COPPE/UFRJ

IMPLEMENTAÇÃO EM PARALELO DO MÉTODO DE ARNOLDI/LANCZOS COM REINÍCIO IMPLÍCITO

George Oliveira Ainsworth Jr.

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia Civil, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia Civil.

Orientadores: Carlos Magluta

Fernando Luiz Bastos Ribeiro

Rio de Janeiro

Março de 2009

IMPLEMENTAÇÃO EM PARALELO DO MÉTODO DE ARNOLDI/LANCZOS COM REINÍCIO IMPLÍCITO

George Oliveira Ainsworth Jr.

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA CIVIL.

Aprovada por:

Prof. Carlos Magluta, D.Sc.

Prof. Fernando Luiz Bastos Ribeiro, D.Sc.

Prof. José Cláudio de Faria Telles, Ph.D.

Prof. Webe João Mansur, Ph.D.

Prof. Paulo Batista Gonçalves, D.Sc.

Prof. Philippe Remy Bernard Devloo, Ph.D.

RIO DE JANEIRO, RJ – BRASIL

MARÇO DE 2009

Ainsworth Jr., George Oliveira

Implementação em paralelo do método de Arnoldi/Lanczos com reinício implícito/George Oliveira Ainsworth Jr.. – Rio de Janeiro: UFRJ/COPPE, 2009.

XV, 96 p.: il.; 29, 7cm.

Orientadores: Carlos Magluta

Fernando Luiz Bastos Ribeiro

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia Civil, 2009.

Referências Bibliográficas: p. 89 – 96.

1. Método de Arnoldi/Lanczos. 2. Problemas de Autovalor. 3. Computação Paralela. 4. Elementos Finitos. I. Magluta, Carlos *et al.*. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Civil. III. Título.

“ O Freunde, nicht diese Töne!”

L. van Beethoven na sinfonia número 9

Op. 125.

À minha mãe pelo dom da vida e pelo amparo ao longo desses anos.
*Às minhas tias Vanete e Vanilde (*in memoriam*).*

Agradecimentos

Aos professores Fernando Bastos Ribeiro e Carlos Magluta pela orientação e paciência ao longo do doutorado.

Aos professores Roberto Fernandes e Webe Mansur pelo imprescindível auxílio na minha formação e disponibilidade.

À todos os funcionários da secretaria do PEC e principalmente à Jairo e Rita, a vocês minhas sinceras desculpas pela minha negligência com prazos e documentos. Meus agradecimentos também a Thelmo, Célio e Orlando pela ajuda nas diversas horas.

À Ábia Jamni pelo companheirismo e lealdade, principalmente nos últimos anos do doutorado.

À meu irmão Igor, que mesmo de longe, por sempre estar torcendo por mim.

À Pedrinho e Naty pelos momentos de descontração e suas inocências infantis.

Aos meus primos Lenize, Eric e Rick, obrigado pelo carinho demonstrado.

Ao meu grande amigo Vicente Helano: como bônus, o doutorado me deu a sua preciosa amizade!

Aos colegas que dividiram apartamento comigo ao longo do doutorado: Danilo, Euler Wagner e Gustavo Domingos.

Aos colegas de mestrado: Adcleides, Daniel, Emerson e Tiagão, que grandes companheiros são vocês!

Aos eternos amigos do tempo de graduação: Gabriel, Darman, Euler, Van, Fernando Torresmus, Smera Duarte e Renato Nucaeman.

Aos amigos de Feira de Santana, meu muito obrigado pelo auxílio e companheirismo: Koji, Gadéa e prof. Paulo Roberto.

Aos colegas do Labest: Silvoso, Walber, Guilherme (S.R.N.) e Iuri.

À Elvis Presley e L. van Beethoven pela companhia nas horas mais solitárias.

Não poderia deixar de citar Voltaire: “Que Deus me proteja dos meus amigos. Dos inimigos, cuido eu.”.

Agradeço ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo suporte financeiro.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

IMPLEMENTAÇÃO EM PARALELO DO MÉTODO DE ARNOLDI/LANCZOS COM REINÍCIO IMPLÍCITO

George Oliveira Ainsworth Jr.

Março/2009

Orientadores: Carlos Magluta

Fernando Luiz Bastos Ribeiro

Programa: Engenharia Civil

Este trabalho apresenta uma implementação em paralelo do método de Arnoldi/Lanczos com reinício implícito para a solução de problemas de autovalor discretizados pelo método dos elementos finitos. O método de Arnoldi/Lanczos utiliza um esquema de reinício implícito para melhorar a convergência da faixa do espectro desejada, controlando a ortogonalidade da base do subespaço de Krylov. A implementação é voltada para arquiteturas de memória distribuída, visando a utilização de *clusters* de *pc's*. Na solução em paralelo empregou-se uma estratégia de subdomínio por subdomínio e dois tipos de particionamentos de malha são utilizados: particionamento sem sobreposição e particionamento com sobreposição de domínios. Para o armazenamento dos coeficientes foi utilizado estruturas de dados comprimidas no formato CSRC e CSRC/CSR. Discute-se a paralelização das operações de álgebra linear computacional presentes nos métodos baseados no subespaço de Krylov e no esquema de reinício implícito. Para comprovar a eficiência e aplicabilidade da implementação, mostram-se exemplos numéricos envolvendo problemas simétricos e não-simétricos.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

PARALLEL IMPLEMENTATION OF THE IMPLICITLY RESTARTED ARNOLDI/LANCZOS METHOD

George Oliveira Ainsworth Jr.

March/2009

Advisors: Carlos Magluta

Fernando Luiz Bastos Ribeiro

Department: Civil Engineering

This work presents a parallel implementation of the implicitly restarted Arnoldi/Lanczos method for the solution of eigenproblems approximated by the finite element method. The implicitly restarted Arnoldi/Lanczos uses a restart scheme in order to improve the convergence of the desired portion of the spectrum, maintaining the orthogonality of the Krylov basis. The presented implementation is suitable to be used in distributed memory architectures, specially pc's clusters. In the parallel solution, the subdomain by subdomain approach was implemented and overlapping and non-overlapping mesh partitions were used. Compressed data structures in the formats CSRC and CSRC/CSR were used to store the global matrices coefficients. The parallelization of the operations of numerical linear algebra presented in both Krylov and implicitly restarted methods are discussed. In order to point out the efficiency and applicability of the proposed algorithms, numerical examples are shown.

Sumário

Lista de Figuras **xii**

Lista de Tabelas **xv**

1 Introdução **1**

1.1 Motivação e relevância 1

1.2 Objetivos 4

1.3 Revisão bibliográfica 4

2 O Método de Arnoldi/Lanczos com Reinício Implícito **8**

2.1 O Método de Arnoldi/Lanczos 8

2.1.1 Um Método Clássico para a Redução da Ordem do Problema . 8

2.1.2 O Método de Lanczos 10

2.1.3 O Método de Arnoldi Derivado do Método de Lanczos 13

2.2 Implementação do Algoritmo 15

2.2.1 Transformação Espectral 15

2.2.2 Fatoração de Arnoldi em k passos 16

2.2.3 Processo de Reortogonalização 19

2.2.4 Método de Arnoldi com Reinício Implícito 21

3 Implementação em Paralelo **24**

3.1 Introdução 24

3.2 Particionamento sem sobreposição (“non-overlapping”) 26

3.3 Particionamento com sobreposição (“overlapping”) 28

3.4 Particionamento 1d para grafos de matrizes 30
3.5 Estruturas de dados comprimidas 30
3.6 Operações de níveis 1,2 e 3 em paralelo 35
3.6.1 Paralelização das operações de nível 1 38
3.6.2 Paralelização das operações de nível 2 39
3.6.3 Paralelização das operações de nível 3 41
3.7 Comunicação 42
3.7.1 Comunicação nas operações de nível 1 42
3.7.2 Comunicação nas operações de nível 2 42
3.7.3 Comunicação nas operações de nível 3 45

4 Exemplos Analisados **48**
4.1 Critério de Convergência Adotado 48
4.2 Resultados Obtidos com Execução Sequencial 49
4.2.1 Arquibancada do estádio Mário Filho (Maracanã) 49
4.2.2 Benchmarks da Coleção do Matrix-Market 54
4.3 Resultados obtidos com execução em paralelo 58
4.3.1 Caso 1: problema de elasticidade plana 59
4.3.2 Caso 2: casa de força da UHE de Peixe Angical 68
4.3.3 Caso 3: problema de convecção-difusão 82

5 Conclusões **87**

Referências Bibliográficas **89**

Lista de Figuras

1.1 Participação das arquiteturas SMP, MPP e cluster no top 500 supercomputadores - fonte: top500 website [1] 3

3.1 Esquema de particionamento sem sobreposição 28
3.2 Esquema de particionamento com sobreposição 29
3.3 Estrutura de dados para matrizes retangulares 34
3.4 Mapeamento entre posições do arranjo local p para o buffer de envio f_1 (caso sem sobreposição) 44
3.5 Mapeamento entre posições do arranjo local p para o buffer de envio f_1 (caso com sobreposição) 45
3.6 Mapeamento entre posições do buffer f_2 para o arranjo local (caso sem sobreposição) 46
3.7 Mapeamento entre posições do buffer f_2 para o arranjo local (caso com sobreposição) 47

4.1 Modo de vibração correspondente à $f = 2.6387$ 51
4.2 Modo de vibração correspondente à $f = 3.2587$ 52
4.3 Modo de vibração correspondente à $f = 4.1682$ 53
4.4 Modo de vibração correspondente à $f = 7.1227$ 54
4.5 Autovalores do *benchmark* do problema de convecção-difusão (fonte: site do *Matrix-Market*) 58
4.6 Particionamento em 4 subdomínios 60
4.7 Particionamento em 32 subdomínios 61

4.8 Particionamento em 64 subdomínios 61
4.9 Primeira forma modal $f = 19.59$ Hz 62
4.10 Segunda forma modal $f = 109.78$ Hz 63
4.11 Terceira forma modal $f = 269.35$ Hz 64
4.12 *Speedups* para a montagem de elementos do caso 1. 65
4.13 *Speedups* para o produto matriz-vetor caso 1. 66
4.14 *Speedups* para a etapa de solução do problema de autovalor do caso 1. 66
4.15 *Speedups* para a etapa de solução do sistema de equações (PCG) do caso 1. 67
4.16 *Speedups* para o tempo total de execução do caso 1. 67
4.17 Malha do caso 2 69
4.18 Primeira forma modal 71
4.19 Segunda forma modal 72
4.20 Terceira forma modal 73
4.21 Quarta forma modal 74
4.22 Quinta forma modal 75
4.23 Sexta forma modal 76
4.24 Particionamento da malha do caso 2 em 64 processos 77
4.25 Particionamento da malha do caso 2 em 64 processos (casa de força em detalhe) 78
4.26 *Speedups* para a montagem de elementos do caso 2. 79
4.27 *Speedups* para o produto matriz-vetor caso 2. 79
4.28 *Speedups* para a etapa de solução do problema de autovalor do caso 2. 80
4.29 *Speedups* para a etapa de solução do sistema de equações (PCG) do caso 2. 80
4.30 *Speedups* para o tempo total de execução do caso 2. 81
4.31 Particionamento em 64 subdomínios 83
4.32 *Speedups* para a montagem de elementos do caso 3. 83
4.33 *Speedups* para o produto matriz-vetor caso 3. 84

4.34 *Speedups* para a etapa de solução do problema de autovalor do caso 3. 84
4.35 *Speedups* para a etapa de solução do sistema de equações (GMRES) do caso 3. 85
4.36 *Speedups* para o tempo total de execução do caso 3. 85

Lista de Tabelas

2.1 Comparação entre as diversas implementações do Método de Arnoldi (fonte [2]) 18

4.1 Comparação das frequências naturais obtidas numericamente 50

4.2 Os seis autovalores com maior parte imaginária da matriz de Tolosa (n=2000) calculados com o ARPACK (fonte: [**?**]) 55

4.3 Os seis autovalores com maior parte imaginária da matriz de Tolosa (n=2000) obtidos com a implementação proposta 55

4.4 Os seis autovalores do problema de convecção-difusão ($m = 31$,$p_1 = 25$,$p_2 = 50$ e $p_3 = 250$) 57

4.5 Os seis autovalores do problema de convecção-difusão obtidos numericamente 57

4.6 Informações da simulação do caso 1 60

4.7 *Speedups* máximos obtidos para o caso 1 68

4.8 Propriedades físicas utilizadas no modelo da UHE Peixe Angical . . . 69

4.9 Informações da simulação do caso 2 70

4.10 *Speedups* máximos obtidos para o caso 2 82

4.11 Informações da simulação do caso 3 82

4.12 *Speedups* máximos obtidos para o caso 3 86

Capítulo 1

Introdução

1.1 Motivação e relevância

Diversas aplicações científicas são modeladas através de problemas de autovalor generalizado do tipo:

$$Ax = \lambda Bx \tag{1.1}$$

onde A , $B \in \mathbb{C}^{n\times n}$, λ é o autovalor e $x \neq \mathbf{0}$ é o autovetor. Quando $B = I$ (Matriz Identidade), o problema de autovalor é chamado de problema padrão.

Dentre as áreas que possuem aplicações onde autovalores e autovetores precisam ser calculados, podem-se citar: vibrações de sistemas mecânicos, macro-economia, física quântica, técnicas de cadeia de Markov, reações químicas, estabilidade em hidrodinâmica [2, 3], ranqueamento de páginas de internet [4]. Essas aplicações dão origem a problemas com características distintas no que se refere a simetria e ao tipo do problema, padrão ou generalizado.

Desde Jacobi que em 1846 propôs, apesar de não conhecer a notação matricial, o primeiro algoritmo para cálculo de autovalores e autovetores de problemas auto-adjuntos [5], vários métodos têm sido propostos. A maioria desses métodos baseiam-se num procedimento de Rayleigh-Ritz do tipo:

Algoritmo 1.1 Algoritmo do método de Rayleigh-Ritz para $Ax = \lambda x$

1: Calcule uma matriz ortonormal V cujas colunas gerem o subespaço $\mathcal{S}$ de dimensão k

2: Calcule a projeção de ordem $k \times k$ de A em $\mathcal{S}$: $T = V^T AV$

3: Resolva $Ty = \theta y$

4: $\theta = \lambda$ e $x = Vy$

São exemplos desses métodos: iteração por subespaços [6, 7, 8], que é um método muito difundido na análise estrutural, o método de Jacobi-Davidson [9, 10, 11]. Particularmente, quando o subespaço $\mathcal{S}$ é um subespaço de Krylov, gerado pela base:

$$\left\{v_1, Av_1, A^2v_1, \cdots, A^{k-1}v_1\right\} \tag{1.2}$$

onde, $v_1 \neq \mathbf{0}$, é um vetor arbitrário, k é a ordem do subespaço, e uma projeção ortogonal (condição de Galerkin) é utilizada para calcular V, o algoritmo (1.1) dá origem aos métodos de Arnoldi [12] e Lanczos simétrico [13]. Quando no passo 2 do Algoritmo (1.1) é utilizada uma projeção oblíqua (condição de Petrov-Galerkin), o método resultante é a versão não simétrica do método de Lanczos.

Estes métodos são bastante adequados quando a quantidade de autovalores/autovetores relevantes na análise é muito menor que a ordem n do sistema de equações original. Este é o caso dos problemas físicos que ocorrem em Engenharia. Pacotes de domínio público como ARPACK [14], Anasazi [15] e SLEPc [16], entre outros, ajudaram a popularizar os métodos de Arnoldi e Lanczos.

A complexidade dos problemas da engenharia moderna requer a modelagem de problemas físicos com alto grau de discretização, envolvendo sistemas da ordem de milhões e até dezenas de milhões de equações. Tornam-se necessários, portanto, o desenvolvimento de algoritmos eficientes e o emprego de supercomputadores para viabilizar a solução destes problemas. Nas últimas décadas, com a redução do custo dos computadores pessoais (PC's) e a difusão da tecnologia de rede local de computadores, os clusters de PC's têm tomado conta do cenário que antes era dominado

pelos supercomputadores de processamento paralelo massivo (da sigla em inglês MPP - massive parallel processing) e multiprocessamento simétrico (da sigla em inglês SMP - symmetric multiprocessing), como mostra o gráfico da figura (1.1) :

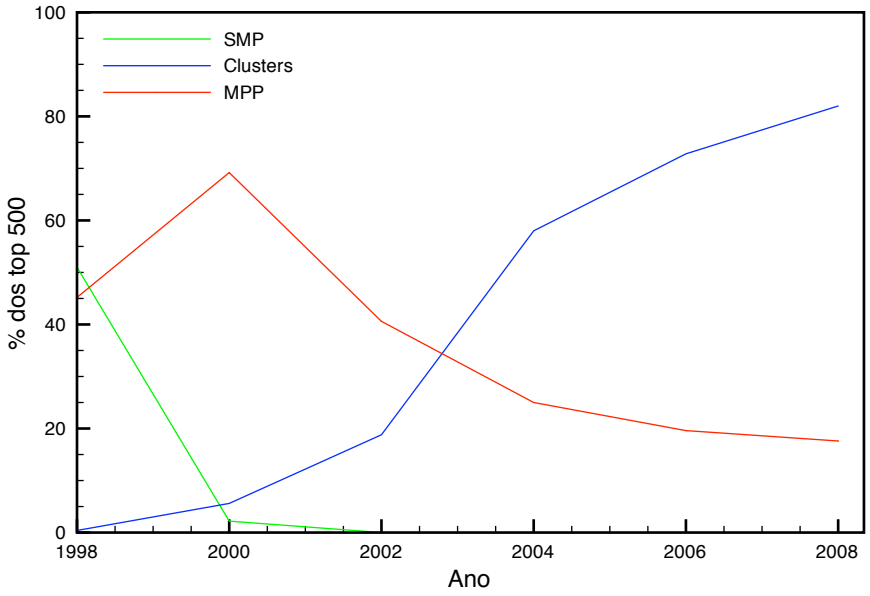

Figura 1.1: Participação das arquiteturas SMP, MPP e cluster no top 500 supercomputadores - fonte: top500 website [1]

Métodos baseados no subespaço de Krylov, por sua característica iterativa, têm se mostrado como uma excelente opção no que se refere à paralelização. Por se basearem em operações algébricas simples envolvendo os três níveis de operações da BLAS (Basic Linear Algebra Set - conjunto básico de álgebra linear) [17, 18, 19] sua paralelização é bastante fácil. Em arquiteturas de memória distribuída, os métodos de Lanczos e Arnoldi podem ser implementados, quando há um domínio associado ao problema, em esquemas de decomposição de domínio [20, 21, 22]. Alternativamente, pode-se empregar técnicas de subdomínio por subdomínio (para a solução de sistemas de equações lineares: [23, 24, 25]). No caso de não haver uma malha associada, métodos de particionamento de grafos de matrizes, tais como particionamentos 1d ou 2d [26] podem ser usados, inclusive em pacotes de domínio público como PARPACK e SLEPc.

1.2 Objetivos

Este trabalho tem como objetivo a implementação em paralelo do método de Arnoldi/Lanczos com reinício implícito para a solução de problemas de autovalor generalizado discretizados pelo método dos elementos finitos. A implementação é voltada a arquiteturas de memória distribuída, visando, principalmente, a utilização de clusters de pc's. Na solução do problema de autovalor em paralelo, adotou-se, de maneira inédita, a estratégia de subdomínio por subdomínio e o emprego de estruturas de dados comprimidas.

Os algoritmos implementados permitem um tratamento geral de problemas simétricos e não-simétricos, abrangendo uma vasta gama de problemas da engenharia moderna que devido à complexos modelos físicos e geométricos requerem discretizações com número de incógnitas de ordem superior a 10^6. O esquema de reinício implícito utilizado é o mais eficiente e estável método empregado para reiniciar a fatoração de Arnoldi/Lanczos, evitando o aumento do número de passos da fatoração e os problemas decorrentes desse aumento. Apresenta-se também a paralelização do esquema de de reortogonalização proposto por Daniel et al. [27] que é um algoritmo que combina eficiência computacional e precisão. Para o armazenamento dos coeficientes derivados da aproximação de elementos finitos, empregam-se estruturas de dados comprimidas para alto desempenho nos formatos CSRC e CSRC/CSR. O primeiro para blocos quadrados de matrizes e o segundo para blocos retangulares, como seá visto no capítulo 3. O modelo de comunicação adotado [24, 23] é baseado no protocolo MPI [28] de troca de mensagens, tratando-se de um esquema ótimo, no sentido que não há comunicação desnecessária entre processos adjacentes.

1.3 Revisão bibliográfica

Os trabalhos pioneiros, ainda nos anos cinquenta de Lanczos [13] e Arnoldi [12] lançaram os fundamentos básicos dos métodos baseados no subespaço de Krylov. Esses trabalhos descreviam processos de projeção de uma matriz no subespaço de

Krylov, onde a matriz resultante era de ordem menor do que o sistema original. Ao final do processo, os autovalores eram calculados via método da bisseção aplicado ao polinômio característico da matriz projetada.

Em termos práticos, esses algoritmos não tiveram grande aplicabilidade, pois em muitas aplicações, os autovalores relevantes na análise aparecem fora de ordem e por isso uma grande quantidade de passos precisam ser computados implicando em perda de ortogonalidade dos vetores da base do subespaço. A perda de ortogonalidade está associada a erros de truncamento e, principalmente, segundo Wilkinson [29], a convergência dos valores de Ritz (isso é facilmente demonstrado através de um argumento simples como será mostrado no capítulo 2). Paige [30], quase vinte anos após Lanczos e Arnoldi, propôs o uso de processos de reortogonalização utilizando os autovetores dos passos já computados. Um aspecto que deve ser levado em conta nesta metodologia é que os autovetores podem não estar suficientemente aproximados. Como alternativa, Daniel e outros propuseram uma técnica [27] que usa os vetores da base já computados, corrigindo a falta de ortogonalidade.

Além do problema de ortogonalidade, a quantidade de passos a serem calculados ainda era um empecilho ao desenvolvimento dos métodos de Arnoldi e Lanczos. O custo de armazenamento dos vetores da base do subespaço de Krylov eram proibitivos, principalmente numa época onde os computadores não dispunham de muita memória. Com o entendimento das propriedades do subespaço de Krylov e sua relação com a minização de $||p(A)v||_2$ (Ederlyi, [31]), Manteuffel [32] foi o primeiro a usar técnicas de aceleração polinomial na solução de sistemas de equações algébricas. Saad [33], baseando-se nessas técnicas, propôs um algoritmo iterativo baseado no método de Arnoldi, onde a iteração era reiniciada através de um vetor de partida pré-condicionado por um polinômio de Tchebychev ou um polinômio formado por uma combinação linear dos vetores de Ritz associados aos valores de Ritz da porção desejada do espectro. Essas técnicas de reinício (reinício explícito) possibilitaram manter o número de passos da fatoração de Arnoldi/Lanczos num tamanho pequeno, ao mesmo tempo que a contribuição dos autovetores desejados na aproximação era

incrementada. Ainda nos anos 80, foram publicados importantes trabalhos sobre transformação espectral, cujo objetivo é a melhoria da convergência dos autovalores e autovetores. Vale destacar Ericsson e Ruhe [34] e Nour-Omid e outros [35].

Sorensen [36], por sua vez, sugeriu que ao invés de reiniciar a iteração com os polinômios sugeridos por Saad, poderia-se usar uma combinação da fatoração de Arnoldi/Lanczos em k passos (para $k \ll$ que a ordem do sistema) com o *implicitly shifted QR* [37, 38]. A combinação desses dois algoritmos dá origem ao método de Arnoldi/Lanczos com reinício implícito. Lehoucq [39] analisou e propôs a primeira implementação desse algoritmo em 1995 que posteriormente deu origem ao ARPACK [14]. O próprio Lehoucq [40, 41] fez estudos importantes sobre o comportamento de convergência do método de Arnoldi e sua relação com a iteração por subespaços.

Na COPPE, alguns trabalhos sobre o algoritmos de Lanczos foram realizados ainda na década de 80. Vale destacar os trabalhos Marques [8, 42] que estudaram as versões do método de Lanczos por blocos, baseando-se em Cullumm e Donath [43]. Métodos por blocos são bastante adequados quando há muitos autovalores com multiplicidade maior que um. Marques [42] também fez estudos considerando problemas de ortogonalidade e algoritmos para tratar esses problemas. Mais recentemente, o autor [44] realizou um trabalho sobre o método de Arnoldi/Lanczos, focando nas particularidades da solução do problema de autovalor generalizado.

Métodos baseados no subespaço de Krylov têm sido implementados tanto em arquiteturas de memórias distribuídas e compartilhadas. Para arquiteturas de memória distribuída, há pacotes de domínio público como PARPACK [45], SLEPc [16] e Anasazi [15]. A paralelização do método de Arnoldi/Lanczos com reinício implícito pode ser dividida em dois estágios, a fatoração em k passos e a fase do *implicitly shifted QR*. Na fase da fatoração, o produto da matriz pelo vetor é a operação de maior custo computacional. Uma implementação eficiente em paralelo dessa operação deve levar em consideração um esquema de comunicação ponto a ponto envolvendo processos vizinhos. Versões em blocos podem ser usadas, de forma a melhorar a eficiência do produto matriz-vetor, tal como foi explorado por

Guarracino e Perla [46]. Avanços significativos na paralelização do *implicitly shifted QR* para arquiteturas de memória distribuída foram dados por Henry et al. [47], os quais propuseram um esquema em paralelo no qual a matriz de Hessenberg obtida durante o processo é particionada, resultando em blocos que são designados a cada processo. Neste esquema o próprio *implicitly shifted QR* é empregado na solução do problema de autovalor não-simétrico. Como alternativa, Sorensen havia proposto, anos antes, que todos os processos tivessem a mesma cópia da matriz de Hessenberg, desta forma, não há necessidade de comunicação durante o *implicitly shifted QR*, esta alternativa é bastante adequada para o método de Arnoldi/Lanczos com reinício implícito, uma vez que a ordem da matriz de Hessenberg/tridiagonal é muito menor que a ordem do sistema original. A versão paralela do consagrado pacote ARPACK, o PARPACK [45] usa essa estratégia, o usuário deve prover o produto matriz-vetor correspondente à estrutura de dados utilizada na discretização do problema, o que pode exigir a solução de um sistema de equações algébricas. O PARPACK não possui, portanto, nenhuma rotina que realiza a comunicação dos resultados do produto matriz-vetor nem estruturas de dados apropriadas para a computação paralela. Além disso, tanto o ARPACK quanto o PARPACK utilizam um modelo de comunicação reversa com o usuário, que é um modelo de implementação que tem pontos falhos do ponto de vista da boa prática de programação.

Novos esquemas de reinício da fatoração de Arnoldi e Lanczos foram propostos por Emad et al [48], utilizando múltiplos reinícios explícitos para melhorar a convergência. Com o objetivo de minimizar as etapas que envolvem operações com comunicação global no método de Arnoldi, Hernandez et al [49] propuseram modificações no processo de Gram-Schmidt, modificando a ordem e sequência das operações envolvendo produtos e escalares e cálculo de normas.

Capítulo 2

O Método de Arnoldi/Lanczos com Reinício Implícito

Neste capítulo será apresentado o método de Arnoldi/Lanczos com Reinício Implícito. Princípios variacionais serão utilizados para a formulação dos métodos de Arnoldi [12] e Lanczos [13]. Por fim, um esquema para reiniciar a fatoração de Lanczos ou Arnoldi serão apresentados.

2.1 O Método de Arnoldi/Lanczos

2.1.1 Um Método Clássico para a Redução da Ordem do Problema

Sem perda de generalidade, será adotado $B = I$ no problema de autovalor generalizado (eq. 1.1):

$$Ax = \lambda x \tag{2.1}$$

ou, equivalentemente, seja o problema de autovalor padrão:

$$(\lambda I - A)x = \mathbf{0} \tag{2.2}$$

A solução da eq. (2.2), de ordem n, será substituída pela solução de um problema de ordem menor k. Para tanto, admite-se que os autovetores x podem ser aproximados por uma base de vetores v_i que gera o espaço k-dimensional $\mathcal{K}$:

$$\hat{x} = \sum_{i=1}^{k} c_i v_i = [v_i]c \tag{2.3}$$

Na expressão anterior $[v_i]$ é uma matriz retangular cujas colunas são os vetores v_i e c é o vetor formado pelas constantes c_i. Substituindo a expressão (2.3) na eq. (2.2), obtém-se a expressão:

$$(\hat{\lambda}I - A)[v_i]c = \mathbf{0} \tag{2.4}$$

onde, $\hat{\lambda}$ são os autovalores aproximados.

Considerando-se agora o problema padrão de autovalor à esquerda, têm-se:

$$\hat{y}^T(\hat{\lambda}I - A) = \mathbf{0} \tag{2.5}$$

onde y^T é o transposto dos autovetores à esquerda y.

De forma semelhante, os autovetores aproximados $\hat{y}$ são dados por:

$$\hat{y} = \sum_{j=1}^{k} d_j u_j = [u_j]d \tag{2.6}$$

onde u_j são os vetores da base do espaço $\mathcal{L}$ de aproximação dos autovetores à esquerda e d_j são coeficientes constantes.

A eq. (2.4) será ponderada por $[u_j]$ e, para tanto, basta pré-multiplicar a mesma equação por $[u_j]$, para obter:

$$[u_j]^T(\hat{\lambda}I - A)[v_i]c = \mathbf{0} \tag{2.7}$$

ou, em uma forma mais concisa,

$$(\hat{\lambda}I - [u_j^T v_i]^{-1}[u_j^T A v_i])c = \mathbf{0}. \tag{2.8}$$

A equação (2.8) é chamada de forma reduzida da eq. (2.2) e a matriz

$$[u_j^T v_i]^{-1}[u_j^T A v_i]$$

é chamada de matriz reduzida.

Como foi usada uma aproximação com k elementos, a ordem da matriz reduzida é, obviamente, k. Os vetores v_i formam uma base para o espaço $\mathcal{K}$ que será usado para aproximar os autovetores à direita e os vetores u_j formam uma base para o espaço $\mathcal{L}$ que, por sua vez, será usado na aproximação dos autovetores à esquerda. Esse tipo de aproximação é conhecido como sendo o método de Petrov-Galerkin, podendo ser escrito na forma:

$$\left(\left(\hat{\lambda}I - A\hat{x}\right), u\right) = 0, \forall u \in \mathcal{L} \tag{2.9}$$

onde $(\cdot, \cdot)$ denota o produto interno tal que para $x, y \in \mathbb{R}^n$:

$$(x, y) = x^T y. \tag{2.10}$$

Para o cálculo da matriz reduzida, é necessária a inversão de $[u_j^T v_i]$. Por motivos práticos, esse processo deve ser evitado, sobretudo devido à instabilidade numérica. Lanczos [13] propôs uma condição que, uma vez adotada, supera essa dificuldade.

2.1.2 O Método de Lanczos

Lanczos adotou uma condição de bi-ortogonalidade[1] entre as bases v_i e u_j, ou seja, $u_j^T v_i = 0$ quando $i \neq j$, tornando a avaliação de $[u_j^T v_i]^{-1}$ trivial, uma vez que essa matriz se reduz a uma matriz diagonal. A eq. (2.8) se transformaria em:

$$(\hat{\lambda}I - \left[\frac{u_j^T A v_i}{u_i^T v_i}\right])c = \mathbf{0} \tag{2.11}$$

A partir de vetores v_1 e u_1 iniciais pode-se obter as expressões recorrentes que fornecerão os demais vetores da base dos espaços de aproximação ($\mathcal{K}$) e ponderação

[1] Bi-ortogonalidade é uma generalização de ortogonalidade em espaços de Hilbert, ver Brezinski [50]

($\mathcal{L}$):

$$v_{i+1} = Av_i - \alpha_i v_i - \beta_{i-1} v_{i-1} - \gamma_{i-2} v_{i-2} - \delta_{i-3} v_{i-3} - ... \tag{2.12a}$$

$$u_{i+1} = A^T u_i - \alpha_i^{'} u_i - \beta_{i-1}^{'} u_{i-1} - \gamma_{i-2}^{'} v_{i-2} - \delta_{i-3}^{'} v_{i-3} - ... \tag{2.12b}$$

As constantes α_i, β_{i-1}, γ_{i-2}, ..., $\alpha_i^{'}$, $\beta_{i-1}^{'}$, $\gamma_{i-2}^{'}$, ..., são facilmente determinadas aplicando a relação de bi-ortogonalidade $u_j^T v_i = 0$, se $i \neq j$, como pode ser visto a seguir:

$$u_i^T v_{i+1} = u_i^T A v_i - \alpha_i u_i^T v_i = 0 \tag{2.13a}$$

$$u_i^T v_{i+2} = u_i^T A v_{i+1} - \beta_i u_i^T v_i = 0 \tag{2.13b}$$

$$u_i^T v_{i+3} = u_i^T A v_{i+2} - \gamma_i u_i^T v_i = 0, \tag{2.13c}$$

etc...

Da mesma forma que,

$$u_{i+1}^T v_i = u_i^T A v_i - \alpha_i^{'} u_i^T v_i = 0 \tag{2.14a}$$

$$u_{i+2}^T v_i = u_{i+1}^T A v_i - \beta_i^{'} u_i^T v_i = 0 \tag{2.14b}$$

$$u_{i+3}^T v_i = u_{i+2}^T A v_i - \gamma_i^{'} u_i^T v_i = 0, \tag{2.14c}$$

etc...

O conjunto de eqs. (2.13a - 2.14c) torna possível a determinação das constantes:

$$\alpha_i = \frac{u_i^T A v_i}{u_i^T v_i} \tag{2.15a}$$

$$\beta_i = \frac{u_i^T A v_{i+1}}{u_i^T v_i} \tag{2.15b}$$

$$\gamma_i = \frac{u_i^T A v_{i+2}}{u_i^T v_i}, \tag{2.15c}$$

$$\alpha_i^{'} = \frac{u_i^T A v_i}{u_i^T v_i} \tag{2.16a}$$

$$\beta_i^{'} = \frac{u_{i+1}^T A v_i}{u_i^T v_i} \tag{2.16b}$$

$$\gamma_i^{'} = \frac{u_{i+2}^T A v_i}{u_i^T v_i}, \tag{2.16c}$$

etc...

Pelas eqs. (2.15a e 2.16a), $\alpha_i = \alpha_i^{'}$. Outras identidades úteis podem ser encontradas através de:

$$A v_i = v_{i+1} + \alpha_i v_i + \beta_{i-1} v_{i-1} + ... \tag{2.17a}$$

$$u_{i+1}^T A v_i = u_{i+1}^T v_{i+1} \tag{2.17b}$$

De forma semelhante,

$$u_i^T A = u_{i+1}^T + \alpha_i^{'} u_i^T + \beta_{i-1}^{'} u_{i-1}^T + ... \tag{2.17c}$$

$$u_i^T A v_{i+1} = u_{i+1}^T v_{i+1}, \tag{2.17d}$$

portanto, $\beta_i = \beta_i^{'}$. Um processo semelhante demonstra que $u_i^T A v_{i+2} = 0$, logo $\gamma_i = 0$, da mesma forma que $\gamma_i^{'} = 0$. As outras constantes, através do mesmo processo também se anulam e com isso as eqs. (2.12a e 2.12b) tornam-se:

$$v_{i+1} = A v_i - \alpha_i v_i - \beta_{i-1} v_{i-1}, \tag{2.18}$$

$$u_{i+1} = A^T u_i - \alpha_i u_i - \beta_{i-1} u_{i-1}, \tag{2.19}$$

onde,

$$\alpha_i = \frac{u_i^T A v_i}{u_i^T v_i} \tag{2.20}$$

$$\beta_{i-1} = \frac{u_{i-1}^T A v_i}{u_{i-1}^T v_{i-1}}. \tag{2.21}$$

Conhecendo-se esses termos, pode-se, agora, determinar a matriz reduzida de ordem k:

$$\left[\frac{u_j^T A v_i}{u_i v_i}\right] = \begin{bmatrix} \alpha_1 & \beta_1 & 0 & 0 & \cdots & 0 \\ 1 & \alpha_2 & \beta_2 & 0 & \cdots & 0 \\ 0 & 1 & \alpha_3 & \beta_3 & \cdots & 0 \\ 0 & 0 & 1 & \alpha_4 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 1 & \alpha_k \end{bmatrix} \tag{2.22}$$

O subespaço de Krylov aparece naturalmente através da abordagem adotada, como pode ser visto nas eqs. (2.18 e 2.19). A forma tridiagonal da matriz mostrada na eq. (2.22) é uma das consequências desse método.

2.1.3 O Método de Arnoldi Derivado do Método de Lanczos

Em seu trabalho, Arnoldi [12] , diferentemente de Lanczos, adotou o subespaço de ponderação igual ao subespaço de aproximação para reduzir a ordem do problema. Dessa forma, a matriz retangular que pré-multiplica a eq. (2.4) é igual à matriz utilizada na aproximação:

$$[v_j]^T(\hat{\lambda} I - A)[v_i]c = \mathbf{0} \tag{2.23}$$

Em outras palavras, trocou-se uma condição de Petrov-Galerkin, por uma condição de Galerkin:

$$\left(\left(\hat{\lambda}I - A\hat{x}\right), v\right) = 0, \forall v \in \mathcal{K} \tag{2.24}$$

Novamente fazendo com que $v_j^T v_i$ para $i \neq j$, a equação reduzida fica:

$$(\hat{\lambda}I - \frac{[v_j^T A v_i]}{[v_i^T v_i]})c = \mathbf{0}. \tag{2.25}$$

Através da seguinte fórmula de recorrência, pode-se determinar os v_{i+1}:

$$v_{i+1} = Av_i - \alpha_i v_i - \beta_{i-1} v_{i-1} - \gamma_{i-2} v_{i-2} - ... \tag{2.26}$$

As constantes que determinam o vetor v_{i+1} são obtidas de maneira semelhante ao método de Lanczos, logo:

$$\alpha_i = \frac{v_i^T A v_i}{v_i^T v_i}, \tag{2.27a}$$

$$\beta_i = \frac{v_i^T A v_{i+1}}{v_i^T v_i}, \tag{2.27b}$$

$$\gamma_i = \frac{v_i^T A v_{i+2}}{v_i^T v_i}, \tag{2.27c}$$

etc...

Nenhuma dessas constantes se anulam, no entanto pode-se observar que:

$$v_{i+1} A v_i = v_{i+1}^T v_{i+1} \tag{2.28}$$

e

$$v_{i+j}^T A v_i = 0 \quad se \quad j > 1 \tag{2.29}$$

Agora, a matriz reduzida de ordem k se apresenta desse jeito:

$$\left[\frac{v_j^T A v_i}{v_i v_i}\right] = \begin{bmatrix} \alpha_1 & \beta_1 & \gamma_i & \delta_i & \cdots \\ 1 & \alpha_2 & \beta_2 & \gamma_2 & \cdots \\ 0 & 1 & \alpha_3 & \beta_3 & \cdots \\ 0 & 0 & 1 & \alpha_4 & \cdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 1 & \alpha_k \end{bmatrix}. \tag{2.30}$$

Essa matriz é uma matriz de **Hessenberg** superior, cujos elementos h_{ij} são iguais a zero para $i > j + 1$.

Observado as eqs. (2.12a, 2.12b e 2.26), nota-se que, quando se utiliza uma aproximação do tipo Galerkin e quando a matriz A é simétrica ($A = A^T$), as eqs. (2.12a e 2.12b) se reduzem a uma só, que, por sua vez, é exatamente igual à eq. (2.26), ou seja, o método de Lanczos se reduz ao método de Arnoldi. Inclusive quando $A = A^T$, a matriz de Hessenberg obtida pela fatoração de Arnoldi se reduz à matriz tridiagonal característica do método de Lanczos. Por isso será adotada a nomenclatura Arnoldi/Lanczos que, daqui para frente, significará que quando o problema for não simétrico o método de Arnoldi será empregado, caso contrário será usado o método de Lanczos.

2.2 Implementação do Algoritmo

2.2.1 Transformação Espectral

A convergência de algoritmos iterativos baseados em potências do tipo $A, A^2, A^3, ..., A^k$ é dominada pelo raio espectral[1] da matriz A, portanto, os autovalores que apresentarão melhor convergência serão os de maior valor absoluto (Golub e Van Loan [52]). A depender do problema em questão, os autovalores de menor valor absoluto são os mais importantes para a análise, como, por exemplo, na análise

[1] Autovalor de maior valor absoluto de um operador (Bachman, Narici [51])

dinâmica de estruturas, portanto, é preciso recorrer a uma transformação espectral, para melhorar a aproximação desses autovalores:

$$(A - \sigma B)^{-1} B x = \theta x \tag{2.31}$$

onde,

$$\theta = \frac{1}{\lambda - \sigma}, \sigma \neq \lambda \tag{2.32}$$

Aplicando esta transformação espectral, melhora-se a convergência dos valores de λ próximos a σ. Para o cálculo dos menores autovalores, basta escolher $\sigma = 0$.

Em geral, $(A - \sigma B)^{-1} B$ não é simétrica em relação ao produto interno usual, $(x, y) = x^T y$, mesmo quando A e B o são. Para problemas simétricos , o produto interno usual será substituído (para B não singular) por:

$$(x, y)_B = x^T B y \tag{2.33}$$

Desta forma, $(A - \sigma B)^{-1} B$ passa a ser simétrica em relação a esse produto interno. Sendo assim, dois vetores, x e y, serão B-ortogonais quando $x^T B y = 0$. A norma induzida por esse produto interno será denotada por $|| \cdot ||_B$.

Quando esse tipo de transformação é utilizada, o produto matriz–vetor será realizado implicitamente nas etapas:

$$z = Bx \tag{2.34a}$$

e depois resolve-se um sistema do tipo :

$$(A - \sigma B) y = z \tag{2.34b}$$

em função de y.

2.2.2 Fatoração de Arnoldi em k passos

O algoritmo do método de Arnoldi, para o problema de autovalor generalizado, pode ser sumarizado tal como se segue:

Algoritmo 2.1 Algoritmo da fatoração de Arnoldi em k passos se $k_0 = 0$, se não estende a fatoração em mais j passos (problema generalizado)

1: Entrada : A, B, $V_{k_0} = [v_1, v_2, \cdots, v_{k_0}]$, H_{k_0}, f_{k_0}.

2: $k = k_0 + j$;

3: Se $(k_0 = 0)$ então

4: Gera o vetor de partida v_1, tal que $||v_1|| = 1$;

5: $w_1 = B^{-1}Av_1$; $T_1 = v_1^T w_1$;

6: $f_1 = w_1 - H_1 v_1$;

7: $k_0 = k_0 + 1$;

8: Fim Se

9: Para $i = k_0, \cdots, k-1$ faça:

10: $\beta_i = ||f_i||$;

11: $v_{i+1} = \dfrac{f_i}{\beta_i}$; $V_{i+1} = [V_i, v_{i+1}]$;

12: $w_{i+1} = B^{-1}Av_{i+1}$;

13: $\hat{H}_i = \begin{bmatrix} H_i \\ \beta_i e_i^T \end{bmatrix}$;

14: $h = V_{i+1}^T w_{i+1}$;

15: $H_i = \begin{bmatrix} \hat{H}_i & h \end{bmatrix}$;

16: $f_{i+1} = w_{i+1} - V_{i+1}h$;

17: Processo de Reortogonalização.

18: Fim Para

19: $k_0 = k$

O algoritmo (2.1) utiliza como processo de ortogonalização o método de Gram–Schmidt com reortogonalização [53]. Esse método foi escolhido devido a sua simplicidade de implementação. Na tabela (2.1) é mostrada uma comparação entre o método escolhido e os métodos: *MGS* – Gram–Schmidt modificado, *GSR* – Gram-Schmidt com reortogonalização e *HO* – Householder. dt modificado ou o método de Householder [52]. Como pode ser visto nesta tabela, os métodos *GS* e *GSM* realizam a mesma quantidade de operações e demandam a mesma quantidade de armazenamento. Com relação ao método de Householder, sua implementação é um pouco mais trabalhosa por usar vetores de diferentes ordens. Essa característica lhe confere a ligeira vantagem no armazenamento e no número de operações, porém, para valores de k usuais ($k \ll n$), essa vantagem torna-se irrelevante.

Tabela 2.1: Comparação entre as diversas implementações do Método de Arnoldi (fonte [2])

	GS	GSM	GSMR	HO
Operações com ponto flutuante	k^2n	k^2n	$2k^2n$	$2k^2n - \frac{2}{3}k^3$
Armazenamento	$(k+1)\,n$	$(k+1)\,n$	$(k+1)\,n$	$(k+1)\,n - \frac{1}{2}k^2$

Conforme mostrado anteriormente, o método de Lanczos, em sua versão não simétrica, gera dois subespaços, implicando na geração de duas bases bi-ortogonais. Por outro lado, o método de Arnoldi só precisa de uma única base ortogonal, o que reduz em quase metade o número de operações e, além disso, reduz os problemas com perda de ortogonalidade. Diante dessas vantagens, fica justificada a escolha do método de Arnoldi/Lanczos. O algoritmo da versão simétrica do método de Lanczos é mostrado a seguir.

Algoritmo 2.2 : Fatoração de Lanczos em k passos ($\sigma = 0$)

1: Input : A, B, $V_{k_0} = [v_1, v_2, \cdots, v_{k_0}]$, T_{k_0}, f_{k_0}.

2: $k = k_0 + j$;

3: Se ($k_0 = 0$) então

4: Gerar vetor de partida: v_1, com $||v_1|| = v_1^T B v_1 = 1$;

5: $w_1 = A^{-1}Bv_1$; $T_1 = v_1^T B w_1$;

6: $f_1 = w_1 - T_1 v_1$;

7: $k_0 = k_0 + 1$;

8: Fim Se

9: Para $i = k_0, \cdots, k-1$ faça:

10: $\beta_i = ||f_i||_B$;

11: $v_{i+1} = \dfrac{f_i}{\beta_i}$; $V_{i+1} = [V_i, v_{i+1}]$;

12: $w_{i+1} = A^{-1}Bv_{i+1}$;

13: $\hat{T}_i = \begin{bmatrix} T_i \\ \beta_i e_i^T \end{bmatrix}$;

14: $t = V_{i+1}^T B w_{i+1}$;

15: $T_i = \begin{bmatrix} \hat{T}_i & t \end{bmatrix}$;

16: $f_{i+1} = w_{i+1} - V_{i+1}h$;

17: Processo de reortogonalização.

18: Fim Para

19: $k_0 = k$

2.2.3 Processo de Reortogonalização

Um dos grandes problemas de algoritmos baseados no subespaço de Krylov, é a perda de ortogonalidade dos vetores que geram esse subespaço. Há dois motivos que causam esse problema:

- Perda de precisão devido à aritmética de ponto flutuante com precisão finita;

- A base calculada gera um subespaço invariante.

Devido à aritmética finita, a perda de ortogonalidade ocorre nos passos 17 e 18 do algoritmo (2.1), portanto há a necessidade de reortogonalizar os vetores que formam a base do subespaço de Krylov. Vários esquemas têm sido propostos ([54] e [55]), o esquema implementado foi o método proposto em [39], mostrado a seguir:

Algoritmo 2.3 Algoritmo do Processo de Reortogonalização

1: Se $(||f_{i+1}|| \leq 0.717 \cdot ||h||)$ então
2: Reortogonalização:
3: $rnorm = ||f_{i+1}||$;
4: $s_{i+1} = V_{i+1}^T f_{i+1}$; $f_{i+1} = f_{i+1} - V_{i+1} s$;
5: $rnorm1 = ||f_{i+1}||$;
6: $h = h + s_{i+1}$;
7: Refinamento Iterativo:
8: Se $(rnorm1 > 0.717 \cdot rnorm)$ então
9: $||f_{i+1}|| = rnorm1$
10: Senão
11: $||f_{i+1}|| = rnorm1$
12: Se (Primeira vez no refinamento iterativo) então
13: Reortogonalize novamente
14: Fim Se
15: Fim Se
16: Fim Se

Durante o processo de reortogonalização, há a necessidade de se calcular o vetor resíduo, $f_i = Av_i - \alpha_i v_i$. Esse vetor será nulo se v_i for um autovetor e portanto α_i é exatamente um autovalor do problema. Isso significa que foi computado um subespaço invariante do problema. Como o vetor $f = \mathbf{0}$, o processo não consegue avançar mais um passo. Quando isso acontece, um novo vetor de partida deve ser gerado e a fatoração reiniciada.

2.2.4 Método de Arnoldi com Reinício Implícito

Uma das desvantagens do método de Arnoldi é a impossibilidade de se prever *a priori* o número de passos necessários para que os valores e vetores de Ritz sejam boas aproximações dos autovalores e autovetores desejados. Em certos problemas, com autovalores muito próximos ou repetidos, para a convergência da faixa do espectro desejada, é preciso aumentar o número de passos realizados pela fatoração. O aumento do número de passos acarreta em problemas de ortogonalidade e mais demanda por memória, pois é preciso armazenar todos os vetores da base de Krylov, sendo que cada vetor gerado tem a dimensão do número de equações do problema original. Outro ponto negativo, é que para maiores valores de k (número de passos) o custo computacional do cálculo dos autovalores da matriz de Hessenberg ou tridiagonal na ordem de $\mathcal{O}(k^3)$, torna-se proibitivo.

Com essa dificuldade em mente, Saad [56] propôs reiniciar a fatoração com um vetor de partida pré-condicionado. O pré-condicionador utilizado é um polinômio construído de tal modo a "amortecer" as componentes relativas aos autovalores fora da porção do espectro desejado na série finita de aproximação dos autovetores. Após o pré-condicionamento, a fatoração de Arnoldi/Lanczos é reiniciada por completo. Por isso, esse esquema recebeu o nome de reinício explícito.

Sorensen [36] baseou-se na conexão que há entre o *double shift QR* ([37] e [38]) e o método de Arnoldi/Lanczos para criar o método de Arnoldi/Lanczos com reinício implícito. Nesse método, a fatoração é reiniciada sem precisar descartar todos os passos já computados, ou seja, recomeçando a partir de um determinado ponto, mantendo o número de passos finais (k) constante. O método de reinício implícito idealizado por Sorensen utiliza um polinômio calculado baseado no espectro da matriz de Hessenberg ou tridiagonal de ordem k. O espectro é calculado e separado em dois conjuntos disjuntos Ω_d e Ω_i, onde as aproximações dos autovalores desejados estão no primeiro conjunto. As aproximações para o segundo conjunto serão usadas como os *shifts* no *double shift QR*. De maneira análoga ao método de Arnoldi com reinício explícito, esse processo também se equivale a reiniciar a fato-

ração com um vetor de partida pré-condicionado, porém a aplicação do polinômio pré-condicionador (q) é feita implicitamente. Esse polinômio tem como raízes os elementos do conjunto Ω_i :

$$q(\lambda) = \prod_{i=1}^{p} (\lambda - \mu_i) \tag{2.35}$$

onde, μ_i são os *shifts* associados aos autovalores da porção do espectro cuja convergência não é importante para o problema.

Após a decomposição QR, as matrizes H_k e V_k são atualizadas e uma nova fatoração é reiniciada a partir do passo k_0, onde $k_0 = k - p$. O processo é repetido até que a convergência seja satisfeita. À medida que os autovalores convergem, o número de *shifts* é decrescido, diminuindo o número de passos restantes que serão recalculados. O algoritmo do método de Arnoldi/Lanczos com reinício implícito pode ser visto a seguir:

Algoritmo 2.4 Algoritmo do Método de Arnoldi/Lanczos com Reinício Implícito

1: Entrada : A, B, V_k, H_k e f_k obtidos após o Algoritmo 2.1.

2: Para $i = 1, 2, \cdots$, até convergência faça:

3: Calcular o espectro de H_k e selecionar p "shifts"$\mu_1, \mu_2, \cdots, \mu_p$

4: $q^T = e_k^T$;

5: Para $j = 1, 2, \cdots, p$ faça:

6: Fatorar $[Q, R] = qr\,(H_k - \mu_j I)$;

7: $H_k = Q^T H_k Q$; $V_k = V_k Q$;

8: $q = q^T Q$;

9: Fim Para

10: $k_0 = k - p$;

11: $\gamma_{k_0} = H_k(k_0 + 1, k_0)$

12: $\zeta_{k_0} = Q(k, k_0)$

13: $V_{k_0} = V_k\,(1 : n, 1 : k_0)$;

14: $f_{k_0} = \gamma_{k_0} V_k e_{k_0+1} + \zeta_{k_0} f_k$;

15: Recomeçar uma nova fatoração utilizando o algoritmo 2.1 partindo do passo k_0 e aplicar mais p passos até atingir novamente o passo k.

16: Teste de convergência.

17: Fim Para

O algoritmo (2.4) é o mesmo para o método de Lanczos, apenas substituindo a matriz de Hessenberg pela matriz tridiagonal. O processo de Schur [52] é utilizado para o cálculo dos *shifts*. Após a convergência do processo de reinício implícito, o mesmo processo de Schur é usado no problema de autovalor reduzido:

$$H_k c = \hat{\lambda} c \tag{2.36}$$

Por fim, o autovetor do problema original é dado por:

$$x = V c \tag{2.37}$$

Capítulo 3

Implementação em Paralelo

3.1 Introdução

Descreve-se neste capítulo uma implementação em paralelo do método dos elementos finitos para a solução de problemas de autovalor generalizado utilizando o método de Arnoldi/Lanczos com reinício implícito. Esta implementação é voltada para arquiteturas de memória distribuída, visando a utilização de clusters de pc's. A formulação paralela desenvolvida neste trabalho adota o paralelismo do tipo SPMD (single program multiple data - programa único, múltiplos dados), que requer uma cópia idêntica do programa em cada processo. Cada cópia, por sua vez, opera em conjuntos diferentes de dados. Este esquema é típico de arquiteturas de memória distribuída, onde cada processo realiza tarefas locais e a comunicação é realizada através de um protocolo de troca de mensagens. Na solução em paralelo adotou-se a estratégia de "subdomínio por subdomínio" (SBS - subdomain by subdomain) [23]. A estratégia SBS é, no caso de métodos iterativos, uma alternativa aos métodos de decomposição de domínio. Nos métodos de decomposição de domínio cada processo executa tarefas locais relacionadas à sua partição e esquemas baseados na matrix de complemento de Schur ou no método de Schwarz proveem a solução nas interfaces. Diferentemente dos métodos de decomposição de domínio, no esquema SBS os códigos sequencial e paralelo são exatamente os mesmos. A convergência, portanto, não é alterada. Para o particionamento das malhas de elementos finitos, utilizou-se

dois esquemas: com e sem sobreposição de domínios.

Uma implementação em paralelo eficiente do método dos elementos finitos requer comunicação somente na etapa de solução do sistemas de equações. Assim sendo, a comunicação presente na formulação desenvolvida requer comunicação somente nos algoritmos dos métodos de Arnoldi e Lanczos. De fato, a presente formulação requer comunicação somente nas operações de produto escalar e produto matriz-vetor. No caso de um código de elementos finitos que utiliza métodos iterativos na solução de equações algébricas, como o método do gradiente conjugado, consegue-se ter redundância praticamente nula em termos de operações. Já no caso do GMRES (generalized minimal residual- método do resíduo mínimo generalizado), em favor da eficiência, é mais adequado permitir uma certa redundância referente à replicação da matriz de Hessenberg, que é do tamanho da base do subespaço de Krylov utilizada, cuja base deve ser muito menor que o número de equações do sistema original. No caso do método de Arnoldi/Lanczos, a redundância nas matrizes de Hessenberg e tridiagonal, respectivamente, eliminam a comunicação na fase de reinício implícito. Uma alternativa a essa abordagem é o particionamento da matrizes de Hessenberg ou tridiagonal, eliminando a redundância, tal como foi proposto por [47]. Esse esquema comprovou-se ser menos eficiente e tem pior escabilidade, pois afeta a convergência.

As estruturas de dados utilizadas são comprimidas no formato CSRC e CSRC/CSR [23]. Estas estruturas de dados permitem uma otimização do uso da memória, redução das operações de ponto flutuante e elimina a comunicação na fase de montagem dos coeficientes globais. São, portanto, estruturas de dados projetadas para computação de alto desempenho, viabilizando a solução de problemas com elevado número de equações. Nas etapas que requerem comunicação, o esquema empregado [24] é bastante eficiente, pois trata-se de um esquema ótimo no sentido que não há comunicação desnecessária entre processos adjacentes.

3.2 Particionamento sem sobreposição ("non-overlapping")

No particionamento sem sobreposição, o domínio original Ω é sub-dividido em n sub-domínios Ω^i, $i = 0, \ldots, n-1$, onde n indica o número de processos, de tal forma que:

$$\Omega = \bigcup_{i=0}^{n-1} \Omega^i; \tag{3.1a}$$

$$\Omega^i \cap \Omega^j = \emptyset \quad \text{para } i \neq j \tag{3.1b}$$

O contorno original de Ω será denotado por $\partial\Omega$, enquanto que as fronteiras internas de Ω serão chamadas por $\partial\Omega_{int}$. O fecho de Ω é $\bar{\Omega} = \Omega \cup \partial\Omega$, enquanto que o fecho de uma partição Ω^i é $\bar{\Omega}^i = \Omega^i \cup \partial\Omega^i$, onde $\partial\Omega^i$ é o contorno de Ω^i.

Consideremos agora a malha $M_{\bar{\Omega}}$ como uma discretização de $\bar{\Omega}$. Supondo ser possível obter uma malha da partição sem sobreposição $M_{\bar{\Omega}^i} \subset M_{\bar{\Omega}}$, cada sub-domínio Ω^i conterá nel^i elementos (E^i) e um conjunto N^i formado pelos $nnode^i$ nós da partição i.

$$N^i = \{n_1^i, n_2^i, \ldots, n_{nnode^i}^i\} \tag{3.2a}$$

$$E^i = \{e_1^i, e_2^i, \ldots, e_{nel^i}^i\} \tag{3.2b}$$

Cada nó da fronteira interna de um sub-domínio, pode pertencer a apenas uma partição, o que não exclui o fato desse mesmo nó aparecer em duas ou mais partições. Conseqüentemente, o conjunto N^i dessa partição será formado por três sub-conjuntos:

- o subconjunto $N^{i,1}$ que contem os nós que se situam no interior do sub-domínio ou no contorno externo da partição ($\partial\Omega$);

- o subconjunto $N^{i,2}$ formado por nós que pertencem ao sub-domínio e estão na fronteira de duas ou mais partições;

- e por fim, o subconjunto $N^{i,3}$ que tem como elementos os nós que estão na fronteira interna da partição i, mas que pertencem a uma outra partição.

$$N^i = \{N^{i,1}, N^{i,2}, N^{i,3}\} \tag{3.3}$$

Por sua vez, o conjunto de todo os nós da malha original é dado por:

$$N = \bigcup_{i=0}^{n-1} \{N^{i,1}, N^{i,2}\} \tag{3.4}$$

e o conjunto de nós que compõem o contorno interno das partições ($N^{\partial\Omega^{int}}$) é formado por:

$$N^{\partial\Omega^{int}} = \bigcup_{i=0}^{n-1} N^{i,2} \tag{3.5}$$

A figura (3.1) mostra esquematicamente um particionamento sem sobreposição de um domínio em 4 subdomínios.

Para melhor ilustrar o esquema de particionamento sem sobreposição, uma matriz de coeficientes globais obtida para cada partição Ω^i é mostrada a seguir:

$$A^i = \begin{bmatrix} A^i_{11} & A^i_{12} & A^i_{13} \\ A^i_{21} & A^i_{22} & A^i_{23} \\ A^i_{31} & A^i_{32} & A^i_{33} \end{bmatrix} \tag{3.6}$$

onde os sub-índices 1,2 e 3, são referentes a N^i_1, N^i_2 e N^i_3 respectivamente. Os coeficientes $A^i_{11}, A^i_{12}, A^i_{13}, A^i_{21}$ e A^i_{31} são globais, isto é, estão computadas todas as suas contribuições e os demais são, nesse sentido, coeficientes parciais. Mantendo os coeficientes de A_{ij}, para $i, j = 2, 3$, parciais, elimina-se a comunicação na fase de montagem das matrizes globais.

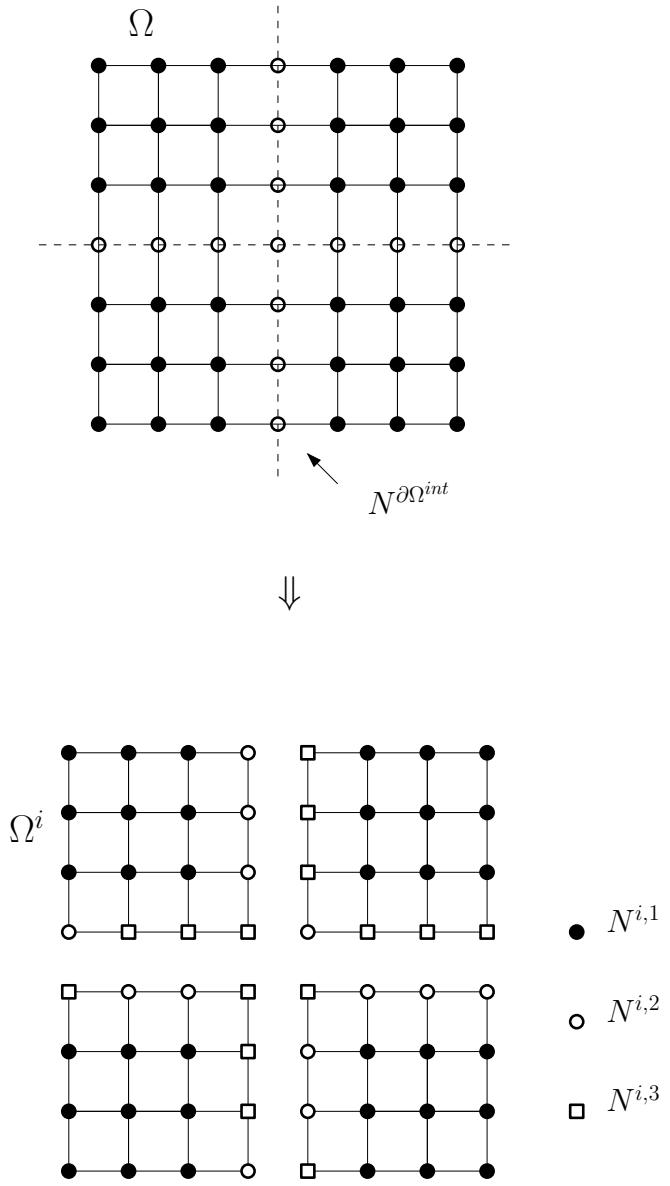

Figura 3.1: Esquema de particionamento sem sobreposição

3.3 Particionamento com sobreposição ("overlapping")

No particionamento com sobreposição, será adicionada à malha sem sobreposição da partição i ($M_{\tilde{\Omega}}$) um conjunto de elementos $E^{i,*} = \{e_k^j\}$, para $j \neq i$. Cada elemento desse conjunto possui ao menos um nó que pertence a $N^{i,2}$ e define-se um novo conjunto de nós $N^{i,4}$ formado pelos nós que estão conectados a algum nó

pertencente a $N^{i,2}$.

$$\widetilde{E}^i = \{E^i \cup E^{i,*}\} \tag{3.7}$$

$$\widetilde{N}^i = \{N^{i,1}, N^{i,2}, N^{i,3}, N^{i,4}\} \tag{3.8}$$

Há ainda, a necessidade de se definir mais um subconjunto de nós, $N^{i,1a}$, que são nós $N^{i,1}$ no interior de Ω^i que correspondem a $N^{j,4}$ numa partição vizinha Ω^j. Um exemplo deste tipo de particionamento pode ser visto na fig. (3.2).

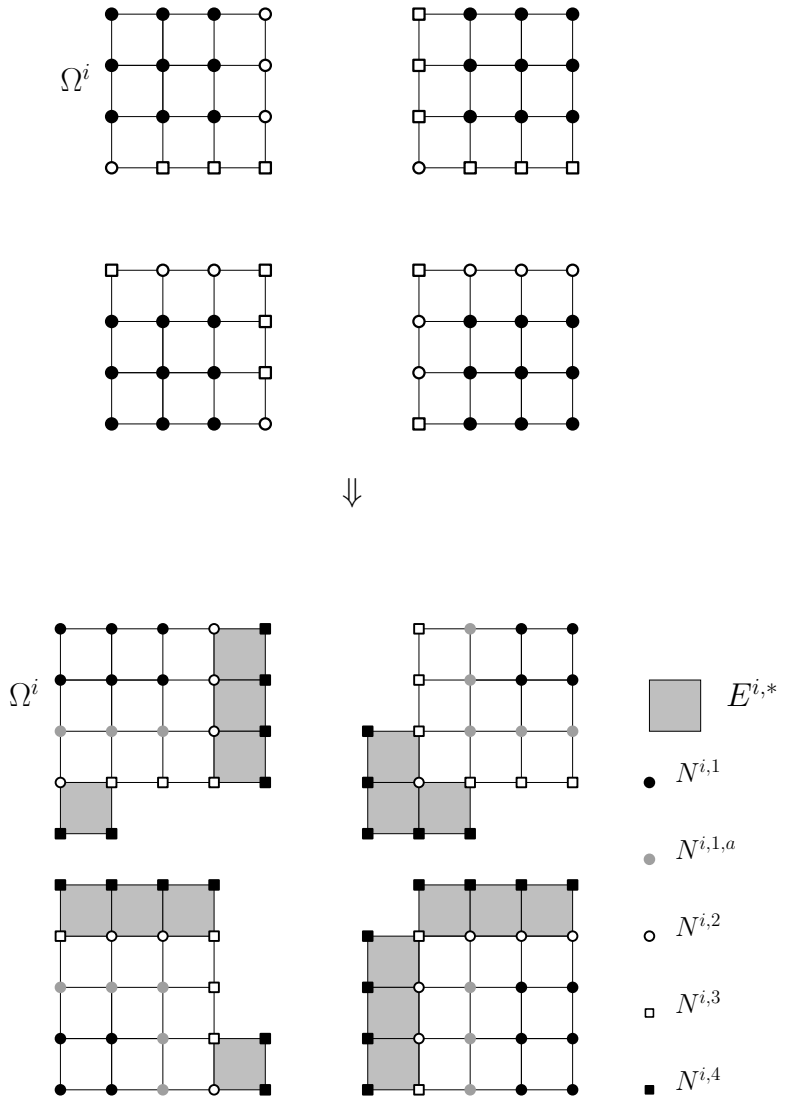

Figura 3.2: Esquema de particionamento com sobreposição

As matrizes de coeficientes obtidas pelo particionamento com sobreposição é

retangular e todos os coeficientes são globais:

$$A^i = \begin{bmatrix} A^i_{11} & A^i_{12} & A^i_{13} & 0 \\ A^i_{21} & A^i_{22} & A^i_{23} & A^i_{24} \end{bmatrix} \tag{3.9}$$

No método com sobreposição, todos os processos dispõem de toda a informação para a montagem das matrizes globais, não há a necessidade de comunicação nessa etapa.

3.4 Particionamento 1d para grafos de matrizes

Quando o problema de autovalor envolver matrizes de origem diversas, sem uma malha associada, um esquema de particionamento por linhas 1d [26] será utilizado. O particionamento 1d de uma matriz que pode ser visto na eq.:

$$A = \begin{bmatrix} A_0 \\ A_1 \\ A_2 \\ \vdots \\ A_{n-1} \end{bmatrix} \tag{3.10}$$

onde, cada bloco A_i tem dimensões $neq^i \times neq$, sendo neq^i o número de linhas pertencentes ao processo i e neq é o número total de equações $neq = \sum_{i=0}^{n-1} neq^i$. Esse particionamento é equivalente ao particionamento de malhas com sobreposição e por isso a mesma estrutura de dados pode ser utilizada.

3.5 Estruturas de dados comprimidas

No método dos elementos finitos, as funções de interpolação e ponderação têm suporte compacto e por isso, as matrizes de coeficientes são esparsas. Dependendo da complexidade das hipóteses físicas e geométricas adotadas na modelagem, o número de equações resultantes torna o emprego de métodos diretos proibitivo. Como alternativa, há os métodos iterativos. Nestes métodos, a operação de maior custo

computacional é o produto matriz-vetor. Por isso, a escolha da estruturas de dados é primordial no desempenho dessa operação. Essa escolha deve levar em consideração: o número de operações de ponto flutuante, endereçamento indireto e a quantidade de memória necessária para armazenar esses coeficientes.

Estruturas de dados comprimidas omo o CSR [57, 58] e o EDS (Edge-based data structure - estrutura de dados baseada em arestas) [59, 60] são baseadas em grafos. No método dos elementos finitos, o grafo está associado à malha e sempre será não-direcionado, ou seja, sua matriz de incidências é simétrica, o que implica que sempre que existir uma aresta ligando um vértice i ao j, haverá uma aresta ligando o vértice j ao i. Ribeiro e Coutinho [61] realizaram um estudo comparativo entre essas estruturas de dados e técnica de armazenamento de elemento por elemento, neste estudo conclui-se que CSR e o EDS são equivalentes, diferenciando-se apenas no modo como os coeficientes são armazenados. No EDS, os coeficientes são acessados percorrendo as arestas do grafo, no CSR, esse acesso é realizado percorrendo os vértices do grafo. O EDS traz algumas desvantagens por depender de uma ordenação que permita, ao percorrer as arestas, acessar as equações de forma o mais suave possível. O CSR, por sua vez, apenas necessita de uma otimização de largura de banda para tornar o acesso ao vetor independente mais suave. Recentemente, Ribeiro e Ferreira [23] propuseram uma variação do CSR, chamada de CSRC. Nesta variante, a parte inferior da matriz é armazenada em formato CSR e a parte superior em CSC, totalizando três arranjos reais que armazenam os coeficientes não-nulos e dois arranjos inteiros. Os termos da diagonal principal são armazenados no arranjo ad, o arranjo au armazena a parte superior da matriz e o arranjo al armazena os coeficientes da parte inferior. Para matrizes simétricas, só é preciso armazenar a parte inferior ou a parte superior da matriz. O arranjo inteiro ja contem o índice da coluna para os coeficientes a_{ij} para $j < i$ e o arranjo inteiro ia que aponta para o primeiro coeficiente de cada linha. Por levar em consideração que o grafo é não-direcionado, o CSRC precisa da metade da quantidade de memória usada pelo CSR para armazenar o arranjo ja , outra vantagem reside no fato de usar dois arranjos

reais para armazenar os termos de fora da diagonal, o que implica em acesso a memória mais rápido. Os algoritmos que realizam o produto matriz-vetor para os casos não simétricos são mostrados a seguir:

Algoritmo 3.1 Produto matriz-vetor ($p = Au$) utilizando o CSRC (não-simétrico)

1: Laço nas equações:
2: Para $i = 1, ..., n$ faça:
3: $ui = u(i)$;
4: $t = ad(i) \cdot ui$
5: Laço sobre os coeficientes não-nulos da arte inferior das equações
6: Para $k = ia(i), ..., ia(i+1) - 1$ faça:
7: $jak = ja(k)$;
8: $t = t + al(k) \cdot u(jak)$;
9: Coluna Superior :
10: $p(jak) = p(jak) + au(k) \cdot ui$;
11: Fim Para
12: $p(i) = t$;
13: Fim Para

Algoritmo 3.2 Produto matriz-vetor ($p = Au$) utilizando o CSRC (versão simétrica)

1: Laço nas equações:
2: Para $i = 1, ..., n$ faça:
3: $ui = u(i)$;
4: $t = ad(i) \cdot ui$
5: Laço sobre os coeficientes não-nulos da arte inferior das equações
6: Para $k = ia(i), ..., ia(i+1) - 1$ faça:
7: $jak = ja(k)$;
8: $s = al(k)$;
9: $t = t + s \cdot u(jak)$;
10: Coluna Superior :
11: $p(jak) = p(jak) + s \cdot ui$;
12: Fim Para
13: $p(i) = t$;
14: Fim Para

Como foi visto, o particionamento com sobreposição origina matrizes retangulares. Para estes casos serão empregados as versões simétricas e não-simétricas do CSRC/CSR, dividindo a matriz em duas submatrizes: uma com o grafo orientado que será armazenada no esquema CSRC e uma outra com o grafo não orientado que será armazenada no formato CSR tradicional [57], como pode ser visto na Fig. (3.3). Para a parte armazenada com CSR, será necessário apenas um arranjo real a para armazenar os coeficientes e dois arranjos inteiros, *ia1* que aponta para o primeiro coeficiente de cada linha e outro arranjo *ja1* que contém os índices da coluna de cada elemento em a.

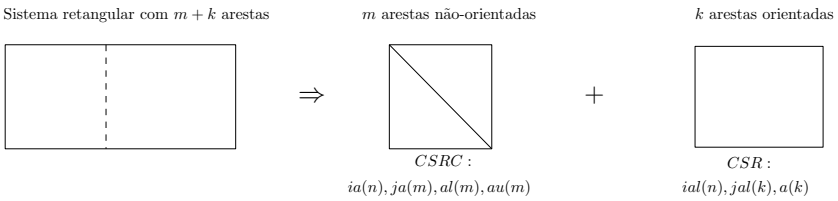

Figura 3.3: Estrutura de dados para matrizes retangulares

Algoritmo 3.3 Produto matriz-vetor retangular ($p = Au$) utilizando o CSRC/CSR (não-simétrico)

1: Laço nas equações:
2: Para $i = 1, ..., n$ faça:
3: $ui = u(i)$;
4: $t = ad(i) \cdot ui$
5: Laço sobre os coeficientes não-nulos da arte inferior das equações
6: Para $k = ia(i), ..., ia(i+1) - 1$ faça:
7: $jak = ja(k)$;
8: $t = t + al \cdot u(jak)$;
9: Coluna Superior :
10: $p(jak) = p(jak) + au(k) \cdot ui$;
11: Fim Para
12: Para $k = ia1(i), ..., ia1(i+1) - 1$ faça:
13: $jak = ja1(k)$;
14: $t = t + a(k) \cdot u(jak)$;
15: Fim Para
16: $p(i) = t$;
17: Fim Para

Algoritmo 3.4 Produto matriz-vetor retangular ($p = Au$) utilizando o CSRC (simétrico)

1: Laço nas equações:
2: Para $i = 1, ..., n$ faça:
3: $ui = u(i)$;
4: $t = ad(i) \cdot ui$
5: Laço sobre os coeficientes não-nulos da parte inferior das equações
6: Para $k = ia(i), ..., ia(i+1) - 1$ faça:
7: $jak = ja(k)$;
8: $s = al(k)$;
9: $t = t + s \cdot u(jak)$;
10: Coluna Superior :
11: $p(jak) = p(jak) + s \cdot ui$;
12: Fim Para
13: Para $k = ia1(i), ..., ia1(i+1) - 1$ faça:
14: $jak = ja1(k)$;
15: $t = t + a(k) \cdot u(jak)$;
16: Fim Para
17: $p(i) = t$;
18: Fim Para

3.6 Operações de níveis 1,2 e 3 em paralelo

Os Métodos de Arnoldi/Lanczos utlizam o processo de ortogonalização de Gram-Schmidt [53] para gerar a base do subespaço de Krylov ($\left\{A^0 v_1, A^1 v_1, \ldots, A^{k-1} v_1\right\}$). Esse processo constitui o principal núcleo computacional desses métodos e sua principais operações são: produto escalar e combinação linear entre vetores. No caso específico do subespaço de Krylov, há ainda o produto matriz-vetor que é realizado em cada passo do processo para que se possa obter o novo elemento da base. O processo de ortogonalização de Gram-Schmidt pode ser visto no algoritmo:

Algoritmo 3.5 Processo de ortogonalização de Gram-Schmidt

1: Entrada: conjunto linearmente independente de vetores $\{x_1, x_2, \ldots, x_n\}$

2: Cálculo $\alpha_{11} = ||x_1||$; $v_1 = \frac{x_1}{\alpha_{11}}$

3: Para $j = 1, ..., n$ faça:

4: Para $i = 1, 2, \ldots, j-1$ faça:

5: $\alpha_{ij} = (x_j, v_i)$

6: $\tilde{v} = x_j - \alpha_{ij} v_i$

7: Fim Para

8: $\alpha_{jj} = ||\tilde{v}||$

9: $v_j = \frac{\tilde{v}}{\alpha_{jj}}$

10: Fim Para

Se o problema de autovalor é generalizado, nos métodos de Arnoldi/Lanczos é preciso a solução de um sistema de equações em cada passo do processo de Gram-Schmidt. O mesmo se aplica se uma transformação espectral é utilizada no problema padrão. Para a solução do sistema de equações empregou-se os métodos do gradiente conjugado pré-condicionado (PCG - preconditioned conjugate gradient) para matrizes simétricas ou o GMRES (generalized minimal residual - resíduo mínimo generalizado) para matrizes não-simétricas. O PCG (Algoritmo 3.6) é a versão do algoritmo de Lanczos simétrico para a solução de sistemas de equações lineares [62], enquanto que o GMRES (Algoritmo 3.7) é derivado do método de Arnoldi [63]. Logo, esses métodos têm em comum as mesmas operações da álgebra linear computacional. Essas operações podem ser classificadas, segundo a quantidade de operações realizadas, em três níveis [52]. Um produto escalar e combinação linear entre vetores são classificadas como sendo de nível um. Produtos envolvendo uma matriz e um vetor são operações do nível dois. Na etapa do reinício implícito da fatoração (Algoritmo 2.4) há produtos envolvendo duas matrizes, essas operações com complexidade de $\mathcal{O}(n^3)$ são classificadas como nível 3.

Algoritmo 3.6 Algoritmo do método dos gradientes conjugados com pré-condicionador diagonal

1: Entrada : A, D, b e x_0

2: calcule $r_0 = b - Ax_0$, $z_0 = D^{-1}r_0$ e $p_0 = z_0$

3: Para $i = 0, 1, \cdots$, até convergência faça:

4: $\quad \alpha_j = \frac{(r_j, z_j)}{(Ap_j, p_j)}$

5: $\quad x_{j+1} = x_j + \alpha_j p_j$

6: $\quad r_{j+1} = r_j - \alpha_j A p_j$

7: $\quad z_{j+1} = D^{-1} r_{j+1}$

8: $\quad \beta_j = \frac{(r_{j+1}, z_{j+1})}{(r_j, z_j)}$

9: $\quad p_{j+1} = z_{j+1} - \beta_j p_j$

10: $\quad$ Teste de convergência.

11: Fim Para

Para melhorar o grau de condicionamento das matrizes, foi utilizado um pré-condicionador diagonal no PCG e no GMRES. A escolha desse pré-condicionador pode ser justificada pela sua facilidade de implementação em esquemas de particionamentos com ou sem sobreposição e pelo seu baixo custo computacional. Vale destacar que, ao contrário de outros pré-condicionadores computacionalmente mais caros, como a fatoração incompleta LU [57], o pré-condicionador diagonal não afeta a convergência do método iterativo executado em paralelo.

Algoritmo 3.7 Algoritmo do GMRES

1: Entrada : A, D, b e x_0

2: calcule $r_0 = D^{-1}(b - Ax_0)$, $\beta = ||r_0||$ e $v_0 = \frac{r_0}{\beta}$

3: Para $j = 0, 1, \cdots, m$ faça:

4: $\quad w = D^{-1}Av_j$

5: $\quad$ Para $i = 0, 1, \cdots, j$ faça:

6: $\quad\quad h_{i,j} = (w, v_i)$

7: $\quad\quad w = w - h_{i,j}v_i$

8: $\quad$ Fim Para

9: $\quad h_{j+1,j} = ||w||$

10: $\quad v_{j+1} = \frac{w}{h_{j+1,j}}$

11: Fim Para

12: $V_m = [v_1, \cdots, v_m]$

13: $H_m = [h_{i,j}]$, $i \leq i \leq j + 1$, $1 \leq j \leq m$

14: $y_m = H_m^{-1}\beta e_1$

15: $x_m = x_0 + V_m y_m$

16: Se convergência foi atingida então

17: $\quad$ pare

18: Senão

19: $\quad$ vá para o passo 1

20: Fim Se

A paralelização das operações de cada nível será descrita a seguir.

3.6.1 Paralelização das operações de nível 1

O produto escalar, em ambos esquemas de particionamento, deve ser realizado apenas com os coeficientes de $N^{i,1}$ e $N^{i,2}$, evitando erros de redundância:

$$\alpha^i = u^i \cdot p^i = u^{i,1} \cdot p^{i,1} + u^{i,2} \cdot p^{i,2} \tag{3.11}$$

por fim, os produtos locais são acumulados em todos os processos via comunicação coletiva:

$$\alpha = \sum_{i=0}^{n-1} \alpha^i \tag{3.12}$$

No particionamento sem sobreposição, as combinações lineares de dois vetores são realizadas com todos os coeficientes em cada processo:

$$\begin{bmatrix} u_1^i \\ u_2^i \\ u_3^i \end{bmatrix} = \begin{bmatrix} u_1^i \\ u_2^i \\ u_3^i \end{bmatrix} + \alpha \begin{bmatrix} p_1^i \\ p_2^i \\ p_3^i \end{bmatrix} \tag{3.13}$$

Essa mesma operação em partições com sobreposição é realizada apenas com as componentes globais:

$$\begin{bmatrix} u_1^i \\ u_2^i \end{bmatrix} = \begin{bmatrix} u_1^i \\ u_2^i \end{bmatrix} + \alpha \begin{bmatrix} p_1^i \\ p_2^i \end{bmatrix} \tag{3.14}$$

3.6.2 Paralelização das operações de nível 2

O produto matriz-vetor é a principal operação dos métodos iterativos. Nos algoritmos adotados, há produtos matriz-vetor envolvendo matrizes esparsas e densas.

Matrizes esparsas:

No esquema sem sobreposição, cada processo realiza o seguinte produto:

$$\begin{bmatrix} p^{i,1} \\ p^{i,2} \\ p^{i,3} \end{bmatrix} = \begin{bmatrix} A_{11}^i & A_{12}^i & A_{13}^i \\ A_{21}^i & A_{22}^i & A_{23}^i \\ A_{31}^i & A_{32}^i & A_{33}^i \end{bmatrix} \begin{bmatrix} u^{i,1} \\ u^{i,2} \\ u^{i,3} \end{bmatrix} \tag{3.15}$$

Após ser realizada essa operação, os coeficientes resultantes $p^{i,2}$ e $p^{i,3}$ são parciais. No entanto, esses coeficientes de partições vizinhas se complementam, o que requer a comunicação desses valores entre os processos para a sequência do processo iterativo.

O produto matriz-vetor no esquema com sobreposição tem a forma:

$$\begin{bmatrix} p^{i,1} \\ p^{i,2} \end{bmatrix} = \begin{bmatrix} A^i_{11} & A^i_{12} & A^i_{13} & 0 \\ A^i_{21} & A^i_{22} & A^i_{23} & A^i_{24} \end{bmatrix} \begin{bmatrix} u^{i,1} \\ u^{i,2} \\ u^{i,3} \\ u^{i,4} \end{bmatrix} \tag{3.16}$$

As componentes $u^{i,3}$ e $u^{i,4}$ precisam ser obtidas de suas partições originais, nas quais foram globalmente calculadas, após o resgate o produto matriz-vetor é realizado em cada processo.

Matrizes densas:

No método de Arnoldi, um produto matriz-vetor denso precisa ser realizado para se obter o vetor resíduo do passo $i + 1$ (passo 16 do Algoritmo 2.2):

$$f_{i+1} = w_{i+1} - V_{i+1}h \tag{3.17}$$

Um passo correspondente a esse também é encontrado no GMRES (passo 7 do Algoritmo 3.7). Cada coluna da matriz densa V (ordem $neq \times k$) é resultante do produto-matriz vetor esparso realizado na ortonormalização de Gram-Schmidt, portanto a matriz V já se encontra particionada e seus coeficientes já estão globalmente computados. Enquanto o vetor h de ordem k é obtido via:

$$h = V^T_{i+1}w_i + 1 \tag{3.18}$$

Há duas formas de se executar esta operação: pode-se fazer um produto matriz-vetor com a transposta de V_{i+1}, ou alternativamente, adotando o seguinte produto interno:

$$(x, y) = x^T y \tag{3.19}$$

pode-se trocar o produto matriz-vetor por uma série de $i + 1$ produtos internos, onde i designa o passo que está sendo computado pelo algoritmo. Utilizando os

procedimentos para cálculo dos produtos internos já descritos, conclui-se que h é exatamente o mesmo em todos os processos. Esta operação, aliada ao cálculo do parâmetro β (norma do vetor resíduo), são as responsáveis pelo fato da matriz de Hessenberg está replicada em todos os processos. No caso simétrico, a Eq. 3.18 reduz-se a apenas o cálculo do novo α, o que é feito através de um produto escalar. Por isso, não há necessidade de comunicação no produto matriz-vetor denso mostrado na Eq. 3.17. Nessa operação, utiliza-se a rotina de nível dois *dgemv* da biblioteca BLAS [19] (Basic linear algebra set - conjunto básico de álgebra linear). Da mesma forma, na fase final do algoritmo, alguns produtos matriz-vetor precisam ser realizados para se obter os autovetores do problema original x a partir dos vetores de Ritz y:

$$x = Vy \tag{3.20}$$

onde y os vetores de Ritz, são obtidos pela solução do problema de autovalor da matriz de Hessenberg ou tridiagonal. Mais uma vez, a redundância dessas matrizes é vantajosa, eliminando-se a comunicação nessas operações, pois y também se encontra replicado. Porém, durante a fase de impressão final dos resultados, todos os processos precisam enviar seus autovetores x para o processo raiz.

3.6.3 Paralelização das operações de nível 3

Durante a fase de reinício implícito e da decomposição de Schur, dois produtos matriz-matriz densos:

$$H = Q^T HQ \tag{3.21}$$

$$T = Q^T TQ \text{ (caso simétrico)} \tag{3.22}$$

$$V = VQ \tag{3.23}$$

A matriz Q utilizada nesses produtos é obtida da fatoração QR com "shifts"duplos da matriz de Hessenberg ou tridiagonal. Devido a redundância dessas matrizes, os fatores Q e R são os mesmos em todos os processos. De forma semelhante ao produto matriz-vetor denso, não há comunicação nessa operação. A rotina de nível três da BLAS, *dgemm* é usada nesta operação.

3.7 Comunicação

O esquema de comunicação adotado utiliza uma biblioteca baseada no padrão MPI [28] de troca de mensagens. As vantagens do uso do MPI residem na sua portabilidade e disponibilidade, pois várias implementações deste padrão estão gratuitamente disponíveis em diversos repositórios na internet.

3.7.1 Comunicação nas operações de nível 1

Como foi visto, métodos baseados no processo de Gram-Schmidt requerem o cálculo de produtos escalares e normas dos vetores da base a ser gerada. Quando realizadas em paralelo, essas operações demandam comunicação coletiva, afim de que todos os processos tenham o resultado do produto interno ou da norma corretamente computado. Nesse caso, a rotina de comunicação coletiva do MPI, *MPI_ALLREDUCE*, são usadas para comunicar e somar os valores parciais de cada processo.

Para a combinação linear de vetores, não há necessidade de comunicação, pois a soma é realizada sobre a extensão local dos arranjos (Eqs. 3.13 para o caso sem sobreposição e Eqs. 3.14 para o caso com sobreposição). Para o caso sem sobreposição, há uma redundância na operação de soma devido aos coeficientes do tipo três.

3.7.2 Comunicação nas operações de nível 2

A eficiência de um produto matriz-vetor realizado em paralelo depende diretamente do esquema de comunicação adotado. Estudos comparativos sobre esquemas de

comunicação em rotinas de produto matriz vetor podem ser vistos em [64, 65]. Estes estudos comprovaram que rotinas de comunicação ponto a ponto do MPI, *MPI_ISEND* e *MPI_IRECV* , têm melhor desempenho do que as rotinas de comunicação coletiva. As rotinas de comunicação ponto a ponto imediata têm como vantagem o não bloqueio do fluxo ide execução do programa, sendo bastante adequadas para a estratégia de subdomínio por subdomínio.

No esquema de comunicação adotado [24], cada processo recebe um número de identificação (*id*) único que varia de zero a $n-1$, onde n é o número de processos. Cada processo, por sua vez, possui dois *buffers* de comunicação, f_1 (envio) e f_2 (recebimento) e uma lista de seus processos adjacentes. Os dois *buffers* estão ordenados de tal forma que os coeficientes a serem recebidos ou enviados sejam acessados seguindo a ordem crescente dos *id's* dos vizinhos. O tamanho dos *buffers* é igual ao total de coeficientes que devem ser comunicados. Dois arranjos contém os mapeamentos que relacionam as posições a serem comunicadas dos arranjos locais com suas posições nos respectivos buffers. O mapeamento das posições dos coeficientes a serem comunicados do aranjo p para o *buffer* de envio f_1 pode ser visto nas Figs. (3.4) e (3.5) para os casos sem e com sobreposição respectivamente. Como já foi visto na seção 3.7.2, os coeficientes que devem ser comunicados no particionamento sem sobreposição são aqueles do tipo 2 e 3 (blocos em azul e verde da Fig. (3.4)). Enquanto que no particionamento com sobreposição, os coeficientes a serem enviados são do tipo 1a e 2 (blocos em amarelo e azul da Fig (3.5)).

Durante a fase de comunicação imediata com *MPI_ISEND* e *MPI_IRECV*, um arranjo auxiliar contém o mapeamento de quais blocos de coeficientes do *buffer* f_1 devem ser enviados a cada vizinho. De forma semelhante, durante o recebimento, há um arranjo auxiliar que faz a relação das posições do *buffer* f_2 que devem ser escritas por cada vizinho. Um segundo mapeamento é realizado entre as posições do *buffer* f_2 e as posições do arranjo p. Para o caso sem sobreposição, mostrado na Fig. (3.6), os coeficientes recebidos são mapeados e somados com os respectivos coeficientes do tipo 2 e 3 do arranjo local p. No caso do particionamento com

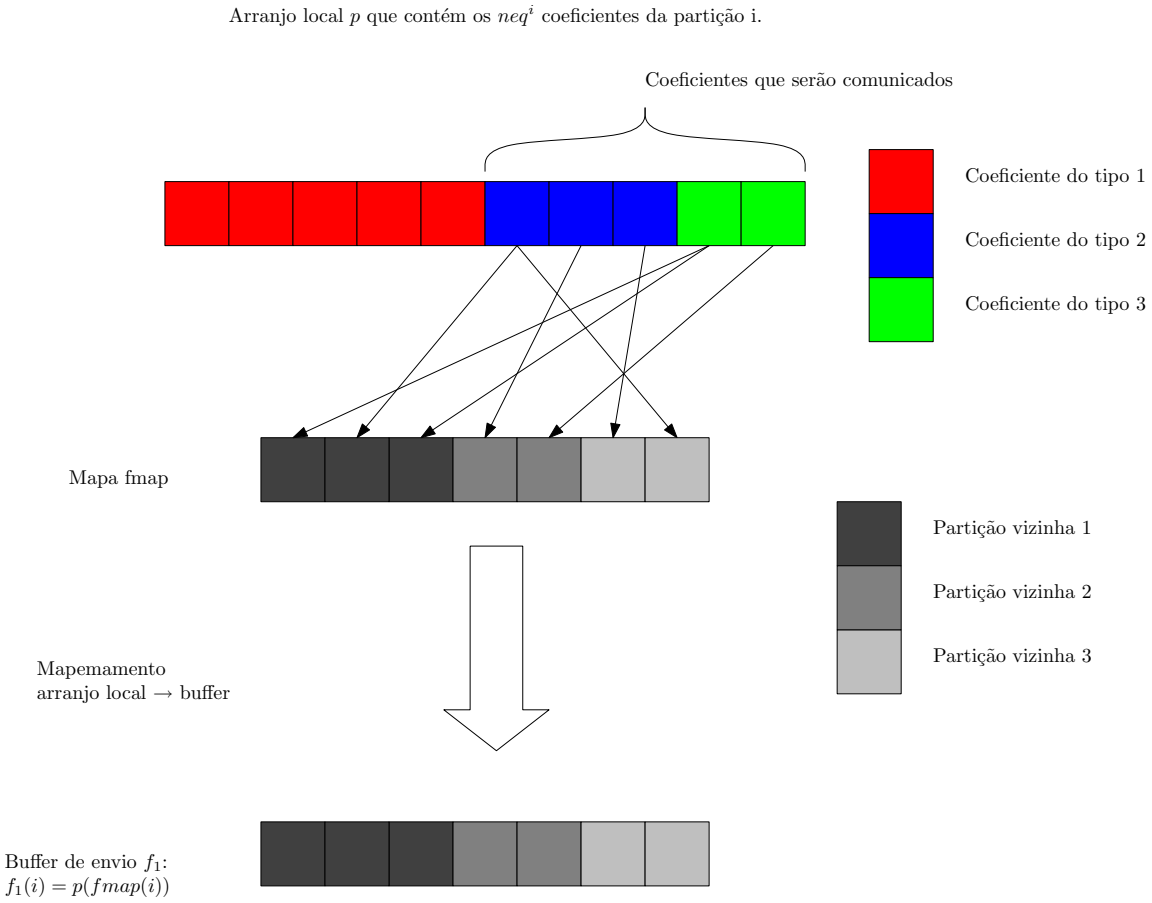

Figura 3.4: Mapeamento entre posições do arranjo local p para o buffer de envio f_1 (caso sem sobreposição)

sobreposição, mostrado na Fig. (3.7), os coeficientes são mapeados para as posições 3 e 4 do arranjo local p. Como os coeficientes recebidos já se encontram globalmente computados, os valores em p referentes a esses coeficientes são sobrescritos.

Uma característica do método sem sobreposição é que toda a comunicação entre os processos é bidirecional. Isso se deve ao próprio esquema de particionamento, pois na fronteira entre dois processos, todo nó do tipo dois equivale a um nó do tipo três na partição vizinha e vice-versa. Como consequência, os mapeamentos que relacionam as posições dos *buffers* de comunicação com os arranjos locais são idênticos. O mesmo não acontece com o esquema com sobreposição, a comunicação entre dois processos pode ser unidirecional, pois no particionamento com sobreposição, não há uma obrigatória correspondência bidirecional como no caso de partições sem

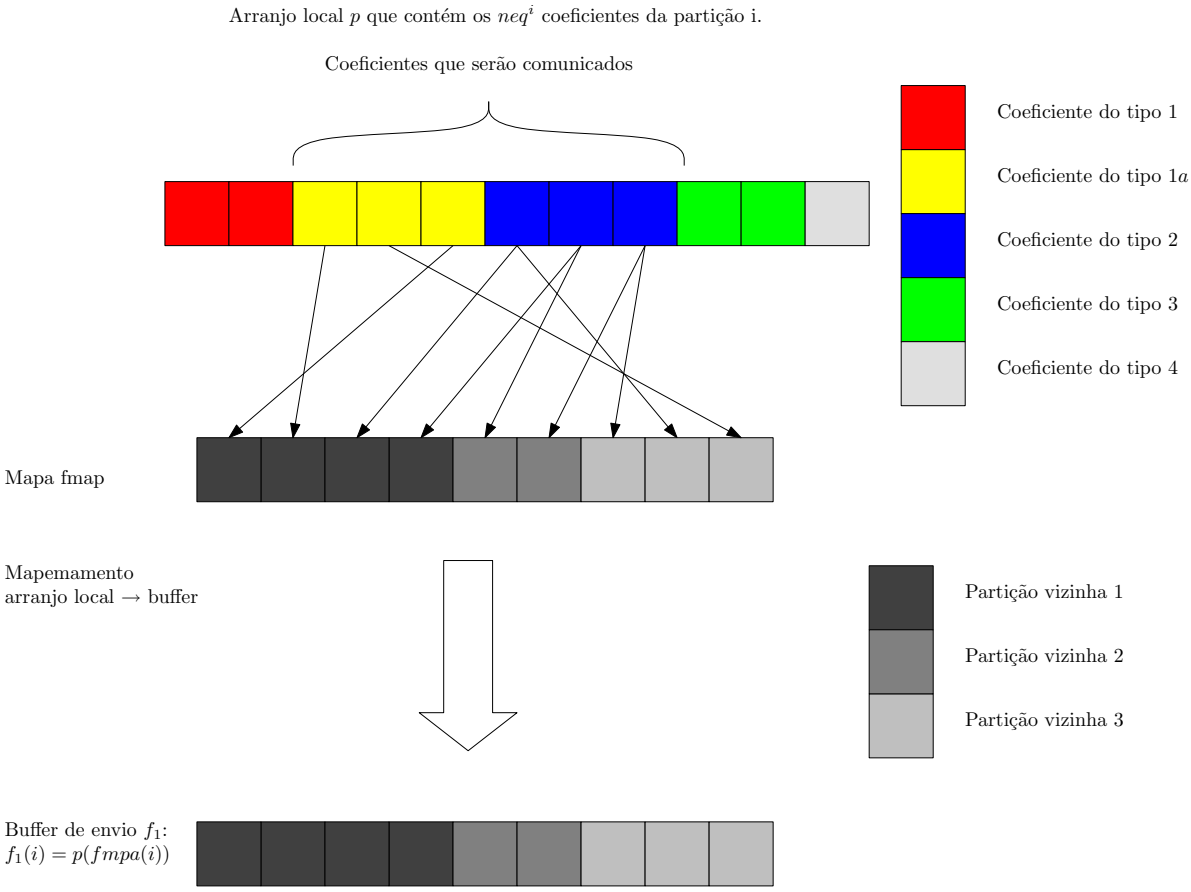

Figura 3.5: Mapeamento entre posições do arranjo local p para o buffer de envio f_1 (caso com sobreposição)

sobreposição.

3.7.3 Comunicação nas operações de nível 3

Conforme já foi dito na seção 3.6.3, não há comunicação nessas operações, pois as matrizes envolvidas nesse tipo de operação, H, T e Q, são replicadas em cada processador (ver Eqs. 3.21 a 3.23).

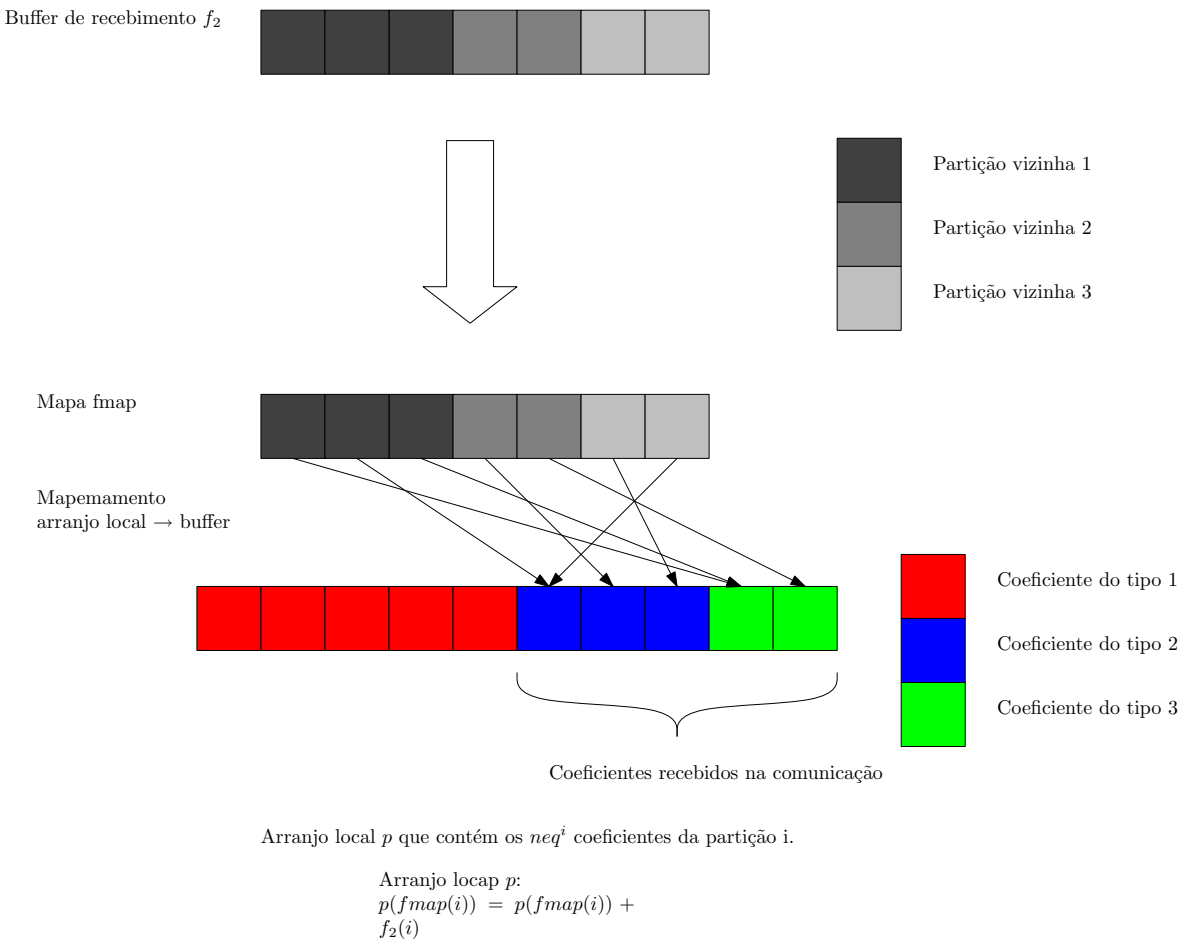

Figura 3.6: Mapeamento entre posições do buffer f_2 para o arranjo local (caso sem sobreposição)

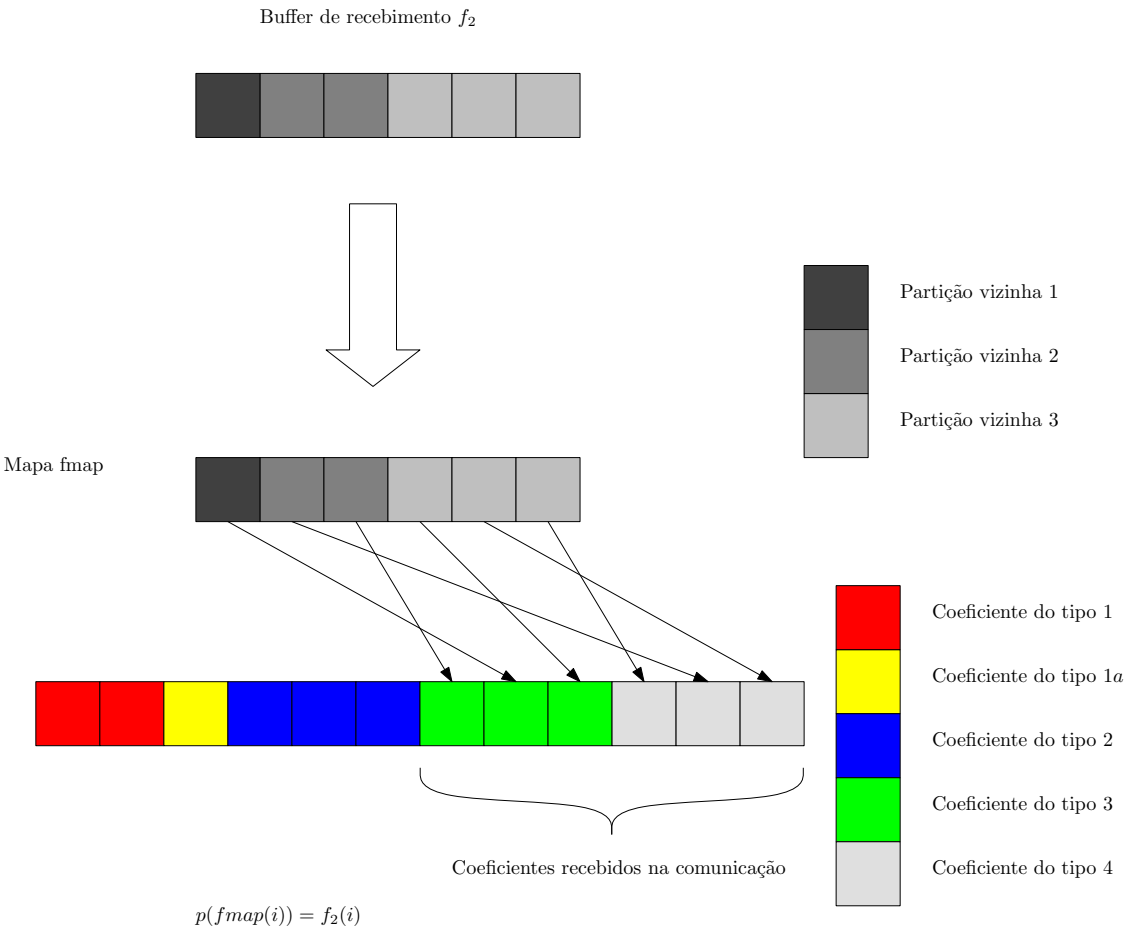

Figura 3.7: Mapeamento entre posições do buffer f_2 para o arranjo local (caso com sobreposição)

Capítulo 4

Exemplos Analisados

Apresenta-se neste capítulo alguns resultados da implementação proposta do método de Arnoldi/Lanczos com Reinício Implícito. Alguns exemplos foram escolhidos por serem tradicionais "benchmarks" para rotinas de determinação de autovalores e autovetores, outros exemplos são típicos da análise estrutural.

4.1 Critério de Convergência Adotado

Após k passos do algoritmo (2.1) a fatoração de Arnoldi/Lanczos é obtida:

$$CV_k = V_k H_k + f_k e_k^T \tag{4.1}$$

onde, V_k é a matriz cujas colunas são os vetores de Lanczos/Arnoldi, H_k é a matriz de Hessenberg de ordem k, f_k é o k-ésimo vetor resíduo e e_k é o k-ésimo vetor da base canônica do $\mathbb{R}^n$. Para problemas de autovalor generalizado, a matriz C pode ser $C = A^{-1}B$ se os autovalores desejados forem o de menor valor absoluto ou caso contrário, $C = B^{-1}A$.

Uma vez que foi adotada uma aproximação para os autovetores do tipo ($\hat{x} = V_k c$), o problema de autovalor original é aproximado por:

$$H_k c = \hat{\lambda} c \tag{4.2}$$

Como o par de Ritz $(\hat{x}, \hat{\lambda})$ é uma aproximação, a norma do resíduo $(r = C\hat{x} - \hat{\lambda}\hat{x})$ é determinada por:

$$||C\hat{x} - \hat{\lambda}\hat{x}|| = ||CV_k c - V_k H_k c|| = ||f_k|||e_k^T c| \tag{4.3}$$

Lehoucq ([39]) baseando-se na eq. (4.3) e nas propriedades de não-normalidade $(A^H A \neq AA^H)$ e defectividade (presença de autovalores repetidos), propôs que o par $(\hat{x}, \hat{\lambda})$ é uma boa aproximação para os autovalores e autovetores de C, quando a seguinte desigualdade é satisfeita:

$$||f_k|||e_k^T c| \leq tol \cdot |\hat{\lambda}| \tag{4.4}$$

onde, *tol* é a tolerância especificada.

4.2 Resultados Obtidos com Execução Sequencial

Os exemplos numéricos foram escolhidos com o intuito de testar o método implementado, bem como verificar sua aplicabilidade, tanto em problemas simétricos quanto não-simétricos. O primeiro exemplo é um problema de autovalor generalizado simétrico típico da análise estrutural. Os dois exemplos seguintes são *benchmarks* conhecidos que fazem parte do conjunto de matrizes para testes de rotinas do repositório *Matrix Market* [66]. Esses exemplos são problemas de autovalor padrão não-simétricos.

4.2.1 Arquibancada do estádio Mário Filho (Maracanã)

Na dinâmica de estrutural a determinação das frequências naturais e modos de vibração recai na solução de um problema de autovalor simétrico. Em geral, apenas os modos mais baixos são relevantes. O modelo utilizado representa a arquibancada do estádio do Maracanã. A malha de elementos finitos gerada contém 115 elementos de pórtico plano, totalizando 285 equações. Para este caso, foram calculados os quatro menores autovalores, sendo necessário apenas um reinício implícito, com

tolerância de 1×10^{-25}. O método de iteração por subespaço [6, 67] é um método consagrado na engenharia estrutural e foi utilizado na verificação dos resultados obtidos da implementação proposta. Uma comparação entre as quatro primeiras frequências naturais obtidas numericamente pode ser vista na tabela:

Tabela 4.1: Comparação das frequências naturais obtidas numericamente

Frequências obtidas com o método proposto (hz)	Frequências obtidas com o método de iteração por subespaço (hz)	Diferença (%)
2.6387	2.6385	0.008
3.2587	3.2579	0.025
4.1682	4.1683	0.002
7.1227	7.1311	0.118

como pode ser visto, as frequências naturais obtidas pelo método de Lanczos com reinício implícito estão bastante próximas das frequências calculadas pela iteração por subespaço.

As formas modais podem ser vistas nas figuras:

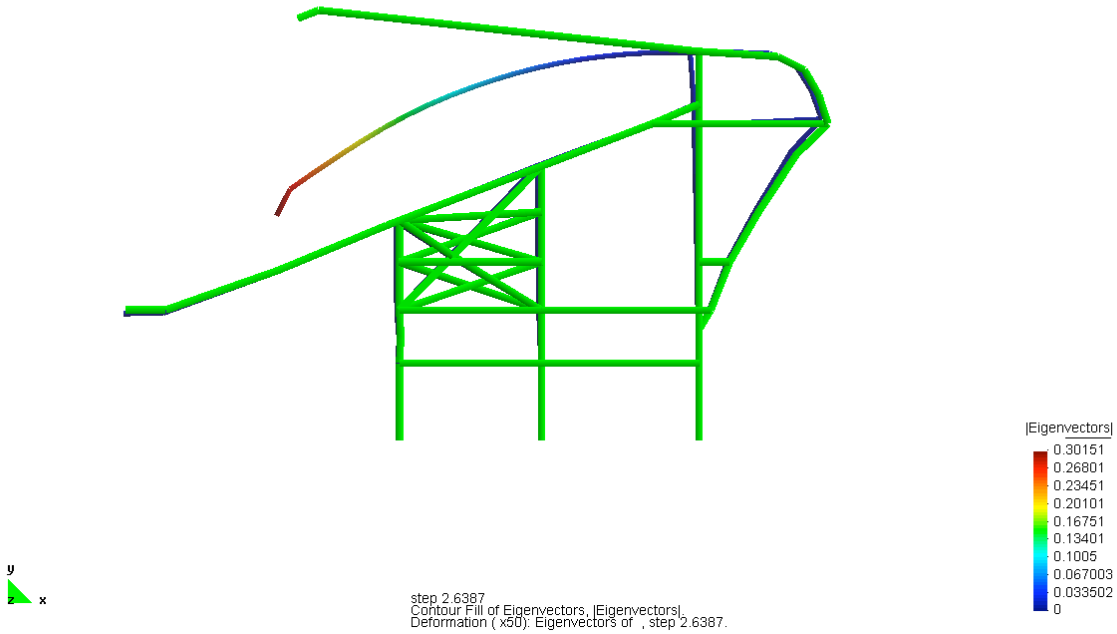

Figura 4.1: Modo de vibração correspondente à $f = 2.6387$

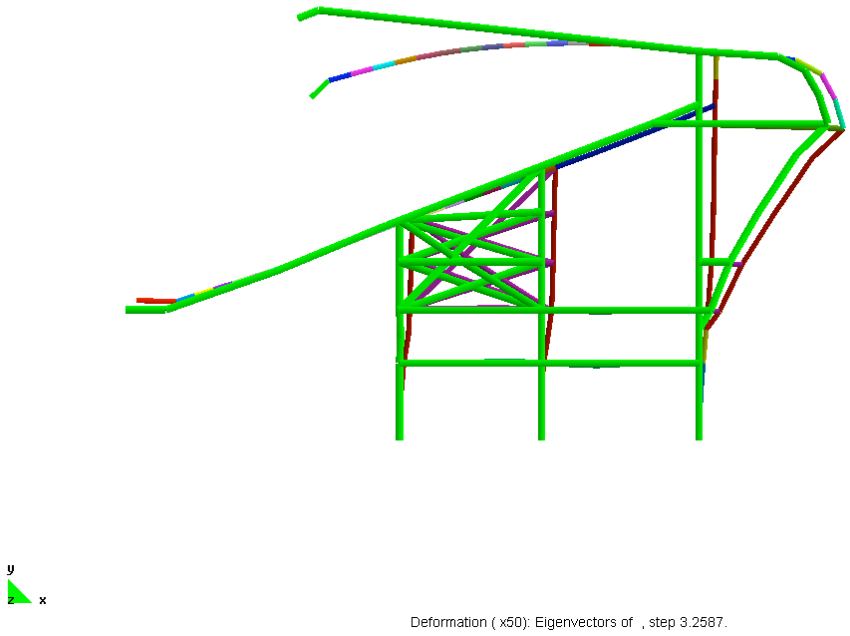

Figura 4.2: Modo de vibração correspondente à $f = 3.2587$

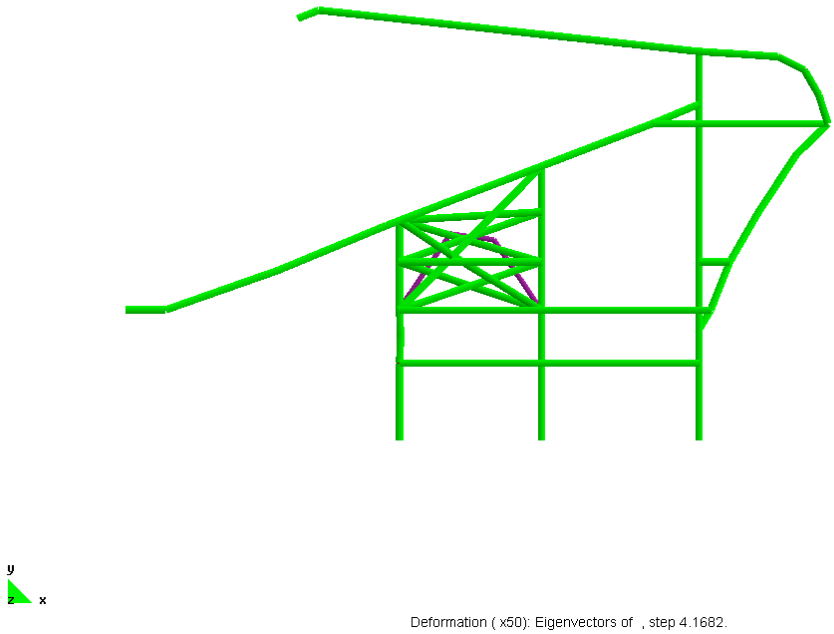

Figura 4.3: Modo de vibração correspondente à $f = 4.1682$

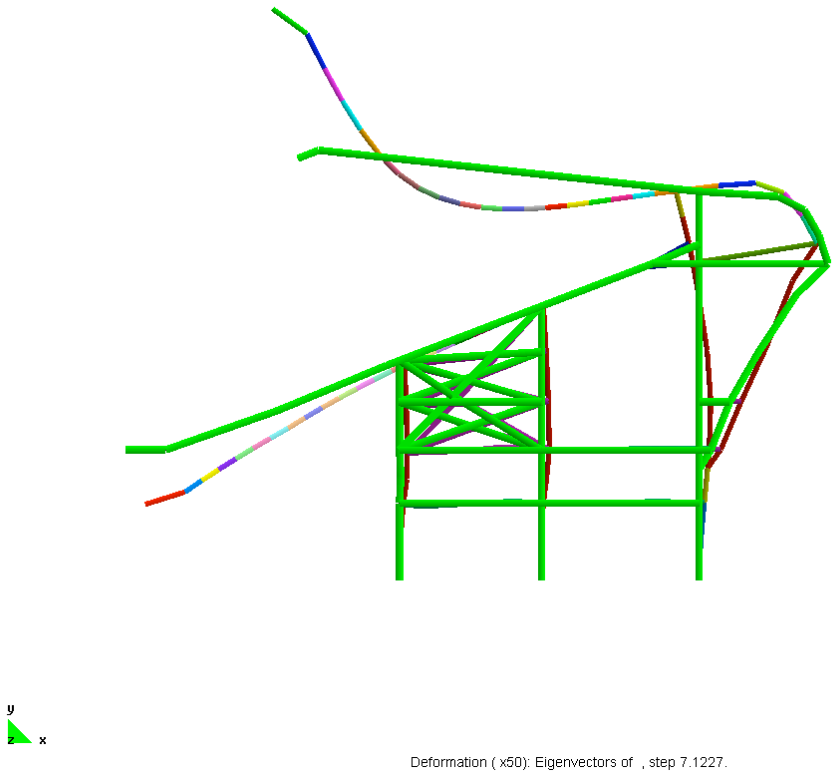

Figura 4.4: Modo de vibração correspondente à $f = 7.1227$

4.2.2 Benchmarks da Coleção do Matrix-Market

Matriz de Tolosa

A matriz de Tolosa é um conhecido *benchmark* da coleção NEP (*Non-Hermitian Eigenvalue Problem* - problema de autovalor não-hermitiano) [66]. Essa matriz é originada de problemas que surgem na análise da estabilidade de asas de avião em pleno vôo. Neste exemplo, foi simulada a matriz de Tolosa com 2000 equações, os autovalores mais relevantes são aqueles que possuem maior parte imaginária. Trata-se de um problema não-simétrico mal condicionado com muitos autovalores próximos. Para efeito de comparação dos resultados, são mostrados na tabela (4.2)

os resultados obtidos com o ARPACK, nesta simulação foi usada uma base de Krylov de dimensão igual a 300 e dois reinícios foram executados.

Tabela 4.2: Os seis autovalores com maior parte imaginária da matriz de Tolosa (n=2000) calculados com o ARPACK (fonte: [**?**])

parte real de λ	parte imaginária λ
-723.2940	-2319.859
-723.2940	2319.859
-726.9866	-2324.992
-726.9866	2324.992
-730.6886	-2330.120
-730.6886	2330.120

Com o objetivo de avaliar a implementação do método de Arnoldi com reinício implícito, o mesmo problema foi simulado, calculando-se os seis autovalores de maior parte imaginária, com uma base do subespaço de Krylov de mesma dimensão usada no ARPACK. O processo de reinício implícito não precisou ser executado, isto é, os autovalores convergiram após a aplicação dos primeiros 300 passos. Os resultados podem ser vistos na tabela (4.3):

Tabela 4.3: Os seis autovalores com maior parte imaginária da matriz de Tolosa (n=2000) obtidos com a implementação proposta

parte real de λ	parte imaginária λ
-723.2940	-2319.8590
-723.2940	2319.8590
-726.9866	-2324.9917
-726.9866	2324.9917
-730.6886	-2330.1198
-730.6886	2330.1198

Como pode ser visto, os resultados obtidos pelo ARPACK e pela rotina proposta

são numericamente os mesmos. A diferença no número de iterações pode ser justificada pela tolerância menor que o ARPACK usa por padrão, nesse caso a tolerância usada na implementação proposta foi de 1×10^{-15}, a qual, como pode ser visto, conduz a bons resultados.

Problema de convecção-difusão

O problema de autovalor associado à discretização dos métodos usados na aproximação do problema de convecção-difusão (4.5) é importante na análise da estabilidade e do comportamento oscilatório da solução aproximada [68]. A aproximação desse problema foi feita pelo método das diferenças finitas utilizando um *grid* de $m \times m$ pontos e integra a coleção NEP. Neste *benchmark* a equação do problema de convecção-difusão é dada por:

$$-\Delta u(x,y) + 2p_1 u_x(x,y) + 2p_2 u_y(x,y) = \lambda(p_3 u(x,y)) \tag{4.5}$$

definida num quadrado unitário ($\Omega = [0,1] \times [0,1]$), com $u = 0$ em todo o contorno. Neste problema de autovalor padrão de ordem 961, foram calculados seis autovalores de maior módulo, sendo necessárias 79 iterações para a convergência dos mesmos. A dimensão da base do subespaço de Krylov utilizada nesse exemplo foi 40. Para este exemplo, o (k,l)-ésimo autovalor é dado pela equação (4.6):

$$\lambda_{k,l} = 4 - \sigma + 2(1-\beta^2)^{\frac{1}{2}} \cos(kh\pi) + 2(1-\gamma^2)^{\frac{1}{2}} \cos(lh\pi) \tag{4.6}$$

onde, $h = \frac{1}{m+1}$, $\beta = p_1 h$, $\gamma = p_2 h$, $\sigma = p_3 h^2$.

Os seis maiores autovalores obtidos pela Eq. (4.6) são mostrados na tabela:

Tabela 4.4: Os seis autovalores do problema de convecção-difusão ($m = 31$,$p_1 = 25$,$p_2 = 50$ e $p_3 = 250$)

parte real de λ	parte imaginária de λ	(k, l)
4.9983	2.3896	(1,1)
4.9983	-2.3896	(1,31)
4.9982	2.3551	(1,2)
4.9982	-2.3551	(1,30)
4.9803	2.3896	(2,1)
4.9803	-2.3896	(2,31)

Tabela 4.5: Os seis autovalores do problema de convecção-difusão obtidos numericamente

parte real de λ	parte imaginária de λ
4.9982	2.3896
4.9982	-2.3896
4.9982	2.3550
4.9982	-2.3550
4.9803	2.3896
4.9803	-2.3896

A distribuição de todos os autovalores desse *benchmark* pode ser visto na figura:

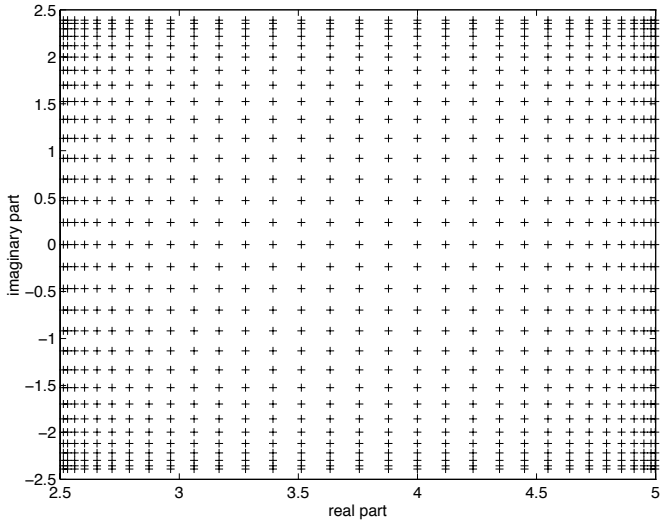

Figura 4.5: Autovalores do *benchmark* do problema de convecção-difusão (fonte: site do *Matrix-Market*)

Apesar de ser um problema melhor condicionado do que a matriz de Tolosa, o mair número de iterações necessárias para a convergência dos autovalores é justificada pela presença de muitos autovalores próximos, como pode ser observado na Fig (4.5). Comparando os autovalores exatos (tab. 4.4) e numéricos (tab. 4.5), nota-se que estão bem próximos, indicando que a implementação proposta está bem acurada.

4.3 Resultados obtidos com execução em paralelo

Uma medida de performance do código paralelo é o *speedup*. O *speedup* de um programa paralelo é definido como sendo a razão entre o tempo que uma tarefa é executa por um único processo e o tempo de execução da mesma tarefa executada por n processos, isto é, o *speedup* mostra quantas vezes o código paralelo é mais rápido que o sequencial, esta razão é expressa por:

$$S(n) = \frac{T(1)}{T(n)} \tag{4.7}$$

Em geral, $1 \leq S(n) \leq n$, quando o *speedup* é maior que n, tem-se um *superspeedup*. Além do *speedup*, a eficiência é outro parâmetro utilizado como medida do uso

dos processos, sendo dada por:

$$e(n) = \frac{S(n)}{n} \tag{4.8}$$

Para a execução em paralelo dos exemplos, foi utilizado um *cluster* de *pc's* com 32 nós, cada nó possui um processador de núcleo duplo Intel Core2Duo com 8GB de memória RAM conectados via LAN com um *switch* de 1GB. Os processadores têm *clock* de 2.66 GHz e *cache* compartilhado de 6 MB. Todos os nós rodam o Cent OS Linux com kernel 2.6.18, o compilador utilizado foi o Intel Fortran versão 10.1. Para a obtenção dos particionamentos da malhas foi utilizada uma biblioteca do programa METIS [69]. Como o processador Intel Core2Duo possui núcleo duplo, pode-se executar dois processos em cada máquina. Para melhor comparar os resultados dos *speedups* , adotou-se o seguinte critério: as curvas de *speedups* foram traçadas com referência ao número de nós ou máquinas ao invés do número total de processos. Quando apenas um núcleo do processador foi usado, o número de processos e máquinas (nós) são os mesmos. Na utilização da tecnologia de núcleo duplo, cada nó executou dois processos diferentes, sendo o *speedup* dado por:

$$S(n)_{\text{núcleo duplo}} = \frac{T(2)}{T(2 * n)} \tag{4.9}$$

onde n designa o número de nós.

4.3.1 Caso 1: problema de elasticidade plana

Neste exemplo de elasticidade plana foi utilizado uma viga engastada discretizada com elementos quadriláteros bilineares. Trata-se de um problema de autovalor generalizado simétrico, no qual foram calculados os três menores autovalores, sendo utilizado um total de 20 passos no algoritmo de Lanczos com reinício implícito. Neste exemplo, foi preciso apenas dois reinícios implícitos para que todos os autovalores convergissem. Como era esperado, o número de iterações do PCG e do método de Lanczos com reinício implícito foi o mesmo, tanto na execução sequencial quanto na paralela. A tolerância empregada no cálculo dos autovalores foi de 1×10^{-15}. A viga

simulada tem 4 metros de comprimento, sessenta centímetros de altura e espessura de 0.20 cm . Possui módulo de elasticidade de $24 \times 10^9 N/m^2$ e peso específico de 2200 N/m^3. As principais características desse exemplo são mostradas na tabela:

Tabela 4.6: Informações da simulação do caso 1

Número de elementos	140,000
Número de equações	280,800
Número de iterações do método de Lanczos com reinício implícito	2
Número de soluções do sistema de equações (PCG)	36

Exemplos de particionamentos para 4, 32 e 64 processos são mostrados nas figuras:

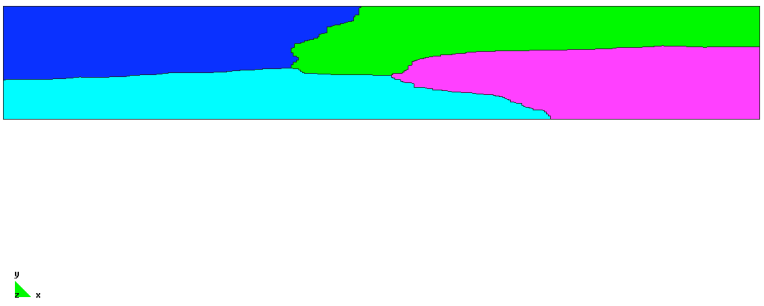

Figura 4.6: Particionamento em 4 subdomínios

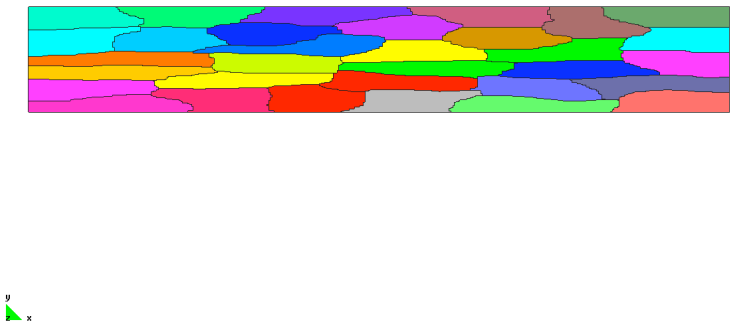

Figura 4.7: Particionamento em 32 subdomínios

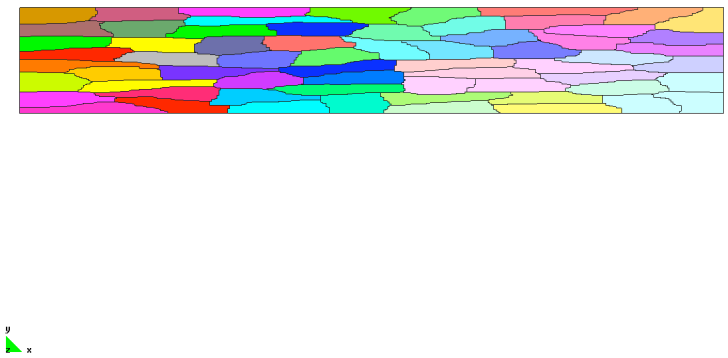

Figura 4.8: Particionamento em 64 subdomínios

As três formas modais obtidas são mostradas nas figuras:

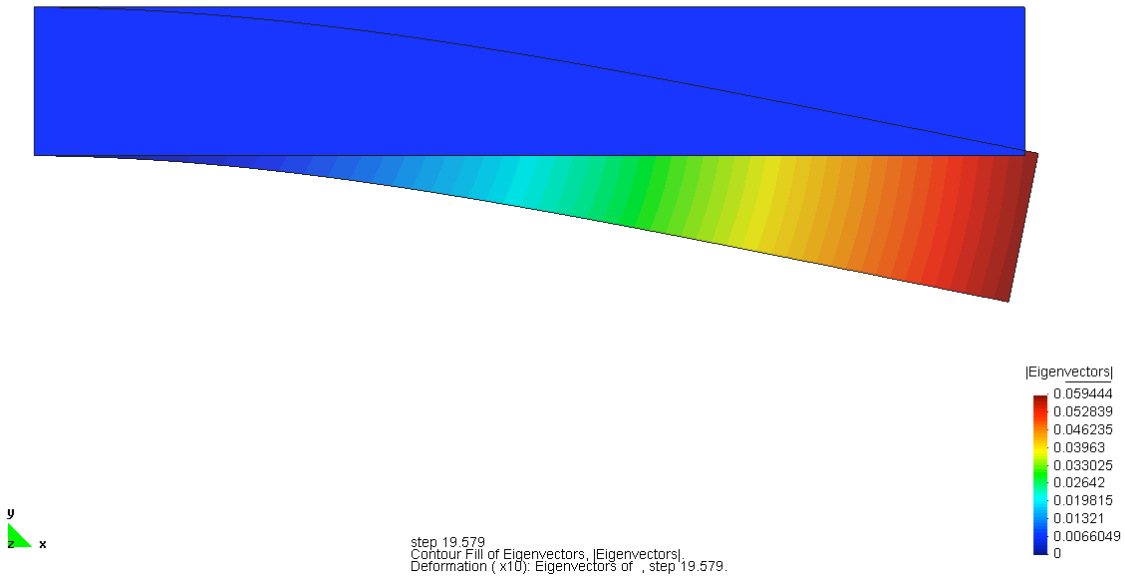

Figura 4.9: Primeira forma modal $f = 19.59$ Hz

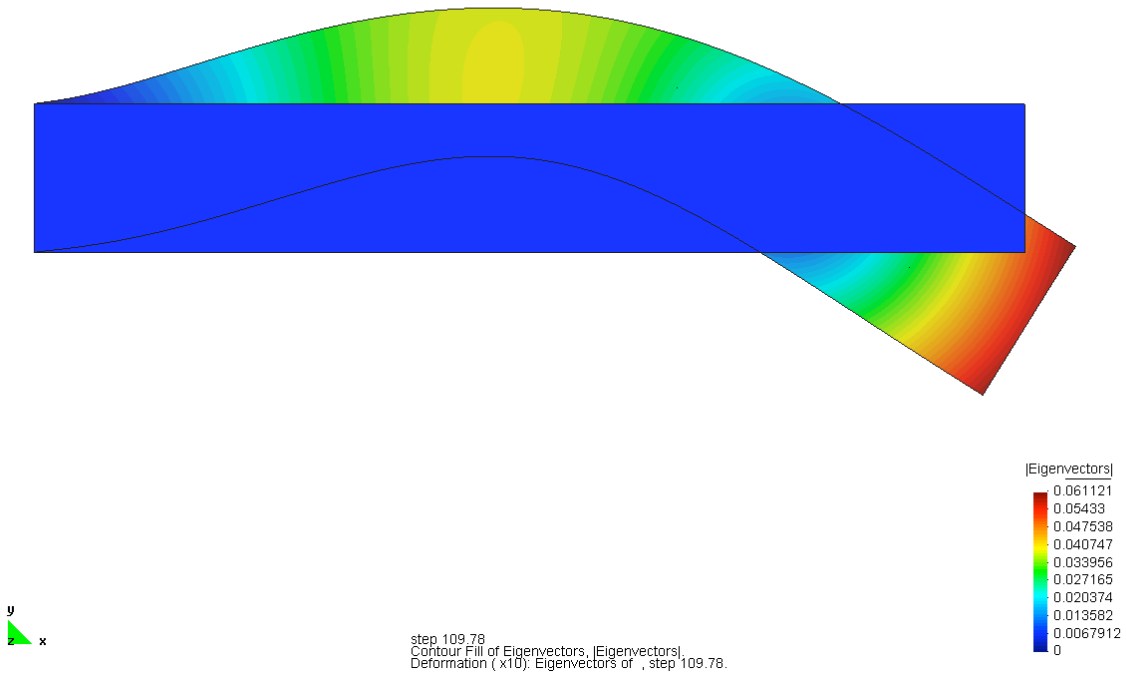

Figura 4.10: Segunda forma modal $f = 109.78$ Hz

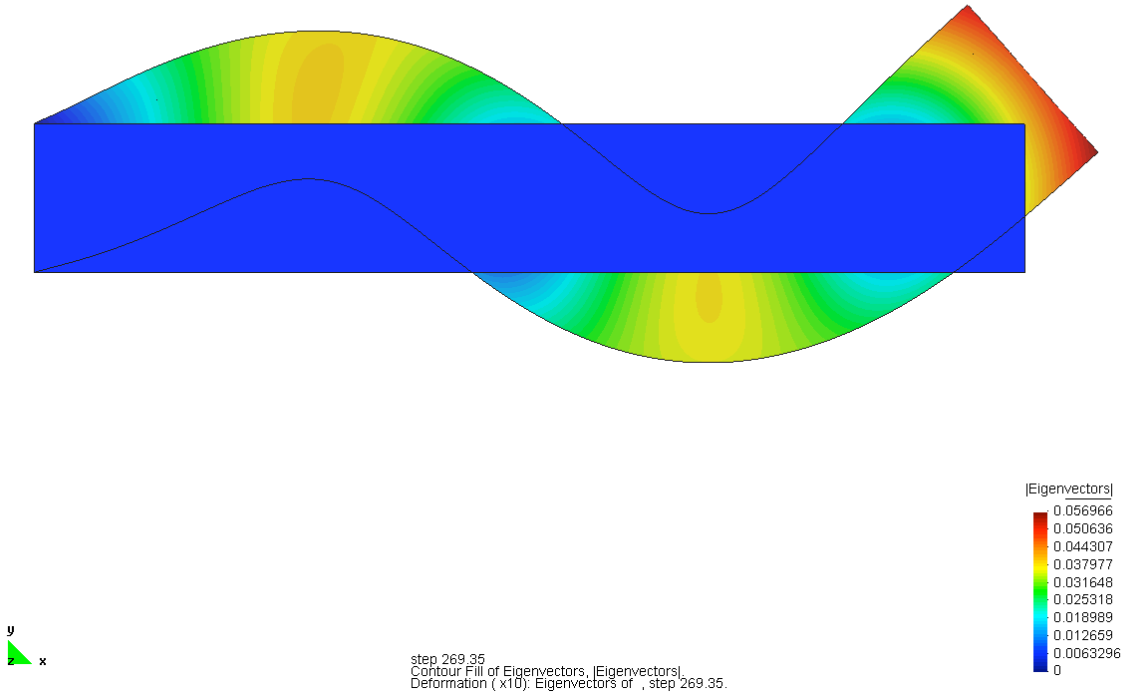

Figura 4.11: Terceira forma modal $f = 269.35$ Hz

A Fig. (4.12) apresenta as curvas de *speedup* para a etapa de cálculo das matrizes de elemento e montagem das matrizes globais.

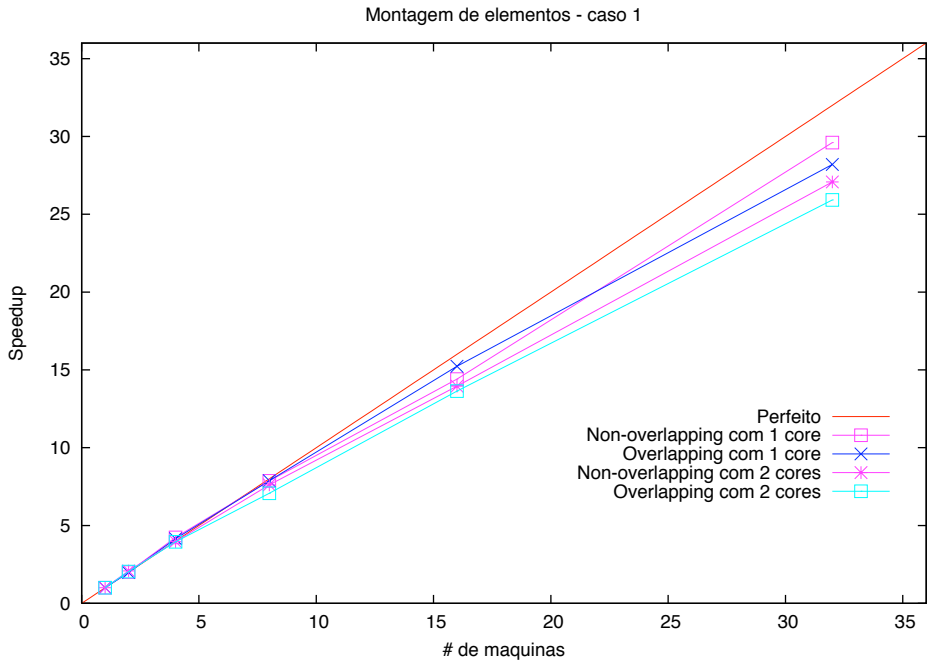

Figura 4.12: *Speedups* para a montagem de elementos do caso 1.

Por não possuir redundância no cálculo das matrizes de coeficientes, o particionamento sem sobreposição (*non-overlapping*) é mais vantajoso do que o caso com sobreposição (*overlapping*) no cálculo das matrizes de elemento e espalhamento na matriz global. Para um número de processos menor que 16, os *speedups* estão próximos ao ideal.

São apresentados na Fig. (4.13) os *speedups* para o produto matriz-vetor. Nesta operação não estão computados os tempos de comunicação, por isso a ocorrência de *superspeedups* para número de máquinas maior que dois. Um aumento de número de processos em paralelo e subsequente diminuição da quantidade de memória utilizada por processo na operação, melhorou o uso da memória *cache*, resultando em melhores *speedups*, principalmente quando os dois núcleos da máquina foram usados.

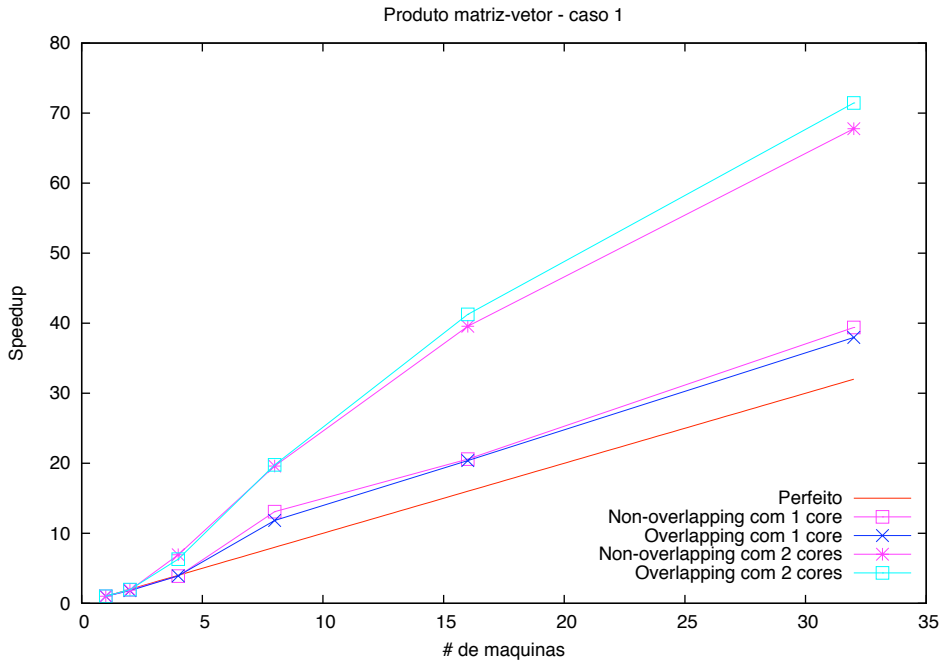

Figura 4.13: *Speedups* para o produto matriz-vetor caso 1.

A comunicação referente à operação de produto-matriz vetor é contabilizada na etapa de solução do sistema de equações. Os *speedups* do método de Lanczos com reinício implícito e do tempo total de execução é governado pelo *speedup* do PCG, como pode ser visto nas figuras (4.14) a (4.16):

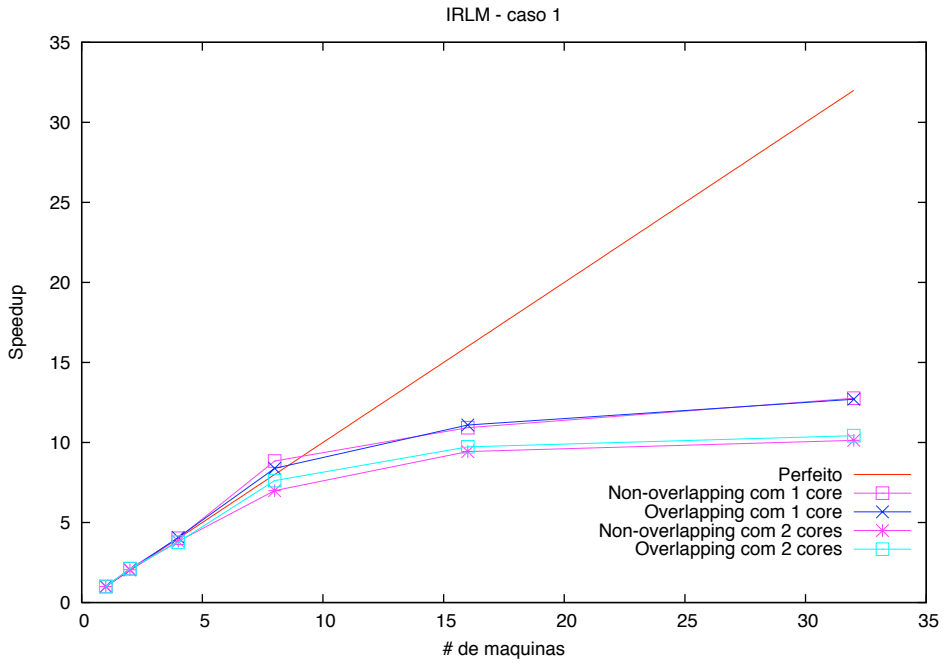

Figura 4.14: *Speedups* para a etapa de solução do problema de autovalor do caso 1.

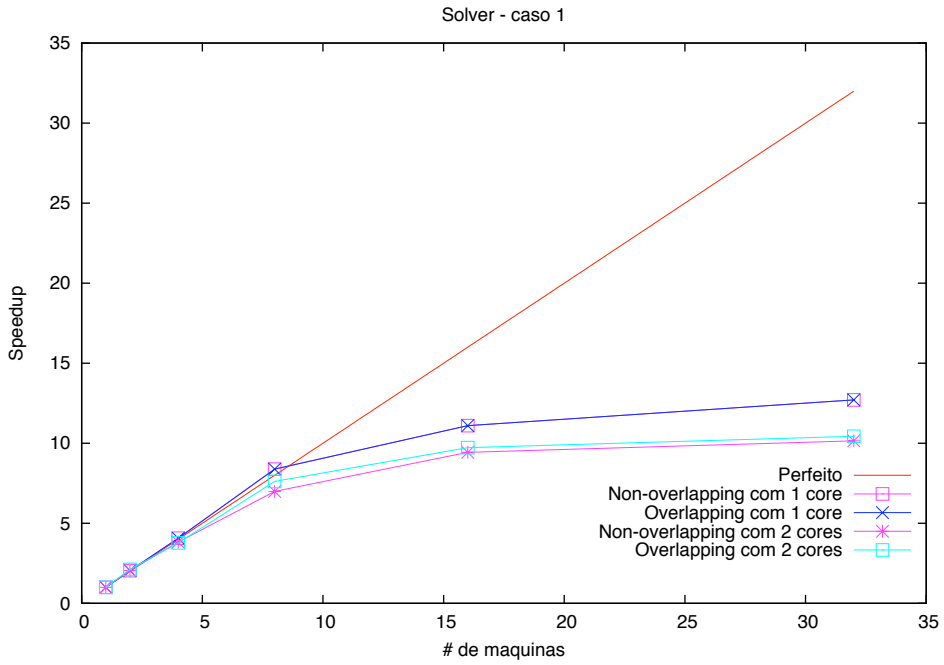

Figura 4.15: *Speedups* para a etapa de solução do sistema de equações (PCG) do caso 1.

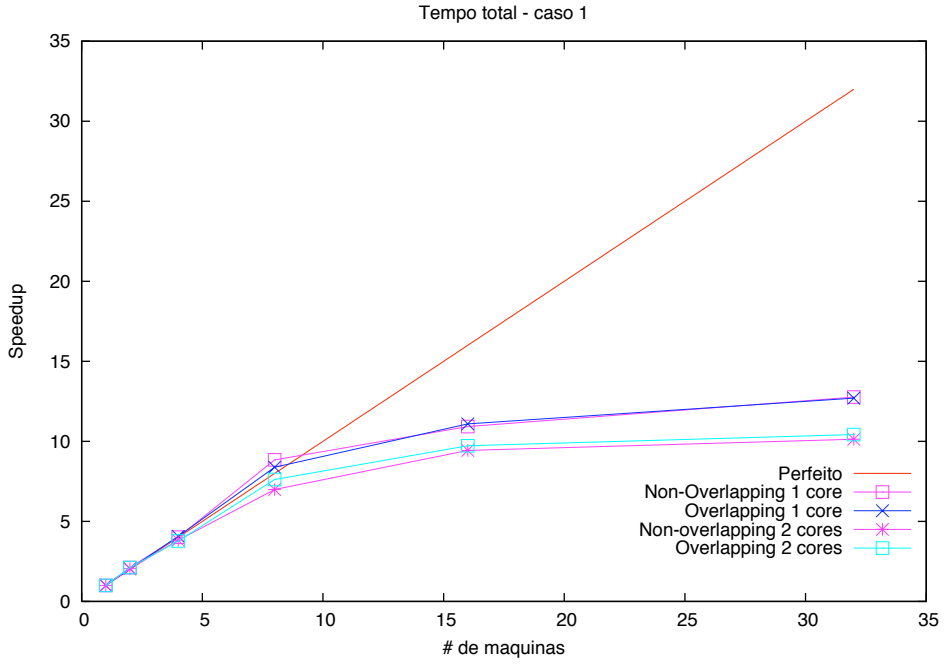

Figura 4.16: *Speedups* para o tempo total de execução do caso 1.

Obteve-se *superspeedups* para um número de processos menor que 16, tal fato pode ser explicado pela otimização da memória *cache*. Nesses casos, a fronteira

interna gerada é pequena, por isso o tempo de comunicação é pequeno se comparado às outras operações. À medida que o número de processos aumentou, mais fronteira interna foi crida (vide Figs. 4.6,4.7,4.8) afetando a eficiência. A convergência no PCG e no método de Lanczos não foi alterada, independente do número de processos usados. Os *speedups* máximos obtidos são mostrados na tabela (4.7):

Tabela 4.7: *Speedups* máximos obtidos para o caso 1

Etapa	Máximo *speedup*	Método de particionamento (número de processos)
Montagem de elementos	29.62	Sem sobreposição (1 ×32)
Produto matriz-vetor (sem comunicação)	71.43	Com sobreposição (2 ×32)
PCG	12.70	Sem sobreposição (1 ×32)
Lanczos com reinício impl.	12.76	Sem sobreposição (1 ×32)
Total	12.77	Sem sobreposição (1 ×32)
Tempos totais em (s)	Sequencial	Mínimo paralelo
	5,346.24	418.77

4.3.2 Caso 2: casa de força da UHE de Peixe Angical

Como segundo exemplo, foram calculados as frequências naturais e modos de vibração de um modelo tridimensional discretizado com elementos tetraédricos da UHE Peixe Angical. A malha correspondente ao modelo pode ser vista na figura (4.17):

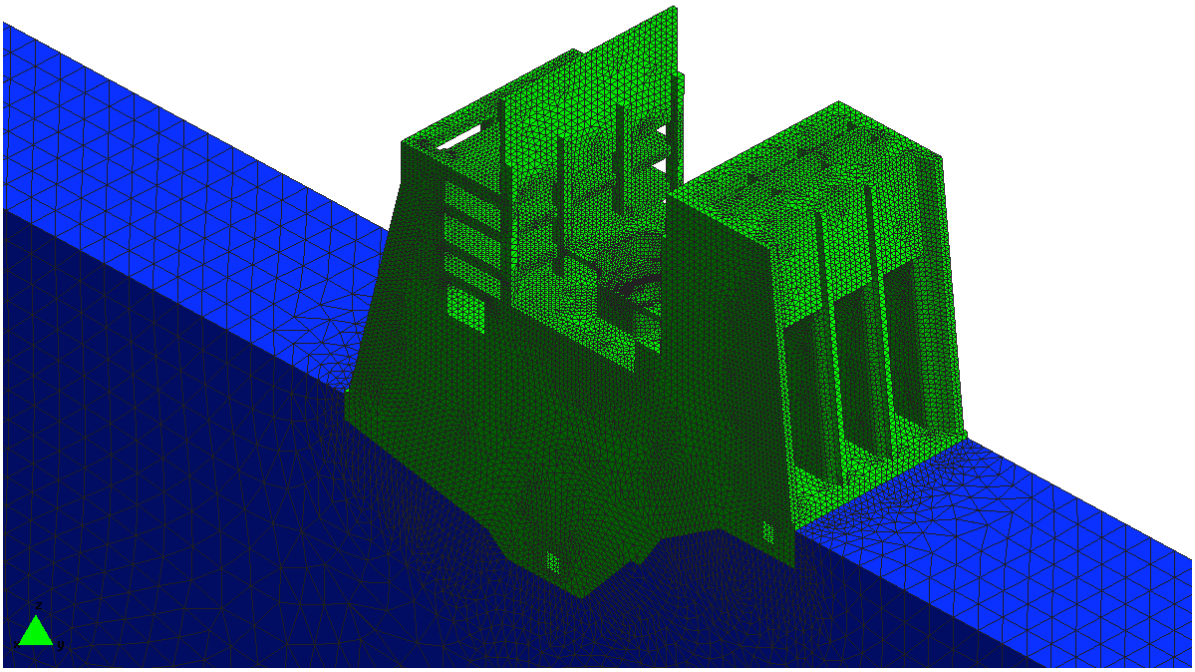

Figura 4.17: Malha do caso 2

As propriedades físicas utilizadas nesse exemplo estão explicitadas na tabela (4.8):

Tabela 4.8: Propriedades físicas utilizadas no modelo da UHE Peixe Angical

Propriedades Físicas	Casa de Força	Solo
Módulo de elasticidade (MPa)	25×10^3	53×10^3
Coef. de Poisson (sem dimensão)	0.3	0.3
Peso específico (N/m^3)	23×10^3	0.0

O solo foi considerado na modelagem para melhor representar as condições de restrição à casa de força, seus efeitos inerciais foram considerados nulos. Neste exem-

plo foram calculados 15 autovalores, sendo executados, no total, 60 passos. Para a convergência dos autovalores, foram realizados dois reinícios implícitos e tolerância adotada no critério de convergência foi de 1×10^{-13}. As principais características da simulação são mostradas na tabela (4.9):

Tabela 4.9: Informações da simulação do caso 2

Número de elementos	899,104
Número de equações	546,587
Número de iterações do método de Lanczos com reinício implícito	2
Número de soluções do sistema de equações (PCG)	115

As seis primeiras frequências são mostradas nas figuras de (4.18) a (4.23):

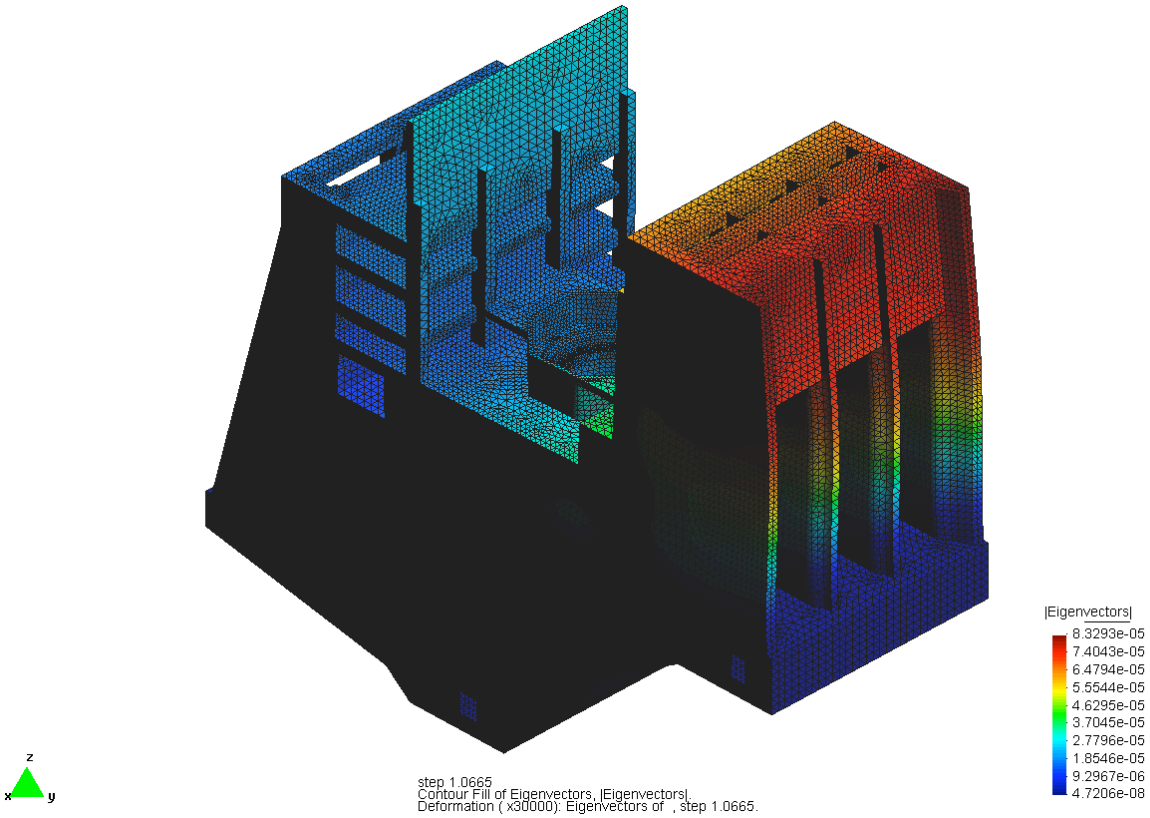

Figura 4.18: Primeira forma modal

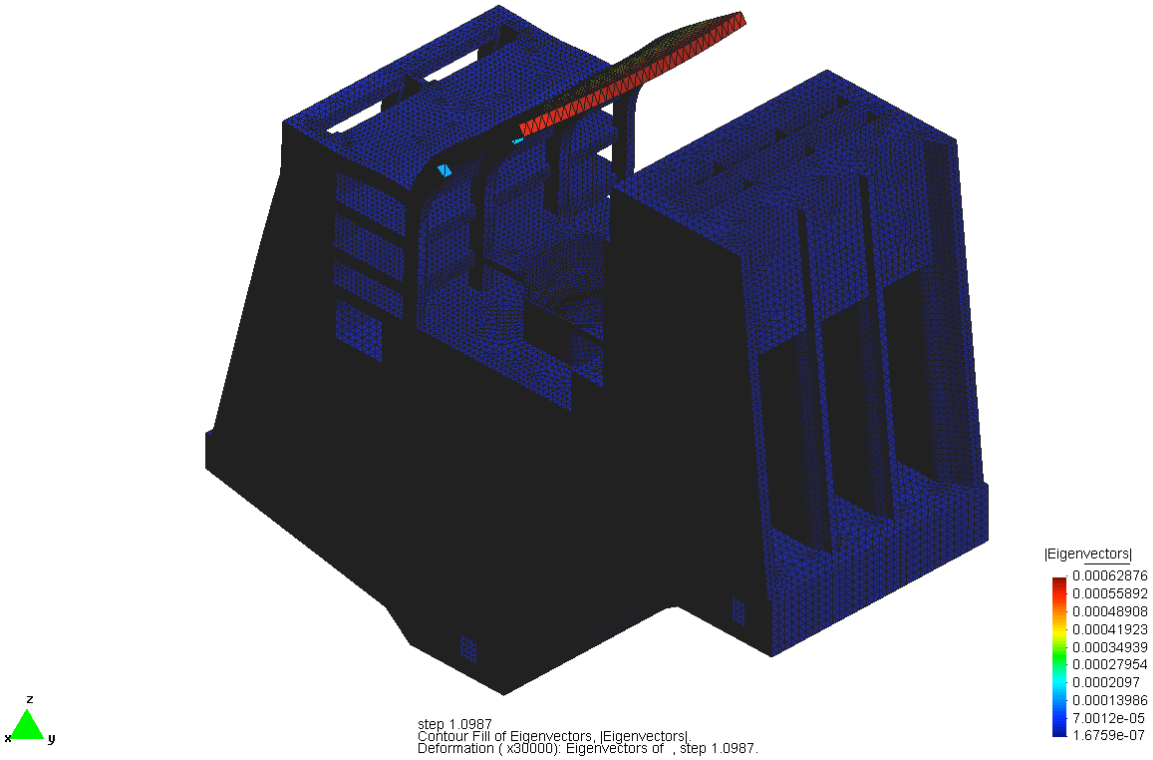

Figura 4.19: Segunda forma modal

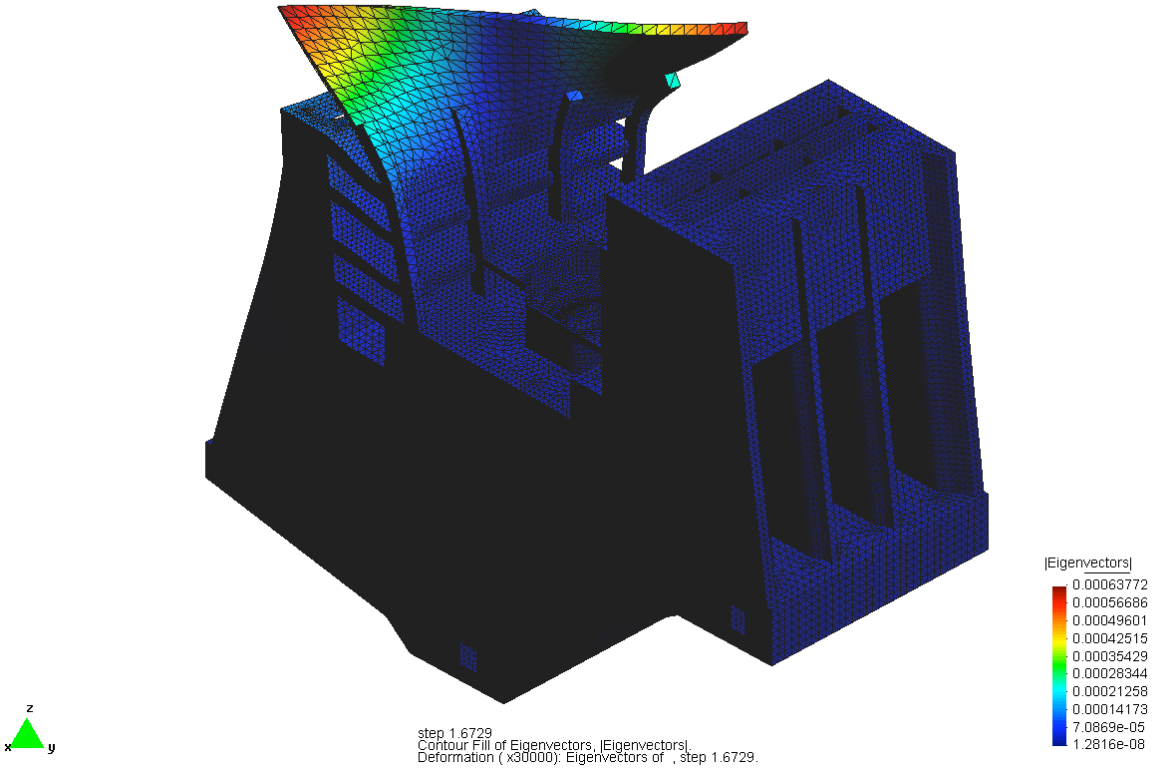

Figura 4.20: Terceira forma modal

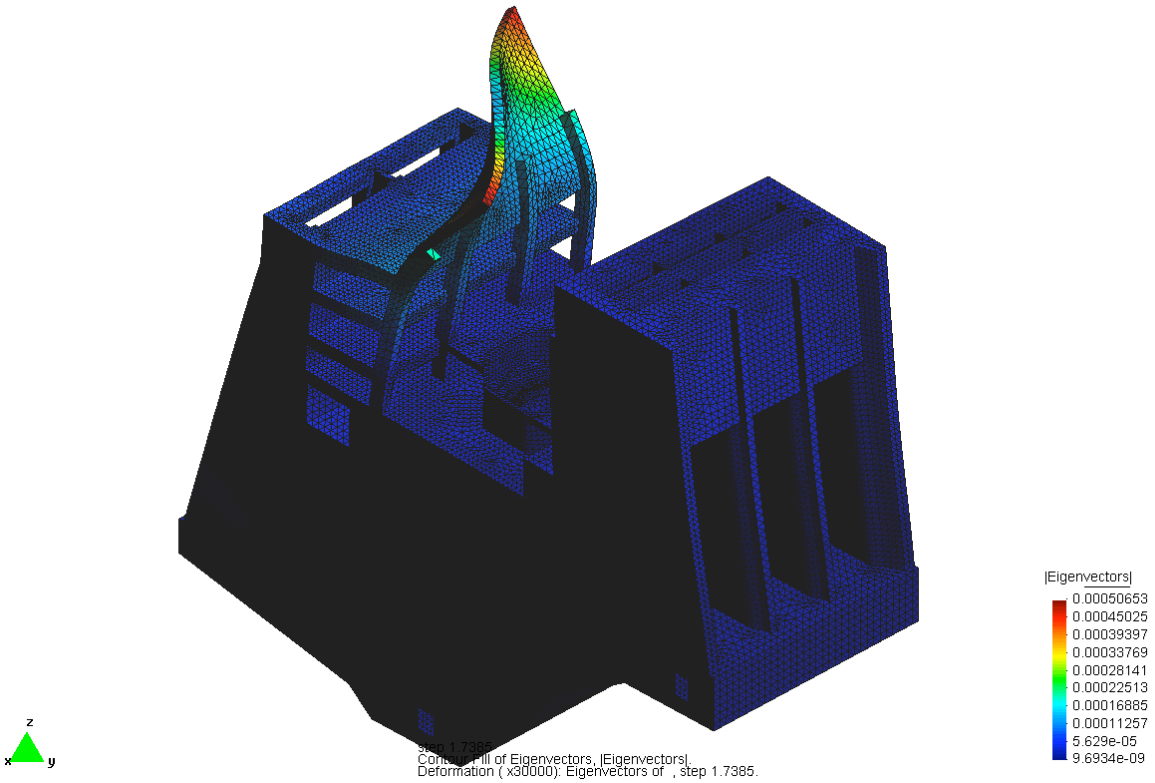

Figura 4.21: Quarta forma modal

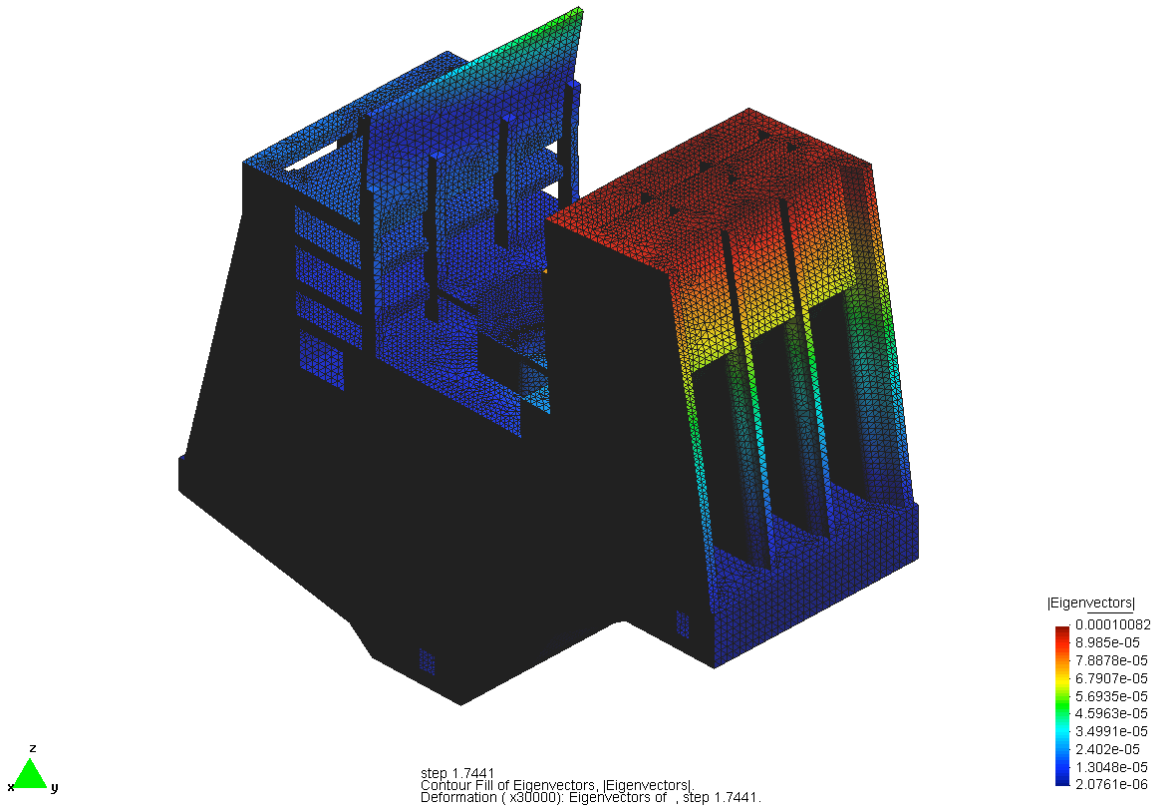

Figura 4.22: Quinta forma modal

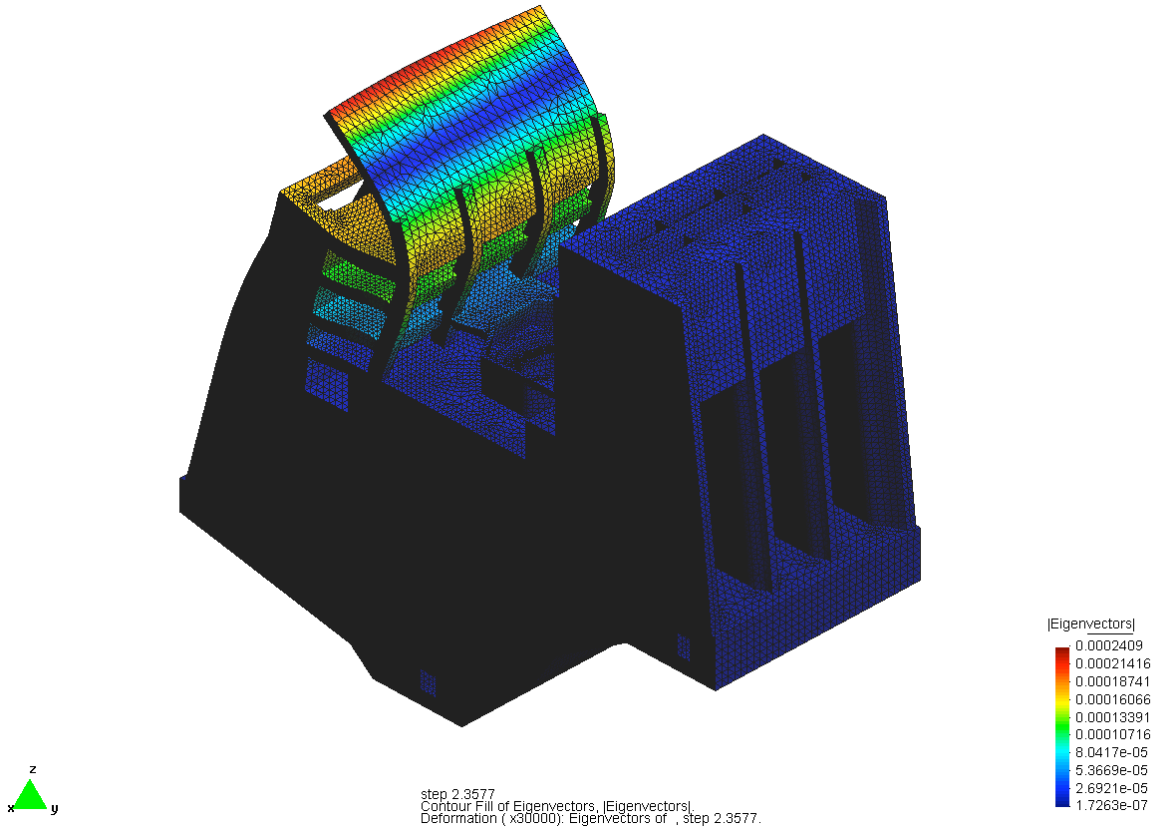

Figura 4.23: Sexta forma modal

As figuras (4.24) e (4.25) mostram um exemplo de particionamento para 64 processos:

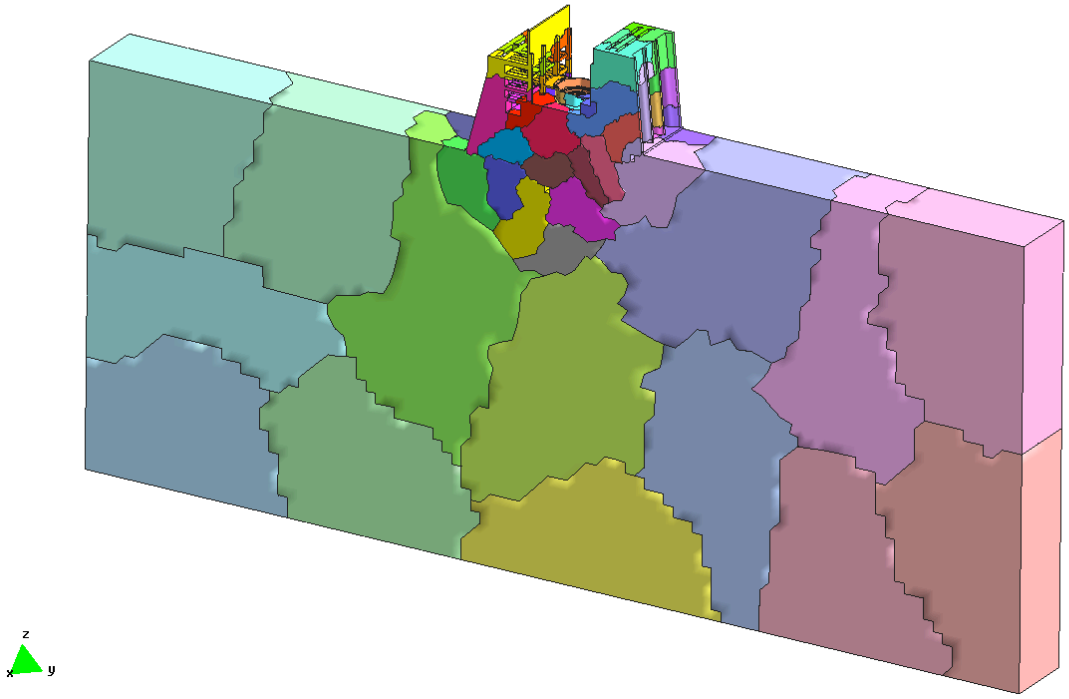

Figura 4.24: Particionamento da malha do caso 2 em 64 processos

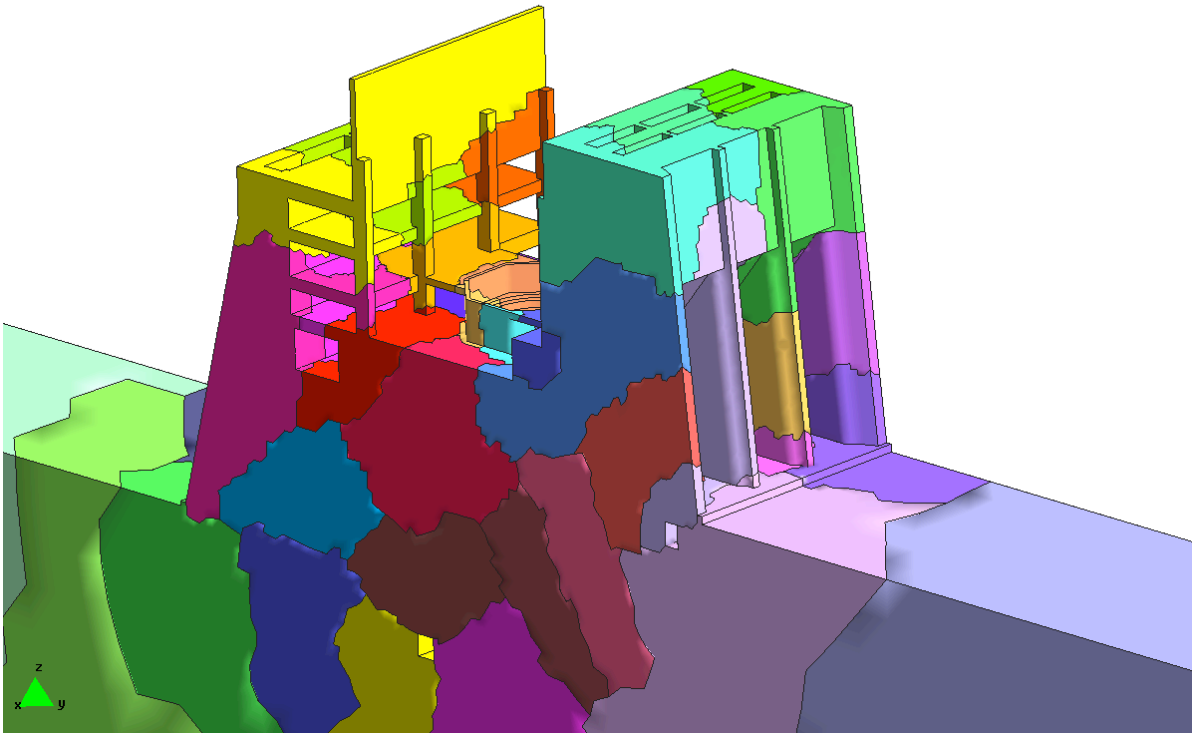

Figura 4.25: Particionamento da malha do caso 2 em 64 processos (casa de força em detalhe)

As curvas de *speedups* obtidas para esse exemplo são mostradas nas figuras de (4.26) a (4.30):

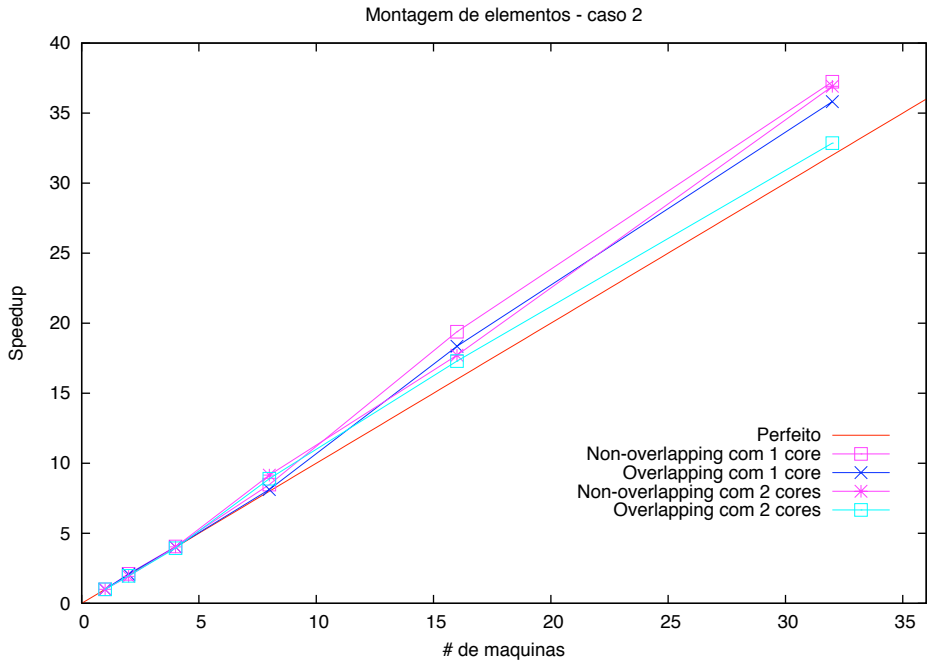

Figura 4.26: *Speedups* para a montagem de elementos do caso 2.

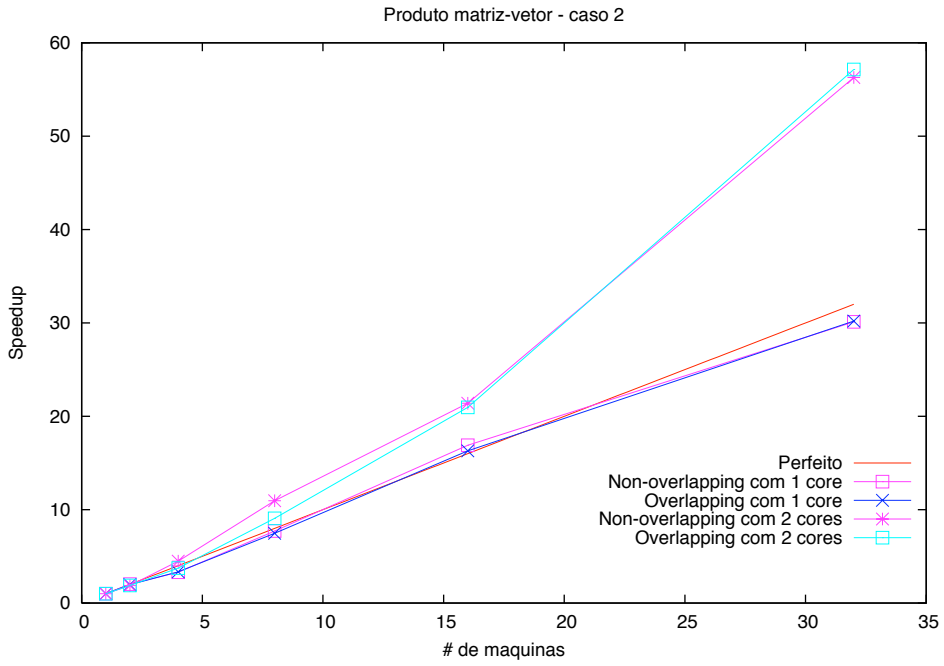

Figura 4.27: *Speedups* para o produto matriz-vetor caso 2.

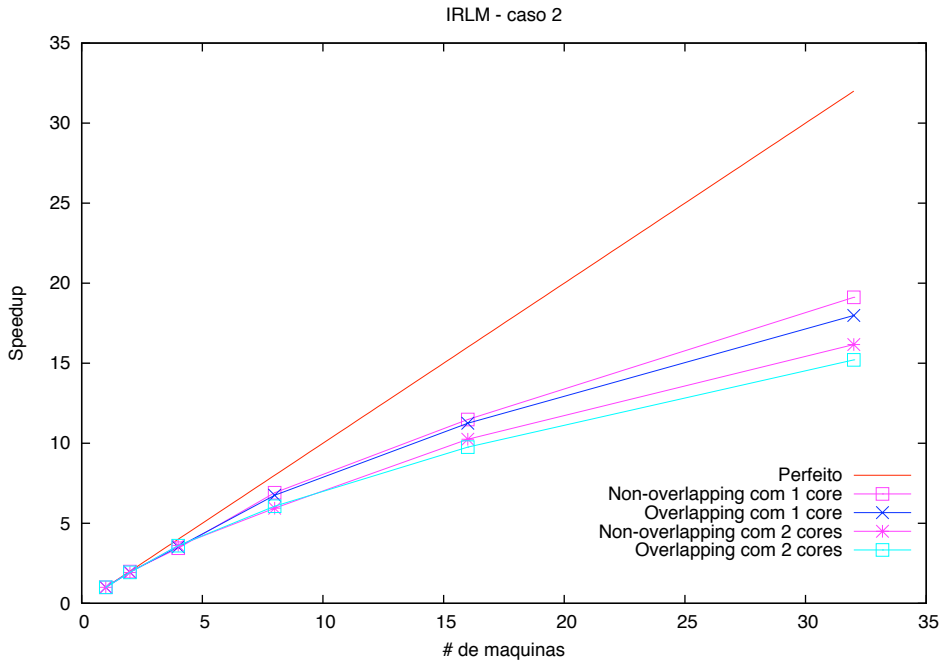

Figura 4.28: *Speedups* para a etapa de solução do problema de autovalor do caso 2.

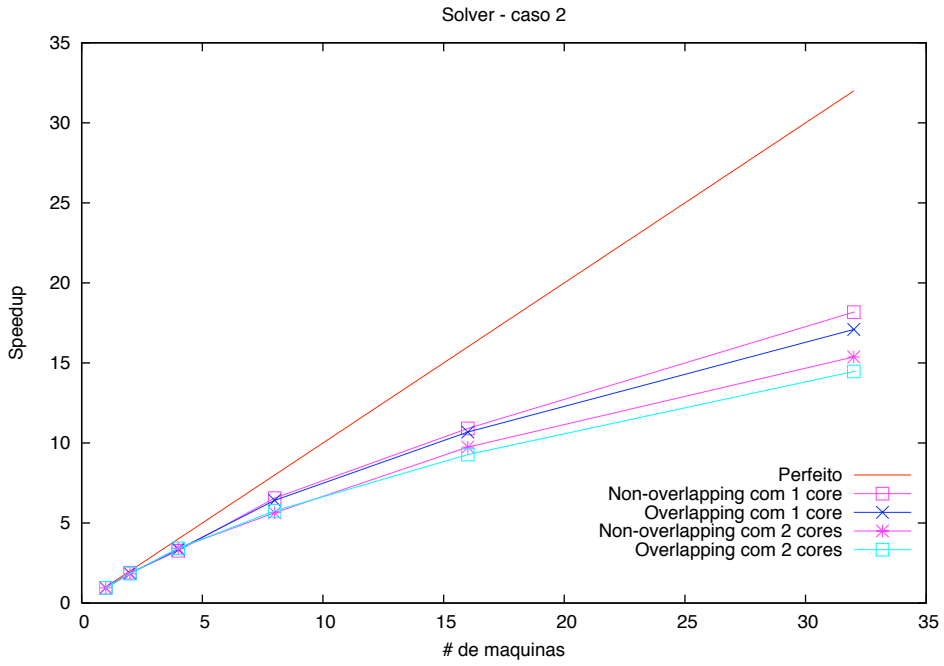

Figura 4.29: *Speedups* para a etapa de solução do sistema de equações (PCG) do caso 2.

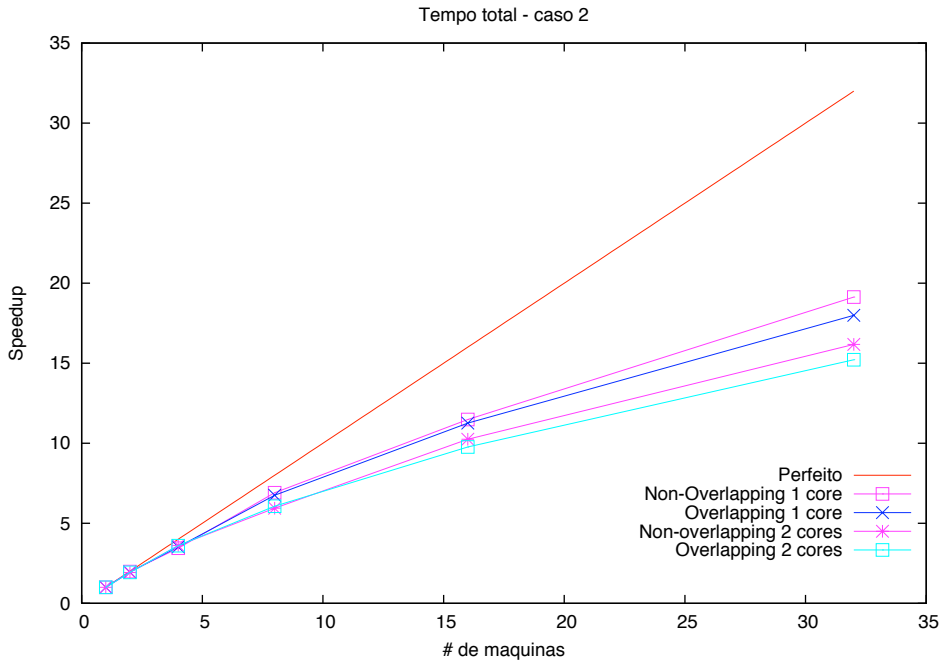

Figura 4.30: *Speedups* para o tempo total de execução do caso 2.

Por não ter redundância na fase de montagem das matrizes globais, o particionamento sem sobreposição (*non-overlapping*) teve melhor desempenho do que o particionamento com sobreposição (*overlapping*). Para o particionamento sem sobreposição, a utilização dos dois núcleos do processador foi vantajosa comparado ao particionamento com sobreposição. O aumento no número de processos teve um grande impacto na melhora da performance do produto matriz-vetor, otimizando o acesso à memória *cache*. Assim como no exemplo anterior, os *speedups* do tempo total e do método de Lanczos com reinício implícito foram governados pelo desempenho do PCG, uma vez que o tempo de solução do sistema de equações compõe o tempo de solução do problema de autovalor que, por sua vez, compõe o tempo total. Como característica intrínseca ao esquema de subdomínio por subdomínio, o número de iterações do PCG e da fase de reinício implícito é o mesmo e independe do número de processos. Os *speedups* máximos obtidos são mostrados na tabela (4.10):

Tabela 4.10: *Speedups* máximos obtidos para o caso 2

Etapa	Máximo *speedup*	Método de particionamento (número de processos)
Montagem de elementos	37.41	Sem sobreposição (1×32)
Produto matriz-vetor	57.14	Com sobreposição (2×32)
PCG	19.13	Sem sobreposição (1×32)
Lanczos com reinício impl.	19.12	Sem sobreposição (1×32)
Total	19.13	Sem sobreposição (1×32)
Tempos totais em (s)	Sequencial	Mínimo paralelo
	16,273.33	850.50

4.3.3 Caso 3: problema de convecção-difusão

Como terceiro exemplo, um problema de autovalor generalizado de um problema de convecção-difusão aproximado com a formulação estabilizada SUPG (Streamline Upwind Petrov-Galerkin) em elementos finitos. Como já foi visto, nesse tipo de problema, os autovalores relevantes na análise de estabilidade do método são os de maior valor absoluto. Para este caso, o algoritmo de reinício implícito realizou 34 iterações, resultando num tempo sequencial total de cerca de 14 horas. As principais características dessa simulação são mostradas na tabela (4.11):

Tabela 4.11: Informações da simulação do caso 3

Número de elementos	1,280,000
Número de equações	1,824,803
Número de iterações do método de Arnoldi com reinício implícito	34
Número de soluções do sistema de equações	3260

A malha de hexaedros desse problema é uma malha estruturada. O algoritmo de particionamento do METIS [69] tem melhor resultados, em termos de minimização

das fronteiras internas, para malhas não-estruturadas. Por isso, o particionamento desse exemplo foi feito numa ferramenta computacional específica para esse fim, podendo ser visto para o caso de 64 partições na figura (4.31):

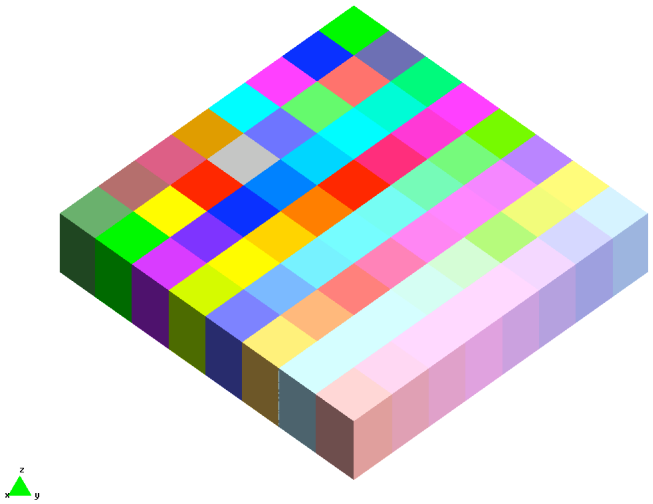

Figura 4.31: Particionamento em 64 subdomínios

As curvas de *speedups* desse exemplo são mostradas nas figuras de (4.32) a (4.36):

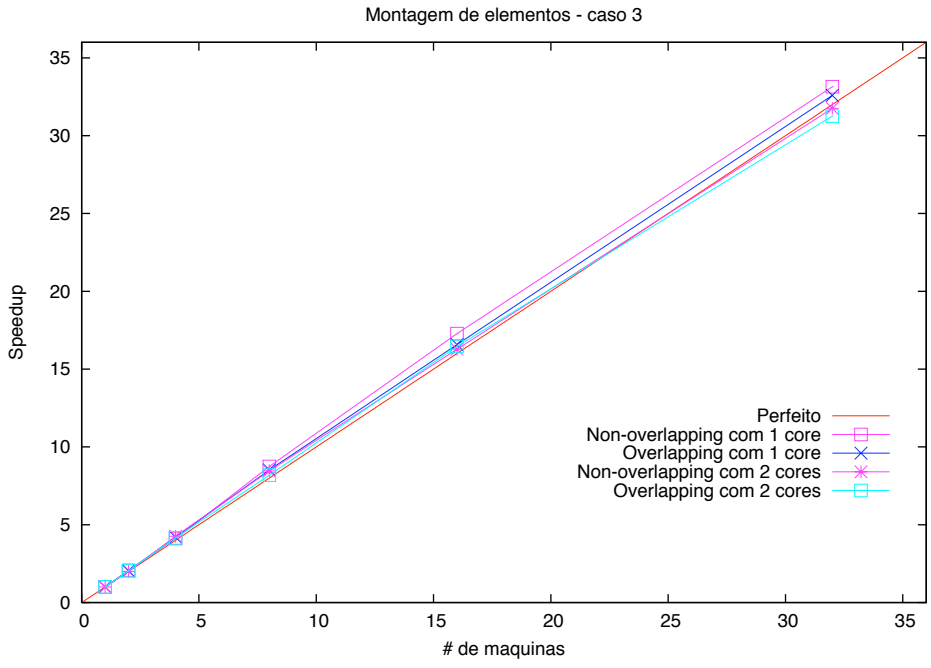

Figura 4.32: *Speedups* para a montagem de elementos do caso 3.

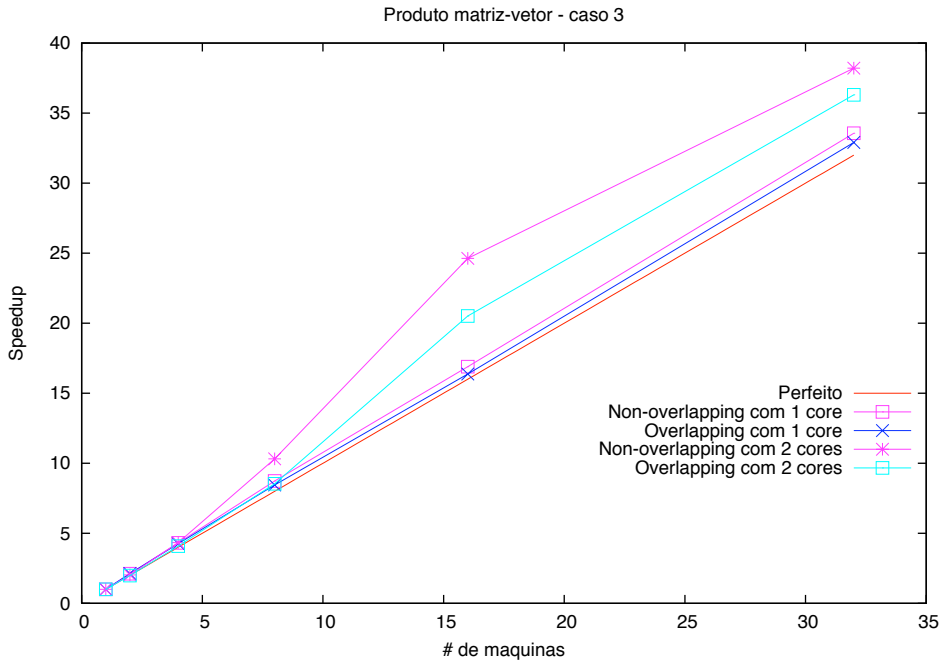

Figura 4.33: *Speedups* para o produto matriz-vetor caso 3.

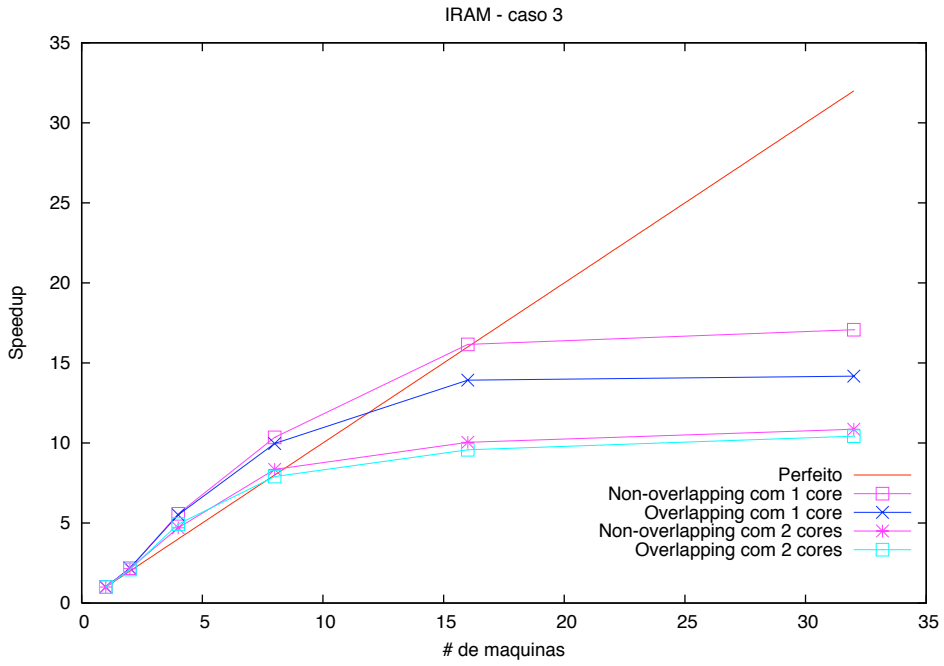

Figura 4.34: *Speedups* para a etapa de solução do problema de autovalor do caso 3.

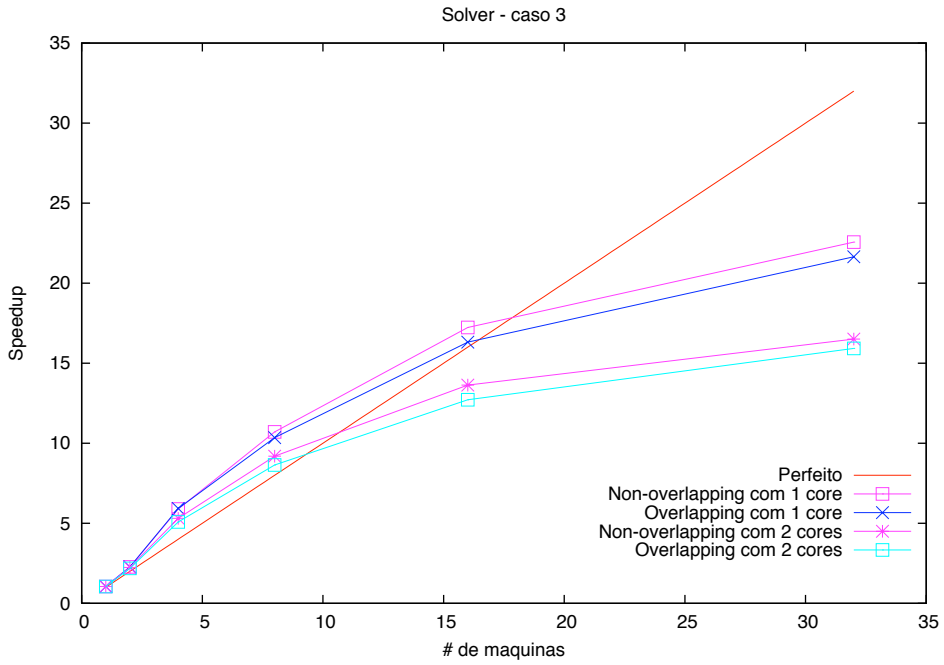

Figura 4.35: *Speedups* para a etapa de solução do sistema de equações (GMRES) do caso 3.

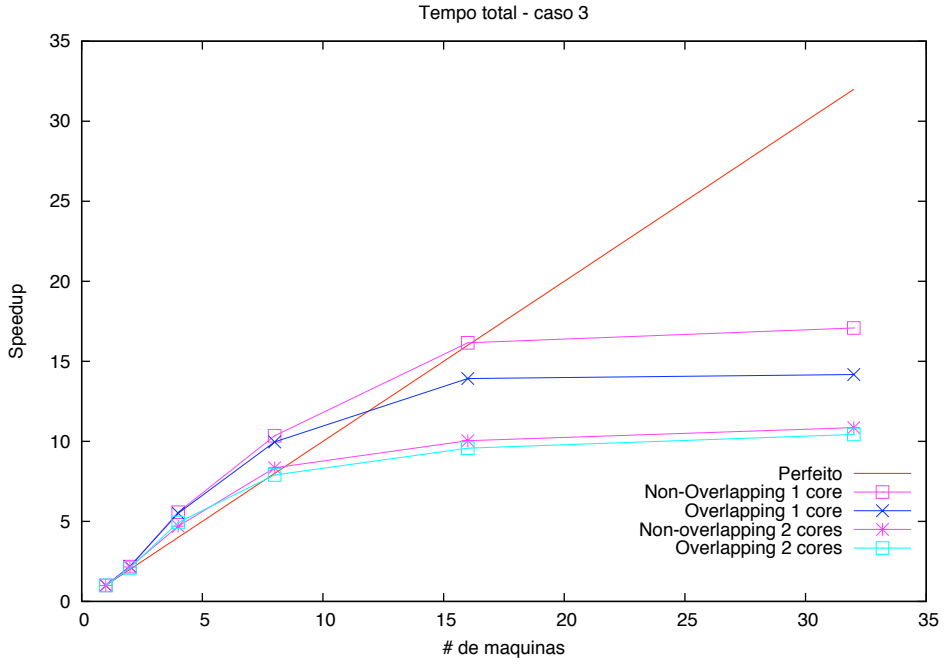

Figura 4.36: *Speedups* para o tempo total de execução do caso 3.

Devido ao gargalo sequencial no cálculo dos *shifts* causado pela redundância da matriz de Hessenberg, a solução do problema de autovalor teve pior eficiência do

que o GMRES, ficando mais evidente nos casos com mais de 16 nós. Isso afetou os *speedups* totais, diminuindo o impacto dos *speedups* do GMRES. Da mesma forma que nos casos simétricos, as convergências do método de Arnoldi com reinício implícito e do GMRES são as mesmas obtidas na execução sequencial e paralela. Os *speedups* máximos para cada etapa da simulação são mostrados na tabela (4.12):

Tabela 4.12: *Speedups* máximos obtidos para o caso 3

Etapa	Máximo *speedup*	Método de particionamento (número de processos)
Montagem de elementos	33.03	Sem sobreposição (1×32)
Produto matriz-vetor	38.21	Sem sobreposição (2×32)
GMRES	21.48	Sem sobreposição (1×32)
Arnoldi com reinício impl.	17.07	Sem sobreposição (1×32)
Total	17.08	Sem sobreposição (1×32)
Tempos totais em (s)	Sequencial	Mínimo paralelo
	50,008.54	2,927.27

Capítulo 5

Conclusões

Apresentou-se neste trabalho, uma implementação em paralelo do método de Arnoldi/Lanczos com reinício implícito para a solução de problemas de autovalor generalizado discretizados pelo método dos elementos finitos. O esquema de reinício implícito juntamente com o procedimento de reortogonalização adotado demonstraram ser bastante eficientes, mantendo a dimensão do subespaço de Krylov pequena se comparada ao número de equações do problema original, porém sem perda de precisão dos autovalores e respectivos autovetores. Em problemas com muitos autovalores repetidos, ou até mesmo próximos, a escolha do tamanho da base é fundamental, pois o tamanho da base determina o número de *shifts* que serão usados no esquema de reinício implícito para melhorar a convergência dos autovalores desejados. Uma má escolha do tamanho da base pode acarretar a convergência da faixa indesejada do espectro de maneira mais rápida que os autovalores desejados conduzindo a resultados errôneos. Para problemas com muitos autovalores repetidos ou próximos, a versão do método de Arnoldi/Lanczos em blocos é mais adequada e pode ser implementada futuramente.

A implementação em paralelo proposta usando o esquema de subdomínio por subdomínio utilizando estruturas de dados comprimidas demonstrou ser bastante adequada na solução de problemas de autovalor pelos métodos de Arnoldi/Lanczos e é a principal contribuição deste trabalho. Vale destacar que na solução em paralelo, a convergência do esquema de reinício implícito independe do número de processos,

sendo esta a principal característica do esquema de subdomínio por subdomínio. O esquema de comunicação adotado, baseado no padrão MPI, está presente apenas nas etapas de solução do problema de autovalor e do sistema de equações e não há comunicação desnecessária entre processos adjacentes. Os exemplos analisados comprovam que, na estratégia de subdomínio por subdomínio, os esquemas de particionamento com e sem sobreposição podem ser implementados com eficiência equivalente, com ligeira vantagem para o participamento sem sobreposição. As estruturas de dados comprimidas nos formatos CSRC e CSRC/CSR comprovaram ser estruturas eficientes no uso de memória, sendo apropriadas para a computação de alto desempenho.

A implementação proposta é voltada para arquiteturas de memória distribuída e foi testada num *cluster* de *pc's* com processadores de núcleo duplo. O uso do padrão MPI com o objetivo de aproveitar os dois núcleos do processador não é a melhor alternativa para esse fim, apesar de apresentar *superspeedups* nas fases de montagem de elemento e na operação de produto matriz-vetor. Na etapa de solução do sistema de equações o uso do MPI para aproveitar a tecnologia de núcleo duplo apresentou *speedups* bastante pobres. O padrão OpenMP é o mais adequado para a utilização da tecnologia *multicore* e como sugestão de trabalho futuro, pode-se implementar um esquema paralelo híbrido, utilizando o MPI para a memória distribuída e o OpenMP para utilizar os diversos núcleos dos processadores num esquema de memória compartilhada.

Referências Bibliográficas

[1] "www.top500.org", .

[2] SAAD, Y., *Numerical Methods for Large Eigenvalue Problems*. 1st ed. John Wiley & Sons Inc.: New York, NY, 1992.

[3] VOEMEL, C., S. TOMOV, O. A. M., A. CHANNING, L.-W.-W., et al., "State-of-the-art eigensolvers for eletronic structure calculations of large scale nano systems", *Journal of Computational Physics*, v. 227, pp. 7113–7124, 2008.

[4] GOLUB, G. H., GREIF, C., "An Arnoldi-type algorithm for computing page rank", *Bit Numerical Mathematics*, v. 46, n. 4, pp. 759–771, 2006.

[5] JACOBI, C. G. J., "Ueber ein leichtes Verfahren, die in der Theorie der Sacularlartorungen vorkommenden Gleichungen numerisch aufzulosen", *Journal fur die reine und angewandte Mathematik*, , n. 51-94, 1846.

[6] BATHE, K.-J., *Finite Elements Procedures*. 1st ed. Prentice-Hall, Inc.: Upper-Saddle River, N. J., 1996.

[7] WILSON, E. L., "Numerical Methods for Dynamics Analysis". Swansea, January 1977.

[8] MARQUES, O. A., *Solução de Problemas de Autovalor Generalizados Através do Método de Iteração por Subespaços em Blocos e do Algoritmo de Lanczos com Ortogonalização Seletiva*, Master's Thesis, Universiade Federal do Rio de Janeiro - COPPE, Rio de Janeiro, RJ, 1986.

[9] DAVIDSON, E. R., "The iterative calculation of a few of yhe lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices", *Journal of Computational Physics*, v. 17, pp. 87–94, 1975.

[10] CROUZEIX, M., PHILIPPE, B., SADKANE, M., "The Davidison Method", *SIAM Journal of Scientific Computing*, v. 15, n. 1, pp. 62–76, 1994.

[11] SLEIJPEN, G. L. G., DER VORST, H. A. V., "A Jacobi-Davidson iteration method for linear eigenvalue problem", *SIAM Journal of Matrix Anal. and Appl.*, v. 17, pp. 401–425, 1996.

[12] ARNOLDI, W. E., "The Principle of minimized iterations in the solution of the matrix eigenvalue problem", *Quart. appl. Math*, v. 9, pp. 17–25, 1951.

[13] LANCZOS, C., "An iteration method for the solution of the eigenvalue problem of linear differential and integral operators", *Journal of Research, Nat Bureau of Standards*, v. 45, pp. 255–282, 1950.

[14] LEHOUCQ, R. B., SORENSEN, D. C., YANG, C., *ARPACK Users' Guide*. 1st ed. SIAM: Philadelphia, PA, 1998.

[15] BAKER, C. G., HETMANIUK, U. L., LEHOUCQ, R. B., et al., "Anasazi software for the numerical solution of scale eigenvalue problems", *ACM trans. Math. Soft.*, v. 5, pp. 1–22, 2008.

[16] HERNÁNDEZ, V., ROMÁN, J. E., TOMÁS, A., et al., *SLEPc Users Manual - Scalable Library for Eigenvalue Problem Computations*, Universidad Politecnica de Valencia, June 2007.

[17] LAWSON, C. L., HANSON, R. J., KINCAID, D., et al., "Basic Linear Algebra Subprograms for FORTRAN usage", *ACM trans. Math. Soft.*, v. 5, pp. 308 – 323, 1979.

[18] DONGARRA, J. J., CROZ, J. D., HAMMARLING, S., et al., "An extended set of FORTRAN Basic Linear Algebra Subprograms", *ACM trans. Math. Soft.*, v. 14, pp. 1 – 17, 1988.

[19] BLACKFORD, L. S., DEMMEL, J., DONGARRA, J., et al., "An Updated Set of Basic Linear Algebra Subprograms (BLAS)", *ACM trans. Math. Soft.*, v. 28, n. 2, pp. 135 – 151, 2002.

[20] KNYAZEV, A. V., SKOROKHODOV, L., "The preconditioned gradient-type iterative methods in a subspace for partial generalized symmetric eigenvalue problem", *SIAM J. Numer. Anal.*, v. 31, 1994.

[21] LUI, S. H., "Some Recents Results on Domain Decomposition Methods for Eigenvalue Problems". 1998.

[22] PARKS, M. L., BOCHEV, P. B., LEHOUCQ, R. B., "Connecting Atomistic-to-Continuum Coupling and Domain Decomposition", *Multiscale Model. Simul*, v. 7, n. 1, pp. 362–380, 2008.

[23] RIBEIRO, F. L. B., FERREIRA, I. A., "Parallel implementation of the finite element method using compressed data structures." *Computational Mechanics*, v. 1, pp. 187–204, December 2007.

[24] FERREIRA, I. A., *SOLUÇÃO EM PARALELO DE UM MODELO TERMO-QUÍMICO-MECÂNICO PARA CONCRETO JOVEM*, Ph.D. Thesis, Universiadde Federal do Rio de Janeiro - COPPE, 2008.

[25] LIU, C. H., LEUNG, D. Y. C., WOO, C. M., "Development of scalable finite element solution to the Navie-Stokes equations", *Computational Mechanics*, v. 32, n. 3, pp. 185–198, 2003.

[26] ICIAM07, *A nested dissection approach to sparse matrix partitioning*, v. 7, 2007.

[27] DANIEL, J., W. B. GRAGG, L. K., STEWART, G. W., "Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization." *Mathematics of Computation*, v. 30, pp. 772–795, 1976.

[28] *LAM/MPI User's Guide*, http://lam-mpi.org/.

[29] WILKINSON, J. H., *The Algebraic Eigenvalue Problem*. 4th ed. Oxford: London, 1972.

[30] PAIGE, C. C., *The Computation of Eigenvalues and Eigenvectors of Very Large Sparse Matrices*, Ph.D. Thesis, University of London, 1971.

[31] EDERLYI, I., "An iterative least squares algorithm suiyable for computing partial eigensystems", *SIAM J. Numer. Anal.*, v. 3, n. 2, 1965.

[32] MANTEUFFEL, T. A., "Adaptive procedure for estimating parameters for the nonsymmetric Tchebychev iteration", *Numer. Math.*, v. 31, pp. 183–208, 1978.

[33] SAAD, Y., "Chebyshev acceleration techniques for solving nonsymmetric eigenvalue problems", *Mathematics of Computation*, v. 42, pp. 567–588, 1984.

[34] ERICSSON, T., RUHE, A., "The spectral transformation Lanczos method for the numerical solution of large sparse generalized symmetric eigenvalue problems." *Mathematics of Computation*, v. 35, pp. 1251–1268, 1980.

[35] NOUR-OMID, B., PARLETT, B. N., ERICSSON, T., et al., "How to implement the spectral transformation", *Mathematics of Computation*, v. 48, n. 178, pp. 663–673, 1987.

[36] SORENSEN, D. C., "Implicit application of polynomial filters in a k-step Arnoldi method", *SIAM Journal on Matrix Analysis and Applications*, v. 13, n. 1, pp. 357–385, 1992.

[37] FRANCIS, J. G. F., "The QR transformation-part 1", *The Computer Journal*, v. 4, pp. 265–271, 1961.

[38] FRANCIS, J. G. F., "The QR transformation-part 2", *The Computer Journal*, v. 4, pp. 332–345, 1962.

[39] LEHOUCQ, R. B., *Analysis and Implementation of an Implicitly Restarted Arnoldi Iteration*, Ph.D. Thesis, Rice University, Houston, TX, 1995.

[40] LEHOUCQ, R. B., "Implicitly Restarted Arnoldi method and Subspace Iteration", *SIAM Journal of Matrix Anal. and Appl.*, v. 23, n. 2, pp. 551–562, 2001.

[41] LEHOUCQ, R. B., *On the convergence of an implicitly restarted Arnoldi method*, Tech. rep., Sandia National Laboratories, Albuquerque, NM, 2005.

[42] MARQUES, O. A., *Utilização do Algoritmo de Lanczos em Blocos na Análise Dinâmica de Estruturas*, Ph.D. Thesis, Universidade Federal do Rio de Janeiro - COPPE, Rio de Janeiro, RJ, 1991.

[43] CULLUM, J., DONATH, W. E., "A Block Lanczos Algorithm for Computing q Algebraically Largest Eigenvalues and a Corresponding Eigenspace of Large, Sparse Symmetric Matrices". pp. 505–509, IEEE Press: New York, NY, 1974.

[44] AINSWORTH, G. O., *Desenvolvimento de Um Algoritmo Baseado no Método de Arnoldi para Solução de Problemas de Autovalor Generalizado*, Master's Thesis, Universiadde Federal do Rio de Janeiro - COPPE, Rio de Janeiro, RJ, 2003.

[45] *P_ARPACK: an efficient portable large scale eigenvalue package for distributed memory parallel architectures*, Lyngby, Denmark, 1996.

[46] GUARRACINO, M. R., PERLA, F., "A parallel modified block Lanczos algorithm for distributed memory architectures", *International Journal of Parallel, Emergent and Distributed Systems*, 1994.

[47] HENRY, G., WATKINS, D., DONGARRA, J., "A parallel implementation of the nonsymmetric QR algorithm for distributed memory architectures", *SIAM J. Sci. Comput*, v. 24, n. 1, pp. 284–311, 1997.

[48] EMAD, N., PETITON, S., EDJLALI, G., "Multiple Explicitly Restarted Arnoldi Method for Solving Large Eigenproblems", *SIAM J. Sci. Comput.*, v. 27, n. 1, pp. 253–277, 2005.

[49] HERNANDEZ, V., ROMAN, J. E., TOMAS, A., "Parallel Arnoldi eigensolvers with enhanced scalability via global communications rearrangement", *Parallel Computing*, v. 33, n. 7-8, pp. 521–540, 2007.

[50] BREZINSKI, C., *Biorthogonality and Its Applications to Numerical Methods*. 1st ed. Marcel Dekker, Inc.: New York, NY, 1992.

[51] BACHMAN, G., NARICI, L., *Functional Analysis*. 2nd ed. Dover Publications: Mineola, NY, 2000.

[52] GOLUB, G. H., LOAN, C. F. V., *Matrix Computations*. 3rd ed. The Jonh Hopkins University Press: Baltimore ML, 1996.

[53] LIMA, E. L., *Álgebra Linear*. 5th ed. *Coleção Matemática Universitária*, IMPA: Rio de Janeiro, RJ, 2001.

[54] PARLETT, B. N., *The Symetric Eigenvalue Problem*. 1st ed. *Prentice-Hall Series in Computational Mathematics*, Prentice-Hall Inc.: Englewood Cliffs, N. J., 1980.

[55] PAIGE, C. C., "Pratical Use of the Symmetric Lanczos Process with Re-Orthogonalization", *BIT*, v. 10, pp. 183–195, 1970.

[56] SAAD, Y., "Variations on Arnoldi's method for computing eigenelements of large unsymetric matrices", *Linear Algebra and Its Applications*, v. 34, pp. 269–295, 1980.

[57] SAAD, Y., *Iterative methods for sparse linear systems*. PWS Publishing Company: Boston, 1996.

[58] SAAD, Y., *Sparsekit: a basic tool kit for sparse matrix computations*, Tech. rep., Computer Science Department, University of Minessota, 1994.

[59] RIBEIRO, F. L. B., GALEÃO, A. C., LANDAU, L., "Edge-based finite element method for shallow water equations", *International Journal for Numerical Methods in Engineering*, v. 36, pp. 659–685, 2001.

[60] LOHNER, R., "Minimization of indirect adressing for edge-based field solves", *Commun. Numerical Methods Eng.*, v. 18, pp. 335–343, 2202.

[61] RIBEIRO, F. L. B., COUTINHO, A. L. G. A., "Comparison Between Element, Edge and Compressed Storage Schemes for Iterative Solutions in Finite Element Analyses", *International Journal for Numerical Methods in Engineering*, v. 63, n. 4, pp. 569–588, 2005.

[62] HESTENES, M., STIEFEL, E., "Methods of Conjugate Gradients for Solving Linear Systems", *Journal of Research of the National Bureau of Standards*, v. 49, n. 6, 1952.

[63] SAAD, Y., SCHULTZ, M., "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems", *SIAM J. Sci. Stat. Comput.*, v. 7, pp. 856–869, 1986.

[64] RIBEIRO, F. L. B., FERREIRA, I. A., FAIRBAIN, E. M. R., "Parallel FE iterative solutions for distributed memory architectures using MPI". In: *XXVI CILAMCE*, 2005.

[65] FERREIRA, I. A., FAIRBAIN, E. M. R., RIBEIRO, F. L. B., et al., "A coupled numerical model for concrete parallelized with MPI". In: *XXVII CILAMCE*, 2006.

[66] *Matrix Market: a repository of test data*, http://math.nist.gov/MatrixMarket/.

[67] HUGHES, T. J. R., *The Finite Element Method - Linear Static and Dynamic Finite Element Analysis*. 2nd ed. Dover Publ., Inc.: Mineola, N. Y., 2000.

[68] CAREY, C. F., "An analysis of stability and oscillations in convection-diffusion computations". In: *Finite element methods for convection dominated flows; Proceedings of the Winter Annual Meeting*, December 1979.

[69] KARYPIS, G., KUMAR, V., "A Fast and Highly Quality Multilevel Scheme for Partitioning Irregular Graphs", *SIAM J. Sci. Comput*, v. 20, n. 1, pp. 359–392, 1999.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Administração
Baixar livros de Agronomia
Baixar livros de Arquitetura
Baixar livros de Artes
Baixar livros de Astronomia
Baixar livros de Biologia Geral
Baixar livros de Ciência da Computação
Baixar livros de Ciência da Informação
Baixar livros de Ciência Política
Baixar livros de Ciências da Saúde
Baixar livros de Comunicação
Baixar livros do Conselho Nacional de Educação - CNE
Baixar livros de Defesa civil
Baixar livros de Direito
Baixar livros de Direitos humanos
Baixar livros de Economia
Baixar livros de Economia Doméstica
Baixar livros de Educação
Baixar livros de Educação - Trânsito
Baixar livros de Educação Física
Baixar livros de Engenharia Aeroespacial
Baixar livros de Farmácia
Baixar livros de Filosofia
Baixar livros de Física
Baixar livros de Geociências
Baixar livros de Geografia
Baixar livros de História
Baixar livros de Línguas

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo