \mathcal{N}_{k-1}} v(n_j)\phi_{k-1,j}(n_i) \right] \phi_{k,i}(x) \\ &= \Phi_k(x) \cdot \mathbf{N}_{k-1}^k \mathbf{v}_{k-1} \end{aligned} \tag{4.33}$$

com,

$$\mathbf{N}_{k-1}^k = \begin{bmatrix} \phi_{k-1,1}(n_1) & \phi_{k-1,2}(n_1) & \cdots & \phi_{k-1,N_{k-1}}(n_1) \\ \phi_{k-1,1}(n_2) & \phi_{k-1,2}(n_2) & \cdots & \phi_{k-1,N_{k-1}}(n_2) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_{k-1,1}(n_{N_k}) & \phi_{k-1,2}(n_{N_k}) & \cdots & \phi_{k-1,N_{k-1}}(n_{N_k}) \end{bmatrix}_{N_{k-1} \times N_k} \tag{4.34}$$

Desta maneira, tem-se,

$$\begin{aligned} \tilde{b}(v) &= \int_\Omega r_k \Phi_k(x) \, d\Omega \cdot \mathbf{N}_{k-1}^k \mathbf{v}_{k-1} = (\mathbf{N}_{k-1}^k)^T \int_\Omega r_k \Phi_k(x) \, d\Omega \cdot \mathbf{v}_{k-1} \\ &= (\mathbf{N}_{k-1}^k)^T \mathbf{r}_k \cdot \mathbf{v}_{k-1} = \mathbf{r}_{k-1} \cdot \mathbf{v}_{k-1} \end{aligned} \tag{4.35}$$

A expressão anterior proporciona o operador que permite transferir o resíduo $\mathbf{r}_k$ em $\mathcal{T}_k$ para $\mathbf{r}_{k-1}$ em $\mathcal{T}_{k-1}$, dado por,

$$\mathbf{r}_{k-1} = I_k^{k-1} \mathbf{r}_k = (\mathbf{N}_{k-1}^k)^T \mathbf{r}_k \tag{4.36}$$

onde $I_k^{k-1} = (\mathbf{N}_{k-1}^k)^T$.

A partir daí, verifica-se que o operador I_k^{k-1} é uma matriz esparsa de ordem $N_{k-1} \times N_k$, cujo elemento genérico $(I_k^{k-1})_{ij}$ corresponde ao valor,

$$\phi_{k-1,i}(n_j) \quad i \in \{1, \ldots, N_{k-1}\}, j \in \{1, \ldots, N_k\} \tag{4.37}$$

ou seja, é o valor da função base, associada ao nó n_i da triangulação $\mathcal{T}_{k-1}$, calculada sob o nó n_j da triangulação $\mathcal{T}_k$.

Do ponto de vista computacional, este operador de restrição é obtido através da contribuição local do resíduo, correspondente ao nó n_j de $\mathcal{T}_k$, sob os nós do triângulo $T_g \in \mathcal{T}_{k-1}$, onde n_j está localizado, conforme a Figura 4.7. Da própria definição, este operador é consistente, no sentido que o resíduo se conserva ao se passar de uma triangulação para outra.

2. outra maneira de calcular esta transferência, conduzindo a uma outra definição de operador I_k^{k-1}, é dada por,

$$\tilde{b}(v) = b(v) - a(\tilde{u}_k, v) \quad \forall v \in V_{k-1} \tag{4.38}$$

A dificuldade desta última expressão reside no cálculo da forma $a(\tilde{u}_k, v)$, já que a mesma envolve funções definidas em dois espaços diferentes.

Neste trabalho, adotou-se a primeira forma de definição do operador I_k^{k-1}. Desta maneira, após ν_1 iterações no nível k, resolve-se no nível $k-1$ o problema,

$$a(e_{k-1}, v) = \tilde{b}(v) \quad \forall v \in V_{k-1} \tag{4.39}$$

com $\tilde{b}(v) = b(I_k v) - a(\tilde{u}_k, I_k v)\ \forall v \in V_{k-1}$.

O outro processo necessário nos métodos multigrid consiste em pronlogar a solução determinada no nível $k-1$ para o nível k. Este pronlogamento deve ser consistente com o operador de restrição empregado. Assim, tem-se

1. $$e_k = I_{k-1}^k e_{k-1} = I_k e_{k-1} = \sum_{n_i \in N_k} \phi_{k,i}(x) e_{k-1}(n_i) = \Phi_k \cdot \mathbf{N}_{k-1}^k \mathbf{e}_{k-1} = \Phi_k \cdot \mathbf{e}_k \tag{4.40}$$

   onde $\mathbf{e}_k = \mathbf{N}_{k-1}^k \mathbf{e}_{k-1}$. Portanto, $I_{k-1}^k = \mathbf{N}_{k-1}^k$, mostrando que $I_{k-1}^k = (I_k^{k-1})^T$.

2. no caso de se adotar a segunda forma de definição do operador de restrição, o operador de pronlogamento será definido por: *determinar o elemento* $I_{k-1}^k e_{k-1}$ *dado por*,

   $$a(I_{k-1}^k e_{k-1}, v) = a(e_{k-1}, v) \quad \forall v \in V_k \tag{4.41}$$

   Novamente, este processo implica num esforço de cálculo da mesma ordem que problema original a ser resolvido, sendo esta opção não recomendada.

Finalmente, é interessante observar que esta forma de dedução dos operadores permite estendê-los para modelos mais gerais, tais como, por exemplo, problemas de placas, cascas ou mistos.

4.6 Convergência dos Métodos Multigrid

Um aspecto importante de qualquer método numérico está relacionado a sua convergência. Para problemas lineares elípticos modelados por diferenças finitas ou elementos finitos, considerando malhas aninhadas uniformes, demonstrou-se a convergência dos procedimentos multigrid em [11, 38, 56, 69, 75], sendo o custo da solução linear com o número de equações, ou seja, tem-se uma ordem $\mathcal{O}(N)$ para $N \to \infty$.

No entanto, como mencionado anteriormente, neste trabalho consideram-se malhas não-aninhadas. A principal motivação para este fato é a aplicação de estratégias multigrid em problemas tridimensionais discretizados com tetraedros lineares. Para casos bidimensionais, é possível refinar um triângulo em 4 subtriângulos congruentes, ligando-se os pontos do meio das arestas. Entretanto, em geral não é possível subdividir um tetraedro em 8 elementos idênticos, podendo-se obter malhas degeneradas numa sequência de refinamentos sucessivos [108, 109].

Observa-se que o conceito de espaços de aproximação aninhados não é essencial em métodos multigrid, permitindo assim utilizar malhas finas adaptadas mais adequadas para a solução de um problema. Além disso, verifica-se a mesma ordem ótima $\mathcal{O}(N)$, podendo-se observar diferenças apenas na taxa de convergência.

Aspectos de convergência em malhas não-aninhadas foram apresentadas inicialmente em [37]. Posteriormente, introduziu-se e demonstrou-se a convergência de um algoritmo aninhado não-padrão, definindo-se um nível extra na malha fina, de tal forma que todos os níveis são tratados como esquemas de correção [38]. Este mesmo procedimento foi estendido para malhas não-aninhadas em [39].

Entretanto, algumas condições devem ser impostas nas malhas dos vários níveis para se alcançar convergência. De forma geral, a relação entre os números de elementos entre duas malhas sucessivas

pode ser arbitrária. No entanto, deseja-se que as mesmas sejam comparáveis, no sentido que cada elemento de um nível k pode ser coberto por um número finito de elementos das malhas grossa $k-1$ e fina $k+1$. Além disso, o número de incógnitas de cada nível deve crescer de forma geométrica com uma taxa maior que 2 [109].

A convergência de métodos multigrid não-aninhados, aplicados a PVC's elípticos de segunda ordem, está demonstrada para malhas quasi-uniformes [109], não-quasi-uniformes [110] e com polinômios de interpolação de qualquer ordem [94]. Resultados semelhantes são alcançados em [27] para o caso de algoritmos simétricos, ou seja, tem-se um mesmo número de pré e pós-relaxações em cada nível.

Nesta seção, apresenta-se um breve resumo, baseado em [109], da convergência em malhas quasi-uniformes para problemas bidimensionais num domínio poligonal e limitado Ω e condições de contorno de Dirichlet em Γ.

A aproximação por elementos finitos em forma variacional é dada por: *encontrar* $u_k \in V_k$ *tal que*,

$$a(u_k, v) = (f, v) \qquad \forall v \in V_k \qquad k = 1, 2 \ldots \tag{4.42}$$

onde $V_k = \{v \in C_0(\Omega) \mid v|_K \text{ é linear } \forall K \in \mathcal{T}_k\}$ e $\{\mathcal{T}_k, k = 1, 2, \ldots\}$ uma família de triangulações em Ω.

Dado um elemento $K \in \mathcal{T}_k$, denota-se por h_K o seu diâmetro, definindo-se $h_k = \max_{K \in \mathcal{T}_k} h_K$ e $\delta = \min_{K \in \mathcal{T}_k} h_K / h_k$. Logo, a constante δ é uma medida da uniformidade da malha $\mathcal{T}_k$. Assume-se que todas as triangulações são quasi-uniformes, ou seja,

$$h_K > \alpha_0 h_k \qquad \alpha_K > \alpha_0 \tag{4.43}$$

sendo α_0 uma constante positiva e α_K é o menor ângulo de K.

Assume-se que as malhas são não-aninhadas, ou seja, $\mathcal{T}_{k-1} \not\subset \mathcal{T}_k$. Além disso, $\mathcal{T}_k$ pode ser coberta de forma finita por $\mathcal{T}_{k-1}$ e $\mathcal{T}_{k+1}$. Logo [109],

$$\begin{aligned} &\sup_{K \in \mathcal{T}_k} \{\text{cardinalidade}(\{K' \in \mathcal{T}_{k-1} \mid K' \cap K \neq \emptyset\})\} \leq \beta_0 \qquad k = 2, 3, \ldots \\ &\sup_{K \in \mathcal{T}_k} \{\text{cardinalidade}(\{K' \in \mathcal{T}_{k+1} \mid K' \cap K \neq \emptyset\})\} \leq \beta_0 \qquad k = 1, 2, \ldots \end{aligned} \tag{4.44}$$

A seguinte condição é imposta,

$$\alpha_1 N_k \leq N_{k+1} \qquad \alpha_2^{-1} h_k \leq h_{k+1} \leq \alpha_2 h_k \qquad k = 1, 2, \ldots \tag{4.45}$$

para constantes $\alpha_1 > 2$ e $\alpha_2 \geq 1$ sendo $N_k = \dim(V_k) \simeq h_k^{-2}$. Para malhas aninhadas, tem-se $\beta_0 = 4$, $\alpha_1 \sim 4$ e $\alpha_2 \sim 2$.

Como operadores de transferência entre malhas, emprega-se o interpolante nodal padrão definido na equação (4.28). Finalmente, impõem-se ainda condições de regularidade para a solução u do problema [109].

Define-se, então, um procedimento multigrid constituído de dois processos iterativos. No primeiro, tem-se iterações aninhadas para se obter aproximações sequenciais para (4.42) da forma $\tilde{u}_k \simeq u_k$ com $k = 1, 2, \ldots$. Para resolver (4.42) em cada nível k, toma-se $\tilde{u}_{k-1}$ como solução inicial e emprega-se recursivamente esquemas de correção para determinar $\tilde{u}_k$. A equação do resíduo em forma variacional é denotada como,

$$a(u_k, v) = \tilde{F}(v) \qquad \forall v \in V_k \tag{4.46}$$

sendo $\tilde{F}$ um funcional linear em V_k, o qual pode ser representado em $(V_k, (\cdot, \cdot))$ como $\tilde{f}$: $\tilde{F}(v) = (\tilde{f}, v)$ $\forall v \in V_k$.

De maneira formal, os esquemas multigrid apresentados na seção 4.4 podem ser descritos como [109]:

Definição : (Esquema no nível k)

1. Para $k = 1$, resolver (4.42) ou (4.46) empregando qualquer método:

$$a(\tilde{u}_1, v) = (f, v) \quad \text{ou} \quad a(\tilde{u}_1, v) = \tilde{F}(v) \quad \forall v \in V_1 \tag{4.47}$$

2. Para $k > 1$, obtém-se w_{m+1} a partir de uma aproximação inicial w_0, aplicando-se primeiramente m passos de relaxações do tipo Jacobi, isto é,

$$(w_l - w_{l-1}, v) = \Lambda_k^{-1}(F(v) - a(w_{l-1}, v)) \quad \forall v \in V_k \quad l = 1, 2, \ldots, m \tag{4.48}$$

onde Λ_k é o maior autovalor associado a forma $a(\cdot, \cdot)$ em V_k.
Aplicar, então, um esquema de correção de malha grossa: encontrar $\bar{e} \in V_{k-1}$ tal que,

$$a(\bar{e}, v) = F(I_k v) - a(w_m, I_k v) = \tilde{F}(v) \quad \forall v \in V_{k-1} \tag{4.49}$$

Seja e a aproximação para $\bar{e}$ obtida aplicando-se ν_0 iterações do esquema no nível $k-1$ à equação do resíduo com aproximação inicial nula. Finalmente, chega-se a w_{m+1} com,

$$w_{m+1} = w_m + I_k e \tag{4.50}$$

Na prática, a matriz de massa consistente em (4.48) pode ser substituída por uma matriz diagonal, empregando-se por exemplo técnicas de integração nodais, simplificando o esquema de relaxação. Tal modificação é expressa como [109],

$$b_k(w_l - w_{l-1}, v) = \tilde{\Lambda}_k^{-1}(F(v) - a(w_{l-1}, v)) \quad \forall v \in V_k \quad l = 1, 2, \ldots, m \tag{4.51}$$

A partir das hipóteses apresentadas, da definição anterior e de alguns resultados preliminares, demonstra-se em [109] que os seguintes teoremas, garantindo a convergência e a ótima ordem do algoritmo proposto, são válidos.

Teorema 1 : Suponha que as condições (4.43) a (4.45) e algumas desigualdades presentes em [109] sejam válidas. Considere ainda $\nu_0 > 1$. Logo, existe uma constante $0 < \gamma < 1$ e um inteiro $m \geq 1$, independentes do nível k, tal que se,

$$||\bar{e} - e|| \leq \gamma^{\nu_0} ||\bar{e}||$$

então,

$$||u_k - w_{m+1}|| \leq \gamma ||u_k - u_0||$$

onde $||u|| = \sqrt{a(u,u)}$ é a norma de energia.

Teorema 2 : Assumindo válidas as condições do Teorema 1, considere $2 \leq \nu_0 < \alpha_1$. Logo, existem constantes r e $\delta > 0$ tal que se

$$||u_1 - \hat{u}_1|| \leq \delta C h_1$$

é válido para alguma constante C, então,

1. $||u_k - \hat{u}_k|| \leq \delta C h_k \quad k \geq 2$
2. $||u - \hat{u}_k|| \leq (1 + \delta) C h_k \quad k \geq 1$
3. o custo para calcular $\hat{u}_k$ é limitado por $C_0 N_k$, onde C_0 é independente do número do nível k.

Neste caso, $\hat{u}_k$ é obtido executando-se r vezes o esquema do nível k, empregando-se relaxações simplificadas dadas por (4.51) com solução inicial $\hat{u}_{k-1}$, sendo $\hat{u}_1$ calculado resolvendo-se (4.42) de forma direta.

4.7 Aplicações a Problemas Bidimensionais

Visando avaliar o desempenho dos algoritmos multigrid, tomaram-se inicialmente vários exemplos planos apresentados a seguir. Primeiramente, tem-se uma chapa submetida à tração com malhas aninhadas e não-aninhadas. Posteriormente, consideram-se os casos planos do Capítulo 2, especificamente a placa com furo e o problema de fratura, assim como a aplicação do algoritmo adaptável para estes exemplos. Todos os problemas analisados foram modelados empregando malhas com triângulos lineares. Procurou-se realizar uma comparação com os métodos direto de Gauss e iterativo GCGS discutidos anteriormente. Observa-se que em todos os exemplos analisados, considerou-se armazenamento em matriz esparsa. Para o método GCGS e algoritmos multigrid, utilizou-se o critério de convergência (2.129) na norma euclidiana com $\xi = 10^{-4}$. Nos esquemas multigrid, as relaxações foram efetuadas pelo método de Gauss-Seidel.

Para comparar as soluções obtidas pelos métodos de Gauss ($\mathbf{u}_{dir}$), GCGS ($\mathbf{u}_{ite}$) e multigrid ($\mathbf{u}_{mg}$), consideram-se os seguintes erros relativos na norma euclidiana análogos a expressão (2.130),

$$||e_r^{dir/ite}||_2 = \frac{||\mathbf{u}_{dir} - \mathbf{u}_{ite}||_2}{||\mathbf{u}_{dir}||_2} \qquad ||e_r^{dir/mg}||_2 = \frac{||\mathbf{u}_{dir} - \mathbf{u}_{mg}||_2}{||\mathbf{u}_{dir}||_2} \qquad ||e_r^{ite/mg}||_2 = \frac{||\mathbf{u}_{ite} - \mathbf{u}_{mg}||_2}{||\mathbf{u}_{ite}||_2} \quad (4.52)$$

4.7.1 Chapa submetida a tração

A Figura 4.13 ilustra uma chapa submetida a uma carga distribuída, estando rigidamente apoiada num ponto e com deslocamento restrito apenas na direção x nos demais nós. Considera-se este problema como estado plano de tensão com espessura unitária. Para aplicar as técnicas multigrid, foram geradas 4 malhas aninhadas e mais 4 não-aninhadas apresentadas, respectivamente, nas Figuras 4.14 e 4.15. As principais características destas malhas estão dadas na Tabela 4.1: números de nós, elementos e equações; número médio de elementos em cada linha e número total de coeficientes na matriz, respectivamente, para método direto ($\text{NElems}_{esp}^{dir}$, m_{esp}^{dir}) e iterativo/multigrid ($\text{NElems}_{esp}^{ite}$, m_{esp}^{ite}).

Observa-se que como o coeficiente de Poisson é nulo, este caso é equivalente a um problema unidimensional com solução exata dentro do espaço de aproximação obtido com elementos finitos lineares.

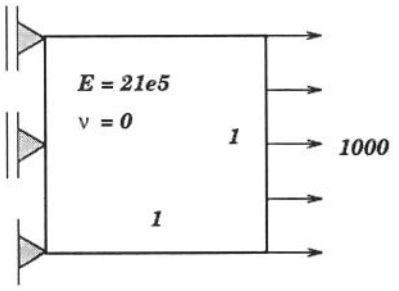


Figura 4.13: Chapa com carregamento distribuído.

Por sua vez, as Tabelas 4.2 e 4.3 apresentam para estes exemplos o número de iterações equivalentes de malha fina para os métodos de Gauss, GCGS e várias estratégias multigrid, indicando neste caso o número total de ciclos realizados e a quantidade de pré e pós-relaxações. Tem-se ainda os erros relativos dados pelas expressão (4.52). Os resultados em termos do número de operações em MFlop e espaço de memória em Kbytes estão ilustrados nas Figuras 4.16 e 4.17, estando os coeficientes angulares das retas ajustadas dados entre colchetes. Observa-se que no caso de multigrid, tomaram-se sucessivamente 1, 2, 3 e 4 malhas, obtendo-se os pontos e as retas ilustradas nas Figuras 4.16 e 4.17.

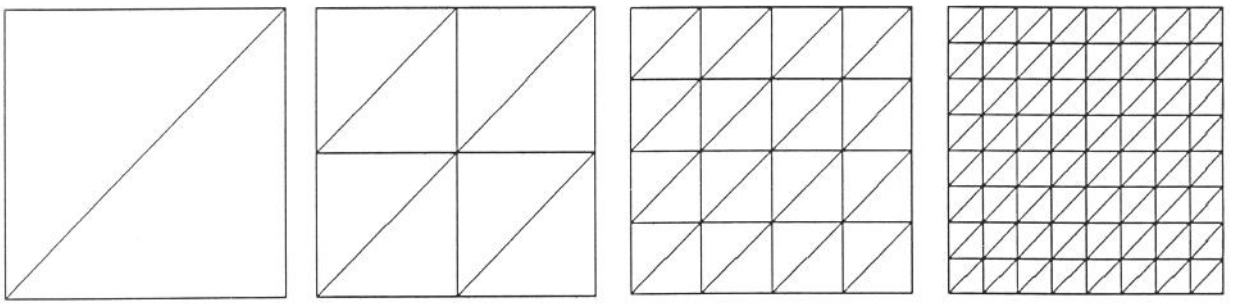


Figura 4.14: Malhas aninhadas geradas para a chapa em tração.

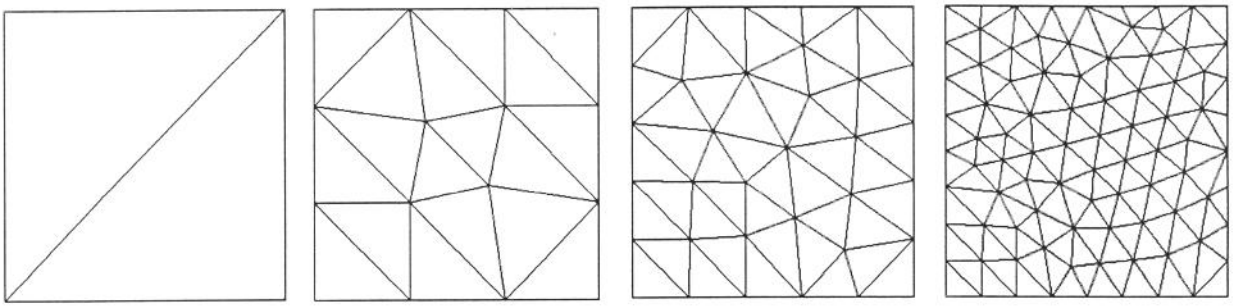


Figura 4.15: Malhas não-aninhadas geradas para a chapa em tração.

Aninhadas							
Malha	Nós	Elementos	Equações	$NElems_{esp}^{dir}$	m_{esp}^{dir}	$NElems_{esp}^{ite}$	m_{esp}^{ite}
1	4	2	5	13	2,6	13	2,6
2	9	8	14	73	5,2	63	4,5
3	25	32	44	357	8,1	253	5,8
4	81	128	152	2183	14,4	993	6,5

Não-aninhadas							
Malha	Nós	Elementos	Equações	$NElems_{esp}^{dir}$	m_{esp}^{dir}	$NElems_{esp}^{ite}$	m_{esp}^{ite}
1	4	2	5	13	2,6	13	2,6
2	16	18	27	173	6,4	145	5,4
3	35	48	63	548	8,7	380	6,0
4	89	144	168	2223	13,2	1113	6,6

Tabela 4.1: Atributos das malhas para a chapa em tração.

A solução na primeira malha foi obtida de forma direta, enquanto nos demais casos empregou-se a mesma estratégia multigrid indicada nas Figuras 4.16 e 4.17.

Método	NIT	Ciclos, ν_1, ν_2	$\|\|e_r^{dir/ite}\|\|_2$	$\|\|e_r^{dir/mg}\|\|_2$	$\|\|e_r^{ite/mg}\|\|_2$
Gauss	12	–	–	–	–
GCGS	58	–	$1,21 \times 10^{-5}$	–	–
V	93	28,1,0	$1,21 \times 10^{-5}$	$1,78 \times 10^{-4}$	$1,82 \times 10^{-4}$
	70	15,1,1	$1,21 \times 10^{-5}$	$4,24 \times 10^{-4}$	$4,29 \times 10^{-4}$
	93	20,2,0	$1,21 \times 10^{-5}$	$1,55 \times 10^{-4}$	$1,60 \times 10^{-4}$
	78	13,2,1	$1,21 \times 10^{-5}$	$3,90 \times 10^{-4}$	$3,95 \times 10^{-4}$
	73	10,2,2	$1,21 \times 10^{-5}$	$8,21 \times 10^{-4}$	$8,26 \times 10^{-4}$
W	59	13,1,0	$1,21 \times 10^{-5}$	$5,88 \times 10^{-5}$	$6,43 \times 10^{-5}$
	44	7,1,1	$1,21 \times 10^{-5}$	$1,01 \times 10^{-4}$	$1,06 \times 10^{-4}$
	57	9,2,0	$1,21 \times 10^{-5}$	$6,26 \times 10^{-5}$	$6,81 \times 10^{-5}$
	48	6,2,1	$1,21 \times 10^{-5}$	$4,31 \times 10^{-5}$	$4,94 \times 10^{-5}$
	49	5,2,2	$1,21 \times 10^{-5}$	$4,65 \times 10^{-5}$	$5,23 \times 10^{-5}$
FMV	5	1,1,0	$1,21 \times 10^{-5}$	$4,33 \times 10^{-15}$	$1,21 \times 10^{-5}$
FMW	6	1,1,0	$1,21 \times 10^{-5}$	$3,57 \times 10^{-15}$	$1,21 \times 10^{-5}$

Tabela 4.2: Resultados para a chapa em tração – malhas aninhadas.

Método	NIT	Ciclos, ν_1, ν_2	$\|\|e_r^{dir/ite}\|\|_2$	$\|\|e_r^{dir/mg}\|\|_2$	$\|\|e_r^{ite/mg}\|\|_2$
Gauss	11	–	–	–	–
GCGS	63	–	$9,85 \times 10^{-6}$	–	–
V	85	23,1,0	$9,85 \times 10^{-6}$	$2,29 \times 10^{-4}$	$2,28 \times 10^{-4}$
	57	11,1,1	$9,85 \times 10^{-6}$	$6,46 \times 10^{-4}$	$6,49 \times 10^{-4}$
	67	13,2,0	$9,85 \times 10^{-6}$	$1,27 \times 10^{-4}$	$1,29 \times 10^{-4}$
	73	11,2,1	$9,85 \times 10^{-6}$	$4,04 \times 10^{-4}$	$4,07 \times 10^{-4}$
	73	9,2,2	$9,85 \times 10^{-6}$	$8,11 \times 10^{-4}$	$8,13 \times 10^{-4}$
W	45	8,1,0	$9,85 \times 10^{-6}$	$3,18 \times 10^{-5}$	$3,32 \times 10^{-5}$
	39	5,1,1	$9,85 \times 10^{-6}$	$6,80 \times 10^{-5}$	$7,18 \times 10^{-5}$
	47	6,2,0	$9,85 \times 10^{-6}$	$1,80 \times 10^{-5}$	$2,32 \times 10^{-5}$
	49	5,2,1	$9,85 \times 10^{-6}$	$5,35 \times 10^{-5}$	$5,72 \times 10^{-5}$
	49	4,2,2	$9,85 \times 10^{-6}$	$9,79 \times 10^{-5}$	$1,02 \times 10^{-4}$
FMV	6	1,1,0	$9,85 \times 10^{-6}$	$6,59 \times 10^{-8}$	$9,87 \times 10^{-6}$
FMW	8	1,1,0	$9,85 \times 10^{-6}$	$3,10 \times 10^{-8}$	$9,86 \times 10^{-6}$

Tabela 4.3: Resultados para a chapa em tração – malhas não-aninhadas.

4.7.2 Placa com furo e problema de fratura

Nesta seção, consideram-se os exemplos de placa com furo e de fratura analisados no Capítulo 2. Empregaram-se elementos lineares nas 4 malhas consideradas, estando as mesmas ilustradas nas Figuras 4.18 e 4.19, respectivamente, para a placa com furo e fratura. Da mesma maneira, os atributos das malhas estão dados na Tabela 4.4.

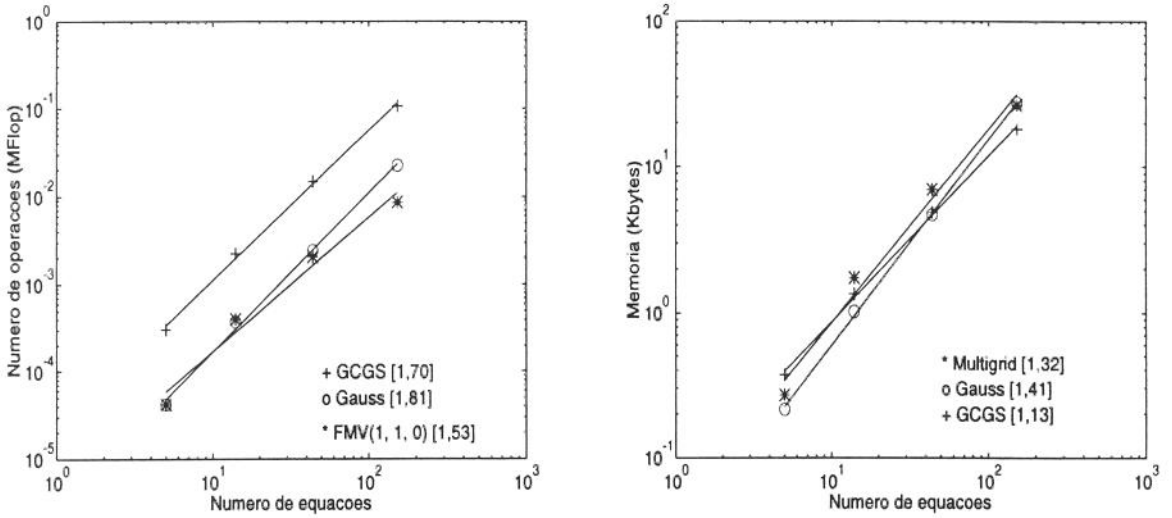


Figura 4.16: Número de operações e espaço de memória para os métodos de Gauss, GCGS e Multigrid na chapa com malhas aninhadas.

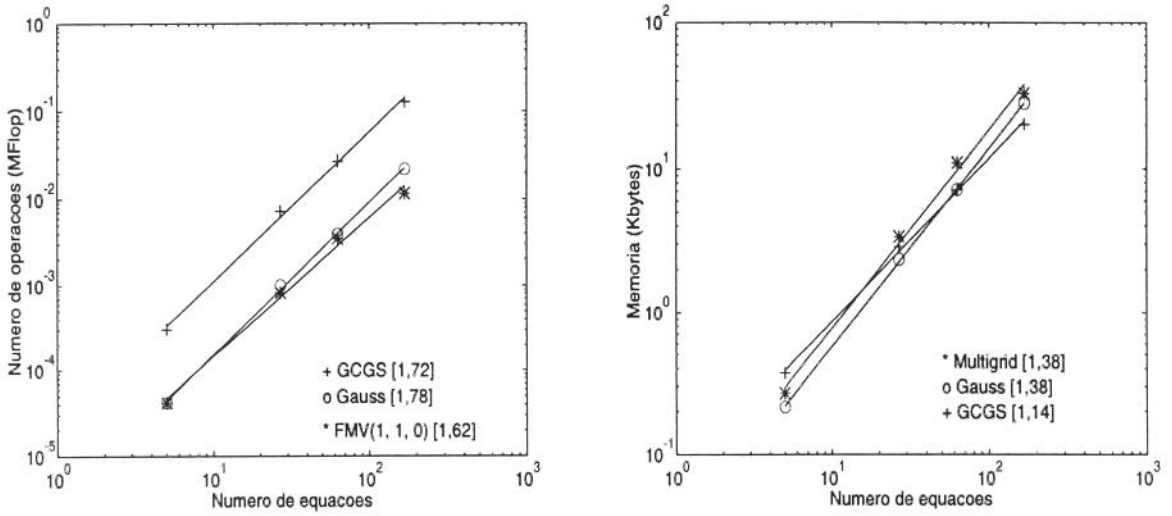


Figura 4.17: Número de operações e espaço de memória para os métodos de Gauss, GCGS e Multigrid na chapa com malhas não-aninhadas.

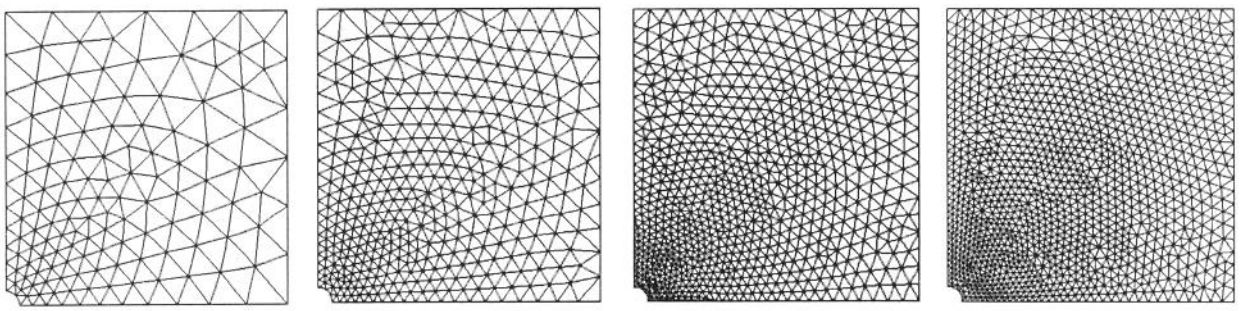


Figura 4.18: Malhas com elementos lineares para a placa com furo.

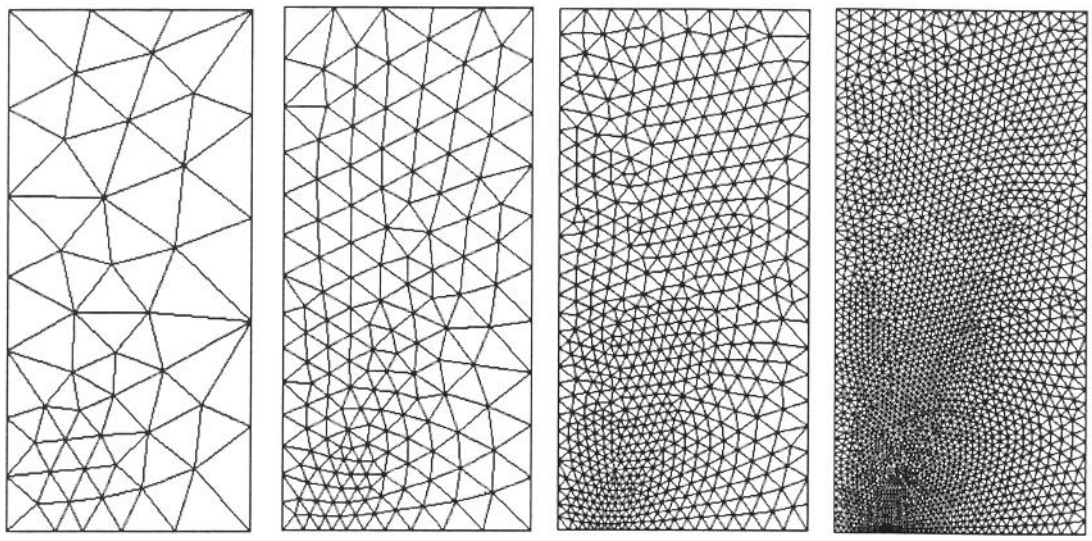


Figura 4.19: Malhas com elementos lineares para o problema de fratura.

Placa com furo							
Malha	Nós	Elementos	Equações	$\text{NElems}_{esp}^{dir}$	m_{esp}^{dir}	$\text{NElems}_{esp}^{ite}$	m_{esp}^{ite}
1	38	57	64	593	9,3	381	6,0
2	121	206	220	3504	15,9	1471	6,7
3	445	817	846	21620	25,6	5985	7,1
4	1679	3215	3272	126794	38,8	23833	7,3

Fratura							
Malha	Nós	Elementos	Equações	$\text{NElems}_{esp}^{dir}$	m_{esp}^{dir}	$\text{NElems}_{esp}^{ite}$	m_{esp}^{ite}
1	51	80	89	911	10,2	555	6,2
2	170	299	317	5607	17,7	2171	6,8
3	626	1171	1207	32422	26,9	8647	7,2
4	2383	4608	4679	204249	43,7	34312	7,3

Tabela 4.4: Atributos das malhas para os problemas de placa com furo e fratura.

Analogamente, ao exemplo anterior as Tabelas 4.5 e 4.6 apresentam os resultados da aplicação dos métodos direto de Gauss, iterativo GCGS e multigrid a estes dois problemas. Da mesma maneira, os resultados em termos do número de operações em MFlop e espaço de memória em Kbytes estão ilustrados nas Figuras 4.20 e 4.21. Observa-se que não se utilizou nenhum estimador de erro para refinar as malhas, apenas reduzindo-se pela metade os tamanhos dos elementos associados a malha de controle [54].

Método	NIT	Ciclos, ν_0, ν_1, ν_2	$\|\|e_r^{dir/ite}\|\|_2$	$\|\|e_r^{dir/mg}\|\|_2$	$\|\|e_r^{ite/mg}\|\|_2$
Gauss	99	–	–	–	–
GCGS	208	–	$8,72 \times 10^{-6}$	–	–
FMV	34	4,1,1,1	$8,72 \times 10^{-6}$	$4,71 \times 10^{-6}$	$1,01 \times 10^{-5}$
	32	3,1,2,1	$8,72 \times 10^{-6}$	$1,12 \times 10^{-5}$	$1,47 \times 10^{-5}$
	42	3,2,1,1	$8,72 \times 10^{-6}$	$1,06 \times 10^{-6}$	$8,85 \times 10^{-6}$
	35	2,2,2,1	$8,72 \times 10^{-6}$	$1,09 \times 10^{-5}$	$1,44 \times 10^{-5}$
FMVV	38	8,1,1,1	$8,72 \times 10^{-6}$	$2,05 \times 10^{-5}$	$2,28 \times 10^{-5}$
	37	6,1,2,1	$8,72 \times 10^{-6}$	$2,82 \times 10^{-5}$	$3,01 \times 10^{-5}$
	35	7,2,1,1	$8,72 \times 10^{-6}$	$2,55 \times 10^{-5}$	$2,74 \times 10^{-5}$
	39	6,2,2,1	$8,72 \times 10^{-6}$	$1,74 \times 10^{-5}$	$2,00 \times 10^{-5}$
FMW	41	4,1,1,1	$8,72 \times 10^{-6}$	$2,52 \times 10^{-7}$	$8,72 \times 10^{-6}$
	39	3,1,2,1	$8,72 \times 10^{-6}$	$3,97 \times 10^{-7}$	$8,73 \times 10^{-6}$
	34	2,2,1,1	$8,72 \times 10^{-6}$	$1,12 \times 10^{-6}$	$8,80 \times 10^{-6}$
	44	2,2,2,1	$8,72 \times 10^{-6}$	$2,19 \times 10^{-8}$	$8,73 \times 10^{-6}$

Tabela 4.5: Resultados para a placa com furo.

Método	NIT	Ciclos, ν_0, ν_1, ν_2	$\|\|e_r^{dir/ite}\|\|_2$	$\|\|e_r^{dir/mg}\|\|_2$	$\|\|e_r^{ite/mg}\|\|_2$
Gauss	130	–	–	–	–
GCGS	273	–	$1,37 \times 10^{-5}$	–	–
FMV	43	5,1,1,1	$1,37 \times 10^{-5}$	$3,09 \times 10^{-6}$	$1,51 \times 10^{-5}$
	39	4,1,2,1	$1,37 \times 10^{-5}$	$8,76 \times 10^{-6}$	$1,86 \times 10^{-5}$
	42	3,2,1,1	$1,37 \times 10^{-5}$	$4,45 \times 10^{-6}$	$1,58 \times 10^{-5}$
	47	3,2,2,1	$1,37 \times 10^{-5}$	$2,78 \times 10^{-6}$	$1,49 \times 10^{-5}$
FMVV	43	9,1,1,1	$1,37 \times 10^{-5}$	$2,56 \times 10^{-5}$	$3,30 \times 10^{-5}$
	44	9,1,2,1	$1,37 \times 10^{-5}$	$2,08 \times 10^{-5}$	$2,86 \times 10^{-5}$
	44	9,2,1,1	$1,37 \times 10^{-5}$	$1,40 \times 10^{-5}$	$2,28 \times 10^{-5}$
	41	8,2,2,1	$1,37 \times 10^{-5}$	$2,42 \times 10^{-5}$	$3,17 \times 10^{-5}$
FMW	41	4,1,1,1	$1,37 \times 10^{-5}$	$1,77 \times 10^{-6}$	$1,44 \times 10^{-5}$
	38	3,1,2,1	$1,37 \times 10^{-5}$	$3,43 \times 10^{-6}$	$1,52 \times 10^{-5}$
	54	3,2,1,1	$1,37 \times 10^{-5}$	$2,22 \times 10^{-7}$	$1,38 \times 10^{-5}$
	43	2,2,2,1	$1,37 \times 10^{-5}$	$1,69 \times 10^{-6}$	$1,44 \times 10^{-5}$

Tabela 4.6: Resultados para o problema de fratura.

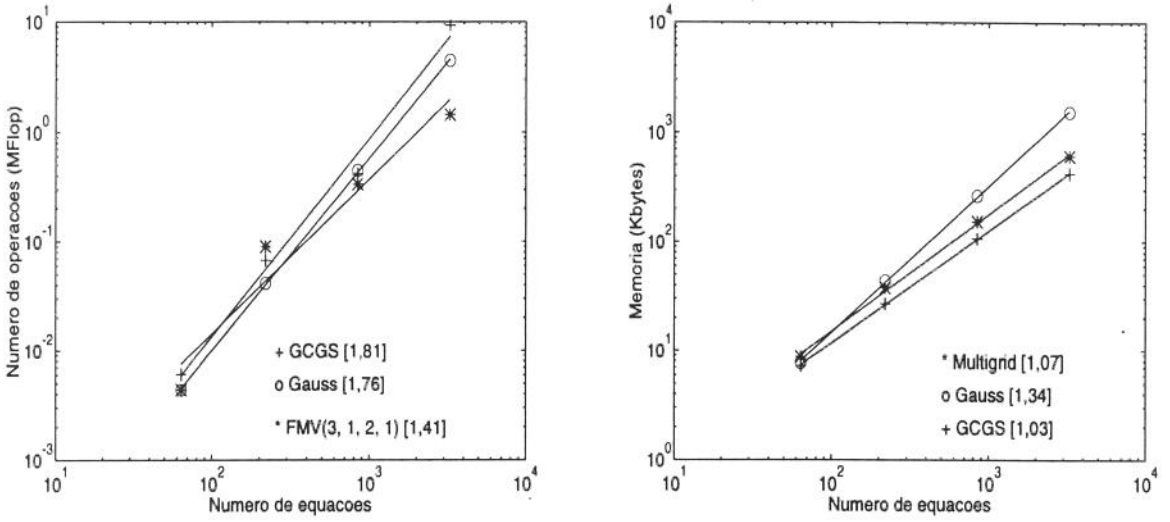


Figura 4.20: Número de operações e espaço de memória para os métodos de Gauss, GCGS e Multigrid para a placa com furo.

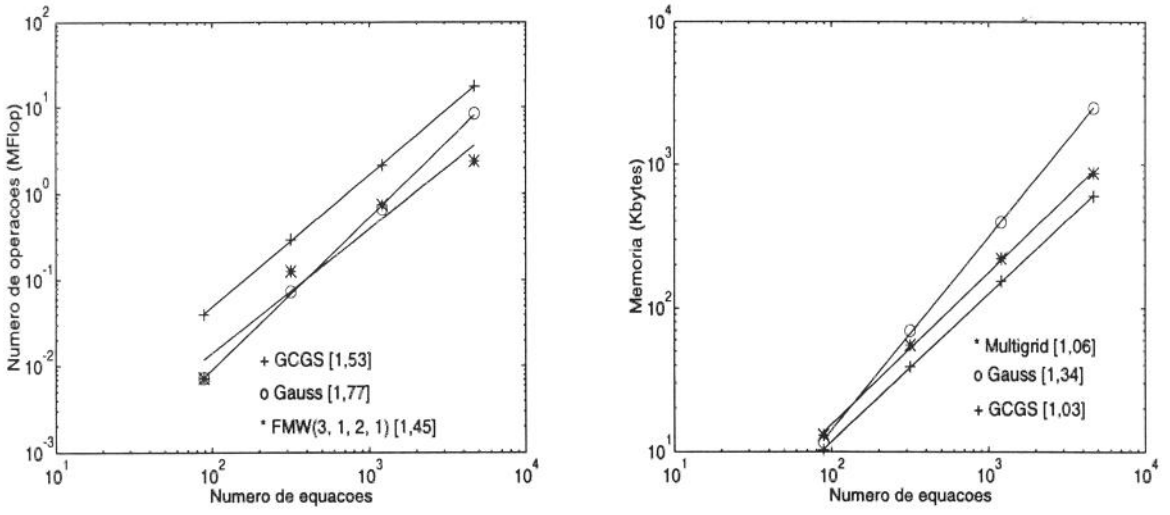


Figura 4.21: Número de operações e espaço de memória para os métodos de Gauss, GCGS e Multigrid para o problema de fratura.

4.7.3 Aplicação do procedimento adaptável

A seguir apresentam-se os resultados da aplicação do algoritmo multigrid adaptável discutido anteriormente aos problemas de placa com furo e fratura. Como o critério de refinamento das malhas é distinto em relação àquele empregado no método de Gauss ou GCGS, comparam-se as performances das estratégias multigrid com estes métodos para as diferentes sequências de malhas geradas, de tal forma que ao final da análise tenha-se um erro estimado próximo do valor $\bar{\eta}$ especificado, partindo-se sempre da mesma malha grossa.

Placa com furo

As Figuras 4.22 e 4.23 ilustram, respectivamente, as sequências de malhas obtidas para a solução do exemplo de placa com furo com $\bar{\eta} = 1\%$ através de análises adaptáveis, empregando método de Gauss ou GCGS e via o procedimento multigrid adaptável apresentado na seção 4.5.1. As características destas malhas, incluindo o erro estimado η, estão dadas na Tabela 4.7. Ressalta-se que para o caso de multigrid, o valor de η indicado é aquele obtido ao final do processo de geração das malhas. O valor final após atingir a precisão de $\xi = 10^{-4}$ é igual a $\eta = 1,1$.

Em ambos procedimentos foram necessárias 4 malhas para se alcançar um erro estimado próximo ao valor de $\bar{\eta}$ dado. Da mesma maneira que nos casos anteriores, tem-se na Tabela 4.8 os resultados da aplicação dos métodos de Gauss, GCGS e algoritmos multigrid. Por sua vez, a Figura 4.24 apresenta gráficos de barra com o número equivalente de iterações na malha fina, operações em MFlop e espaço de memória em Kbytes.

Observa-se que para multigrid, os números de ciclos indicados na Tabela 4.8 não incluem a iteração FMV com $\nu_0 = 2$, $\nu_1 = 1$, $\nu_2 = 1$ empregada para se determinar a sequência de malhas. Assim, indica-se apenas a quantidade necessária para alcançar a solução na malha fina com precisão $\xi = 10^{-4}$. No entanto, observa-se que o número de iterações equivalente (NIT) leva em conta este ciclo FMV para determinar o custo da solução.

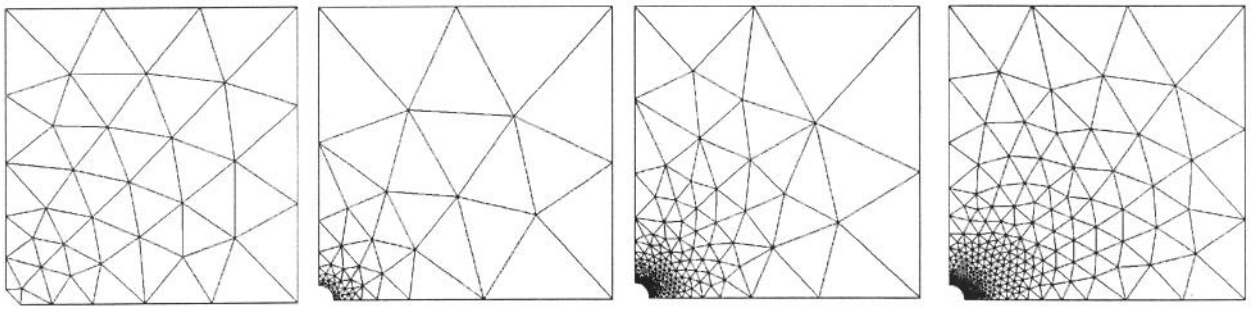


Figura 4.22: Malhas para a placa com furo obtidas por análise adaptável com método de Gauss ou GCGS.

Problema de fratura

De forma análoga ao caso da placa com furo, analisou-se de forma adaptável o problema de fratura. Inicialmente, tomou-se $\bar{\eta} = 2\%$ com precisões $\xi = 10^{-3}$ e $\xi = 10^{-4}$, estando as sequências de malhas obtidas pela resolução por método de Gauss e multigrid dadas nas Figuras 4.25 e 4.26, com as respectivas características indicadas na Tabela 4.9. Observa-se que no caso de multigrid foram necessárias 5 malhas para se obter um erro estimado de 2,3% ao final do processo de geração, alcançando 2,5% após a convergência nas duas precisões utilizadas. Os resultados obtidos com os métodos de Gauss, GCGS e multigrid estão apresentados na Tabela 4.10 e na Figura 4.27.

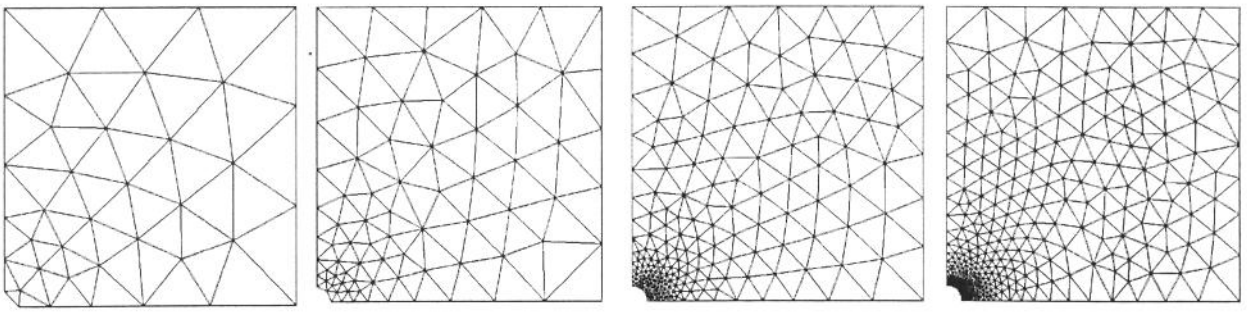


Figura 4.23: Malhas para a placa com furo obtidas através do procedimento multigrid adaptável.

Direto/iterativo adaptável								
Malha	η (%)	Nós	Elementos	Equações	m_{esp}^{dir}	$\text{NElems}_{esp}^{dir}$	m_{esp}^{ite}	$\text{NElems}_{esp}^{ite}$
1	2,9	38	57	64	9,3	593	6,0	381
2	2,6	62	95	106	10,4	1104	6,1	650
3	1,4	231	414	434	19,6	8514	6,9	3003
4	1,1	379	694	723	21,7	15709	7,0	5094

Multigrid adaptável								
Malha	η (%)	Nós	Elementos	Equações	m_{esp}^{dir}	$\text{NElems}_{esp}^{dir}$	m_{esp}^{ite}	$\text{NElems}_{esp}^{ite}$
1	2,9	38	57	64	9,3	593	6,0	381
2	3,1	80	129	141	12,0	1686	6,4	905
3	1,8	208	368	386	19,0	7313	6,9	2652
4	1,0	438	799	832	22,9	19063	7,0	5848

Tabela 4.7: Atributos das malhas da placa com furo obtidas pelos procedimentos adaptáveis.

Método	NIT	Ciclos, ν_0, ν_1, ν_2	$\|\|e_r^{dir/ite}\|\|_2$	$\|\|e_r^{dir/mg}\|\|_2$	$\|\|e_r^{ite/mg}\|\|_2$
Gauss	36	–	–	–	–
GCGS	157	–	$2,60 \times 10^{-5}$	–	–
FMV	39	1,2,1,1	$2,60 \times 10^{-5}$	$1,40 \times 10^{-4}$	$1,44 \times 10^{-4}$
	43	2,1,1,1	$2,60 \times 10^{-5}$	$7,00 \times 10^{-5}$	$7,65 \times 10^{-5}$
V	46	5,1,1,1	$2,60 \times 10^{-5}$	$1,59 \times 10^{-4}$	$1,63 \times 10^{-4}$
	47	4,1,2,1	$2,60 \times 10^{-5}$	$1,77 \times 10^{-4}$	$1,81 \times 10^{-4}$
W	45	3,1,1,1	$2,60 \times 10^{-5}$	$2,29 \times 10^{-5}$	$3,60 \times 10^{-5}$

Tabela 4.8: Resultados para a placa com furo utilizando procedimento adaptável.

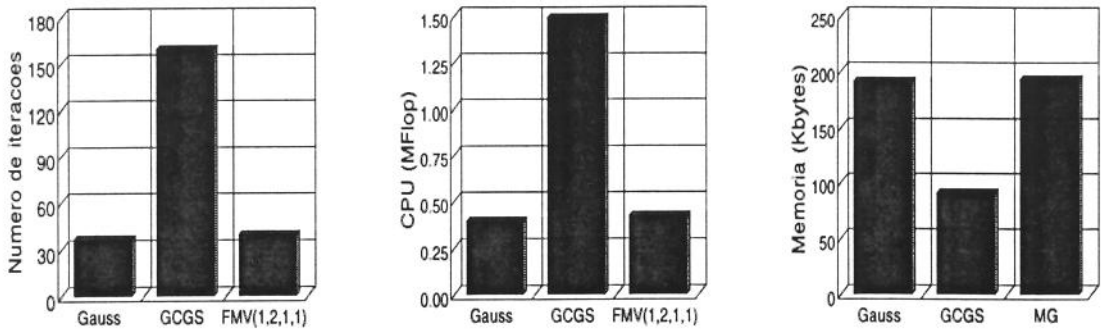


Figura 4.24: Resultados dos procedimentos adaptáveis aplicados ao problema de placa com furo.

Repetiu-se o mesmo procedimento anterior, considerando $\bar{\eta} = 1\%$ e $\xi = 10^{-4}$. Foram necessárias 3 e 6 malhas, respectivamente, para os métodos direto e multigrid, chegando-se ao final com erros estimados de 1,1% e 1,3%. As malhas obtidas estão ilustradas nas Figuras 4.28 e 4.29 com os respectivos parâmetros dados na Tabela 4.11. Os resultados estão apresentados na Tabela 4.12 e na Figura 4.30.

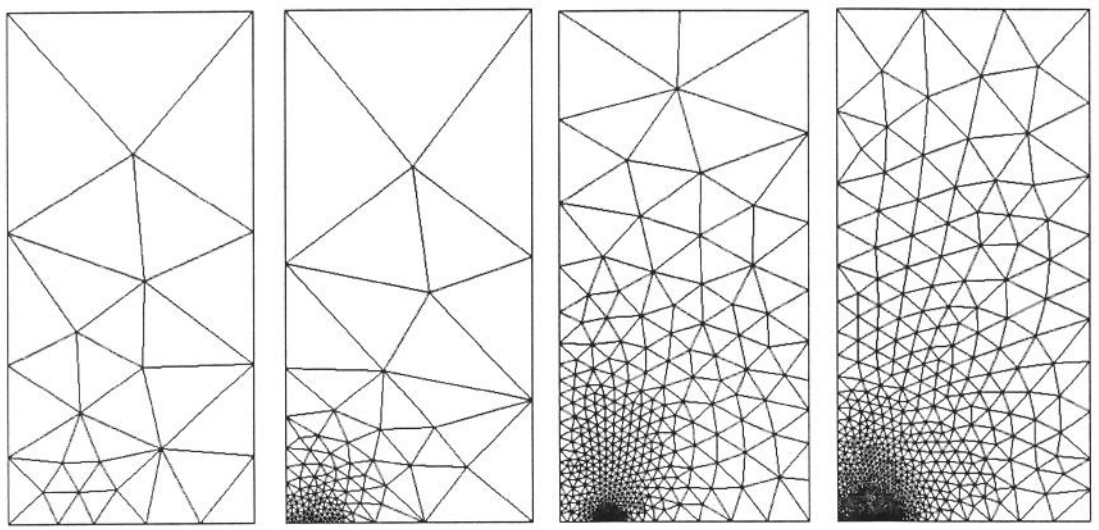


Figura 4.25: Malhas para o problema de fratura obtidas por análise adaptável com método de Gauss utilizando $\bar{\eta} = 2\%$.

4.7.4 Discussão dos resultados

A partir dos resultados obtidos, os seguintes comentários podem ser efetuados:

- de forma geral, as mesmas conclusões obtidas no Capítulo 2 para os métodos de Gauss e GCGS aplicam-se para os problemas aqui analisados.
- apesar do problema de chapa sob tração ser bastante simples, algumas conclusões podem ser mencionadas. Para os ciclos V e W, torna-se interessante empregar uma pós-relaxação, como pode ser observado nas Tabelas 4.2 e 4.3. Apesar de necessitar de um número total de ciclos maior, a estratégia com $\nu_1 = \nu_2 = 1$ parece ser mais adequada. No entanto, os procedimentos

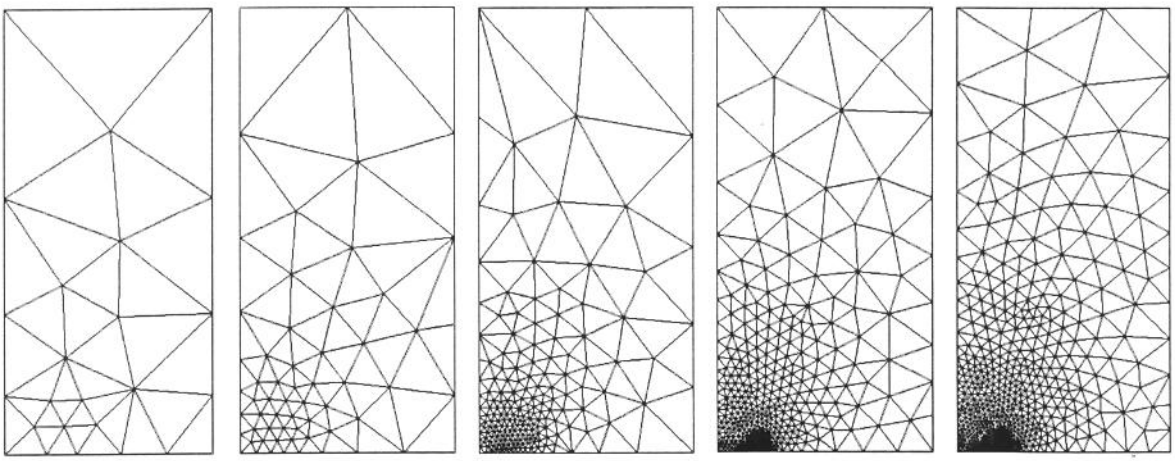


Figura 4.26: Malhas para o problema de fratura obtidas por multigrid adaptável com $\bar{\eta} = 2\%$.

Direto/iterativo adaptável								
Malha	η (%)	Nós	Elementos	Equações	m_{esp}^{dir}	NElems$_{esp}^{dir}$	m_{esp}^{ite}	NElems$_{esp}^{ite}$
1	9,3	25	34	40	7,1	283	5,5	220
2	6,3	115	196	212	14,7	3122	6,7	1413
3	3,0	556	1043	1074	27,5	29495	7,2	7702
4	2,1	968	1818	1869	29,6	55370	7,2	13433

Multigrid adaptável								
Malha	η (%)	Nós	Elementos	Equações	m_{esp}^{dir}	NElems$_{esp}^{dir}$	m_{esp}^{ite}	NElems$_{esp}^{ite}$
1	9,3	25	34	40	7,1	283	5,5	220
2	7,7	61	97	107	11,8	1261	6,3	676
3	4,9	200	359	375	19,4	7263	6,9	2597
4	3,2	461	858	885	27,6	24386	7,1	6310
5	2,3	883	1672	1714	30,9	52930	7,2	12381

Tabela 4.9: Atributos das malhas do problema de fratura obtidas pelos procedimentos adaptáveis com $\bar{\eta} = 2\%$.

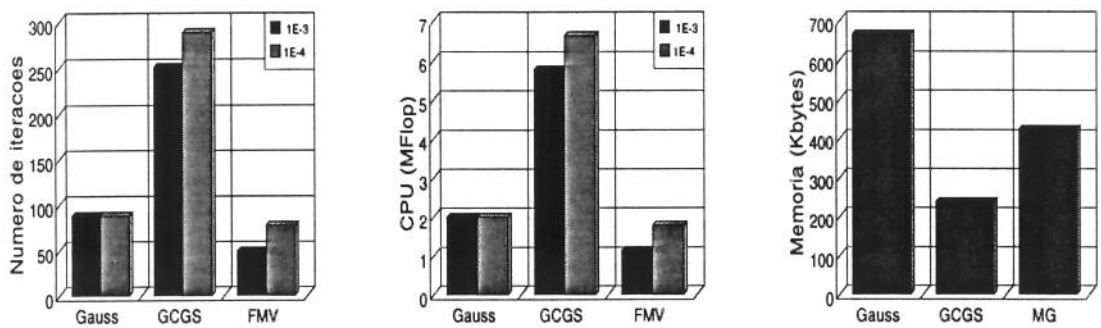


Figura 4.27: Resultados dos procedimentos adaptáveis aplicados ao problema de fratura com $\bar{\eta} = 2\%$.

Precisão $\xi = 10^{-3}$					
Método	NIT	Ciclos, ν_0, ν_1, ν_2	$\|\|e_r^{dir/ite}\|\|_2$	$\|\|e_r^{dir/mg}\|\|_2$	$\|\|e_r^{ite/mg}\|\|_2$
Gauss	88	–	–	–	–
GCGS	251	–	$6,59 \times 10^{-4}$	–	–
FMV	69	2,2,1,1	$6,59 \times 10^{-4}$	$1,96 \times 10^{-4}$	$6,15 \times 10^{-4}$
	52	1,2,2,1	$6,59 \times 10^{-4}$	$7,18 \times 10^{-4}$	$7,95 \times 10^{-4}$
	49	2,1,1,1	$6,59 \times 10^{-4}$	$8,05 \times 10^{-4}$	$8,28 \times 10^{-4}$
	58	2,1,2,1	$6,59 \times 10^{-4}$	$3,38 \times 10^{-4}$	$6,26 \times 10^{-4}$
V	45	4,1,1,1	$6,59 \times 10^{-4}$	$1,89 \times 10^{-3}$	$1,78 \times 10^{-3}$
	44	3,1,2,1	$6,59 \times 10^{-4}$	$1,96 \times 10^{-4}$	$1,87 \times 10^{-3}$

Precisão $\xi = 10^{-4}$					
Método	NIT	Ciclos, ν_0, ν_1, ν_2	$\|\|e_r^{dir/ite}\|\|_2$	$\|\|e_r^{dir/mg}\|\|_2$	$\|\|e_r^{ite/mg}\|\|_2$
Gauss	88	–	–	–	–
GCGS	289	–	$3,24 \times 10^{-5}$	–	–
FMV	93	3,2,1,1	$3,24 \times 10^{-5}$	$3,34 \times 10^{-5}$	$4,45 \times 10^{-5}$
	84	2,2,2,1	$3,24 \times 10^{-5}$	$6,03 \times 10^{-5}$	$6,57 \times 10^{-5}$
	92	5,1,1,1	$3,24 \times 10^{-5}$	$2,93 \times 10^{-5}$	$4,19 \times 10^{-5}$
	77	3,1,2,1	$3,24 \times 10^{-5}$	$6,95 \times 10^{-5}$	$7,39 \times 10^{-5}$
V	76	9,1,1,1	$3,24 \times 10^{-5}$	$2,13 \times 10^{-4}$	$2,11 \times 10^{-4}$
	76	7,1,2,1	$3,24 \times 10^{-5}$	$1,96 \times 10^{-4}$	$1,94 \times 10^{-4}$

Tabela 4.10: Resultados para o problema de fratura aplicando procedimento adaptável com $\bar{\eta} = 2\%$.

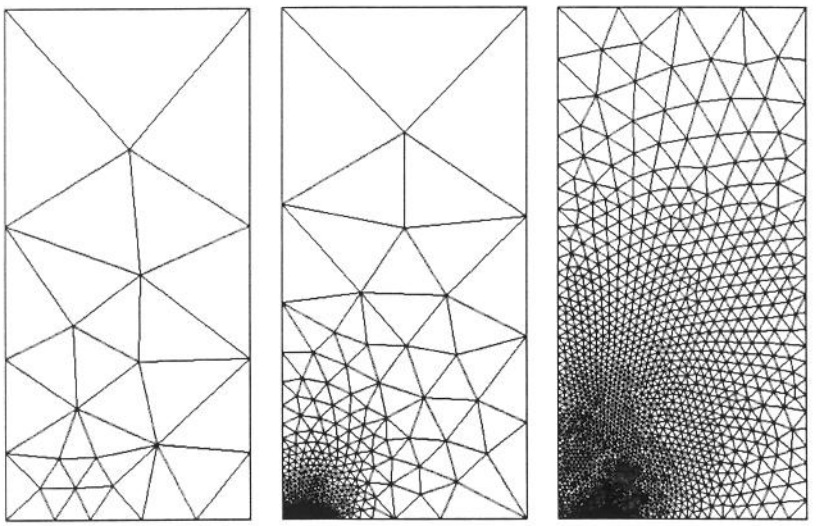


Figura 4.28: Malhas para o problema de fratura obtidas por análise adaptável com método de Gauss utilizando $\bar{\eta} = 1\%$.

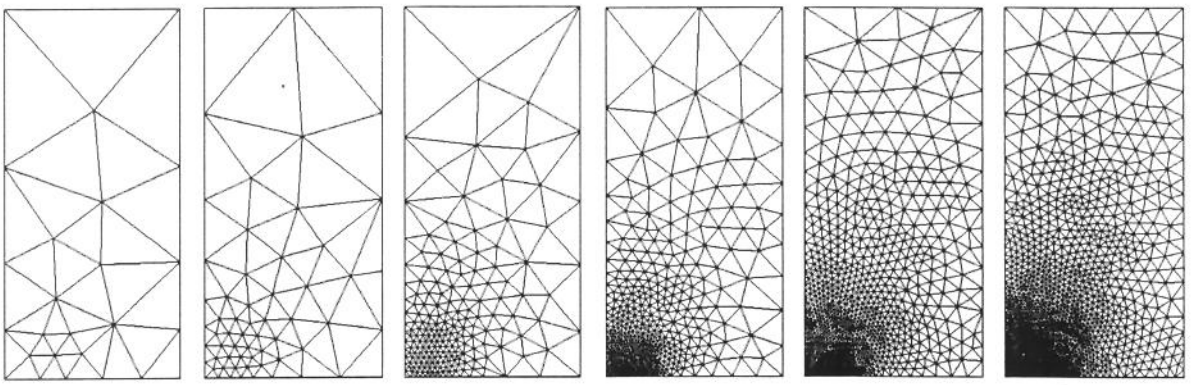


Figura 4.29: Malhas para o problema de fratura obtidas por multigrid adaptável com $\bar{\eta} = 1\%$.

Direto/iterativo adaptável								
Malha	η (%)	Nós	Elementos	Equações	m_{esp}^{dir}	$\text{NElems}_{esp}^{dir}$	m_{esp}^{ite}	$\text{NElems}_{esp}^{ite}$
1	9,3	25	34	40	7,1	283	5,5	220
2	4,2	404	747	775	25,9	20062	7,1	5503
3	1,3	3772	7350	7440	49,7	369737	7,4	54836

Multigrid adaptável								
Malha	η (%)	Nós	Elementos	Equações	m_{esp}^{dir}	$\text{NElems}_{esp}^{dir}$	m_{esp}^{ite}	$\text{NElems}_{esp}^{ite}$
1	9,3	25	34	40	7,1	283	5,5	220
2	7,7	61	97	107	11,8	1261	6,3	676
3	5,6	198	358	371	19,9	7393	6,9	2575
4	3,8	592	1119	1144	29,4	33666	7,2	8228
5	2,2	1681	3240	3293	40,3	132808	7,3	24076
6	1,4	3369	6560	6641	46,7	310313	7,4	48904

Tabela 4.11: Atributos das malhas para o problema de fratura obtidas pelos procedimentos adaptáveis com $\bar{\eta} = 1\%$.

Método	NIT	Ciclos, ν_0, ν_1, ν_2	$\|\|e_r^{dir/ite}\|\|_2$	$\|\|e_r^{dir/mg}\|\|_2$	$\|\|e_r^{ite/mg}\|\|_2$
Gauss	207	–	–	–	–
GCGS	409	–	$3,24 \times 10^{-5}$	–	–
FMV	89	4,2,1,1	$3,24 \times 10^{-5}$	$3,34 \times 10^{-5}$	$4,45 \times 10^{-5}$
	80	2,2,2,1	$3,24 \times 10^{-5}$	$6,03 \times 10^{-5}$	$6,57 \times 10^{-5}$
	88	5,1,1,1	$3,24 \times 10^{-5}$	$2,93 \times 10^{-5}$	$4,19 \times 10^{-5}$
	74	3,1,2,1	$3,24 \times 10^{-5}$	$6,95 \times 10^{-5}$	$7,39 \times 10^{-5}$
V	73	9,1,1,1	$3,24 \times 10^{-5}$	$2,13 \times 10^{-4}$	$2,11 \times 10^{-4}$
	73	7,1,2,1	$3,24 \times 10^{-5}$	$1,96 \times 10^{-4}$	$1,94 \times 10^{-4}$

Tabela 4.12: Resultados para o problema de fratura aplicando procedimento adaptável com $\bar{\eta} = 1\%$.

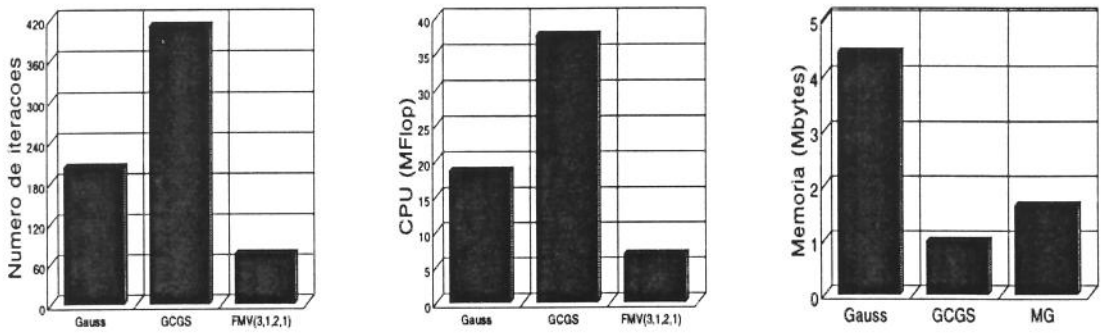


Figura 4.30: Resultados dos procedimentos adaptáveis aplicados ao problema de fratura com $\bar{\eta} = 1\%$.

FMV e FMW, combinando iterações aninhadas com correção de malha grossa, são realmente superiores aos algoritmos V e W, além do que o condicionamento numérico das soluções, medido pelo erros relativos dados em (4.52), é bem melhor que as estratégias V e W. Apenas um ciclo FMV ou FMW com $\nu_1 = 1$ e $\nu_2 = 0$ é necessário na solução deste problema.

O número de operações executados para os algoritmos FMV e FMW é bem inferior em relação aos métodos de Gauss e GCGS, como pode ser visto nas Tabelas 4.2 e 4.3, comparando-se o número equivalente de iterações de malha fina, ou nas Figuras 4.16 e 4.17. No que se refere a demanda por memória, como esperado o método GCGS requer menos espaço, enquanto que as estratégias multigrid e Gauss possuem comportamentos semelhantes para este caso como ilustrado nas Figuras 4.16 e 4.17. Desta forma, mesmo para um exemplo de ordem bem modesta, algumas técnicas multigrid mostram-se superiores aos procedimentos de Gauss e GCGS.

- nos casos da placa com furo e fratura, a partir das Tabelas 4.5 e 4.6, verifica-se que os procedimentos FMV, FMVV e FMW possuem um comportamento semelhante entre si, sendo as performances bem superiores aos métodos de Gauss e GCGS, tomando-se o número equivalente de iterações de malha fina, assim como as Figuras 4.20 e 4.21. No entanto, no caso de FMVV tem-se um número de ciclos maior para se atingir um mesmo condicionamento numérico. Em relação ao número de pré-relaxações, tanto $\nu_1 = 1$ como $\nu_1 = 2$ mostraram-se razoáveis, observando-se que para este último caso foi possível reduzir, na maioria dos casos, o número total de ciclos. Esta mesma observação se aplica ao se utilizar $\nu_0 = 2$. No que se refere ao espaço de memória, as estratégias multigrid são, nestes casos, superiores ao método de Gauss, pois demandam menos memória como pode ser visto nas Figuras 4.20 e 4.21.

- os coeficientes angulares das retas de número de operações para as estratégias multigrid são inferiores em relação àqueles dos procedimentos de Gauss e GCGS. No que se refere ao espaço de memória, os coeficientes de multigrid e GCGS são praticamente iguais, sendo inferiores ao método de Gauss e próximos de 1 nos exemplos de placa com furo e fratura.

- no procedimento adaptável, foram obtidas sequências de malhas aplicando-se um estimador de erro, visando resolver de forma ótima os problemas propostos através dos métodos de Gauss, GCGS e multigrid.

  No caso da placa com furo, observa-se um equilíbrio dos métodos de Gauss e multigrid, tanto em relação ao número de operações como em espaço de memória, assim como no número de malhas necessárias para se atingir um erro estimado em torno de 1%. Logo, o procedimento multigrid adaptável mostra-se viável mesmo neste problema de baixa ordem e de singularidade moderada.

As estratégias multigrid aplicadas são semelhantes no que se refere ao número equivalente de iterações e ao condicionamento numérico. Apenas no caso do ciclo V, tem-se um número total de ciclos ligeiramente maior. Apesar da performance da técnica GCGS, em termos de operações, ser inferior aos demais, a memória requerida é bem menor.

A forte singularidade do problema de fratura implica num maior número de equações para atingir os erros admissíveis de 1% e 2%. Ao contrário do exemplo de placa com furo, neste caso as técnicas multigrid são superiores ao método de Gauss, tanto no número de operações como em espaço de memória, como pode ser visto nas Tabelas 4.10 e 4.12, assim como nas Figuras 4.27 e 4.30. No entanto, devido a limitação (4.17) para alterar o tamanho do elemento, o número de malhas para multigrid é superior em relação ao método de Gauss neste exemplo bem singular.

No que se refere a estratégia FMV adotada para a geração das malhas, alguns testes com $\nu_0 = 1$ foram efetuados, implicando num número total de ciclos maior para atingir a precisão ξ estabelecida. Analogamente, $\nu_0 = 3$ representou um custo elevado ao final do procedimento adaptável. Desta forma, selecionou-se $\nu_0 = 2$ por apresentar uma melhor performance computacional.

Já na parte de solução por multigrid, após obter as malhas, verificam-se comportamentos semelhantes para as estratégias adotadas, observando-se um melhor condicionamento para as soluções baseadas em FMV.

De forma geral, o procedimento multigrid adaptável mostrou-se viável, apresentando performance semelhante ou superior em relação ao método de Gauss mesmo em problemas de pequena ordem, necessitando porém de um número maior de malhas, principalmente em problemas muito singulares. Talvez o principal incoveniente dos métodos multigrid em malhas não-aninhadas é a necessidade de fornecer todas as malhas. O procedimento apresentado permite automatizar a geração das malhas, obtendo ainda uma sequência mais apropriada para alcançar o erro admissível $\bar{\eta}$ especificado.

4.8 Aplicações a Problemas Tridimensionais

Nesta seção, deseja-se estudar o comportamento dos métodos de Gauss, iterativos e multigrid quando aplicados a problemas tridimensionais. Como será visto, as conclusões apresentadas no Capítulo 2 não se extendem para os exemplos analisados a seguir. No entanto, os métodos iterativos e multigrid têm um desempenho semelhante aos resultados obtidos para os casos bidimensionais. Empregaram-se tetraedros lineares em todas as malhas geradas e precisão $\xi = 10^{-4}$.

4.8.1 Viga em balanço

A Figura 4.31 ilustra uma viga em balanço com suas dimensões e o carregamento aplicado, além das quatro malhas geradas com tetraedros lineares, estando as características das mesmas indicadas na Tabela 4.13.

Para a solução deste problema, aplicaram-se os algoritmos iterativos GC, GCD, GCSS e GCGS, além do método de Gauss e estratégias multigrid. A Tabela 4.14 apresenta o número de iterações e a norma euclidiana do resíduo para os métodos iterativos empregados. Já na Tabela 4.15, tem-se o número equivalente de iterações na malha fina para todos os algoritmos utilizados, sendo ainda indicados os erros relativos da expressão (4.52). As Figuras 4.32 a 4.34 contém os gráficos com o comportamento em termos do número de operações, espaço de memória e número médio de elementos por linha na matriz esparsa para Gauss e técnicas iterativas/multigrid. A Tabela 4.16 apresenta os coeficientes das retas das Figuras 4.32 e 4.33, além do número de equações onde ocorre o cruzamento das retas da Figura 4.32 entre os métodos de Gauss e iterativos.

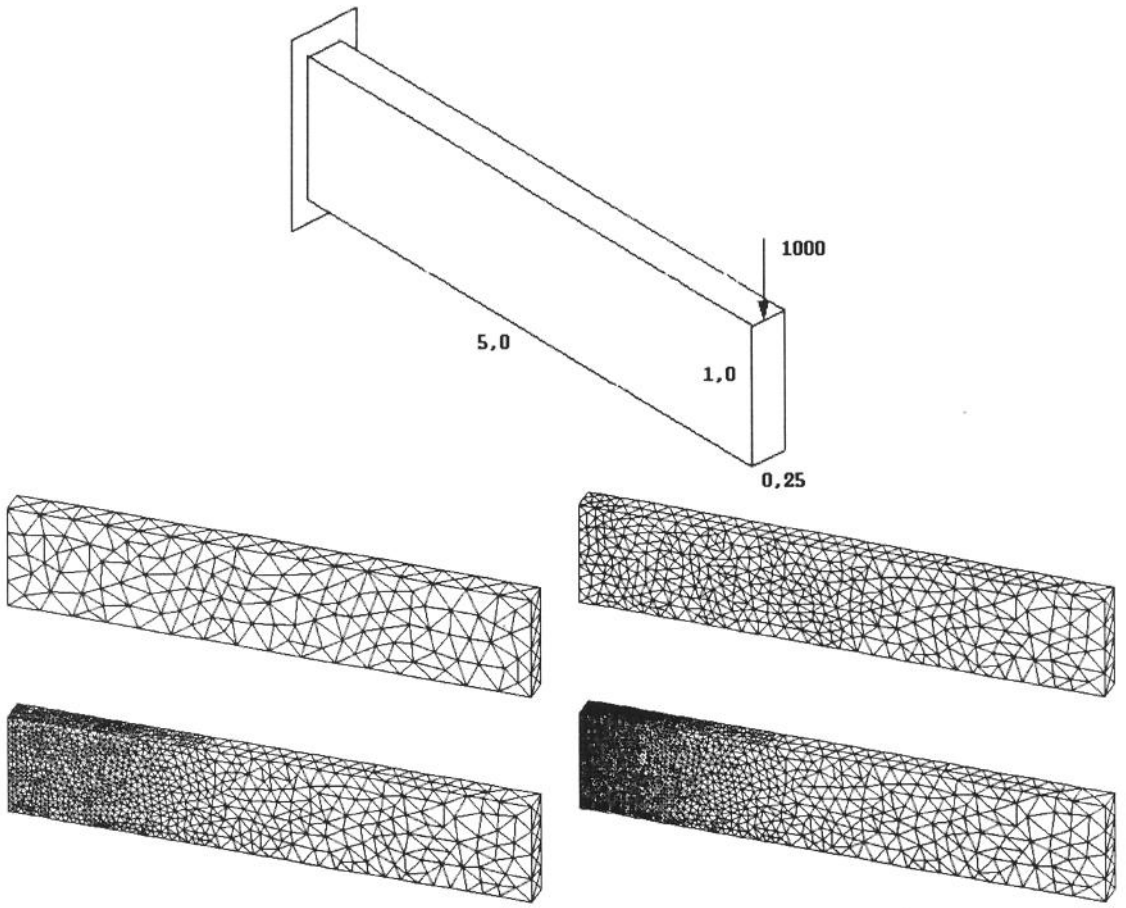


Figura 4.31: Malhas geradas para a viga em balanço ($E = 1,0 \times 10^5$; $\nu = 0,3$).

Malha	Nós	Elementos	Equações	m_{esp}^{dir}	$\text{NElems}_{esp}^{dir}$	m_{esp}^{ite}	$\text{NElems}_{esp}^{ite}$
1	25	921	963	40,0	38511	16,0	15444
2	1130	3922	3279	81,2	266226	17,6	57804
3	3024	11813	8724	164,2	1432572	18,6	162276
4	8633	37976	24606	361,9	8904168	19,6	482364

Tabela 4.13: Atributos das malhas da viga em balanço.

Método	Malha	1	2	3	4
GC	NIT	505	910	1426	2088
	$\|\|\mathbf{r}\|\|_2 (\times 10^{-5})$	9,50	8,98	9,83	9,94
GCD	NIT	404	664	887	1105
	$\|\|\mathbf{r}\|\|_2 (\times 10^{-5})$	9,01	9,37	9,06	8,97
GCSS	NIT	276	442	569	698
	$\|\|\mathbf{r}\|\|_2 (\times 10^{-5})$	9,73	9,69	7,29	9,80
GCGS	NIT	228	375	462	620
	$\|\|\mathbf{r}\|\|_2 (\times 10^{-5})$	5,98	8,61	8,99	8,80

Tabela 4.14: Resultados para os métodos iterativos na viga em balanço.

Método	NIT	Ciclos, ν_0, ν_1, ν_2	$\|\|e_r^{dir/ite}\|\|_2$	$\|\|e_r^{dir/mg}\|\|_2$	$\|\|e_r^{ite/mg}\|\|_2$
Gauss	4345	–	–	–	–
GC	2361	–	–	–	–
GCD	1279	–	–	–	–
GCSS	844	–	–	–	–
GCGS	672	–	–	–	–
FMV	89	9,1,1,1	$3,19 \times 10^{-8}$	$2,80 \times 10^{-6}$	$2,80 \times 10^{-6}$
	105	8,1,2,1	$3,19 \times 10^{-8}$	$3,77 \times 10^{-6}$	$3,77 \times 10^{-6}$
	98	6,2,1,1	$3,19 \times 10^{-8}$	$9,83 \times 10^{-7}$	$9,84 \times 10^{-7}$
	108	5,2,2,1	$3,19 \times 10^{-8}$	$3,40 \times 10^{-6}$	$3,40 \times 10^{-6}$
FMW	101	8,1,1,1	$3,19 \times 10^{-8}$	$5,74 \times 10^{-8}$	$6,60 \times 10^{-8}$
	100	6,1,2,1	$3,19 \times 10^{-8}$	$1,76 \times 10^{-7}$	$1,79 \times 10^{-7}$
	111	5,2,1,1	$3,19 \times 10^{-8}$	$3,59 \times 10^{-8}$	$4,82 \times 10^{-8}$
	116	4,2,2,1	$3,19 \times 10^{-8}$	$4,06 \times 10^{-8}$	$5,18 \times 10^{-8}$
FMVV	117	25,1,1,1	$3,19 \times 10^{-8}$	$1,28 \times 10^{-5}$	$1,28 \times 10^{-5}$
	140	23,1,2,1	$3,19 \times 10^{-8}$	$1,67 \times 10^{-5}$	$1,67 \times 10^{-5}$
	115	23,2,1,1	$3,19 \times 10^{-8}$	$1,17 \times 10^{-5}$	$1,17 \times 10^{-5}$
	137	21,2,2,1	$3,19 \times 10^{-8}$	$1,53 \times 10^{-5}$	$1,53 \times 10^{-5}$

Tabela 4.15: Resultados para a viga em balanço.

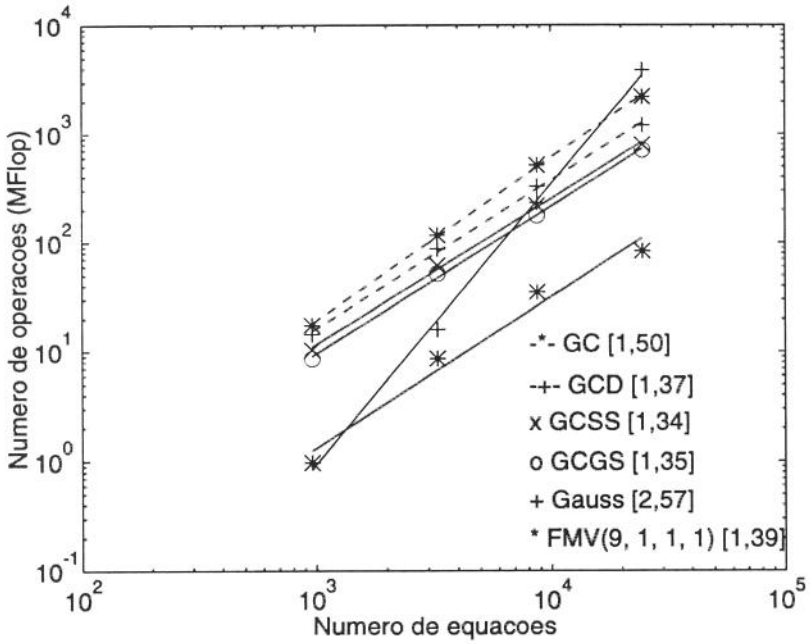


Figura 4.32: Número de operações (MFlop) para os métodos de Gauss, GC, GCD, GCSS, GCGS e FMV(9, 1, 1, 1) na viga em balanço.

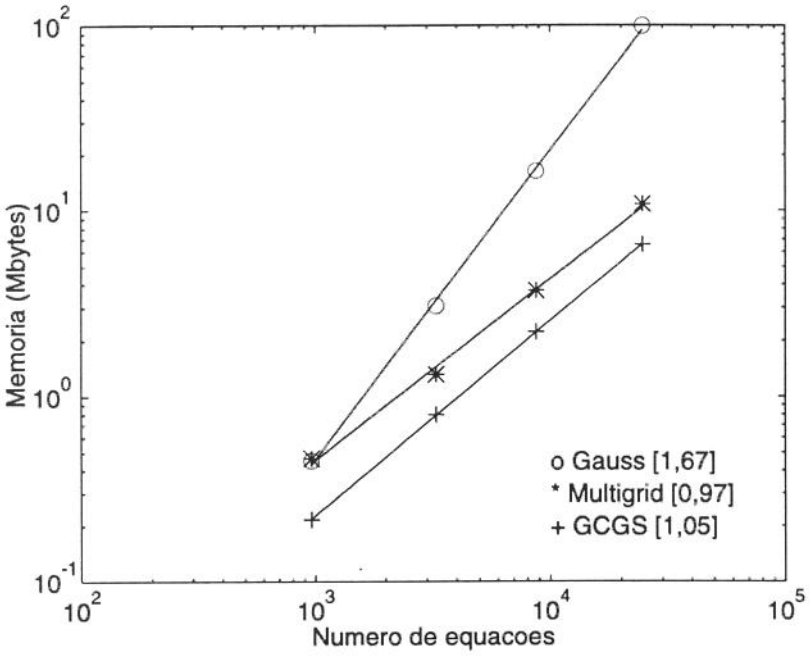


Figura 4.33: Espaço de memória (Mbytes) para os métodos de Gauss, iterativos e multigrid na viga em balanço.

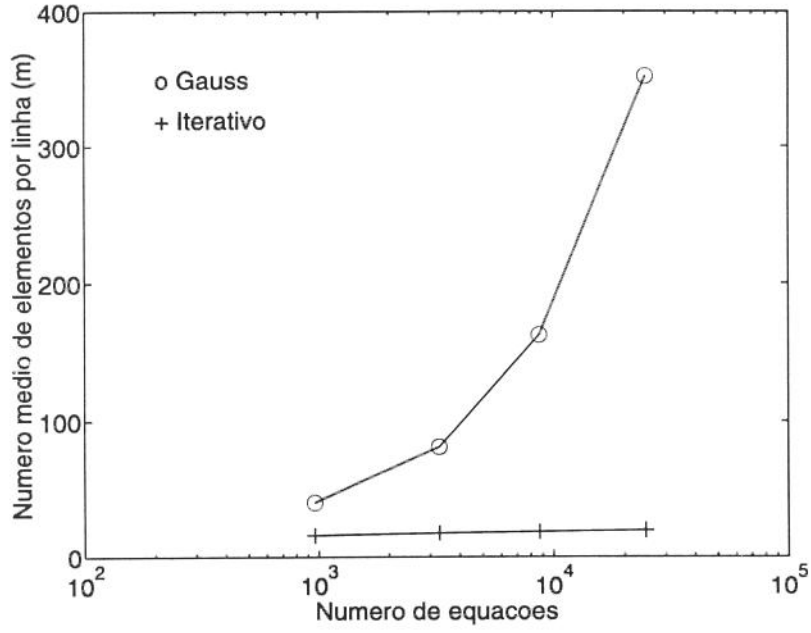


Figura 4.34: Número médio de elementos por linha para o método de Gauss e técnicas iterativas/multigrid na viga em balanço.

Método	Operações	Memória	Equações
Gauss	2,57	1,67	–
GC	1,50	1,05	16252
GCD	1,37	1,05	10442
GCSS	1,34	1,05	7608
GCGS	1,35	1,05	6659

Tabela 4.16: Coeficientes angulares das retas dos gráficos com número de operações e espaço de memória, além do ponto de cruzamento entre os métodos de Gauss e iterativos para a viga em balanço.

4.8.2 Pistão

O último exemplo analisado foi um pistão, sendo consideradas as 4 malhas ilustradas na Figura 4.35, estando as características dadas na Tabela 4.17. Pode-se observar na malha mais grossa, as restrições e a força concentrada com valor de 100 aplicados.

De forma análoga ao exemplo anterior, a Tabela 4.18 contém o número de iterações e a norma euclidiana do resíduo para os métodos iterativos GC, GCD, GCSS e GCGS. A Tabela 4.19 apresenta o número equivalente de iterações na malha fina para os métodos utilizados, além do erro relativo $||e_r^{ite/mg}||_2$ entre as técnicas multigrid e o método GCGS. As Figuras 4.36 a 4.38 ilustra os gráficos com o comportamento em termos do número de operações, espaço de memória e número médio de elementos por linha na matriz esparsa para Gauss e técnicas iterativas/multigrid. A Tabela 4.20 apresenta os coeficientes das retas das Figuras 4.36 e 4.37, além do número de equações onde ocorre o cruzamento das retas da Figura 4.36 entre os métodos de Gauss e iterativos.

Malha	Nós	Elementos	Equações	m_{esp}^{dir}	$\text{NElems}_{esp}^{dir}$	m_{esp}^{ite}	$\text{NElems}_{esp}^{ite}$
1	861	2622	2416	75,2	181675	16,4	39654
2	2943	10888	8444	154,5	1304874	18,0	151630
3	12727	55480	37188	373,2	13900617	19,4	722428
4	32348	155792	95277	722,1	68798689	20,5	1948811

Tabela 4.17: Atributos das malhas para o pistão.

Método	Malha	1	2	3	4
GC	NIT	367	567	987	1281
	$\|\|\mathbf{r}\|\|_2 (\times 10^{-5})$	9,51	9,89	9,94	9,96
GCD	NIT	254	428	783	1087
	$\|\|\mathbf{r}\|\|_2 (\times 10^{-5})$	9,71	9,84	9,90	9,89
GCSS	NIT	172	285	484	698
	$\|\|\mathbf{r}\|\|_2 (\times 10^{-5})$	9,40	9,87	9,99	9,80
GCGS	NIT	141	234	421	592
	$\|\|\mathbf{r}\|\|_2 (\times 10^{-5})$	9,95	9,68	9,92	9,74

Tabela 4.18: Resultados para os métodos iterativos no pistão.

4.8.3 Discussão dos resultados

A partir dos resultados obtidos para os dois problemas analisados, observa-se que:

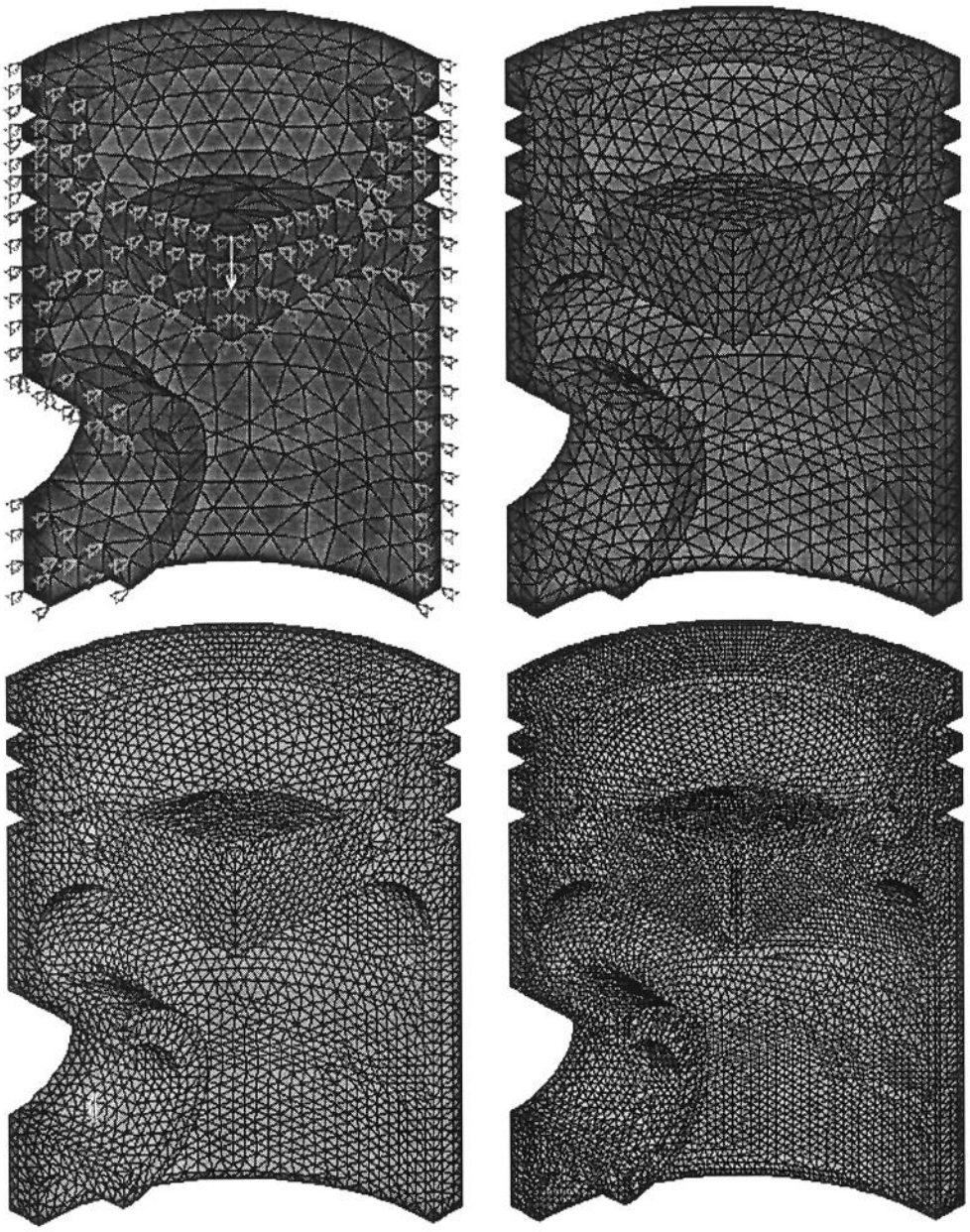


Figura 4.35: Malhas geradas para o pistão ($E = 1,0 \times 10^5$; $\nu = 0,3$).

Método	NIT	Ciclos, ν_0, ν_1, ν_2	$\|\|e_r^{ite/mg}\|\|_2$
Gauss	16934	–	–
GC	5492	–	–
GCD	4764	–	–
GCGS	3192	–	–
GCSS	2707	–	–
FMV	66	7,1,1,1	$9,29 \times 10^{-5}$
	75	6,1,2,1	$8,82 \times 10^{-5}$
	63	4,2,1,1	$1,07 \times 10^{-4}$
	83	4,2,2,1	$8,56 \times 10^{-5}$
FMW	71	6,1,1,1	$8,36 \times 10^{-5}$
	61	4,1,2,1	$8,96 \times 10^{-5}$
	62	3,2,1,1	$8,76 \times 10^{-5}$
	80	3,2,2,1	$8,32 \times 10^{-5}$
FMVV	78	16,1,1,1	$1,27 \times 10^{-4}$
	89	14,1,2,1	$1,62 \times 10^{-4}$
	75	15,2,1,1	$1,30 \times 10^{-4}$
	85	13,2,2,1	$1,65 \times 10^{-4}$

Tabela 4.19: Resultados para o pistão.

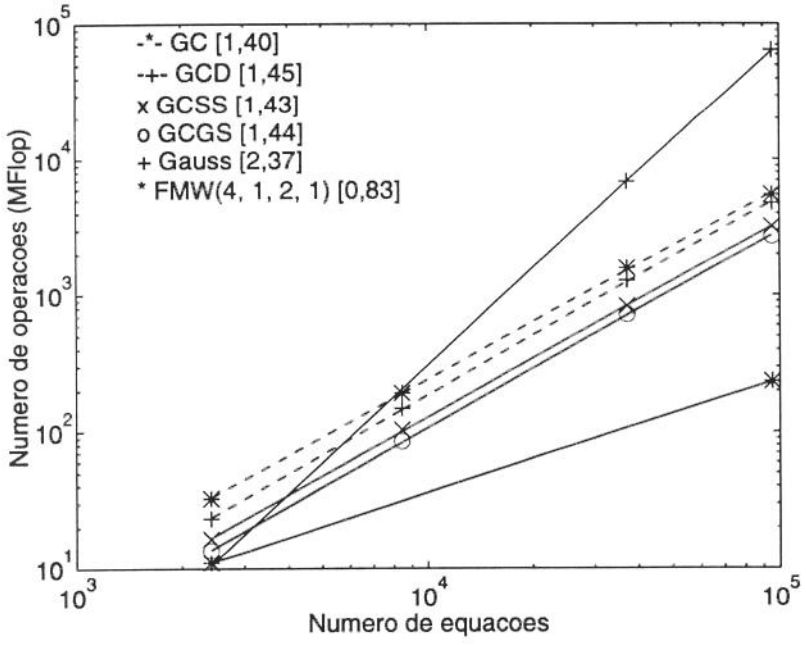


Figura 4.36: Número de operações (GFlop) para os métodos de Gauss, GC, GCD, GCSS, GCGS e FMV(4, 2, 1, 1) no pistão.

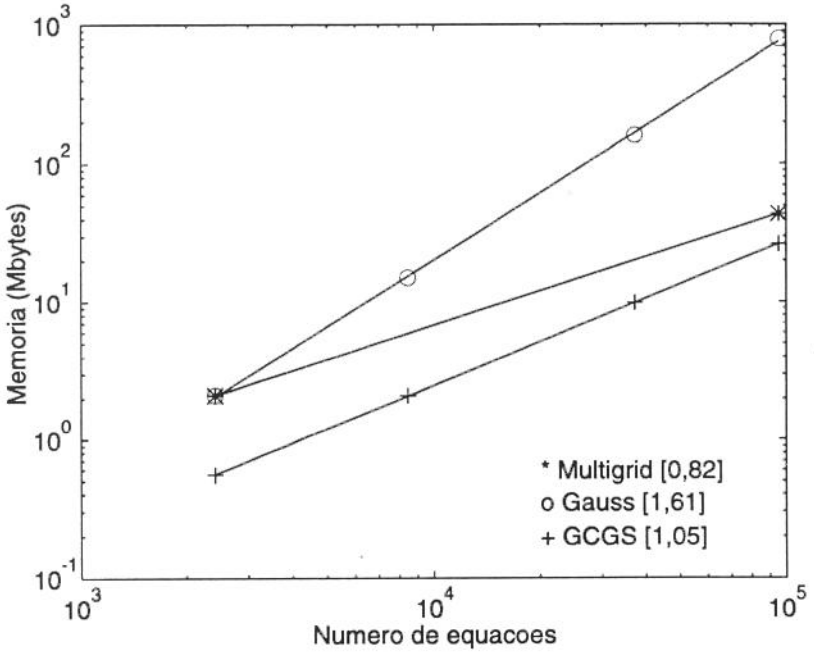


Figura 4.37: Espaço de memória (Mbytes) para os métodos de Gauss, iterativos e multigrid no pistão.

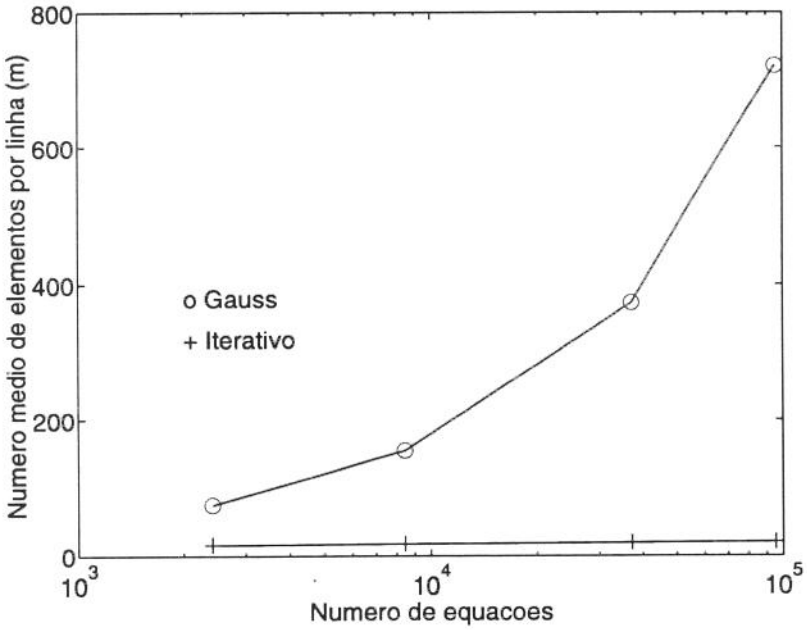


Figura 4.38: Número médio de elementos por linha para o método de Gauss e técnicas iterativas/multigrid no pistão.

Método	Operações	Memória	Equações
Gauss	2,37	1,61	–
GC	1,40	1,05	7825
GCD	1,45	1,05	5793
GCSS	1,43	1,05	3944
GCGS	1,44	1,05	3195

Tabela 4.20: Coeficientes angulares das retas dos gráficos com número de operações e espaço de memória, além do ponto de cruzamento entre os métodos de Gauss e iterativos para o pistão.

- o comportamento dos métodos iterativos baseados em gradiente conjugado é o mesmo encontrado para os casos bidimensionais do Capítulo 2, ou seja, os coeficientes angulares das retas das Figuras 4.32 e 4.36 são inferiores ao do método de Gauss. No entanto, para este último o número de operações cresce sensivelmente, fazendo com que o coeficiente angular fique próximo de 2,5 para os dois exemplos analisados. A partir dos números de equações indicados das Tabelas 4.16 e 4.20, verifica-se que todas as variantes de gradiente conjugado necessitam de um número menor de operações que o algoritmo direto para a solução da viga e do pistão.

  Além disso, a demanda em termos de memória também aumenta significativamente, tornando inviável a solução da malha mais fina do pistão por método de Gauss. Neste caso, a matriz e o vetor auxiliar necessitam cerca de 800 Mbytes de memória. Assim, caso seja necessário resolver de forma direta este problema, deve-se aplicar procedimentos de redução, como subestruturação, ou ainda técnicas frontais.

  Desta maneira, conclui-se que os métodos iterativos em problemas tridimensionais são os mais recomendados, tanto no que se refere ao espaço de memória, quanto ao número de operações para a solução.

- apesar das malhas da viga possuírem um menor número de equações que o exemplo do pistão, o número de iterações para convergência, tanto nos métodos iterativos quanto multigrid, é maior que no caso do pistão, devido a presença de maiores gradientes na solução. Tal fato explica o comportamento dos coeficientes das estratégias multigrid nas Figuras 4.32 e 4.36.

- as estratégias multigrid baseadas em FMV e FMW são superiores em relação a FMVV, necessitando um menor número de ciclos para atingir a precisão especificada.

- para problemas tridimensionais, torna-se interessante aplicar técnicas iterativas devido a menor demanda em termos de memória. A aceleração via procedimentos multigrid mostra-se bastante superior aos métodos de gradiente conjugado em relação ao número de operações. O espaço de memória acrescentado pelos métodos multigrid não é tão sensível, permitindo que todos os dados nas estratégias de solução sejam carregados na memória principal.

  Logo, como conclusão verifica-se que sem dúvidas as estratégias multigrid mostram-se bastante superiores aos algoritmos iterativos baseados em gradiente conjugado, e consequentemente também em relação ao método direto de Gauss.

- no exemplo de viga, as malhas foram geradas procurando-se simular o comportamento do estimador de erros. Devido a menor demanda por memória e menor número número de iterações para convergência, o procedimento adaptável pode, sem dúvida, ser recomendado para problemas tridimensionais.

- de forma geral, tanto para casos bi e tridimensionais, torna-se essencial ter ferramentas de geração de malhas, para o tratamento efetivo de problemas com multigrid não-aninhado.
- as Figuras 4.39 e 4.40 apresentam o comportamento médio dos métodos de Gauss, GCGS e multigrid tomando os resultados obtidos nos exemplos estudados. O comportamento geral é semelhante ao já encontrado. No entanto, para multigrid em problemas tridimensionais, obteve-se o número de operações variando linearmente com o número de equações.

  Desta forma, mesmo tomando o comportamento médio de apenas dois exemplos, alcançou-se um custo de solução da ordem $\mathcal{O}(N)$, para um número de equações inferior a 100.000, em malhas não-aninhadas, comprovando assim os resultados teóricos previstos para os métodos multigrid.

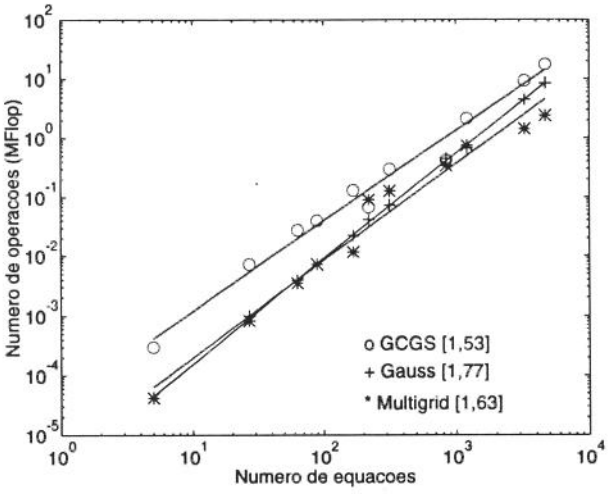


Figura 4.39: Número de operações (MFlop) para os método de Gauss, GCGS e Multigrid para os exemplos bidimensionais.

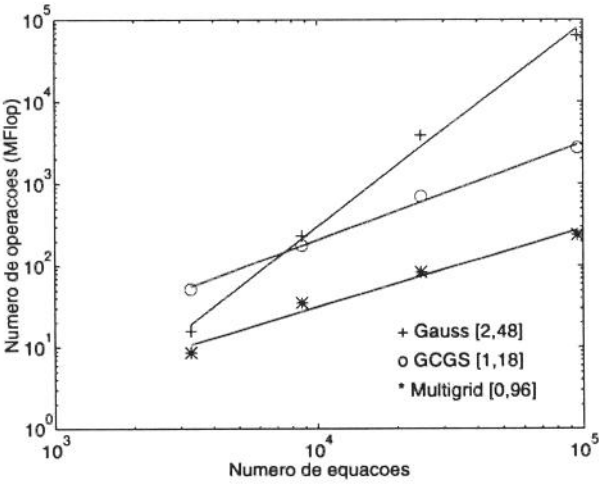


Figura 4.40: Número de operações (MFlop) para os método de Gauss, GCGS e Multigrid para os exemplos tridimensionais.

Capítulo 5

PROGRAMAÇÃO POR OBJETOS EM ELEMENTOS FINITOS

Neste capítulo, apresenta-se um resumo da aplicação de conceitos de programação orientada por objetos em elementos finitos. Inicialmente, considera-se um conjunto de classes para análise linear pelo MEF. Posteriormente, tem-se dois ambientes para o tratamento de problemas elásticos bidimensionais, classes para os métodos multigrid e recuperadores de tensão.

5.1 Classes para Elementos Finitos

O objetivo, neste caso, foi implementar um conjunto de classes para análise linear estática de estruturas modeladas por elementos isoparamétricos. Estas classes foram organizadas em 4 níveis como ilustrado na Figura 5.1. Observa-se que foram utilizados procedimentos do sistema ACDP [53] para o tratamento de erros, gravação e recuperação de dados em disco, assim como rotinas para manipulação de vetores e matrizes através da biblioteca MATEMATC [10].

5.1.1 NÍVEL 1

Envolve a definição das classes básicas do programa, compreendendo as estruturas de dados empregadas nos demais níveis. Tem como principal objetivo gerenciar a alocação dinâmica de memória.

Matrix : define uma matriz de elementos reais. Tem como principais variáveis os números de linhas e colunas, além de um vetor de com os elementos da matriz. Implementa métodos para inicialização da matriz, busca de informação (máximo, mínimo, norma, ordem, elemento), bem como operações envolvendo matrizes, onde estão implementados algoritmos de transposição, inserção de elementos, adição, multiplicação e multiplicação por escalar. Implementa, ainda, métodos diretos e iterativos para a solução de sistemas de equações, tais como Gauss e Gauss-Seidel.

SymmetricMatrix : classe análoga a **Matrix**, mas restrita a matrizes simétricas. Armazena apenas a parte inferior da matriz, economizando, desta forma, espaço na memória. Considera ainda métodos iterativos baseados em gradiente conjugado.

SymmetricSkyline : classe análoga a **Matrix** para matrizes do tipo skyline. É aplicada para definição da matriz global do sistema de equações. Armazena a ordem da matriz, altura das colunas e os elementos da parte inferior. Possui métodos para superposição de matrizes simétricas e resolução de sistemas de equações através de métodos diretos e iterativos.

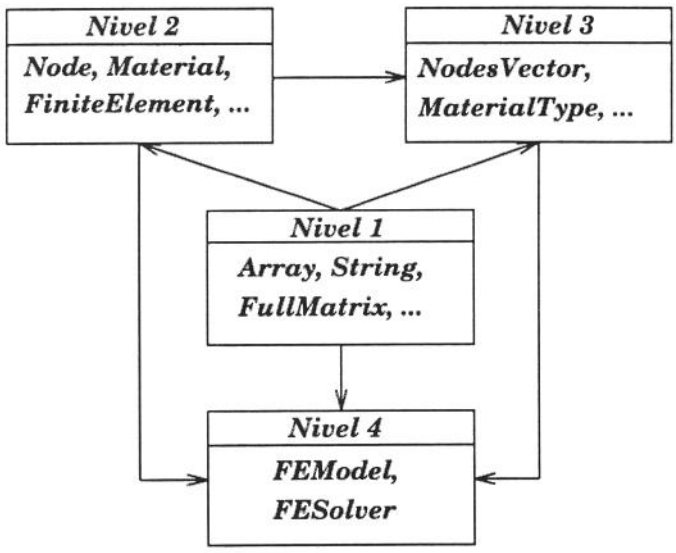


Figura 5.1: Níveis de organização das classes.

SymmetricSparse : é semelhante a classe análoga a **SymmetricSkyline**, considerando a estrutura comprimida por linhas para matriz esparsa apresentada no Capítulo 2.

Vector : caso particular de **Matrix** constituída de apenas uma coluna. Além dos procedimentos para adição, subtração e multiplicação, contém métodos para produto escalar, normas e multiplicação de matrizes por vetor.

String : classe responsável pelo manuseio de cadeias de caracteres ao longo do programa. Possui métodos de acesso a elementos, concatenação e comparação de caracteres.

Array : Tem por objetivo armazenar conjuntos de objetos. Na sua implementação, utilizou-se classes parametrizadas, onde o tipo de dado é também um parâmetro. Assim pode-se definir um Array de nós e de strings pelas declarações:

```
Array<Node> Nodes(5);
Array<String> Strings(10);
```

Em C++, o comando `template` permite a definição de classes e funções parametrizadas. Foram implementados métodos para acesso e alteração do número de elementos, dentre outros. Este tipo de dado tem como objetivo gerenciar conjuntos de qualquer tipo de dado ao longo do programa. Por questões de eficiência, considerou-se as especializações `Array<int>` e `Array<double>` para o tratamento de conjuntos de números inteiros e reais, respectivamente.

5.1.2 NÍVEL 2

Neste nível, consideram-se as classes relativas ao modelo de elementos finitos.

Node : possui como principais atributos o número do nó e as coordenadas nodais. O número de coordenadas nodais depende da dimensão do problema em estudo, ou seja, uni, bi ou tridimemsional. Possui ainda uma variável para o número ótimo do nó obtido por um algoritmo de renumeração

DefinitionDOF : esta classe tem por objetivo armazenar os nomes dos graus de liberdade armazenados como uma variável do tipo `Array<String>`.

EliminatedDOF : esta classe constitui-se numa tabela de graus de liberdade a serem eliminados, correspondendo aos deslocamentos nulos, no processo de montagem do sistema de equações. Para isso, armazena o número dos nós e a cardinalidade dos graus de liberdade eliminados, utilizando variáveis do tipo `Array<long>`.

PrescribedBC : é nesta classe que são armazenadas as condições de contorno prescritas do problema a ser solucionado pelo modelo de elementos finitos. Como exemplo, podem-se citar as cargas concentradas, distribuídas e de corpo; gradientes de temperatura; e deslocamentos. Utilizam-se variáveis dos tipos `Array<double>` e `Array<long>` para armazenar os nós, os graus de liberdade e a intensidade dos carregamentos, deslocamentos e temperaturas.

DOFEquation : armazena para cada nó, o número ótimo determinado por um algoritmo de renumeração. Além disso, o número e a numeração dos graus de liberdade são armazenados nesta classe. Possui métodos para realizar a numeração dos graus de liberdade.

GeometricProperties : é uma tabela para armazenar as propriedades geométricas dos elementos, tais como espessura, momentos de inércia, área de secção, etc. Esta tabela é identificada pelo seu número representada por uma variável inteira.

ElasticMaterial : é uma classe genérica, definindo as características básicas de outras classes que implementam as informações referentes ao comportamento dos materiais elásticos. Possui uma variável para o número do material. Declara várias funções virtuais para inicialização e acesso às propriedades dos materiais e outras para obtenção das matrizes de elasticidade nos casos de estado plano de tensão, estado plano de deformação, sólidos axissimétricos e estado geral de solicitação. A partir desta classe, derivam-se outras duas: **ElasticIsotropicMaterial** e **ElasticOrtotropicMaterial**.

ElasticIsotropicMaterial : declara variáveis para o armazenamento de informações relativas aos materiais elásticos isotrópicos, como o módulo de elasticidade longitudinal, coeficiente de Poisson, coeficiente de expansão térmica e densidade. Implementa as funções declaradas virtuais na classe **ElasticMaterial**. A classe **ElasticOrtotropicMaterial** é análoga a esta classe, considerando no entanto, materiais ortotrópicos.

FiniteElement : constitui-se numa classe genérica de onde derivam-se cada um dos diferentes tipos de elementos finitos. Declara variáveis número do elemento, número ótimo, número total de graus de liberdade, número do material, número da tabela de propriedades geométricas, número da tabela de sistemas locais de referência, número de pontos de integração e incidência.

Possui métodos para inicialização e acesso a estas informações, bem como métodos virtuais para o cálculo da matriz de rigidez, matriz de massa, tensões, deformações e erro em energia no elemento.

PlaneStressTriangular : é um dos tipos de elementos implementados. Através do mecanismo de herança, possui acesso a todos os atributos da classe **FiniteElement**. Implementa as operações declaradas virtuais em **FiniteElement** para o caso de estado plano de tensão.

Tem-se os métodos de cálculo das matrizes de rigidez e de massa, para os vetores de tensão e deformação, sendo estas últimas calculados nos pontos de interpolação de Gauss-Legendere e nas coordenadas locais dos nós. Estão implementados, também, métodos para o cálculo de tensão e deformação principais em cada um desses casos.

Foram implementadas as classes **PlaneStrainTriangle** e **AxyssimetricTriangle**, semelhantes a esta classe, para os casos de estado plano de deformação e sólidos axissimétricos, respectivamente.

TriangularShapeFunctions : implementa as funções de forma de Serendipty até o quarto grau para o caso de elementos finitos triangulares. Possui métodos para o cálculo das derivadas em relação às coordenadas locais e globais do elemento, assim como para a matriz e o determinante do Jacobiano.

TriangularGaussLegendre : armazena os pontos de integração e os coeficientes de ponderação para a integração de Gauss-Legendre utilizadas nas classes **PlaneStressTriangle**, **PlaneStrainTriangle** e **AxyssimetricTriangle**.

LoadVectorTriangle : classe derivada de **PrescribedBC**, herdando todos os seus atributos. Apresenta métodos para o cálculo dos vetores de carregamento térmico para os casos de estado plano de tensão, deformação e sólidos axissimétricos. Calcula ainda os vetores de carga de corpo e de superfície.

TriVectorShapeFunctions : o cálculo do vetor de carga distribuída envolve uma integral ao longo de uma linha. Esta classe contém as funções de forma e suas derivadas agrupadas de acordo com cada lado do triângulo considerado. Apresenta ainda o método para o cálculo da matriz do jacobiano.

LinearGaussLegendre : contém os pontos de integração e suas ponderações para a integração de Gauss-Legendre para o caso unidimensional, utilizados na determinação do vetor de carga distribuída.

Tem-se ainda as classes **SolidTethraedric**, **TethraedricShapeFunctions**, **TethraedricGaussLegendre** e **LoadVectorTethraedric**, com as mesmas características descritas para os triângulos, mas implementando as matrizes e vetores para os tetraedros. Da mesma maneira, foram implementadas classes para os elementos quadrangulares planos e espaciais.

5.1.3 NÍVEL 3

Neste caso, consideram-se conjuntos das classes descritas no nível anterior. Constitui-se basicamente no grupo de classes que vai organizar as informações dos atributos do modelo de elementos finitos. Permitem a inicialização, modificação e acesso aos atributos das classes utilizadas.

NodeVector : é um vetor de nós implementado através de uma variável do tipo `Array<Node>`.

DOFBoundaryConditions : armazena as tabelas de definição dos nomes dos graus de liberdade, graus eliminados e prescritos. Trata as informações das classes **DefinitionDOF**, **EliminatedDOF** e **PrescribedBC**.

MaterialGroup : implementa um vetor de apontadores da classe **ElesticMaterial**, armazenando ainda o tipo dos materiais (isotrópico ou ortotrópico) aplicados na análise.

FiniteElementGroup : um grupo de elementos finitos é constituído por elementos do mesmo tipo. Esta classe declara um vetor do tipo **FiniteElement** e o nome do elemento, a fim de gerenciar os vários tipos de elementos da malha.

LoadCase : é um vetor da classe **PrescribedBC** para o tratamento das condições de carregamento aplicadas à estrutura.

5.1.4 NÍVEL 4

Neste nível, tem-se classes para o armazenamento de todos os atributos do modelo de elementos finitos para que o problema seja posteriormente resolvido.

FEModel : esta classe armazena todos os atributos do modelo de elementos finitos. Define variáveis dos tipos descritos no Nível 3 para os nós, materiais, condições de contorno, grupos de elementos e condições de carregamento. Tem-se ainda uma variável da classe **String** para armazenar o título do modelo.

Assim, a partir de um arquivo de entrada, especificando as características do modelo e das operações implementadas nesta classe, inicializam-se todas as demais já apresentadas.

FESolver : é uma classe derivada de FEModel. Implementa operações para a resolução do modelo armazenado. Para isso, acessa todas as variáveis definidas em FEModel.

5.2 Ambientes para Análise de Problemas Elásticos Bidimensionais

A aplicação de métodos numéricos para a resolução de problemas práticos de engenharia tem apresentado um crescimento considerável. Estas ferramentas computacionais possibilitam otimizar projetos na sua fase inicial, assim como verificar o comportamento de um componente já existente visando validar a concepção atual, permitindo ainda a sua posterior otimização.

Vários programas comerciais estão disponíveis para esta finalidade. Sob o ponto de vista do usuário destes pacotes, exige-se um bom conhecimento da técnica de análise, assim como uma boa experiência na utilização do programa visto que em geral a interface de comandos é um tanto complexa. Tais fatos limitam uma maior disseminação destes programas, pois o usuário deve possuir uma base sólida de conhecimentos para o efetivo emprego destes recursos a problemas de engenharia. Uma hipótese mal formulada, implica em resultados não refletindo o real comportamento mecânico do componente. Desta maneira, o programa deve ser de fácil utilização para permitir ao usuário dedicar a maior parte do tempo na definição e simulação de modelos para a estrutura considerada.

Observa-se que, em geral, as tarefas de definição da geometria do componente e da respectiva malha de elementos finitos são responsáveis pela maior demanda em termos de tempo no estudo de um problema. Em relação aos pacotes existentes, observa-se uma maior integração entre programas de CAD (*Computer Aided Design*) e de análise. Empregam-se as ferramentas CAD para a definição da geometria e possivelmente da malha. A partir daí, utiliza-se o programa de análise para a obtenção dos resultados. No entanto, em vários casos não se verifica uma efetiva integração ou onde esta ocorre tem-se uma interface complexa dificultando a tarefa do usuário.

Desta forma, algoritmos robustos e eficientes para a geração da geometria do componente e da respectiva malha de elementos, solução, estimação de erros e refinamento, acessíveis através de uma interface de comandos simples, são pré-requisitos fundamentais para a análise numérica por elementos finitos.

Dentro deste contexto, desenvolveu-se inicialmente o ambiente SAFE [54] para análise bidimensional de problemas elásticos planos. A Figura 5.2 ilustra a janela principal e alguns dos módulos do programa, tais como o editor para a especificação dos parâmetros da estrutura e da análise, o gerador de malhas, a visualização de resultados, além da malha refinada através do procedimento adaptável baseado no estimador ZZ com recuperador PA.

A entrada de dados é feita através de 2 arquivos de dados, com extensões **.ara** e **.a2f**, contendo várias palavras chaves [54]. O primeiro especifica o contorno do domínio, parâmetros de controle da

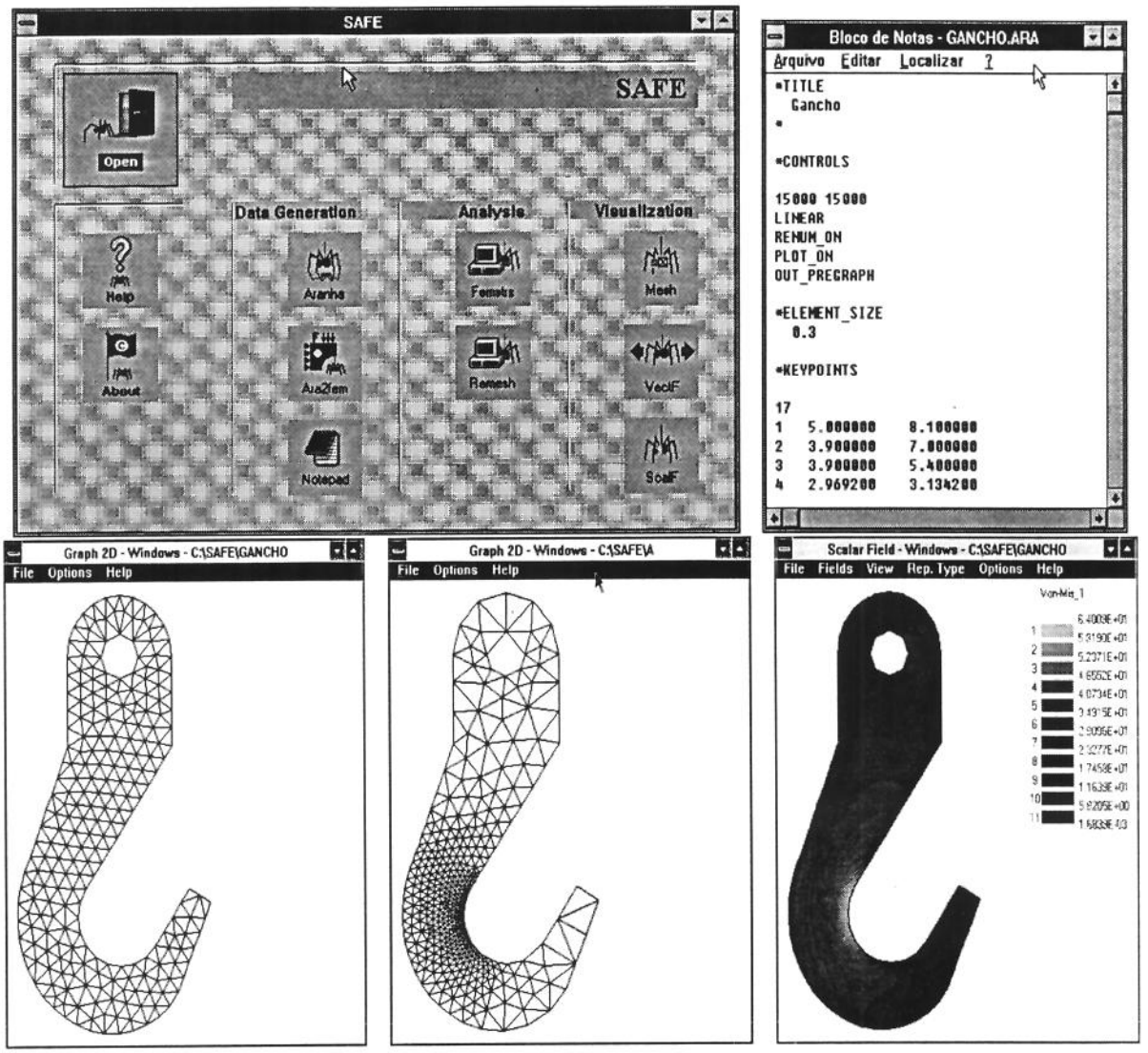


Figura 5.2: Módulos do programa SAFE.

malha e erro admissível. O outro arquivo considera as propriedades dos materiais, tipos de elementos, carregamentos, restrições, graus de liberdade e propriedades geométricas. Com o arquivo **.ara**, efetua-se a geração da malha através do programa ARANHA [43]. A partir daí, aplicam-se os parâmetros disponíveis no arquivo **.a2f**, gerando como saída um outro arquivo com extensão **.fem** a ser submetido ao módulo de solução. Estas tarefas constituem a parte de geração dos dados do programa.

Na parte de análise, resolve-se o modelo de elementos finitos determinando deslocamentos e tensões. Tem-se ainda o módulo para efetuar o procedimento adaptável gerando um novo conjunto de arquivos de entrada. Finalmente, pode-se efetuar a visualização da malha e de campos vetoriais e escalares. Observa-se que a comunicação entre os vários módulos é realizadas através de arquivos e bancos de dados.

A partir da experiência do programa SAFE, desenvolveu-se um segundo ambiente, denominado SAT, onde todos os dados do modelo são especificados de forma interativa. A Figura 5.3 ilustra a janela principal e as interfaces de alguns módulos. Inicialmente, fornece-se o nome do projeto, definindo o nome dos bancos de dados a serem empregados nas demais partes do programa. Alguns argumentos genéricos do projeto, tais como título, data da última modificação, autor e observações podem ser

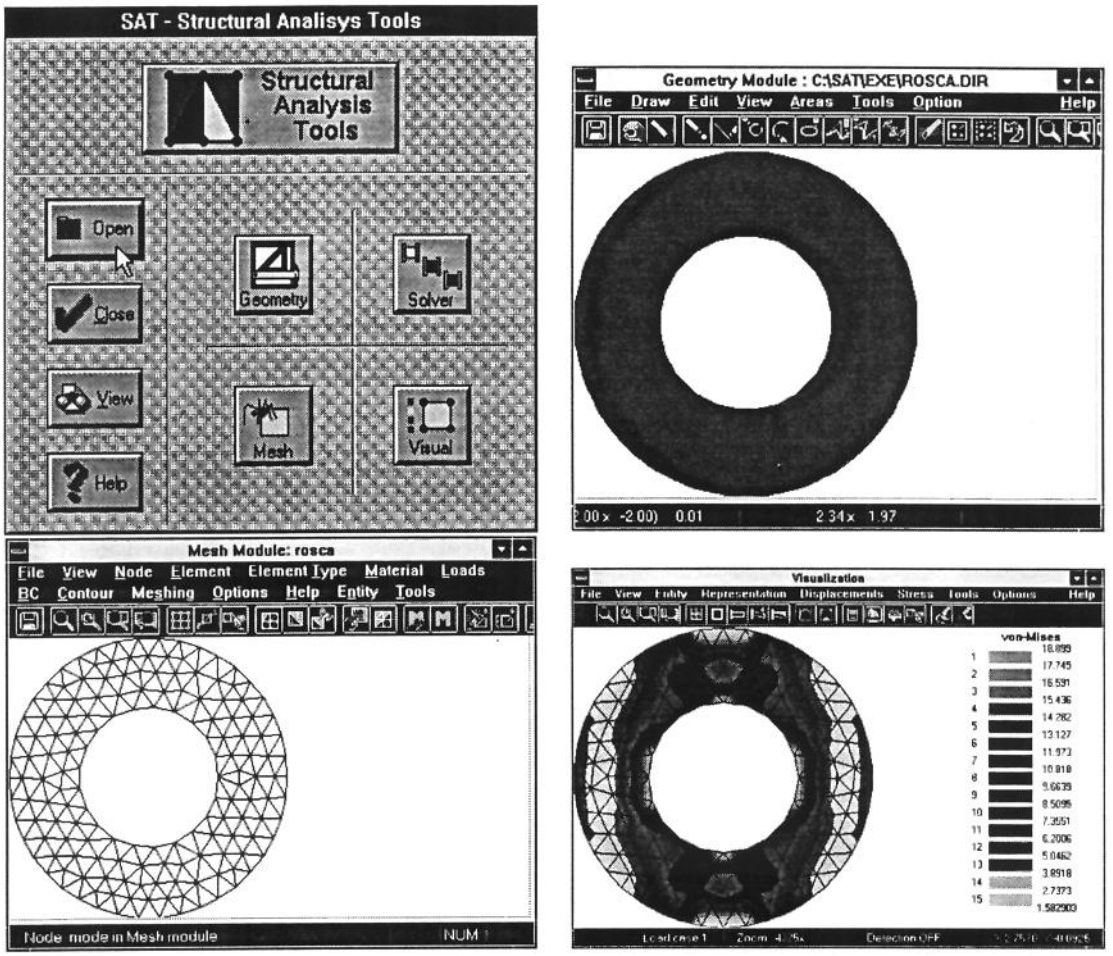


Figura 5.3: Módulos do programa SAT.

colocadas neste arquivo de projeto.

O módulo GEOMETRY permite a definição do contorno da geometria do componente a ser analisado. Está baseado no conceito de NURBS (*Non-uniform Rational B-Splines*) [51], pondendo-se ainda importar arquivos no formato DXF [98]. Possui como primitivas principais linha, arco, círculo, curva, dentre outras. Várias ferramentas de edição estão disponíveis tais como rotação, translação, *zoom* e divisão. O objetivo principal é definir as áreas sobre as quais serão gerados os elementos, gravando-se ao final o arquivo com extensão **.ara**.

No caso do módulo MESH [99], partindo-se das áreas armazenadas no banco de dados do GEOMETRY, agregam-se informações de controle da malha, como por exemplo o tamanho médio dos elementos. A partir daí, executa-se o programa ARANHA [43] para a geração da malha. Todos os atributos do modelo de elementos finitos tais como carregamentos, propriedades dos materias e condições de contorno podem ser especificados de forma interativa através de diálogos, gravando-se ao final o arquivo **.a2f**. Com os arquivos **.ara** e **.a2f**, gera-se o **.fem** para ser submetido ao módulo SOLVER, o qual é o mesmo do programa SAFE, não tendo sido implementado ainda o estimador de erro.

Ao final, os resultados escalares e vetoriais podem ser apresentados de formas numérica e gráfica no módulo VISUALIZATION [80]. De forma análoga ao SAFE, tem-se mapas de cores em faixas e curvas de níveis, geometria original e/ou deformada, agregando ainda um procedimento de animação. Um aspecto importante foi a organização dos campos escalares e vetoriais, onde os nomes e a quantidade dos

mesmos são gravados no banco de dados do SOLVER. Assim, o programa determina dinamicamente os campos a serem apresentados, permitindo a sua aplicação para a visualização de resultados de outros tipos de análise.

Cada um dos módulos constitue-se num programa independente estando a comunicação entre os mesmos efetuada através de bancos de dados. A representação das entidades gráficas em todos os módulos é feita por uma mesma estrutura de dados baseada em NURBS. Isto permite uniformizar todas as operações efetuadas tais como rotação, translação, *zoom*, *fill*, dentre outras.

Os resultados obtidos para os dois programas foram bastante satisfatórios. Apesar de tratarem de um único tipo de problema, os programas apresentam ferramentas efetivamente integradas, desde a parte de geração de contornos e de malha, visualização dos resultados, estimação de erros e refinamento adaptável. As interfaces gráficas permitem um acesso simples e intuitivo aos vários módulos disponíveis. Outros tipos de problemas podem ser facilmente integrados na estrutura já existente, criando-se por exemplo novos módulos de análise. Pretende-se agora continuar na mesma linha de desenvolvimento para o tratamento de casos tridimensionais.

5.3 Implementação dos Métodos Multigrid

Para os métodos multigrid foram implementados dois programas para os casos bi e tridimensionais, possuindo basicamente as seguintes opcções:

geração de banco de dados : através dos arquivos **.fem** das várias malhas, gera-se um banco de dados em formato binário contendo as coordenadas nodais, as incidências dos elementos, os mapeamentos malha fina/grossa, as matrizes de rigidez em forma esparsa, vetores de carregamentos e numeração dos graus de liberdade. Observa-se que na matriz de rigidez da malha grossa aplica-se a decomposição de Gauss antes da sua gravação.

solução por multigrid : a partir do banco de dados, pode-se selecionar uma estratégia multigrid (V, W, FMV, FMW, etc) e os parâmetros tais como precisão e números de iterações, pré e pós-relaxações. Os resultados são armazenados num arquivo de saída especificado.

solução por método direto e/ou iterativo : é possível resolver a malha mais fina através do método direto de Gauss ou uma das técnicas iterativas empregadas no Capítulo 2. Neste último caso, selecionam-se o número máximo de iterações, critério de convergência, precisão e norma euclidiana ou infinita. Novamente, armazenam-se os resultados num arquivo texto.

cálculo dos erros relativos : a partir das soluções obtidas pelos métodos direto, iterativo e multigrid, calculam-se os erros relativos conforme as expressões indicadas em (4.52).

multigrid adaptável : aplica a estratégia adaptável discutida no Capítulo 4. Observa-se que a geração das malhas é realizada separadamente a partir dos arquivos **.ara** e **.a2f** gravados após a execução do estimador de erro ZZ e do procedimento de refinamento.

parâmetros do banco de dados : permite alterar o nome do banco de dados e o número de malhas, sendo empregado para analisar um novo problema ou atualizar a base de dados no procedimento adaptável. Neste último caso, deve-se gerar novamente todo o banco de dados.

Observa-se que o mapeamento entre as malhas é realizado por dois programas independentes para problemas bi e tridimensionais [35], sendo as informações incluídas nos arquivos **.fem**. Desta forma, verifica-se que os programas implementados demandam um certo trabalho manual na preparação dos arquivos de entrada, pois a principal preocupação, neste caso, foi avaliar a performance das várias

estratégias multigrid. Além disso, como não se sabia de início a real demanda em termos de memória, todos os atributos são lidos do banco de dados cada vez que são empregados, demandando um tempo de processamento maior que aquele realmente necessário. Após os resultados obtidos no Capítulo 4, algumas modificações simples serão efetuadas para carregar todos os dados na memória. Da mesma forma, pretedem-se integrar todos os programas num ambiente único.

No que se refere as estratégias multigrid, implementou-se apenas uma classe, contendo como dados principais o nome do banco de dados, o número de malhas, parâmetros tais como números de iterações, pré e pós-relaxações, além de dois vetores de apontadores para as incógnitas e os termos independentes de cada malha. O desenvolvimento das estratégias foi bastante simplificado, considerandos os dois métodos seguintes:

```
int FineToCoarsePart(int Level);
int CoarseToFinePart(int Level);
```

O primeiro deles realiza a parte descendente dos métodos multigrid, ou seja, aplicam-se ν_1 relaxações, calcula-se e transfere-se o resíduo sucessivamente até atingir a malha mais grossa. A partir daí, o segundo método resolve de forma direta o sistema da malha grossa, pronloga a correção e realiza ν_2 pós-relaxações até atingir o nível mais fino.

Desta forma, as estratégias multigrid apresentadas anteriormente são facilmente implementadas. Por exemplo, para um ciclo V, o núcleo central é feito pelos seguintes comandos em C++:

```
//fine to coarse part
for(k = NumMeshes - 1; k >= 0; k--) FineToCoarsePart(k);

//coarse to fine part
for(k = 1; k < NumMeshes; k++) CoarseToFinePart(k);
```

Observa-se que os dois procedimentos anteriores fazem uso extensivo de outros métodos para os operadores de restrição e pronlogamento, além de outro para cálculo do resíduo. Assim, através destes procedimentos, pode-se implementar de forma bastante simples qualquer uma das estratégias multigrid discutidas no Capítulo 4.

5.4 Implementação dos Recuperadores de Tensão

Neste caso, partiu-se da implementação do algoritmo ZZ com recuperador PA dada em [44]. Basicamente, tem-se uma classe armazenando todos os dados necessários, tais como coordenadas nodais, incidência dos elementos, propriedades dos materiais, tensões nodais e erro admissível. Como métodos principais, tem-se aqueles para o cálculo do estimador ZZ, recuperador PA e determinação dos novos tamanhos dos elementos.

Foram agregados outros procedimentos para os recuperadores discutidos no Capítulo 3, assim como o cálculo do tamanho dos elementos para as malhas em métodos multigrid considerando a restrição (4.17).

Capítulo 6

CONCLUSÕES E PERSPECTIVAS FUTURAS

A partir dos resultados obtidos nos capítulos anteriores, apresentam-se, para problemas lineares e elípticos, as seguintes conclusões finais do trabalho;

- torna-se fundamental empregar estruturas de dados em matrizes esparsas para o armazenamento da matriz do sistema de equações tanto em métodos diretos quanto iterativos.

- para casos bidimensionais, o método de Gauss é superior às técnicas iterativas, mesmo demandando um maior espaço de memória. No entanto, para problemas tridimensionais, os algoritmos iterativos baseados em gradiente conjugado apresentam uma melhor performance no que se refere ao número de operações e espaço de memória, principalmente a medida que se aumenta o número de equações [19].

- de forma geral, as estratégias multigrid, incorporando iterações aninhadas e correção de malha grossa, apresentam um comportamento superior aos métodos diretos e iterativos. O emprego de malhas não-aninhadas é de grande importância para o tratamento de domínios complexos, não tendo sido observado prejuízo na taxa de convergência dos métodos multigrid empregados [20, 21, 22]. Além disso, um único programa é capaz de tratar malhas aninhadas e não-aninhadas [23].

- procedimentos automáticos para a geração de malhas são fundamentais na análise numérica de problemas de engenharia. Neste trabalho, a utilização de geradores frontal e de Delaunay foram cruciais, permitindo obter malhas adequadas para os métodos multigrid e processos adaptáveis.

- por sua vez, a utilização de estimadores de erro e técnicas de refinamento com métodos multigrid permitem não só resolver com ordem ótima o sistema de equações do nível mais fino, mas também obter uma sequência ótima de malhas para alcançar uma solução final dentro de um certo erro especificado. Desta forma, tem-se uma maneira mais natural e automática para a obtenção das malhas e da solução, evitando assim o incoveniente de se fornecer logo de início todas as malhas para um programa multigrid [24].

- efetivamente, o modelo de programação por objetos com C++ permite aumentar significativamente a produtividade e a qualidade dos programas. A partir de um conjunto de classes para elementos finitos, foi possível implementar num curto período algumas estratégias multigrid.

No que se refere a eficiência de C++, pode-se obter resultados superiores a outras linguagens aplicadas em análise numérica, como por exemplo Fortran, sem no entanto perder as principais características da programação por objetos.

Os resultados obtidos com os dois ambientes de análise discutidos são até então bastante satisfatórios, incentivando a continuação do trabalho.

- como mencionado no Capítulo 1, a extensão das conclusões para outros tipos de problemas, tais como não-lineares e transientes, não é direta, sendo necessário estudar o comportamento dos métodos abordados neste trabalho para cada caso considerado [25].

- os objetivos iniciais estabelecidos foram plenamente alcançados. Foi possível obter ao final ferramentas computacionais eficientes para a solução de sistemas de equações de alta ordem. Neste sentido, a performance dos métodos multigrid em malhas não-aninhadas e geradas por um critério adaptável foi bastante razoável, incentivando a continuação do trabalho.

Assim, em relação as perspectivas de trabalho futuras pretende-se:

- revisar todos os programas implementados e integrá-los definitivamente na base de softwares já disponível.

- incorporar estratégias adaptáveis do tipo p com métodos multigrid em malhas não-aninhadas.

- considerar outros tipos de estimadores de erro, tais como aqueles baseados em técnicas de resíduo.

- aplicar estratégias multigrid em problemas de flexão de placas, cascas, modelos mistos, transientes e não-lineares.

- estender os ambientes de análise para o tratamento de problemas elásticos tridimensionais.

Bibliografia

[1] ADELI, H., YU, G. An integrated computing environment for solution of complex engineering problems using the object-oriented programming paradigm and a blackboard architeture. *Computer & Structures,* v.54, n.2, p.255-265, 1995.

[2] AXELSSON, O., BARKER, V. A. *Finite element solution of boundary value problems – theory and computation.* Orlando: Academic Press, 1984.

[3] AINSWORTH, M., ZHU, J. Z., CRAIG, A. W., ZIENKIEWICZ, O. C. Analysis of the Zienkiewicz-Zhu *a-posteriori* error estimatior in the finite element method. *International Journal for Numerical Methods in Engineering,* v.28, p.2161-2174, 1989.

[4] BABUŠKA, I., RHEINBOLDT, W. C. *A posteriori* error estimates for the finite element method. *International Journal for Numerical Methods in Engineering,* v.12, p.1597-1615, 1978.

[5] BABUŠKA, I., RHEINBOLDT, W. C. Error estimates for adaptive finite element computations. *SIAM Journal on Numerical Analysis,* v.15, p.736-754, 1978.

[6] BABUŠKA, I., RHEINBOLDT, W. C. A *posteriori* error analysis of finite element solutions for onde dimensional problems. *SIAM Journal on Numerical Analysis,* v.18, p.565-589, 1981.

[7] BABUŠKA, I., STROUBOULIS, T., UPADHYAY C. S. A model study of the quality of a *posteriori* error estimators for linear elliptic problems. Error estimation in the interior of patchwise uniform grids of triangles. *Computer Methods in Applied Mechanics and Engineering,* v.114, p.307-378, 1994.

[8] BABUŠKA, I., STROUBOULIS, T., UPADHYAY C. S., GANGARAJ, S. K., COPPS, K. Validation of *a posteriori* error estimators by numerical approach. *International Journal for Numerical Methods in Engineering,* v.37, p.1073-1123, 1994.

[9] BAKHVALOV, N. S. Convergence of a relaxation method with natural constraints on an elliptic operator. *Ž. Vyčisl. Mat. i Mat. Fiz.*, v.6, p.861-885, 1966.

[10] BALLESTÉ, A., GUIMARÃES, A. C. S., GARCIA, R. E., FEIJÓO, R. A. *Biblioteca MATEMATC - Manual de Referência.* Laboratório Nacional de Computação Científica (LNCC), Rio de Janeiro, Julho de 1991.

[11] BANK, R. E., DUPONT, T. An optimal order process for solving finite element equations. *Matemathics of Computation,* v.36, n.153, p.35-51, 1981.

[12] BANK, R. E., SHERMAN, A. H. An adaptive, multi-level method for elliptic boundary value problems. *Computing,* v.26, p.91-105, 1981.

[13] BANK, R. E. *PLTMG: a software package for solving elliptic partial differential equations – user's guide 6.0.* Philadelphia: SIAM Frontiers in Applied Mathematics, v.7, 1989.

[14] BARLOW, J. Optimal stress location in finite element method. *International Journal for Numerical Methods in Engineering,* v.10, p.243-251, 1976.

[15] BARRET, R. et all. *Templates for the solution of linear systems: building blocks for iterative methods*. Edição preliminar (ftp netlib2.cs.utk.edu), 1994.

[16] BATHE, K., WILSON, E. L. *Numerical methods in finite element analysis.* Englewood-Cliffs: Prentice Hall, 1976.

[17] BEY, J. Tetrahedral grid refinement. *Computing,* v.55, p.355-378, 1995.

[18] BITTENCOURT, M. L. *Análise estática e dinâmica por subestrururação e programação orientada por objetos*. Campinas: DPM/FEM/UNICAMP, 1990, Tese (Mestrado).

[19] BITTENCOURT, M. L., FEIJÓO, R. A. Uma análise comparativa entre métodos diretos e iterativos para a solução de sistemas de equações. *Trabalho a ser publicado na Revista Internacional de Métodos Numéricos y Diseño em Ingenieria*, 1996.

[20] BITTENCOURT, M. L., FEIJÓO, R. A. Métodos diretos, iterativos e multigrid para a solução de sistemas de equações. *Trabalho a ser publicado na Revista Internacional de Métodos Numéricos y Diseño em Ingenieria*, 1996.

[21] BITTENCOURT, M. L., FEIJÓO, R. A. Non-nested and non-structured multigrid methods applied to elastic problems: the two-dimensional case. *Trabalho submetido ao Journal of Computational Physics*, 1996.

[22] BITTENCOURT, M. L., FEIJÓO, R. A. Non-nested and non-structured multigrid methods applied to elastic problems: the three-dimensional case. *Trabalho submetido ao Journal of Computational Physics*, 1996.

[23] BITTENCOURT, M. L., FEIJÓO, R. A. Non-nested and non-structured multigrid methods applied to elastic problems: the three-dimensional case. *Trabalho submetido ao Journal of Computational Physics*, 1996.

[24] BITTENCOURT, M. L., FEIJÓO, R. A. A non-nested and non-structured object-oriented multigrid program. *Trabalho a ser submetido a um periódico internacional*, 1996.

[25] BITTENCOURT, M. L., FEIJÓO, R. A. Object-oriented environments for elastic problems. *Trabalho a ser submetido a um periódico internacional*, 1996.

[26] BONET, J., PERAIRE, J. An alternating digital tree (adt) algorithim for 3D geometric searching. *International Journal for Numerical Methods in Engineering,* v.38, p.3529-3544, 1995.

[27] BRAMBLE, J. H., PASCIAK, J. E., XU, J. The analysis of multigrid algorithms with nonnested quadratic forms. *Mathematics of Computation,* v. 56, n.193, p.1-34, 1991.

[28] BRANDT, A. Multi-level adaptive technique (MLAT) for fast numerical solution to boundary value problems. In: THIRD INTERNATIONAL CONFERENCE ON NUMERICAL METHODS IN FLUID MECHANICS, 1972, Paris. Lecture Notes in Physics, Berlin: Springer-Verlag, 1973, v.18, p.82-89.

[29] BRANDT, A. Multi-level adaptive solutions to boundary-value problems. *Mathematics of Computation,* v.31, n.138, p.333-390, 1977.

[30] BRANDT, A. Algebraic multigrid theory: the symmetric case. In: INTERNATIONAL MULTIGRID CONFERENCE, 1983, Copper Mountain. Appl. Math. Comp. (S.F. McCormick, U, Trotenberg, editores).

[31] BRIGGS, W. L. *A Multigrid tutorial.* Pennsylvania: SIAM, 1987.

[32] BULGAKOV, V. E., KUHN, G. High-performance multilevel iterative aggregation solver for large finite element structural analysis problems. *International Journal for Numerical Methods in Engineering,* v.38, p.3529-3544, 1995.

[33] CARDONA, A., KLAPKA, I., GERADIN, M. Design of a new finite element programming environment. *Engineering Computations,* v.11, p.365-381, 1994.

[34] CUTHILL, E., MCKEE, J. Reducing the bandwidth of sparse symmetric matrices. Proc. 24th Nat. Conf. Assoc. Comp. Mach, ACM Publ., p.157-172, 1969.

[35] DARI, E. *Contribuciones a la Triangulación Automática de Dominios Tridimensionales.* Bariloche, Argentina: Instituto Balseiro, 1994, Tese (Doutorado).

[36] DEUFLHARD, P., LEINEN, P., YSERENTANT, H. Concepts of an adaptive hierarchical finite element code. *Impact Comput. Sci. Eng.,* v.1, p.3-35, 1995.

[37] DOUGLAS, C. C. Multi-grid algorithms with applications to elliptic boundary-value problems. *SIAM Journal on Numerical Analysis,* v.21, p.236-254. 1984.

[38] DOUGLAS, C. C., DOUGLAS JR., J. A unified convergence theory for abstract multigrid on multilevel algorithms, serial and parallel. *SIAM Journal on Numerical Analysis,* v.30, n.1, p.136-158, 1993.

[39] DOUGLAS, C. C., DOUGLAS JR., J, FYFE, D. E. A unified convergence theory for non-nested grids and/or quadrature. *East-West Journal Numerical Math.,* (submetido).

[40] DUBOIS-PÈLERIN, Y., ZIMMERMANN, T., BOMME, P. Object-oriented finite element programming: II. a prototype program in Smalltalk. *Computer Methods in Applied Mechanics and Engineering,* v.98, p.361-397, 1992.

[41] DUBOIS-PÈLERIN, Y., ZIMMERMANN, T., BOMME, P. Object-oriented finite element programming: III. an efficient implementation in C++. *Computer Methods in Applied Mechanics and Engineering,* v.108, p.165-183, 1993.

[42] EL HADJ, M., DEMAY, Y., CHENOF, J. L. A two-grid method applied to plane elasticity. *Computer & Structures,* v.44, n.2, p.169-175, 1986.

[43] FANCELLO, E. A., GUIMARÃES, A. C. S., FEIJÓO, R. A., VENERE, M. Geração automática de malhas 2D em programação orientada a objetos. In: XI COBEM - CONGRESSO BRASILEIRO DE ENGENHARIA MECÂNICA, 1991, São Paulo, p.635-638.

[44] FANCELLO, E. A., FEIJÓO, R. A. *ADAPTE: estimador de erro para problemas planos em elasticidade linear.* Relatório de Pesquisa e Desenvolvimento, Nº 19/92, Laboratório Nacional de Computação Científica (LNCC), Rio de Janeiro, 1992.

[45] FEDORENKO, R. P. On the speed of convergence of an iteration process. *Ž. Vyčisl. Mat. i Mat. Fiz.*, v.4, p.559-564, 1964.

[46] FEIJÓO, R. A. Relatórios do Sistema SDP formado por 15 manuais. Laboratório Nacional de Computação Científica (LNCC), Rio de Janeiro, 1987.

[47] FEIJÓO, R. A., GUIMARÃES, A. C. S., FANCELLO, E. A. *Algumas experiencias en la programación orientada por objetos y su aplicación en el método de los elementos finitos*. Relatório de Pesquisa e Desenvolvimento, Nº 15/91, Laboratório Nacional de Computação Científica (LNCC), Rio de Janeiro, 1991.

[48] FIELD, D. A., PRESSBURGER, Y. An $h - p$ multigrid method for finite element analysis. *International Journal for Numerical Methods in Engineering*, v.36, p.893-908, 1993.

[49] FILHO, J. S. R. A., DEVLOO, P. Object oriented programming in scientific computations: the beginning of a new era. *Engineering Computations*, v.8, p.81-87, 1991.

[50] FORDE, B. W. R., FOSCHI, R. B., STIEMER, S. F. Object-oriented finite element analysis. *Computer & Structures*, v.34, p.355-374, 1990.

[51] GALVÃO, M. C. *Ambiente baseado em NURBS para a definição de contornos em malhas de elementos finitos*. Relatório de Iniciação Científica, FAPESP (Proc. 95/1621-9), 1995.

[52] GEORGE, A., LIU, J. W. *Computer solution of large sparse positive definite systems*. Englewood-Cliffs: Prentice-Hall Series in Computational Mathematics, 1981.

[53] GUIMARÃES, A. C. S., FEIJÓO, R. A. *O sistema ACDP*. Relatório de Pesquisa e Desenvolvimento, Nº 027/89, Laboratório Nacional de Computação Científica (LNCC), Rio de Janeiro, Agosto, 1989.

[54] GUIMARÃES, A. C. S., FANCELLO, E. A., FEIJÓO, G. R., FEIJÓO, R. A. *SAFE - integrated system for structural finite element analysis – Version 1.0*. Laboratório Nacional de Computação Científica (LNCC), Rio de Janeiro, 1994.

[55] GOLUB, G. H., VAN LOAN, C. F. *Matrix computations*. Segunda edição. Baltimore: The Johns Hopkins University Press, 1989.

[56] HACKBUSH, W. A fast iterative method for solving Poisson's equation in general region. *Numerical Treatment of Differential Equations (R. Bulirsch et al., editores), Lectures Notes in Mathematics*. Berlin: Springer-Verlag, 1977.

[57] HACKBUSH, W., TROTTENBERG, U. *Multigrid methods*. Berlin: Springer-Verlag, 1982.

[58] HACKBUSH, W. *Multigrid methods and applications*. Berlin: Springer, 1985.

[59] HAGEMAN, L. A., YOUNG, D. M. *Applied iterative methods*. New York: Academic Press, 1981.

[60] KOOBUS, B., LALLEMAND, M., DERVIEUX, A. Unstructured volume-agglomeration MG: solution of the Poisson equation. *International Journal for Numerical Methods in Fluids*, v.18, p.27-42, 1994.

[61] KŎVARA, M. An adaptive multigrid technique for three-dimensional elasticity. *International Journal for Numerical Methods in Engineering*, v.36, p.1703-1716, 1994.

[62] LAWRENCE. K. L., NAMBIAR, R. V., BERGMANN, B. Closed form stifness matrices and error estimators for plane hierachic triangular elements. *International Journal for Numerical Methods in Engineering,* v.31, p.879-894, 1991.

[63] LEBON, F., RAOUS, M., LATIL, J. C., GREGO, L. Multigrid method in non linear structure mechanics. In: EUROPEAN CONFERENCE ON NEW ADVANCES IN COMPUTATIONAL STRUCTURAL MECHANICS, 1991, Geens.

[64] LEINEN, P. Data structures and concepts for adaptive finite element methods. *Computing,* v.55, p.325-354, 1995.

[65] LIPPMAN, S. B. *C++ primer*. Reading: Addison-Wesley, 1991.

[66] LIU, J. W., SHERMAN, A. H. Comparative analysis of the Cuthill-McKee and the reverse Cuthill-McKee ordering algorithms for sparse matrices. *SIAM Journal of Numerical Analysis,* v.13, p.198-213, 1977.

[67] LÖHNER, R., MORGAN, K. An unstructured multigrid method for elliptic problems. *International Journal for Numerical Methods in Engineering,* v.24, p.101-115, 1987.

[68] MAVRIPLIS, D. J. Multigrid solution of the two-dimensional Euler equations on unstructured triangular meshes. *AIAA Journal,* v.26, n.7, 1987.

[69] MCCORMICK, S. F. *Multigrid methods.* Philadelphia: SIAM Frontiers Series, v.3, 1987.

[70] MCCORMICK, S. F. *Multilevel adaptive methods for partial differential equations.* Philadelphia: SIAM Frontiers in Applied Mathematics, v.6, 1989.

[71] MCCORMICK, S. F., THOMAS, J. The fast adaptive composite grid (fac) method for elliptic problems. *Mathematics of Computation,* v.46, n.174, p.439-456, 1986.

[72] MENÉTREY, PH., ZIMMERMANN, T. Object-oriented non-linear finite element analysis: application to j2 plasticity. *Computer & Structures,* v.49, n.5, p.767-777, 1993.

[73] MILLER, G. R. An object-oriented approach to structural analysis and design. *Computer & Structures,* v.40, p.75-82, 1991.

[74] MULLER, A., HUGHES, T. J. R. Precondicionadores elemento-por-elemento y globales. Una perspectiva. *Revista Internacional de Métdos Numéricos para Cálculo y Diseño en Ingeniería,* v.2, n.1, p.27-41, 1986.

[75] NICOLAIDES, R. A. On multiple grid and related techniques for solving discrete elliptic systems. *Journal of Computational Physics,* v.19, p.418-431, 1975.

[76] NICOLAIDES, R. A. On the l^2 convergence of an algorithm for solving finite element equations. *Mathematics of Computation,* v.31, n.140, p.892-906, 1977.

[77] NICOLAIDES, R. A. On some theoretical and practical aspects of multigrid methods. *Mathematics of Computation,* v.33, n.147, p.933-952, 1979.

[78] ODEN, J. T. *Applied functional analysis.* Englewood-Cliffs: Prentice-Hall, Inc., 1979.

[79] PADDON, D. J., HOLSTEIN, H. (Editors). *Multigrid methods for integral and differential equations*. Clarendon Press-Oxford, 1985.

[80] RAMOS, S. H. P. *Visualização de resultados de engenharia estrutural em ambiente windows.* Relatório de Iniciação Científica, FAPESP (Proc. 95/4948-9), 1996.

[81] PERAIRE, J., PEIRO, J., MORGAN, K. Multigrid solution of the 3D compressible Euler equations on unstructured tetrahedral grids. *International Journal for Numerical Methods in Engineering,* v.36, p.1029-1044, 1993.

[82] PISSANETZKY, S. *Sparse matrix technology.* London, Orlando, San Diego: Academic Press, Inc. Harcout Brace Jovanovich Publishers, 1984.

[83] POPOV, E. P. *Engineering mechanics of solids.* London: Prentice-Hall International Editions, 1990.

[84] PRESSBURGER, Y., PERUCCHIO, R. A hierarchical two-level multigrid solver. *Computer & Structures,* v.55, n.3, p.471-483, 1995.

[85] PRESSBURGER, Y., PERUCCHIO, R. An self adaptive fe system based on recursive spatial decomposition and multigrid analysis. *International Journal for Numerical Methods in Engineering,* v.38, p.1399-1421, 1995.

[86] PRESSMAN, R. S. *Software engineering - a practitioner's approach.* New York: McGraw-Hill, 1987.

[87] RAPHAEL, B., KRISHNAMOORTHY, C. S. Automating finite element development using object-oriented techniques. *Engineering Computations,* v.10, p.267-278, 1993.

[88] REKTORYS, K. *Variational methods in mathematics, science and engineering.* Dordrecht-Holland: D. Reidel Publishing Company, Segunda Edição, 1977.

[89] ROSS, T. J., WAGNER, L. R., LUGER, G. F. Object-oriented programming for scientific codes. II: examples in C++. *Journal of Computing in Civil Engineering,* v.6, n.4, p.497-514, 1992.

[90] RÜDE, U. The hierarchical basis extrapolation method. *SIAM Journal on Scientific and Statistical Computing,* v.13, n.1, p.307-318, 1992.

[91] RÜDE, U. Fully adaptive multigrid methods. *SIAM Journal on Numerical Analysis,* v.30, n.1, p.230-248, 1993.

[92] SAMARSKII, A. A., NIKOLAEV, E. S. *Numerical methods for grid equations - v.1/2.* Basel: Birkhauser, 1989.

[93] SCHOLZ, S. P. Elements of an object-oriented fem++ program in C++. *Computer & Structures,* v.43, n.3, p.517-529, 1992.

[94] SCOTT, L. R., ZHANG, S. Higher-dimensional nonnested multigrid methods. *Mathematics of Computation,* v. 58, n. 198, p.457-466, 1992.

[95] SKALICKý, T. *LASPack reference manual - Version 1.12.* Dresden University of Technology, 1995.

[96] SOUTHWELL, R. V. *Relaxation Methods in Engineering Science.* Oxford: Oxford University Press, 1940.

[97] TAYLOR, V. E., NOUR-OMID, B. A study of the factorization fill-in for a parallel implementation of the finite element method. *International Journal for Numerical Methods in Engineering,* v.37, p.3809-3823, 1994.

[98] THEES, A. M. *Desenvolvimento de classe de banco de dados e conversor de arquivos de geometria para programas de análise estrutural.* Relatório de Iniciação Científica, CNPq, 1995.

[99] THEES, A. M. *Ferramentas computacionais para um programa de análise estrutural por elementos finitos.* Relatório de Iniciação Científica, FAPESP (Proc. 95/3148-9), 1996.

[100] TWORZYDLO, W. W., ODEN, J. T. Towards an automated environment in computational mechanics. *Computer Methods in Applied Mechanics and Engineering,* v.104, p.87-143, 1993.

[101] TWORZYDLO, W. W., ODEN, J. T. Knowledge-based methods and smart algorithms in computational mechanics. *Engineering Fracture Mechanics,* v.50, n.5-6, p.759-800, 1995.

[102] VARGA, R. *Matrix iterative analysis.* Englewood-Cliffs: Prentice-Hall, 1962.

[103] VERFÜRTH, R. Posteriori error estimation and adaptive mesh-refinement techniques. *Journal of Computational and Applied Mathematics,* v.50, n.1-3, p.67-83, 1994.

[104] YU, G, ADELI, H. Object-oriented finite element analysis using eer model. *Journal of Structural Engineering,* v.119, n.9, p.2763-2781, 1993.

[105] WIBERG, N., ABDULWAHAB, F. Patch recovery based on superconvergent derivatives and equilibrium. *International Journal for Numerical Methods in Engineering,* v.36, p.2703-2724, 1993.

[106] WIBERG, N., ABDULWAHAB, F., ZIUKAS, S. Enhanced superconvergent patch recovery incorporating equilibrium and boundary conditions. *International Journal for Numerical Methods in Engineering,* v.37, p.3417-3440, 1994.

[107] ZEGLINSKI, G. W., HAN, R. P. S., AITCHISON, P. Object oriented matrix classes for use in a finite element code using C++. *International Journal for Numerical Methods in Engineering,* v.37, p.3921-3937, 1994.

[108] ZHANG, S. *Multi-level iterative techniques.* Pennsylvania: Dept. of Mathematics, Pennstate University, 1988, Tese (PhD).

[109] ZHANG, S. Optimal-order nonnested multigrid methods for solving finite element equations I: on quasi-uniform meshes. *Mathematics of Computation,* v.55, n. 191, p.23-26, 1990.

[110] ZHANG, S. Optimal-order nonnested multigrid methods for solving finite element equations II: on non-quasi-uniform meshes. *Mathematics of Computation,* v.55, n.192, p.439-450, 1990.

[111] ZHU, J. Z., ZIENKIEWICZ, O. C. Superconvergence recovery technique and *a posteriori* error estimators. *International Journal for Numerical Methods in Engineering,* v.30, p.1321-1339, 1990.

[112] ZIENKIEWICZ, O. C., ZHU, J. Z. A simple error estimator and adaptative procedure for practical engineering analysis. *International Journal for Numerical Methods in Engineering,* v.24, p.337-357, 1987.

[113] ZIENKIEWICZ, O. C., LIU, Y. C., HUANG, G. C. Error estimation and adaptivity in flow formulation for forming problems. *International Journal for Numerical Methods in Engineering,* v.25, p.23-42, 1988.

[114] ZIENKIEWICZ, O. C., ZHU, J. Z., GONG, N. G. Effective and practical *h-p* version adaptive analysis procedures for the finite element method. *International Journal for Numerical Methods in Engineering,* v.28, p.879-891, 1989.

[115] ZIENKIEWICZ, O. C., ZHU, J. Z. Error estimates and adaptive refinement for plate bending problems. *International Journal for Numerical Methods in Engineering*, v.28, p.2839-2853, 1989.

[116] ZIENKIEWICZ, O. C., ZHU, J. Z. The three R's of engineering analysis and error estimation and adaptivity. *Computer Methods in Applied Mechanics and Engineering,* v.82, p.95-113, 1990.

[117] ZIENKIEWICZ, O. C., ZHU, J. Z. The superconvergent patch recovery and *a posteriori* error estimates. part 1: the recovery technique. *International Journal for Numerical Methods in Engineering,* v.33, p.1331-1364, 1992.

[118] ZIENKIEWICZ, O. C., ZHU, J. Z. The superconvergent patch recovery and *a posteriori* error estimates. part 2: error estimates and adaptivity. *International Journal for Numerical Methods in Engineering,* v.33, p.1365-1382, 1992.

[119] ZIENKIEWICZ, O. C., ZHU, J. Z, WU, J. Superconvergent patch recovery techniques - some further tests. *Communications in Numerical Methods in Engineering,* v.9, p.251-258, 1993.

[120] ZIMMERMANN, T., DUBOIS-PÈLERIN, Y., BOMME, P. Object-oriented finite element programming: I. govening principles. *Computer Methods in Applied Mechanics and Engineering,* v.98, p.291-303, 1992.

Apêndice A

CONCEITOS DE PROGRAMAÇÃO POR OBJETOS

O modelo orientado por objetos, de maneira análoga aos vários paradigmas de programação, introduz alguns conceitos particulares, tais como objetos, mensagens, mecanismo de herança, dentre outros. Ressalta-se, porém, que nem todas as linguagens orientadas por objetos implementam todos os conceitos a serem descritos. Assim, a apresentação feita a seguir tem um caráter mais geral, não procurando destacar as particularidades de uma linguagem [18].

A.1 Modulação

O conceito de modulação é de certa forma intuitivo, ou seja: dado um problema complexo, subdivide-se o mesmo em subproblemas menos complexos visando obter a solução global.

No entanto, a aplicação do conceito de modulação não pode ser realizada indefinidamente no desenvolvimento de um programa. Ao mesmo tempo em que o esforço de desenvolvimento diminui substancialmente quando se aumenta o número de módulos, o esforço de interfaceamento destes módulos cresce na mesma proporção.

Alguns tipos de módulos são encontrados em linguagens de programação como por exemplo a declaração `class` em C++. Este módulos são componentes de programas que combinam abstrações de dados e procedimentos, incentivando assim, o desenvolvimento de programas modulares.

A.2 Ocultamento de Informação

A aplicação do conceito de ocultamento de informação no desenvolvimento de um programa permite construir módulos onde a interdependência entre os mesmos é pequena. Além disso, aumenta-se a confiabilidade e as modificações são efetuadas localmente dentro de cada módulo, preservando assim, a disseminação das alterações ao longo de todo o sistema.

Para se alcançar uma modulação efetiva, define-se um conjunto de módulos interdependentes, os quais se comunicam entre si apenas através das informações necessárias para acessar uma característica do módulo ou executar uma de suas operações. O estado de um módulo é descrito por variáveis locais, visíveis apenas dentro do escopo deste módulo, e um conjunto de procedimentos que manipulam estes dados. Observa-se que o uso de abstrações de dados permite definir e utilizar o conceito de ocultamento de informação no desenvolvimento de programas.

A.3 Abstração

Para problemas onde se aplicam o conceito de modulação, vários níveis de abstração podem ser considerados. No nível mais alto de abstração, a solução adotada para o problema é colocada em termos de uma linguagem próxima ao ambiente do problema. Nos níveis mais baixos, descrevem-se os procedimetos a serem implementados. Assim, o uso de abstração permite ao programador concentrar-se em um problema, considerando um nível de generalização qualquer sem se preocupar com detalhes irrelevantes ao problema.

Várias linguagens de programação, tais como *Ada*, *Modula* e *Smalltalk*, permitem a criação de tipos abstratos de dados, os quais consistem de uma representação interna para os dados e um conjunto de procedimentos para acessar e manipular os dados.

A.4 Ligação Dinâmica

Nas linguagens convencionais, tais como *C*, *Pascal* e *FORTRAN*, a partir do conhecimento dos tipos de variáveis e constantes utilizadas em uma declaração, o compilador gera o código de máquina correspondente. Este processo de definir os tipos de dados aplicáveis a um operador em uma declaração, anterior a sua execução, é denominado *ligação estática*.

Entretanto, algumas linguagens, como por exemplo C++, permite a flexibilidade de redefinir operadores convencionais para os tipos declarados pelo usuário. Assim, pode-se definir o tipo `Vetor` onde a soma de dois vetores `A` e `B` é expressa por `A+B`. No entanto, esta ligação entre tipos e operandos é realizada ainda, em tempo de compilação do código fonte.

A ligação dinâmica permite associar um tipo de dado a um operando em tempo de execução do programa. Em C++, esta característica está implementada através de declarações do tipo `virtual`.

A.5 Objeto

Um objeto se constitui na unidade básica de modulação no modelo orientado por objetos, constituindo-se um tipo abstrato de dados.

Para se manipular a informação representada por um objeto, deve-se solicitar ao mesmo que execute uma de suas operações, através do envio de uma mensagem. O objeto que recebe a mensagem é denominado receptor, devendo responder esta mensagem através da seleção da função correspondente, executar esta operação e retornar o controle para o objeto emissor da mensagem.

A.6 Mensagens

As mensagens se constituem nas especificações das operações de um objeto. Assim, quando um objeto recebe uma mensagem, deve determinar como manipulá-la para obter a resposta requerida. Uma mensagem inclui um seletor, descrevendo o tipo de manipulação desejada, e argumentos, os quais podem ser outros objetos ou valores das variáveis de um objeto.

A característica principal do mecanismo de mensagem é que o seletor é um nome de uma operação, descrevendo apenas a ação a ser executada. Portanto, uma mesma mensagem pode ser interpretada de maneiras distintas.

Assim, por exemplo, considere as classes `Vector` e `String`. Definindo-se os objetos `Vector A,B` e `String Str1,Str2`. As instruções `A+B` e `Str1+Str2`, apesar de utilizar o mesmo operador, possuem significados diferentes.

A.7 Classes e Instâncias

Vários sistemas orientados por objetos fazem uma distinção entre um objeto e a sua descrição. Assim, definem-se os conceitos de classe e instância.

Uma classe é uma descrição geral de um conjunto de objetos semelhantes, provendo todas as informações necessárias para a criação e utilização dos objetos. Uma instância, por sua vez, é um objeto descrito por uma classe particular. Cada objeto é instância de uma classe. Assim, no exemplo anterior `A,B` são instâncias da classe `Vector`.

Todas as instâncias de uma classe utilizam o mesmo método para responder a uma mensagem particular. A diferença na resposta obtida para duas instâncias distintas é resultado dos diferentes valores armazenados nas variáveis.

Portanto, pode-se dizer que um sistema orientado por objetos é desenvolvido a partir da criação das classes que descrevem os objetos constituintes do sistema.

A.8 Métodos

Os métodos são procedimentos invocados pelo envio de mensagens para as instâncias de uma classe. Portanto, um método, como um procedimento, é a descrição de uma sequência de ações a serem executadas. De forma análoga aos procedimentos, os métodos devem conhecer os tipos de dados que manipulam.

A.9 Mecanismo de Herança

O mecanismo de herança permite compartilhar as informações entre objetos. Supondo então, uma hierarquia, tem-se que objetos situados em um nível inferior desta hierarquia herdam todas as características (dados e operações) dos objetos situados em níveis superiores.

A maioria das linguagens orientadas por objetos implementam o mecanismo de herança entre as classes do sistema. Uma classe pode ser alterada para criar uma outra. Nesta relação, a primeira classe é denominada *superclasse* e a segunda *subclasse*. Uma subclasse pode adicionar novas variáveis e métodos, assim como redefinir os dados e operações da superclasse.

Apêndice B

RESULTADOS PARA OS PROBLEMAS PLANOS

Neste apêndice, apresentam-se tabelas com resultados da aplicação dos métodos iterativos aos problemas planos da Figura 2.3 do Capítulo 2. Para cada uma das 12 malhas geradas, tem-se tabelas com valores obtidos nas normas euclideana $||\cdot||_2$ e infinita $||\cdot||_\infty$. A seguinte notação é empregada para os parâmetros apresentados nas tabelas:

- NIT = número de iterações para convergência;
- $||\mathbf{r}||_l$ = norma l ($l = 2, \infty$) do resíduo segundo o critério de convergência (2.129);
- $||e_r^{dir/ite}||_l$ = norma l ($l = 2, \infty$) do erro relativo entre a solução direta e iterativa segundo a equação (2.130);
- ξ = precisão adotada.

As Tabelas B.1 a B.4, B.5 a B.8 e B.9 a B.12 contém, respectivamente, os resultados para o cilindro vertical, placa com furo e problema de fratura.

No caso do método CHSS, o número de iterações para o cálculo das estimativas dos autovalores extremos, via o algoritmo de Lanczos, está dada entre parênteses abaixo da denominação CHSS. Empregou-se uma precisão $\xi = 10^{-4}$ no cálculo destas estimativas, com exceção das malhas 2 a 4 do exemplo de placa com furo onde foi necessário utilizar $\xi = 10^{-6}$.

Método / Parâmetro	GS	SOR	CHSS (89)	GC	GCD	GCSS	GCGS
NIT	1115	116	101	124	102	75	57
$\|\|\mathbf{r}\|\|_2$ (10^{-4})	2.50	9.75	9.91	9.24	9.59	9.15	9.28
$\|\|e_r^{dir/ite}\|\|_2$ (%)	0.0725	0.0651	0.0512	0.0890	0.0408	0.0140	0.0375
ξ (10^{-3})	0.25	1.0	1.0	1.0	1.0	1.0	1.0
NIT	1022	138	108	141	112	73	62
$\|\|\mathbf{r}\|\|_\infty$ (10^{-4})	2.49	2.49	2.40	2.26	1.97	9.06	2.28
$\|\|e_r^{dir/ite}\|\|_\infty$ (%)	0.0518	0.0112	0.0205	0.0481	0.0940	0.0129	0.0112
ξ (10^{-4})	2.5	2.5	2.5	2.5	2.5	2.5	2.5

Tabela B.1: Cilindro vertical (malha 1) - resultados nas normas 2 e ∞.

Método / Parâmetro	SOR	CHSS (150)	GC	GCD	GCSS	GCGS
NIT	337	175	193	162	111	91
$\|\|\mathbf{r}\|\|_2\ (10^{-4})$	9.99	9.87	9.41	9.56	9.47	9.23
$\|\|e_r^{dir/ite}\|\|_2\ (\%)$	0.0720	0.0381	0.0915	0.0461	0.0133	0.0446
$\xi\ (10^{-3})$	1.0	1.0	1.0	1.0	1.0	1.0
NIT	360	183	214	171	107	96
$\|\|\mathbf{r}\|\|_\infty\ (10^{-4})$	2.48	2.48	2.22	2.19	9.45	2.01
$\|\|e_r^{dir/ite}\|\|_\infty\ (\%)$	0.0209	0.0163	0.0578	0.0135	0.0240	0.0138
$\xi\ (10^{-4})$	2.5	2.5	2.5	2.5	10	2.5

Tabela B.2: Cilindro vertical (malha 2) - resultados nas normas 2 e ∞.

Método / Parâmetro	SOR	CHSS (203)	GC	GCD	GCSS	GCGS
NIT	677	257	265	187	165	102
$\|\|\mathbf{r}\|\|_2\ (10^{-4})$	9.97	9.98	9.68	9.90	9.78	9.52
$\|\|e_r^{dir/ite}\|\|_2\ (\%)$	0.0832	0.0243	0.0542	0.0431	0.0254	0.0602
$\xi\ (10^{-3})$	1.0	1.0	1.0	1.0	1.0	1.0
NIT	574	229	243	164	159	92
$\|\|\mathbf{r}\|\|_\infty\ (10^{-4})$	9.97	8.11	7.64	9.95	8.56	7.69
$\|\|e_r^{dir/ite}\|\|_\infty\ (\%)$	0.0924	0.0929	0.0484	0.0767	0.0275	0.0507
$\xi\ (10^{-3})$	1.0	1.0	1.0	1.0	1.0	1.0

Tabela B.3: Cilindro vertical (malha 3) - resultados nas normas 2 e ∞.

Método / Parâmetro	SOR	CHSS (203)	GC	GCD	GCSS	GCGS
NIT	1121	348	355	245	220	132
$\|\|\mathbf{r}\|\|_2\ (10^{-4})$	9.98	9.99	9.83	9.64	9.78	9.82
$\|\|e_r^{dir/ite}\|\|_2\ (\%)$	0.0999	0.0258	0.0996	0.0466	0.0249	0.0557
$\xi\ (10^{-3})$	1.0	1.0	0.25	1.0	1.0	1.0
NIT	1265	360	304	208	210	112
$\|\|\mathbf{r}\|\|_\infty\ (10^{-4})$	2.49	2.43	9.67	9.72	9.02	9.59
$\|\|e_r^{dir/ite}\|\|_\infty\ (\%)$	0.0266	0.0180	0.0820	0.0578	0.0291	0.0770
$\xi\ (10^{-3})$	0.25	0.25	1.0	1.0	1.0	1.0

Tabela B.4: Cilindro vertical (malha 4) - resultados nas normas 2 e ∞.

Método / Parâmetro	GS	SOR	CHSS (114)	GC	GCD	GCSS	GCGS
NIT	2648	287	127	138	129	64	73
$\|\|\mathbf{r}\|\|_2\ (10^{-4})$	1.00	2.50	9.33	8.70	9.29	9.98	9.53
$\|\|e_r^{dir/ite}\|\|_2\ (\%)$	0.0594	0.0341	0.0589	0.0273	0.0246	0.0153	0.0247
$\xi\ (10^{-3})$	0.1	0.25	1.0	1.0	1.0	1.0	10^{-3}
NIT	2934	237	128	139	130	59	68
$\|\|\mathbf{r}\|\|_\infty\ (10^{-4})$	0.00997	2.49	2.37	2.14	2.25	8.36	7.88
$\|\|e_r^{dir/ite}\|\|_\infty\ (\%)$	0.0202	0.0943	0.0651	0.0300	0.0236	0.0444	0.0783
$\xi\ (10^{-4})$	0.1	2.5	2.5	2.5	2.5	10	10

Tabela B.5: Placa com furo (malha 1) - resultados nas normas 2 e ∞.

Método / Parâmetro	SOR	CHSS (214)	GC	GCD	GCSS	GCGS
NIT	1045	228	247	203	142	109
$\|\|\mathbf{r}\|\|_2$ (10^{-4})	0.993	9.99	9.92	9.97	9.81	9.30
$\|\|e_r^{dir/ite}\|\|_2$ (%)	0.0120	0.0579	0.0214	0.0613	0.0116	0.0277
ξ (10^{-3})	0.1	1.0	1.0	1.0	1.0	1.0
NIT	892	242	255	205	136	100
$\|\|\mathbf{r}\|\|_\infty$ (10^{-4})	0.995	2.35	2.12	2.28	8.38	7.50
$\|\|e_r^{dir/ite}\|\|_\infty$ (%)	0.0317	0.0239	0.0130	0.0874	0.0212	0.0663
ξ (10^{-3})	0.1	0.25	0.25	0.25	1.0	1.0

Tabela B.6: Placa com furo (malha 2) - resultados nas normas 2 e ∞.

Método / Parâmetro	SOR	CHSS (306)	GC	GCD	GCSS	GCGS
NIT	1783	317	351	285	204	157
$\|\|\mathbf{r}\|\|_2$ (10^{-4})	2.50	9.89	9.93	9.69	9.76	9.60
$\|\|e_r^{dir/ite}\|\|_2$ (%)	0.0360	0.0868	0.0179	0.0231	0.0119	0.0282
ξ (10^{-3})	0.25	1.0	1.0	1.0	1.0	1.0
NIT	1495	363	347	258	173	160
$\|\|\mathbf{r}\|\|_\infty$ (10^{-4})	2.50	0.930	4.12	7.70	9.92	2.46
$\|\|e_r^{dir/ite}\|\|_\infty$ (%)	0.0935	0.0159	0.0367	0.0949	0.0921	0.0278
ξ (10^{-4})	2.5	1.0	5.0	10	10	2.5

Tabela B.7: Placa com furo (malha 3) - - resultados nas normas 2 e ∞.

Método / Parâmetro	SOR	CHSS (398)	GC	GCD	GCSS	GCGS
NIT	3040	481	448	429	257	236
$\|\|\mathbf{r}\|\|_2$ (10^{-4})	2.50	2.49	9.90	9.72	9.55	9.91
$\|\|e_r^{dir/ite}\|\|_2$ (%)	0.0389	0.0179	0.0136	0.0145	0.0170	0.0154
ξ (10^{-3})	0.25	0.25	1.0	1.0	1.0	1.0
NIT	2818	414	438	424	221	237
$\|\|\mathbf{r}\|\|_\infty$ (10^{-4})	0.998	2.17	6.68	2.49	8.43	2.13
$\|\|e_r^{dir/ite}\|\|_\infty$ (%)	0.0518	0.0665	0.0584	0.0397	0.0809	0.0286
ξ (10^{-4})	1.0	2.5	10	2.5	10	2.5

Tabela B.8: Placa com furo (malha 4) - resultados nas normas 2 e ∞.

Método / Parâmetro	GS	SOR	CHSS (94)	GC	GCD	GCSS	GCGS
NIT	3432	369	147	153	140	86	79
$\|\|\mathbf{r}\|\|_2$ (10^{-4})	0.998	2.49	9.63	9.73	9.80	9.78	9.81
$\|\|e_r^{dir/ite}\|\|_2$ (%)	0.0591	0.0304	0.0404	0.0197	0.0255	0.0111	0.0257
ξ (10^{-3})	0.1	0.25	1.0	1.0	1.0	1.0	1.0
NIT	3061	305	137	137	125	80	72
$\|\|\mathbf{r}\|\|_\infty$ (10^{-4})	0.499	2.46	4.01	9.06	9.52	9.98	7.84
$\|\|e_r^{dir/ite}\|\|_\infty$ (%)	0.0746	0.0752	0.0525	0.0729	0.0705	0.0302	0.0528
ξ (10^{-3})	0.05	0.25	0.5	1.0	1.0	1.0	1.0

Tabela B.9: Fratura (malha 1) - resultados nas normas 2 e ∞.

Método / Parâmetro	SOR	CHSS (147)	GC	GCD	GCSS	GCGS
NIT	1139	252	250	237	139	131
$\|\|\mathbf{r}\|\|_2\ (10^{-4})$	2.50	9.56	9.76	9.25	48.2	9.59
$\|\|e_r^{dir/ite}\|\|_2\ (\%)$	0.0376	0.0494	0.0169	0.0177	0.0512	0.0203
$\xi\ (10^{-3})$	0.25	1.0	1.0	1.0	5.0	1.0
NIT	1070	254	236	219	150	120
$\|\|\mathbf{r}\|\|_\infty\ (10^{-4})$	0.994	2.41	8.93	6.52	7.93	7.99
$\|\|e_r^{dir/ite}\|\|_\infty\ (\%)$	0.0378	0.0377	0.0475	0.0496	0.0238	0.0613
$\xi\ (10^{-3})$	0.1	0.25	1.0	1.0	1.0	1.0

Tabela B.10: Fratura (malha 2) - resultados nas normas 2 e ∞.

Método / Parâmetro	SOR	CHSS (215)	GC	GCD	GCSS	GCGS
NIT	2242	373	378	349	207	191
$\|\|\mathbf{r}\|\|_2\ (10^{-4})$	2.50	9.56	9.93	9.57	9.70	9.24
$\|\|e_r^{dir/ite}\|\|_2\ (\%)$	0.0480	0.0308	0.0170	0.0116	0.0118	0.0123
$\xi\ (10^{-3})$	0.25	1.0	1.0	1.0	1.0	1.0
NIT	2112	339	354	329	191	180
$\|\|\mathbf{r}\|\|_\infty\ (10^{-4})$	0.998	4.89	9.38	8.90	9.02	8.45
$\|\|e_r^{dir/ite}\|\|_\infty\ (\%)$	0.0451	0.0485	0.0438	0.0365	0.0344	0.0384
$\xi\ (10^{-3})$	0.1	0.5	1.0	1.0	1.0	1.0

Tabela B.11: Fratura (malha 3) - resultados nas normas 2 e ∞.

Método / Parâmetro	SOR	CHSS (244)	GC	GCD	GCSS	GCGS
NIT	3704	476	477	447	251	244
$\|\|\mathbf{r}\|\|_2\ (10^{-4})$	2.50	9.77	9.90	9.50	24.8	9.50
$\|\|e_r^{dir/ite}\|\|_2\ (\%)$	0.0504	0.0325	0.0156	0.0152	0.0283	0.0177
$\xi\ (10^{-3})$	0.25	1.0	1.0	1.0	0.25	1.0
NIT	3404	427	438	420	246	226
$\|\|\mathbf{r}\|\|_\infty\ (10^{-4})$	0.998	4.78	9.59	7.93	9.91	8.65
$\|\|e_r^{dir/ite}\|\|_\infty\ (\%)$	0.0598	0.0629	0.0664	0.0503	0.0285	0.0626
$\xi\ (10^{-3})$	0.1	0.5	1.0	1.0	1.0	1.0

Tabela B.12: Fratura (malha 4) - resultados nas normas 2 e ∞.