UNIVERSIDADE ESTADUAL DE CAMPINAS

Instituto de Matemática, Estatística e Computação Científica

ALFREDO VITORINO

UM ALGORITMO EFICIENTE PARA A OTIMIZAÇÃO TOPOLÓGICA DE ESTRUTURAS TRIDIMENSIONAIS DE GRANDE PORTE

CAMPINAS
2023

ALFREDO VITORINO

UM ALGORITMO EFICIENTE PARA A OTIMIZAÇÃO TOPOLÓGICA DE ESTRUTURAS TRIDIMENSIONAIS DE GRANDE PORTE

Tese apresentada ao Instituto de Matemática, Estatística e Computação Científica da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Doutor em Matemática Aplicada.

Orientador: FRANCISCO DE ASSIS MAGALHÃES GOMES NETO

Este trabalho corresponde à versão final da Tese defendida pelo aluno Alfredo Vitorino e orientada pelo Prof. Dr. Francisco de Assis Magalhães Gomes Neto.

CAMPINAS
2023

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

V833a

Vitorino, Alfredo, 1995-
Um algoritmo eficiente para a otimização topológica de estruturas tridimensionais de grande porte / Alfredo Vitorino. – Campinas, SP : [s.n.], 2023.

Orientador: Francisco de Assis Magalhães Gomes Neto.
Tese (doutorado) – Universidade Estadual de Campinas, Instituto de Matemática, Estatística e Computação Científica.

1. Otimização topológica. 2. Programação não-linear. 3. Método dos elementos finitos. 4. Métodos multigrid (Análise numérica). I. Gomes Neto, Francisco de Assis Magalhães, 1964-. II. Universidade Estadual de Campinas. Instituto de Matemática, Estatística e Computação Científica. III. Título.

Informações Complementares

Título em outro idioma: An efficient algorithm for the topology optimization of large-scale tridimensional structures
Palavras-chave em inglês:
Topology optimization
Nonlinear programming
Finite element method
Multigrid methods (Numerical analysis)
Área de concentração: Matemática Aplicada
Titulação: Doutor em Matemática Aplicada
Banca examinadora:
Francisco de Assis Magalhães Gomes Neto [Orientador]
Maicon Ribeiro Correa
Roberto Andreani
Thadeu Alves Senne
Luiz Leduino de Salles Neto
Data de defesa: 02-08-2023
Programa de Pós-Graduação: Matemática Aplicada

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0002-5970-3785
- Currículo Lattes do autor: http://lattes.cnpq.br/6318510083480739

Tese de Doutorado defendida em 02 de agosto de 2023 e aprovada

pela banca examinadora composta pelos Profs. Drs.

Prof(a). Dr(a). FRANCISCO DE ASSIS MAGALHÃES GOMES NETO

Prof(a). Dr(a). MAICON RIBEIRO CORREA

Prof(a). Dr(a). ROBERTO ANDREANI

Prof(a). Dr(a). THADEU ALVES SENNE

Prof(a). Dr(a). LUIZ LEDUINO DE SALLES NETO

A Ata da Defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria de Pós-Graduação do Instituto de Matemática, Estatística e Computação Científica.

Dedico o presente trabalho às minhas avós Hollanda (in memoriam) e Elide e aos meus avôs Alfredo (in memoriam) e Leonidas (in memoriam).

Agradecimentos

Agradeço a Deus por me proporcionar uma vida maravilhosa, por me guiar e me dar clareza em todos os momentos desta jornada.

Aos meus pais, Marilene e Valdir, pelo amor incondicional, pelos cuidados, pela educação que me deram, por não medirem esforços para me ajudar ao longo de todos os momentos da minha vida e pelos ensinamentos de quem lutou bravamente para oferecer uma vida digna a mim e aos meus irmãos, Caroline e Valdir, a quem também sou grato pelo afeto, pelos momentos de alegria e pelos conselhos que me tornaram uma pessoa melhor. Aos demais familiares, agradeço o apoio e o estímulo para continuar em busca dos meus sonhos. Em especial, à minha sobrinha Maitê, pelas brincadeiras e risadas.

Ao meu orientador, Prof. Dr. Francisco de Assis Magalhães Gomes Neto (meu querido amigo Chico), pela sua paciência, pelos conhecimentos transmitidos, pelas conversas e discussões, por todas as preciosas contribuições e pelas inúmeras horas carinhosamente dedicadas a este trabalho tão enriquecedor para a minha trajetória acadêmica.

À Profa. Dra. Sandra Augusta Santos e ao Prof. Dr. Thadeu Alves Senne, agradeço pelas valiosas críticas e sugestões dadas durante o exame de qualificação, que contribuíram para a melhoria desta tese. Aos membros da banca de defesa, também agradeço a disponibilidade e as contribuições ofertadas a este trabalho.

Aos demais professores, de todos os níveis de ensino, sou eternamente grato pelos ensinamentos e pelo incentivo, que me permitiram chegar até aqui. Em especial, agradeço à professora Márcia, pela excelente orientação na iniciação científica, aos professores Aurelio e Pulino, por enviarem as cartas de recomendação para o meu ingresso no doutorado, e aos professores Estevão e Kelly, pela ajuda e orientação durante o PED.

Aos funcionários da UNICAMP, em particular do IMECC, pelos excelentes serviços prestados, que viabilizam tudo o que fazemos no instituto.

A todos os meus amigos, agradeço pelo apoio, pelas boas conversas, pelas risadas e pelos momentos de descontração. Em particular, aos meus amigos Gabriel, Lucas, Noemi, Poronga, Vinicius e Vítor, pelas gargalhadas e pelas noites de jogos, ao Sr. Luiz, pelo seu entusiasmo, e ao Jean, pela amizade desde a graduação e por me ensinar como acessar os computadores do laboratório remotamente, o que foi essencial para a obtenção dos resultados desta tese.

À Mayara, reservo um agradecimento singular, pelo amor, carinho, companheirismo, por ser meu conforto e por sempre me inspirar com seu jeito único de ser. A caminhada até aqui se tornou muito mais colorida e divertida ao seu lado.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Resumo

Neste trabalho, estudamos o problema de otimização topológica estrutural tridimensional, cujo propósito é auxiliar a produção de estruturas que tenham a máxima rigidez (ou a mínima flexibilidade), para que sejam capazes de suportar cargas externas sem sofrerem grandes deslocamentos e deformações, mantendo o equilíbrio estático e satisfazendo uma restrição de volume. Utilizamos o método dos elementos finitos para discretizar o domínio no qual a estrutura estará contida e aproximar os deslocamentos da mesma. Por sua vez, a topologia ótima da estrutura é determinada pela melhor distribuição das densidades de material na malha de elementos finitos. O problema na forma discretizada é um problema de otimização não linear, que resolvemos aplicando um algoritmo de programação linear sequencial. Apresentamos resultados computacionais, obtidos com uma implementação feita em Matlab, na qual utilizamos o método *multigrid*, nas versões geométrica e algébrica, como precondicionador do método dos gradientes conjugados na resolução dos sistemas lineares de equilíbrio, reduzindo substancialmente o tempo gasto nessa etapa. Além disso, aplicamos um esquema de multirresolução, cuja ideia é trabalhar com discretizações diferentes em cada etapa da otimização topológica, reduzindo o custo computacional enquanto mantemos uma resolução alta da estrutura. Em contrapartida, a multirresolução pode apresentar regiões com rigidez artificial, formando artefatos indesejados na estrutura, especialmente quando reduzimos o raio de aplicação do filtro de densidades. Para amenizar esse inconveniente e obter soluções mais precisas sem prejudicar demais a eficiência do algoritmo, propomos uma estratégia adaptativa de aumento do grau de aproximação dos deslocamentos, com uma técnica para suprimir variáveis do problema. Também propomos uma nova estratégia de arredondamento das densidades, baseada em informações do vetor gradiente, para obter estruturas compostas por regiões completamente sólidas ou vazias.

Palavras-chave: Otimização Topológica. Programação Não Linear. Método dos Elementos Finitos. Métodos *Multigrid*. Multirresolução.

Abstract

In this work, we study the three-dimensional structural topology optimization problem, whose purpose is to assist the production of structures that have maximum stiffness (or minimum compliance), so that they can bear external loads without suffering large displacements and strains, keeping the static equilibrium and satisfying a volume constraint. We use the finite element method to discretize the domain in which the structure must be contained and to approximate its displacements. The optimal topology of the structure is determined by the best distribution of material densities in the finite element grid. The problem in the discretized form is a nonlinear optimization problem, that we solve applying a sequential linear programming algorithm. We present computational results, obtained with an implementation written in Matlab, in which we use the multigrid method, in geometric and algebraic versions, as preconditioner of the conjugate gradient method for solving the equilibrium linear systems, substantially reducing the time spent on this stage. Furthermore, we apply a multiresolution scheme, whose idea is to work with different discretizations at each stage of the topology optimization, reducing the computational cost while keeping a high resolution for the structure. However, the multiresolution may present regions with artificial stiffness, forming undesired artefacts in the structure, especially when we reduce the application radius of the density filter. To alleviate this inconvenient and to obtain more accurate solutions without undermining too much the algorithm efficiency, we propose an adaptive strategy to increase the degree of approximation of the displacements, with a procedure to suppress variables from the problem. We also propose a new density rounding strategy, based on the gradient vector information, to obtain structures composed by fully solid or void regions.

Keywords: Topology Optimization. Nonlinear Programming. Finite Element Method. Multigrid Methods. Multiresolution.

Sumário

Introdução

A aplicação de métodos matemáticos e computacionais é fundamental para a obtenção de soluções confiáveis e eficazes para problemas reais. Na engenharia estrutural, em particular, um dos avanços mais importantes certamente foi o desenvolvimento de técnicas de otimização capazes de melhorar as estruturas e os seus componentes de maneira rápida e econômica. A criação de uma nova estrutura deve satisfazer certas exigências, pois ela precisa ser segura e, ao mesmo tempo, sua produção deve ser economicamente viável. A elaboração de um modelo de estrutura que cumpra todas as exigências requer grande capacidade, experiência e criatividade dos projetistas ou engenheiros. Além disso, com o avanço tecnológico, exige-se que os projetos sejam desenvolvidos de forma rápida e tenham maior funcionalidade e qualidade. A otimização topológica estrutural é uma metodologia matemática que auxilia na seleção de um formato inicial adequado para uma nova estrutura, facilitando e acelerando o trabalho de desenvolvimento da mesma.

Na formulação mais simples, o problema de otimização topológica estrutural consiste em obter uma estrutura que tenha a maior rigidez (ou, equivalentemente, a menor flexibilidade) possível, de modo que ela consiga suportar a aplicação de cargas externas sem sofrer deslocamentos e deformações excessivos, mantendo-se em equilíbrio estático e satisfazendo uma restrição de volume máximo.

Existem outras técnicas de otimização estrutural que permitem melhorar o desempenho de uma estrutura (vide Haslinger e Mäkinen [28]). Na otimização paramétrica, o objetivo é otimizar um determinado parâmetro (por exemplo, a espessura ou a área da seção transversal), de modo que a estrutura tenha máxima rigidez ou mínima deflexão. Outra estratégia é a otimização de forma, na qual busca-se otimizar o contorno de uma estrutura previamente conhecida. Diferentemente das duas técnicas citadas, na otimização topológica não é necessário conhecer de antemão o formato da estrutura. No início, definimos apenas o domínio no qual a estrutura deve ser construída, as forças externas aplicadas, os apoios responsáveis pela sustentação da estrutura, a quantidade disponível de material e, eventualmente, regiões do domínio onde não pode haver material ou onde deve haver material. A topologia ótima da estrutura é determinada pela melhor distribuição do material no domínio.

O primeiro algoritmo computacional para resolver problemas de otimização topológica foi apresentado por Bendsøe e Kikuchi [5]. Os autores utilizaram um método de homogeneização e propuseram a formulação do problema como um problema de distribuição de material, cuja resolução consiste em decidir se cada um dos pontos do domínio deve conter ou não conter material. Matematicamente, isso pode ser descrito por meio de uma função que assume valor 1 se o ponto contém material ou valor 0 em caso contrário. Todavia, sob o ponto de vista computacional, é extremamente difícil e caro fazer com que cada ponto do domínio seja representado apenas por essas duas possibilidades. Assim,

utiliza-se geralmente uma função contínua que pode assumir valores no intervalo $[0, 1]$ e que representa a densidade de material em cada ponto do domínio. Com o intuito de eliminar densidades intermediárias, Bendsøe [4] sugeriu o uso do modelo SIMP (do inglês *Solid Isotropic Material with Penalization*) no qual a densidade é elevada a um expoente maior que 1. Neste trabalho, fazemos uso dessa abordagem, que passou a ser conhecida como abordagem de densidade.

Além da abordagem citada acima, existem outras formas de tratar o problema de otimização topológica, como as que utilizam métodos de conjuntos de nível [17], análise isogeométrica [55], *moving morphable components* [61], entre outras. O artigo de Sigmund e Maute [47] fornece uma revisão comparativa entre algumas das diferentes abordagens.

Embora, neste texto, trabalhemos apenas com o problema estrutural na forma simples, é possível expandir as estratégias apresentadas para outros tipos de problemas de otimização topológica mais complexos, como os que envolvem restrições adicionais de manufatura [60, 63], permitindo a integração com a impressão 3D, ou o projeto de uma classe especial de estruturas, denominadas mecanismos flexíveis [23, 44, 62], que são desenvolvidas para que haja deslocamentos em algumas direções, permitindo segurar um determinado objeto ou acionar algum dispositivo, por exemplo.

Desde a sua introdução, o problema de otimização topológica vem ganhando destaque na academia e na indústria, com diversas aplicações nas engenharias civil, mecânica, aeroespacial, automobilística, entre outras áreas. O livro de Bendsøe e Sigmund [6] fornece uma compilação de estudos teóricos, métodos computacionais e aplicações da otimização topológica. Outras revisões bibliográficas podem ser consultadas em [15, 18, 47].

No contexto computacional, um dos maiores desafios da área tem sido o desenvolvimento de algoritmos eficientes para a resolução de problemas de grande porte, especialmente os tridimensionais. Em Matlab, Sigmund [45] disponibilizou uma implementação para a resolução de problemas de otimização topológica bidimensionais. Posteriormente, Andreassen *et al.* [2] melhoraram essa implementação, explorando o uso das operações matriciais com o Matlab. Uma extensão para problemas tridimensionais foi apresentada por Liu e Tovar [30], na qual os autores sugeriram o uso de métodos iterativos para a resolução dos sistemas de equilíbrio. Uma versão mais atual e eficiente dessas implementações foi proposta por Ferrari e Sigmund [19]. Ao longo dos anos, surgiram diversos outros trabalhos com estratégias para acelerar o processo de otimização topológica, conforme podemos ver em [35]. Nesta tese, buscamos fornecer um algoritmo robusto e eficiente para a resolução de problemas de otimização topológica de estruturas tridimensionais, combinando algumas técnicas já existentes e propondo novas estratégias que permitam obter soluções de melhor qualidade. Além disso, desejamos que os textos apresentados sejam proveitosos para aqueles que estudam o assunto.

Na abordagem de densidade, a otimização topológica estrutural é combinada com o método dos elementos finitos [3, 14], utilizado para discretizar o domínio no qual a estrutura deve estar contida, o que evita a resolução do problema em um espaço de dimensão infinita. Ao invés de buscarmos o valor da densidade de material para cada um dos infinitos pontos do domínio, buscamos a densidade de cada elemento finito. Neste trabalho, consideramos problemas tridimensionais, dividindo os domínios em pequenas regiões iguais com o formato de um prisma retangular reto, criando uma malha regular. Inicialmente, trabalhamos com elementos lineares e, posteriormente, empregamos também elementos de grau 2 e 3 do tipo Lagrange e do tipo *serendipity*, com o intuito de melhorar

a aproximação dos deslocamentos da estrutura.

No Capítulo 1, estudamos a formulação física do problema de otimização topológica estrutural e, em seguida, fazemos uma introdução ao método dos elementos finitos, descrevendo os detalhes do tipo de elemento utilizado, bem como a caracterização do sistema de equilíbrio estático, e apresentamos o problema na forma discretizada. Além disso, comentamos sobre algumas instabilidades numéricas [48] que podem surgir na resolução do problema e uma possível salvação, que é a aplicação de um filtro [9]. Utilizamos o filtro da média ponderada das densidades, proposto em [12], que possui a vantagem de manter a linearidade na restrição de volume e permitir o cálculo dos gradientes da função objetivo e das restrições do problema de forma analítica.

O problema na forma discretizada é um problema de otimização não linear. Existem diversos métodos para resolução desse tipo de problema (vide [32, 41]). No caso da otimização topológica, por se tratar de um problema com muitas variáveis, os métodos que envolvem segundas derivadas costumam ser evitados. Os mais comumente aplicados à resolução do problema em questão são o método do critério de otimalidade (consulte [6], Seção 1.2), o método das assíntotas móveis [51] e a programação linear sequencial [22, 44]. Neste trabalho, implementamos um algoritmo globalmente convergente de programação linear sequencial, descrito por Gomes e Senne [22]. Em nossa implementação, adaptamos o algoritmo para problemas tridimensionais, fazendo alguns ajustes nos parâmetros, e empregamos um critério de parada baseado nas condições Karush-Kuhn-Tucker do problema, semelhante ao que é proposto em [23]. No Capítulo 2, explicamos o método de programação linear sequencial e apresentamos os detalhes do algoritmo.

Um dos maiores desafios para a resolução eficiente dos problemas de otimização topológica de grande porte é o cálculo da função objetivo, que envolve a resolução de um sistema linear diferente a cada iteração, relacionado à condição de equilíbrio no caso em que a relação entre as deformações e as tensões na estrutura é linear. Sob certas condições, a matriz desse sistema, denominada matriz de rigidez, é simétrica, definida positiva e esparsa. No caso tridimensional, a dimensão dessa matriz cresce demasiadamente quando aumentamos a quantidade de elementos na malha para obter soluções mais precisas. Nesta tese, estendemos o trabalho realizado na dissertação de mestrado [53], na qual verificamos que a resolução dos sistemas lineares é a etapa mais cara do processo de otimização topológica e que, para problemas de grande porte, é mais vantajoso utilizar um método de resolução iterativo, como o método dos gradientes conjugados, ao invés da decomposição de Cholesky. Entretanto, com os precondicionadores frequentemente utilizados, como uma matriz diagonal ou a fatoração incompleta de Cholesky, a resolução dos sistemas lineares continua sendo uma etapa bastante custosa. Isso nos motivou a buscar precondicionadores melhores e, neste trabalho, estudamos os métodos *multigrid* [10, 52], que são empregados para a resolução de sistemas lineares provenientes de problemas com discretizações.

No trabalho de Amir *et al.* [1], o *multigrid* na forma geométrica foi aplicado como precondicionador do método dos gradientes conjugados no contexto da otimização topológica, mostrando resultados promissores. Em um trabalho mais recente, Peetz e Elbanna [40] fizeram uma comparação entre o *multigrid* geométrico e o algébrico como precondicionadores, utilizando o método GMRES para a resolução dos sistemas lineares. Em nossa implementação, utilizamos o *multigrid*, tanto na forma geométrica quanto na forma algébrica, como precondicionador do método dos gradientes conjugados. A versão algébrica que empregamos é diferente daquela utilizada em [40] e é baseada nos trabalhos de Franceschini *et al.* [20] e Magri *et al.* [33], que propõem uma versão do método voltada

para problemas estruturais. Testamos essa versão do *multigrid* algébrico para o caso da otimização topológica, fazendo ajustes de parâmetros. Além disso, a maioria dos trabalhos utilizam o *multigrid* geométrico com malhas compostas por elementos lineares. Dentre os trabalhos que combinam o *multigrid* com o método dos elementos finitos de ordem maior, encontramos o artigo de Sundar *et al.* [50], que compara algumas alternativas nessa linha. A alternativa denominada *h-multigrid* é parecida com a que propomos neste trabalho, na qual utilizamos as funções de forma do método dos elementos finitos para construir a matriz de prolongação, permitindo aplicar o *multigrid* geométrico com elementos do tipo Lagrange e do tipo *serendipity* de grau maior do que 1. No Capítulo 3, apresentamos os métodos iterativos estacionários utilizados como suavizadores do método *multigrid* e, em seguida, descrevemos as ideias do *multigrid* geométrico e do *multigrid* algébrico na versão clássica, bem como na versão utilizada em nossa implementação.

Outras estratégias têm sido aplicadas para reduzir o custo computacional relacionado à resolução dos sistemas lineares do problema de otimização topológica, como a reanálise [43] e os modelos de ordem reduzida [13, 21, 57]. A primeira estratégia consiste em reaproveitar a fatoração de Cholesky da matriz de rigidez em iterações subsequentes do algoritmo de otimização. Já a segunda estratégia consiste em utilizar soluções obtidas em iterações anteriores para construir uma base, denominada base reduzida, e formar um sistema linear de dimensão menor, com o qual é possível obter uma aproximação para a solução do sistema original da iteração atual. No Capítulo 4, descrevemos com mais detalhes a ideia dos modelos de ordem reduzida e apresentamos duas possibilidades para a construção da base reduzida, uma delas utilizando o processo de ortonormalização de Gram-Schmidt e a outra utilizando a decomposição em valores singulares (SVD).

A obtenção de estruturas com melhor qualidade e mais detalhes requer que utilizemos malhas com um número grande de elementos finitos. Além de aumentar o tempo de resolução, isso provoca um consumo elevado de memória computacional. Esse contratempo se mantém mesmo quando utilizamos estratégias como o *multigrid* ou a redução de ordem, citadas acima. Ademais, essas estratégias reduzem apenas o tempo computacional relacionado ao cálculo da função objetivo, muito embora outras etapas do processo de otimização topológica também possam ser custosas. Com o intuito de tornar o algoritmo mais eficiente, estudamos a multirresolução, conforme proposta por Nguyen *et al.* [37, 38]. Nessa técnica, utilizamos malhas com refinamentos distintos para as etapas do processo de otimização topológica: uma malha mais grossa para a aproximação dos deslocamentos nodais (resolução dos sistemas de equilíbrio), uma malha intermediária para a resolução do problema de otimização e uma malha mais fina na qual é definida a distribuição das densidades de material. Com isso, a resolução final da estrutura é alta, enquanto o custo computacional das etapas mais caras do algoritmo é reduzido. Dentre outros trabalhos que utilizam a multirresolução em diferentes aplicações e abordagens da otimização topológica, citamos os artigos [24, 25, 39, 59]. Neste trabalho, combinamos a multirresolução com a programação linear sequencial e o *multigrid*, obtendo um algoritmo bastante eficaz para a resolução de problemas tridimensionais. No Capítulo 5, explicamos os detalhes dessa estratégia de separação das malhas.

Um inconveniente da multirresolução é a baixa precisão na aproximação dos deslocamentos da estrutura (feita na malha mais grossa), que pode causar imprecisões na distribuição das densidades de material (feita na malha mais fina). Essas imprecisões não são percebidas pelo método de otimização e acabam gerando soluções com rigidez artificial, que podem conter os chamados artefatos (vide Gupta *et al.* [26]), que são regiões

com material distribuído de maneira aleatória, gerando barras incompletas ou partes "flutuando" na estrutura. O surgimento dos artefatos, embora ainda sem essa nomenclatura, também foi relatado nos trabalhos de Groen *et al.* [24], Gupta *et al.* [27] e Nguyen *et al.* [36]. Os resultados desses trabalhos sugerem que o uso de um filtro de densidades e o aumento no grau de aproximação dos deslocamentos podem amenizar o surgimento dos artefatos. Em [24], os autores utilizam o método das células finitas e em [36] utilizam a versão-p do método dos elementos finitos, ambos com funções de forma hierárquicas para melhorar a precisão na aproximação dos deslocamentos. Em nossos resultados, também observamos o surgimento de artefatos nas soluções obtidas com a multirresolução, especialmente quando reduzimos o raio de aplicação do filtro de densidades, com o intuito de obter estruturas mais detalhadas. Além disso, testamos o uso de elementos finitos clássicos, do tipo Lagrange e do tipo *serendipity*, aumentando o grau dos polinômios que compõem as funções de forma a fim de melhorar a aproximação dos deslocamentos e, consequentemente, a qualidade das soluções.

Entretanto, aumentar o grau dos elementos finitos causa uma elevação no tempo de resolução dos sistemas lineares, uma vez que o número de nós da malha aumenta. Para amenizar esse infortúnio, Gupta *et al.* [25] propõem uma estratégia adaptativa que modifica o grau dos elementos finitos e a quantidade de variáveis de projeto em cada elemento no decorrer do processo de otimização. A ideia é aumentar esses valores nas regiões do domínio que necessitam de uma precisão maior, como as que podem conter artefatos, e reduzir os valores em regiões cuja precisão pode ser menor. Os resultados obtidos com essa estratégia são promissores, embora os problemas testados sejam apenas bidimensionais. Em nosso trabalho, propomos uma outra estratégia adaptativa de aumento do grau dos elementos finitos, que consiste em (i) resolver o problema com um grau baixo; (ii) suprimir algumas variáveis (mantendo os valores fixados) em regiões do domínio cujo formato da estrutura já esteja bem definido na solução obtida; e (iii) resolver o problema novamente aumentando o grau de aproximação dos deslocamentos. Esse processo de fixação de variáveis e resolução de um problema com uma aproximação de maior grau para os elementos é repetido até que a solução encontrada não contenha artefatos e seja adequada. No Capítulo 6, descrevemos os detalhes dessa estratégia.

Também no Capítulo 6, propomos uma nova estratégia de arredondamento de densidades, com o objetivo de reduzir as densidades intermediárias na solução, obtendo estruturas formadas por regiões completamente sólidas ou vazias. O uso do modelo SIMP, junto com a restrição de volume, faz com que a solução tenha poucas densidades intermediárias. Contudo, a aplicação do filtro pode fazer com que essas densidades apareçam com mais frequência. Ademais, as densidades intermediárias podem ter influência sobre o valor da flexibilidade. O desenvolvimento de técnicas para obter soluções binárias é, portanto, um assunto relevante na otimização topológica. Os trabalhos que utilizam estratégias de arredondamento incluem os artigos de Wang *et al.* [54] e Xu *et al.* [58]. Nossa estratégia é baseada em informações do vetor gradiente do Lagrangiano do problema e permite obter soluções com melhor qualidade, que estão próximas de minimizadores locais.

Para finalizar, no Capítulo 7, apresentamos os experimentos numéricos realizados ao longo do desenvolvimento deste trabalho com uma implementação feita em Matlab, bem como uma análise detalhada dos resultados computacionais obtidos, seguindo algumas das recomendações feitas em [46]. Testamos a resolução do problema de otimização topológica estrutural para diversos tipos de estruturas tridimensionais e analisamos a eficácia das estratégias mencionadas acima para a resolução de problemas de grande porte.

Capítulo 1

Otimização Topológica Estrutural

A *otimização topológica estrutural* é uma metodologia matemática desenvolvida com o propósito de auxiliar a produção de estruturas que tenham a máxima *rigidez* (ou, equivalentemente, a mínima *flexibilidade*), para que sejam capazes de suportar cargas externas sem sofrerem grandes deslocamentos e deformações, mantendo o equilíbrio estático e satisfazendo certas restrições, como ter um volume máximo.

Algumas técnicas permitem otimizar o contorno (*otimização de forma*) ou algum parâmetro (*otimização paramétrica*) de uma estrutura cujo formato inicial seja previamente conhecido (vide Haslinger e Mäkinen [28]), mas encontrar um novo modelo de estrutura requer tempo, experiência e criatividade do projetista ou engenheiro. A otimização topológica é um conjunto de ferramentas que facilita a seleção de um formato inicial adequado. De início, conhecemos o domínio $\Omega \subset \mathbb{R}^3$ no qual a estrutura deve ficar contida, as cargas externas aplicadas, os apoios responsáveis pela sustentação da estrutura e a quantidade máxima de material que poderá ser utilizada. A topologia ótima da estrutura é determinada pela melhor distribuição do material no domínio Ω.

O problema de distribuição de material foi introduzido de forma computacional por Bendsøe e Kikuchi [5] no final da década de 1980. Desde então, a otimização topológica vem ganhando destaque na academia e na indústria, com diversas aplicações nas engenharias e outros campos. A literatura da área é bastante ampla e revisões bibliográficas podem ser consultadas no livro de Bendsøe e Sigmund [6] e nos artigos [15, 18, 47].

1.1 O problema de distribuição de material

Na resolução do problema de otimização topológica, conhecemos o espaço de projeto, isto é, o domínio $\Omega \subset \mathbb{R}^3$ no qual a estrutura deve ficar contida, e queremos determinar em quais pontos desse domínio haverá material. Dito de outra forma, queremos saber se cada ponto $\mathrm{x} \in \Omega$ será vazio ou sólido. Dessa maneira, a topologia da estrutura pode ser representada por uma função $\mathcal{X} : \Omega \longrightarrow \{0, 1\}$, dada por

$$\mathcal{X}(\mathrm{x}) = \begin{cases} 1, & \text{se o ponto } \mathrm{x} \text{ contém material,} \\ 0, & \text{se o ponto } \mathrm{x} \text{ não contém material.} \end{cases}$$

Podemos imaginar que cada ponto do domínio será colorido de branco (sem material) ou preto (com material), permitindo que se tenha uma ilustração do formato da estrutura.

O uso da função discreta $\mathcal{X}$ torna mal condicionado o problema numérico a ser resolvido, devido à mudança brusca no valor das variáveis de decisão, além de tornar o problema mais difícil e custoso computacionalmente. Uma alternativa é permitir que as variáveis assumam valores intermediários entre 0 e 1, substituindo a função $\mathcal{X}$ por uma função contínua $\rho : \Omega \longrightarrow [0,1]$ que representa a *densidade de material* em cada ponto $\mathrm{x} \in \Omega$. Assim, além do branco e do preto, os pontos podem assumir tons de cinza, que representam as densidades entre 0 e 1, conforme ilustra a Figura 1.1.

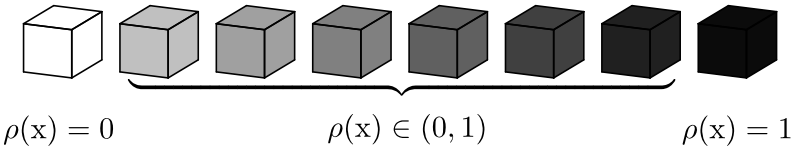


Figura 1.1: Representação das densidades de material em pontos do domínio.

1.1.1 Modelo de densidades (SIMP)

A versão relaxada do problema, utilizando a função ρ para descrever a distribuição de material, é mais simples de ser resolvida. Porém, a solução com densidades intermediárias pode não ter interpretação física e é impossível produzir uma estrutura utilizando materiais com diferentes densidades em cada um dos pontos.

Para amenizar esse inconveniente, aplicamos o *modelo de densidades*, ou *Solid Isotropic Material with Penalization* (SIMP) (Bendsøe [4]), no qual, ao invés de ρ, utiliza-se a função ρ^p para controlar a distribuição de material, sendo $p \geq 1$ um parâmetro de penalização responsável pela diminuição da ocorrência de densidades intermediárias.

De acordo com Rietz [42], na prática devemos ser cautelosos com a penalização excessiva das densidades, ou seja, com o emprego de um valor do expoente p muito alto, pois, conforme o parâmetro aumenta, as densidades no intervalo $(0,1)$ ficam cada vez mais próximas de zero, como ilustra a Figura 1.2 para alguns valores de p. Assim, à medida que p aumenta, a variação entre as densidades se torna cada vez mais abrupta, o que pode trazer de volta o mau condicionamento do problema.

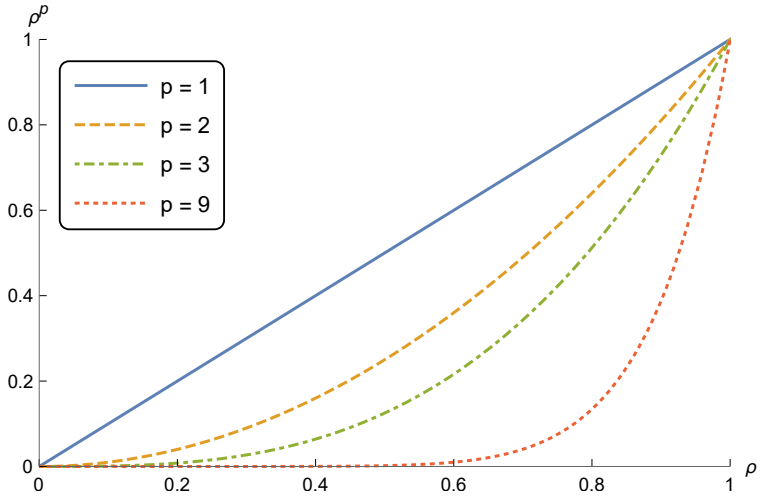


Figura 1.2: Efeito do aumento do parâmetro p do modelo SIMP sobre as densidades.

Sob certas hipóteses, pode-se mostrar que é possível encontrar um valor finito do parâmetro de penalização tal que o valor ótimo de cada densidade seja 0 ou 1 (veja Martínez [34] e Rietz [42]), ou seja, existe um valor finito de p tal que a solução do problema de otimização topológica é discreta. Na prática, costuma-se utilizar $p = 3$, ou então uma estratégia de aumentar o valor de $p = 1$ até $p = 3$ gradativamente, e o modelo de densidades apresenta bons resultados.

O modelo SIMP, junto com a restrição de volume máximo da estrutura, faz com que a solução obtida tenha poucas densidades intermediárias. Entretanto, o desenvolvimento de novas técnicas para obter soluções completamente binárias ainda é um assunto de bastante interesse na otimização topológica.

1.2 Mínima flexibilidade

A distribuição de material no espaço de projeto deve ser ótima, no sentido de que a estrutura obtida seja a mais rígida possível. A rigidez está inversamente relacionada ao conceito de *flexibilidade*, de tal forma que maximizar a rigidez de uma estrutura equivale a minimizar a sua flexibilidade. Assim, o problema básico de otimização topológica estrutural pode ser formulado matematicamente com o objetivo de minimizar a flexibilidade da estrutura, garantindo que ela esteja em equilíbrio estático e satisfaça uma restrição de volume.

Consideremos a estrutura como um corpo elástico, em um domínio $\Omega \subset \mathbb{R}^3$, que deve ser mantido fixo em uma região $\Gamma_s \subset \Omega$ e que está submetido à aplicação de uma carga externa q na região $\Gamma_q \subset \Omega$, conforme ilustra a Figura 1.3.

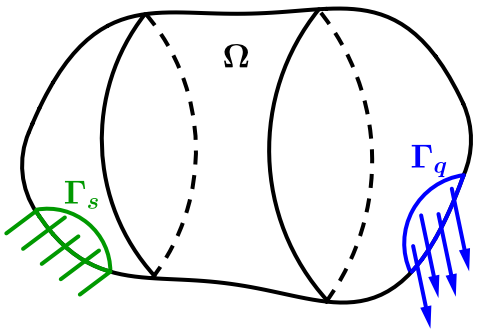


Figura 1.3: Exemplo de domínio com apoios e a aplicação de cargas.

Estando fixado na região Γ_s, o corpo deve se manter em equilíbrio após receber a aplicação da carga externa. Contudo, seus pontos são deslocados, de modo que o carregamento externo gera um campo de deslocamentos, dado por vetores

$$\mathrm{u} = \begin{bmatrix} u(\mathrm{x}) & v(\mathrm{x}) & w(\mathrm{x}) \end{bmatrix}^T, \tag{1.1}$$

em que as funções $u, v, w : \Omega \longrightarrow \mathbb{R}$ representam os deslocamentos do ponto $\mathrm{x} \in \Omega$ nas direções coordenadas x, y e z, respectivamente. Cada uma dessas funções deve pertencer a um conjunto de *funções admissíveis*, que devem ser diferenciáveis e quadrado-integráveis em Ω e devem se anular nos pontos da região Γ_s, onde estão os apoios que impedem alguns deslocamentos. Denotamos por V o conjunto dos campos de deslocamentos admissíveis, formados por vetores na forma (1.1), com as três funções u, v e w admissíveis.

Além dos deslocamentos, o corpo também sofre deformações ao receber o carregamento externo. Supomos que o corpo é composto por um material *isotrópico*, isto é, que possui as mesmas propriedades independentemente da direção, e que ele está em regime elástico linear. Neste caso, o *trabalho interno* associado às deformações está relacionado com a forma bilinear $a : \mathrm{V} \times \mathrm{V} \longrightarrow \mathbb{R}$ dada por

$$a(\mathrm{u}, \mathrm{v}) = \int_\Omega \epsilon(\mathrm{u})^T \mathrm{D} \epsilon(\mathrm{v}) \, d\Omega, \quad \mathrm{u}, \mathrm{v} \in \mathrm{V},$$

em que $\epsilon(\mathrm{u})$ e $\epsilon(\mathrm{v})$ são os vetores de deformações associados aos campos de deslocamentos u e v, respectivamente, e a matriz D é um tensor de elasticidade simétrico, que contém as propriedades elásticas do material.

O *trabalho externo* realizado pela carga q está relacionado com a forma linear $l : \mathrm{V} \longrightarrow \mathbb{R}$ dada por

$$l(\mathrm{v}) = \int_{\Gamma_q} \mathrm{v}^T q \, d\Gamma_q, \quad \mathrm{v} \in \mathrm{V}.$$

Para que a estrutura esteja em equilíbrio, o trabalho interno deve se igualar ao trabalho externo para qualquer campo de deslocamentos admissível, de modo que

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

sendo $\bar{\mathrm{u}} \in \mathrm{V}$ o campo de deslocamentos gerado pela carga externa quando o corpo fica em equilíbrio estático.

Definimos a *flexibilidade média* da estrutura como sendo o trabalho externo realizado pela carga para o campo de deslocamentos $\bar{\mathrm{u}}$, ou seja,

$$l(\bar{\mathrm{u}}) = \int_{\Gamma_q} \bar{\mathrm{u}}^T q \, d\Gamma_q.$$

O objetivo do problema é minimizar essa função. Além disso, para construir a estrutura dispomos de uma certa quantidade máxima de material. Assim, sendo V_{max} o volume máximo de material que poderá ser utilizado, devemos satisfazer a restrição

$$\int_\Omega \rho(\mathrm{x}) \, d\Omega \leq V_{max}, \tag{1.2}$$

em que ρ é a função que representa as densidades de material em cada ponto x de Ω. As densidades também devem satisfazer as restrições $0 \leq \rho(\mathrm{x}) \leq 1$, $\forall \mathrm{x} \in \Omega$.

Reunindo os elementos apresentados acima, formulamos o problema de otimização topológica estrutural com o objetivo de encontrar as densidades de material nos pontos do domínio que minimizem a flexibilidade média, garantindo o equilíbrio da estrutura, sujeito à restrição de volume e com as densidades no intervalo $[0, 1]$:

$$\begin{aligned} \min_{\rho} \quad & l(\bar{\mathrm{u}}) \\ \text{s. a} \quad & a(\bar{\mathrm{u}}, \mathrm{v}) = l(\mathrm{v}), \quad \forall \mathrm{v} \in \mathrm{V}, \quad \bar{u} \in \mathrm{V}, \\ & \int_\Omega \rho(\mathrm{x}) \, d\Omega \leq V_{max}, \\ & 0 \leq \rho(\mathrm{x}) \leq 1, \quad \forall \mathrm{x} \in \Omega. \end{aligned}$$

O problema nessa forma possui infinitas variáveis e é extremamente complicado de ser resolvido. Na prática, trabalhamos com ele em sua forma discretizada, aplicando o método dos elementos finitos, que descrevemos na próxima seção.

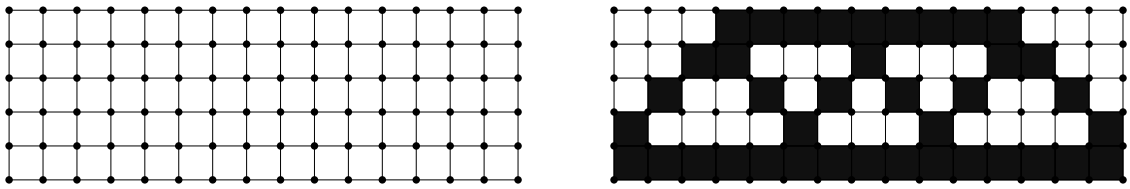
1.3 Tratamento numérico do problema pelo método dos elementos finitos

Na seção anterior, a formulação do problema de otimização topológica estrutural foi considerada em um domínio $\Omega \subset \mathbb{R}^3$, que é um meio contínuo. Na prática, a resolução do problema nessa forma é complicada, pois envolve a resolução de um problema de otimização não linear em um espaço de dimensão infinita. Para resolvê-lo numericamente, aplicamos o método dos elementos finitos, discretizando o domínio Ω.

O método dos elementos finitos é um procedimento numérico bastante empregado em diversas áreas. Particularmente na análise de estruturas, ele é utilizado para se obter aproximações para os deslocamentos, deformações ou tensões em um corpo sujeito à aplicação de cargas externas (consulte, por exemplo, Assan [3] e Cook *et al.* [14]). A obtenção desses dados para todos os pontos do domínio no qual o corpo está contido é um problema difícil. A estratégia do método consiste em discretizar esse domínio, dividindo-o em uma quantidade finita de pequenas regiões, denominadas *elementos finitos*, e definir problemas mais simples em cada elemento.

Após ser dividido em pequenas regiões, o domínio passa a ser chamado de *malha* de elementos finitos e os pontos que conectam as linhas da malha são chamados de *nós*. A quantidade de divisões feitas no domínio, isto é, o número de elementos e de nós na malha, depende da precisão exigida para a solução aproximada do problema. Se queremos uma solução mais precisa e detalhada, é necessário utilizar uma malha mais refinada, ou seja, com uma grande quantidade de elementos, entretanto o custo computacional para obter essa solução será maior.

Com a malha de elementos finitos, ao invés de tentarmos encontrar as densidades de material para cada um dos infinitos pontos do domínio Ω, vamos obter as densidades em cada elemento da malha, supondo que em todos os pontos do elemento a densidade seja uniforme. A Figura 1.4 ilustra uma malha bidimensional formada por elementos quadrados e uma possível distribuição do material nessa malha.



(a) Malha inicial. (b) Malha com o material distribuído.

Figura 1.4: Exemplo de uma malha bidimensional de elementos finitos.

Neste trabalho, consideramos que o domínio Ω possui a forma de um prisma retangular reto e utilizamos elementos que também têm esse formato, chamados de elementos prismáticos retangulares. Cada nó do elemento possui três *graus de liberdade*, que representam os deslocamentos desse nó nas direções x, y e z. A Figura 1.5 mostra um elemento prismático retangular com oito nós, localizados nos vértices, e a ordem adotada para a numeração dos nós do elemento: de baixo para cima, da esquerda para a direita e de trás para a frente. Pode-se numerar os nós em qualquer outra ordem, desde que ela seja mantida fixa até o final.

1.3.1 Formulação do problema

Para a descrição do método dos elementos finitos, vamos considerar o elemento prismático retangular da Figura 1.5 com comprimento $2a$, altura $2b$ e largura $2c$. Além disso, vamos adotar um sistema de coordenadas locais (ξ, η, ζ) cuja origem está localizada no baricentro do elemento.

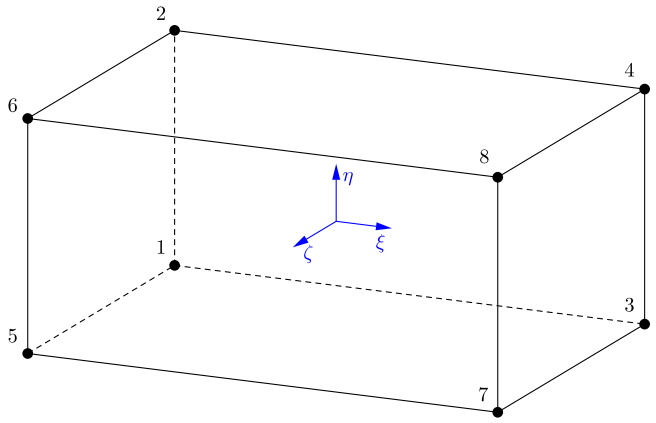


Figura 1.5: Elemento finito prismático retangular com oito nós.

Consideramos que as coordenadas locais (ξ, η, ζ) dos pontos do elemento assumem valores no intervalo $[-1, 1]$. Assim, as relações entre as coordenadas locais e as coordenadas globais (x, y, z) são dadas por

$$\xi = \frac{x}{a}, \qquad \eta = \frac{y}{b}, \qquad \zeta = \frac{z}{c}.$$

Nesse caso, as coordenadas locais de cada nó, conforme a numeração estabelecida, são mostradas na Tabela 1.1.

Tabela 1.1: Coordenadas locais dos nós do elemento.

Nó	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
ξ	-1	-1	$+1$	$+1$	-1	-1	$+1$	$+1$
η	-1	$+1$	-1	$+1$	-1	$+1$	-1	$+1$
ζ	-1	-1	-1	-1	$+1$	$+1$	$+1$	$+1$

Seja $\Omega_e \subset \Omega$ o conjunto dos pontos do elemento e. Escolhemos as funções $u, v, w : \Omega_e \longrightarrow \mathbb{R}$, dadas por

$$\begin{aligned}
u(\xi, \eta, \zeta) &= d_1 + d_2\xi + d_3\eta + d_4\zeta + d_5\xi\eta + d_6\eta\zeta + d_7\xi\zeta + d_8\xi\eta\zeta, \\
v(\xi, \eta, \zeta) &= d_9 + d_{10}\xi + d_{11}\eta + d_{12}\zeta + d_{13}\xi\eta + d_{14}\eta\zeta + d_{15}\xi\zeta + d_{16}\xi\eta\zeta, \\
w(\xi, \eta, \zeta) &= d_{17} + d_{18}\xi + d_{19}\eta + d_{20}\zeta + d_{21}\xi\eta + d_{22}\eta\zeta + d_{23}\xi\zeta + d_{24}\xi\eta\zeta,
\end{aligned}$$

para aproximar os deslocamentos dos pontos de Ω_e nas direções x, y e z, respectivamente. As funções u, v e w são trilineares, isto é, lineares em cada uma das variáveis. Por esse motivo, dizemos que o elemento prismático retangular com 8 nós é um elemento linear.

Mais adiante, nesta seção, apresentaremos elementos finitos que utilizam polinômios de grau maior para as funções aproximadoras.

A aproximação dos deslocamentos é feita por meio de uma interpolação, utilizando os nós do elemento como pontos interpoladores. Os coeficientes $d_1, d_2, ..., d_{24}$ são determinados particularizando as funções interpoladoras para cada nó. Considerando a função u, por exemplo, e utilizando a Tabela 1.1, temos

$$\begin{aligned}
\text{Nó } 1 &\longrightarrow u_1 = d_1 - d_2 - d_3 - d_4 + d_5 + d_6 + d_7 - d_8,\\
\text{Nó } 2 &\longrightarrow u_2 = d_1 - d_2 + d_3 - d_4 - d_5 - d_6 + d_7 + d_8,\\
\text{Nó } 3 &\longrightarrow u_3 = d_1 + d_2 - d_3 - d_4 - d_5 + d_6 - d_7 + d_8,\\
\text{Nó } 4 &\longrightarrow u_4 = d_1 + d_2 + d_3 - d_4 + d_5 - d_6 - d_7 - d_8,\\
\text{Nó } 5 &\longrightarrow u_5 = d_1 - d_2 - d_3 + d_4 + d_5 - d_6 - d_7 + d_8,\\
\text{Nó } 6 &\longrightarrow u_6 = d_1 - d_2 + d_3 + d_4 - d_5 + d_6 - d_7 - d_8,\\
\text{Nó } 7 &\longrightarrow u_7 = d_1 + d_2 - d_3 + d_4 - d_5 - d_6 + d_7 - d_8,\\
\text{Nó } 8 &\longrightarrow u_8 = d_1 + d_2 + d_3 + d_4 + d_5 + d_6 + d_7 + d_8,
\end{aligned}$$

em que $u_1, u_2, ..., u_8$ são os deslocamentos dos nós, chamados *deslocamentos nodais*, na direção x. Considerando que os deslocamentos nodais sejam conhecidos e resolvendo esse sistema de equações nas incógnitas $d_1, d_2, ..., d_8$, obtemos

$$\begin{aligned}
d_1 &= \frac{1}{8}\left(u_1 + u_2 + u_3 + u_4 + u_5 + u_6 + u_7 + u_8\right),\\
d_2 &= \frac{1}{8}\left(-u_1 - u_2 + u_3 + u_4 - u_5 - u_6 + u_7 + u_8\right),\\
d_3 &= \frac{1}{8}\left(-u_1 + u_2 - u_3 + u_4 - u_5 + u_6 - u_7 + u_8\right),\\
d_4 &= \frac{1}{8}\left(-u_1 - u_2 - u_3 - u_4 + u_5 + u_6 + u_7 + u_8\right),\\
d_5 &= \frac{1}{8}\left(u_1 - u_2 - u_3 + u_4 + u_5 - u_6 - u_7 + u_8\right),\\
d_6 &= \frac{1}{8}\left(u_1 - u_2 + u_3 - u_4 - u_5 + u_6 - u_7 + u_8\right),\\
d_7 &= \frac{1}{8}\left(u_1 + u_2 - u_3 - u_4 - u_5 - u_6 + u_7 + u_8\right),\\
d_8 &= \frac{1}{8}\left(-u_1 + u_2 + u_3 - u_4 + u_5 - u_6 - u_7 + u_8\right).
\end{aligned}$$

De maneira análoga, determinamos $d_9, d_{10}, ..., d_{16}$ em função dos deslocamentos nodais $v_1, v_2, ..., v_8$ na direção y e determinamos $d_{17}, d_{18}, ..., d_{24}$ em função dos deslocamentos nodais $w_1, w_2, ..., w_8$ na direção z. Substituindo os coeficientes nas funções interpoladoras e explicitando-as com relação aos deslocamentos nodais, obtemos

$$u(\xi, \eta, \zeta) = \sum_{i=1}^{8} \phi_i(\xi, \eta, \zeta) u_i, \tag{1.3}$$

$$v(\xi, \eta, \zeta) = \sum_{i=1}^{8} \phi_i(\xi, \eta, \zeta) v_i, \tag{1.4}$$

$$w(\xi, \eta, \zeta) = \sum_{i=1}^{8} \phi_i(\xi, \eta, \zeta) w_i, \tag{1.5}$$

em que as funções $\phi_i : \Omega_e \longrightarrow \mathbb{R}$, $i = 1, 2, ..., 8$, dadas por

$$\begin{aligned}
\phi_1(\xi,\eta,\zeta) &= \frac{1}{8}(1-\xi)(1-\eta)(1-\zeta),\\
\phi_2(\xi,\eta,\zeta) &= \frac{1}{8}(1-\xi)(1+\eta)(1-\zeta),\\
\phi_3(\xi,\eta,\zeta) &= \frac{1}{8}(1+\xi)(1-\eta)(1-\zeta),\\
\phi_4(\xi,\eta,\zeta) &= \frac{1}{8}(1+\xi)(1+\eta)(1-\zeta),\\
\phi_5(\xi,\eta,\zeta) &= \frac{1}{8}(1-\xi)(1-\eta)(1+\zeta),\\
\phi_6(\xi,\eta,\zeta) &= \frac{1}{8}(1-\xi)(1+\eta)(1+\zeta),\\
\phi_7(\xi,\eta,\zeta) &= \frac{1}{8}(1+\xi)(1-\eta)(1+\zeta),\\
\phi_8(\xi,\eta,\zeta) &= \frac{1}{8}(1+\xi)(1+\eta)(1+\zeta),
\end{aligned}$$

são denominadas *funções de forma*. Note que a função ϕ_i tem valor 1 no nó i e 0 nos demais nós. As equações (1.3), (1.4) e (1.5) descrevem os deslocamentos aproximados em um ponto do elemento como combinações lineares dos deslocamentos nodais.

Sejam $\mathrm{u}^{(e)} = \begin{bmatrix} u_1 & v_1 & w_1 & u_2 & v_2 & w_2 & \cdots & u_8 & v_8 & w_8 \end{bmatrix}^T$ o vetor de deslocamentos nodais do elemento e, $\hat{\mathrm{u}} = \begin{bmatrix} u(\xi,\eta,\zeta) & v(\xi,\eta,\zeta) & w(\xi,\eta,\zeta) \end{bmatrix}^T$ o vetor com os deslocamentos aproximados para um ponto do elemento e $\Phi \in \mathbb{R}^{3\times 24}$ a matriz dada por

$$\Phi = \begin{bmatrix} \phi_1 & 0 & 0 & \phi_2 & 0 & 0 & \cdots & \phi_8 & 0 & 0 \\ 0 & \phi_1 & 0 & 0 & \phi_2 & 0 & \cdots & 0 & \phi_8 & 0 \\ 0 & 0 & \phi_1 & 0 & 0 & \phi_2 & \cdots & 0 & 0 & \phi_8 \end{bmatrix}, \tag{1.6}$$

contendo as funções de forma aplicadas no mesmo ponto em que estamos aproximando os deslocamentos. Nesse caso, as equações (1.3), (1.4) e (1.5) podem ser escritas em forma matricial como

$$\hat{\mathrm{u}} = \Phi \mathrm{u}^{(e)}. \tag{1.7}$$

Logo, para aproximar os deslocamentos em qualquer ponto da estrutura precisamos obter o vetor de deslocamentos nodais $\mathrm{u}^{(e)}$. Para tanto, utilizamos alguns conceitos de resistência dos materiais, que podem ser vistos com mais detalhes em Hibbeler [29], por exemplo.

Por consequência do carregamento externo, a estrutura sofre *deformações*, que dependem dos deslocamentos nas direções x, y e z. Considerando os deslocamentos aproximados dados pelas funções u, v e w, as componentes de deformações lineares em um elemento finito são dadas por

$$\epsilon_x = \frac{\partial u}{\partial x}, \qquad \epsilon_y = \frac{\partial v}{\partial y}, \qquad \epsilon_z = \frac{\partial w}{\partial z},$$

e as componentes de deformações angulares são dadas por

$$\gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}, \qquad \gamma_{yz} = \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}, \qquad \gamma_{xz} = \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}.$$

Em forma matricial, temos

$$\begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \epsilon_z \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{xz} \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x} & 0 & 0 \\ 0 & \frac{\partial}{\partial y} & 0 \\ 0 & 0 & \frac{\partial}{\partial z} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial z} & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} & 0 & \frac{\partial}{\partial x} \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix}.$$

Denotando por $\epsilon = \begin{bmatrix} \epsilon_x & \epsilon_y & \epsilon_z & \gamma_{xy} & \gamma_{yz} & \gamma_{xz} \end{bmatrix}^T$ o vetor de deformações, L a matriz de operadores diferenciais e $\hat{u} = \begin{bmatrix} u & v & w \end{bmatrix}^T$, temos $\epsilon = L\hat{u}$. Com a equação (1.7), obtemos

$$\epsilon = L\Phi u^{(e)} = Bu^{(e)}, \tag{1.8}$$

em que $B = L\Phi$, denominada *matriz de deformação*, relaciona as deformações do elemento com os deslocamentos nodais.

Como uma forma de reação às cargas externas, surgem diversas forças no interior da estrutura. A intensidade média das forças internas distribuídas em uma seção do corpo é chamada de *tensão*. Para cada elemento finito tridimensional, consideramos três componentes de tensões normais: σ_x, σ_y e σ_z, e três componentes de tensões de cisalhamento: τ_{xy}, τ_{yz} e τ_{xz}.

Quando o material que compõe a estrutura está no regime elástico linear, a *lei de Hooke* nos diz que as deformações e as tensões se relacionam de forma linear, ou seja,

$$\sigma = D\epsilon, \tag{1.9}$$

em que $\sigma = \begin{bmatrix} \sigma_x & \sigma_y & \sigma_z & \tau_{xy} & \tau_{yz} & \tau_{xz} \end{bmatrix}^T$ é o vetor de tensões, ϵ o vetor de deformações e D é uma matriz que contém os parâmetros de elasticidade do material. Supomos que o material também é isotrópico, isto é, que possui as mesmas propriedades em todas as direções. Assim, a matriz D que aparece em (1.9) é dada por

$$D = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix}, \tag{1.10}$$

em que as constantes E e ν são, respectivamente, o *módulo de Young* (ou *módulo de elasticidade*) e o *coeficiente de Poisson* do material, com $E > 0$ e $0 < \nu < 1$. Observamos que, quando aplicamos o modelo SIMP, na prática podemos calcular a matriz D considerando $E = 1$ e o valor real do módulo de Young é inserido posteriormente, junto com a penalização das densidades, conforme veremos adiante.

Estamos considerando que a estrutura está sujeita à aplicação de uma carga externa q em uma região do domínio. Para um elemento finito e, o trabalho externo realizado pela carga q é dado por

$$\mathcal{W}_e = \int_{\Gamma_e} \hat{\mathrm{u}}^T q \, d\Gamma_e,$$

sendo Γ_e a fronteira do elemento. O trabalho interno, associado com a *energia de deformação*, é dado por

$$\mathcal{U}_e = \frac{1}{2} \int_{\Omega_e} \epsilon^T \sigma \, d\Omega_e,$$

em que Ω_e é o conjunto dos pontos do elemento. A *energia potencial total* do elemento finito é definida como a diferença entre o trabalho interno e o externo:

$$\Pi_e = \mathcal{U}_e - \mathcal{W}_e = \frac{1}{2} \int_{\Omega_e} \epsilon^T \sigma \, d\Omega_e - \int_{\Gamma_e} \hat{\mathrm{u}}^T q \, d\Gamma_e.$$

A presença de apoios que mantêm uma região da estrutura fixa permitem que ela se mantenha em equilíbrio estático após sofrer os deslocamentos. De acordo com o *princípio da mínima energia potencial total*, dentre os possíveis campos de deslocamentos, aquele que minimiza a energia potencial total é o que mantém o corpo em equilíbrio. Assim, queremos minimizar o funcional Π_e, que representa a energia potencial total para um elemento finito. Pode-se mostrar que a segunda variação desse funcional é sempre positiva no estado de equilíbrio. Logo, para minimizar esse funcional basta tomar a sua primeira variação igual a zero, ou seja,

$$\delta\Pi_e = \delta\mathcal{U}_e - \delta\mathcal{W}_e = 0. \tag{1.11}$$

De (1.8) obtemos $\delta\epsilon = \mathrm{B}\delta\mathrm{u}^{(e)}$. Daí, com a relação (1.9), temos que

$$\delta\mathcal{U}_e = \int_{\Omega_e} \delta\epsilon^T \sigma \, d\Omega_e = \int_{\Omega_e} \left(\delta\mathrm{u}^{(e)}\right)^T \mathrm{B}^T\mathrm{DB}\mathrm{u}^{(e)} \, d\Omega_e.$$

Agora, utilizando a equação (1.7) obtemos $\delta\hat{\mathrm{u}} = \Phi\delta\mathrm{u}^{(e)}$. Assim,

$$\delta\mathcal{W}_e = \int_{\Gamma_e} \delta\hat{\mathrm{u}}^T q \, d\Gamma_e = \int_{\Gamma_e} \left(\delta\mathrm{u}^{(e)}\right)^T \Phi^T q \, d\Gamma_e.$$

Logo, substituindo em (1.11), obtemos

$$\left(\delta\mathrm{u}^{(e)}\right)^T \left(\int_{\Omega_e} \mathrm{B}^T\mathrm{DB} \, d\Omega_e\right) \mathrm{u}^{(e)} = \left(\delta\mathrm{u}^{(e)}\right)^T \int_{\Gamma_e} \Phi^T q \, d\Gamma_e.$$

Como os deslocamentos $\delta\mathrm{u}^{(e)}$ são arbitrários, devemos ter

$$\left(\int_{\Omega_e} \mathrm{B}^T\mathrm{DB} \, d\Omega_e\right) \mathrm{u}^{(e)} = \int_{\Gamma_e} \Phi^T q \, d\Gamma_e.$$

Denotamos por

$$\mathrm{K}^{(e)} = \int_{\Omega_e} \mathrm{B}^T\mathrm{DB} \, d\Omega_e \tag{1.12}$$

a *matriz de rigidez do elemento* e por

$$\mathrm{f}^{(e)} = \int_{\Gamma_e} \Phi^T q \, d\Gamma_e \tag{1.13}$$

o *vetor de cargas nodais equivalentes do elemento*. Assim, obtemos o sistema linear

$$\mathrm{K}^{(e)}\mathrm{u}^{(e)} = \mathrm{f}^{(e)},$$

conhecido como sistema de equilíbrio estático para o elemento e.

A maneira como obtivemos o sistema de equilíbrio é similar para qualquer tipo de elemento finito tridimensional. Na próxima seção, veremos como calcular a matriz $\mathrm{K}^{(e)}$ e o vetor $\mathrm{f}^{(e)}$ para um elemento prismático retangular. A resolução desse sistema linear nos fornece o vetor de deslocamentos nodais. A flexibilidade média de cada elemento é então aproximada por

$$l(\hat{\mathrm{u}}) = \int_{\Gamma_e} \hat{\mathrm{u}}^T q \, d\Gamma_e = \int_{\Gamma_e} \mathrm{u}^{(e)^T} \Phi^T q \, d\Gamma_e = \mathrm{u}^{(e)^T} \int_{\Gamma_e} \Phi^T q \, d\Gamma_e = \mathrm{u}^{(e)^T} \mathrm{f}^{(e)} = \mathrm{f}^{(e)^T} \mathrm{u}^{(e)}.$$

Como a rigidez do elemento finito está diretamente relacionada às propriedades efetivas do material que o compõe, a penalização das densidades de material intermediárias, por meio da aplicação do modelo SIMP, é introduzida na matriz de rigidez do elemento, empregando-se a fórmula

$$\mathrm{K}^{(e)}(\rho_e) = \left(E_{min} + (\rho_e)^p (E_0 - E_{min})\right) \mathrm{K}^{(e)}, \tag{1.14}$$

para $e = 1, 2, ..., n_{el}$, em que ρ_e é a densidade do elemento e, p é o parâmetro de penalização do modelo SIMP, n_{el} é a quantidade total de elementos na malha, E_0 é o módulo de Young do material sólido e E_{min} é um número real positivo e pequeno se comparado com E_0. Neste caso, note que a matriz D em (1.10) deve ser calculada com o parâmetro $E = 1$.

Na prática, obtemos os deslocamentos nodais de todos os elementos de uma só vez, resolvendo um único sistema linear global, contendo informações de toda a malha de elementos finitos,

$$\mathrm{K}(\rho)\mathrm{u} = \mathrm{f}.$$

Além disso, a flexibilidade média de toda a estrutura é aproximada por $\mathrm{f}^T\mathrm{u}$. A matriz $\mathrm{K} \equiv \mathrm{K}(\rho)$ é denominada *matriz de rigidez global*, u é o *vetor global de deslocamentos nodais* e f é o *vetor global de cargas nodais equivalentes*.

A matriz de rigidez global é obtida pela superposição das matrizes de rigidez dos elementos, que dependem de ρ, logo pode ocorrer que K seja singular devido à presença de densidades muito próximas de zero. Uma maneira de evitar isso consiste em atribuir um valor mínimo positivo (mas pequeno o suficiente), E_{min}, para o módulo de Young do elemento vazio, como aparece em (1.14). Outra maneira equivalente consiste em atribuir um valor mínimo, ρ_{min}, para o limitante inferior das densidades.

Neste trabalho, utilizamos a estratégia do valor E_{min}, de modo que as densidades ρ_e possam assumir valores em todo o intervalo $[0, 1]$. A restrição de volume (1.2) após a discretização fica sendo

$$\sum_{e=1}^{n_{el}} v_e \rho_e \leq V_{max},$$

em que v_e é o volume do elemento e.

Portanto, o problema de otimização topológica estrutural na forma discretizada é dado pelo seguinte problema de otimização não linear:

$$\begin{aligned} \underset{\rho,\,\mathrm{u}}{\text{Min}} \quad & \mathrm{f}^T\mathrm{u} \\ \text{s. a} \quad & \mathrm{K}(\rho)\mathrm{u} = \mathrm{f}, \\ & \sum_{e=1}^{n_{el}} v_e\rho_e \leq V_{max}, \\ & 0 \leq \rho_e \leq 1, \quad e = 1, 2, ..., n_{el}. \end{aligned} \tag{1.15}$$

Após a imposição das condições de contorno do problema, relacionadas aos apoios que sustentam a estrutura, a matriz de rigidez K se torna definida positiva e, portanto, não singular, sendo possível resolver o sistema linear e obter o vetor de deslocamentos nodais $\mathrm{u} = \mathrm{K}(\rho)^{-1}\mathrm{f}$. Nesse caso, o problema (1.15) pode ser reescrito como

$$\begin{aligned} \underset{\rho}{\text{Min}} \quad & \mathrm{f}^T\mathrm{K}(\rho)^{-1}\mathrm{f} \\ \text{s. a} \quad & \sum_{e=1}^{n_{el}} v_e\rho_e \leq V_{max}, \\ & 0 \leq \rho_e \leq 1, \quad e = 1, 2, ..., n_{el}. \end{aligned} \tag{1.16}$$

Note que a matriz K depende de ρ, de modo que precisamos resolver um sistema linear diferente a cada vez que ρ se altera durante a resolução do problema.

1.3.2 Elementos prismáticos retangulares do tipo Lagrange

O método dos elementos finitos foi descrito utilizando o elemento prismático retangular com oito nós, representado na Figura 1.5. Para a aplicação do método, precisamos escolher funções u, v e w que aproximam os deslocamentos dos pontos do elemento em cada uma das direções x, y e z, respectivamente. Para o elemento com oito nós, as funções escolhidas são trilineares, isto é, lineares em cada uma das três variáveis ξ, η e ζ. Dito de outra forma, as funções aproximadoras são combinações dos termos polinomiais

$$1, \quad \xi, \quad \eta, \quad \zeta, \quad \xi\eta, \quad \eta\zeta, \quad \xi\zeta \quad \text{e} \quad \xi\eta\zeta,$$

nos quais as variáveis aparecem com expoente igual a 0 ou 1. Utilizando os oito nós do elemento como pontos interpoladores, é possível determinar unicamente os oito coeficientes da combinação que define cada função aproximadora.

Dizemos que o elemento prismático retangular com oito nós é linear ou que ele é um elemento de grau 1. As funções de forma desse elemento são dadas por

$$\phi_i(\xi, \eta, \zeta) = \frac{1}{8}(1 + \xi_i\xi)(1 + \eta_i\eta)(1 + \zeta_i\zeta),$$

para $i = 1, 2, ..., 8$, em que ξ_i, η_i e ζ_i são as coordenadas locais do nó i. Note que essas funções de forma são produtos das funções unidimensionais dadas por

$$l_i(\xi) = \frac{1}{2}(1 + \xi_i\xi), \quad l_i(\eta) = \frac{1}{2}(1 + \eta_i\eta) \quad \text{e} \quad l_i(\zeta) = \frac{1}{2}(1 + \zeta_i\zeta),$$

que são polinômios de Lagrange de grau 1. Por esse motivo, também dizemos que esse elemento pertence à família de elementos do tipo Lagrange.

O elemento linear é bastante popular devido ao seu baixo número de nós, o que é atrativo para problemas tridimensionais. Além disso, ele apresenta bons resultados para os problemas de otimização topológica. Entretanto, pode ser que a aproximação linear dos deslocamentos da estrutura não seja adequada, sendo necessária uma precisão melhor para essa aproximação. Nesse caso, podemos utilizar elementos cujas funções aproximadoras são compostas por polinômios de grau maior.

O elemento prismático retangular de grau 2 (quadrático) do tipo Lagrange possui 27 nós, conforme a Figura 1.6. As funções escolhidas para aproximar os deslocamentos são combinações dos termos polinomiais

$$\begin{aligned}&1, \quad \xi, \quad \eta, \quad \zeta, \quad \xi\eta, \quad \eta\zeta, \quad \xi\zeta, \quad \xi\eta\zeta, \quad \xi^2, \quad \eta^2, \quad \zeta^2,\\ &\xi^2\eta, \quad \xi^2\zeta, \quad \xi\eta^2, \quad \eta^2\zeta, \quad \xi\zeta^2, \quad \eta\zeta^2, \quad \xi^2\eta^2, \quad \xi^2\zeta^2, \quad \eta^2\zeta^2,\\ &\xi^2\eta\zeta, \quad \xi\eta^2\zeta, \quad \xi\eta\zeta^2, \quad \xi^2\eta^2\zeta, \quad \xi^2\eta\zeta^2, \quad \xi\eta^2\zeta^2 \quad \text{e} \quad \xi^2\eta^2\zeta^2,\end{aligned}$$

em que as variáveis aparecem com expoente igual a 0, 1 ou 2.

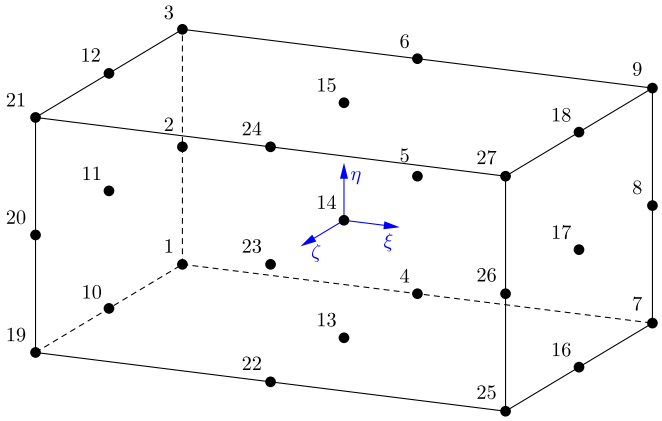


Figura 1.6: Elemento finito prismático retangular de grau 2 do tipo Lagrange.

As funções de forma são obtidas pelo produto de polinômios quadráticos de Lagrange unidimensionais. Para os nós localizados nos vértices do prisma, temos

$$\phi_i(\xi,\eta,\zeta) = \frac{1}{8}(\xi^2+\xi_i\xi)(\eta^2+\eta_i\eta)(\zeta^2+\zeta_i\zeta),$$

para $i = 1, 3, 7, 9, 19, 21, 25, 27$. No interior das arestas do prisma, temos

$$\begin{aligned}\phi_i(\xi,\eta,\zeta) = &\frac{1}{4}\eta_i^2\zeta_i^2(1-\xi^2)(\eta^2+\eta_i\eta)(\zeta^2+\zeta_i\zeta)\\ &+\frac{1}{4}\xi_i^2\zeta_i^2(\xi^2+\xi_i\xi)(1-\eta^2)(\zeta^2+\zeta_i\zeta)+\frac{1}{4}\xi_i^2\eta_i^2(\xi^2+\xi_i\xi)(\eta^2+\eta_i\eta)(1-\zeta^2),\end{aligned}$$

para $i = 2, 4, 6, 8, 10, 12, 16, 18, 20, 22, 24, 26$. No interior das faces do prisma, temos

$$\begin{aligned}\phi_i(\xi,\eta,\zeta) = &\frac{1}{2}(1-\xi^2)(1-\eta^2)(\zeta^2+\zeta_i\zeta)\\ &+\frac{1}{2}(\xi^2+\xi_i\xi)(1-\eta^2)(1-\zeta^2)+\frac{1}{2}(1-\xi^2)(\eta^2+\eta_i\eta)(1-\zeta^2),\end{aligned}$$

para $i = 5, 11, 13, 15, 17, 23$. Por fim, no centro do elemento, temos

$$\phi_{14}(\xi, \eta, \zeta) = (1 - \xi^2)(1 - \eta^2)(1 - \zeta^2).$$

Note que cada função de forma ϕ_i assume valor 1 no nó i e zero nos demais nós, assim como ocorre com as funções do elemento linear.

Quando utilizamos o elemento quadrático, a aproximação dos deslocamentos da estrutura fica mais precisa do que com o elemento linear, porém isso implica em um custo computacional maior, uma vez que a quantidade de nós em cada elemento aumenta de 8 para 27, de modo que o número total de nós na malha cresce, aumentando bastante a dimensão dos sistemas lineares de equilíbrio. Caso ainda se deseje uma precisão melhor, é possível utilizar elementos de grau maior. O próximo elemento prismático retangular do tipo Lagrange é o de grau 3 (cúbico), que possui 64 nós. Entretanto, esse elemento, em geral, não é competitivo com o elemento de grau 3 do tipo *serendipity*, que apresentamos a seguir.

1.3.3 Elementos prismáticos retangulares do tipo *serendipity*

Os elementos do tipo *serendipity* utilizam menos termos polinomiais nas funções que aproximam os deslocamentos e, consequentemente, possuem menos nós. Por outro lado, o grau da aproximação obtida é equivalente ao do elemento correspondente do tipo Lagrange, o que faz com que os elementos *serendipity* sejam bastante empregados.

O elemento linear é igual ao da família Lagrange. Já o elemento prismático retangular de grau 2 do tipo *serendipity* possui 20 nós, localizados nos vértices e nos pontos médios das arestas do prisma, conforme ilustra a Figura 1.7. As funções escolhidas para aproximar os deslocamentos são combinações dos termos polinomiais

$$\begin{aligned} &1, \quad \xi, \quad \eta, \quad \zeta, \quad \xi\eta, \quad \eta\zeta, \quad \xi\zeta, \quad \xi\eta\zeta, \quad \xi^2, \quad \eta^2, \quad \zeta^2, \\ &\xi^2\eta, \quad \xi^2\zeta, \quad \xi\eta^2, \quad \eta^2\zeta, \quad \xi\zeta^2, \quad \eta\zeta^2, \quad \xi^2\eta\zeta, \quad \xi\eta^2\zeta, \quad \xi\eta\zeta^2, \end{aligned}$$

em que não aparecem os termos contendo mais do que uma variável elevada ao quadrado, que estão na aproximação do elemento quadrático de Lagrange. Apesar da aproximação ficar menos precisa, ela continua sendo de grau 2 e há uma economia de 21 graus de liberdade por elemento, o que explica a popularidade do elemento *serendipity* em aplicações.

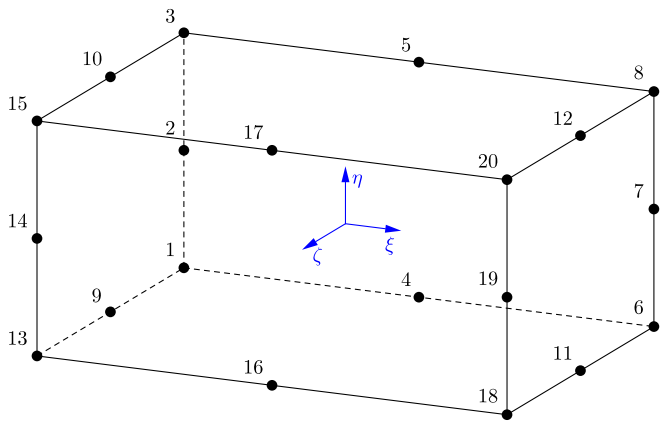


Figura 1.7: Elemento finito prismático retangular de grau 2 do tipo *serendipity*.

As funções de forma para os nós nos vértices do prisma são dadas por

$$\phi_i(\xi,\eta,\zeta) = \frac{1}{8}(1+\xi_i\xi)(1+\eta_i\eta)(1+\zeta_i\zeta)(\xi_i\xi+\eta_i\eta+\zeta_i\zeta-2),$$

para $i = 1, 3, 6, 8, 13, 15, 18, 20$. E no interior das arestas do prisma, temos

$$\phi_i(\xi,\eta,\zeta) = \frac{1}{4}(1-\xi^2)(1+\eta_i\eta)(1+\zeta_i\zeta), \quad \text{para } i = 4, 5, 16, 17;$$

$$\phi_i(\xi,\eta,\zeta) = \frac{1}{4}(1+\xi_i\xi)(1-\eta^2)(1+\zeta_i\zeta), \quad \text{para } i = 2, 7, 14, 19;$$

$$\phi_i(\xi,\eta,\zeta) = \frac{1}{4}(1+\xi_i\xi)(1+\eta_i\eta)(1-\zeta^2), \quad \text{para } i = 9, 10, 11, 12.$$

O elemento prismático retangular de grau 3 do tipo *serendipity* possui apenas 32 nós, sendo 8 nos vértices e 24 ao longo das arestas do prisma, conforme a Figura 1.8, o que o torna muito mais eficiente do que o elemento cúbico de Lagrange que possui 64 nós. As funções aproximadoras são combinações dos termos polinomiais

$$\begin{array}{l} 1, \quad \xi, \quad \eta, \quad \zeta, \quad \xi\eta, \quad \eta\zeta, \quad \xi\zeta, \quad \xi\eta\zeta, \quad \xi^2, \quad \eta^2, \quad \zeta^2, \quad \xi^2\eta, \\ \xi^2\zeta, \quad \xi\eta^2, \quad \eta^2\zeta, \quad \xi\zeta^2, \quad \eta\zeta^2, \quad \xi^2\eta\zeta, \quad \xi\eta^2\zeta, \quad \xi\eta\zeta^2, \quad \xi^3, \quad \eta^3, \quad \zeta^3, \\ \xi^3\eta, \quad \xi^3\zeta, \quad \xi\eta^3, \quad \eta^3\zeta, \quad \xi\zeta^3, \quad \eta\zeta^3, \quad \xi^3\eta\zeta, \quad \xi\eta^3\zeta \quad \text{e} \quad \xi\eta\zeta^3, \end{array}$$

de modo que a aproximação é de grau 3.

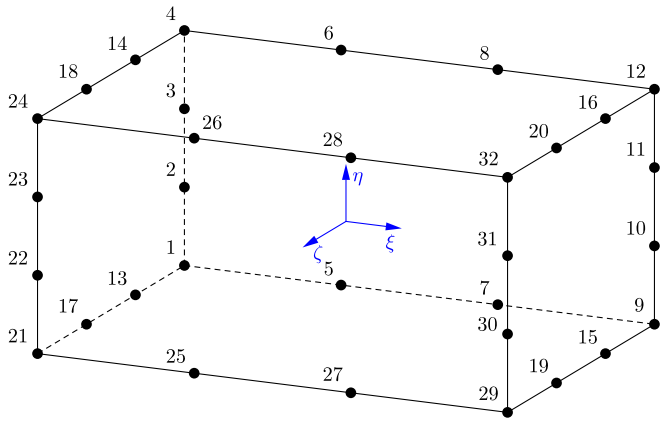


Figura 1.8: Elemento finito prismático retangular de grau 3 do tipo *serendipity*.

As funções de forma nos vértices são dadas por

$$\phi_i(\xi,\eta,\zeta) = \frac{1}{64}(1+\xi_i\xi)(1+\eta_i\eta)(1+\zeta_i\zeta)(9\xi^2+9\eta^2+9\zeta^2-19),$$

para $i = 1, 4, 9, 12, 21, 24, 29, 32$. Já no interior das arestas do prisma, temos

$$\phi_i(\xi,\eta,\zeta) = \frac{9}{64}(1-\xi^2)(1+9\xi_i\xi)(1+\eta_i\eta)(1+\zeta_i\zeta), \quad \text{para } i = 5, 6, 7, 8, 25, 26, 27, 28;$$

$$\phi_i(\xi,\eta,\zeta) = \frac{9}{64}(1+\xi_i\xi)(1-\eta^2)(1+9\eta_i\eta)(1+\zeta_i\zeta), \quad \text{para } i = 2, 3, 10, 11, 22, 23, 30, 31;$$

$$\phi_i(\xi,\eta,\zeta) = \frac{9}{64}(1+\xi_i\xi)(1+\eta_i\eta)(1-\zeta^2)(1+9\zeta_i\zeta), \quad \text{para } i = 13, 14, 15, 16, 17, 18, 19, 20.$$

Em problemas tridimensionais, utilizar elementos de grau maior do que 3 pode tornar o algoritmo de resolução ineficiente, além de deixar a implementação mais complicada. Por isso, neste trabalho, consideramos apenas elementos de grau 1, 2 ou 3.

1.4 Sistema de equilíbrio estático

Conforme vimos anteriormente, o cálculo da função objetivo do problema de otimização topológica estrutural (1.15) envolve o vetor de deslocamentos nodais, u, solução do sistema linear

$$\mathrm{K}(\rho)\mathrm{u} = \mathrm{f},$$

que representa as condições de equilíbrio estático para toda a estrutura. Isso traz uma das maiores dificuldades da resolução, uma vez que a matriz K depende das densidades ρ, que são variáveis do problema.

A matriz de rigidez global, K, que representa todo o domínio discretizado, é obtida por meio de uma superposição das matrizes de rigidez de todos os elementos da malha. De maneira análoga, o vetor global de deslocamentos nodais, f, é obtido por meio de uma superposição dos vetores de cargas nodais equivalentes de cada elemento. Nesta seção, explicamos como calcular a matriz $\mathrm{K}^{(e)}$ e o vetor $\mathrm{f}^{(e)}$ para um elemento prismático retangular e como construir o sistema de equilíbrio global.

1.4.1 Matriz de rigidez do elemento

A matriz de rigidez de um elemento é dada pela integral em (1.12). No caso do elemento prismático retangular com comprimento $2a$, altura $2b$ e largura $2c$, utilizando as coordenadas locais (ξ, η, ζ), temos $d\Omega_e = dx\, dy\, dz = abc\, d\xi\, d\eta\, d\zeta$. Logo,

$$\mathrm{K}^{(e)} = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} abc\, \mathrm{B}^T\mathrm{DB}\, d\xi\, d\eta\, d\zeta. \tag{1.17}$$

Essa matriz é simétrica, uma vez que D, dada em (1.10), é simétrica. Considerando uma malha uniforme, isto é, com todos os elementos do mesmo tipo e tamanho, a matriz $\mathrm{K}^{(e)}$ só precisa ser calculada uma vez, pois será a mesma para todos os elementos da malha.

No caso do elemento linear, com 8 nós, essa matriz tem ordem 24×24. Embora seja possível calcular a integral tripla em (1.17) de forma analítica e obter cada entrada da matriz, esse cálculo se torna muito difícil se usarmos elementos de grau maior que 1 ou se a malha não for uniforme, além de não ser conveniente para implementações. Na prática, aproximamos a integral numericamente, utilizando a quadratura Gaussiana.

Para exemplificar, considere inicialmente uma função contínua, f, com apenas uma variável, ξ, no intervalo $[-1, 1]$ e suponha que queiramos calcular a integral definida de f nesse intervalo. Uma regra de quadratura Gaussiana fornece o valor aproximado da integral definida utilizando uma combinação linear de valores da função em certos pontos ξ_i, com $-1 \leq \xi_i \leq 1$, e pesos w_i. Ou seja, a integral é calculada somando-se os produtos do peso em cada ponto pelo valor da função no mesmo ponto:

$$\int_{-1}^{1} f(\xi)\, d\xi \approx w_1 f(\xi_1) + w_2 f(\xi_2) + \cdots + w_n f(\xi_n) = \sum_{i=1}^{n} w_i f(\xi_i). \tag{1.18}$$

Os pontos ξ_i e os pesos w_i são determinados de modo que a regra forneça a integral de forma exata para qualquer polinômio de grau $2n-1$, sendo n a quantidade de pontos tomados no intervalo $[-1,1]$.

As regras de quadratura são definidas para o domínio de integração $[-1,1]$. A Tabela 1.2 mostra as coordenadas $\xi_i \in [-1,1]$ dos pontos de integração e os pesos w_i calculados para integrais de ordem $n = 1,2,3,4$.

Tabela 1.2: Quadratura Gaussiana para $\xi \in [-1,1]$.

Ordem n	$\boldsymbol{\xi_i}$	$\boldsymbol{w_i}$
1	0	2
2	$-\frac{1}{\sqrt{3}} \quad \frac{1}{\sqrt{3}}$	$1 \quad 1$
3	$-\sqrt{\frac{3}{5}} \quad 0 \quad \sqrt{\frac{3}{5}}$	$\frac{5}{9} \quad \frac{8}{9} \quad \frac{5}{9}$
4	$-\sqrt{\frac{3}{7}+\frac{2}{7}\sqrt{\frac{6}{5}}} \quad -\sqrt{\frac{3}{7}-\frac{2}{7}\sqrt{\frac{6}{5}}}$ $\sqrt{\frac{3}{7}-\frac{2}{7}\sqrt{\frac{6}{5}}} \quad \sqrt{\frac{3}{7}+\frac{2}{7}\sqrt{\frac{6}{5}}}$	$\frac{18-\sqrt{30}}{36} \quad \frac{18+\sqrt{30}}{36}$ $\frac{18+\sqrt{30}}{36} \quad \frac{18-\sqrt{30}}{36}$

A quadratura Gaussiana pode ser estendida para duas e três dimensões. Seja $f(\xi,\eta,\zeta)$ uma função que queiramos integrar no domínio $[-1,1]^3$. Então,

$$\int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} f(\xi,\eta,\zeta)\, d\xi\, d\eta\, d\zeta \approx \sum_{k=1}^{n_z}\sum_{j=1}^{n_y}\sum_{i=1}^{n_x} w_k w_j w_i\, f(\xi_i,\eta_j,\zeta_k), \tag{1.19}$$

em que n_x, n_y e n_z são os números de pontos de integração nas direções ξ, η e ζ, respectivamente, e w_i, w_j e w_k são os pesos correspondentes. Não é necessário considerar o mesmo número de pontos de integração nas três direções, mas, por conveniência, em nossa implementação utilizamos $n_x = n_y = n_z$ e esse valor foi considerado como sendo o grau do elemento mais um.

Utilizando a fórmula de quadratura Gaussiana em (1.19) e os dados da Tabela 1.2, podemos calcular a integral em (1.17) e obter a matriz de rigidez do elemento. Neste caso, consideramos a função matricial $f(\xi,\eta,\zeta) = \mathrm{B}(\xi,\eta,\zeta)^T \mathrm{D}\mathrm{B}(\xi,\eta,\zeta)$.

1.4.2 Vetor de cargas nodais do elemento

O vetor de cargas nodais do elemento é dado pela integral em (1.13). Na descrição do método, consideramos a aplicação de uma única carga externa q, embora

possa haver um conjunto de mais cargas sendo aplicadas na estrutura. Nesse caso, cada uma das cargas geraria um trabalho externo e todos os trabalhos se somariam produzindo o trabalho total, $\mathcal{W}_e$, do elemento finito. Assim, a contribuição de cada carga é somada no vetor $\mathrm{f}^{(e)}$.

Na prática, o cálculo da integral só é necessário nos casos em que o carregamento é não uniforme ou quando a área de aplicação da carga é irregular. Neste trabalho, consideramos problemas com cargas de três tipos: concentrada em um ponto; uniformemente distribuída em uma das direções x, y ou z; e uniformemente distribuída em um retângulo paralelo a um dos planos xy, yz ou xz. Logo, as regiões de aplicação das cargas são pontos, segmentos de reta ou retângulos, respectivamente.

As três primeiras entradas do vetor $\mathrm{f}^{(e)}$ representam a contribuição das componentes x, y e z (nessa ordem) das cargas no primeiro nó do elemento, as três seguintes correspondem ao segundo nó e assim por diante. Para o elemento linear, com 8 nós, o vetor $\mathrm{f}^{(e)}$ tem 24 entradas e numeramos os nós do elemento conforme a Figura 1.5.

Consideremos o exemplo ilustrado na Figura 1.9, em que uma carga concentrada, $q_1 = \left[\begin{array}{ccc} 1\ N & -1\ N & 0\ N \end{array}\right]^T$, está aplicada no nó de número 7 do elemento destacado, e uma carga uniformemente distribuída na direção x, $q_2 = \left[\begin{array}{ccc} 0\ \frac{N}{mm} & -1\ \frac{N}{mm} & 0\ \frac{N}{mm} \end{array}\right]^T$, está aplicada por toda a aresta que une os nós 2 e 4 do elemento destacado e também passa pela aresta de um elemento vizinho na malha. Suponhamos que as dimensões dos elementos sejam todas iguais a 1 mm.

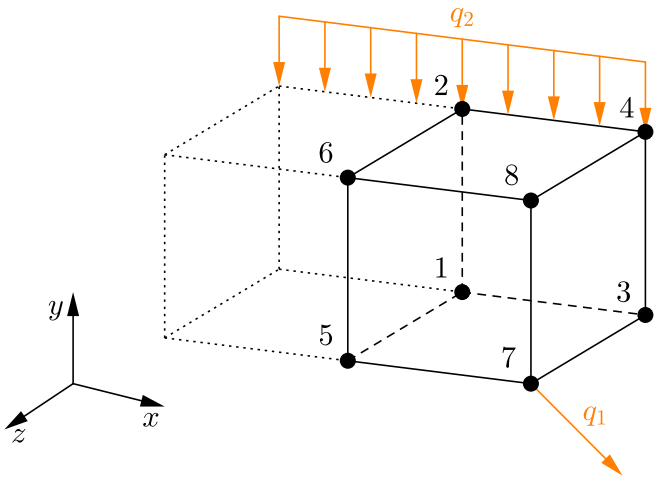


Figura 1.9: Exemplo de aplicação de cargas.

Iniciamos o vetor $\mathrm{f}^{(e)}$ como um vetor nulo com 24 entradas. Para cargas concentradas, toda contribuição é passada para o nó mais próximo do ponto de aplicação. No exemplo, as componentes da carga q_1 são passadas para as entradas 19, 20 e 21 do vetor, que correspondem ao nó de número 7.

No caso da carga uniformemente distribuída, q_2, sua contribuição é separada da seguinte maneira. Sejam a_1 a aresta que une os nós 2 e 4 do elemento e a_2 a aresta do elemento vizinho, paralela a a_1, que também passa pelo nó 2. A parte da carga que está depois do ponto médio da aresta a_2 não contribui para o elemento em destaque. A parte da carga no segmento que vai do ponto médio da aresta a_2 ao ponto médio da aresta a_1 é passada para o nó 2, de modo que as componentes da carga são multiplicadas pelo comprimento desse segmento e passadas para as entradas 4, 5 e 6 do vetor. Por fim, a

parte da carga no segmento que vai do ponto médio da aresta a_1 ao nó 4 é passada para o nó 4, de modo que as componentes da carga são multiplicadas pelo comprimento desse segmento e passadas para as entradas 10, 11 e 12 do vetor. Portanto, temos

$$\mathrm{f}^{(e)} = \begin{bmatrix} 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & -0{,}5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \end{bmatrix}^T.$$

Caso algum nó recebesse as contribuições de duas ou mais cargas distintas, elas seriam somadas nas entradas correspondentes do vetor. A análise para cargas uniformemente distribuídas nas direções y ou z é análoga. No caso de cargas uniformemente distribuídas em um retângulo paralelo a algum dos planos xy, yz ou xz, fazemos algo semelhante, mas ao invés de calcular os comprimentos das partes da carga correspondentes a cada nó, precisamos calcular áreas de retângulos.

1.4.3 Sistema global

Para obter o sistema de equilíbrio global, precisamos adotar uma convenção para a numeração global dos elementos, dos nós e dos graus de liberdade associados a cada nó, de modo a obedecer a conectividade entre os elementos da malha. Convencionamos que a numeração dos nós e dos elementos deve começar no canto inferior esquerdo mais ao fundo da malha e prosseguir sempre de baixo para cima, da esquerda para a direita e de trás para a frente, de tal forma que o último nó (e o último elemento) seja o do canto superior direito mais à frente da malha. A Figura 1.10 mostra um exemplo de malha com a numeração global dos nós. A numeração dos elementos segue a mesma ordem.

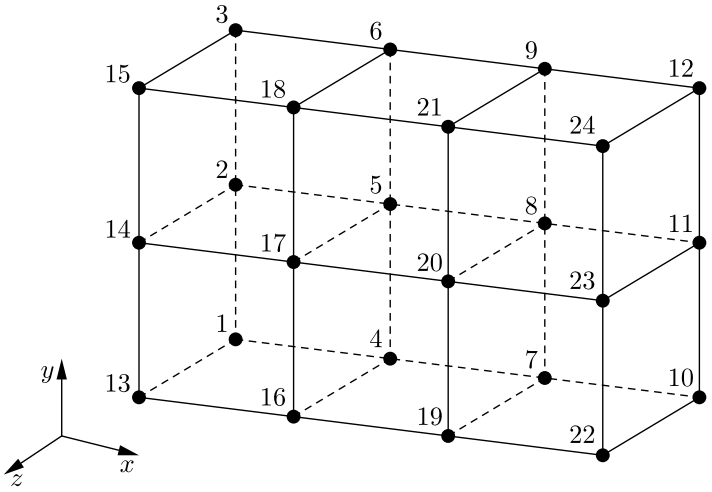


Figura 1.10: Exemplo de uma malha com a numeração global dos nós.

Cada nó j possui três graus de liberdade, cujos índices são $3j-2$ (deslocamento nodal na direção x), $3j-1$ (deslocamento nodal na direção y) e $3j$ (deslocamento nodal na direção z). As três primeiras linhas (e colunas) da matriz K correspondem aos graus de liberdade do nó 1, as linhas (e colunas) 4, 5 e 6 correspondem aos graus de liberdade do nó 2, e assim por diante. Dessa forma, a dimensão inicial da matriz K é $3n_n \times 3n_n$ e a dimensão inicial dos vetores u e f é $3n_n \times 1$, em que n_n é o número total de nós na malha.

Na malha da Figura 1.10, o elemento de número 3, por exemplo, está conectado aos nós globais 4, 5, 7, 8, 16, 17, 19 e 20, nesta ordem, conforme a numeração local dos

nós em cada elemento, seguindo a Figura 1.5. Os graus de liberdade associados a esse elemento são os de índices 10, 11, 12, 13, 14, 15, 19, 20, 21, 22, 23, 24, 46, 47, 48, 49, 50, 51, 55, 56, 57, 58, 59 e 60.

Para cada elemento finito e, obtemos a matriz de rigidez $\mathrm{K}^{(e)}$ e guardamos os índices dos 24 graus de liberdade associados aos 8 nós do elemento, em ordem, em um vetor, digamos *dofs*. Cada componente $\mathrm{K}_{ij}^{(e)}$, para $i, j = 1, 2, ..., 24$, é adicionada na posição $(dofs_i,\ dofs_j)$ da matriz K. Na prática, construímos a matriz de rigidez global de forma esparsa, armazenando o vetor $i\mathrm{K}$ com os índices das linhas dos elementos não nulos de K, o vetor $j\mathrm{K}$ com os índices das colunas correspondentes e o vetor $s\mathrm{K}$ com os valores dos elementos da matriz. Os vetores $i\mathrm{K}$ e $j\mathrm{K}$ podem ser construídos uma única vez, desde que o refinamento da malha e o tipo dos elementos não mudem durante o processo; já o vetor $s\mathrm{K}$ deve ser atualizado a cada vez que as densidades ρ se alteram.

A Figura 1.11 mostra a disposição das entradas não nulas da matriz de rigidez global para a malha com seis elementos da Figura 1.10. Destacadas com "×", em vermelho, estão as entradas correspondentes à matriz de rigidez do elemento de número 3. Nesse caso, a matriz K é de ordem 72×72. Note que ela possui uma estrutura em bandas e é esparsa. Quanto maior for o refinamento da malha, mais afastadas estarão as bandas.

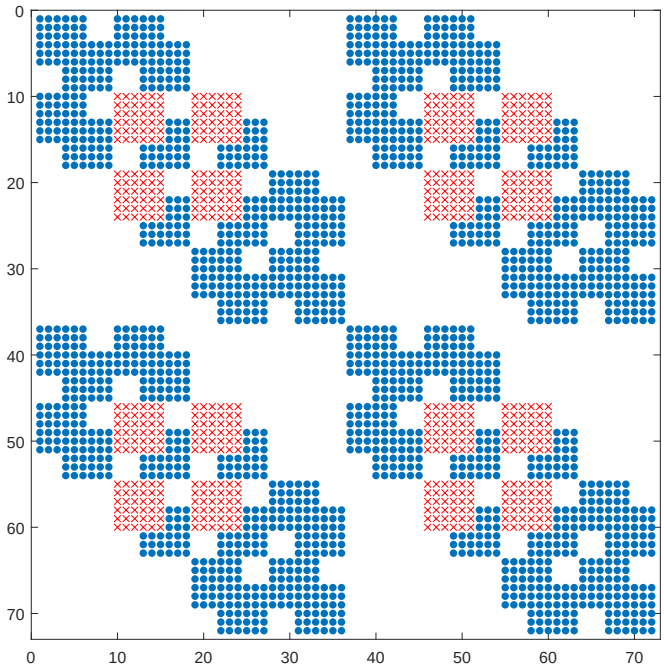


Figura 1.11: Padrão de esparsidade da matriz de rigidez global para uma malha com $3 \times 2 \times 1$ elementos finitos. As entradas com o marcador "×", em vermelho, correspondem à contribuição do elemento de número 3.

Se nenhuma *condição de contorno* relativa aos deslocamentos nodais for imposta ao problema, a matriz K não terá posto completo. Fisicamente, isso acontece porque um corpo sem nenhum tipo de restrição aos deslocamentos pode apresentar movimentos de corpo rígido, ou seja, pode não estar em equilíbrio. No caso tridimensional, desconsiderando as condições de contorno, o posto da matriz de rigidez será igual ao número de linhas (ou colunas) menos 6. O valor 6 representa os possíveis movimentos dos pontos da estrutura, sendo 3 translações nas direções dos eixos x, y, z e 3 rotações em torno de cada eixo. Precisamos restringir cada um desses movimentos em pelo menos um dos nós

da malha. Fazemos isso impondo a presença de apoios em determinadas regiões do domínio, com o objetivo de impedir certos deslocamentos. Na prática, eliminamos as linhas e colunas de K, bem como as componentes dos vetores u e f, correspondentes aos graus de liberdade que foram restringidos nos nós com apoio. Por exemplo, se no nó 3 há um apoio que impede o deslocamento desse nó nas direções x e z, então eliminamos de K as linhas e colunas de índices 7 e 9.

Após introduzir as condições de contorno, a matriz K, que é simétrica, passa a ser definida positiva, sendo possível utilizar a fatoração de Cholesky ou o método dos gradientes conjugados (veja Watkins [56], seções 1.4 e 7.6, respectivamente), por exemplo, para resolver o sistema linear.

Como a matriz K depende das densidades ρ, precisamos resolver um sistema linear com matriz diferente a cada vez que ρ se altera. Em diversos trabalhos com implementações de algoritmos para resolver problemas de otimização topológica [19, 30, 35, 53], constata-se que a resolução desses sistemas lineares é a etapa mais cara – ou seja, que consome mais tempo – na resolução dos problemas, atingindo, em média, mais da metade do tempo total de resolução. Diante dessa constatação, faz-se importante o estudo de métodos mais eficientes para a resolução desses sistemas lineares.

A fatoração de Cholesky é um método direto e acurado, bastante utilizado e recomendado para resolver sistemas lineares com matriz simétrica definida positiva. Para aplicá-lo precisamos obter o fator de Cholesky da matriz, que geralmente possui muitas entradas não nulas, apesar da matriz de rigidez ser esparsa. Dessa forma, o armazenamento desses fatores demanda muita memória do computador. Esta situação se agrava nos problemas tridimensionais, nos quais as dimensões da matriz crescem demasiadamente sem que aumentemos muito o número de elementos na malha.

Experimentos computacionais mostram que um método iterativo, como o dos gradientes conjugados, é mais eficiente nos casos em que o número de elementos na malha é muito grande (veja [30, 53]). Entretanto, para se obter uma melhor eficácia do método, precisamos utilizar um precondicionador adequado. Costuma-se utilizar como precondicionador a fatoração incompleta de Cholesky, que apresenta bons resultados, mas ainda assim o tempo de resolução dos sistemas lineares ocupa a maior parcela do tempo de resolução dos problemas de otimização topológica tridimensionais.

Neste trabalho, estudamos um outro tipo de método, conhecido como *multigrid*, que promete ser mais eficiente na resolução dos sistemas lineares que aparecem em problemas envolvendo discretizações, em particular no problema de otimização topológica (vide Amir [1], Peetz e Elbanna [40]), podendo também ser utilizado como um precondicionador eficiente para o método dos gradientes conjugados.

1.5 Instabilidades numéricas

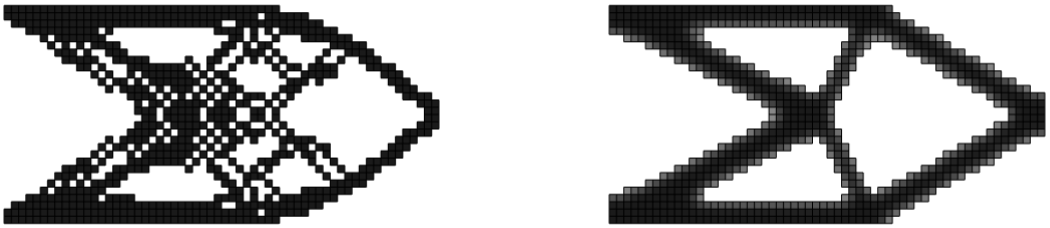
A resolução do problema de otimização topológica na forma discretizada (1.16) pode apresentar algumas *instabilidades numéricas*. Os autores Sigmund e Petersson [48] classificaram essas instabilidades em três categorias principais: mínimos locais, dependência da malha e tabuleiros de xadrez.

O problema dos mínimos locais se refere ao fato de que a função objetivo

possui muitos mínimos locais, de modo que podemos encontrar soluções distintas quando utilizamos diferentes parâmetros do algoritmo de resolução. Geralmente, nos problemas de otimização, desejamos encontrar o minimizador (ou maximizador) global da função, mas isso é muito difícil no caso da otimização topológica, de modo que nos contentamos com alguma solução local.

A dependência da malha refere-se ao caso em que não obtemos, qualitativamente, a mesma estrutura para diferentes refinamentos da malha. Isso pode ser bom ou ruim, dependendo do que se deseja. Ao refinar a malha, podemos estar tentando encontrar mais detalhes no formato da estrutura. Por outro lado, podemos querer apenas suavizar os contornos e melhorar a qualidade de um formato que já obtivemos em uma malha mais grosseira, de tal modo que desejamos resultados independentes da malha.

Já o que chamamos de tabuleiro de xadrez é a distribuição do material formando regiões com elementos vazios e sólidos, de maneira alternada, semelhante a um tabuleiro de xadrez, como na Figura 1.12(a). De acordo com Díaz e Sigmund [16], esse padrão torna a estrutura artificialmente mais rígida e uma das causas do tabuleiro é o tipo de elemento finito utilizado (retangular com nós apenas nos vértices). Entretanto, a produção da estrutura nesse formato é inviável. O ideal é que o material se acumule de maneira mais homogênea, formando barras sólidas e bem definidas.



(a) Sem filtro. (b) Com filtro.

Figura 1.12: Exemplo de uma estrutura na qual aparece o tabuleiro de xadrez.

Uma maneira de prevenir a ocorrência do tabuleiro de xadrez consiste em aumentar o número de nós em cada elemento da malha, com o objetivo de obter melhores aproximações dos deslocamentos. Isso faz com que a dimensão da matriz de rigidez global aumente, tornando a resolução do sistema linear de equilíbrio ainda mais cara. Por esta razão, preferimos utilizar uma outra estratégia, que consiste na aplicação de um operador matemático, denominado *filtro espacial*, cujo objetivo é redistribuir as densidades de material ao redor de cada elemento. A Figura 1.12(b) mostra a mesma estrutura da Figura 1.12(a), obtida com a aplicação de um filtro.

1.5.1 Filtros de densidade

Um filtro espacial de raio $r > 0$ pode ser definido como uma função F que satisfaz algumas propriedades. A aplicação do filtro sobre as densidades é feita por meio do produto de convolução entre as funções F e ρ, ou seja,

$$(F * \rho)(\mathrm{x}) = \int_{\mathbb{R}^3} F(\mathrm{x} - \mathrm{y})\rho(\mathrm{y})\, d\mathrm{y}.$$

Para mais detalhes sobre os filtros em otimização topológica, consulte Bourdin [9].

Na prática, para aplicar o filtro devemos obter a vizinhança de cada elemento e, isto é, os elementos que estão a uma distância menor ou igual a r do elemento e. Denotamos essa vizinhança por $B(e,r)$, indicando que ela está centrada no elemento e e tem raio r. A representação genérica de um filtro espacial no meio discretizado, para cada elemento da malha, é então dada por

$$(F * \rho)_e = \sum_{i \in B(e,r)} \left(\rho_i \int_i F(\mathrm{x} - \mathrm{c}_e)\, d\mathrm{x} \right),$$

em que c_e representa o vetor de coordenadas do centro do elemento e. O filtro produz efeito na densidade do elemento central, levando em conta as densidades dos vizinhos. Além disso, quanto mais distante um elemento $i \in B(e,r)$ estiver do elemento e, menor será sua contribuição à ação do filtro sobre este elemento.

Neste trabalho, utilizamos o filtro da média ponderada das densidades, conforme proposto por Bruns e Tortorelli [12]. A ideia deste tipo de filtro é substituir a densidade de cada elemento e por uma média ponderada das densidades dos elementos na vizinhança $B(e,r)$. Assim, a densidade ρ_e é substituída por

$$\tilde{\rho}_e \equiv \tilde{\rho}_e(\rho) = \sum_{i \in B(e,r)} \frac{\omega_{ei}}{\omega_e} \rho_i,$$

em que

$$\omega_{ei} = \begin{cases} \dfrac{\exp\left(-\dfrac{dist(e,i)^2}{2\left(\frac{r}{3}\right)^2} \right)}{2\pi\left(\frac{r}{3}\right)} & \text{se } dist(e,i) \leq r, \\ 0 & \text{se } dist(e,i) > r, \end{cases}$$

são os *fatores de peso*, $dist(e,i)$ denota a distância euclidiana entre os centros dos elementos e e i, e ω_e é a soma dos fatores de peso de todos os vizinhos do elemento e, ou seja,

$$\omega_e = \sum_{i \in B(e,r)} \omega_{ei}.$$

Note que apenas as densidades dos elementos tais que $dist(e,i) \leq r$ são consideradas na aplicação do filtro no elemento e.

As densidades originais de cada elemento devem ser substituídas pelas densidades filtradas na função objetivo e nas restrições do problema de otimização topológica estrutural. Sendo assim, a função objetivo do problema se torna $f(\tilde{\rho}) = \mathrm{f}^T \mathrm{u}(\tilde{\rho})$, o sistema de equilíbrio global passa a ser dado por $\mathrm{K}(\tilde{\rho})\mathrm{u}(\tilde{\rho}) = \mathrm{f}$ e a restrição de volume se torna

$$c(\tilde{\rho}) = \sum_{e=1}^{n_{el}} v_e \tilde{\rho}_e - V_{max} \leq 0.$$

As principais vantagens desse tipo de filtro são que ele preserva a linearidade da restrição de volume e mantém compatíveis as condições de otimalidade do problema de otimização topológica estrutural. Além disso, a sua aplicação é barata se o raio r não é muito grande e ele permite que calculemos os gradientes da função objetivo e das restrições do problema de forma analítica.

Na literatura, encontramos vários outros tipos de filtros espaciais. Sigmund [45] utiliza um filtro que modifica o vetor gradiente da função objetivo, com base em uma média ponderada das derivadas primeiras da função calculadas em uma vizinhança de cada elemento. Nesse caso, como o gradiente é modificado e a função objetivo não é alterada, as condições de otimalidade do problema ficam incompatíveis, o que dificulta a aplicação do filtro, apesar dele apresentar bons resultados. Bruns [11] apresenta uma estratégia que une a filtragem das densidades com uma alternativa ao modelo SIMP para a penalização das densidades intermediárias, denominada modelo SINH, em referência à utilização da função seno hiperbólico no cálculo das novas densidades. Nesse caso, a penalização é aplicada na restrição de volume, que se torna não linear. Uma vantagem do modelo SINH é a obtenção de resultados com poucas densidades intermediárias.

1.6 Cálculo dos gradientes

Nesta seção, veremos como calcular analiticamente os gradientes da função objetivo e das restrições do problema de otimização topológica estrutural, ou seja, como obter as primeiras derivadas das funções que representam a flexibilidade da estrutura e a restrição de volume. Os gradientes são necessários para a resolução dos problemas por meio da maioria dos métodos de otimização, como é o caso da programação linear sequencial, que descreveremos no próximo capítulo.

Após a aplicação do filtro da média ponderada das densidades, conforme descrito na Subseção 1.5.1, cada densidade ρ_e de um elemento finito e é substituída por

$$\tilde{\rho}_e \equiv \tilde{\rho}_e(\rho) = \sum_{i \in B(e,r)} \frac{\omega_{ei}}{\omega_e} \rho_i.$$

Utilizamos as densidades filtradas na função objetivo e nas restrições do problema de otimização topológica. Com isso, a restrição de volume se torna

$$c(\tilde{\rho}) = \sum_{e=1}^{n_{el}} v_e \tilde{\rho}_e - V_{max} \leq 0.$$

Sua derivada com relação à variável ρ_e é dada por

$$\frac{\partial c(\tilde{\rho})}{\partial \rho_e} = \sum_{i \in B(e,r)} \frac{\partial c(\tilde{\rho})}{\partial \tilde{\rho}_i} \frac{\partial \tilde{\rho}_i}{\partial \rho_e} = \sum_{i \in B(e,r)} v_i \frac{\omega_{ie}}{\omega_i}.$$

Assim, calculamos as componentes da matriz Jacobiana das restrições, que neste caso possui uma única linha contendo o transposto do vetor gradiente da restrição de volume. Se considerarmos que todos os elementos na malha possuem volume fixo, é possível calcular a matriz Jacobiana uma única vez.

Depois de impor as condições de contorno no problema e resolver o sistema linear de equilíbrio, a função objetivo fica sendo

$$f(\tilde{\rho}) = \mathrm{f}^T \mathrm{u}(\tilde{\rho}) = \mathrm{f}^T \mathrm{K}(\tilde{\rho})^{-1} \mathrm{f},$$

cuja derivada com relação à variável ρ_e é dada por

$$\frac{\partial f(\tilde{\rho})}{\partial \rho_e} = \sum_{i \in B(e,r)} \frac{\partial f(\tilde{\rho})}{\partial \tilde{\rho}_i} \frac{\partial \tilde{\rho}_i}{\partial \rho_e} = \sum_{i \in B(e,r)} \frac{\partial f(\tilde{\rho})}{\partial \tilde{\rho}_i} \frac{\omega_{ie}}{\omega_i}.$$

Vejamos como obter as derivadas de f com relação às densidades filtradas $\tilde{\rho}_i$. O termo que depende de $\tilde{\rho}$ na função objetivo é a matriz de rigidez global $\mathrm{K}(\tilde{\rho})$. Como vimos, essa matriz é formada por uma superposição das matrizes de rigidez dos elementos, que recebem a penalização das densidades intermediárias pelo modelo SIMP. Considerando K_e a matriz com mesma dimensão da matriz K, mas formada apenas pelas componentes da matriz de rigidez de um único elemento e, temos que

$$\mathrm{K}(\tilde{\rho}) = \sum_{e=1}^{n_{el}} (E_{min} + (\tilde{\rho}_e)^p (E_0 - E_{min}))\, \mathrm{K}_e.$$

O sistema de equilíbrio global é dado por

$$\mathrm{K}(\tilde{\rho})\mathrm{u}(\tilde{\rho}) = \mathrm{f}.$$

Derivando-o com relação a $\tilde{\rho}_i$, temos

$$\frac{\partial \mathrm{K}(\tilde{\rho})}{\partial \tilde{\rho}_i}\mathrm{u}(\tilde{\rho}) + \mathrm{K}(\tilde{\rho})\frac{\partial \mathrm{u}(\tilde{\rho})}{\partial \tilde{\rho}_i} = 0 \Rightarrow \frac{\partial \mathrm{u}(\tilde{\rho})}{\partial \tilde{\rho}_i} = -\mathrm{K}(\tilde{\rho})^{-1}\frac{\partial \mathrm{K}(\tilde{\rho})}{\partial \tilde{\rho}_i}\mathrm{u}(\tilde{\rho}) \Rightarrow$$

$$\Rightarrow \frac{\partial \mathrm{u}(\tilde{\rho})}{\partial \tilde{\rho}_i} = -\mathrm{K}(\tilde{\rho})^{-1}\left[p(\tilde{\rho}_i)^{p-1}(E_0 - E_{min})\mathrm{K}_i\right]\mathrm{u}(\tilde{\rho}).$$

Dessa forma, obtemos

$$\frac{\partial f(\tilde{\rho})}{\partial \tilde{\rho}_i} = \mathrm{f}^T\frac{\partial \mathrm{u}(\tilde{\rho})}{\partial \tilde{\rho}_i} = \mathrm{u}(\tilde{\rho})^T\mathrm{K}(\tilde{\rho})\frac{\partial \mathrm{u}(\tilde{\rho})}{\partial \tilde{\rho}_i} = -\mathrm{u}(\tilde{\rho})^T\left[p(\tilde{\rho}_i)^{p-1}(E_0 - E_{min})\mathrm{K}_i\right]\mathrm{u}(\tilde{\rho}).$$

Assim, calculamos as derivadas da função objetivo usando

$$\frac{\partial f(\tilde{\rho})}{\partial \rho_e} = -\sum_{i \in B(e,r)} \mathrm{u}(\tilde{\rho})^T\left[p(\tilde{\rho}_i)^{p-1}(E_0 - E_{min})\mathrm{K}_i\right]\mathrm{u}(\tilde{\rho})\frac{\omega_{ie}}{\omega_i},$$

para $e = 1, 2, ..., n_{el}$, e obtemos o vetor gradiente. Na prática, ao invés de expandir a matriz de rigidez do elemento formando a matriz K_e, podemos manter a matriz $\mathrm{K}^{(e)}$ de ordem 24×24 (no caso do elemento linear) e extrair do vetor u as 24 componentes correspondentes aos índices dos graus de liberdade dos nós do elemento e.

Capítulo 2

Programação Linear Sequencial

O problema de otimização topológica estrutural de mínima flexibilidade (1.16), formulado no Capítulo 1, é um problema de otimização não linear que pode ser resolvido com diversos métodos conhecidos na literatura (vide [32, 41]). Entretanto, como esse problema pode possuir muitas variáveis de projeto, os métodos que envolvem segundas derivadas costumam ser evitados, pois apresentam desvantagens como: a dificuldade no cálculo das segundas derivadas de maneira exata, a demanda de muita memória computacional, mesmo quando o cálculo das segundas derivadas é feito de forma aproximada e o custo computacional elevado para resolver subproblemas de programação quadrática. Os métodos mais comumente utilizados para a resolução do problema de otimização topológica são o *método do critério de otimalidade* (consulte [6], Seção 1.2), o *método das assíntotas móveis* [51] e a *programação linear sequencial* [22, 44].

Na implementação feita para obter os resultados deste trabalho, utilizamos uma versão globalmente convergente da programação linear sequencial (PLS), baseada no trabalho de Gomes e Senne [22]. Esse método consiste em resolver uma sequência de subproblemas de programação linear que aproximam o problema de otimização não linear. Ele apresenta resultados eficientes na resolução de problemas de otimização topológica, possui uma implementação simples e utiliza apenas informações das derivadas de primeira ordem, o que é conveniente para os problemas de grande porte.

2.1 Descrição do método

Considere o seguinte problema de otimização não linear com restrições:

$$\begin{aligned} \underset{\mathrm{x}}{\text{Min}} \quad & f(\mathrm{x}) \\ \text{s. a} \quad & c(\mathrm{x}) = 0, \\ & \mathrm{x}_l \leq \mathrm{x} \leq \mathrm{x}_u, \end{aligned} \tag{2.1}$$

em que a função objetivo, $f : \mathbb{R}^n \longrightarrow \mathbb{R}$, e as funções de restrições, $c : \mathbb{R}^n \longrightarrow \mathbb{R}^m$, possuem as primeiras derivadas Lipschitz contínuas, o vetor $\mathrm{x} = \left[\begin{array}{ccc} x_1 & \cdots & x_n \end{array} \right]^T \in \mathbb{R}^n$ contém as variáveis de projeto e os vetores $\mathrm{x}_l, \mathrm{x}_u \in \mathbb{R}^n$ definem os limitantes inferiores e superiores, respectivamente, para as componentes do vetor x. Note que, utilizando variáveis de folga e de excesso, qualquer problema de otimização não linear com restrições de igualdade e de desigualdade pode ser escrito na forma (2.1).

No caso do problema (1.16) de otimização topológica estrutural, as variáveis de projeto são as densidades de cada elemento finito, isto é, $\mathrm{x} = \rho$. O número de variáveis é $n = n_{el}$, que é o número total de elementos na malha. A função objetivo é não linear, dada por $f(\rho) = \mathrm{f}^T\mathrm{K}(\rho)^{-1}\mathrm{f}$, temos uma ($m = 1$) restrição linear de desigualdade,

$$c(\rho) = \sum_{e=1}^{n_{el}} v_e \rho_e - V_{max} \leq 0$$

e as restrições de caixa $0 \leq \rho_e \leq 1$, para $e = 1, 2, ..., n_{el}$, ou seja, $\mathrm{x}_l = \left[\begin{array}{ccc} 0 & \cdots & 0 \end{array}\right]^T$ e $\mathrm{x}_u = \left[\begin{array}{ccc} 1 & \cdots & 1 \end{array}\right]^T$.

Dizemos que $\bar{\mathrm{x}} \in \mathbb{R}^n$ é um ponto *factível* (ou *viável*) para o problema (2.1) se todas as restrições são satisfeitas nele, ou seja, $c_i(\bar{\mathrm{x}}) = 0$, $\forall i = 1, 2, ..., m$, e $\mathrm{x}_l \leq \bar{\mathrm{x}} \leq \mathrm{x}_u$. Se alguma dessas restrições não for satisfeita em $\bar{\mathrm{x}}$, dizemos que ele é um ponto *infactível* (ou *inviável*) para o problema.

Como as funções f e c possuem as primeiras derivadas Lipschitz contínuas, podemos obter aproximações lineares (aproximações de Taylor de primeira ordem) para cada uma delas, em uma vizinhança de um ponto $\mathrm{x}^{(k)} \in \mathbb{R}^n$:

$$\begin{aligned} f\left(\mathrm{x}^{(k)} + \mathrm{s}\right) &\approx f\left(\mathrm{x}^{(k)}\right) + \nabla f\left(\mathrm{x}^{(k)}\right)^T \mathrm{s}, \\ c\left(\mathrm{x}^{(k)} + \mathrm{s}\right) &\approx c\left(\mathrm{x}^{(k)}\right) + \mathrm{A}\left(\mathrm{x}^{(k)}\right) \mathrm{s}, \end{aligned}$$

em que $\nabla f\left(\mathrm{x}^{(k)}\right)$ e $\mathrm{A}\left(\mathrm{x}^{(k)}\right) = \left[\begin{array}{ccc} \nabla c_1\left(\mathrm{x}^{(k)}\right) & \cdots & \nabla c_m\left(\mathrm{x}^{(k)}\right) \end{array}\right]^T$ denotam o vetor gradiente da função objetivo e a matriz Jacobiana das restrições, respectivamente, no ponto $\mathrm{x}^{(k)}$. Assim, dado um ponto $\mathrm{x}^{(k)} \in \mathbb{R}^n$, o problema (2.1) pode ser aproximado pelo problema de programação linear (PL):

$$\begin{aligned} \underset{\mathrm{s}}{\mathrm{Min}} \quad & \nabla f\left(\mathrm{x}^{(k)}\right)^T \mathrm{s} \\ \text{s. a} \quad & \mathrm{A}\left(\mathrm{x}^{(k)}\right) \mathrm{s} + c\left(\mathrm{x}^{(k)}\right) = 0, \\ & \mathrm{x}_l \leq \mathrm{x}^{(k)} + \mathrm{s} \leq \mathrm{x}_u. \end{aligned} \tag{2.2}$$

Os problemas lineares são, em geral, mais simples de resolver e temos métodos bem desenvolvidos para tal, como os métodos Simplex e os métodos de pontos interiores (vide Luenberger *et al.* [32]).

A programação linear sequencial é um método iterativo que consiste na resolução aproximada do problema não linear (2.1) por meio de uma sequência de resoluções de subproblemas lineares na forma (2.2). Para aplicar o método, devemos fornecer um ponto inicial $\mathrm{x}^{(0)}$ e um critério de parada adequado. A cada iteração k, o ponto $\mathrm{x}^{(k)}$ é utilizado para gerar o problema de programação linear, do qual obtemos uma solução $\tilde{\mathrm{s}}$. A solução aproximada do problema original (2.1) é então atualizada por $\mathrm{x}^{(k+1)} = \mathrm{x}^{(k)} + \tilde{\mathrm{s}}$. Esse processo é repetido até que o critério de parada estabelecido seja satisfeito.

Observamos, contudo, que nem sempre é possível resolver o PL (2.2), pois este pode ser infactível, isto é, não possuir nenhum ponto factível. Ademais, as funções lineares utilizadas para definir o problema podem não ser boas aproximações para as funções originais em um ponto $\mathrm{x}^{(k)} + \mathrm{s}$ muito distante de $\mathrm{x}^{(k)}$, ou seja, não podemos exagerar no tamanho do passo. Nas próximas seções, apresentamos estratégias para evitar esses inconvenientes e garantir a convergência global do método.

2.2 Regiões de confiança

Para controlar o tamanho do passo s, introduzimos uma *região de confiança*, que define a região na qual confiamos que as aproximações lineares das funções sejam suficientemente boas. Assim, exigimos que $\|\mathrm{s}\|_\infty \leq \delta$, em que $\delta > 0$ é o raio da região de confiança. Quando adotamos a norma infinito, a região de confiança pode ser combinada com as restrições de caixa do problema linear (2.2). Para tanto, observamos que

$$|s_i| \leq \max_{1\leq j\leq n} |s_j| = \|\mathrm{s}\|_\infty \leq \delta \Rightarrow -\delta \leq s_i \leq \delta, \quad \forall i = 1, 2, ..., n.$$

Além disso, pelas restrições de caixa em (2.2),

$$\mathrm{x}_l \leq \mathrm{x}^{(k)} + \mathrm{s} \leq \mathrm{x}_u \Rightarrow (\mathrm{x}_l)_i - \mathrm{x}_i^{(k)} \leq s_i \leq (\mathrm{x}_u)_i - \mathrm{x}_i^{(k)}, \quad \forall i = 1, 2, ..., n.$$

Então, cada s_i deve satisfazer

$$\max\left\{-\delta,\, (\mathrm{x}_l)_i - \mathrm{x}_i^{(k)}\right\} \leq s_i \leq \min\left\{\delta,\, (\mathrm{x}_u)_i - \mathrm{x}_i^{(k)}\right\}.$$

Portanto, podemos escrever o problema de programação linear (2.2) acrescido da região de confiança como

$$\begin{aligned} \underset{\mathrm{s}}{\text{Min}} \quad & \nabla f\left(\mathrm{x}^{(k)}\right)^T \mathrm{s} \\ \text{s. a} \quad & \mathrm{A}\left(\mathrm{x}^{(k)}\right)\mathrm{s} + c\left(\mathrm{x}^{(k)}\right) = 0, \\ & \mathrm{s}_l \leq \mathrm{s} \leq \mathrm{s}_u, \end{aligned} \tag{2.3}$$

em que $(\mathrm{s}_l)_i = \max\left\{-\delta,\, (\mathrm{x}_l)_i - \mathrm{x}_i^{(k)}\right\}$ e $(\mathrm{s}_u)_i = \min\left\{\delta,\, (\mathrm{x}_u)_i - \mathrm{x}_i^{(k)}\right\}$, para $i = 1, 2, ..., n$.

2.3 Factibilidade do PL

A cada iteração k do método, desejamos obter uma solução do subproblema linear (2.3), que utilizaremos para melhorar a aproximação da solução do problema (2.1). As restrições de caixa combinadas com a região de confiança garantem que o problema (2.3) é limitado. Todavia, ele ainda pode ser infactível, a menos que $\mathrm{x}^{(k)}$ seja factível para o problema original. Uma maneira de verificar a factibilidade do PL é por meio de uma estratégia parecida com a fase 1 do método Simplex de duas fases, na qual resolvemos o seguinte problema:

$$\begin{aligned} \underset{(\mathrm{s},\,\mathrm{z})}{\text{Min}} \quad & \mathrm{e}^T\mathrm{z} \\ \text{s. a} \quad & \mathrm{A}\left(\mathrm{x}^{(k)}\right)\mathrm{s} + c\left(\mathrm{x}^{(k)}\right) + \mathrm{E}\left(\mathrm{x}^{(k)}\right)\mathrm{z} = 0, \\ & \bar{\mathrm{s}}_l \leq \mathrm{s} \leq \bar{\mathrm{s}}_u, \\ & \mathrm{z} \geq 0, \end{aligned} \tag{2.4}$$

em que $\mathrm{e}^T = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} \in \mathbb{R}^q$, $\mathrm{z} \in \mathbb{R}^q$ é um vetor de variáveis artificiais correspondentes às q restrições infactíveis de (2.1) no ponto $\mathrm{x}^{(k)}$, $\mathrm{E}\left(\mathrm{x}^{(k)}\right) \in \mathbb{R}^{m\times q}$ é uma matriz formada por colunas da matriz identidade, $\mathrm{I} \in \mathbb{R}^{m\times m}$, ou de $-\mathrm{I}$, e

$$(\bar{\mathrm{s}}_l)_i = \max\left\{-0{,}8\,\delta;\, (\mathrm{x}_l)_i - \mathrm{x}_i^{(k)}\right\},$$

$$(\bar{\mathrm{s}}_u)_i = \max\left\{0{,}8\,\delta;\, (\mathrm{x}_u)_i - \mathrm{x}_i^{(k)}\right\},$$

para $i = 1, 2, ..., n$. Note que a região de confiança utilizada no problema artificial é um pouco menor, para que o problema (2.3) tenha uma região factível suficientemente grande. A escolha do valor 0,8 é experimental, assim como outros parâmetros do algoritmo.

Se na solução ótima do problema artificial (2.4) tivermos $\mathrm{e}^T\mathrm{z} = 0$, então conseguimos um ponto factível para o PL (2.3). Caso contrário, descobrimos que o problema (2.3) é infactível.

Na otimização topológica estrutural, podemos iniciar o método com um ponto $\mathrm{x}^{(0)}$ que já é factível e, em geral, os demais pontos $\mathrm{x}^{(k)}$ continuam factíveis, especialmente quando a restrição de volume continua linear após a aplicação do filtro. Dessa forma, a estratégia adotada é tentar resolver diretamente o problema linear (2.3) para obter o passo $\tilde{\mathrm{s}}$. Caso o método de resolução do PL detecte que ele não possui solução, então resolvemos o problema artificial (2.4) até obter uma solução s_n e consideramos $\tilde{\mathrm{s}} = \mathrm{s}_n$. Felizmente, em todos os problemas testados, conseguimos resolver diretamente o problema (2.3).

2.4 Critério de aceitação do passo

Após obter o passo $\tilde{\mathrm{s}}$, precisamos verificar se $\mathrm{x}^{(k+1)} = \mathrm{x}^{(k)} + \tilde{\mathrm{s}}$ é uma aproximação melhor do que $\mathrm{x}^{(k)}$ para a solução do problema original (2.1). Queremos que a nova aproximação reduza suficientemente o valor da função objetivo e também precisamos controlar a factibilidade do problema.

Consideramos que a medida de infactibilidade de um ponto $\mathrm{x} \in \mathbb{R}^n$ com relação ao problema (2.1) é dada pela função $\varphi : \mathbb{R}^n \longrightarrow \mathbb{R}$,

$$\varphi(\mathrm{x}) = \frac{1}{2} \left\| c(\mathrm{x}) \right\|_2^2,$$

e a medida de infactibilidade do problema (2.3) é dada pela função $M : \mathbb{R}^n \times \mathbb{R}^n \longrightarrow \mathbb{R}$,

$$M(\mathrm{x}, \mathrm{s}) = \frac{1}{2} \left\| \mathrm{A}(\mathrm{x})\mathrm{s} + c(\mathrm{x}) \right\|_2^2.$$

A escolha do fator 1/2 e da norma euclidiana elevada ao quadrado é feita para simplificar algumas demonstrações de convergência.

A cada iteração do método queremos que o novo ponto obtido reduza os valores da função objetivo f e da função φ, ou seja, ficamos felizes se $f\left(\mathrm{x}^{(k+1)}\right) << f\left(\mathrm{x}^{(k)}\right)$ e $\varphi\left(\mathrm{x}^{(k+1)}\right) << \varphi\left(\mathrm{x}^{(k)}\right)$. Porém, não sabemos ao certo o que ocorre nos casos em que

$$f\left(\mathrm{x}^{(k+1)}\right) < f\left(\mathrm{x}^{(k)}\right) \quad \text{e} \quad \varphi\left(\mathrm{x}^{(k+1)}\right) > \varphi\left(\mathrm{x}^{(k)}\right)$$

ou

$$f\left(\mathrm{x}^{(k+1)}\right) > f\left(\mathrm{x}^{(k)}\right) \quad \text{e} \quad \varphi\left(\mathrm{x}^{(k+1)}\right) < \varphi\left(\mathrm{x}^{(k)}\right),$$

isto é, quando o valor da função objetivo é reduzido, mas a factibilidade é perdida, ou quando o ponto melhora a factibilidade, mas não reduz o valor da função objetivo.

Nesta situação, definimos uma *função de mérito* que combina as duas medidas (otimalidade e factibilidade). Assim, consideramos a função $\psi : \mathbb{R}^n \times \mathbb{R} \longrightarrow \mathbb{R}$, dada por

$$\psi(\mathrm{x}, \theta) = \theta f(\mathrm{x}) + (1 - \theta)\varphi(\mathrm{x}),$$

em que $\theta \in (0, 1]$ é um parâmetro de penalização, cujo objetivo é dizer se a otimalidade terá maior relevância sobre a factibilidade, ou vice-versa.

Agora, a cada iteração queremos reduzir a função de mérito. A aceitação do passo será baseada em uma comparação entre a redução *prevista* pelo modelo linear do problema (2.3) e a redução *real* obtida no problema original (2.1). A redução real de ψ para os pontos $\mathrm{x}^{(k)}$ e $\mathrm{x}^{(k+1)}$ é definida por

$$A_{red} = \theta A_{red}^{opt} + (1 - \theta) A_{red}^{fsb},$$

em que

$$A_{red}^{opt} = f\left(\mathrm{x}^{(k)}\right) - f\left(\mathrm{x}^{(k+1)}\right)$$

é a redução real da função objetivo e

$$A_{red}^{fsb} = \varphi\left(\mathrm{x}^{(k)}\right) - \varphi\left(\mathrm{x}^{(k+1)}\right)$$

é a redução real da infactibilidade. A redução prevista de ψ é definida por

$$P_{red} = \theta P_{red}^{opt} + (1 - \theta) P_{red}^{fsb},$$

em que

$$P_{red}^{opt} = \nabla f\left(\mathrm{x}^{(k)}\right)^T 0 - \nabla f\left(\mathrm{x}^{(k)}\right)^T \tilde{\mathrm{s}} = -\nabla f\left(\mathrm{x}^{(k)}\right)^T \tilde{\mathrm{s}}$$

é a redução prevista da função objetivo e

$$P_{red}^{fsb} = M\left(\mathrm{x}^{(k)}, 0\right) - M\left(\mathrm{x}^{(k)}, \tilde{\mathrm{s}}\right)$$

é a redução prevista da infactibilidade.

Se a redução real for muito pior do que a redução prevista pelo modelo linear, significa que o modelo não fez uma previsão adequada, então não aceitamos o passo. Por exemplo, se a redução real corresponder a menos de 10% da redução prevista, ou seja, se $A_{red} < 0{,}1 P_{red}$, então o passo $\tilde{\mathrm{s}}$ é rejeitado. Neste caso, estamos tentando dar um passo muito grande, já que o modelo prevê uma redução maior da função. Assim, precisamos diminuir o raio da região de confiança, tomando por exemplo

$$\delta_{k+1} = \min\left\{0{,}25 \left\|\tilde{\mathrm{s}}\right\|_\infty ;\, 0{,}1\, \delta_k\right\}.$$

O passo $\tilde{\mathrm{s}}$ é aceito se $A_{red} \geq 0{,}1 P_{red}$. Além disso, se a redução real for boa o suficiente (por exemplo, $A_{red} \geq 0{,}5 P_{red}$), podemos aumentar o raio da região de confiança para tentar acelerar a convergência, tomando

$$\delta_{k+1} = \min\left\{2{,}0\, \delta_k;\, \left\|\mathrm{x}_u - \mathrm{x}_l\right\|_\infty\right\}.$$

Ademais, sempre que o passo é aceito, atualizamos o raio δ verificando se ele é maior do que um raio mínimo $\delta_{min} > 0$, para evitar passos muito pequenos quando estamos longe da solução. Os valores aqui apresentados para os parâmetros utilizados para verificar a aceitação do passo e para aumentar ou diminuir o raio da região de confiança foram obtidos experimentalmente. Aqui, decidimos manter os valores numéricos que utilizamos nos resultados obtidos com a implementação, para simplificar a notação.

O parâmetro de penalização θ da função de mérito também precisa ser atualizado a cada iteração. Para tanto, começamos com $\theta_0 = 1$ e definimos $\theta_{max} = 1$. Na iteração k calculamos

$$\theta_k = \min\left\{\theta_k^{large}, \theta_k^{sup}, \theta_{max}\right\},$$

em que

$$\theta_k^{large} = \left[1 + \frac{N}{(k+1)^{1,1}}\right] \min\{\theta_0, ..., \theta_{k-1}\}, \tag{2.5}$$

$$\theta_k^{sup} = \sup_{\theta\in[0,1]} \left\{P_{red} \geq 0{,}5P_{red}^{fsb}\right\} = \begin{cases} 0{,}5\left(\dfrac{P_{red}^{fsb}}{P_{red}^{fsb} - P_{red}^{opt}}\right); & \text{se } P_{red}^{opt} \leq 0{,}5P_{red}^{fsb}; \\ 1; & \text{caso contrário.} \end{cases} \tag{2.6}$$

Além disso, quando o passo é rejeitado, consideramos $\theta_{max} = \theta_k$, e quando o passo é aceito, tomamos $\theta_{max} = 1$. A constante $N \geq 0$, utilizada para calcular θ_k^{large}, deve ser ajustada para permitir que θ tenha um decréscimo não monótono ao longo das iterações. Na implementação feita neste trabalho, consideramos $N = 10^6$.

2.5 Critério de parada

No método de programação linear sequencial, geramos uma sequência de iterandos $\mathrm{x}^{(k)}$, que se aproximam cada vez mais de um ponto estacionário do problema (2.1). Como o valor da função objetivo decresce, esse ponto tende a ser um minimizador local. Na prática, precisamos parar em algum momento, seguindo algum critério adequado.

Um critério bastante simples é baseado no tamanho do passo, no qual interrompemos o algoritmo quando $\|\tilde{\mathrm{s}}\|_\infty < \varepsilon_s$ por três (ou mais) vezes seguidas, sendo $\varepsilon_s > 0$ uma tolerância preestabelecida. A obtenção de passos pequenos consecutivos indica que o método não consegue caminhar e podemos estar próximos de uma solução. Outro critério parecido com esse é baseado no decréscimo da função objetivo, no qual paramos quando $\left|f(\mathrm{x}^{(k)}) - f(\mathrm{x}^{(k-1)})\right| < \varepsilon_f$ por três iterações consecutivas.

Além dos critérios citados acima, utilizamos um critério de parada mais robusto, baseado nas condições Karush-Kuhn-Tucker (KKT) do problema (2.1) (consulte Ribeiro e Karas [41], Seção 7.2), similar ao que é proposto em [23]. Seja $P_X(\mathrm{y})$ a projeção ortogonal de um vetor y no conjunto $X = \{\mathrm{x} \in \mathbb{R}^n \mid \mathrm{x}_l \leq \mathrm{x} \leq \mathrm{x}_u\}$ das restrições de caixa do problema. Considere $\nabla\mathcal{L}\left(\mathrm{x}^{(k)}, \lambda^{(k)}\right)$ o gradiente do Lagrangiano associado apenas às restrições de igualdade, em que $\lambda^{(k)}$ é o vetor dos multiplicadores de Lagrange associados a essas restrições. Uma aproximação para esses multiplicadores nos é fornecida na resolução do subproblema de programação linear. Assim, a cada iteração obtemos o vetor

$$g_P\left(\mathrm{x}^{(k)}\right) = P_X\left(\mathrm{x}^{(k)} - \nabla\mathcal{L}\left(\mathrm{x}^{(k)}, \lambda^{(k)}\right)\right) - \mathrm{x}^{(k)}$$

e paramos o algoritmo quando $\left\|g_P\left(\mathrm{x}^{(k)}\right)\right\|_\infty < \varepsilon_g$ por três iterações consecutivas. Cada componente i do vetor $g_P\left(\mathrm{x}^{(k)}\right)$ pode ser calculada por

$$g_P\left(\mathrm{x}^{(k)}\right)_i = \min\left\{(\mathrm{x}_u)_i, \max\left\{(\mathrm{x}_l)_i, \mathrm{x}_i^{(k)} - \nabla_i\mathcal{L}\left(\mathrm{x}^{(k)}, \lambda^{(k)}\right)\right\}\right\} - \mathrm{x}_i^{(k)},$$

para $i = 1, 2, ..., n$.

Na implementação, consideramos $\varepsilon_s = 10^{-4}$, $\varepsilon_f = 5 \times 10^{-2}$, $\varepsilon_g = 10^{-3}$ e combinamos ambos os critérios, de modo que o algoritmo para quando as tolerâncias ε_f e ε_g são atingidas ao mesmo tempo por três iterações seguidas ou quando a tolerância ε_s é atingida por três iterações seguidas. Ademais, colocamos um limite de 500 iterações como critério de parada adicional. Na prática, o critério do gradiente projetado junto com o decréscimo da função foi o primeiro a ser atingido para todos os resultados apresentados.

2.6 Algoritmo

Dados um ponto inicial $\mathrm{x}^{(0)} \in \mathbb{R}^n$ com $\mathrm{x}_l \leq \mathrm{x}^{(0)} \leq \mathrm{x}_u$, um raio da região de confiança inicial $\delta_0 \geq \delta_{min} > 0$, $\theta_0 = \theta_{max} = 1$ e $k = 0$, o Algoritmo 1 resolve o problema de otimização não linear (2.1) utilizando a programação linear sequencial.

Algoritmo 1 Programação Linear Sequencial (PLS)

enquanto algum critério de parada não é satisfeito **faça**
  encontre a solução $\tilde{\mathrm{s}}$ do problema linear (2.3),

$$\begin{aligned} \underset{\mathrm{s}}{\text{Min}} \quad & \nabla f\left(\mathrm{x}^{(k)}\right)^T \mathrm{s} \\ \text{s. a} \quad & \mathrm{A}\left(\mathrm{x}^{(k)}\right) \mathrm{s} + c\left(\mathrm{x}^{(k)}\right) = 0, \\ & \mathrm{s}_l \leq \mathrm{s} \leq \mathrm{s}_u. \end{aligned}$$

  se o problema (2.3) for infactível **então**
    encontre a solução s_n do problema artificial (2.4),

$$\begin{aligned} \underset{(\mathrm{s},\,\mathrm{z})}{\text{Min}} \quad & \mathrm{e}^T \mathrm{z} \\ \text{s. a} \quad & \mathrm{A}\left(\mathrm{x}^{(k)}\right) \mathrm{s} + c\left(\mathrm{x}^{(k)}\right) + \mathrm{E}\left(\mathrm{x}^{(k)}\right) \mathrm{z} = 0, \\ & \bar{\mathrm{s}}_l \leq \mathrm{s} \leq \bar{\mathrm{s}}_u, \\ & \mathrm{z} \geq 0. \end{aligned}$$

    $\tilde{\mathrm{s}} \longleftarrow \mathrm{s}_n$.
  fim
  atualize $\theta_k = \min\left\{\theta_k^{large}, \theta_k^{sup}, \theta_{max}\right\}$, utilizando (2.5) e (2.6).
  calcule as reduções real (A_{red}) e prevista (P_{red}) da função de mérito.
  se $A_{red} < 0{,}1 P_{red}$ (o passo é rejeitado) **então**
    $\delta_k \longleftarrow \min\left\{0{,}25 \left\|\tilde{\mathrm{s}}\right\|_\infty ;\, 0{,}1\, \delta_k\right\}$.
    $\theta_{max} \longleftarrow \theta_k$.
  senão
    $\mathrm{x}^{(k)} \longleftarrow \mathrm{x}^{(k)} + \tilde{\mathrm{s}}$.
    se $A_{red} \geq 0{,}5 P_{red}$ **então**
      $\delta_k \longleftarrow \min\left\{2{,}0\, \delta_k;\, \left\|\mathrm{x}_u - \mathrm{x}_l\right\|_\infty\right\}$.
    fim
    $\delta_k \longleftarrow \max\left\{\delta_k, \delta_{min}\right\}$.
    atualize $\mathrm{A}\left(\mathrm{x}^{(k)}\right)$ e $\nabla f\left(\mathrm{x}^{(k)}\right)$.
    $\theta_{max} \longleftarrow 1$.
    $k \longleftarrow k + 1$.
  fim
fim

Observe que contamos uma nova iteração externa apenas quando o passo $\tilde{\mathrm{s}}$ é aceito. Entretanto, mesmo que isso não ocorra, temos que resolver o problema linear e calcular o valor da função objetivo, o que envolve resolver um sistema linear no caso da otimização topológica. Já o cálculo dos gradientes é feito a cada iteração externa.

As demonstrações de que o algoritmo está bem definido e dos resultados de convergência podem ser vistas com detalhes em [22].

2.7 Fluxograma da otimização topológica com a PLS

Na Figura 2.1, apresentamos um fluxograma simplificado do processo de resolução do problema de otimização topológica estrutural com a programação linear sequencial. A primeira etapa consiste em discretizar o domínio no qual a estrutura estará contida, formando a malha de elementos finitos, definir os apoios, o vetor de cargas nodais, a quantidade e a numeração dos elementos, dos nós e dos graus de liberdade. Em seguida, as variáveis de projeto (distribuição das densidades de material antes da aplicação do filtro) e outros dados iniciais são fornecidos.

A etapa de pré-filtragem corresponde à obtenção das vizinhanças de cada elemento finito e ao cálculo dos fatores de peso, necessários para a aplicação do filtro de densidades. Na preparação da matriz de rigidez global, encontramos os índices das linhas e colunas dessa matriz que podem conter entradas não nulas e calculamos a matriz de rigidez dos elementos. Essas etapas podem ser realizadas uma única vez. Antes de iniciar o algoritmo da PLS, também precisamos calcular os valores iniciais das densidades filtradas, dos deslocamentos, da função objetivo (flexibilidade) e dos gradientes.

As próximas etapas do processo correspondem ao algoritmo da programação linear sequencial. Primeiro, verificamos se a solução inicial já é ótima, seguindo os critérios de parada descritos na Seção 2.5. Se a solução for ótima, chegamos ao fim. Caso contrário, aproximamos o problema original por um problema de programação linear e resolvemos o PL obtendo um passo com o qual atualizamos as variáveis de projeto. Em seguida, aplicamos o filtro para obter as novas densidades, com as quais podemos montar a matriz de rigidez global. Então, resolvemos o sistema linear de equilíbrio para calcular os novos deslocamentos e o novo valor da função objetivo. Com essas informações, calculamos as reduções real e prevista da função de mérito e verificamos se o passo será aceito, seguindo os critérios da Seção 2.4. Se o passo for rejeitado, retomamos os valores das variáveis de projeto anteriores e reduzimos o raio da região de confiança. Caso contrário, calculamos o novo vetor gradiente da função objetivo. Aceitando o passo ou não, precisamos atualizar a região de confiança e outros parâmetros, conforme o Algoritmo 1. Depois, tornamos a verificar se a solução atual é ótima. Esse processo se repete até que o critério de parada seja satisfeito e tenhamos a distribuição ótima de material, formando a estrutura desejada.

Observamos que, na prática, chamamos de solução ótima um ponto que satisfaz o critério de parada com as tolerâncias preestabelecidas e não necessariamente o minimizador global do problema. A função objetivo do problema de otimização topológica possui muitos mínimos locais, de modo que qualquer alteração nos parâmetros do algoritmo pode causar diferenças nos resultados. Além disso, resolvemos de maneira aproximada os PLs e os sistemas lineares. Dessa forma, a solução obtida é apenas uma aproximação para a melhor distribuição de material.

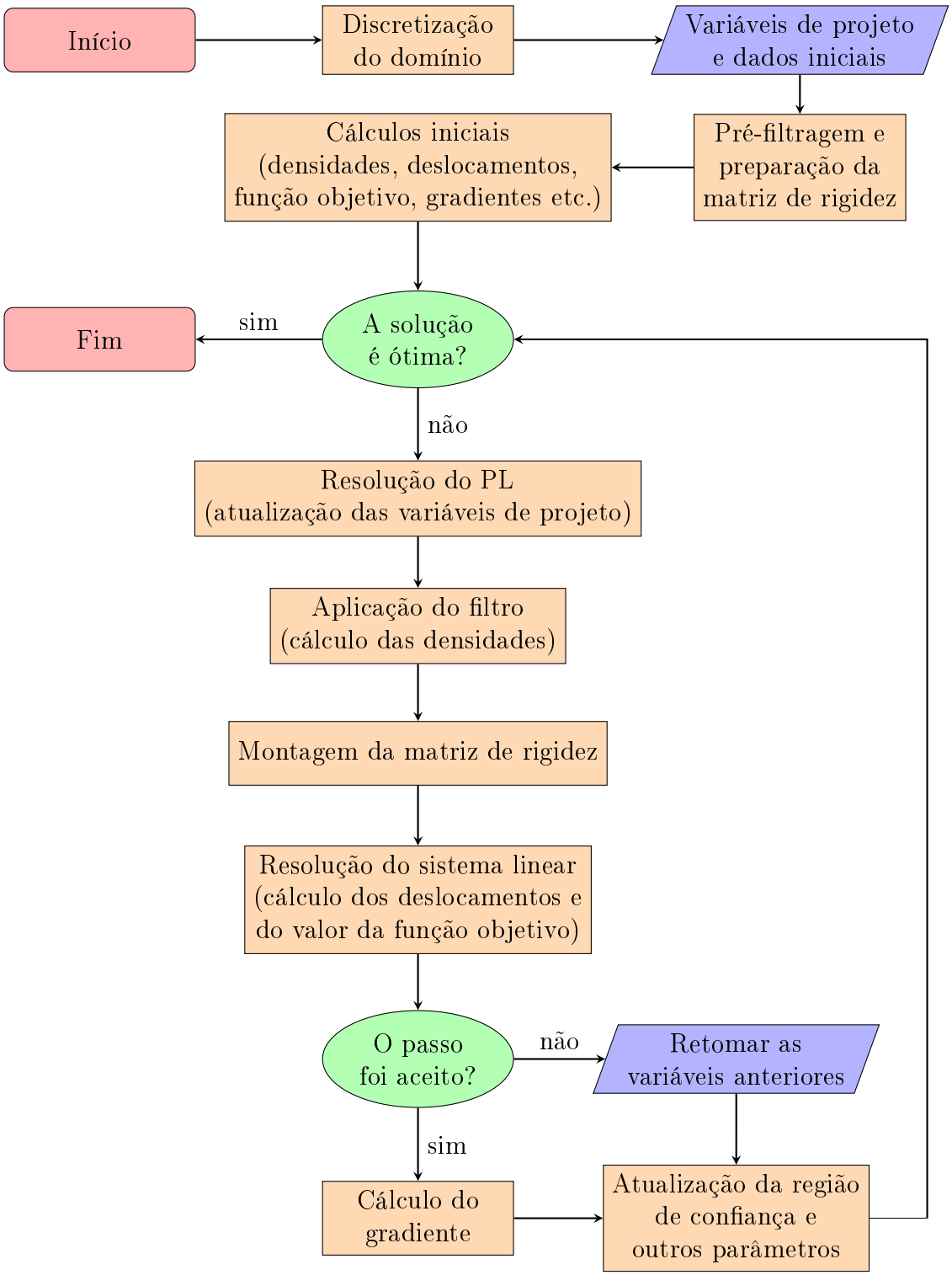


Figura 2.1: Fluxograma da otimização topológica com a programação linear sequencial.

Capítulo 3

Multigrid

Durante a resolução dos problemas de otimização topológica estrutural, no cálculo da função objetivo, precisamos lidar com sistemas lineares muito grandes e que possuem a matriz de coeficientes esparsa. Os métodos de resolução iterativos costumam ser mais eficientes nesses casos. Neste trabalho, estudamos o método *multigrid*, que possui boa aplicação na resolução dos sistemas lineares provenientes de problemas com discretizações, em particular da otimização topológica. Antes de descrevê-lo, apresentamos um breve estudo sobre alguns métodos iterativos mais clássicos (vide Watkins [56], Seção 7.2), que são fundamentais para o desenvolvimento do *multigrid*.

3.1 Métodos iterativos estacionários para resolução de sistemas lineares e o princípio da suavização

Nosso objetivo é resolver um sistema de equações lineares, que denotaremos $\mathrm{Ax} = \mathrm{b}$. Em geral, estamos interessados no caso em que a matriz $\mathrm{A} : n \times n$ é esparsa, simétrica e definida positiva, mas, para a descrição dos métodos desta seção, pode-se considerar apenas que $\mathrm{A} : n \times n$ é qualquer matriz não-singular com todas as entradas da diagonal principal não nulas, ou seja, $a_{ii} \neq 0$, para todo $i = 1, 2, ..., n$.

Os métodos iterativos apresentados a seguir também são conhecidos como *métodos estacionários*, o que significa que a forma como atualizamos a aproximação para a solução é sempre a mesma em todas as iterações. Veremos que cada método pode ser completamente descrito pela maneira como um iterando $\mathrm{x}^{(k)}$ é utilizado para gerar o próximo iterando $\mathrm{x}^{(k+1)}$. Além disso, cada método pode ser descrito na forma matricial

$$\mathrm{x}^{(k+1)} = \mathrm{x}^{(k)} + \mathrm{M}^{-1}\left(\mathrm{b} - \mathrm{Ax}^{(k)}\right) = \mathrm{x}^{(k)} + \mathrm{M}^{-1}\mathrm{r}^{(k)},$$

em que M é denominada *matriz de iteração* e $\mathrm{r}^{(k)} = \mathrm{b} - \mathrm{Ax}^{(k)}$ é denominado *resíduo* da aproximação $\mathrm{x}^{(k)}$.

3.1.1 O método de Jacobi

A ideia do método de Jacobi é utilizar a i-ésima equação do sistema linear para corrigir a aproximação da i-ésima incógnita, para cada $i = 1, 2, ..., n$. A i-ésima equação

do sistema Ax = b é dada por

$$\sum_{j=1}^{n} a_{ij}x_j = b_i.$$

Isolando nesta equação a incógnita x_i, obtemos

$$x_i = \frac{1}{a_{ii}}\left(b_i - \sum_{j\neq i} a_{ij}x_j\right),$$

o que pode ser feito pois supomos que $a_{ii} \neq 0$, para todo $i = 1, 2, ..., n$. Dada uma aproximação $\mathrm{x}^{(k)}$ para a solução do sistema, se substituirmos x por $\mathrm{x}^{(k)}$ na igualdade acima, em geral ela deixa de ser válida. Definimos então $x_i^{(k+1)}$ como sendo o valor que tornaria a i-ésima igualdade verdadeira:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j\neq i} a_{ij}x_j^{(k)}\right). \tag{3.1}$$

Fazendo isso para $i = 1, 2, ..., n$, obtemos o próximo iterando $\mathrm{x}^{(k+1)}$.

Note que a atualização $x_i^{(k)} \longrightarrow x_i^{(k+1)}$ corrige a i-ésima equação, mas também afeta todas as outras equações nas quais aparece a i-ésima incógnita. Desse modo, a nova aproximação $\mathrm{x}^{(k+1)}$ ainda não é uma solução exata do sistema. A esperança é que a sequência de iterandos $\mathrm{x}^{(k)}$, $k = 0, 1, 2, ...$, convirja para a solução exata. Na prática, paramos o método em alguma iteração na qual a aproximação seja suficientemente boa, seguindo algum critério de parada.

Seja D a matriz diagonal cuja diagonal principal é a mesma da matriz A. As equações da forma (3.1) para $i = 1, 2, ..., n$ podem ser escritas da seguinte maneira:

$$\begin{aligned}
a_{11}x_1^{(k+1)} &= b_1 - \left(a_{12}x_2^{(k)} + a_{13}x_3^{(k)} + \cdots + a_{1n}x_n^{(k)}\right),\\
a_{22}x_2^{(k+1)} &= b_2 - \left(a_{21}x_1^{(k)} + a_{23}x_3^{(k)} + \cdots + a_{2n}x_n^{(k)}\right),\\
&\vdots\\
a_{nn}x_n^{(k+1)} &= b_n - \left(a_{n1}x_1^{(k)} + a_{n2}x_2^{(k)} + \cdots + a_{n,n-1}x_{n-1}^{(k)}\right).
\end{aligned}$$

Na forma matricial, temos

$$\begin{bmatrix} a_{11} & & & \\ & a_{22} & & \\ & & \ddots & \\ & & & a_{nn} \end{bmatrix}\begin{bmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \\ \vdots \\ x_n^{(k+1)} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} - \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ a_{21} & 0 & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & 0 \end{bmatrix}\begin{bmatrix} x_1^{(k)} \\ x_2^{(k)} \\ \vdots \\ x_n^{(k)} \end{bmatrix}$$

$$\Leftrightarrow \mathrm{Dx}^{(k+1)} = \mathrm{b} - (\mathrm{A} - \mathrm{D})\mathrm{x}^{(k)}$$

$$\Leftrightarrow \mathrm{x}^{(k+1)} = \mathrm{D}^{-1}\left[\mathrm{b} - (\mathrm{A} - \mathrm{D})\mathrm{x}^{(k)}\right].$$

Ou, equivalentemente,

$$\mathrm{x}^{(k+1)} = \mathrm{x}^{(k)} + \mathrm{D}^{-1}\left(\mathrm{b} - \mathrm{Ax}^{(k)}\right) = \mathrm{x}^{(k)} + \mathrm{D}^{-1}\mathrm{r}^{(k)},$$

em que $\mathrm{r}^{(k)}$ é o resíduo da aproximação $\mathrm{x}^{(k)}$. Observe que D é invertível, pois $a_{ii} \neq 0$, para todo $i = 1, 2, ..., n$. Assim, a matriz de iteração do método de Jacobi é $\mathrm{M} = \mathrm{D}$.

O processo de correção de uma equação modificando uma das variáveis muitas vezes é chamado de *relaxação*. Dizemos que o método de Jacobi realiza relaxação simultânea, pois todas as equações do sistema podem ser relaxadas ao mesmo tempo. Em muitos casos, a convergência do método pode ser acelerada introduzindo um parâmetro de relaxação $\omega \in (0, 1]$ e substituindo a fórmula (3.1) por

$$x_i^{(k+1)} = \frac{\omega}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right) + (1 - \omega) x_i^{(k)},$$

para $i = 1, 2, ..., n$. Na forma matricial, temos

$$\begin{aligned} \mathrm{x}^{(k+1)} &= \omega \mathrm{D}^{-1} \left[\mathrm{b} - (\mathrm{A} - \mathrm{D}) \mathrm{x}^{(k)} \right] + (1 - \omega) \mathrm{x}^{(k)} \\ &= \omega \mathrm{D}^{-1} \mathrm{b} - \omega \mathrm{D}^{-1} \mathrm{A} \mathrm{x}^{(k)} + \omega \mathrm{D}^{-1} \mathrm{D} \mathrm{x}^{(k)} + \mathrm{x}^{(k)} - \omega \mathrm{x}^{(k)} \\ &= \omega \mathrm{D}^{-1} \mathrm{b} - \omega \mathrm{D}^{-1} \mathrm{A} \mathrm{x}^{(k)} + \mathrm{x}^{(k)}. \end{aligned}$$

Ou, equivalentemente,

$$\mathrm{x}^{(k+1)} = \mathrm{x}^{(k)} + \omega \mathrm{D}^{-1} \left(\mathrm{b} - \mathrm{A} \mathrm{x}^{(k)} \right) = \mathrm{x}^{(k)} + \omega \mathrm{D}^{-1} \mathrm{r}^{(k)}.$$

Assim, a matriz de iteração do método de Jacobi com relaxação é $\mathrm{M} = \frac{1}{\omega} \mathrm{D}$. Note que, para $\omega = 1$, recuperamos o método de Jacobi original.

3.1.2 O método de Gauss-Seidel e o SOR

No método de Jacobi, descrito anteriormente, utilizamos a i-ésima equação do sistema linear para atualizar a aproximação da i-ésima incógnita. Todas as incógnitas podem ser atualizadas simultaneamente, pois apenas os valores da aproximação atual são utilizados nas atualizações. A ideia do método de Gauss-Seidel é utilizar os novos valores das incógnitas assim que eles forem sendo obtidos, por meio de um processo sequencial.

A partir de uma aproximação $\mathrm{x}^{(k)}$, primeiro utilizamos a equação 1 para calcular $x_1^{(k+1)}$, depois usamos a equação 2 para calcular $x_2^{(k+1)}$ e assim por diante. Quando chegarmos na equação i, já teremos calculado $x_1^{(k+1)}, x_2^{(k+1)}, ..., x_{i-1}^{(k+1)}$. No cálculo de $x_i^{(k+1)}$ podemos utilizar esses novos valores ou os antigos, $x_1^{(k)}, x_2^{(k)}, ..., x_{i-1}^{(k)}$. O método de Jacobi utiliza os valores antigos e o de Gauss-Seidel utiliza os novos. Então, uma iteração do método de Gauss-Seidel é descrita da seguinte forma:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right), \tag{3.2}$$

para $i = 1, 2, ..., n$.

A ordem em que as atualizações das incógnitas são feitas é importante. Por exemplo, se fossem realizadas na ordem reversa (decrescente), ou seja, começando na equação n e indo até a equação 1, a iteração teria um resultado diferente. Vamos assumir que a iteração de Gauss-Seidel é realizada na ordem padrão (crescente), $i = 1, 2, ..., n$, a não ser que se diga o contrário.

Considere D a matriz diagonal cuja diagonal é a mesma de A, L a matriz formada pela parte estritamente triangular inferior de A e U a matriz formada pela parte estritamente triangular superior de A. Dessa forma, A = D+L+U. As equações da forma (3.2) para $i = 1, 2, ..., n$ podem ser escritas como

$$\begin{aligned}
a_{11}x_1^{(k+1)} &= b_1 - \left(a_{12}x_2^{(k)} + \cdots + a_{1n}x_n^{(k)}\right), \\
a_{22}x_2^{(k+1)} &= b_2 - \left(a_{21}x_1^{(k+1)}\right) - \left(a_{23}x_3^{(k)} + \cdots + a_{2n}x_n^{(k)}\right), \\
a_{33}x_3^{(k+1)} &= b_3 - \left(a_{31}x_1^{(k+1)} + a_{32}x_2^{(k+1)}\right) - \left(a_{34}x_4^{(k)} + \cdots + a_{3n}x_n^{(k)}\right), \\
&\vdots \\
a_{nn}x_n^{(k+1)} &= b_n - \left(a_{n1}x_1^{(k+1)} + \cdots + a_{n,n-1}x_{n-1}^{(k+1)}\right).
\end{aligned}$$

Na forma matricial, temos

$$\begin{bmatrix} a_{11} & & & \\ & a_{22} & & \\ & & \ddots & \\ & & & a_{nn} \end{bmatrix} \begin{bmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \\ \vdots \\ x_n^{(k+1)} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} - \begin{bmatrix} 0 & 0 & \cdots & 0 \\ a_{21} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & 0 \end{bmatrix} \begin{bmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \\ \vdots \\ x_n^{(k+1)} \end{bmatrix} -$$

$$- \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ 0 & 0 & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} x_1^{(k)} \\ x_2^{(k)} \\ \vdots \\ x_n^{(k)} \end{bmatrix}$$

$$\Leftrightarrow \mathrm{D}\mathrm{x}^{(k+1)} = \mathrm{b} - \mathrm{L}\mathrm{x}^{(k+1)} - \mathrm{U}\mathrm{x}^{(k)}.$$

Juntando os termos que envolvem $\mathrm{x}^{(k+1)}$, obtemos

$$(\mathrm{D} + \mathrm{L})\mathrm{x}^{(k+1)} = \mathrm{b} - \mathrm{U}\mathrm{x}^{(k)}$$

$$\Leftrightarrow \mathrm{x}^{(k+1)} = (\mathrm{D} + \mathrm{L})^{-1}\left(\mathrm{b} - \mathrm{U}\mathrm{x}^{(k)}\right).$$

Além disso, note que U = A − (D + L). Assim,

$$\mathrm{x}^{(k+1)} = \mathrm{x}^{(k)} + (\mathrm{D} + \mathrm{L})^{-1}\left(\mathrm{b} - \mathrm{A}\mathrm{x}^{(k)}\right) = \mathrm{x}^{(k)} + (\mathrm{D} + \mathrm{L})^{-1}\mathrm{r}^{(k)}.$$

Logo, a matriz de iteração do método de Gauss-Seidel é M = D + L, que é uma matriz formada pela parte triangular inferior da matriz A.

Diferentemente do método de Jacobi, que realiza relaxação simultânea nas equações do sistema, o método de Gauss-Seidel realiza relaxação sucessiva, pois caminha de equação em equação, relaxando uma após a outra. Muitas vezes, a convergência do método pode ser acelerada por meio da sobre-relaxação, na qual escolhemos um fator $\omega > 1$ e substituímos (3.2) por

$$x_i^{(k+1)} = x_i^{(k)} + \frac{\omega}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij}x_j^{(k)}\right),$$

para $i = 1, 2, ..., n$, dando origem ao método SOR (do inglês *Successive Over-Relaxation*).

Na forma matricial, temos

$$\mathrm{x}^{(k+1)} = \mathrm{x}^{(k)} + \omega \mathrm{D}^{-1}\left(\mathrm{b} - \mathrm{L}\mathrm{x}^{(k+1)} - (\mathrm{D}+\mathrm{U})\mathrm{x}^{(k)}\right).$$

Equivalentemente,

$$\begin{aligned}
&\mathrm{D}\mathrm{x}^{(k+1)} = \mathrm{D}\mathrm{x}^{(k)} + \omega\mathrm{b} - \omega\mathrm{L}\mathrm{x}^{(k+1)} - \omega(\mathrm{D}+\mathrm{U})\mathrm{x}^{(k)} \\
\Leftrightarrow\ &\frac{1}{\omega}\mathrm{D}\mathrm{x}^{(k+1)} + \mathrm{L}\mathrm{x}^{(k+1)} = \mathrm{b} + \frac{1}{\omega}\mathrm{D}\mathrm{x}^{(k)} - \mathrm{D}\mathrm{x}^{(k)} - \mathrm{U}\mathrm{x}^{(k)} \\
\Leftrightarrow\ &\left(\frac{1}{\omega}\mathrm{D}+\mathrm{L}\right)\mathrm{x}^{(k+1)} = \mathrm{b} + \left[\left(\frac{1-\omega}{\omega}\right)\mathrm{D} - \mathrm{U}\right]\mathrm{x}^{(k)} \\
\Leftrightarrow\ &\mathrm{x}^{(k+1)} = \left(\frac{1}{\omega}\mathrm{D}+\mathrm{L}\right)^{-1}\left\{\mathrm{b} + \left[\left(\frac{1-\omega}{\omega}\right)\mathrm{D} - \mathrm{U}\right]\mathrm{x}^{(k)}\right\}.
\end{aligned}$$

Lembrando que $\mathrm{U} = \mathrm{A} - \mathrm{D} - \mathrm{L}$, obtemos

$$\mathrm{x}^{(k+1)} = \mathrm{x}^{(k)} + \left(\frac{1}{\omega}\mathrm{D}+\mathrm{L}\right)^{-1}\left(\mathrm{b} - \mathrm{A}\mathrm{x}^{(k)}\right) = \mathrm{x}^{(k)} + \left(\frac{1}{\omega}\mathrm{D}+\mathrm{L}\right)^{-1}\mathrm{r}^{(k)}.$$

Logo, a matriz de iteração do método SOR é $\mathrm{M} = \frac{1}{\omega}\mathrm{D} + \mathrm{L}$. Note que para $\omega = 1$ recuperamos o método de Gauss-Seidel. Também podemos utilizar $\omega < 1$, caso em que teríamos uma sub-relaxação. Além disso, verifica-se que o método diverge para $\omega \geq 2$ e, portanto, consideramos $\omega \in (0, 2)$.

Como dito anteriormente, a ordem em que as atualizações das incógnitas são feitas no método de Gauss-Seidel é importante. Dependendo do problema, pode ser que as propriedades do método melhorem ao utilizarmos uma ordenação diferente da padrão. A ordenação *red-black*, por exemplo, é representada na Figura 3.1, em que os pontos são rotulados de forma alternada como vermelho (*red, r*) e preto (*black, b*), no padrão de um tabuleiro de xadrez.

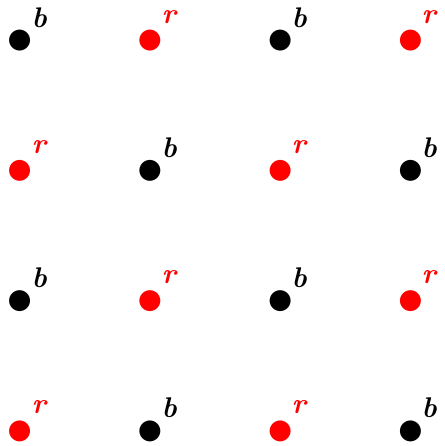


Figura 3.1: Ordenação *red-black* do método de Gauss-Seidel.

Separamos os índices das equações (e das incógnitas) do sistema linear em dois subconjuntos (vermelho e preto). Todas as incógnitas do sistema relacionadas aos pontos vermelhos são atualizadas primeiro e, em seguida, atualizamos as incógnitas relacionadas aos pontos pretos. Observe que os vizinhos mais próximos (na horizontal e na vertical)

de cada ponto vermelho são sempre pontos pretos e vice-versa. Logo, a correção das incógnitas do tipo vermelho deve depender apenas dos valores das incógnitas do tipo preto e vice-versa. Isso significa que a relaxação pode ser feita de forma simultânea em todas as equações do tipo vermelho e, uma vez obtidos os novos valores das incógnitas vermelhas, aplicamos a relaxação de forma simultânea em todas as equações do tipo preto. Assim, o método de Gauss-Seidel com a ordenação *red-black* se torna paralelizável.

Segundo Trottenberg *et al.* [52] (Subseção 2.1.3), no contexto do *multigrid*, as propriedades de suavização do método de Gauss-Seidel podem melhorar com a ordenação *red-black*. Entretanto, para diversos sistemas lineares de grande porte que surgem em aplicações, pode ser difícil ou até impossível separar as equações e incógnitas na forma *red-black*. Nesses casos, existem ainda estratégias de ordenação *multicolor*, que envolvem mais do que duas cores na separação das equações.

Neste trabalho, testamos o método de Gauss-Seidel apenas com a ordenação padrão (crescente), bem como a ordenação reversa (decrescente), que gera o método de Gauss-Seidel simétrico (e o SSOR). Essa última versão, que será apresentada a seguir, pode ser mais efetiva para os problemas de otimização topológica estrutural, nos quais a matriz do sistema linear é simétrica.

3.1.3 O método de Gauss-Seidel simétrico e o SSOR

Podemos aplicar o método de Gauss-Seidel na ordem reversa. Nesse caso, começamos utilizando a equação n para calcular $\mathrm{x}_n^{(k)}$, depois utilizamos a equação $n-1$ para calcular $\mathrm{x}_{n-1}^{(k)}$ e assim por diante. Na forma matricial, temos

$$(\mathrm{D}+\mathrm{U})\mathrm{x}^{(k+1)} = \mathrm{b} - \mathrm{L}\mathrm{x}^{(k)}.$$

O método de Gauss-Seidel simétrico consiste em aplicar uma iteração de Gauss-Seidel na ordem padrão, seguida de uma iteração na ordem reversa.

Assim, a partir de uma aproximação $\mathrm{x}^{(k)}$, aplicamos Gauss-Seidel padrão obtendo um ponto intermediário $\mathrm{x}^{\left(k+\frac{1}{2}\right)}$:

$$\mathrm{x}^{\left(k+\frac{1}{2}\right)} = (\mathrm{D}+\mathrm{L})^{-1}(\mathrm{b} - \mathrm{U}\mathrm{x}^{(k)}).$$

Em seguida, aplicamos Gauss-Seidel reverso, transformando $\mathrm{x}^{\left(k+\frac{1}{2}\right)}$ em $\mathrm{x}^{(k+1)}$:

$$\begin{aligned}(\mathrm{D}+\mathrm{U})\mathrm{x}^{(k+1)} &= \mathrm{b} - \mathrm{L}\mathrm{x}^{\left(k+\frac{1}{2}\right)} \\ &= \mathrm{b} - \mathrm{L}(\mathrm{D}+\mathrm{L})^{-1}(\mathrm{b} - \mathrm{U}\mathrm{x}^{(k)}) \\ &= \left[\mathrm{I} - \mathrm{L}(\mathrm{D}+\mathrm{L})^{-1}\right]\mathrm{b} + \mathrm{L}(\mathrm{D}+\mathrm{L})^{-1}\mathrm{U}\mathrm{x}^{(k)}.\end{aligned}$$

Observando, então, que

$$\begin{aligned}\mathrm{I} - \mathrm{L}(\mathrm{D}+\mathrm{L})^{-1} &= (\mathrm{D}+\mathrm{L})(\mathrm{D}+\mathrm{L})^{-1} - \mathrm{L}(\mathrm{D}+\mathrm{L})^{-1} \\ &= (\mathrm{D}+\mathrm{L}-\mathrm{L})(\mathrm{D}+\mathrm{L})^{-1} = \mathrm{D}(\mathrm{D}+\mathrm{L})^{-1},\end{aligned}$$

obtemos

$$\begin{aligned}&(\mathrm{D}+\mathrm{U})\mathrm{x}^{(k+1)} = \mathrm{L}(\mathrm{D}+\mathrm{L})^{-1}\mathrm{U}\mathrm{x}^{(k)} + \mathrm{D}(\mathrm{D}+\mathrm{L})^{-1}\mathrm{b} \\ \Leftrightarrow\ &(\mathrm{D}+\mathrm{L})\mathrm{D}^{-1}(\mathrm{D}+\mathrm{U})\mathrm{x}^{(k+1)} = (\mathrm{D}+\mathrm{L})\mathrm{D}^{-1}\mathrm{L}(\mathrm{D}+\mathrm{L})^{-1}\mathrm{U}\mathrm{x}^{(k)} + \mathrm{b}\end{aligned}$$

$$\Leftrightarrow \mathrm{M}\mathrm{x}^{(k+1)} = \mathrm{N}\mathrm{x}^{(k)} + \mathrm{b},$$

em que $\mathrm{M} = (\mathrm{D}+\mathrm{L})\mathrm{D}^{-1}(\mathrm{D}+\mathrm{U})$ e $\mathrm{N} = (\mathrm{D}+\mathrm{L})\mathrm{D}^{-1}\mathrm{L}(\mathrm{D}+\mathrm{L})^{-1}\mathrm{U}$. Note que

$$\begin{aligned}
\mathrm{N} &= (\mathrm{D}+\mathrm{L})\mathrm{D}^{-1}\mathrm{L}(\mathrm{D}+\mathrm{L})^{-1}\mathrm{U} \\
&= \mathrm{D}\mathrm{D}^{-1}\mathrm{L}(\mathrm{D}+\mathrm{L})^{-1}\mathrm{U} + \mathrm{L}\mathrm{D}^{-1}\mathrm{L}(\mathrm{D}+\mathrm{L})^{-1}\mathrm{U} \\
&= \mathrm{L}(\mathrm{I}+\mathrm{D}^{-1}\mathrm{L})(\mathrm{D}+\mathrm{L})^{-1}\mathrm{U} \\
&= \mathrm{L}\mathrm{D}^{-1}(\mathrm{D}+\mathrm{L})(\mathrm{D}+\mathrm{L})^{-1}\mathrm{U} \\
&= \mathrm{L}\mathrm{D}^{-1}\mathrm{U}.
\end{aligned}$$

Além disso,

$$\begin{aligned}
\mathrm{M}-\mathrm{N} &= (\mathrm{D}+\mathrm{L})\mathrm{D}^{-1}(\mathrm{D}+\mathrm{U}) - \mathrm{L}\mathrm{D}^{-1}\mathrm{U} \\
&= \mathrm{D}\mathrm{D}^{-1}\mathrm{D} + \mathrm{D}\mathrm{D}^{-1}\mathrm{U} + \mathrm{L}\mathrm{D}^{-1}\mathrm{D} + \mathrm{L}\mathrm{D}^{-1}\mathrm{U} - \mathrm{L}\mathrm{D}^{-1}\mathrm{U} \\
&= \mathrm{D}+\mathrm{U}+\mathrm{L} = \mathrm{A}.
\end{aligned}$$

Dessa forma, obtemos

$$\mathrm{M}\mathrm{x}^{(k+1)} = \mathrm{N}\mathrm{x}^{(k)} + \mathrm{b} = (\mathrm{M}-\mathrm{A})\mathrm{x}^{(k)} + \mathrm{b} = \mathrm{M}\mathrm{x}^{(k)} + \mathrm{b} - \mathrm{A}\mathrm{x}^{(k)}$$

$$\Leftrightarrow \mathrm{x}^{(k+1)} = \mathrm{x}^{(k)} + \mathrm{M}^{-1}(\mathrm{b}-\mathrm{A}\mathrm{x}^{(k)}) = \mathrm{x}^{(k)} + \mathrm{M}^{-1}\mathrm{r}^{(k)}.$$

Logo, a matriz de iteração do método de Gauss-Seidel simétrico é $\mathrm{M} = (\mathrm{D}+\mathrm{L})\mathrm{D}^{-1}(\mathrm{D}+\mathrm{U})$. No caso em que a matriz A é simétrica, então $\mathrm{U} = \mathrm{L}^T$ e $\mathrm{M} = (\mathrm{D}+\mathrm{L})\mathrm{D}^{-1}(\mathrm{D}+\mathrm{L})^T$ também é simétrica.

Escolhendo um parâmetro de relaxação $\omega \in (0,2)$, podemos obter o método SSOR (do inglês *Symmetric Successive Over-Relaxation*) que consiste em aplicar uma iteração do método SOR na ordem padrão, seguida de uma iteração do SOR na ordem reversa. De maneira parecida com a que fizemos anteriormente, podemos verificar que

$$\mathrm{M} = \frac{\omega}{2-\omega}\left(\frac{1}{\omega}\mathrm{D}+\mathrm{L}\right)\mathrm{D}^{-1}\left(\frac{1}{\omega}\mathrm{D}+\mathrm{U}\right)$$

é a matriz de iteração do método SSOR.

3.1.4 Suavização

Consideremos agora um sistema linear, $\mathrm{A}\mathrm{u} = \mathrm{f}$, proveniente da discretização de um problema de equações diferenciais, com matriz $\mathrm{A} : n \times n$ simétrica definida positiva. Vamos estudar o comportamento dos métodos estacionários (Jacobi, Gauss-Seidel, SSOR) para resolver este tipo de sistema. Nesta seção, utilizaremos como exemplo o problema modelo de Poisson unidimensional com condições de contorno de Dirichlet, que consiste em encontrar uma função $u : \Omega \subset \mathbb{R} \longrightarrow \mathbb{R}$ que satisfaça

$$\begin{aligned}
-u''(x) &= f, \\
u(0) = u(1) &= 0,
\end{aligned}$$

no domínio $\Omega = [0,1]$. Após a discretização desse domínio em pontos com espaçamento $h = \frac{1}{n}$ e a aplicação do método de diferenças finitas centradas, obtemos um sistema linear

na forma Au = f, com matriz tridiagonal. Também vamos considerar um sistema vindo da discretização, pelo método dos elementos finitos, de um problema de otimização topológica da viga em balanço tridimensional, que será apresentado no Exemplo 1 da Seção 7.2.

É mais simples analisarmos a resolução do sistema linear homogêneo associado, Au = 0, pois, neste caso, conhecemos a solução exata (u = 0) e, para qualquer aproximação v, temos que o erro é dado por e = −v. Assim, podemos obter informações sobre as componentes do erro olhando apenas para as componentes da aproximação v.

Obtemos informações interessantes ao considerarmos as componentes da aproximação como sendo os chamados *modos de Fourier* (do inglês, *Fourier modes*):

$$v_j = \sin\left(\frac{jk\pi}{n}\right), \qquad j = 1, 2, ..., n.$$

O inteiro k é denominado *número de onda* e indica a quantidade de meias ondas de seno que constituem o vetor v no domínio do problema. Valores pequenos de k correspondem a ondas mais longas e *suaves* (de baixa frequência), enquanto valores grandes de k correspondem a ondas mais *oscilatórias* (de alta frequência), como ilustra a Figura 3.2.

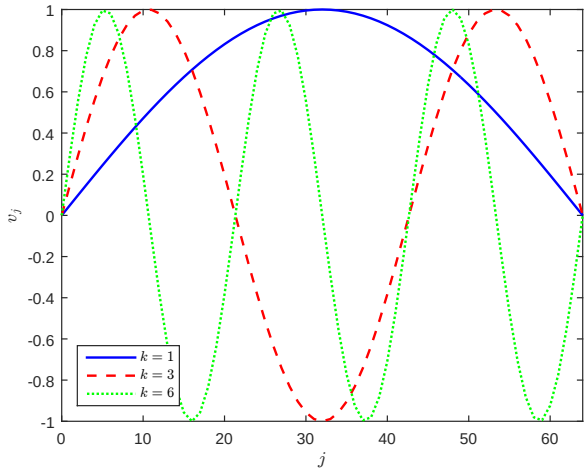
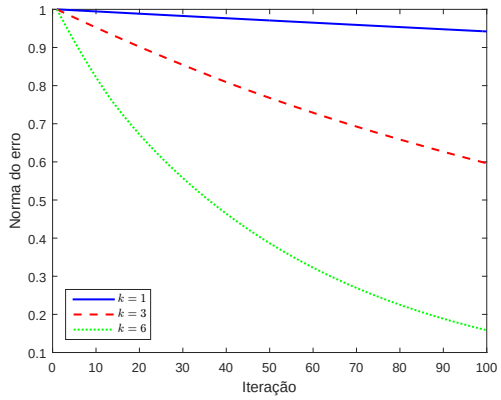


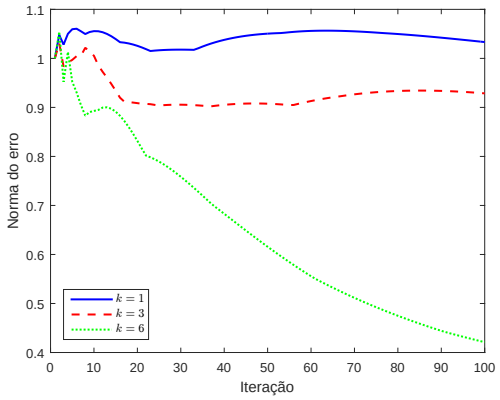
Figura 3.2: Modos de Fourier com diferentes números de onda.

Partindo de aproximações iniciais desta forma, com diferentes valores de k, aplicamos 100 iterações do método estacionário para resolver o sistema Au = 0 do problema de Poisson com $n = 64$ e de um problema da viga em balanço com $n = 41616$. As Figuras 3.3, 3.4 e 3.5 mostram os gráficos do número de iterações pela norma infinito do erro ($\|e\|_\infty$), para os métodos de Jacobi, Gauss-Seidel e SSOR, respectivamente. Vemos que o erro diminui no decorrer das iterações e que a taxa de decréscimo é maior quanto maior for o valor de k. Ou seja, os métodos estacionários reduzem mais rapidamente o erro com alta frequência, porém são mais lentos quando o erro é suave (com baixa frequência).

Agora, considerando a aproximação inicial com um número de onda fixo, $k = 3$, vamos observar o que acontece quando mudamos a dimensão n do problema, ou seja, quando consideramos malhas com refinamentos diferentes. Novamente, realizamos 100 iterações do método estacionário para resolver sistemas do problema de Poisson e da viga em balanço. As Figuras 3.6, 3.7 e 3.8 mostram os resultados para os métodos de Jacobi, Gauss-Seidel e SSOR, respectivamente. Observe que a convergência é melhor para as malhas mais grossas, isto é, para valores menores de n. Isso indica que, quando o método não consegue progredir, é melhor trabalhar em uma malha mais grossa.

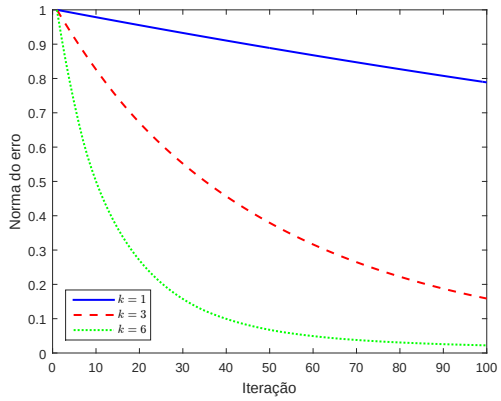


(a) Problema de Poisson com $n = 64$.

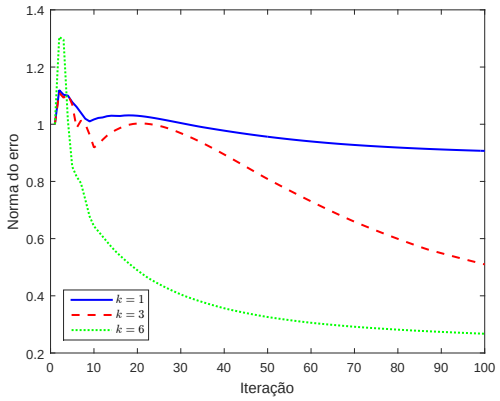


(b) Viga em balanço com $n = 41616$.

Figura 3.3: Convergência do método de Jacobi com relaxação $\omega = 0{,}5$ para aproximações iniciais compostas por modos de Fourier com diferentes parâmetros k.

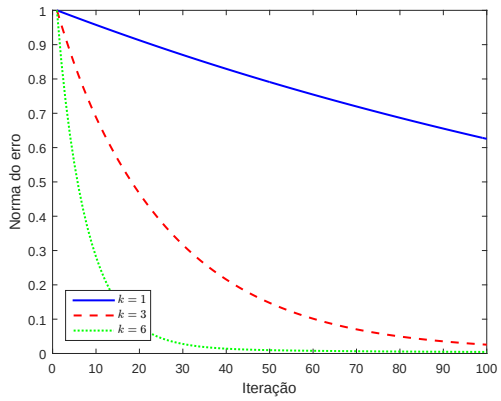


(a) Problema de Poisson com $n = 64$.

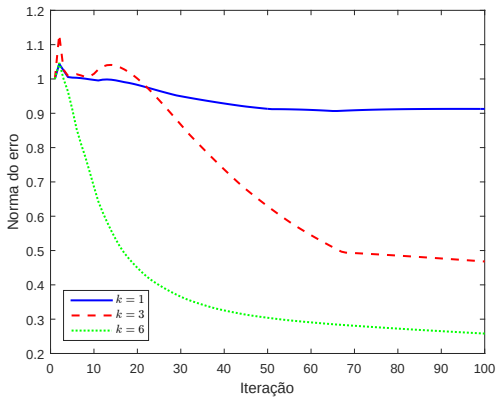


(b) Viga em balanço com $n = 41616$.

Figura 3.4: Convergência do método de Gauss-Seidel (SOR com $\omega = 1$) para aproximações iniciais compostas por modos de Fourier com diferentes parâmetros k.

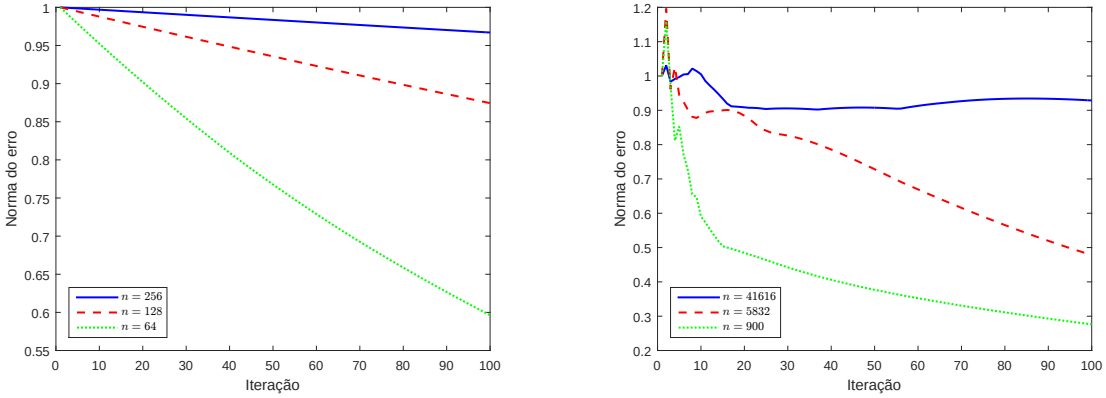


(a) Problema de Poisson com $n = 64$.



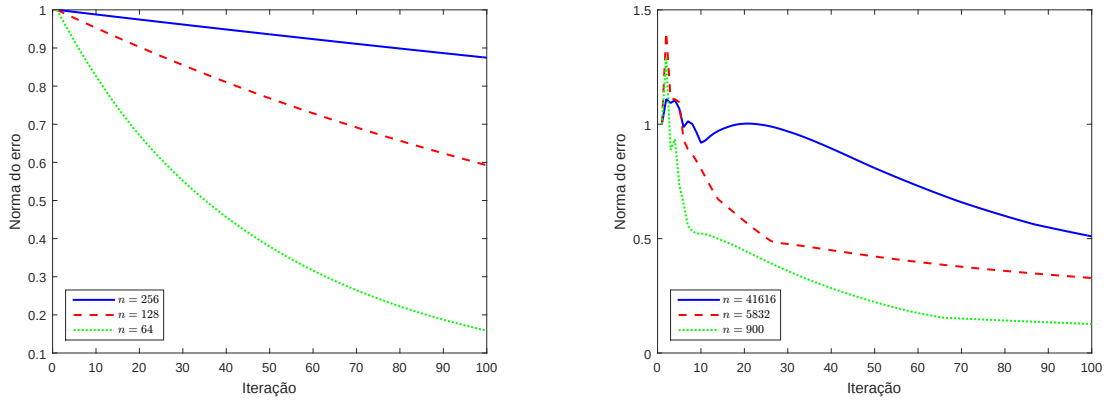
(b) Viga em balanço com $n = 41616$.

Figura 3.5: Convergência do método SSOR com $\omega = 1$ para aproximações iniciais compostas por modos de Fourier com diferentes parâmetros k.



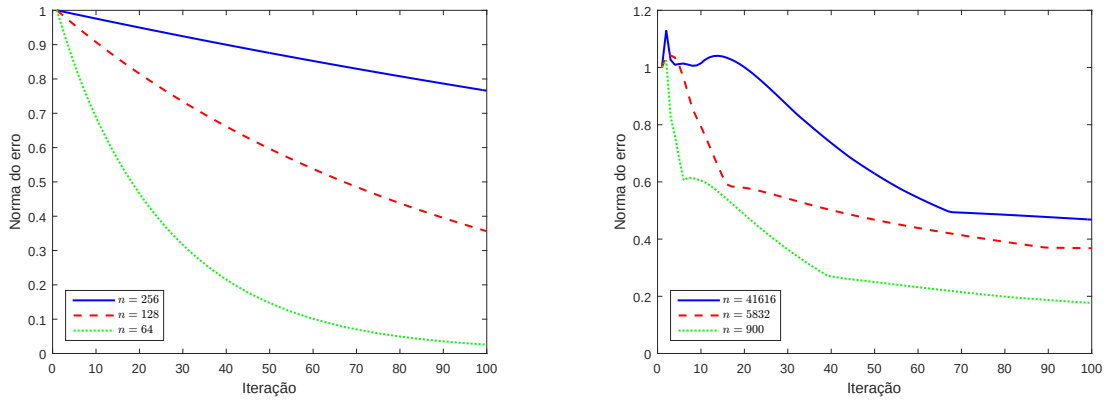
(a) Problema de Poisson. (b) Viga em balanço.

Figura 3.6: Convergência do método de Jacobi com relaxação $\omega = 0{,}5$ para aproximações iniciais compostas por modos de Fourier com $k = 3$ em diferentes refinamentos da malha.



(a) Problema de Poisson. (b) Viga em balanço.

Figura 3.7: Convergência do método de Gauss-Seidel (SOR com $\omega = 1$) para aproximações iniciais compostas por modos de Fourier com $k = 3$ em diferentes refinamentos da malha.



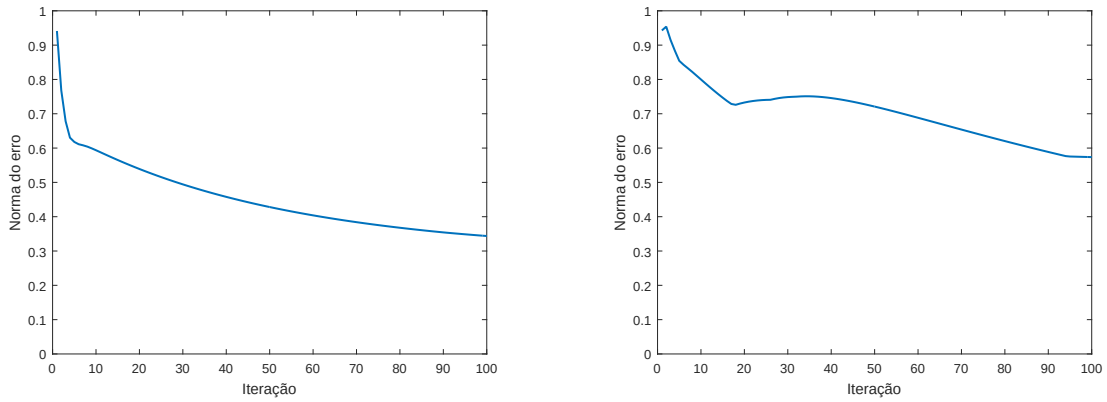
(a) Problema de Poisson. (b) Viga em balanço.

Figura 3.8: Convergência do método SSOR com $\omega = 1$ para aproximações iniciais compostas por modos de Fourier com $k = 3$ em diferentes refinamentos da malha.

Em geral, os vetores não possuem componentes com a mesma frequência. Então, outro experimento interessante é considerar a aproximação v constituída por três modos de Fourier: um de baixa frequência ($k = 1$), um de média frequência ($k = 6$) e outro de alta frequência ($k = 32$), da forma

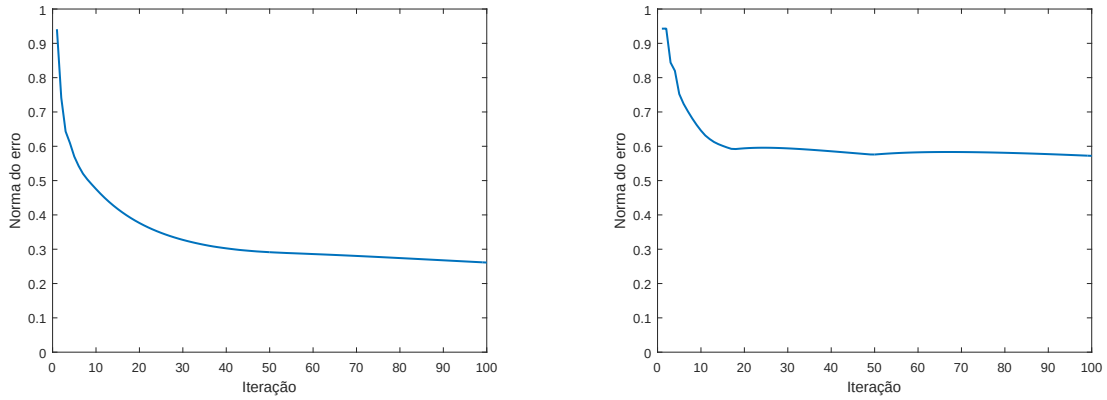
$$v_j = \frac{1}{3}\left[\sin\left(\frac{j\pi}{n}\right) + \sin\left(\frac{6j\pi}{n}\right) + \sin\left(\frac{32j\pi}{n}\right)\right], \qquad j = 1, 2, ..., n.$$

Aplicamos 100 iterações do método estacionário a partir desta aproximação inicial para resolver o sistema $Au = 0$ do problema de Poisson com $n = 64$ e de um problema da viga em balanço com $n = 41616$. Traçando o gráfico do número de iterações pela norma infinito do erro, vemos que o erro é reduzido rapidamente nas primeiras iterações e depois decresce de forma mais lenta. As Figuras 3.9, 3.10 e 3.11 mostram esses resultados para os métodos de Jacobi, Gauss-Seidel e SSOR, respectivamente. O decréscimo inicial rápido corresponde à eliminação das componentes de alta frequência do erro. Quando sobram apenas as componentes de baixa frequência, o método se torna lento e encontra dificuldades para convergir.



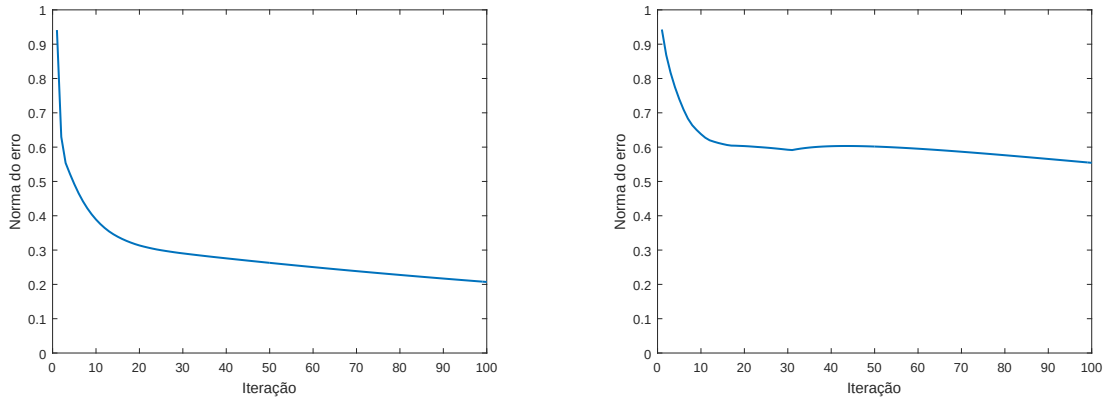
(a) Problema de Poisson com $n = 64$. (b) Viga em balanço com $n = 41616$.

Figura 3.9: Convergência do método de Jacobi com relaxação $\omega = 0{,}5$ para uma aproximação inicial composta por três modos de Fourier com diferentes frequências.



(a) Problema de Poisson com $n = 64$. (b) Viga em balanço com $n = 41616$.

Figura 3.10: Convergência do método de Gauss-Seidel para uma aproximação inicial composta por três modos de Fourier com diferentes frequências.



(a) Problema de Poisson com $n = 64$. (b) Viga em balanço com $n = 41616$.

Figura 3.11: Convergência do método SSOR com $\omega = 1$ para uma aproximação inicial composta por três modos de Fourier com diferentes frequências.

A eliminação das componentes de alta frequência é chamada de *suavização*. Podemos observar a suavização analisando as componentes do vetor erro em cada iteração, conforme ilustra a Figura 3.12 para o método de Jacobi aplicado à resolução de um sistema linear do problema de Poisson com $n = 64$. Note que, na aproximação inicial, o erro era bastante oscilatório e com apenas 10 iterações ele se torna mais suave.

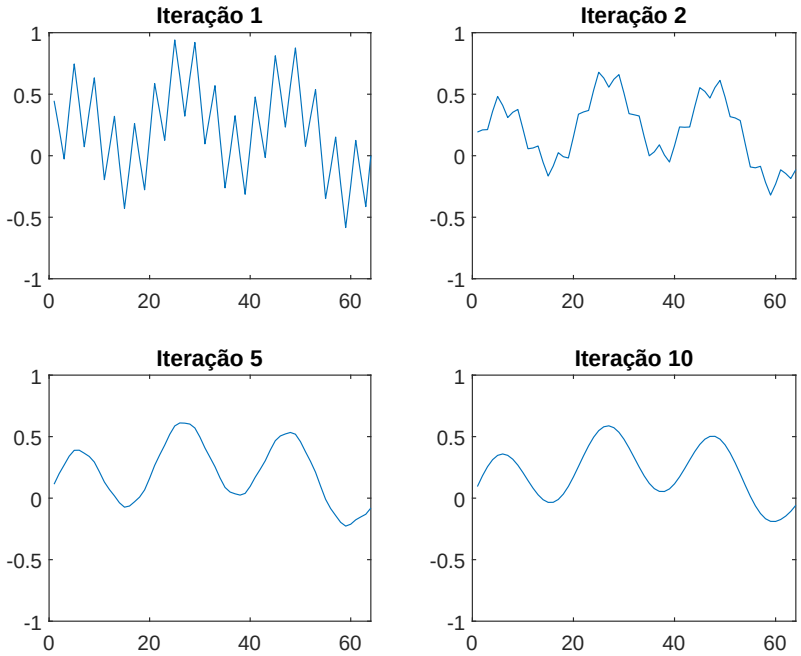


Figura 3.12: Suavização do erro pelo método de Jacobi com $\omega = 0{,}5$ aplicado ao problema de Poisson com $n = 64$.

Estes experimentos exemplificam a constatação de que os métodos estacionários conseguem eliminar rapidamente as componentes de alta frequência dos erros, fazendo com que eles fiquem mais suaves. Contudo, esses métodos encontram dificuldades em trabalhar com os erros suaves e se tornam lentos. Além disso, um erro suave em uma determinada malha é, em geral, menos suave em uma malha mais grossa, onde o método estacionário consegue ter mais efeito sobre ele. Isso motiva a ideia do método *multigrid*, que pode ser dividido em duas classes: *multigrid* geométrico e *multigrid* algébrico.

3.2 *Multigrid* geométrico

O *multigrid* é um método iterativo desenvolvido para a resolução de sistemas lineares provenientes da discretização de equações diferenciais ou problemas de valor de contorno (PVC), que surgem em diversas aplicações, como é o caso dos sistemas de equilíbrio que aparecem na resolução do problema de otimização topológica. Os livros de Briggs *et al.* [10] e Trottenberg *et al.* [52] são ótimas referências para o método *multigrid*.

Consideremos, por exemplo, um domínio $\Omega \subset \mathbb{R}^3$ com fronteira Γ. Em um PVC, queremos encontrar uma função $u : \Omega \longrightarrow \mathbb{R}$ que satisfaça um sistema de equações diferenciais e algumas condições de contorno sobre os pontos da fronteira Γ. Em muitos casos, não conseguimos encontrar uma solução analítica e nos contentamos em obter aproximações para os valores da função em um conjunto finito de pontos.

Para isso, discretizamos o domínio Ω em uma malha Ω_h, em que $h = (h_x, h_y, h_z)$ contém informações das dimensões dos elementos da malha nas três direções coordenadas x, y e z. Utilizando o método dos elementos finitos ou diferenças finitas, por exemplo, após aplicar as condições de contorno, obtemos um sistema linear na forma $\mathrm{A}_h\mathrm{u}_h = \mathrm{f}_h$, do qual queremos obter o vetor u_h com as aproximações dos valores da função nos nós da malha. Em geral, a matriz A_h é esparsa, em bandas, simétrica e definida positiva. Entretanto, a dimensão dessa matriz pode ser muito grande, dependendo da quantidade de elementos na malha, de modo que a resolução do sistema linear se torna uma etapa muito cara na resolução dos problemas. O *multigrid* surge como uma alternativa mais eficaz para a resolução de sistemas lineares desse tipo.

3.2.1 Considerações iniciais

Considere o sistema linear $\mathrm{Au} = \mathrm{f}$. Seja v uma aproximação para a solução u. Definimos o vetor erro: $\mathrm{e} = \mathrm{u} - \mathrm{v}$ e o vetor resíduo: $\mathrm{r} = \mathrm{f} - \mathrm{Av}$. Então, podemos escrever

$$\mathrm{Ae} = \mathrm{A}(\mathrm{u} - \mathrm{v}) = \mathrm{Au} - \mathrm{Av} = \mathrm{f} - \mathrm{Av} = \mathrm{r},$$

ou seja, $\mathrm{Ae} = \mathrm{r}$, que chamamos de *sistema do resíduo*. Se resolvermos esse sistema, obtendo o vetor erro, então teremos a solução do sistema original, dada por $\mathrm{u} = \mathrm{v} + \mathrm{e}$.

Como o próprio nome sugere, o método *multigrid* trabalha em múltiplas malhas, com elementos de diferentes tamanhos. Ele utiliza informações do sistema linear original e do sistema do resíduo em cada malha para obter aproximações da solução, tentando explorar a vantagem de trabalhar com malhas mais grossas (isto é, com menos elementos), nas quais as dimensões dos sistemas são menores.

A motivação para o desenvolvimento deste método surge da observação de alguns fenômenos que vimos experimentalmente na Subseção 3.1.4. Para os sistemas lineares em questão notam-se dois princípios interessantes:

- **Princípio da suavização:** alguns métodos iterativos estacionários (Jacobi, Gauss-Seidel etc.) conseguem suavizar rapidamente as componentes do erro de qualquer aproximação para a solução do sistema.
- **Princípio da malha grossa:** um erro suave pode ser bem aproximado, sem perda de informação, em uma malha mais grossa, na qual o método estacionário consegue ter mais efeito sobre ele.

Com base nestes princípios, uma primeira descrição da ideia do método *multigrid* é: aplicar algumas iterações de um método iterativo estacionário para resolver o sistema linear, até que o erro se torne suave. Então, mudar para uma malha mais grossa e obter uma aproximação do erro, resolvendo o sistema do resíduo. Por fim, voltar para a malha mais fina e corrigir a aproximação da solução do sistema original.

3.2.2 Ciclo em duas malhas

Apresentamos o algoritmo do método *multigrid* utilizando duas malhas, que é a base para o desenvolvimento de outros ciclos mais complexos. Considere a discretização do domínio Ω em uma malha Ω_h e o sistema linear $\mathrm{A}_h\mathrm{u}_h = \mathrm{f}_h$. Considere também uma outra malha mais grossa, Ω_H, em que $H > h$. Normalmente tomamos $H = 2h$, ou seja, cada elemento da malha grossa contém dois elementos da malha fina em cada direção.

Para aplicar o método, precisamos de mecanismos capazes de transferir as informações de uma malha para a outra, isto é, precisamos definir os operadores:

- R: leva informações da malha fina, Ω_h, para a malha grossa, Ω_H, denominado *operador de restrição.*
- P: leva informações da malha grossa, Ω_H, para a malha fina, Ω_h, denominado *operador de prolongação (ou interpolação).*

Além disso, precisamos definir a matriz A_H com informações do sistema linear na malha Ω_H. Veremos estratégias para obter esses operadores nas próximas seções. A aplicação do *multigrid* é dividida em duas fases: a construção dos operadores, denominada fase de *setup*, e a fase de resolução. Considerando que já tenhamos realizado a fase de *setup*, podemos escrever o algoritmo do ciclo *multigrid* em duas malhas.

Algoritmo do ciclo em duas malhas

Dada uma aproximação $\mathrm{u}_h^{(k)}$ para a solução do sistema linear, obtemos uma nova aproximação $\mathrm{u}_h^{(k+1)}$ por meio das etapas:

(1) **Pré-suavização:** aplicar η_1 iterações de um método iterativo estacionário (Jacobi, Gauss-Seidel etc.) ao sistema linear $\mathrm{A}_h\mathrm{u}_h = \mathrm{f}_h$, com aproximação inicial $\mathrm{u}_h^{(k)}$, obtendo uma nova aproximação v_h;

(2) **Correção na malha grossa**

   (a) Calcular o resíduo: $\mathrm{r}_h = \mathrm{f}_h - \mathrm{A}_h\mathrm{v}_h$;

   (b) Restringir o resíduo na malha grossa: $\mathrm{r}_H = \mathrm{R}\mathrm{r}_h$;

   (c) Resolver $\mathrm{A}_H\mathrm{e}_H = \mathrm{r}_H$ na malha Ω_H;

   (d) Prolongar o erro na malha fina: $\mathrm{e}_h = \mathrm{P}\mathrm{e}_H$;

   (e) Corrigir a aproximação: $\overline{\mathrm{v}}_h = \mathrm{v}_h + \mathrm{e}_h$;

(3) **Pós-suavização:** aplicar η_2 iterações do método iterativo estacionário ao sistema linear $\mathrm{A}_h\mathrm{u}_h = \mathrm{f}_h$, com aproximação inicial $\overline{\mathrm{v}}_h$, obtendo $\mathrm{u}_h^{(k+1)}$.

O diagrama na Figura 3.13 ilustra as etapas de uma iteração do algoritmo do ciclo em duas malhas.

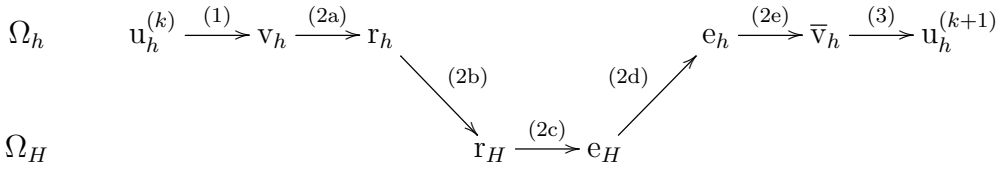


Figura 3.13: Esquema do ciclo em duas malhas.

Esse ciclo é repetido até atingirmos algum critério de parada, que pode ser, por exemplo, a norma do resíduo menor que alguma tolerância preestabelecida. Note que o sistema da etapa (2c) possui dimensão menor do que o sistema original, por estar na malha mais grossa, de modo que a sua resolução é mais simples e barata. Ademais, nem sempre precisamos resolver o sistema do resíduo de forma exata e podemos, inclusive, aplicar novamente o método *multigrid*, de forma recursiva, dando origem a ciclos com mais malhas. Na prática, para aplicar o método precisamos determinar:

- O processo de suavização;
- As quantidades de iterações na pré e na pós-suavização (η_1 e η_2);
- O operador de restrição R;
- O operador de prolongação P;
- A matriz A_H.

Em geral, o processo de suavização é algum método estacionário como Jacobi ou Gauss-Seidel e as quantidades de iterações η_1 e η_2 são pequenas. Essas escolhas dependem do problema a ser resolvido e devem ser testadas experimentalmente. Veremos a seguir estratégias para construir os operadores de restrição e prolongação de forma geométrica e, posteriormente, como obter a matriz do sistema na malha grossa.

3.2.3 Operador de restrição

Como podemos levar as informações da malha fina (Ω_h) para a malha grossa (Ω_H)? Ou seja, conhecido um vetor v_h na malha fina, como podemos obter o vetor correspondente, $v_H = Rv_h$, na malha grossa?

Vamos considerar $H = 2h$, isto é, o tamanho do elemento na malha grossa é o dobro do tamanho do elemento na malha fina, em cada direção, de modo que cada elemento na malha grossa contém dois elementos da malha fina, em cada direção. Ademais, consideramos que as malhas estão sobrepostas e que os elementos são lineares.

Uma ideia bastante simples é associar cada nó x na malha grossa ao seu correspondente na malha fina, fazendo $v_H(x) = v_h(x)$, para todo $x \in \Omega_H \subset \Omega_h$. Todavia, esta ideia não apresenta bons resultados, já que não leva em conta as informações dos nós da malha fina que não estão na malha grossa. A Figura 3.14 ilustra esta ideia em 1D.

Os pontos com marcador "×", em azul, correspondem aos nós da malha fina que também estão na malha grossa.

Outra estratégia mais elaborada é a denominada restrição *full-weighting*, que leva em consideração os valores em todos os nós da malha fina. O valor em cada nó da malha grossa é dado por uma combinação dos valores do nó correspondente na malha fina e dos seus vizinhos. A Figura 3.15 ilustra esta estratégia em 1D.

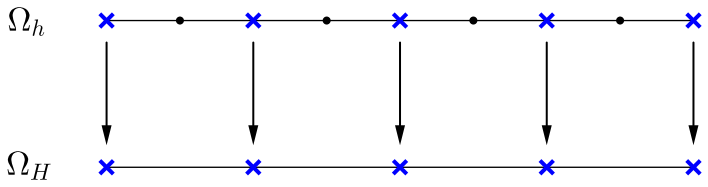


Figura 3.14: Restrição simples em 1D.

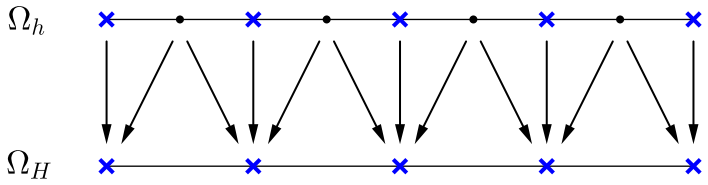


Figura 3.15: Restrição *full-weighting* em 1D.

Restrição 1D: considere um problema unidimensional. Neste caso, o valor em cada nó x da malha grossa receberá como contribuição metade do valor do nó correspondente na malha fina e um quarto de cada um dos seus vizinhos:

$$\mathrm{v}_H(x) = \frac{1}{4}\mathrm{v}_h(x-h) + \frac{1}{2}\mathrm{v}_h(x) + \frac{1}{4}\mathrm{v}_h(x+h).$$

Note que os nós nas extremidades da malha possuem apenas um vizinho, enquanto os nós interiores possuem dois vizinhos.

Por exemplo, considerando a malha unidimensional Ω_h com 9 nós e a malha Ω_H com 5 nós, como na Figura 3.15, suponha que já conhecemos os valores $v_1, v_2, ..., v_9$ para os nós da malha fina. Obtemos os valores $z_1, z_2, ..., z_5$ para os nós da malha grossa por meio da seguinte relação:

$$\begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \end{bmatrix}_H = \underbrace{\frac{1}{4}\begin{bmatrix} 2 & 1 & & & & & & & \\ & 1 & 2 & 1 & & & & & \\ & & & 1 & 2 & 1 & & & \\ & & & & & 1 & 2 & 1 & \\ & & & & & & & 1 & 2 \end{bmatrix}}_{\mathrm{R}} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \\ v_9 \end{bmatrix}_h . \tag{3.3}$$

Neste exemplo, a matriz de restrição R é de ordem 5×9.

Restrição 2D: no caso bidimensional, cada nó (x, y) da malha grossa recebe contribuição do nó correspondente na malha fina e de até oito vizinhos. As frações de

contribuição são: 1/4 do valor do próprio nó, 1/8 de cada vizinho mais próximo (a um passo de distância) e 1/16 de cada vizinho mais distante (a dois passos de distância), conforme ilustra o esquema na Figura 3.16. O ponto central corresponde ao nó da malha grossa e os pontos pretos ao redor são nós que estão apenas na malha fina. Neste caso, para um nó (x, y) interior da malha grossa, temos:

$$\begin{aligned} \mathrm{v}_H(x,y) = \frac{1}{16}\,[&4\mathrm{v}_h(x,y) + 2\mathrm{v}_h(x+h,y) + 2\mathrm{v}_h(x-h,y) \\ &+ 2\mathrm{v}_h(x,y+h) + \mathrm{v}_h(x,y-h) + \mathrm{v}_h(x+h,y+h) \\ &+ \mathrm{v}_h(x+h,y-h) + \mathrm{v}_h(x-h,y+h) + \mathrm{v}_h(x-h,y-h)]\,. \end{aligned}$$

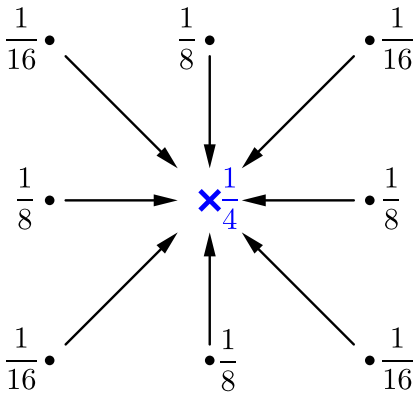


Figura 3.16: Restrição *full-weighting* em 2D.

Restrição 3D: no caso tridimensional, cada nó (x, y, z) da malha grossa recebe contribuição do nó correspondente na malha fina e de até 26 vizinhos. As frações de contribuição são: 1/8 do valor do próprio nó, 1/16 de cada vizinho mais próximo (a um passo de distância), 1/32 de cada vizinho intermediário (a dois passos de distância) e 1/64 de cada vizinho mais distante (a três passos de distância).

3.2.4 Operador de prolongação

Como podemos levar as informações da malha grossa (Ω_H) para a malha fina (Ω_h)? Ou seja, conhecido um vetor v_H na malha grossa, como podemos obter o vetor correspondente, $\mathrm{v}_h = \mathrm{P}\mathrm{v}_H$, na malha fina?

No método *multigrid*, em geral, utilizamos o operador de prolongação para levar o vetor erro da malha grossa para a malha fina. Como as componentes do erro ficam suaves, uma boa estratégia consiste em utilizar a interpolação linear. Nesse caso, os valores nos nós da malha fina que também estão na malha grossa se mantêm iguais. Já um nó da malha fina que não está na malha grossa recebe uma média dos valores nos seus vizinhos que estão na malha grossa.

Prolongação 1D: no caso unidimensional, cada elemento na malha grossa corresponde a dois elementos na malha fina. Dado um nó x da malha fina, se $x \in \Omega_H$, então $\mathrm{v}_h(x) = \mathrm{v}_H(x)$. Caso contrário, consideramos os dois pontos $(x-h, \mathrm{v}_H(x-h))$ e $(x+h, \mathrm{v}_H(x+h))$, fazemos a interpolação linear traçando o segmento de reta cujos extremos são esses dois pontos e tomamos o ponto médio desse segmento como sendo o

ponto $(x, \mathrm{v}_h(x))$. Assim,

$$\mathrm{v}_h(x) = \begin{cases} \mathrm{v}_H(x), & \text{se } x \in \Omega_H, \\ \frac{1}{2}\left[\mathrm{v}_H(x-h) + \mathrm{v}_H(x+h)\right], & \text{se } x \notin \Omega_H. \end{cases}$$

A Figura 3.17 ilustra a ideia de prolongação no caso unidimensional. Aqui, os nós da malha estão numerados no segmento horizontal e os pontos representam os pares $(x, \mathrm{v}_h(x))$.

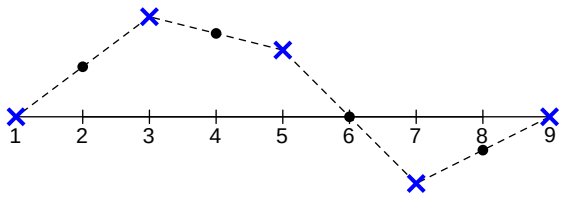


Figura 3.17: Prolongação (interpolação) em 1D.

Por exemplo, considerando as malhas unidimensionais Ω_h com 9 nós e Ω_H com 5 nós, suponha que já conheçamos os valores $z_1, z_2, ..., z_5$ para os nós da malha grossa. Obtemos os valores $v_1, v_2, ..., v_9$ para os nós da malha fina por meio da seguinte relação:

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \\ v_9 \end{bmatrix}_h = \underbrace{\frac{1}{2}\begin{bmatrix} 2 & & & & \\ 1 & 1 & & & \\ & 2 & & & \\ & 1 & 1 & & \\ & & 2 & & \\ & & 1 & 1 & \\ & & & 2 & \\ & & & 1 & 1 \\ & & & & 2 \end{bmatrix}}_{\mathrm{P}} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \end{bmatrix}_H .$$

Neste exemplo, a matriz de prolongação P é de ordem 9×5. Note também que $\mathrm{P} = 2\mathrm{R}^T$, sendo R a matriz do exemplo de restrição 1D em (3.3).

Prolongação 2D: no caso bidimensional, cada elemento na malha grossa corresponde a quatro elementos na malha fina. Para um nó (x, y) que está em alguma aresta do elemento na malha grossa, podemos fazer algo parecido com o caso 1D, acrescentando a componente y. Já para o nó no interior do elemento, o valor será uma média dos valores nos quatro nós da malha grossa que estão ao seu redor.

O diagrama na Figura 3.18 ilustra a prolongação em 2D. Os nós representados por $\bullet$ estão na malha grossa, já os representados por $\triangle$, $\square$ ou $\bigcirc$ estão apenas na malha fina. Os valores nas setas indicam as frações de contribuição que cada nó da malha fina recebe dos nós da malha grossa. Dessa forma, temos:

$$\mathrm{v}_h(x,y) = \begin{cases} \mathrm{v}_H(x,y), & \text{se } (x,y) = \bullet, \\ \frac{1}{2}\left[\mathrm{v}_H(x-h,y) + \mathrm{v}_H(x+h,y)\right], & \text{se } (x,y) = \triangle, \\ \frac{1}{2}\left[\mathrm{v}_H(x,y-h) + \mathrm{v}_H(x,y+h)\right], & \text{se } (x,y) = \square, \\ \frac{1}{4}\left[\mathrm{v}_H(x-h,y-h) + \mathrm{v}_H(x-h,y+h) + \right. & \\ \left. \quad + \mathrm{v}_H(x+h,y-h) + \mathrm{v}_H(x+h,y+h)\right], & \text{se } (x,y) = \bigcirc. \end{cases}$$

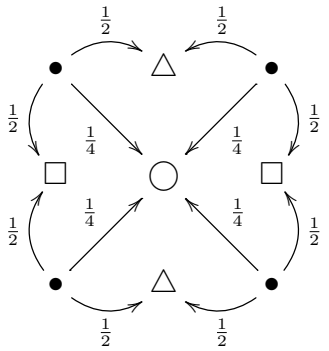


Figura 3.18: Prolongação em 2D.

Prolongação 3D: no caso tridimensional, cada elemento na malha grossa corresponde a oito elementos na malha fina. Para um nó (x, y, z) que está em alguma face do elemento na malha grossa, podemos fazer algo parecido com o caso 2D, acrescentando a componente z. Já para o nó no interior do elemento, o valor será uma média dos valores nos oito nós da malha grossa que estão ao seu redor, isto é,

$$\begin{aligned} \mathrm{v}_h(x,y,z) = \frac{1}{8}\,[&\mathrm{v}_H(x+h,y+h,z+h) + \mathrm{v}_H(x+h,y+h,z-h) \\ &+ \mathrm{v}_H(x+h,y-h,z+h) + \mathrm{v}_H(x-h,y+h,z+h) \\ &+ \mathrm{v}_H(x+h,y-h,z-h) + \mathrm{v}_H(x-h,y-h,z+h) \\ &+ \mathrm{v}_H(x-h,y+h,z-h) + \mathrm{v}_H(x-h,y-h,z-h)]\,. \end{aligned}$$

Na prática, se trabalharmos com o operador de restrição *full-weighting* e a interpolação, é possível mostrar que um operador é o transposto do outro multiplicado por uma constante $c \in \mathbb{R}$, isto é,

$$\mathrm{P} = c\mathrm{R}^T,$$

em que c depende da dimensão ($c = 2$ em 1D, $c = 4$ em 2D e $c = 8$ em 3D). Neste caso, só precisamos construir uma dessas matrizes. Na implementação, optamos por construir apenas a matriz de prolongação.

Prolongação com elementos de grau maior

A maneira como definimos o operador de prolongação utiliza informações geométricas das malhas, como a localização dos nós. Os valores para os nós da malha fina são obtidos por meio de uma interpolação, utilizando os nós da malha grossa que estão ao seu redor. Normalmente, esse operador é descrito para uma malha com elementos lineares. Entretanto, a construção da matriz P se torna mais complicada se queremos utilizar elementos de grau maior, como os descritos nas Subseções 1.3.2 e 1.3.3, principalmente no caso dos elementos do tipo *serendipity*, em que os nós não estão igualmente espaçados.

Para os elementos do tipo Lagrange, ainda podemos usar a interpolação linear. Já para os elementos *serendipity*, propomos a construção do operador de prolongação utilizando a interpolação por meio das funções de forma do método dos elementos finitos. Para o elemento prismático retangular de grau 1, as funções de forma são dadas por

$$\phi_i(\xi,\eta,\zeta) = \frac{1}{8}(1+\xi_i\xi)(1+\eta_i\eta)(1+\zeta_i\zeta),$$

para $i = 1, 2, ..., 8$, em que ξ_i, η_i e ζ_i são as coordenadas locais do nó i no sistema de coordenadas (ξ, η, ζ) com origem no centro do elemento. Dado o vetor $\mathrm{u}^{(e)}$ contendo os deslocamentos nodais de um elemento e, podemos obter os deslocamentos aproximados para um ponto qualquer do elemento calculando $\hat{\mathrm{u}} = \Phi \mathrm{u}^{(e)}$, em que

$$\Phi = \begin{bmatrix} \phi_1 & 0 & 0 & \phi_2 & 0 & 0 & \cdots & \phi_8 & 0 & 0 \\ 0 & \phi_1 & 0 & 0 & \phi_2 & 0 & \cdots & 0 & \phi_8 & 0 \\ 0 & 0 & \phi_1 & 0 & 0 & \phi_2 & \cdots & 0 & 0 & \phi_8 \end{bmatrix}$$

é uma matriz contendo as funções de forma aplicadas no mesmo ponto em que estamos aproximando os deslocamentos.

Considere agora um elemento da malha grossa, que contém oito elementos da malha fina. Suponha que, ao invés de deslocamentos, conheçamos valores quaisquer, $\mathrm{v}_H(\xi_i, \eta_i, \zeta_i)$, nos nós do elemento da malha grossa, para $i = 1, 2, ..., 8$. O valor $\mathrm{v}_h(\xi, \eta, \zeta)$ para um nó da malha fina contido nesse elemento é obtido por meio de uma combinação dos valores conhecidos, multiplicados pelas funções de forma aplicadas no ponto (ξ, η, ζ), de maneira similar à que fizemos acima para os deslocamentos. Isto é,

$$\mathrm{v}_h(\xi, \eta, \zeta) = \phi_1(\xi, \eta, \zeta)\mathrm{v}_H(\xi_1, \eta_1, \zeta_1) + \cdots + \phi_8(\xi, \eta, \zeta)\mathrm{v}_H(\xi_8, \eta_8, \zeta_8). \tag{3.4}$$

No caso em que o nó da malha fina também está na malha grossa, temos

$$\mathrm{v}_h(\xi_i, \eta_i, \zeta_i) = \mathrm{v}_H(\xi_i, \eta_i, \zeta_i),$$

para $i = 1, 2, ..., 8$, uma vez que o valor da função de forma ϕ_i é igual a 1 no nó i e zero nos demais nós. Agora, se o nó está na aresta que conecta os nós 1 e 2 do elemento da malha grossa, por exemplo, os valores da funções ϕ_1 e ϕ_2 são iguais a 1/2 e os valores de $\phi_3, \phi_4, ..., \phi_8$ são nulos. Assim,

$$\mathrm{v}_h(\xi, \eta, \zeta) = \frac{1}{2}\mathrm{v}_H(\xi_1, \eta_1, \zeta_1) + \frac{1}{2}\mathrm{v}_H(\xi_2, \eta_2, \zeta_2).$$

Os valores para nós em outras arestas são obtidos de maneira similar. Para o nó da malha fina no centro da face formada pelos nós 1, 2, 3 e 4 do elemento da malha grossa, os valores da funções ϕ_1, ϕ_2, ϕ_3 e ϕ_4 são iguais a 1/4 e os valores de ϕ_5, ϕ_6, ϕ_7 e ϕ_8 são nulos. Daí,

$$\mathrm{v}_h(\xi, \eta, \zeta) = \frac{1}{4}\left[\mathrm{v}_H(\xi_1, \eta_1, \zeta_1) + \mathrm{v}_H(\xi_2, \eta_2, \zeta_2) + \mathrm{v}_H(\xi_3, \eta_3, \zeta_3) + \mathrm{v}_H(\xi_4, \eta_4, \zeta_4)\right].$$

Para outros nós nos centros das faces do elemento, o cálculo é similar. Por fim, para o nó da malha fina no centro do elemento da malha grossa, temos

$$\begin{aligned} \mathrm{v}_h(\xi, \eta, \zeta) = \frac{1}{8}\big[&\mathrm{v}_H(\xi_1, \eta_1, \zeta_1) + \mathrm{v}_H(\xi_2, \eta_2, \zeta_2) + \mathrm{v}_H(\xi_3, \eta_3, \zeta_3) + \mathrm{v}_H(\xi_4, \eta_4, \zeta_4) + \\ &+ \mathrm{v}_H(\xi_5, \eta_5, \zeta_5) + \mathrm{v}_H(\xi_6, \eta_6, \zeta_6) + \mathrm{v}_H(\xi_7, \eta_7, \zeta_7) + \mathrm{v}_H(\xi_8, \eta_8, \zeta_8)\big]. \end{aligned}$$

uma vez que todas as funções $\phi_1, \phi_2, ..., \phi_8$ valem 1/8.

Podemos observar que as expressões obtidas são equivalentes àquelas encontradas anteriormente, quando descrevemos a prolongação utilizando interpolação linear. Ou seja, a expressão (3.4) descreve o operador de prolongação utilizando as funções de

forma do elemento finito. Aplicando essa expressão para cada elemento da malha grossa, conseguimos obter os valores prolongados para todos os nós da malha fina.

Portanto, se quisermos utilizar elementos de grau maior, basta trocar as funções de forma e os valores conhecidos para os nós do elemento da malha grossa. No caso do elemento prismático retangular de grau 2 do tipo *serendipity*, por exemplo, para cada elemento da malha grossa, conhecemos os valores nos 20 nós desse elemento. Os valores em nós da malha fina contidos nesse elemento são obtidos por meio de uma combinação dos valores conhecidos multiplicados pelas funções de forma. Neste caso, a prolongação é obtida por uma interpolação quadrática, em que os pontos interpoladores são os 20 nós do elemento na malha grossa.

Na prática, construímos a matriz de prolongação, P, de forma esparsa, considerando a malha global. Cada linha i de P representa um nó i da malha fina. Nas entradas (i, j) dessa linha, cujos índices j correspondem aos índices globais dos nós do elemento da malha grossa que contém o nó i, colocamos os valores das funções de forma aplicadas no ponto correspondente ao nó i. Em particular, no caso da otimização topológica tridimensional, fazemos algo similar, com a diferença que cada linha de P representa um grau de liberdade de um nó da malha fina, ou seja, cada nó está associado a três linhas da matriz. Além disso, impondo as condições de contorno do problema, precisamos retirar de P as linhas e colunas correspondentes aos graus de liberdade que foram restringidos por apoios, assim como fazemos na matriz de rigidez.

3.2.5 O sistema na malha grossa

Conhecemos a matriz A_h do sistema linear original, mas, no algoritmo do ciclo em duas malhas, precisamos resolver o sistema do resíduo na malha grossa, $\mathrm{A}_H \mathrm{e}_H = \mathrm{r}_H$. É possível obter a matriz A_H refazendo a discretização do problema na malha grossa, mas isso seria bastante trabalhoso. Felizmente, conseguimos obter essa matriz de forma mais econômica a partir da matriz A_h e dos operadores de restrição e de prolongação.

Observamos que $\mathrm{e}_h = \mathrm{P}\mathrm{e}_H$ e $\mathrm{r}_H = \mathrm{R}\mathrm{r}_h$. Assim, a partir do sistema do resíduo na malha fina, podemos obter o sistema projetado na malha grossa:

$$\mathrm{A}_h \mathrm{e}_h = \mathrm{r}_h \Rightarrow \mathrm{R}\mathrm{A}_h\mathrm{P}\mathrm{e}_H = \mathrm{R}\mathrm{r}_h \Rightarrow (\mathrm{R}\mathrm{A}_h\mathrm{P})\,\mathrm{e}_H = \mathrm{r}_H.$$

Portanto,

$$\mathrm{A}_H = \mathrm{R}\mathrm{A}_h\mathrm{P}$$

é a matriz do sistema na malha grossa. A matriz obtida por esta abordagem também é conhecida como o *operador de Galerkin* (vide [52], Subseção 2.3.2).

3.2.6 Ciclo V e ciclo W

Uma vez obtidas todas as ferramentas do método *multigrid*, podemos aplicar o algoritmo do ciclo em duas malhas, conforme descrito na Subseção 3.2.2. Na etapa (2c) do algoritmo, precisamos resolver o sistema do resíduo na malha grossa, $\mathrm{A}_H \mathrm{e}_H = \mathrm{r}_H$, que é um sistema linear parecido com o original, mas com dimensão menor, ou seja, mais simples de resolver. Entretanto, esse sistema ainda pode ser grande, dependendo da dimensão do problema original, de modo que ainda seja trabalhoso resolvê-lo diretamente. Neste caso,

é possível aplicar novamente o método *multigrid* para resolvê-lo, gerando um processo recursivo. O verdadeiro poder do *multigrid* é conseguir trabalhar com malhas cada vez mais grossas, até que a dimensão do sistema seja pequena o suficiente para resolvermos com baixo custo.

Suponha que tenhamos uma sequência de malhas: $\Omega_h, \Omega_{2h}, \Omega_{4h}, ..., \Omega_{Lh}$, escolhida de modo que seja possível resolver o sistema linear na malha mais grossa, Ω_{Lh}, diretamente de forma barata, e que tenhamos os operadores de restrição e de prolongação que levam informações de uma malha para a outra. A ideia do ciclo V do *multigrid* é começar na malha mais fina, Ω_h, aplicar a pré-suavização, obter o sistema do resíduo na malha Ω_{2h}, pré-suavizar o sistema nessa malha, obter o sistema do resíduo na malha Ω_{4h}, e assim por diante até chegarmos na malha mais grossa, Ω_{Lh}, na qual o sistema do resíduo pode ser resolvido diretamente. Depois, deve-se corrigir a solução na malha $\Omega_{(L/2)h}$, pós-suavizar e prosseguir dessa maneira até voltar para a malha mais fina, Ω_h.

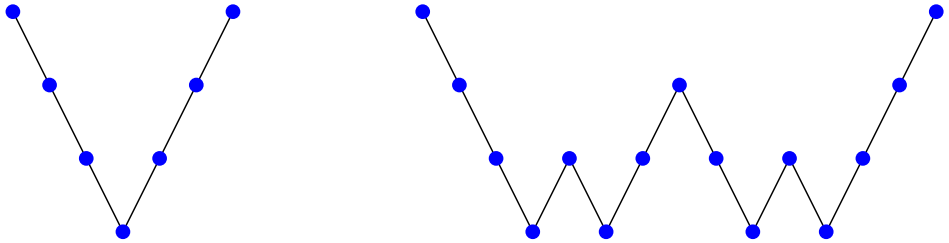
Para simplificar a notação, como feito na descrição do ciclo em duas malhas, vamos denotar por h o espaçamento de uma malha e $H = 2h$ o espaçamento da malha mais grossa em um nível de refinamento imediatamente abaixo. Além disso, suponhamos que R e P representam as matrizes de restrição e prolongação, respectivamente, que levam informações entre as malhas Ω_h e Ω_H. A seguir, exibimos o esquema recursivo do ciclo V.

Algoritmo 2 Ciclo V do *multigrid*

1: **função** CICLOV($\mathrm{v}_h, \mathrm{f}_h, \mathrm{A}_h$)
2: pré-suavizar η_1 vezes o sistema $\mathrm{A}_h\mathrm{u}_h = \mathrm{f}_h$, com aproximação inicial v_h;
3: $\mathrm{f}_H \longleftarrow \mathrm{R}(\mathrm{f}_h - \mathrm{A}_h\mathrm{v}_h)$;
4: **se** Ω_H for a malha mais grossa **então**
5: obter v_H resolvendo o sistema do resíduo, $\mathrm{A}_H\mathrm{v}_H = \mathrm{f}_H$, na malha Ω_H;
6: **senão**
7: $\mathrm{v}_H \longleftarrow 0$;
8: $\mathrm{v}_H \longleftarrow$ CICLOV($\mathrm{v}_H, \mathrm{f}_H, \mathrm{A}_H$);
9: **fim**
10: $\mathrm{v}_h \longleftarrow \mathrm{v}_h + \mathrm{P}\mathrm{v}_H$;
11: pós-suavizar η_2 vezes o sistema $\mathrm{A}_h\mathrm{u}_h = \mathrm{f}_h$, com aproximação inicial v_h;
12: **retornar** v_h;
13: **fim**

O ciclo V é apenas um dos membros de uma família mais geral de ciclos do *multigrid*. Como as malhas mais grossas possuem menos elementos, o trabalho realizado nelas é mais simples e rápido, de modo que pode ser uma boa ideia ficarmos mais tempo nos níveis de malhas mais grossas e obter aproximações melhores com menos repetições do ciclo. Desta forma, temos a família de ciclos μ, que podem ser definidos recursivamente de maneira similar ao algoritmo do ciclo V, com a única diferença na etapa da linha 8, que é realizada μ vezes, sendo $\mu \geq 1$ um número inteiro.

Na prática, costuma-se utilizar apenas $\mu = 1$ (que gera o ciclo V) e $\mu = 2$ (que gera o ciclo W). Os nomes dados aos ciclos fazem sentido quando olhamos para o caminho percorrido por cada um deles na sequência de malhas, conforme ilustra a Figura 3.19. Os pontos mais acima da figura representam o que ocorre na malha mais fina. Quanto mais baixo estiver o ponto, mais grossa é a malha. As linhas representam o caminho percorrido entre as malhas em uma iteração do ciclo.



(a) Ciclo V (b) Ciclo W

Figura 3.19: Esquema dos ciclos do *multigrid* com 4 níveis de malha.

3.2.7 *Multigrid* como precondicionador

O método *multigrid* pode ser utilizado diretamente para resolver o sistema linear ou também como um precondicionador de outro método iterativo, como o método dos gradientes conjugados.

Se a matriz A do sistema linear for mal condicionada, o método dos gradientes conjugados pode convergir muito lentamente. Uma forma de corrigir esse problema é transformar o sistema $\mathrm{Au} = \mathrm{f}$ em um sistema equivalente, cuja matriz dos coeficientes seja melhor condicionada do que A. Dada uma matriz M invertível que aproxima a matriz A, que seja simples de calcular e com a qual seja mais fácil resolver um sistema linear, consideramos o sistema equivalente

$$\mathrm{M}^{-1}\mathrm{Au} = \mathrm{M}^{-1}\mathrm{f}.$$

Escolhendo a matriz M simétrica definida positiva, podemos ainda utilizar sua fatoração de Cholesky para obter um sistema equivalente cuja matriz continua sendo simétrica definida positiva.

O Algoritmo 3 resolve o sistema linear $\mathrm{Au} = \mathrm{f}$ por meio do método dos gradientes conjugados com precondicionamento.

Algoritmo 3 Gradientes conjugados com precondicionamento

1: $\mathrm{r} \longleftarrow \mathrm{f} - \mathrm{Au}$;
2: $\mathrm{s} \longleftarrow \mathrm{M}^{-1}\mathrm{r}$;
3: $\nu \longleftarrow \mathrm{r}^T\mathrm{s}$;
4: $\mathrm{d} \longleftarrow \mathrm{s}$;
5: $k \longleftarrow 1$;
6: **enquanto** algum critério de parada não é satisfeito **faça**
7: $\quad \mathrm{q} \longleftarrow \mathrm{Ad}$;
8: $\quad \alpha \longleftarrow \nu/(\mathrm{d}^T\mathrm{q})$;
9: $\quad \mathrm{u} \longleftarrow \mathrm{u} + \alpha\mathrm{d}$;
10: $\quad \mathrm{r} \longleftarrow \mathrm{r} - \alpha\mathrm{q}$;
11: $\quad \mathrm{s} \longleftarrow \mathrm{M}^{-1}\mathrm{r}$;
12: $\quad \gamma \longleftarrow \mathrm{r}^T\mathrm{s}$;
13: $\quad \beta \longleftarrow \gamma/\nu$;
14: $\quad \mathrm{d} \longleftarrow \mathrm{s} + \beta\mathrm{d}$;
15: $\quad \nu \longleftarrow \gamma$;
16: $\quad k \longleftarrow k + 1$;
17: **fim**

O precondicionamento aparece nas linhas 2 e 11 do algoritmo, nas quais precisamos resolver um sistema linear Ms = r. Para utilizar o *multigrid* como precondicionador, basta substituir essas linhas e obter o vetor s fazendo uma iteração de algum ciclo do *multigrid* para o sistema As = r.

Em nossa implementação, utilizamos o *multigrid* como precondicionador do método dos gradientes conjugados e o critério de parada empregado foi a norma relativa do resíduo menor do que uma tolerância preestabelecida.

3.3 *Multigrid* algébrico clássico

No método *multigrid* padrão, apresentado na Seção 3.2, também chamado de *multigrid geométrico* (GMG), utilizamos as informações geométricas do problema para construir as componentes do método. Dito de outra maneira, conhecemos uma sequência de malhas previamente definidas, nas quais sabemos as posições de cada nó, que são numerados e organizados de forma regular, e utilizamos essas informações para construir os operadores de restrição e de prolongação. Com isso, a aplicação do método se torna difícil ou até impossível para problemas com geometrias complexas ou nos quais só temos disponíveis as equações do sistema linear. Além disso, ficamos limitados a utilizar discretizações nas quais as quantidades de elementos em cada direção são potências de 2, para que possamos construir as malhas mais grossas.

Felizmente, é possível aplicar as técnicas de *multigrid* para problemas em que não temos uma malha definida, ou ainda em que a malha é irregular ou não estruturada, por meio da formulação algébrica do método. Nesta seção, apresentamos a forma clássica do método *multigrid algébrico* (AMG), conforme descrita por Stüben [49] e, posteriormente, estudamos uma versão mais recente do método (Franceschini *et al.* [20]), que promete ser mais eficaz no caso dos problemas estruturais.

Para tanto, precisamos redefinir alguns conceitos e saber como construir as componentes do método: uma sequência de malhas, um suavizador, os operadores de transferência entre malhas (prolongação e restrição) e as versões do sistema linear nas malhas mais grossas, sem utilizar informações de uma malha geométrica, mas apenas com as informações algébricas das equações do sistema linear

$$\mathrm{Au} = \mathrm{f} \qquad \text{ou} \qquad \sum_{j=1}^{n} a_{ij} u_j = f_i, \quad i = 1, 2, ..., n.$$

Uma vez obtidas todas as componentes do método, o algoritmo de resolução será o mesmo do caso geométrico, mudando-se apenas a fase de *setup*. As maiores dificuldades do método algébrico são a obtenção das malhas e a construção dos operadores de transferência entre elas. Para simplificar, descrevemos as ideias para o algoritmo de duas malhas e a descrição dos ciclos do *multigrid* segue de maneira recursiva.

3.3.1 Malha algébrica

Podemos fazer analogias entre os conceitos do método no caso geométrico e no caso algébrico, ainda utilizando termos geométricos para simplificar o entendimento. Começamos definindo o que seria uma "malha" no contexto algébrico.

No caso geométrico, conhecemos a localização e a numeração dos nós na malha, e associamos cada incógnita u_i do sistema linear ao nó i. Então, selecionamos um subconjunto de nós pertencentes à malha grossa e, consequentemente, um subconjunto de incógnitas é utilizado para representar a solução na malha grossa. Seguindo uma ideia similar, no caso algébrico, a malha é definida como sendo o conjunto de índices, $\Omega = \{1, 2, ..., n\}$, das incógnitas $u_1, u_2, ..., u_n$ do sistema. A malha grossa é escolhida como um subconjunto dos índices em Ω, particionando esse conjunto em dois subconjuntos disjuntos,

$$\Omega = \mathcal{C} \cup \mathcal{F},$$

em que $\mathcal{C}$ contém os índices dos nós na malha grossa e $\mathcal{F}$ contém os índices dos nós que estão apenas na malha fina.

Em alguns problemas, cada nó da malha (geométrica) está associado a mais de uma incógnita do sistema linear. Na otimização topológica tridimensional, por exemplo, sabemos que a solução do sistema fornece os deslocamentos nas três direções coordenadas para cada nó. As incógnitas u_1, u_2 e u_3 correspondem aos deslocamentos nas direções x, y e z, respectivamente, do primeiro nó, e assim por diante. Todavia, no caso algébrico, precisamos trabalhar com cada incógnita separadamente, de modo que cada nó $i \in \Omega$ está associado a uma única incógnita u_i do sistema linear.

As conexões entre os nós da malha algébrica são determinadas pelas entradas não nulas da matriz de coeficientes do sistema. Isso pode ser representado por um grafo de adjacência: associamos os vértices do grafo com os nós da malha e desenhamos uma aresta ligando os vértices i e j se $a_{ij} \neq 0$ ou $a_{ji} \neq 0$, conforme ilustra a Figura 3.20.

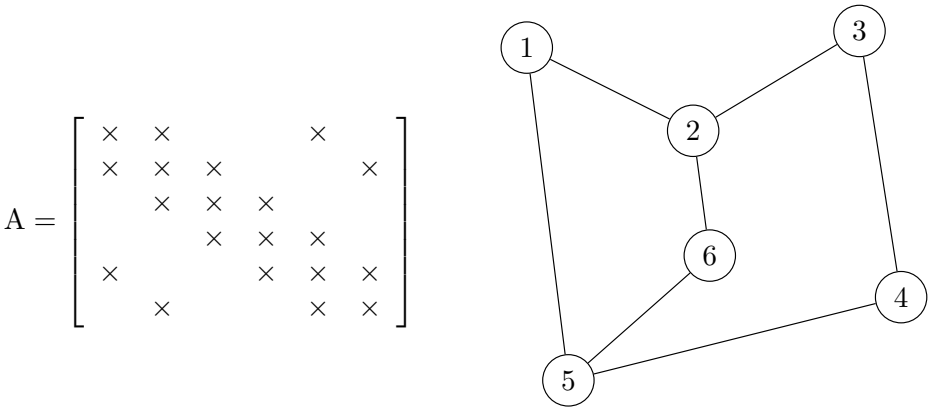


Figura 3.20: Estrutura esparsa de uma matriz A e seu grafo de adjacência.

Dizemos que um nó i está conectado a um outro nó j se a entrada a_{ij} for diferente de zero. Ademais, definimos a vizinhança de um nó i na malha como sendo o subconjunto de todos os nós que estão conectados a ele, isto é,

$$N_i = \{j \in \Omega \mid a_{ij} \neq 0,\ j \neq i\}.$$

Desta forma, toda informação da malha algébrica pode ser obtida olhando apenas para as entradas da matriz A. Quando nos referimos a algum nó $i \in \Omega$, estamos também nos referindo à incógnita u_i ou à equação i do sistema linear $\mathrm{Au} = \mathrm{f}$.

3.3.2 Conexões fortes

Vimos que as conexões entre os nós da malha algébrica são dadas pelas entradas não nulas da matriz de coeficientes do sistema linear. Um outro conceito importante para o *multigrid* algébrico é o de conexões fortes, que será fundamental na escolha da malha grossa e na construção dos operadores de transferência entre as malhas.

Considerando a i-ésima equação do sistema linear e a incógnita u_i, queremos saber quais outras incógnitas u_j, com $j \neq i$, são mais importantes na equação i para determinar u_i. Uma maneira de definir esse conceito de importância é baseada na observação de que, se o coeficiente a_{ij} (que multiplica u_j na equação i) for grande em relação aos demais coeficientes dessa equação, então uma pequena mudança no valor de u_j pode ter mais efeito no valor de u_i do que pequenas mudanças nos valores das outras incógnitas. Isso motiva a seguinte definição.

Dada uma tolerância $\theta^- \in (0, 1]$, dizemos que o nó i *depende fortemente negativamente* do nó j se

$$-a_{ij} \geq \theta^- \max_{a_{ik}<0} \{|a_{ik}|\} .$$

De forma similar, dada uma tolerância $\theta^+ \in (0, 1]$, dizemos que o nó i *depende fortemente positivamente* do nó j se

$$a_{ij} \geq \theta^+ \max_{k \neq i} \{|a_{ik}|\} .$$

Quando dizemos apenas que i *depende fortemente* de j, estamos nos referindo à dependência negativa ou positiva.

Essa definição sugere que um nó j da malha é mais importante na determinação de valores para um nó i se a entrada a_{ij} da matriz A tem magnitude comparável com a maior entrada fora da diagonal na linha i de A. Se o nó i depende fortemente do nó j, também dizemos que o nó j *influencia fortemente* o nó i. Ademais, dizemos que i e j possuem uma *conexão forte*.

3.3.3 Suavidade algébrica

Uma iteração do *multigrid* em duas malhas é composta por duas etapas principais: a suavização e a correção na malha grossa. Na suavização, aplicamos um método iterativo estacionário para a resolução do sistema linear e, após algumas iterações, notamos que as componentes oscilatórias do erro são reduzidas, fazendo com que ele se torne suave. Porém, o método falha ao tentar reduzir as componentes suaves do erro e a convergência se torna lenta. Então, a ideia é tentar acelerar a convergência por meio da mudança para uma malha mais grossa, onde os erros suaves parecem mais oscilatórios e as iterações se tornam mais eficientes.

No contexto geométrico, a ideia de suavidade para os erros tem relação com a escolha de malhas. Um erro suave deve ser bem aproximado em uma malha mais grossa predefinida. A escolha da malha grossa é feita de forma simples, geralmente dividindo a quantidade de elementos em cada direção por dois. Uma vez fixadas as malhas, ajustamos a etapa de suavização para obtermos uma melhor eficácia no método.

No caso algébrico, fixamos um método iterativo (em geral, Jacobi ou Gauss-Seidel) e, a partir dele, definimos o conceito de suavidade. A escolha da malha grossa

se torna mais complicada e precisa ser feita de modo que um erro suave possa ser bem aproximado nela. Essa escolha e a etapa de correção na malha grossa precisam ser bem elaboradas para que o método se torne eficaz.

Uma propriedade importante do erro suave no caso geométrico é que ele não consegue ser reduzido pelo método iterativo estacionário. Assim, por analogia, dizemos que um erro é *algebricamente suave* se ele não for efetivamente reduzido pelo método estacionário. Neste caso, um erro algebricamente suave pode não ser suave no sentido geométrico (veja um exemplo em [10], página 140).

Em geral, os métodos iterativos estacionários são esquemas de relaxação que podem ser descritos de forma matricial por um operador

$$\mathrm{S} = \mathrm{I} - \omega \mathrm{M}^{-1}\mathrm{A},$$

em que ω é o parâmetro de relaxação e M é a chamada matriz de iteração. Dada uma aproximação $\mathrm{v}^{(k)}$ para a solução do sistema $\mathrm{Au} = \mathrm{f}$, obtemos uma nova aproximação por

$$\mathrm{v}^{(k+1)} = \mathrm{v}^{(k)} + \omega \mathrm{M}^{-1}\left(\mathrm{f} - \mathrm{A}\mathrm{v}^{(k)}\right).$$

Por exemplo, no método de Jacobi, $\mathrm{M} = \mathrm{D}$, sendo D a matriz diagonal com a mesma diagonal de A. No método de Gauss-Seidel, $\mathrm{M} = \mathrm{D} + \mathrm{L}$, em que L é a matriz triangular com a parte estritamente triangular inferior de A, conforme vimos na Seção 3.1.

Sejam u a solução exata do sistema linear e $\mathrm{v}^{(k)}$ uma aproximação obtida na iteração k de um método de relaxação. O erro nesta iteração é dado por $e^{(k)} = \mathrm{u} - \mathrm{v}^{(k)}$. Assim, o erro na iteração seguinte satisfaz a relação

$$\begin{aligned} e^{(k+1)} &= \mathrm{u} - \mathrm{v}^{(k+1)} = \mathrm{u} - \mathrm{v}^{(k)} - \omega \mathrm{M}^{-1}\left(\mathrm{f} - \mathrm{A}\mathrm{v}^{(k)}\right) \\ &= e^{(k)} - \omega \mathrm{M}^{-1}\left(\mathrm{Au} - \mathrm{A}\mathrm{v}^{(k)}\right) = e^{(k)} - \omega \mathrm{M}^{-1}\mathrm{A}e^{(k)} \\ &= \left(\mathrm{I} - \omega \mathrm{M}^{-1}\mathrm{A}\right) e^{(k)} = \mathrm{S}e^{(k)}. \end{aligned}$$

Se o método começa a convergir muito lentamente, ou seja, se as aproximações $\mathrm{v}^{(k)}$ e $\mathrm{v}^{(k+1)}$ ficam muito próximas, então o erro também não se altera e, nesse caso, dizemos que ele é suave. Logo, um erro e é algebricamente suave se

$$\mathrm{S}e \approx e.$$

Utilizando a norma definida pela matriz A, que é simétrica definida positiva, um erro algebricamente suave é caracterizado por

$$\|\mathrm{S}e\|_{\mathrm{A}} \approx \|e\|_{\mathrm{A}}.$$

De acordo com Stüben [49], dizemos que o operador de relaxação S satisfaz a *propriedade de suavização* com respeito à matriz A se

$$\|\mathrm{S}e\|_{\mathrm{A}}^2 \leq \|e\|_{\mathrm{A}}^2 - \sigma \|\mathrm{A}e\|_{\mathrm{D}^{-1}}^2, \tag{3.5}$$

em que D é a matriz diagonal com a mesma diagonal de A, para uma constante $\sigma > 0$ independente de e. Dizemos que S satisfaz a propriedade de suavização com respeito a uma classe de matrizes, $\mathcal{A}$, se (3.5) vale uniformemente para toda matriz $\mathrm{A} \in \mathcal{A}$, isto é, com o mesmo σ. É possível verificar que os métodos de Jacobi e Gauss-Seidel satisfazem

a propriedade de suavização uniformemente para a classe de matrizes dos problemas de interesse para o *multigrid* (veja [49], Subseção A.3.2).

Essa definição implica que S é eficiente em reduzir o erro se $\|\mathrm{A}e\|_{\mathrm{D}^{-1}}$ for relativamente grande em comparação com $\|e\|_{\mathrm{A}}$. Entretanto, S se torna ineficiente quando $\|\mathrm{A}e\|_{\mathrm{D}^{-1}} << \|e\|_{\mathrm{A}}$. Então, um erro e algebricamente suave satisfaz

$$\|\mathrm{A}e\|^2_{\mathrm{D}^{-1}} << \|e\|^2_{\mathrm{A}} \Leftrightarrow e^T\mathrm{A}^T\mathrm{D}^{-1}\mathrm{A}e << e^T\mathrm{A}e.$$

Em termos do resíduo, $\mathrm{r} = \mathrm{A}e$, temos que

$$\mathrm{r}^T\mathrm{D}^{-1}\mathrm{r} << e^T\mathrm{r}.$$

Escrevendo essa desigualdade em função das componentes dos vetores, obtemos

$$\sum_{i=1}^{n}\frac{r_i^2}{a_{ii}} << \sum_{i=1}^{n} r_i e_i.$$

Com isso, conclui-se que, pelo menos em média, temos $|r_i| << a_{ii}\,|e_i|$. Então, $r_i \approx 0$, logo

$$r_i = a_{ii}e_i + \sum_{j\neq i} a_{ij}e_j \approx 0 \Leftrightarrow a_{ii}e_i \approx -\sum_{j\neq i} a_{ij}e_j.$$

Portanto, se e for um erro algebricamente suave, algumas componentes e_i podem ser aproximadas por uma média ponderada das componentes "vizinhas". Esta caracterização algébrica dos erros suaves é importante na construção dos operadores de transferência entre malhas.

M-matrizes

Outras propriedades interessantes do erro algebricamente suave podem ser vistas no caso em que A é uma *M-matriz*, isto é, uma matriz simétrica definida positiva com todas as entradas fora da diagonal menores ou iguais a zero.

Seja $\mathrm{A} : n \times n$ simétrica definida positiva e D a matriz diagonal com a mesma diagonal de A. Então, para todo $\mathrm{x} \in \mathbb{R}^n$, vale

$$\|\mathrm{x}\|^2_{\mathrm{A}} \leq \|\mathrm{x}\|_{\mathrm{D}} \|\mathrm{Ax}\|_{\mathrm{D}^{-1}}.$$

De fato, dado $\mathrm{x} \in \mathbb{R}^n$, temos

$$\begin{aligned}\|\mathrm{x}\|^2_{\mathrm{A}} &= \mathrm{x}^T\mathrm{Ax} = \mathrm{x}^T\mathrm{AD}^{-\frac{1}{2}}\mathrm{D}^{\frac{1}{2}}\mathrm{x}\\ &= \left(\mathrm{D}^{-\frac{1}{2}}\mathrm{Ax}\right)^T\left(\mathrm{D}^{\frac{1}{2}}\mathrm{x}\right) \leq \left\|\mathrm{D}^{-\frac{1}{2}}\mathrm{Ax}\right\|_2 \left\|\mathrm{D}^{\frac{1}{2}}\mathrm{x}\right\|_2\\ &= \sqrt{\mathrm{x}^T\mathrm{AD}^{-\frac{1}{2}}\mathrm{D}^{-\frac{1}{2}}\mathrm{Ax}}\,\sqrt{\mathrm{x}^T\mathrm{D}^{\frac{1}{2}}\mathrm{D}^{\frac{1}{2}}\mathrm{x}} = \|\mathrm{x}\|_{\mathrm{D}} \|\mathrm{Ax}\|_{\mathrm{D}^{-1}},\end{aligned}$$

onde utilizamos a desigualdade de Cauchy-Schwarz.

Considere que e é um erro algebricamente suave. Então, como vimos anteriormente, $\|\mathrm{A}e\|_{\mathrm{D}^{-1}} << \|e\|_{\mathrm{A}}$. Pelo resultado acima, obtemos

$$\|e\|^2_{\mathrm{A}} \leq \|e\|_{\mathrm{D}} \|\mathrm{A}e\|_{\mathrm{D}^{-1}} << \|e\|_{\mathrm{D}} \|e\|_{\mathrm{A}}.$$

Assim,

$$\|e\|_{\mathrm{A}} << \|e\|_{\mathrm{D}}\,.$$

Além disso, pode-se verificar que essa desigualdade é equivalente a

$$\frac{1}{2}\sum_{i,j}(-a_{ij})(e_i - e_j)^2 + \sum_{i,j} a_{ij}{e_i}^2 << \sum_i a_{ii}{e_i}^2.$$

Supondo agora que A seja uma M-matriz, ou seja, $|a_{ij}| = -a_{ij}$ para $i \neq j$, e que, para i fixo, $\sum_{j\neq i} |a_{ij}| \approx a_{ii}$, então para um erro algebricamente suave vale

$$\sum_{j\neq i}\left(\frac{|a_{ij}|}{a_{ii}}\right)\left(\frac{e_i - e_j}{e_i}\right)^2 << 1, \qquad i = 1, 2, ..., n.$$

Os produtos no somatório envolvem termos não negativos. Logo, um desses termos ou ambos devem ser pequenos para que a desigualdade seja válida. Se o nó i depende fortemente do nó j, então $|a_{ij}|$ é comparável com a_{ii}, de modo que o primeiro termo no produto não pode ser pequeno. Assim, para um nó i que depende fortemente do nó j, devemos ter $e_i - e_j$ próximo de zero, isto é, $e_i \approx e_j$.

Com isso, dizemos que um erro algebricamente suave varia lentamente na direção de conexões fortes. Isso fornece uma justificativa para a ideia intuitiva de que um valor e_i na malha fina possa ser prolongado a partir de valores e_j na malha grossa, tais que i dependa fortemente de j.

Segundo Stüben [49], se A não for uma M-matriz, mas possuir poucas entradas positivas fora da diagonal, pode-se ainda concluir que um erro algebricamente suave varia lentamente na direção de conexões fortes (negativas ou positivas). Agora, no caso em que há muitas conexões fortes positivas, em particular no caso em que a matriz quase não é diagonalmente dominante, obtém-se a relação

$$\sum_{j\neq i}\left(\frac{|a_{ij}^-|}{a_{ii}}\right)\left(\frac{e_i - e_j}{e_i}\right)^2 + \sum_{j\neq i}\left(\frac{a_{ij}^+}{a_{ii}}\right)\left(\frac{e_i + e_j}{e_i}\right)^2 << 1,$$

em que a_{ij}^- representa uma entrada negativa da matriz e a_{ij}^+ uma entrada positiva. Neste caso, um erro suave varia lentamente na direção de conexões fortes negativas, pois $e_i \approx e_j$. Entretanto, $e_j \approx -e_i$ se $a_{ij} > 0$, ou seja, um erro suave tende a variar bastante ao longo de conexões fortes positivas. Por este motivo, as conexões fortes são separadas em negativas e positivas, e precisamos tomar um certo cuidado ao trabalhar com problemas que possuam muitas conexões fortes positivas.

3.3.4 Operador de prolongação

Suponha que já tenhamos feito a escolha da malha grossa e que disponhamos da partição dos índices $\Omega = \mathcal{C} \cup \mathcal{F}$, em que $\mathcal{C}$ contém os índices dos nós na malha grossa e $\mathcal{F}$ contém os índices dos nós que estão apenas na malha fina. Para cada nó i na malha fina, consideramos a vizinhança N_i como sendo os nós $j \neq i$ tais que $a_{ij} \neq 0$. Podemos separar esses pontos em três subconjuntos:

- $\mathcal{C}_i$: vizinhos na malha grossa que influenciam fortemente o nó i,
- $\mathcal{F}_i$: vizinhos na malha fina que influenciam fortemente o nó i,
- $\mathcal{W}_i$: vizinhos que não influenciam fortemente o nó i.

Queremos definir um operador de prolongação, P, capaz de transferir de forma precisa as componentes do erro da malha grossa para a malha fina. Em geral, como no caso geométrico, se o nó i está na malha grossa, então a componente i do erro se mantém. Caso contrário, a componente i do erro será uma média ponderada das componentes vizinhas na malha grossa que influenciam fortemente i. Assim,

$$(\mathrm{P}e)_i = \begin{cases} e_i, & \text{se } i \in \mathcal{C}, \\ \displaystyle\sum_{k \in \mathcal{C}_i} w_{ik} e_k, & \text{se } i \in \mathcal{F}. \end{cases}$$

Precisamos determinar quais serão os pesos w_{ik} e existem diversas alternativas, cada qual associada a um operador de prolongação.

Vimos que as componentes de um erro algebricamente suave podem ser caracterizadas em termos do resíduo por $\mathrm{r}_i \approx 0$, o que implica que

$$a_{ii} e_i \approx -\sum_{j \in N_i} a_{ij} e_j.$$

Separando o somatório nos três subconjuntos de N_i definidos anteriormente, temos

$$a_{ii} e_i \approx -\sum_{j \in \mathcal{C}_i} a_{ij} e_j - \sum_{j \in \mathcal{F}_i} a_{ij} e_j - \sum_{j \in \mathcal{W}_i} a_{ij} e_j. \tag{3.6}$$

Para obter os pesos w_{ik} precisamos reformular os dois últimos somatórios, substituindo e_j por aproximações em termos de e_i ou de e_k para $k \in \mathcal{C}_i$, uma vez que só conhecemos os valores nos nós da malha grossa.

Alternativa 1

Assumindo que A é uma M-matriz (possui apenas conexões fortes negativas) ou que possui poucas conexões fortes positivas, podemos fazer a aproximação

$$\sum_{j \notin \mathcal{C}_i} a_{ij} e_j \approx \left(\sum_{j \notin \mathcal{C}_i} a_{ij} \right) e_i.$$

Ela é razoável pois, se i e j estão fortemente conectados, então $e_j \approx e_i$. Caso contrário, a entrada a_{ij} é pequena e o erro cometido na aproximação pode ser desprezado.

Assim, combinamos os somatórios em $\mathcal{F}_i$ e em $\mathcal{W}_i$ com o termo do lado esquerdo em (3.6), obtendo

$$\left(a_{ii} + \sum_{j \in \mathcal{F}_i} a_{ij} + \sum_{j \in \mathcal{W}_i} a_{ij} \right) e_i \approx -\sum_{j \in \mathcal{C}_i} a_{ij} e_j.$$

Dessa forma, calculamos os pesos

$$w_{ik} = -\frac{a_{ik}}{\left(a_{ii} + \displaystyle\sum_{j \in \mathcal{F}_i} a_{ij} + \sum_{j \in \mathcal{W}_i} a_{ij} \right)}.$$

Alternativa 2

Conforme proposto por Briggs *et al.* [10], utilizando a mesma aproximação da alternativa anterior, podemos combinar apenas o somatório em $\mathcal{W}_i$ com o termo da diagonal em (3.6), obtendo

$$\left(a_{ii} + \sum_{j\in\mathcal{W}_i} a_{ij}\right) e_i \approx -\sum_{j\in\mathcal{C}_i} a_{ij}e_j - \sum_{j\in\mathcal{F}_i} a_{ij}e_j.$$

Os dois somatórios restantes do lado direito, que envolvem conexões fortes, são combinados aproximando e_j, em que j é um nó da malha fina que influencia fortemente i, por uma combinação de valores e_l, com $l \in \mathcal{C}_i$. Isso é feito considerando que, para cada $j \in \mathcal{F}_i$,

$$e_j \approx \frac{\sum_{l\in\mathcal{C}_i} a_{jl}e_l}{\sum_{l\in\mathcal{C}_i} a_{jl}}.$$

Ou seja, e_j é uma média ponderada dos valores e_l, com $l \in \mathcal{C}_i$. Substituindo e_j no somatório em $\mathcal{F}_i$ e reorganizando os termos em (3.6), obtemos os pesos

$$w_{ik} = -\frac{a_{ik} + \sum_{j\in\mathcal{F}_i} \left(\frac{a_{ij}a_{jk}}{\sum_{l\in\mathcal{C}_i} a_{jl}}\right)}{\left(a_{ii} + \sum_{j\in\mathcal{W}_i} a_{ij}\right)}.$$

Embora essas duas alternativas sejam elaboradas no caso em que A é uma M-matriz, elas também podem ser aplicadas para problemas mais gerais. Porém, as aproximações podem ser imprecisas quando temos muitas conexões fortes positivas e o método *multigrid* fica ineficiente ao utilizar essas prolongações.

Alternativa 3

Como proposto por Stüben [49], podemos também combinar os três somatórios em (3.6) obtendo um único somatório que envolve valores e_k, para $k \in \mathcal{C}_i$.

No caso em que A é uma M-matriz, consideramos a aproximação

$$\frac{1}{\sum_{k\in\mathcal{C}_i} a_{ik}} \sum_{k\in\mathcal{C}_i} a_{ik}e_k \approx \frac{1}{\sum_{j\in N_i} a_{ij}} \sum_{j\in N_i} a_{ij}e_j. \tag{3.7}$$

Então, obtemos

$$a_{ii}e_i \approx -\alpha_i \sum_{k\in\mathcal{C}_i} a_{ik}e_k, \qquad \text{com } \alpha_i = \frac{\sum_{j\in N_i} a_{ij}}{\sum_{k\in\mathcal{C}_i} a_{ik}}.$$

Com isso, calculamos os pesos

$$w_{ik} = -\alpha_i \frac{a_{ik}}{a_{ii}}.$$

No caso em que a matriz A possui poucas conexões positivas, podemos combinar os coeficientes $a_{ij} > 0$ com a diagonal, como nas alternativas anteriores, utilizando a aproximação

$$\sum_j a_{ij}^+ e_j \approx \left(\sum_j a_{ij}^+ \right) e_i.$$

Já no caso geral, em que existem conexões fortes positivas e negativas, a aproximação acima pode não ser boa, pois vimos que $e_j \approx -e_i$ quando $a_{ij} > 0$. Nesta situação, consideramos que aproximações como (3.7) são satisfeitas separadamente para as entradas positivas, a_{ij}^+, e para as entradas negativas, a_{ij}^-. Então, obtemos

$$a_{ii} e_i \approx -\alpha_i \sum_{k \in \mathcal{C}_i} a_{ik}^- e_k - \beta_i \sum_{k \in \mathcal{C}_i} a_{ik}^+ e_k,$$

com

$$\alpha_i = \frac{\sum_{j \in N_i} a_{ij}^-}{\sum_{k \in \mathcal{C}_i} a_{ik}^-} \qquad \text{e} \qquad \beta_i = \frac{\sum_{j \in N_i} a_{ij}^+}{\sum_{k \in \mathcal{C}_i} a_{ik}^+}.$$

Assim, temos os pesos

$$w_{ik} = \begin{cases} -\alpha_i a_{ik}/a_{ii}, & \text{se } k \in \mathcal{C}_i^-, \\ -\beta_i a_{ik}/a_{ii}, & \text{se } k \in \mathcal{C}_i^+, \end{cases}$$

em que $\mathcal{C}_i^-$ e $\mathcal{C}_i^+$ são os vizinhos na malha grossa que influenciam fortemente negativamente o nó i e que influenciam fortemente positivamente o nó i, respectivamente.

Estas são algumas alternativas para a prolongação de forma direta. Existem ainda outras estratégias de construção do operador de prolongação, que podem ser melhores no caso em que temos muitas conexões positivas (veja [33], por exemplo).

3.3.5 Escolha da malha grossa

No caso geométrico, para obter uma malha mais grossa podemos simplesmente agrupar alguns elementos próximos na malha fina, formando elementos maiores. Sem as informações geométricas, a escolha da malha grossa se torna mais complicada e é uma etapa essencial para o bom funcionamento do *multigrid* algébrico.

Novamente, vamos utilizar os conceitos de conexões fortes e de suavidade, como feito na construção do operador de prolongação. Precisamos escolher uma malha grossa na qual um erro suave possa ser bem aproximado, da qual os erros suaves possam ser prolongados de forma precisa e que possua uma quantidade de nós substancialmente menor do que a malha fina.

Para que as aproximações feitas na construção do operador de prolongação sejam mais precisas é necessário que os nós da malha fina possuam uma quantidade suficiente de conexões fortes com os nós da malha grossa. O objetivo é garantir que isso aconteça, ao mesmo tempo em que a quantidade de nós na malha grossa seja pequena.

Sejam S_i o conjunto dos nós que influenciam fortemente i e S_i^T o conjunto dos nós que dependem fortemente do nó i. Faremos a separação do conjunto de índices, Ω, em

dois subconjuntos disjuntos, $\mathcal{C}$ (de nós da malha grossa) e $\mathcal{F}$ (de nós que só pertencem à malha fina), considerando os seguintes critérios heurísticos:

H1: Para cada $i \in \mathcal{F}$, todo nó $j \in S_i$ deve estar na malha grossa e ser vizinho de i (isto é, $j \in \mathcal{C}_i$) ou deve depender fortemente de pelo menos um nó em $\mathcal{C}_i$.

H2: O subconjunto $\mathcal{C}$ deve ser o maior possível de modo que não existam conexões fortes entre seus pontos.

Como vimos na subseção anterior, o valor do erro em um nó $i \in \mathcal{F}$ será prolongado a partir dos valores nos nós $j \in \mathcal{C}_i$. Além disso, as aproximações feitas na construção do operador de prolongação serão mais precisas quanto mais conexões fortes existirem entre nós $j \in S_i$ e nós em $\mathcal{C}_i$. A heurística H1 garante a existência de pelo menos uma destas conexões. A heurística H2 garante que o subconjunto $\mathcal{C}$ seja grande o suficiente para se ter um bom fator de convergência no método, mas também controla o tamanho de $\mathcal{C}$ para que a resolução do sistema na malha grossa possa ser barata.

Na prática, nem sempre é possível garantir que as duas heurísticas sejam satisfeitas simultaneamente. O usual é garantir que H1 seja satisfeita, para que possamos aplicar as fórmulas de prolongação com maior precisão, e utilizar H2 como um guia para não produzirmos um subconjunto $\mathcal{C}$ muito grande.

A seguir, apresentamos um processo clássico de seleção da malha grossa, descrito por Stüben [49], no qual considera-se que existem apenas conexões fortes negativas entre os nós da malha fina. Todas as conexões positivas são consideradas fracas por enquanto. Realizamos uma partição dos subconjuntos $\mathcal{C}$ e $\mathcal{F}$ de tal forma que todos os nós em $\mathcal{F}$ possuam pelo menos uma conectividade forte negativa com os vizinhos em $\mathcal{C}$.

Seja U o conjunto dos nós que ainda não foram escolhidos para estar em $\mathcal{C}$ ou em $\mathcal{F}$. Inicialmente, $U = \Omega$. Para cada $i \in U$, definimos uma medida de importância, λ_i, que será utilizada para decidir se i irá para o subconjunto $\mathcal{C}$. Uma medida usual é a quantidade de nós que são fortemente influenciados por i, ou seja,

$$\lambda_i = |S_i^T|.$$

Escolhemos um nó $i \in U$ com o maior λ_i positivo para ser um nó da malha grossa, isto é, retiramos i de U e o acrescentamos em $\mathcal{C}$. O nó escolhido influencia fortemente outros nós e deve aparecer na fórmula de prolongação para cada um deles. Assim, os nós $j \in S_i^T$ são selecionados para estarem apenas na malha fina, ou seja, para todo $j \in S_i^T$, retiramos j de U e acrescentamos em $\mathcal{F}$. Agora, olhamos para os outros nós (diferentes de i) que influenciam fortemente esse nós que acabaram de entrar em $\mathcal{F}$. Estes são possíveis candidatos a também estarem em $\mathcal{C}$, pois os seus valores podem ser úteis para uma prolongação mais precisa. Desta forma, para cada $j \in S_i^T$ aumentamos em 1 a medida λ_k para cada $k \in S_j \cap U$.

Em seguida, escolhemos um novo nó com máximo λ_i positivo para ser um nó da malha grossa e o procedimento se repete até que todos os nós em U tenham se tornado nós em $\mathcal{C}$ ou em $\mathcal{F}$, ou até que $\max\{\lambda_i\}$ seja zero.

Note que pode haver mais de um nó com máximo λ_i em cada etapa e a escolha do nó que irá para o subconjunto $\mathcal{C}$ pode gerar diferentes malhas grossas. Uma estratégia

comum é incluirmos em $\mathcal{C}$ o primeiro nó (de menor índice) com máximo λ_i. O Algoritmo 4 descreve o processo de escolha da malha grossa.

Em alguns problemas, particularmente os não simétricos, pode acontecer de sobrarem nós em U com $\lambda_i = 0$. Segundo Stüben [49], neste caso, colocamos esses nós no subconjunto $\mathcal{F}$ e seus valores são prolongados de forma indireta, utilizando outros nós de $\mathcal{F}$. Felizmente, notamos que esta situação não ocorre no problema de otimização topológica estrutural.

Algoritmo 4 Escolha da malha grossa algébrica (*coarsening*)

1: $U \longleftarrow \Omega$; $\mathcal{C} \longleftarrow \emptyset$; $\mathcal{F} \longleftarrow \emptyset$;
2: para cada ponto $i \in U$ calcule a medida $\lambda_i = |S_i^T|$;
3: **enquanto** $U \neq \emptyset$ e $\max\{\lambda_i\} \neq 0$ **faça**
4: escolha um ponto $i \in U$ com máximo λ_i;
5: $\mathcal{C} \longleftarrow \mathcal{C} \cup \{i\}$; $U \longleftarrow U \setminus \{i\}$;
6: **para** $j \in S_i^T \cap U$ **faça**
7: $\mathcal{F} \longleftarrow \mathcal{F} \cup \{j\}$; $U \longleftarrow U \setminus \{j\}$;
8: **para** $k \in S_j \cap U$ **faça**
9: $\lambda_k \longleftarrow \lambda_k + 1$;
10: **fim**
11: **fim**
12: **fim**

Exemplo: considere um problema bidimensional cuja matriz A do sistema linear é de ordem 25×25 e a conectividade entre um nó da malha fina e seus vizinhos é representada pela matriz

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 4 & -1 \\ -1 & -1 & -1 \end{bmatrix},$$

isto é, para cada linha i de A, temos $a_{ii} = 4$, $a_{ij} = -1$ para j vizinho de i, e as demais entradas são iguais a zero. Dessa maneira, todo nó i está fortemente negativamente conectado aos seus oito vizinhos, para qualquer tolerância $\theta^- \in (0, 1]$.

A Figura 3.21 ilustra o processo de escolha dos nós da malha grossa para este exemplo. Os números dentro dos círculos representam as medidas λ_i para cada nó da malha. Os círculos em cinza representam nós com potencial para se tornarem nós da malha grossa (que possuem máximo λ_i) em cada etapa. Os círculos preenchidos em preto representam nós do subconjunto $\mathcal{C}$ e os círculos preenchidos em branco representam nós do subconjunto $\mathcal{F}$. A ordem do processo na figura segue a mesma ordem de leitura, isto é, da esquerda para a direita e de cima para baixo.

Quando o procedimento descrito acima é finalizado, todos os nós no subconjunto $\mathcal{F}$ possuem pelo menos uma conexão forte negativa com nós de $\mathcal{C}$. Porém, no caso em que existem conexões fortes positivas, para que a prolongação seja mais precisa é necessário que os nós em $\mathcal{F}$ possuam conexões fortes de ambos os tipos com uma quantidade mínima de nós em $\mathcal{C}$. Nos problemas em que a maioria das conexões fortes é negativa, podemos utilizar a seguinte estratégia descrita em [49]: após a seleção preliminar dos subconjuntos $\mathcal{C}$ e $\mathcal{F}$ conforme explicado anteriormente, verificamos se existem conexões fortes positivas em $\mathcal{F}$. Para cada $i \in \mathcal{F}$, se existir uma conexão forte positiva entre i e pelo me-

nos um nó de $\mathcal{C}$, não precisamos fazer nada. Caso contrário, todo nó $j \in \mathcal{F}$ que influencia fortemente positivamente i será adicionado no conjunto S_i e o nó que corresponde à maior conexão positiva se tornará um nó de $\mathcal{C}$.

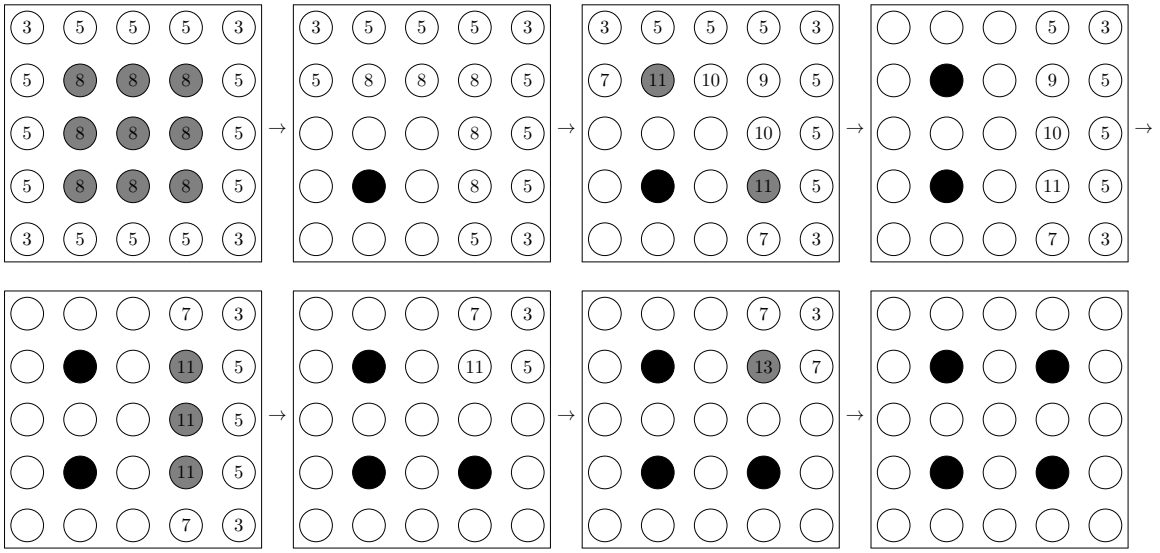


Figura 3.21: Processo de escolha dos nós da malha grossa algébrica.

3.3.6 Operadores na malha grossa

Embora não tenhamos as informações geométricas do problema, nesta subseção continuamos utilizando as notações h e H para fazer referência à malha fina e à malha grossa, respectivamente, de modo que A_h representa a matriz do sistema na malha fina e A_H representa a matriz do sistema na malha grossa. Além disso, denotamos por R e P os operadores de restrição e prolongação, respectivamente.

No contexto algébrico, definimos o operador de restrição como sendo o transposto do operador de prolongação, isto é,

$$\mathrm{R} = \mathrm{P}^T.$$

Neste caso, só precisamos construir a matriz de prolongação P. Ademais, assim como no caso geométrico, a matriz do sistema na malha grossa é dada pelo operador de Galerkin,

$$\mathrm{A}_H = \mathrm{R}\mathrm{A}_h\mathrm{P} = \mathrm{P}^T\mathrm{A}_h\mathrm{P}.$$

É possível verificar que a escolha dos operadores desta forma faz com que a etapa de correção na malha grossa minimize a norma (induzida pela matriz A_h) do erro, no seguinte sentido: para todo vetor e_h,

$$\left\| K_h^H e_h \right\|_{\mathrm{A}_h} = \min_{e_H} \left\| e_h - P e_H \right\|_{\mathrm{A}_h},$$

em que K_h^H é o operador de correção na malha grossa (veja [49], Subseção A.2.4).

Após obtermos todas as componentes do método, a fase de resolução do *multigrid* algébrico é similar à do caso geométrico. O algoritmo do ciclo em duas malhas pode ser revisto na Subseção 3.2.2 e os ciclos do *multigrid* (ciclo V, ciclo W etc.) podem ser descritos de forma recursiva, conforme a Subseção 3.2.6.

3.4 *Multigrid* algébrico adaptativo

Na seção anterior, estudamos o método *multigrid* algébrico na sua forma clássica, o qual apresenta bons resultados para diversos tipos de problemas. Contudo, devido à maneira como definimos as conexões fortes e o operador de prolongação, o método pode ser ineficiente para problemas com conexões fortes positivas, exigindo adaptações complicadas. Conforme verificamos experimentalmente, o método clássico não apresenta bons resultados na resolução dos sistemas lineares que surgem na otimização topológica. Nesta seção, apresentamos uma alternativa mais recente do *multigrid* algébrico, proposta por Magri *et al.* [33] e melhorada pelos mesmos autores em Franceschini *et al.* [20], que promete ser mais eficiente no caso de problemas estruturais.

Considere o sistema linear $\mathrm{Au} = \mathrm{f}$, com matriz $\mathrm{A} : n \times n$, que desejamos resolver pelo método *multigrid*. Seja $\mathrm{S} = \mathrm{I} - \omega \mathrm{M}^{-1}\mathrm{A}$ o suavizador. Para a aplicação do *multigrid* algébrico adaptativo, precisamos inicialmente obter um *espaço de teste* que represente os vetores algebricamente suaves, isto é, vetores $\mathrm{v} \in \mathbb{R}^n$ tais que $\mathrm{Sv} \approx \mathrm{v}$. Esses vetores podem ser aproximados pelas soluções do problema de autovalores generalizado:

$$\mathrm{Av} = \lambda \mathrm{Mv}. \tag{3.8}$$

Buscamos obter uma matriz $\mathrm{V} : n \times n_{tv}$ cujas colunas formam uma base para o espaço de teste. Em geral, essas colunas são os autovetores, soluções da equação (3.8), associados aos n_{tv} menores autovalores. Assim, podemos obter a matriz V utilizando o método das potências inverso, por exemplo. Os estudos em [33] sugeriam que o método de Lanczos fosse a melhor opção neste caso. Já em [20], os autores propõem uma adaptação do método de minimização do *quociente de Rayleigh* por gradientes conjugados, introduzido por Longsine e McCormick [31], que é mais eficiente para matrizes esparsas.

Suponha que já tenhamos a matriz V cujas colunas formam uma base para o espaço de teste. Denotamos por v_i^T a linha i da matriz V, de modo que v_i será um vetor linha em pé. Definimos a *força de conexão* entre um nó i e um nó j por

$$SoC[i,j] = \frac{(\mathrm{v}_i^T \mathrm{v}_j)^2}{(\mathrm{v}_i^T \mathrm{v}_i)(\mathrm{v}_j^T \mathrm{v}_j)}.$$

Utilizaremos essa nova definição para obter as conexões fortes entre os nós e para escolher a malha grossa, fazendo a separação $\Omega = \mathcal{C} \cup \mathcal{F}$.

3.4.1 Geração do espaço de teste

Sejam A e B matrizes $n \times n$ simétricas, definidas positivas. Para construir o espaço de teste, queremos resolver um problema de autovalores generalizado na forma

$$A\mathrm{v} = \lambda B\mathrm{v},$$

e obter uma matriz $\mathrm{V} : n \times n_{tv}$ cujas colunas são aproximações dos autovetores associados aos n_{tv} menores autovalores. A seguir, descrevemos um método cuja ideia é minimizar o quociente de Rayleigh em um subespaço ortogonal aos autovetores previamente calculados, utilizando um método de gradientes conjugados (veja [7, 8, 31] para mais detalhes).

Dados $\mathrm{v}_1, \mathrm{v}_2, ..., \mathrm{v}_{j-1}$, aproximações para os autovetores associados aos autovalores $\lambda_1 < \lambda_2 < ... < \lambda_{j-1}$, vamos obter uma aproximação para o j-ésimo autovetor

minimizando o quociente de Rayleigh no subespaço ortogonal aos $j-1$ autovetores previamente calculados. Isto é, vamos minimizar a função dada por

$$q(\mathrm{x}) = \frac{\mathrm{x}^T A \mathrm{x}}{\mathrm{x}^T B \mathrm{x}}, \tag{3.9}$$

com $\mathrm{x} = \mathrm{y} - \mathrm{V}_j(\mathrm{V}_j^T \mathrm{y})$, em que $\mathrm{V}_j = [\mathrm{v}_1, \mathrm{v}_2, ..., \mathrm{v}_{j-1}]$. O primeiro autovetor, v_1, é obtido minimizando (3.9) com $\mathrm{x} = \mathrm{y} \in \mathbb{R}^n$, ou seja, com $\mathrm{V}_1 = [\,]$ (uma matriz vazia).

Suponha que já obtivemos $\mathrm{V}_j = [\mathrm{v}_1, \mathrm{v}_2, ..., \mathrm{v}_{j-1}]$ e queremos obter v_j. Para minimizar a função q, aplicamos um método de descida do tipo gradientes conjugados. Dada uma aproximação inicial $\mathrm{y}^{(0)}$, calculamos

$$\mathrm{x}^{(0)} = \mathrm{y}^{(0)} - \mathrm{V}_j(\mathrm{V}_j^T \mathrm{y}^{(0)})$$

e vamos encontrar uma sequência de vetores $\mathrm{x}^{(0)}, \mathrm{x}^{(1)}, \mathrm{x}^{(2)}, ...$, de modo que a cada iteração tenhamos $q(\mathrm{x}^{(k+1)}) \leq q(\mathrm{x}^{(k)})$. Para ir de $\mathrm{x}^{(k)}$ até $\mathrm{x}^{(k+1)}$, escolhemos uma direção de descida $\hat{\mathrm{d}}^{(k)}$ e calculamos sua projeção no subespaço ortogonal a V_j, dada por

$$\mathrm{d}^{(k)} = \hat{\mathrm{d}}^{(k)} - \mathrm{V}_j(\mathrm{V}_j^T \hat{\mathrm{d}}^{(k)}).$$

Em seguida, fazemos uma busca linear na direção $\mathrm{d}^{(k)}$, isto é, obtemos o passo

$$\alpha_k = \operatorname{argmin} \left\{ q(\mathrm{x}^{(k)} + \alpha \mathrm{d}^{(k)}) \mid \alpha \in \mathbb{R} \right\},$$

e atualizamos o ponto

$$\mathrm{x}^{(k+1)} = \mathrm{x}^{(k)} + \alpha_k \mathrm{d}^{(k)}.$$

O processo se repete até atingirmos algum critério de parada.

No caso do método dos gradientes conjugados com precondicionador M, calculamos o vetor $\mathrm{s}^{(k)} = -M^{-1}\nabla q(\mathrm{x}^{(k)})$, escolhemos a direção inicial $\hat{\mathrm{d}}^{(0)} = \mathrm{s}^{(0)}$ e obtemos uma sequência de direções conjugadas calculando, para $k \geq 1$, o termo

$$\beta_k = \frac{\nabla q(\mathrm{x}^{(k)})^T \mathrm{s}^{(k)}}{\nabla q(\mathrm{x}^{(k-1)})^T \mathrm{s}^{(k-1)}}$$

e a direção $\hat{\mathrm{d}}^{(k)} = \mathrm{s}^{(k)} + \beta_k \mathrm{d}^{(k-1)}$.

Também é possível obter uma fórmula fechada para o passo α_k. Para simplificar a escrita, vamos omitir o índice k. Temos, então,

$$q(\mathrm{x} + \alpha \mathrm{d}) = \frac{(\mathrm{x} + \alpha \mathrm{d})^T A (\mathrm{x} + \alpha \mathrm{d})}{(\mathrm{x} + \alpha \mathrm{d})^T B (\mathrm{x} + \alpha \mathrm{d})} = \frac{\mathrm{x}^T A \mathrm{x} + 2\alpha \mathrm{d}^T A \mathrm{x} + \alpha^2 \mathrm{d}^T A \mathrm{d}}{\mathrm{x}^T B \mathrm{x} + 2\alpha \mathrm{d}^T B \mathrm{x} + \alpha^2 \mathrm{d}^T B \mathrm{d}}.$$

Definindo os valores

$$\gamma = \mathrm{x}^T A \mathrm{x}, \qquad \eta = \mathrm{x}^T B \mathrm{x},$$
$$a_1 = \mathrm{d}^T A \mathrm{x}, \quad a_2 = \mathrm{d}^T A \mathrm{d}, \quad a_3 = \mathrm{d}^T B \mathrm{x}, \quad a_4 = \mathrm{d}^T B \mathrm{d},$$

temos

$$q(\mathrm{x} + \alpha \mathrm{d}) = \frac{\gamma + 2a_1\alpha + a_2\alpha^2}{\eta + 2a_3\alpha + a_4\alpha^2}.$$

Calculando a derivada de $q(\mathrm{x} + \alpha \mathrm{d})$ com relação a α e igualando-a a zero, obtemos

$$(\eta + 2a_3\alpha + a_4\alpha^2)(2a_1 + 2a_2\alpha) - (\gamma + 2a_1\alpha + a_2\alpha^2)(2a_3 + 2a_4\alpha) = 0 \Leftrightarrow$$

$$\Leftrightarrow (a_2a_3 - a_1a_4)\alpha^2 + (a_2\eta - a_4\gamma)\alpha + (a_1\eta - a_3\gamma) = 0.$$

Denotando por $a = a_2a_3 - a_1a_4$, $b = a_2\eta - a_4\gamma$ e $c = a_1\eta - a_3\gamma$, temos

$$a\alpha^2 + b\alpha + c = 0.$$

Então, calculamos a raiz positiva desta equação polinomial de segundo grau:

$$\alpha = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \tag{3.10}$$

ou, equivalentemente,

$$\alpha = \frac{-2c}{b + \sqrt{b^2 - 4ac}}. \tag{3.11}$$

Para evitar o cancelamento subtrativo, utilizamos a fórmula (3.10) quando $b < 0$ e a fórmula (3.11) quando $b \geq 0$.

O gradiente da função q pode ser calculado por:

$$\nabla q(\mathrm{x}) = \frac{2}{\mathrm{x}^T B\mathrm{x}}\left[A\mathrm{x} - q(\mathrm{x})B\mathrm{x}\right] = \frac{2}{\eta}\mathrm{r},$$

em que $\mathrm{r} = A\mathrm{x} - q(\mathrm{x})B\mathrm{x}$. Os produtos $A\mathrm{x}$ e $B\mathrm{x}$ só precisam ser calculados para o ponto inicial, $\mathrm{x}_A^{(0)} = A\mathrm{x}^{(0)}$ e $\mathrm{x}_B^{(0)} = B\mathrm{x}^{(0)}$, sendo depois atualizados por $\mathrm{x}_A^{(k+1)} = \mathrm{x}_A^{(k)} + \alpha_k \mathrm{d}_A^{(k)}$ e $\mathrm{x}_B^{(k+1)} = \mathrm{x}_B^{(k)} + \alpha_k \mathrm{d}_B^{(k)}$, com $\mathrm{d}_A^{(k)} = A\mathrm{d}^{(k)}$ e $\mathrm{d}_B^{(k)} = B\mathrm{d}^{(k)}$. Além disso, podemos atualizar

$$\gamma_{k+1} = \gamma_k + 2a_1\alpha + a_2\alpha^2,$$
$$\eta_{k+1} = \eta_k + 2a_3\alpha + a_4\alpha^2$$

e $q(\mathrm{x}^{(k)}) = \gamma_k/\eta_k$ em cada iteração k.

Suponha que $\bar{k}$ seja o índice da última iteração, quando atingimos o critério de parada. Neste caso, teremos que $q(x^{(\bar{k})}) \approx \lambda_j$ e $\mathrm{v}_j = \mathrm{x}^{(\bar{k})}/\sqrt{\eta_{\bar{k}}}$ é uma aproximação para o autovetor associado. Assim, acrescentamos v_j na última coluna de V_j e repetimos o processo para obter o próximo autovetor, v_{j+1}. O Algoritmo 5 descreve o processo de geração do espaço de teste.

Na implementação feita neste trabalho, consideramos dois critérios de parada para o algoritmo de geração do espaço de teste: a norma relativa do resíduo menor do que uma tolerância $\varepsilon_r > 0$, isto é,

$$\frac{\left\|\mathrm{r}^{(k)}\right\|_2}{\left\|A\mathrm{x}^{(k)}\right\|_2} < \varepsilon_r,$$

e a diferença relativa entre os valores do quociente de Rayleigh em duas iterações consecutivas menor que uma outra tolerância ε_q, isto é,

$$\frac{|q_k - q_{k-1}|}{q_k} < \varepsilon_q.$$

Em geral, as tolerâncias não precisam ser muito pequenas, pois as aproximações dos autovetores podem ser imprecisas.

Com a matriz do espaço de teste V, podemos calcular a força de conexão entre um nó i e um nó j por

$$SoC[i,j] = \frac{(\mathrm{v}_i^T\mathrm{v}_j)^2}{(\mathrm{v}_i^T\mathrm{v}_i)(\mathrm{v}_j^T\mathrm{v}_j)}.$$

Então, verificamos quais destes valores são maiores do que algum parâmetro $\theta_s \in (0, 1]$ e obtemos as conexões fortes entre os nós da malha. Com isso, escolhemos a malha grossa, fazendo a separação $\Omega = \mathcal{C} \cup \mathcal{F}$ da mesma maneira que no *multigrid* algébrico clássico.

Algoritmo 5 Geração do espaço de teste

$\mathrm{V} \longleftarrow [\,]$;
para $j = 1, ..., n_{tv}$ **faça**
  escolha uma aproximação inicial $\mathrm{y}^{(0)}$;
  $\mathrm{x}^{(0)} \longleftarrow \mathrm{y}^{(0)} - \mathrm{V}(\mathrm{V}^T\mathrm{y}^{(0)})$;
  $\mathrm{x}_A^{(0)} \longleftarrow A\mathrm{x}^{(0)}$; $\mathrm{x}_B^{(0)} \longleftarrow B\mathrm{x}^{(0)}$;
  $\gamma \longleftarrow \mathrm{x}^{(0)^T}\mathrm{x}_A^{(0)}$; $\eta \longleftarrow \mathrm{x}^{(0)^T}\mathrm{x}_B^{(0)}$;
  $q_0 \longleftarrow \gamma/\eta$;
  $\mathrm{r}^{(0)} \longleftarrow \mathrm{x}_A^{(0)} - q_0\mathrm{x}_B^{(0)}$;
  $k \longleftarrow 0$;
  enquanto algum critério de parada não é satisfeito **faça**
    $\nabla q(\mathrm{x}^{(\mathrm{k})}) \longleftarrow \frac{2}{\eta}\mathrm{r}^{(k)}$;
    $\mathrm{s}^{(k)} \longleftarrow -M^{-1}\nabla q(\mathrm{x}^{(k)})$;
    se $k = 0$ **então**
      $\hat{\mathrm{d}}^{(k)} \longleftarrow \mathrm{s}^{(k)}$;
    senão
      $\beta_k \longleftarrow \dfrac{\nabla q(\mathrm{x}^{(k)})^T\mathrm{s}^{(k)}}{\nabla q(\mathrm{x}^{(k-1)})^T\mathrm{s}^{(k-1)}}$;
      $\hat{\mathrm{d}}^{(k)} \longleftarrow \mathrm{s}^{(k)} + \beta_k\mathrm{d}^{(k-1)}$;
    fim
    $\mathrm{d}^{(k)} \longleftarrow \hat{\mathrm{d}}^{(k)} - \mathrm{V}(\mathrm{V}^T\hat{\mathrm{d}}^{(k)})$;
    $\mathrm{d}_A^{(k)} \longleftarrow A\mathrm{d}^{(k)}$; $\mathrm{d}_B^{(k)} \longleftarrow B\mathrm{d}^{(k)}$;
    $a_1 \longleftarrow \mathrm{d}^{(k)^T}\mathrm{x}_A^{(k)}$; $a_2 \longleftarrow \mathrm{d}^{(k)^T}\mathrm{d}_A^{(k)}$; $a_3 \longleftarrow \mathrm{d}^{(k)^T}\mathrm{x}_B^{(k)}$; $a_4 \longleftarrow \mathrm{d}^{(k)^T}\mathrm{d}_B^{(k)}$;
    $a \longleftarrow a_2a_3 - a_1a_4$; $b \longleftarrow a_2\eta - a_4\gamma$; $c \longleftarrow a_1\eta - a_3\gamma$;
    se $b < 0$ **então**
      $\alpha_k \longleftarrow \dfrac{-b + \sqrt{b^2 - 4ac}}{2a}$;
    senão
      $\alpha_k \longleftarrow \dfrac{-2c}{b + \sqrt{b^2 - 4ac}}$;
    fim
    $\mathrm{x}^{(k+1)} \longleftarrow \mathrm{x}^{(k)} + \alpha_k\mathrm{d}^{(k)}$;
    $\mathrm{x}_A^{(k+1)} \longleftarrow \mathrm{x}_A^{(k)} + \alpha_k\mathrm{d}_A^{(k)}$; $\mathrm{x}_B^{(k+1)} \longleftarrow \mathrm{x}_B^{(k)} + \alpha_k\mathrm{d}_B^{(k)}$;
    $\gamma \longleftarrow \gamma + 2a_1\alpha_k + a_2\alpha_k^2$; $\eta \longleftarrow \eta + 2a_3\alpha_k + a_4\alpha_k^2$;
    $q_{k+1} \longleftarrow \gamma/\eta$;
    $\mathrm{r}^{(k+1)} \longleftarrow \mathrm{x}_A^{(k+1)} - q_{k+1}\mathrm{x}_B^{(k+1)}$;
    $k \longleftarrow k + 1$;
  fim
  $\lambda_j \longleftarrow q_k$; $\mathrm{v}_j \longleftarrow \mathrm{x}^{(k)}/\sqrt{\eta}$;
  $\mathrm{V} \longleftarrow [\mathrm{V}\ \ \mathrm{v}_j]$;
fim

3.4.2 Prolongação DPLS

A matriz de prolongação, P, será construída por linhas. Para cada $i \in \mathcal{F}$, vamos obter um vetor w_i contendo os pesos relacionados aos vizinhos do nó i na malha grossa que contribuirão para a prolongação desse nó.

Escolhemos um parâmetro d_p como comprimento máximo de caminho ao longo de conexões fortes (em geral, utiliza-se $d_p = 1$ ou $d_p = 2$). Para cada nó $i \in \mathcal{F}$, obtemos o conjunto $\mathcal{J}_i$ contendo os nós da malha grossa que estão conectados fortemente ao nó i por um caminho de distância menor ou igual a d_p, ou seja,

$$\mathcal{J}_i = \{j \in \mathcal{C} \mid \text{existe um caminho de } i \text{ para } j \text{ de distância menor ou igual a } d_p\}.$$

Os nós neste conjunto serão os candidatos a contribuírem para a prolongação do nó i. De $\mathcal{J}_i$ vamos extrair gradativamente um subconjunto $\mathcal{C}_i \subseteq \mathcal{J}_i$, com os nós que, de fato, contribuirão para a prolongação do nó i.

Queremos que a prolongação seja a mais precisa possível para os vetores algebricamente suaves. Para tanto, utilizamos as informações do espaço de teste construído anteriormente e, para cada $i \in \mathcal{F}$, requeremos que o vetor linha $\overline{\mathrm{v}}_i$, da matriz V, seja uma combinação linear dos vetores $\overline{\mathrm{v}}_j$ para $j \in \mathcal{C}_i$, isto é,

$$\overline{\mathrm{v}}_i = \sum_{j \in \mathcal{C}_i} \beta_j \overline{\mathrm{v}}_j.$$

Seja n_i a quantidade de elementos em $\mathcal{C}_i$. Denotando por $\mathrm{V}_{c,i}$ a matriz de ordem $n_{tv} \times n_i$ cujas colunas são os vetores $\overline{\mathrm{v}}_j$, $j \in \mathcal{C}_i$, e por $\mathrm{w}_i = [\beta_1 \cdots \beta_{n_i}]^T$ o vetor com os pesos β_j, temos que

$$\mathrm{V}_{c,i}\mathrm{w}_i = \overline{\mathrm{v}}_i.$$

Resolvendo esse sistema linear, obtemos o vetor w_i com os pesos da prolongação do nó i.

Em geral, o sistema acima é inconsistente e o melhor que podemos obter é uma solução de quadrados mínimos. Dessa forma, os pesos da prolongação são calculados de modo a minimizar a norma do resíduo, isto é,

$$\operatorname*{Min}_{\beta_j} \left\| \overline{\mathrm{v}}_i - \sum_{j \in \mathcal{C}_i} \beta_j \overline{\mathrm{v}}_j \right\|_2.$$

A ideia principal do método DPLS (do inglês *Dynamic Pattern Least Squares*), descrito por Magri *et al.* [33], é construir o subconjunto $\mathcal{C}_i$ de forma adaptativa, ao mesmo tempo em que aplicamos transformações de Householder (consulte [56], Seção 3.2) para triangularizar a matriz $\mathrm{V}_{c,i}$ e resolver o problema de quadrados mínimos.

Dado o conjunto $\mathcal{J}_i$, queremos encontrar os n_i nós que possuem mais importância na prolongação do nó i. Supondo que $n_i = 1$, o problema se resumiria em encontrar o índice $\bar{j} \in \mathcal{J}_i$ tal que $\overline{\mathrm{v}}_{\bar{j}}$ é o vetor mais próximo de $\overline{\mathrm{v}}_i$ ou, equivalentemente, tal que o ângulo entre os vetores $\overline{\mathrm{v}}_{\bar{j}}$ e $\overline{\mathrm{v}}_i$ seja mínimo. Sendo θ esse ângulo, temos que

$$\cos^2(\theta) = \frac{(\overline{\mathrm{v}}_i^T \overline{\mathrm{v}}_{\bar{j}})^2}{(\overline{\mathrm{v}}_i^T \overline{\mathrm{v}}_i)(\overline{\mathrm{v}}_{\bar{j}}^T \overline{\mathrm{v}}_{\bar{j}})} = SoC[i, \bar{j}].$$

O ângulo θ será mínimo quando $\cos(\theta)$ for máximo. Assim, escolhemos o nó $\bar{j} \in \mathcal{J}_i$ que possui a maior força de conexão com o nó i, acrescentamos $\bar{j}$ em $\mathcal{C}_i$ e $\overline{\mathrm{v}}_{\bar{j}}$ como a última coluna de $\mathrm{V}_{c,i}$.

Antes de escolher o próximo nó em $\mathcal{C}_i$, já construímos a primeira transformação de Householder, Q, de modo a zerar as componentes $2, 3, ..., n_{tv}$ do vetor $\overline{\mathrm{v}}_{\bar{j}}$. Aplicamos essa transformação nos vetores $\overline{\mathrm{v}}_i$ e $\overline{\mathrm{v}}_j$ com $j \in \mathcal{J}_i \setminus \mathcal{C}_i$. Após atualizar esses vetores, recalculamos as forças de conexões e, novamente, escolhemos o nó com a maior delas para entrar em $\mathcal{C}_i$.

Esse processo se repete até que todos os nós em $\mathcal{J}_i$ sejam colocados em $\mathcal{C}_i$ ou até que um critério de parada seja satisfeito, por exemplo

$$\left\| \overline{\mathrm{v}}_i - \sum_{j \in \mathcal{C}_i} \beta_j \overline{\mathrm{v}}_j \right\|_2 \leq \varepsilon_p \left\| \overline{\mathrm{v}}_i \right\|_2 .$$

Conforme Magri *et al.* [33], na prática podemos verificar se $\|\mathrm{z}\|_2 \leq \varepsilon_p \|\overline{\mathrm{v}}_i\|_2$, em que z é o vetor $\overline{\mathrm{v}}_i$ atualizado após aplicarmos a transformação de Householder em cada iteração. Segundo Franceschini *et al.* [20], a escolha da tolerância ε_p é difícil e dependente do problema que está sendo resolvido, podendo gerar matrizes de prolongação mal-condicionadas e prejudicar a eficiência do *multigrid* algébrico. Os autores sugerem um novo critério de parada, baseado no número de condição da matriz $\mathrm{V}_{c,i}$, que apresentamos a seguir.

Aplicamos as transformações de Householder com o objetivo de triangularizar a matriz $\mathrm{V}_{c,i}$, obtendo uma matriz U triangular superior. Em cada iteração, construímos uma nova coluna de U e verificamos o critério de parada

$$cond(U) > \kappa,$$

em que $cond(U)$ representa o número de condição da matriz U e κ é uma tolerância predefinida. Além disso, sugere-se a utilização de um número máximo de iterações, it_p, que controle a quantidade de nós que contribuirão para a prolongação de cada nó $i \in \mathcal{F}$.

No final do processo, obtemos o vetor w_i com os pesos da prolongação para o nó i resolvendo um sistema triangular superior,

$$U\mathrm{w}_i = \mathrm{z}.$$

Com esses pesos e os índices dos nós que foram colocados em $\mathcal{C}_i$, podemos construir a linha i da matriz P. O Algoritmo 6 descreve o método DPLS.

Na implementação, consideramos $d_p = 1$ e obtivemos o conjunto $\mathcal{J}_i$ como sendo o conjunto de índices dos nós da malha grossa que possuem conexão forte com o nó i. As reflexões de Householder foram obtidas utilizando a fatoração QR do Matlab. Nos resultados computacionais, apresentamos uma análise dos valores κ e it_p, bem como outros parâmetros do método *multigrid* algébrico adaptativo, quando aplicado na resolução dos sistemas lineares provenientes da otimização topológica estrutural.

Algoritmo 6 Prolongação DPLS

para $i \in \mathcal{F}$ **faça**
  $k \longleftarrow 0$; $\mathcal{C}_i \longleftarrow \emptyset$; $\mathrm{z} \longleftarrow \mathrm{v}_i$; $\overline{\mathrm{v}}_i \longleftarrow \mathrm{v}_i$; $U \longleftarrow [\,]$;
  obtenha o conjunto $\mathcal{J}_i$;
  enquanto $cond(U) \leq \kappa$ e $k < it_p$ **faça**
    $k \longleftarrow k + 1$;
    selecione $\bar{j} \in \mathcal{J}_i \setminus \mathcal{C}_i$ que possui maior força de conexão com i;
    $\mathcal{C}_i \longleftarrow \mathcal{C}_i \cup \{\bar{j}\}$;
    obtenha a reflexão Q de HH que zera as últimas $n_{tv} - k$ entradas de $\overline{\mathrm{v}}_{\bar{j}}$;
    adicione $Q\overline{\mathrm{v}}_{\bar{j}}$ como a última coluna de U;
    $\mathrm{z} \longleftarrow Q\mathrm{z}$;
    para $j \in \mathcal{J}_i \setminus \mathcal{C}_i$ **faça**
      $\overline{\mathrm{v}}_j \longleftarrow Q\overline{\mathrm{v}}_j$;
    fim
    atualize as forças de conexões;
  fim
  calcule w_i resolvendo o sistema triangular $U\mathrm{w}_i = \mathrm{z}$;
fim

Capítulo 4

Modelos de Ordem Reduzida

Na resolução do problema de otimização topológica, precisamos resolver, a cada iteração, um sistema linear na forma $\mathrm{Ku} = \mathrm{f}$, o que é a etapa mais cara do processo, pois a dimensão n da matriz de rigidez global, K, pode ser muito grande, principalmente no caso tridimensional. Uma estratégia que tem sido explorada para diminuir esse custo é o uso de Modelos de Ordem Reduzida, ou ROMs (do inglês *Reduced Order Models*), que encontram soluções aproximadas dos sistemas lineares com um custo computacional baixo (veja [13, 21, 57] para mais detalhes).

Uma classe particular das técnicas de redução de ordem é baseada em projeção, também conhecida como abordagem de base reduzida, cujo objetivo é reduzir a quantidade de variáveis do modelo por meio da projeção em um subespaço gerado por uma determinada base. Apresentamos a seguir os conceitos básicos dessa estratégia.

4.1 Sistema reduzido

No processo de otimização topológica, resolvemos uma sequência de sistemas lineares na forma $\mathrm{Ku} = \mathrm{f}$, gerando uma sequência de soluções $\mathrm{u}_1, \mathrm{u}_2, \mathrm{u}_3, ...$ Se o problema for factível, essa sequência de soluções converge para uma solução ótima, u^*. No início, as soluções obtidas variam bastante, de modo que pode ser satisfatório utilizar apenas aproximações para as soluções. Já no final do processo, as soluções não mudam muito, isto é, a solução $\mathrm{u_k}$ fica próxima da solução anterior, $\mathrm{u_{k-1}}$, para um índice de iteração k suficientemente grande. Então, na iteração k, a ideia é buscar uma aproximação para a solução $\mathrm{u_k}$ que esteja no subespaço $\mathcal{V}$ gerado por m (com $m << n$) soluções anteriores. Tipicamente, o subespaço $\mathcal{V}$ é definido por uma base, denominada *base reduzida*. Denotaremos por $\Psi : n \times m$ a matriz cujas colunas são os vetores dessa base.

Tomando a aproximação $\mathrm{u_k} \approx \Psi\alpha$, de modo que $\mathrm{u_k}$ seja uma combinação linear dos vetores da base de $\mathcal{V}$, na qual α contém os coeficientes dessa combinação, e projetando o sistema $\mathrm{Ku} = \mathrm{f}$ no subespaço gerado por Ψ, obtemos

$$(\Psi^T \mathrm{K}\Psi)\alpha = \Psi^T \mathrm{f} \Leftrightarrow \hat{\mathrm{K}}\alpha = \hat{\mathrm{f}}, \tag{4.1}$$

denominado *sistema reduzido*. A matriz $\hat{\mathrm{K}} = \Psi^T \mathrm{K}\Psi$ possui dimensão $m \times m$ e o vetor do lado direito, $\hat{\mathrm{f}} = \Psi^T \mathrm{f}$, tem dimensão $m \times 1$. Como $m << n$, a resolução do sistema reduzido tem um custo muito menor se comparada à do sistema original.

Assim, resolvemos o sistema reduzido para obter o vetor α e encontramos uma aproximação para a solução do sistema original, $\mathrm{u_k} \approx \widetilde{\mathrm{u}}_\mathrm{k} = \Psi\alpha$. A precisão dessa aproximação pode ser medida por meio da norma relativa do resíduo:

$$r_{\mathrm{ROM}} = \frac{\|\mathrm{f} - \mathrm{K}\widetilde{\mathrm{u}}_\mathrm{k}\|}{\|\mathrm{f}\|}.$$

Se este valor for pequeno o suficiente, a aproximação $\widetilde{\mathrm{u}}_\mathrm{k}$ é boa, uma vez que o resíduo da solução exata é igual a zero. Consideramos então a condição

$$r_{\mathrm{ROM}} < \varepsilon_{\mathrm{ROM}},$$

em que $\varepsilon_{\mathrm{ROM}} > 0$ é uma tolerância que indica a precisão exigida para a aproximação. Gogu [21] utiliza, em seus experimentos, valores para a tolerância da ordem de 10^{-1} ou 10^{-2}. Choi *et al.* [13] propõem um critério mais elaborado, baseado nas condições KKT do problema de otimização topológica, para verificar se a solução do sistema está próxima da ótima e não aceitar soluções que sejam imprecisas.

Se a condição acima for satisfeita, consideramos a aproximação obtida pelo modelo reduzido boa o suficiente e a utilizamos no restante do processo de otimização topológica, em lugar da solução exata. Caso contrário, seguimos outra estratégia para obter uma solução mais precisa, como resolver o sistema linear original normalmente, utilizando a fatoração de Cholesky ou o método dos gradientes conjugados. Nos casos em que não aceitamos a solução aproximada, Choi *et al.* [13] propõem uma estratégia denominada *ROM-Recycling*, que reutiliza a base Ψ no algoritmo de gradientes conjugados e conseguem obter uma solução mais precisa com um custo menor do que resolver o sistema de forma exata.

4.2 Construção da base reduzida

Existem diferentes estratégias para construir e atualizar a matriz Ψ. Estudamos algumas técnicas nas quais se atualiza a base reduzida durante o processo de otimização topológica, utilizando valores anteriores dos deslocamentos nodais, ou seja, soluções do sistema linear de iterações anteriores.

Na m-ésima iteração, já teremos calculado as soluções $\mathrm{u}_1, \mathrm{u}_2, ..., \mathrm{u}_\mathrm{m}$, que serão utilizadas para a criação da base reduzida. Uma opção é considerar $\Psi = [\mathrm{u}_1\ \mathrm{u}_2\ \cdots\ \mathrm{u}_\mathrm{m}]$. Entretanto, conforme nos aproximamos da solução ótima, o conjunto das m soluções anteriores fica quase linearmente dependente, de modo que não podemos utilizar os próprios vetores solução na base. Neste caso, utilizamos um processo de ortogonalização da base. Para tanto, são propostas duas abordagens:

- base ortonormal via Gram-Schmidt (Gogu [21]);
- base ortonormal via decomposição SVD, conhecida como Decomposição Ortogonal Apropriada (POD, do inglês *Proper Orthogonal Decomposition*) (Xiao *et al.* [57]).

Uma vez construída a matriz Ψ, nas iterações seguintes obtemos aproximações para as soluções do sistema linear resolvendo o modelo de ordem reduzida (4.1). Caso a condição $r_{\mathrm{ROM}} < \varepsilon_{\mathrm{ROM}}$ não seja satisfeita, isto é, a aproximação não seja boa o suficiente, precisamos

encontrar uma solução mais precisa do sistema e, então, usar essa solução para atualizar a base.

Em geral, consideramos um valor fixo N_b como o tamanho máximo da base. Depois que a matriz Ψ possui N_b colunas, toda vez que um novo vetor entra para atualizar a base, retiramos o vetor mais antigo da base, ou seja, retiramos a primeira coluna de Ψ.

4.2.1 Atualização via Gram-Schmidt

A ideia é construir uma base ortonormal aplicando o processo de ortonormalização de Gram-Schmidt (consulte [56], Seção 3.4) aos vetores solução, conforme eles forem sendo calculados. Iniciamos com a matriz Ψ vazia. Obtida a solução do sistema na primeira iteração, u_1, a primeira coluna de Ψ será o vetor u_1 normalizado, ou seja,

$$\Psi \longleftarrow \left[\begin{array}{c} \dfrac{\mathrm{u}_1}{\|\mathrm{u}_1\|} \end{array} \right].$$

Os próximos vetores da base são calculados pelo processo de Gram-Schmidt. Obtida uma solução $\mathrm{u_i}$, calculamos

$$\mathrm{v} = \mathrm{u_i} - \sum_{j=1}^{i} (\phi_j^T \mathrm{u_i}) \phi_j = \mathrm{u_i} - \Psi(\Psi^T \mathrm{u_i}),$$

em que ϕ_j denota a coluna j da matriz Ψ. Em seguida, acrescentamos uma coluna em Ψ com o vetor v normalizado, isto é,

$$\Psi \longleftarrow \left[\begin{array}{cc} \Psi & \dfrac{\mathrm{v}}{\|\mathrm{v}\|} \end{array} \right].$$

Antes disso, contudo, podemos verificar se $\|\mathrm{v}\| > \varepsilon_{\mathrm{QR}}$, em que $\varepsilon_{\mathrm{QR}} > 0$ é alguma tolerância que indica se o vetor $\mathrm{u_i}$ é linearmente independente dos demais vetores da base (colunas de Ψ). Se a condição não for satisfeita, não incluímos o vetor na base.

Realizamos esse procedimento até que Ψ possua uma quantidade máxima de colunas, N_b. A partir daí, utilizamos essa matriz para obter as soluções aproximadas dos sistemas lineares, por meio do modelo de ordem reduzida (4.1). A qualidade da solução aproximada precisa ser testada em cada iteração e, se a condição $r_{\mathrm{ROM}} < \varepsilon_{\mathrm{ROM}}$ não for satisfeita, precisamos obter uma solução mais precisa de outra maneira e atualizar a base reduzida. Para atualizar a base, descartamos o vetor mais antigo, isto é, a primeira coluna de Ψ, e calculamos a nova coluna como antes, aplicando Gram-Schmidt.

Dados uma solução u, candidata a entrar na base, o número da iteração atual, k, a matriz Ψ atual e a quantidade de colunas da matriz, m_k, atualizamos a base reduzida pelo processo de Gram-Schmidt por meio do Algoritmo 7. A versão do algoritmo que utilizamos em nossa implementação corresponde ao processo de Gram-Schmidt modificado, que é mais estável numericamente e, portanto, evita que a ortogonalidade entre as colunas da base seja perdida no decorrer das iterações.

Algoritmo 7 Atualização da base reduzida via Gram-Schmidt modificado

se k = 1 **então**
    $\Psi \longleftarrow \frac{\mathrm{u}}{\|\mathrm{u}\|}$;
senão
    se $m_k = N_b$ **então**
        retirar a primeira coluna de Ψ;
    fim
    $\mathrm{v} \longleftarrow \mathrm{u} - \Psi(\Psi^T \mathrm{u})$;
    se $\|\mathrm{v}\| > \varepsilon_{\mathrm{QR}}$ **então**
        $\Psi \longleftarrow \left[\begin{array}{cc} \Psi & \frac{\mathrm{v}}{\|\mathrm{v}\|} \end{array} \right]$;
    fim
fim

4.2.2 Atualização via decomposição SVD

Consideremos agora um conjunto de soluções $\mathrm{u}_1, \mathrm{u}_2, ..., \mathrm{u}_\mathrm{m}$ e a matriz cujas colunas são esses vetores, $A = \left[\begin{array}{ccc} \mathrm{u}_1 & \mathrm{u}_2 & \cdots \ \mathrm{u}_\mathrm{m} \end{array} \right]$. Queremos obter uma base para o subespaço $\mathcal{V} = \mathrm{span}\{\mathrm{u}_1, \mathrm{u}_2, ..., \mathrm{u}_\mathrm{m}\} = \mathrm{Im}(A)$. Uma base ortonormal para o subespaço imagem aparece na decomposição em valores singulares (SVD) da matriz:

$$A = UDV^T,$$

em que $U : n \times n$ e $V : m \times m$ são matrizes ortonormais e $D : n \times m$ é tal que $d_{ij} = 0$ para $i \neq j$ e $d_{ii} = \sigma_i$ são os valores singulares de A, com $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r > 0$, sendo $r \leq \min\{n, m\}$ o posto de A.

Sabemos que as primeiras r colunas da matriz U formam uma base ortonormal para o subespaço $\mathrm{Im}(A)$ (veja, por exemplo, Watkins [56], Seção 4.1). Assim, podemos considerar $\Psi = U(:, 1 : r)$, a matriz formada pelas primeiras r colunas de U. Além disso, essa base é tal que Ψ minimiza a distância, na norma de Frobenius, entre a matriz A e a matriz projetada no subespaço $\mathcal{V}$, isto é,

$$\Psi = \arg \min_{B:n\times r,\ B^TB=I} \|A - BB^TA\|_F^2.$$

Entretanto, o cálculo da decomposição SVD é muito caro, mesmo para uma matriz A que possui poucas colunas. Vejamos como obter a base reduzida de forma mais eficiente, fazendo atualizações da SVD a cada inclusão de uma nova coluna na matriz A.

Iniciamos com a matriz Ψ vazia. Obtida a primeira solução do sistema linear, u_1, consideramos a matriz $A_1 = [\mathrm{u}_1]$ cuja decomposição SVD reduzida é dada por

$$A_1 = \left[\begin{array}{c} \frac{\mathrm{u}_1}{\|\mathrm{u}_1\|} \end{array} \right] \left[\begin{array}{c} \|\mathrm{u}_1\| \end{array} \right] \left[\begin{array}{c} 1 \end{array} \right] = U_1 D_1 {V_1}^T.$$

Assim, tomamos $\Psi = U_1$ como a matriz da base reduzida na primeira iteração.

Suponha que, na iteração k, tenhamos a matriz A_k, cujas colunas são vetores soluções obtidos anteriormente, e que conheçamos a decomposição SVD reduzida dessa matriz, $A_k = U_k D_k {V_k}^T$, de modo que $\Psi = U_k$. Obtida uma nova solução u, candidata a entrar para a base, consideramos a nova matriz $A_{k+1} = \left[\begin{array}{cc} A_k & \mathrm{u} \end{array} \right]$ e precisamos obter a

sua decomposição SVD reduzida. Felizmente, como A_{k+1} é a matriz A_k com apenas uma nova coluna, é possível atualizar a decomposição SVD já conhecida, da seguinte forma:

$$\begin{aligned}
A_{k+1} &= \left[\begin{array}{cc} A_k & \mathrm{u} \end{array}\right] = \left[\begin{array}{cc} U_k D_k {V_k}^T & \mathrm{u} \end{array}\right] = \left[\begin{array}{cc} U_k D_k & \mathrm{u} \end{array}\right] \left[\begin{array}{cc} V_k & 0 \\ 0 & 1 \end{array}\right]^T \\
&= \left[\begin{array}{cc} U_k D_k & U_k {U_k}^T \mathrm{u} + \dfrac{\mathrm{u} - U_k {U_k}^T \mathrm{u}}{\|\mathrm{u} - U_k {U_k}^T \mathrm{u}\|} \|\mathrm{u} - U_k {U_k}^T \mathrm{u}\| \end{array}\right] \left[\begin{array}{cc} V_k & 0 \\ 0 & 1 \end{array}\right]^T \\
&= \left[\begin{array}{cc} U_k D_k & U_k \mathrm{z} + \dfrac{\mathrm{u} - U_k \mathrm{z}}{\delta} \delta \end{array}\right] \left[\begin{array}{cc} V_k & 0 \\ 0 & 1 \end{array}\right]^T \\
&= \left[\begin{array}{cc} U_k & \dfrac{\mathrm{u} - U_k \mathrm{z}}{\delta} \end{array}\right] \left[\begin{array}{cc} D_k & \mathrm{z} \\ 0 & \delta \end{array}\right] \left[\begin{array}{cc} V_k & 0 \\ 0 & 1 \end{array}\right]^T \\
&= \left[\begin{array}{cc} U_k & \mathrm{w} \end{array}\right] \left[\begin{array}{cc} D_k & \mathrm{z} \\ 0 & \delta \end{array}\right] \left[\begin{array}{cc} V_k & 0 \\ 0 & 1 \end{array}\right]^T,
\end{aligned}$$

em que $\mathrm{z} = {U_k}^T \mathrm{u}$ são as coordenadas reduzidas do vetor u na base dada por $\Psi = U_k$, $\delta = \|\mathrm{u} - U_k \mathrm{z}\|$ é a norma da diferença entre u e a sua projeção ortogonal no subespaço gerado pela base Ψ, e $\mathrm{w} = (\mathrm{u} - U_k \mathrm{z})/\delta$ é essa diferença normalizada.

A expressão obtida acima é quase a decomposição SVD reduzida da matriz A_{k+1}, a não ser pela matriz central,

$$B_k = \left[\begin{array}{cc} D_k & \mathrm{z} \\ 0 & \delta \end{array}\right],$$

que é quase diagonal, exceto pelo vetor z na última coluna. Além disso, se r_k é o posto de A_k, então a matriz B_k é quadrada de ordem $r_k + 1$, que é um valor relativamente pequeno. Logo, o cálculo da decomposição SVD de B_k é computacionalmente mais barato. Seja

$$B_k = \overline{U}_k \overline{D}_k {\overline{V}_k}^T$$

essa decomposição. Assim, obtemos

$$\begin{aligned}
A_{k+1} &= \left[\begin{array}{cc} U_k & \mathrm{w} \end{array}\right] B_k \left[\begin{array}{cc} V_k & 0 \\ 0 & 1 \end{array}\right]^T \\
&= \left[\begin{array}{cc} U_k & \mathrm{w} \end{array}\right] \overline{U}_k \overline{D}_k {\overline{V}_k}^T \left[\begin{array}{cc} V_k & 0 \\ 0 & 1 \end{array}\right]^T \\
&= U_{k+1} D_{k+1} {V_{k+1}}^T,
\end{aligned}$$

que é a decomposição SVD reduzida de A_{k+1}, em que $U_{k+1} = \left[\begin{array}{cc} U_k & \mathrm{w} \end{array}\right] \overline{U}_k$, $D_{k+1} = \overline{D}_k$ e $V_{k+1} = \left[\begin{array}{cc} V_k & 0 \\ 0 & 1 \end{array}\right] \overline{V}_k$. Com isso, podemos atualizar a matriz da base reduzida, $\Psi = U_{k+1}$.

Assim como na atualização via Gram-Schmidt, é necessário verificar a dependência linear dos vetores da base. Para isso, definimos uma tolerância $\varepsilon_{\mathrm{SVD}} > 0$ e verificamos se $\delta > \varepsilon_{\mathrm{SVD}}$. Se a condição não for satisfeita, consideramos $\delta = 0$ na matriz B_k. Neste caso, a decomposição $B_k = \overline{U}_k \overline{D}_k {\overline{V}_k}^T$ é tal que a matriz $\overline{D}_k$ terá um zero na última componente da diagonal. Assim, só atualizamos as primeiras r_k linhas e colunas

das matrizes, da seguinte forma:

$$U_{k+1} = U_k \overline{U}_k(1 : r_k, 1 : r_k);$$
$$D_{k+1} = \overline{D}_k(1 : r_k, 1 : r_k);$$
$$V_{k+1} = V_k \overline{V}_k(1 : r_k, 1 : r_k).$$

Embora a ortogonalidade da matriz U_k esteja teoricamente garantida por ser o produto de matrizes ortogonais, numericamente essa ortogonalidade pode ser perdida. Uma possibilidade heurística de verificação consiste em conferir se a primeira e a última coluna de U_k são ortogonais. Note que não há garantia de que essa heurística funcione sempre, pois pode ocorrer que duas colunas intermediárias não sejam ortogonais. Entretanto, essa estratégia de verificação da ortogonalidade geralmente apresenta bons resultados e é uma alternativa barata. Se o teste de ortogonalidade falhar, podemos obter a fatoração QR da matriz U_k e tomar $U_k = Q$.

Neste trabalho, tentamos implementar a estratégia dos modelos de ordem reduzida, utilizando tanto Gram-Schmidt quanto a decomposição SVD na atualização da base, para resolver os problemas de otimização topológica estrutural, mas infelizmente não obtivemos bons resultados iniciais ao combiná-la com a programação linear sequencial. O uso das soluções aproximadas do sistema linear, obtidas pelo modelo reduzido, no meio do processo de otimização, pode tornar incompatíveis as reduções real e prevista da função de mérito, de modo que o passo é recusado inúmeras vezes, na primeira tentativa em que tentamos utilizar a base reduzida, prejudicando a convergência do método. Nesse caso, é necessário fazer adaptações complicadas no algoritmo da PLS.

No próximo capítulo, apresentamos outra estratégia, denominada multirresolução, que também tem sido utilizada para reduzir o tempo de resolução dos problemas de otimização topológica e que conseguimos combinar, de forma mais eficiente, com o método *multigrid* e com a PLS.

Capítulo 5

Multirresolução

Na otimização topológica estrutural, encontramos o formato de uma estrutura contida em um determinado domínio dividindo-o em uma malha de elementos finitos e definindo a topologia da estrutura por meio da densidade de material em cada elemento. A *resolução* da estrutura indica o nível de detalhes que ela possui, o que depende da quantidade de elementos na malha, de maneira parecida com a resolução de uma imagem/fotografia, que expressa a qualidade da mesma por meio da quantidade de *pixels* que a compõem. A obtenção de estruturas mais detalhadas e com contornos mais suaves requer uma resolução alta, isto é, uma grande quantidade de elementos na malha.

Aumentar o número de elementos faz com que a quantidade de variáveis do problema de otimização e a dimensão dos sistemas lineares de equilíbrio cresçam demasiadamente, sobretudo no caso tridimensional, acarretando um consumo elevado de memória e tempo computacional para resolver os problemas. Diversos trabalhos propõem estratégias para reduzir esse consumo, geralmente focadas no sistema de equilíbrio, como técnicas de reanálise [43], redução de ordem [13, 21, 57] e o uso de métodos *multigrid* [1, 40], conforme estudamos nos capítulos anteriores.

Neste capítulo, apresentamos uma estratégia de *multirresolução*, proposta por Nguyen *et al.* [37, 38], cuja ideia é trabalhar com resoluções diferentes em cada etapa da otimização topológica: uma malha mais grossa (com menos elementos) para a resolução dos sistemas lineares, uma malha intermediária para a resolução do problema de otimização e uma malha mais fina (com mais elementos) na qual é definida a distribuição das densidades de material. Com isso, o tempo gasto nas etapas mais caras é reduzido, porque trabalhamos em malhas com menos elementos, enquanto a resolução final da estrutura é alta, pois a distribuição do material é definida em uma malha com mais elementos.

5.1 Definição das malhas

Vamos considerar três malhas distintas, isto é, com diferentes quantidades de elementos, para um mesmo domínio de um problema de otimização topológica:

- *Malha de deslocamentos*: utilizada para obter uma aproximação dos deslocamentos da estrutura, resolvendo-se o sistema linear de equilíbrio, $\mathrm{Ku} = \mathrm{f}$;

- *Malha de variáveis de projeto*: utilizada para obter a solução do problema de otimização, aplicando-se a PLS, por exemplo;
- *Malha de densidades*: utilizada para representar a distribuição de material.

Chamaremos de *elemento de deslocamento* um elemento pertencente à malha de deslocamentos, de *elemento de densidade* um elemento pertencente à malha de densidades e de *variável de projeto* (um valor numérico) um elemento da malha de variáveis de projeto. Observamos que esta última malha não possui significado físico.

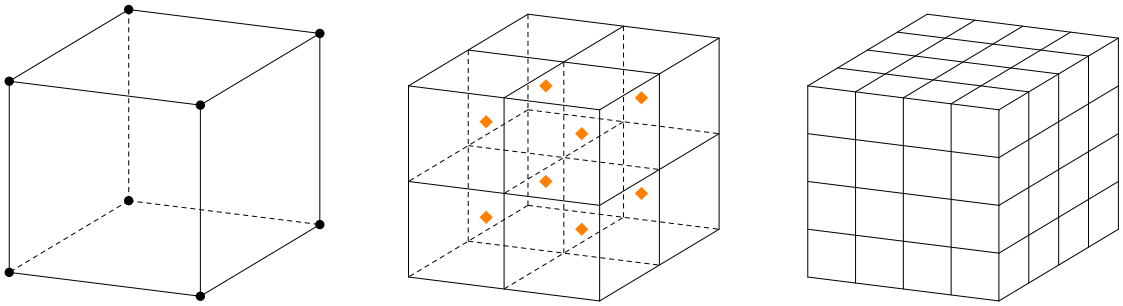
No método tradicional, as três malhas são iguais, de modo que cada elemento de deslocamento possui uma densidade a ele associada, que também é uma variável de projeto. A proposta da multirresolução é trabalharmos com malhas diferentes. Em geral, a malha de deslocamentos é escolhida como a de resolução mais baixa, ou seja, a que possui menos elementos, de modo que a dimensão da matriz de rigidez global, K, seja pequena. Para obtermos uma estrutura com resolução alta, a malha de densidades é escolhida como a mais fina, contendo mais elementos.

No trabalho de Nguyen *et al.* [37], as variáveis de projeto são definidas como os valores das densidades de material nos centros dos elementos de densidade. Entretanto, as malhas de variáveis de projeto e de densidades não precisam necessariamente coincidir. Em [38], os mesmos autores propõem uma melhoria da multirresolução na qual utilizam três malhas distintas entre si, de modo que a economia no tempo gasto para resolver os problemas é ainda maior. No nosso trabalho, testamos o caso em que as variáveis de projeto equivalem às densidades e também a utilização da malha de variáveis de projeto como uma malha intermediária entre a de deslocamentos e a de densidades.

Consideramos como base a malha de deslocamentos, que será a mais grossa. Cada elemento de deslocamento é um elemento prismático retangular. Para definir a malha de densidades, "subdividimos" cada elemento de deslocamento em $(n_{mr})^3$ elementos menores, que serão os elementos de densidade. Cada aresta de um elemento de densidade corresponderá a $1/n_{mr}$ da aresta do elemento de deslocamento. De maneira análoga, "subdividimos" cada elemento de deslocamento em $(d_{mr})^3$ elementos iguais, que formarão a malha de variáveis de projeto. Observamos que $1 < d_{mr} \leq n_{mr}$ e ambos os parâmetros, d_{mr} e n_{mr}, devem ser números inteiros.

Por exemplo, se $d_{mr} = 2$ e $n_{mr} = 4$, cada elemento de deslocamento contém 8 variáveis de projeto e 64 elementos de densidade, conforme ilustra a Figura 5.1. Em 5.1(a) os pontos pretos representam os nós do elemento de deslocamento, em 5.1(b) os pontos laranja são os centros dos elementos na malha de variáveis de projeto e 5.1(c) mostra os elementos de densidade. Neste caso, uma malha de deslocamentos com $30 \times 10 \times 10 = 3000$ elementos, por exemplo, possui $31 \times 11 \times 11 = 3751$ nós, de modo que, desconsiderando-se as condições de contorno, o problema terá 11253 graus de liberdade (3 deslocamentos para cada nó). Com $d_{mr} = 2$, o problema terá $8 \times 3000 = 24000$ variáveis de projeto e, com $n_{mr} = 4$, a resolução final da estrutura terá $120 \times 40 \times 40 = 192000$ elementos de densidade. Para obter uma estrutura com essa mesma resolução na maneira tradicional, seria necessário trabalhar com 192000 variáveis de projeto e 610203 graus de liberdade.

A seguir, descrevemos outros aspectos importantes da multirresolução, explicando como as informações das diferentes malhas se conectam para a resolução do problema de otimização topológica estrutural.



(a) Elemento de deslocamento. (b) Variáveis de projeto. (c) Elementos de densidade.

Figura 5.1: Elementos nas diferentes malhas da multirresolução, com $n_{mr} = 4$ e $d_{mr} = 2$.

5.2 Projeção das variáveis de projeto em densidades

O problema de otimização será resolvido na malha de variáveis de projeto, o que significa que cada elemento j desta malha terá um valor x_j associado e o vetor que contém esses valores será atualizado a cada iteração do método de otimização empregado. Porém, também precisamos conhecer a densidade ρ_i de cada elemento i na malha de densidades.

Denotemos por x o vetor das variáveis de projeto e por ρ o vetor das densidades de material. No caso da multirresolução em que $d_{mr} = n_{mr}$, assim como na otimização topológica tradicional, podemos considerar que cada densidade equivale a uma variável de projeto, de modo que temos $\rho = \mathrm{x}$. Agora, quando $d_{mr} \neq n_{mr}$, o vetor de densidades ρ é obtido por meio de uma projeção do vetor das variáveis de projeto:

$$\rho = f_p(\mathrm{x}),$$

em que f_p denota a função de projeção.

Podemos utilizar uma projeção linear, que funciona de maneira parecida com a aplicação do filtro de densidades. Consideramos um raio $r_{min} > 0$ e, para cada elemento de densidade i, encontramos as variáveis de projeto x_j que estão na vizinhança $B(i, r_{min})$, isto é, encontramos os elementos da malha de variáveis de projeto cujos centros estão a uma distância menor ou igual a r_{min} do centro do elemento de densidade i. A Figura 5.2 ilustra a vizinhança de um elemento de densidade em um problema bidimensional.

A densidade ρ_i é então calculada como uma média ponderada dos valores das variáveis de projeto na vizinhança do elemento i:

$$\rho_i = \sum_{j \in B(i, r_{min})} \frac{\omega_{ij}(dist(i,j))}{\omega_i} x_j, \tag{5.1}$$

em que ω_{ij} são fatores de peso, $dist(i,j)$ denota a distância euclidiana entre os centros do elemento de densidade i e do elemento j na malha de variáveis de projeto, e ω_i é a soma dos fatores de peso ω_{ij} para todo $j \in B(i, r_{min})$, ou seja,

$$\omega_i = \sum_{j \in B(i, r_{min})} \omega_{ij}.$$

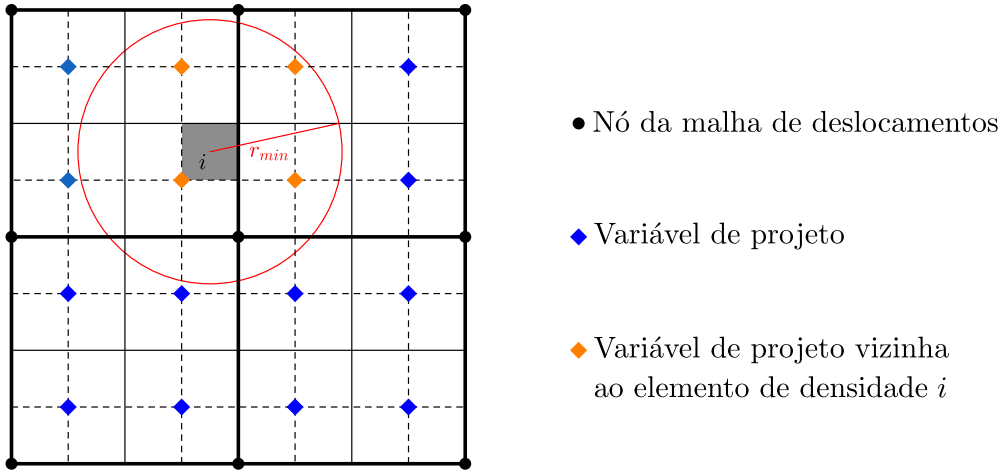


Figura 5.2: Vizinhança de um elemento de densidade i na multirresolução, com $n_{mr} = 4$ e $d_{mr} = 2$, para a projeção das variáveis de projeto em densidades, considerando-se um problema bidimensional.

Em [38] os autores propõem a utilização dos fatores de peso dados por

$$\omega_{ij} = \begin{cases} \dfrac{r_{min} - dist(i,j)}{r_{min}} & \text{se } \ dist(i,j) \leq r_{min}, \\ 0 & \text{se } \ dist(i,j) > r_{min}. \end{cases} \tag{5.2}$$

Também se pode utilizar outros fatores. Neste trabalho, além de (5.2), mantivemos os fatores utilizados na aplicação do filtro de densidades:

$$\omega_{ij} = \begin{cases} \dfrac{\exp\left(-\dfrac{dist(i,j)^2}{2\left(\dfrac{r_{min}}{3}\right)^2}\right)}{2\pi\left(\dfrac{r_{min}}{3}\right)} & \text{se } \ dist(i,j) \leq r_{min}, \\ 0 & \text{se } \ dist(i,j) > r_{min}. \end{cases}$$

Neste caso, observe que, quando as variáveis de projeto equivalem às densidades, ou seja, $\rho = \mathrm{x}$, a projeção para obter as densidades equivale à aplicação do filtro.

Considerando a distinção entre o vetor x das variáveis de projeto e o vetor ρ das densidades, o problema de otimização topológica estrutural pode ser reescrito da seguinte forma.

$$\begin{aligned} \underset{\mathrm{x},\,\mathrm{u}}{\text{Min}} \quad & \mathrm{f}^T\mathrm{u} \\ \text{s. a} \quad & \rho = f_p(\mathrm{x}), \\ & \mathrm{K}(\rho)\mathrm{u} = \mathrm{f}, \\ & \sum_{i=1}^{n_\rho} v_i\rho_i \leq V_{max}, \\ & 0 \leq x_j \leq 1, \quad j = 1, 2, ..., n_{\mathrm{x}}, \end{aligned} \tag{5.3}$$

em que u é o vetor global de deslocamentos nodais, f é o vetor global de cargas nodais equivalentes, $\mathrm{K} \equiv \mathrm{K}(\rho)$ é a matriz de rigidez global, V_{max} é o volume máximo que a

estrutura pode ocupar no domínio, n_{x} é a quantidade total de variáveis de projeto, n_ρ é a quantidade total de elementos de densidade, ρ_i é a densidade e v_i é o volume do elemento de densidade i. As dimensões dos vetores x, ρ e u são, respectivamente, n_{x}, n_ρ e n_{dofs}, em que n_{dofs} é o número total de graus de liberdade dos nós na malha de deslocamentos. Se n_{el} é o número total de elementos de deslocamento, então $n_{\mathrm{x}} = (d_{mr})^3 n_{el}$ e $n_\rho = (n_{mr})^3 n_{el}$.

5.3 Construção da matriz de rigidez

A resolução do sistema linear $\mathrm{K}\mathrm{u} = \mathrm{f}$ consome muito tempo. Por este motivo, ela é feita na malha de deslocamentos (ou malha de análise), que possui menos elementos em comparação com as outras malhas utilizadas na multirresolução. Dessa forma, o vetor u, solução do sistema, contém os deslocamentos dos nós da malha de deslocamentos.

A matriz de rigidez global, K, é formada por uma superposição das matrizes de rigidez $\mathrm{K}^{(e)}$, para cada elemento e na malha de deslocamentos. Tradicionalmente, essas matrizes dos elementos recebem a penalização das densidades de material, por meio da aplicação do modelo SIMP, e são dadas por:

$$\mathrm{K}^{(e)}(\rho_e) = (E_{min} + (\rho_e)^p (E_0 - E_{min})) \int_{\Omega_e} \mathrm{B}^T \mathrm{D}\mathrm{B}\, d\Omega_e, \tag{5.4}$$

em que ρ_e é a densidade do elemento e, p é o parâmetro de penalização do SIMP, E_0 é o módulo de Young do material sólido, E_{min} é um valor mínimo utilizado para evitar que a matriz seja singular, Ω_e é o domínio do elemento e, B é a matriz de deformação e D é uma matriz que contém os parâmetros de elasticidade do material.

O que muda na multirresolução é o cálculo das matrizes $\mathrm{K}^{(e)}$. Agora, não temos mais as densidades ρ_e, mas cada elemento de deslocamento é composto por $N_n = (n_{mr})^3$ elementos de densidade, conforme ilustra a Figura 5.3 para $n_{mr} = 4$.

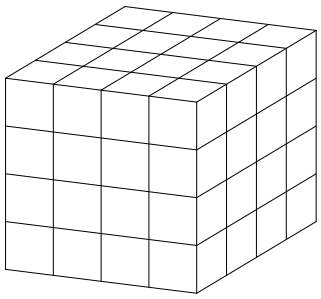


Figura 5.3: Elemento de deslocamento dividido em $N_n = 4^3 = 64$ elementos de densidade.

Consideremos um elemento (de deslocamento) prismático retangular com comprimento 2ℓ, altura $2h$ e largura $2w$. Além disso, adotemos o sistema de coordenadas locais (ξ, η, ζ), cuja origem está no baricentro do elemento, e consideremos que as coordenadas dos pontos no elemento assumem valores no intervalo $[-1, 1]$, de modo que $\xi = x/\ell$, $\eta = y/h$ e $\zeta = z/w$. Dessa forma, temos $\Omega_e = [-1,1] \times [-1,1] \times [-1,1]$ e $d\Omega_e = dx\, dy\, dz = \ell h w\, d\xi\, d\eta\, d\zeta$. Assim, a integral que aparece em (5.4) é dada por

$$\int_{\Omega_e} \mathrm{B}^T \mathrm{D}\mathrm{B}\, d\Omega_e = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} \ell h w\, \mathrm{B}^T \mathrm{D}\mathrm{B}\, d\xi\, d\eta\, d\zeta. \tag{5.5}$$

Ao dividirmos o elemento em $N_n = (n_{mr})^3$ elementos de densidade, também dividimos o intervalo $[-1, 1]$ em pequenos subintervalos de tamanho $1/n_{mr}$. Logo, a integral pode ser calculada como uma soma de integrais em domínios dados por intervalos menores:

$$\int_{\Omega_e} \mathrm{B}^T\mathrm{DB}\, d\Omega_e = \sum_{i=1}^{N_n} \left(\int_{\Omega_{e_i}} \mathrm{B}^T\mathrm{DB}\, d\Omega_{e_i} \right), \tag{5.6}$$

em que Ω_{e_i} é o domínio de um elemento de densidade e_i que está dentro do elemento de deslocamento e, para $i = 1, 2, ..., N_n$.

Sejam $[a_\xi, b_\xi]$, $[a_\eta, b_\eta]$ e $[a_\zeta, b_\zeta]$ os intervalos de integração do elemento de densidade e_i nas direções ξ, η e ζ, respectivamente, de modo que $\Omega_{e_i} = [a_\xi, b_\xi] \times [a_\eta, b_\eta] \times [a_\zeta, b_\zeta]$. Os intervalos de integração são diferentes para cada elemento de densidade e_i, mas aqui omitimos o índice e_i para simplificar a notação. Assim,

$$\mathrm{K}^{(e_i)} = \int_{\Omega_{e_i}} \mathrm{B}^T\mathrm{DB}\, d\Omega_{e_i} = \int_{a_\zeta}^{b_\zeta} \int_{a_\eta}^{b_\eta} \int_{a_\xi}^{b_\xi} \ell h w\, \mathrm{B}^T\mathrm{DB}\, d\xi\, d\eta\, d\zeta. \tag{5.7}$$

Na prática, essa integral é aproximada numericamente, por exemplo, por meio da quadratura Gaussiana.

Considere inicialmente uma função contínua, f, com apenas uma variável, ξ, no intervalo $[-1, 1]$ e suponha que queiramos calcular a integral definida de f nesse intervalo. Uma regra de quadratura Gaussiana fornece o valor aproximado da integral definida utilizando uma combinação linear de valores da função em certos pontos ξ_i, com $-1 \leq \xi_i \leq 1$, e pesos w_i. Ou seja, a integral é calculada somando-se os produtos do peso em cada ponto pelo valor da função no mesmo ponto:

$$\int_{-1}^{1} f(\xi)\, d\xi \approx w_1 f(\xi_1) + w_2 f(\xi_2) + \cdots + w_n f(\xi_n) = \sum_{i=1}^{n} w_i f(\xi_i). \tag{5.8}$$

Os pontos ξ_i e os pesos w_i são determinados de modo que a regra forneça a integral de forma exata para qualquer polinômio de grau $2n - 1$, sendo n a quantidade de pontos tomados no intervalo $[-1, 1]$.

As regras de quadratura são definidas para o domínio de integração $[-1, 1]$ e a Tabela 1.2 mostra as coordenadas $\xi_i \in [-1, 1]$ dos pontos de integração e os pesos w_i calculados para integrais de ordem $n = 1, 2, 3, 4$. Logo, para calcular a integral em outro domínio $[a_\xi, b_\xi]$, precisamos fazer uma mudança de intervalos, transformando a variável ξ em uma variável $\hat{\xi}$. Então, temos a seguinte transformação na integral:

$$\int_{a_\xi}^{b_\xi} f(\xi)\, d\xi = |J_\xi| \int_{-1}^{1} g(\hat{\xi})\, d\hat{\xi}, \tag{5.9}$$

em que o fator $|J_\xi|$ é o Jacobiano da transformação e g é a função f composta com a função que representa a transformação. A mudança de variável é definida pela equação

$$\begin{vmatrix} \xi & \hat{\xi} & 1 \\ a_\xi & -1 & 1 \\ b_\xi & 1 & 1 \end{vmatrix} = 0,$$

de onde obtemos

$$\xi = \left(\frac{b_\xi - a_\xi}{2} \right) \hat{\xi} + \left(\frac{b_\xi + a_\xi}{2} \right). \tag{5.10}$$

Ademais,

$$J_\xi = \frac{d\xi}{d\hat{\xi}} = \frac{b_\xi - a_\xi}{2}.$$

Logo, em (5.9) temos

$$\int_{a_\xi}^{b_\xi} f(\xi)\, d\xi = \frac{b_\xi - a_\xi}{2} \int_{-1}^{1} f\left(\left(\frac{b_\xi - a_\xi}{2}\right)\hat{\xi} + \left(\frac{b_\xi + a_\xi}{2}\right)\right)\, d\hat{\xi}. \tag{5.11}$$

A aproximação por integração numérica utilizando n pontos passa a ser

$$\int_{a_\xi}^{b_\xi} f(\xi)\, d\xi \approx \frac{b_\xi - a_\xi}{2} \sum_{i=1}^{n} w_i f\left(\left(\frac{b_\xi - a_\xi}{2}\right)\hat{\xi}_i + \left(\frac{b_\xi + a_\xi}{2}\right)\right). \tag{5.12}$$

A quadratura Gaussiana pode ser estendida para duas e três dimensões. Seja $f(\xi, \eta, \zeta)$ uma função que queiramos integrar no domínio $[-1, 1]^3$. Então,

$$\int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} f(\xi, \eta, \zeta)\, d\xi\, d\eta\, d\zeta \approx \sum_{k=1}^{n_z}\sum_{j=1}^{n_y}\sum_{i=1}^{n_x} w_k w_j w_i\, f(\xi_i, \eta_j, \zeta_k), \tag{5.13}$$

em que n_x, n_y e n_z são os números de pontos de integração nas direções ξ, η e ζ, respectivamente, e w_i, w_j e w_k são os pesos correspondentes.

Para integrar em outro domínio, $\Omega_{e_i} = [a_\xi, b_\xi] \times [a_\eta, b_\eta] \times [a_\zeta, b_\zeta]$, fazemos a mudança dos intervalos trocando as variáveis ξ, η e ζ por $\hat{\xi}$, $\hat{\eta}$ e $\hat{\zeta}$, de forma análoga àquela indicada em (5.10). Daí, temos

$$\int_{a_\zeta}^{b_\zeta}\int_{a_\eta}^{b_\eta}\int_{a_\xi}^{b_\xi} f(\xi, \eta, \zeta)\, d\xi\, d\eta\, d\zeta \approx |J_{\xi\eta\zeta}| \sum_{k=1}^{n_z}\sum_{j=1}^{n_y}\sum_{i=1}^{n_x} w_k w_j w_i\, g(\hat{\xi}_i, \hat{\eta}_j, \hat{\zeta}_k), \tag{5.14}$$

em que g é a função f composta com a transformação e

$$J_{\xi\eta\zeta} = \begin{vmatrix} \frac{d\xi}{d\hat{\xi}} & \frac{d\xi}{d\hat{\eta}} & \frac{d\xi}{d\hat{\zeta}} \\ \frac{d\eta}{d\hat{\xi}} & \frac{d\eta}{d\hat{\eta}} & \frac{d\eta}{d\hat{\zeta}} \\ \frac{d\zeta}{d\hat{\xi}} & \frac{d\zeta}{d\hat{\eta}} & \frac{d\zeta}{d\hat{\zeta}} \end{vmatrix} = \left(\frac{b_\zeta - a_\zeta}{2}\right)\left(\frac{b_\eta - a_\eta}{2}\right)\left(\frac{b_\xi - a_\xi}{2}\right).$$

Não é necessário considerar o mesmo número de pontos de integração nas três direções, mas, por conveniência, na implementação utilizamos $n_x = n_y = n_z$.

Utilizando a fórmula de quadratura Gaussiana em (5.14) e os dados da Tabela 1.2, podemos calcular a integral em (5.7), que representa a contribuição de um elemento de densidade para a matriz de rigidez do elemento de deslocamento. Neste caso, consideramos a função matricial $f(\xi, \eta, \zeta) = \mathrm{B}(\xi, \eta, \zeta)^T \mathrm{DB}(\xi, \eta, \zeta)$.

Se o elemento de deslocamento está dividido em $N_n = (n_{mr})^3$ elementos de densidade, então precisamos obter N_n matrizes $\mathrm{K}^{(e_i)}$, calculando uma integral da forma (5.7) para cada elemento de densidade e_i, $i = 1, 2, ..., N_n$, que está contido no elemento de

deslocamento e. A matriz é diferente para cada e_i, pois os intervalos de integração mudam. Felizmente, no nosso caso as malhas são regulares, isto é, possuem todos os elementos do mesmo tipo e tamanho. Assim, só precisamos calcular uma vez cada uma das N_n matrizes $\mathrm{K}^{(e_i)}$ dos elementos de densidade, pois elas se repetem em cada elemento de deslocamento.

Por fim, a matriz de rigidez $\mathrm{K}^{(e)}$ de um elemento de deslocamento é calculada como a combinação linear das matrizes $\mathrm{K}^{(e_i)}$ de seus elementos de densidade, ponderadas pelos termos de penalização das densidades de material, ρ_{e_i}. Isto é,

$$\mathrm{K}^{(e)} = \sum_{i=1}^{N_n} (E_{min} + (\rho_{e_i})^p (E_0 - E_{min}))\, \mathrm{K}^{(e_i)}. \tag{5.15}$$

A matriz de rigidez global, K, é construída da mesma forma que na otimização topológica tradicional, superpondo as matrizes $\mathrm{K}^{(e)}$ para todos os elementos da malha.

5.4 Cálculo dos gradientes

Para a aplicação dos métodos de otimização que envolvem primeiras derivadas, como é o caso da programação linear sequencial, precisamos calcular os gradientes da função objetivo e das restrições do problema de otimização topológica estrutural, ou seja, $\nabla f(\mathrm{x}^{(k)})$ e $\mathrm{A}(\mathrm{x}^{(k)})$, sendo $\mathrm{x}^{(k)}$ o vetor das variáveis de projeto na iteração k. Na multirresolução, uma vez que o problema de otimização é resolvido na malha intermediária (das variáveis de projeto), o cálculo dos gradientes também é feito nessa malha.

Vimos em (5.1) que a densidade ρ_i de um elemento de densidade i é calculada como uma média ponderada dos valores das variáveis de projeto na vizinhança desse elemento. A restrição de volume no problema (5.3) depende das densidades e pode ser escrita como

$$c(\rho) = \sum_{i=1}^{n_\rho} v_i \rho_i - V_{max} \leq 0.$$

A derivada dessa restrição com relação à variável x_k é

$$\frac{\partial c(\rho)}{\partial x_k} = \frac{\partial}{\partial x_k} \left(\sum_{i=1}^{n_\rho} v_i \rho_i \right) = \sum_{i=1}^{n_\rho} v_i \frac{\partial \rho_i}{\partial x_k}.$$

Além disso, de (5.1) temos

$$\frac{\partial \rho_i}{\partial x_k} = \frac{\partial}{\partial x_k} \left(\sum_{j \in B(i, r_{min})} \frac{\omega_{ij}}{\omega_i} x_j \right) = \begin{cases} \dfrac{\omega_{ik}}{\omega_i} & \text{se } \; k \in B(i, r_{min}), \\ 0 & \text{se } \; k \notin B(i, r_{min}). \end{cases}$$

Logo,

$$\frac{\partial c(\rho)}{\partial x_k} = \sum_{i \in \bar{B}(k, r_{min})} v_i \frac{\omega_{ik}}{\omega_i},$$

em que $\bar{B}(k, r_{min})$ denota o conjunto dos elementos de densidade cujos centros estão a uma distância menor ou igual a r_{min} do centro do elemento k na malha de variáveis de

projeto. Assim, calculamos as componentes da matriz Jacobiana $\mathrm{A}(\mathrm{x}^{(k)})$, que neste caso possui uma única linha. Se considerarmos que todos os elementos na malha possuem volume fixo, é possível calcular a matriz Jacobiana uma única vez.

A função objetivo do problema é dada por

$$f(\rho) = \mathrm{f}^T \mathrm{u}(\rho) = \mathrm{u}(\rho)^T \mathrm{K}(\rho)\mathrm{u}(\rho).$$

Pela regra da cadeia, a derivada da função f com relação à variável x_k é

$$\frac{\partial f(\rho)}{\partial x_k} = \sum_{i=1}^{n_\rho} \frac{\partial f(\rho)}{\partial \rho_i}\frac{\partial \rho_i}{\partial x_k} = \sum_{i \in \bar{B}(k, r_{min})} \frac{\partial f(\rho)}{\partial \rho_i}\frac{\omega_{ik}}{\omega_i}.$$

Vejamos como calcular as derivadas de f com relação as densidades ρ_i.

Denotando por K_e a matriz com mesma dimensão da matriz K, mas formada apenas pelas componentes da matriz de rigidez de um único elemento de deslocamento e, temos

$$\mathrm{K}(\rho) = \sum_{e=1}^{n_{el}} \mathrm{K}_e = \sum_{e=1}^{n_{el}} \left(\sum_{i \in \Omega_e} (E_{min} + (\rho_i)^p (E_0 - E_{min})) \mathrm{K}_i \right),$$

em que K_i é uma matriz com mesma dimensão da matriz K, mas formada apenas pelas componentes da matriz $\mathrm{K}^{(e_i)}$ do elemento de densidade i. A notação $i \in \Omega_e$ indica que o elemento de densidade i está "dentro" do elemento de deslocamento e, se considerarmos as malhas sobrepostas. Assim,

$$\frac{\partial \mathrm{K}(\rho)}{\partial \rho_i} = p(\rho_i)^{p-1}(E_0 - E_{min})\mathrm{K}_i.$$

Derivando o sistema de equilíbrio, $\mathrm{K}(\rho)\mathrm{u}(\rho) = \mathrm{f}$, com respeito à variável ρ_i, obtemos:

$$\frac{\partial \mathrm{K}(\rho)}{\partial \rho_i}\mathrm{u}(\rho) + \mathrm{K}(\rho)\frac{\partial \mathrm{u}(\rho)}{\partial \rho_i} = 0 \Rightarrow \frac{\partial \mathrm{u}(\rho)}{\partial \rho_i} = -\mathrm{K}(\rho)^{-1}\frac{\partial \mathrm{K}(\rho)}{\partial \rho_i}\mathrm{u}(\rho)$$

$$\Rightarrow \frac{\partial \mathrm{u}(\rho)}{\partial \rho_i} = -\mathrm{K}(\rho)^{-1}\left[p(\rho_i)^{p-1}(E_0 - E_{min})\mathrm{K}_i\right]\mathrm{u}(\rho).$$

Dessa forma, temos

$$\frac{\partial f(\rho)}{\partial \rho_i} = \mathrm{f}^T\frac{\partial \mathrm{u}(\rho)}{\partial \rho_i} = \mathrm{u}(\rho)^T\mathrm{K}(\rho)\frac{\partial \mathrm{u}(\rho)}{\partial \rho_i} = -\mathrm{u}(\rho)^T\left[p(\rho_i)^{p-1}(E_0 - E_{min})\mathrm{K}_i\right]\mathrm{u}(\rho).$$

Assim, calculamos as derivadas da função objetivo com relação às variáveis de projeto,

$$\frac{\partial f(\rho)}{\partial x_k} = -\sum_{i \in \bar{B}(k, r_{min})} \mathrm{u}^T\left[p(\rho_i)^{p-1}(E_0 - E_{min})\mathrm{K}_i\right]\mathrm{u}\,\frac{\omega_{ik}}{\omega_i},$$

obtendo as componentes do vetor gradiente. Na prática, ao invés de expandir a matriz $\mathrm{K}^{(e_i)}$ do elemento de densidade i, formando a matriz K_i, podemos manter a matriz original e extrair do vetor u os deslocamentos dos nós correspondentes ao elemento e no qual o elemento de densidade i está contido.

5.5 O surgimento de artefatos

Infelizmente, a multirresolução pode apresentar um inconveniente na solução do problema de otimização topológica, que é o surgimento de regiões com o material desconexo, com elementos vazios entre elementos sólidos, formando partes "flutuando" ou barras incompletas na estrutura, conforme ilustra a Figura 5.4. Gupta *et al.* [26] fazem um estudo sobre o surgimento dessas regiões, que chamam de *artefatos* ou *padrões de códigos QR* (devido ao padrão criado pelos artefatos em problemas bidimensionais). Nesta seção, apresentamos um resumo das conclusões obtidas pelos autores.

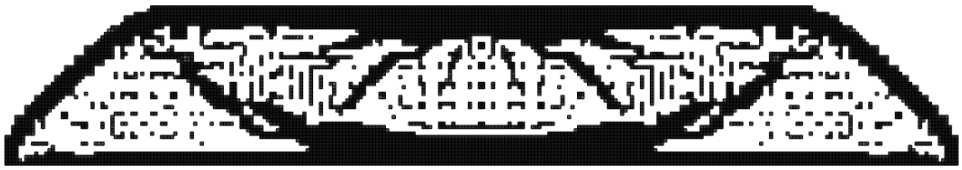


Figura 5.4: Estrutura bidimensional com artefatos.

De maneira geral, as regiões com os artefatos são interpretadas erroneamente como rígidas e são causadas pela baixa precisão no modelo de elementos finitos adotado, isto é, a interpolação dos deslocamentos feita por funções de forma compostas por polinômios de ordem pequena. Como a malha de densidades é mais fina do que a malha de deslocamentos, os erros causados pela baixa precisão crescem ao passarmos as informações de uma malha para a outra, fazendo com que a distribuição de densidades com os artefatos seja artificialmente rígida. Groen *et al.* [24] e Nguyen *et al.* [36] também reportam problemas com o surgimento de artefatos devido ao uso de elementos com grau baixo.

Em [26], os autores examinam casos bidimensionais elementares, nos quais a distribuição de material dentro de um único elemento finito (dividido em elementos de densidade) é otimizada para minimizar a flexibilidade. Utilizando elementos de grau 6 e sem a aplicação do filtro, testes realizados com diferentes condições de carregamento apresentaram resultados contendo artefatos, ainda que os valores da função objetivo fossem muito menores do que aqueles obtidos ao resolver o problema na malha mais fina com o método tradicional, indicando que as estruturas com artefatos são interpretadas como mais rígidas. Além disso, nota-se que os elementos sólidos "flutuando" livremente também são deformados, o que significa que uma carga considerável é transferida através do vazio. Naturalmente, o vazio deveria ter uma rigidez insignificante e ser deformado significativamente. Contudo, as deformações nas áreas vazias ficam comparáveis com aquelas das partes sólidas, indicando que o vazio possui, portanto, uma rigidez artificial.

Em outro teste, os autores consideram que a distribuição inicial do material no elemento contém uma parte sólida e uma lacuna vazia de altura h_v. Tiras finas de vazio podem surgir na estrutura durante o processo de otimização topológica e, neste caso, a força aplicada nessas regiões deve se anular ou as funções de forma devem ser capazes de modelar corretamente a descontinuidade de material. Variando o grau do elemento e o parâmetro h_v, os resultados mostram que:

- Para valores pequenos do grau do elemento (1, 2 ou 3), a precisão obtida no cálculo da flexibilidade é baixa em todos os casos;
- Para lacunas mais largas ($h_v = 0{,}9$), grau 4 foi suficiente;

- Quanto menor o tamanho da lacuna, maior é o grau necessário para se obter uma boa representação do campo de deslocamentos;
- Entretanto, para lacunas muito finas ($h_v = 0{,}1$), nem mesmo grau 12 foi adequado.

Com valores pequenos do grau do elemento, os deslocamentos na parte vazia são bastante subestimados. Já com polinômios interpoladores de ordem maior, os deslocamentos para uma distribuição de material descontínua podem ser representados de maneira mais precisa. Todavia, se a parte vazia for muito fina, até mesmo com uma ordem alta dos polinômios, os deslocamentos são mal previstos, devido à limitação das funções de forma polinomiais em representar a mudança brusca de deslocamento perto da descontinuidade de material.

Também é possível que o algoritmo de otimização convirja para a solução contendo artefatos em virtude da não-unicidade do campo das variáveis de projeto. Gupta *et al.* [27] estudam as limitações presentes na separação das malhas com a multirresolução e concluem que, para se ter unicidade do campo das variáveis de projeto, é necessário que a ordem dos polinômios interpoladores que compõem as funções de forma e os parâmetros da multirresolução sejam escolhidos de acordo com alguns limitantes. Sejam $\mathrm{K}^{(e)}$ a matriz de rigidez do elemento e m o número de variáveis de projeto utilizadas para definir a distribuição de material dentro de cada elemento finito. Nesse caso, devemos ter

$$m \leq posto(\mathrm{K}^{(e)}). \tag{5.16}$$

No caso tridimensional, desconsiderando as condições de contorno do problema, o posto da matriz $\mathrm{K}^{(e)}$ é igual ao número de graus de liberdade do elemento menos 6, por conta dos seis possíveis movimentos de corpo rígido dos pontos do elemento (três translações nas direções dos eixos coordenados e três rotações, em torno de cada eixo). Assim, $posto(\mathrm{K}^{(e)}) = 3n_e - 6$, em que n_e é o número de nós do elemento. Além disso, neste trabalho consideramos que a quantidade de variáveis de projeto contidas em cada elemento finito é $m = (d_{mr})^3$.

Na Tabela 5.1, apresentamos o limitante superior para m, bem como o limitante superior para d_{mr}, nos casos dos elementos prismáticos retangulares do tipo Lagrange e do tipo *serendipity* de grau 1, 2 ou 3, que utilizamos neste trabalho. Na prática, a qualidade dos resultados também depende do raio do filtro e do parâmetro n_{mr}, de modo que é possível encontrarmos resultados razoáveis mesmo ultrapassando o limite para d_{mr}.

Tabela 5.1: Número máximo de variáveis de projeto para alguns tipos de elementos finitos prismáticos retangulares.

Tipo do elemento	**Lim. sup. para m**	**Lim. sup. para d_{mr}**
Lagrange de grau 1	18	2
Lagrange de grau 2	75	4
Lagrange de grau 3	186	5
Serendipity de grau 2	54	3
Serendipity de grau 3	90	4

Ademais, a penalização das densidades de material intermediárias (modelo SIMP) também pode contribuir para a formação dos artefatos, pois ela fornece aos padrões

vazio-sólido (solução discreta) uma vantagem adicional sobre distribuições de material com densidades intermediárias.

Saber como detectar os artefatos ainda é um problema em aberto e, geralmente, só conseguimos percebê-los depois que a solução final é obtida. Entretanto, observamos que os artefatos podem não ser tão aparentes como ocorre na Figura 5.4, ou seja, a solução obtida com a multirresolução pode ser imprecisa mesmo que a estrutura não tenha regiões estranhas visivelmente. Uma maneira heurística de verificar a precisão da solução é comparar o valor da função objetivo com aquele que seria obtido ao resolver o mesmo problema na malha fina com o método tradicional. Se o valor da função for muito menor do que o de referência, é provável que a estrutura contenha artefatos, pois sua rigidez tende a ser artificialmente maior e, portanto, a flexibilidade será menor. Porém, o cálculo do valor da função de referência pode ser impraticável, devido à falta de memória computacional, uma vez que o propósito da multirresolução é obter estruturas com resoluções muito altas, que normalmente não conseguimos obter com o método tradicional.

Atualmente, as estratégias conhecidas para amenizar o problema dos artefatos são: o aumento do grau dos elementos finitos e a aplicação de um filtro de densidades. Assim como no caso dos tabuleiros de xadrez, o filtro faz com que o material seja distribuído de maneira mais homogênea, evitando a formação das regiões com artefatos. Entretanto, para que a estrutura não tenha artefatos, é necessário que o raio do filtro não seja pequeno demais. Quanto menor for o raio, mais detalhes tendem a surgir na estrutura, como a formação de novas barras, de modo que podem aparecer lacunas e mudanças bruscas na distribuição de material. Conforme vimos, isso pode favorecer o surgimento dos artefatos. Neste caso, se desejamos reduzir o raio do filtro, é necessário também aumentar o grau de aproximação dos deslocamentos.

A desvantagem em aumentar o grau dos elementos é o crescimento do custo computacional, uma vez que o número de nós cresce, elevando a dimensão do sistema linear de equilíbrio. Mesmo com o sistema sendo resolvido na malha mais grossa, o aumento excessivo do grau dos elementos pode fazer com que percamos a eficiência conquistada com a multirresolução. Em problemas tridimensionais, deve ser suficiente utilizarmos no máximo grau 3, já que valores maiores tornam o algoritmo ineficiente. Assim, precisamos combinar o aumento do grau com a utilização de um raio mínimo para o filtro de densidades. Além disso, é importante que a desigualdade em (5.16) seja satisfeita, embora, na prática, conseguimos obter bons resultados utilizando valores de m acima do limite, caso a aplicação do filtro seja capaz de eliminar os artefatos.

Neste trabalho, em geral, obtivemos bons resultados utilizando a multirresolução com elementos lineares e a aplicação de um filtro, porém as soluções obtidas são independentes da malha, de modo que não ganhamos informações adicionais sobre o formato da estrutura ao aumentarmos a resolução. Com o intuito de obter estruturas mais detalhadas sem a presença de artefatos, reduzimos o raio de aplicação do filtro até certo ponto e propomos uma estratégia adaptativa para aumentar o grau dos elementos, suprimindo algumas variáveis de modo a prejudicar o mínimo possível a eficiência do algoritmo. Essa estratégia será apresentada no próximo capítulo.

Capítulo 6

Estratégias Adicionais

Neste capítulo, propomos estratégias adicionais para a melhoria do algoritmo de resolução dos problemas de otimização topológica. A primeira delas é o arredondamento de densidades, com o qual podemos obter soluções em que a distribuição de material é binária ou possui poucas densidades intermediárias, ou seja, conseguimos obter estruturas compostas por regiões completamente sólidas ou vazias. Em seguida, propomos uma estratégia adaptativa de aumento do grau dos elementos finitos, com o intuito de obter soluções mais precisas utilizando a multirresolução, reduzindo a presença de artefatos sem prejudicar demais a eficiência do algoritmo.

6.1 Arredondamento de densidades

Na resolução do problema de otimização topológica, dividimos o domínio no qual a estrutura deve ficar contida em pequenos elementos e precisamos decidir se cada elemento será vazio ou sólido, de modo que a distribuição de material minimize a flexibilidade da estrutura e satisfaça uma restrição de volume. Entretanto, para evitar um problema de otimização inteira, permitimos que as variáveis assumam valores entre 0 e 1, considerando que cada elemento terá uma densidade $\rho_i \in [0, 1]$, e aplicamos o modelo SIMP para diminuir a ocorrência de densidades intermediárias.

Sejam V o volume total do domínio e v_{frac} a fração predeterminada de volume que a estrutura poderá ocupar. Supondo que o domínio tenha sido dividido em uma malha com n_{el} elementos, a restrição de volume do problema é dada por

$$\sum_{i=1}^{n_{el}} v_i \rho_i \leq v_{frac} V, \tag{6.1}$$

em que v_i é o volume e ρ_i é a densidade do elemento i. O modelo SIMP, junto com a restrição (6.1), faz com que a solução tenha poucas densidades intermediárias. Contudo, a aplicação do filtro pode fazer com que essas densidades apareçam com mais frequência, geralmente formando camadas intermediárias entre as partes sólidas e as vazias. O desenvolvimento de técnicas para obter soluções completamente binárias é um assunto relevante na otimização topológica.

O objetivo do arredondamento de densidades é eliminar as densidades intermediárias que fazem parte do vetor solução do problema, sem violar a restrição de volume

(6.1). Dessa forma, após o arredondamento, esperamos que exatamente $\lfloor v_{frac}\, n_{el} \rfloor$ elementos tenham densidade 1 e que todos os demais tenham densidade 0.

Para efetuar o arredondamento, dispomos de duas estratégias, das quais uma é baseada apenas no valor das densidades, enquanto a outra leva em conta o gradiente do Lagrangiano. Ao apresentarmos essas estratégias, suporemos que ρ é o vetor de densidades antes do arredondamento e que $\tilde{\rho}$ é o vetor de densidades arredondadas. Além disso, $\nabla\mathcal{L}$ representará o gradiente do Lagrangiano.

Independentemente da estratégia adotada, as densidades são sempre arredondadas quando estão muito próximas dos limites 0 ou 1, da seguinte forma:

- Se $\rho_i \geq t_u$, arredondamos ρ_i para 1.
- Se $\rho_i \leq t_l$, arredondamos ρ_i para 0.

Em nossos testes, utilizamos $t_u = 0{,}95$ e $t_l = 0{,}05$.

6.1.1 Estratégia baseada no valor das densidades

A primeira estratégia de arredondamento adotada pelo algoritmo consiste em realizar os seguintes passos:

(1) Dispor as densidades na ordem decrescente de seus valores;

(2) Selecionar as $\lfloor v_{frac}\, n_{el} \rfloor$ primeiras densidades e arredondá-las para 1;

(3) Atribuir 0 a todas as demais densidades.

Após essa tentativa de arredondamento, conferimos se o vetor $d = \tilde{\rho} - \rho$ é uma direção de descida para o gradiente do Lagrangiano, ou seja, se

$$-\frac{\nabla\mathcal{L}^T d}{\|\nabla\mathcal{L}\|\|d\|} > cos(\vartheta_{max}), \tag{6.2}$$

em que ϑ_{max} é um parâmetro que estipula o ângulo máximo admitido entre $-\nabla\mathcal{L}$ e d. Em nossos testes, utilizamos $\vartheta_{max} = 89{,}9°$.

Se a condição (6.2) for satisfeita, aceitamos o vetor $\tilde{\rho}$ como a nova solução. Caso contrário, recorremos à estratégia baseada no gradiente do Lagrangiano.

6.1.2 Estratégia baseada no gradiente do Lagrangiano

A segunda estratégia de arredondamento das densidades consiste em projetar o vetor $\hat{\rho} = \rho - \alpha\nabla\mathcal{L}$ sobre a caixa $0 \leq \rho \leq 1$. Nesse caso, precisamos determinar o maior valor possível de α para que a solução projetada, $\tilde{\rho}$, satisfaça a condição de ângulo (6.2), bem como as salvaguardas definidas abaixo.

- Só podem ser arredondadas para 1 as densidades que satisfizerem $\rho_i \geq v^u_{min}$.
- Só podem ser arredondadas para 0 as densidades que satisfizerem $\rho_i \leq v^l_{max}$.

Em nossos experimentos, adotamos $v^u_{min} = 0{,}3$ e $v^l_{max} = 0{,}7$. Essas salvaguardas evitam que densidades próximas de 0 sejam arredondadas para 1 e vice-versa, o que pode causar instabilidades no algoritmo devido à mudança brusca na solução.

Dados os vetores ρ e $\nabla\mathcal{L}$, se a componente i do gradiente do Lagrangiano for positiva, o valor da densidade ρ_i diminui quando se anda na direção de $-\nabla\mathcal{L}$, indicando que ela pode ser arredondada para 0. Por outro lado, se $\nabla_i\mathcal{L}$ for negativa, o valor da densidade aumenta, indicando que ela pode ser arredondada para 1. Na prática, podemos determinar, para cada componente i, qual é o valor α_i que faz com que $\tilde{\rho}_i = \rho_i - \alpha_i\nabla_i\mathcal{L} = 0$ (se $\nabla_i\mathcal{L} > 0$) ou $\tilde{\rho}_i = \rho_i - \alpha_i\nabla_i\mathcal{L} = 1$ (se $\nabla_i\mathcal{L} < 0$).

Assim, criamos uma lista dos valores α_i, para $i = 1, 2, ..., n_{el}$, em ordem crescente e adotamos o maior valor de α, tal que $\tilde{\rho} = \rho - \alpha\nabla\mathcal{L}$ satisfaça as duas salvaguardas acima, bem como a condição de ângulo.

Observe que, dadas essas condições, é possível que algumas densidades mudem de valor, mas permaneçam intermediárias, isto é, não sejam arredondadas para 0 ou 1.

6.1.3 Controle da qualidade da solução arredondada

Se a estratégia de arredondamento baseada no valor das densidades é adotada e o vetor $\tilde{\rho}$ é aceito, ou seja, se a condição de ângulo é satisfeita, então a solução obtida contém apenas valores iguais a 0 ou 1. Por outro lado, quando se recorre à estratégia baseada no gradiente do Lagrangiano, não há garantia de que todas as densidades intermediárias sejam eliminadas.

Além disso, a solução arredondada, $\tilde{\rho}$, pode não corresponder a um mínimo local da função objetivo do problema de otimização topológica e pode até mesmo não satisfazer a restrição de volume.

Para controlar a qualidade da solução obtida, aplicamos novamente o algoritmo da programação linear sequencial (PLS), usando $\tilde{\rho}$ como vetor de densidades inicial. Esse processo de resolução do problema e arredondamento das densidades é repetido até que algum dos critérios abaixo seja satisfeito.

- A diferença entre duas soluções arredondadas consecutivas seja relativamente pequena e a violação na restrição de volume esteja dentro de uma tolerância, ou seja,

$$\|\tilde{\rho}_k - \tilde{\rho}_{k-1}\|_1 < \varepsilon_N\|\tilde{\rho}_{k-1}\|_1 \quad \text{e} \quad \sum_{i=1}^{n_{el}} v_i\rho_i \leq v_{frac}\,V + \varepsilon_V.$$

- O número máximo de tentativas de arredondamento seja atingido.

Em nossos experimentos, empregamos as tolerâncias $\varepsilon_N = 0{,}01$ e $\varepsilon_V = 0{,}005$ e o máximo de 10 tentativas de arredondamento.

Para acelerar esse processo iterativo de redução das densidades intermediárias, as novas aplicações da PLS ao problema de otimização topológica podem adotar parâmetros diferentes daqueles empregados para a obtenção da primeira solução. Por exemplo, podemos utilizar apenas elementos finitos de grau 1, mesmo que elementos de grau maior tenham sido empregados anteriormente. Essa é uma boa estratégia quando a solução antes do arredondamento já contém densidades muito próximas de 0 ou 1, mas é perigosa

quando isso não ocorre, motivo pelo qual decidimos não utilizar essa estratégia em nossos testes. Ademais, podemos redefinir o raio de aplicação do filtro. Adotamos $r_{min}^{prj} = 1{,}1$ como raio nas novas aplicações da PLS sempre que esse valor for menor que o raio utilizado inicialmente.

6.1.4 Projeção com base no filtro de Heaviside

Quando aplicamos a estratégia baseada no gradiente do Lagrangiano para arredondar as densidades, costuma haver um número grande de variáveis cujas componentes correspondentes do gradiente são positivas, o que as leva para zero. Nesse caso, é possível que o número de variáveis cujo gradiente é negativo seja menor do que o necessário para manter o volume desejado.

O propósito do filtro suave de Heaviside é tornar mais próximas de 1 as densidades maiores que um valor $\eta \in (0, 1)$, bem como tornar mais próximas de 0 as densidades menores que η. Naturalmente, o valor de η deve ser selecionado de modo que $\lfloor v_{frac}\, n_{el} \rfloor$ densidades se aproximem de 1 e que as demais se aproximem de 0.

Neste trabalho, utilizamos a versão do filtro de Heaviside baseada na tangente hiperbólica, proposta por Wang *et al.* [54]. Nesse caso, a densidade projetada de um elemento i é dada por

$$h(\rho_i, \beta, \eta) = \frac{\tanh(\beta\eta) + \tanh(\beta(\rho_i - \eta))}{\tanh(\beta\eta) + \tanh(\beta(1 - \eta))},$$

em que ρ_i é a densidade original do elemento e β é um parâmetro de projeção que indica quão arrojada será a estratégia de arredondamento. A Figura 6.1 mostra os gráficos da função h com $\eta = 0{,}5$ e diferentes valores de β.

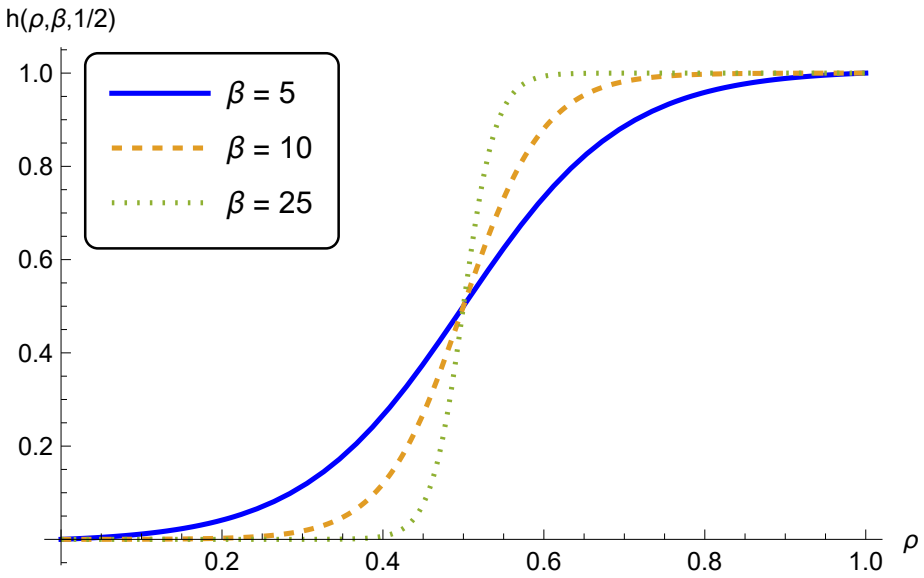


Figura 6.1: Função de Heaviside com $\eta = 0{,}5$ e diferentes valores de β.

Para assegurar que o volume desejado da estrutura seja alcançado, escolhemos o valor de η utilizando a estratégia proposta por Xu *et al.* [58]. Dado um valor do parâmetro β, supondo que ρ seja o vetor original de densidades e que h forneça as densidades

projetadas, o valor de η que preserva o volume atual da estrutura corresponde ao zero da função dada por

$$f(\eta) = \sum_{i=1}^{n_{el}} v_i h(\rho_i, \beta, \eta) - \sum_{i=1}^{n_{el}} v_i \rho_i.$$

Observe que $h(\rho_i, \beta, 0) \geq \rho_i$, de modo que $f(0) > 0$ se existir ao menos um elemento com densidade maior do que 0. De forma análoga, $h(\rho_i, \beta, 1) \leq \rho_i$, o que faz com que $f(1) < 0$. Sendo assim, há sempre um valor $\eta^* \in (0, 1)$ tal que $f(\eta^*) = 0$, pelo teorema do valor intermediário. Na prática, encontramos o valor η^* utilizando o método de Newton.

Em nossa implementação, a projeção de Heaviside é aplicada ao vetor de densidades encontrado como solução após aplicarmos a PLS, imediatamente antes de aplicarmos as estratégias de arredondamento, pois ela impele as densidades para as proximidades dos limites 0 ou 1. Além disso, ela é aplicada novamente após o arredondamento, com o objetivo de preservar o volume desejado.

Fornecemos um valor inicial, β_0, para o parâmetro β. Esse valor é atualizado após o arredondamento das densidades, empregando-se a fórmula

$$\beta_{k+1} \leftarrow \min(\delta_\beta \beta_k, \beta_{max}),$$

em que β_{max} é o valor máximo admitido para β. Com essa estratégia, as densidades tornam-se cada vez mais próximas de 0 ou 1 a cada nova aplicação da PLS. Em nossos testes, utilizamos $\beta_0 = 1{,}0$, $\delta_\beta = 2{,}0$ e $\beta_{max} = 1000$.

6.2 Estratégia adaptativa de aumento do grau dos elementos finitos

A multirresolução, descrita no Capítulo 5, permite resolver os problemas de otimização topológica de grande porte de maneira mais eficiente. A ideia é trabalhar com refinamentos diferentes em cada etapa do processo: uma malha mais grossa para a aproximação dos deslocamentos (malha de deslocamentos), uma malha intermediária para a resolução do problema de otimização (malha de variáveis de projeto) e uma malha mais fina na qual é definida a distribuição das densidades de material (malha de densidades). Assim, o custo computacional é reduzido, pois as etapas mais caras são realizadas em malhas mais grossas, enquanto a resolução final da estrutura é alta. Em contrapartida, a multirresolução pode apresentar regiões com rigidez artificial, formando artefatos indesejados na estrutura. Conforme vimos na Seção 5.5, a causa desse infortúnio é a baixa precisão na aproximação dos deslocamentos, que é feita na malha mais grossa.

As únicas maneiras conhecidas atualmente para amenizar o problema dos artefatos são o aumento do grau dos elementos finitos e a aplicação de um filtro de densidades. Aumentar o grau dos elementos na malha de deslocamentos faz com que a aproximação dos deslocamentos seja mais precisa, porém o custo para isso é elevado, já que o número de nós em cada elemento cresce bastante conforme aumentamos o grau. Por outro lado, a aplicação do filtro de densidades precisa ser feita com um raio que não seja muito pequeno para evitar o aparecimento dos artefatos. Assim, os detalhes no formato da estrutura que seriam obtidos com a resolução alta podem ser perdidos ao utilizarmos o filtro com um raio

grande. Se desejamos obter soluções mais precisas e estruturas mais detalhadas sem perder a eficiência do algoritmo, será necessário balancear o aumento do grau dos elementos e a redução do raio do filtro.

Neste trabalho, testamos a utilização de raios pequenos para o filtro e propomos uma estratégia adaptativa de aumento do grau dos elementos finitos para corrigir os artefatos na estrutura e melhorar a qualidade da solução. Essa estratégia consiste em realizar as seguintes etapas:

(1) Obter a solução utilizando elementos de grau 1;

(2) Escolher variáveis que podem ser fixadas e "eliminá-las" do problema;

(3) Utilizando como aproximação inicial a solução encontrada, resolver o problema novamente com elementos de grau 2. Se necessário, usar elementos de grau 3 ou maior;

(4) Arredondar as densidades e controlar a qualidade da solução.

Inicialmente, resolvemos o problema utilizando elementos lineares. Se o raio do filtro for pequeno (menor que o tamanho dos elementos de deslocamento, por exemplo), é bastante provável que a distribuição de material encontrada contenha artefatos. Contudo, essa solução já possui informações relevantes sobre o formato da estrutura, de modo que ela pode ser uma boa aproximação inicial para a resolução do problema com elementos de grau maior. Como a aproximação inicial não precisa ser exata, para acelerar o processo adaptativo, podemos relaxar a tolerância utilizada no critério de parada da PLS, descrito na Seção 2.5. Se ε_g é a tolerância desejada para a norma do gradiente projetado, utilizamos $\tilde{\varepsilon}_g = 10 \times \varepsilon_g$ como tolerância para obter a solução inicial. Na resolução com o grau máximo, utilizamos a tolerância ε_g original.

Na etapa (2), após obter a solução com elementos de grau 1, com base nas regiões sólidas ou vazias da estrutura, escolhemos algumas variáveis que podem ser suprimidas do problema, tendo seus valores fixados. Essa escolha será descrita com mais detalhes na próxima subseção.

Na etapa (3), aumentamos o grau dos elementos para 2 e, utilizando a solução obtida anteriormente como aproximação inicial, resolvemos o problema novamente. Esperamos que a resolução do problema com elementos de grau maior gaste menos iterações, já que estamos fornecendo uma aproximação inicial que contém informações sobre o formato da estrutura, e que consuma menos tempo, pois suprimimos algumas variáveis do problema na etapa (2). O objetivo é que essa nova resolução elimine os artefatos presentes na solução inicial e seja mais barata do que aquela obtida resolvendo-se o problema com elementos de grau 2 logo de início. Entretanto, pode ser que a nova solução ainda seja imprecisa e contenha artefatos. Nesse caso, podemos resolver o problema novamente aumentando o grau para 3 e assim sucessivamente até atingirmos algum critério de satisfação com relação à solução obtida.

Infelizmente, não sabemos como detectar analiticamente a presença de artefatos. Além disso, em problemas tridimensionais, aumentar o grau dos elementos para 4 ou mais pode prejudicar bastante a eficiência do algoritmo. Em nossa implementação, utilizamos um parâmetro que controla o valor máximo para o grau dos elementos, que pode ser 2 ou 3.

Por fim, na etapa (4), aplicamos a estratégia de arredondamento de densidades, descrita na Seção 6.1, e controlamos a qualidade da solução, conforme explicado na Subseção 6.1.3, para obter a estrutura final.

6.2.1 Escolha das variáveis que serão fixadas

Com o propósito de reduzir o tempo de resolução do problema utilizando a multirresolução com elementos de grau maior do que 1, obtemos a solução inicial com elementos lineares e escolhemos algumas variáveis que podem ser suprimidas do problema, tendo seus valores fixados. A solução inicial fornece informações sobre o formato da estrutura, como a presença de regiões vazias ou sólidas. Essas informações são utilizadas para a escolha das variáveis que serão fixadas.

Para auxiliar o entendimento das ideias propostas, consideremos um problema da viga MBB quase bidimensional (com apenas uma camada de elementos na direção z). A Figura 6.2 mostra a estrutura inicial obtida usando a multirresolução com $n_{mr} = d_{mr} = 2$, elementos de grau 1 e raio do filtro $r_{min} = 0{,}8$. Neste caso, consideramos que os lados dos elementos de deslocamento medem 1 unidade de comprimento, enquanto os lados dos elementos de densidade medem 0,5 unidade de comprimento, de modo que podemos aplicar o filtro com um raio menor do que 1. Na Figura 6.2, exibimos toda a malha de densidades, incluindo os elementos brancos com densidades iguais a zero. Notamos que a estrutura encontrada com elementos lineares possui várias regiões com artefatos. Vejamos como escolher as variáveis que serão fixadas antes de resolver o problema novamente com elementos de grau maior.

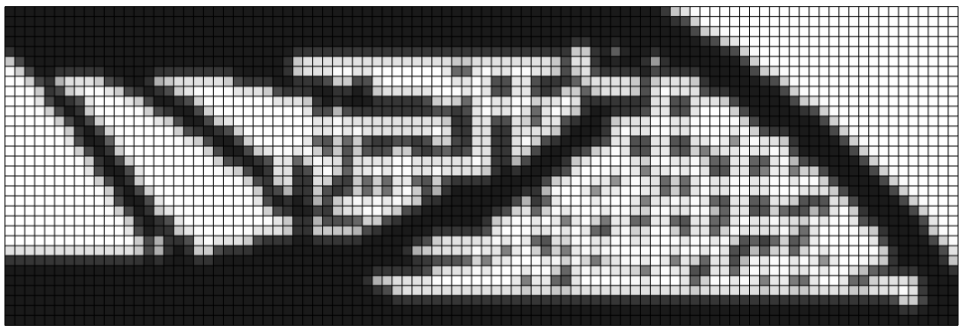


Figura 6.2: Viga MBB bidimensional obtida com a multirresolução, $n_{mr} = d_{mr} = 2$, raio do filtro $r_{min} = 0{,}8$ e elementos finitos de grau 1.

Nos problemas tridimensionais, consideramos que cada elemento de deslocamento é composto por $N_n = (n_{mr})^3$ elementos de densidade e contém $N_d = (d_{mr})^3$ variáveis de projeto. Suponhamos que um elemento de densidade e_i dentro do elemento de deslocamento e possua densidade ρ_{e_i} e denotemos por $\nabla_{e_j} f$ a componente do gradiente da função objetivo correspondente à variável de projeto j contida no elemento de deslocamento e. Um elemento de deslocamento será considerado *vazio* se satisfizer as condições:

- $\rho_{e_i} \leq \rho^l_{fix}$, para todo $i = 1, 2, ..., N_n$,
- $\nabla_{e_j} f \geq -\varepsilon_{grad}$, para todo $j = 1, 2, ..., N_d$,

em que ρ^l_{fix} e ε_{grad} são tolerâncias com valores positivos e próximos de zero. Em outras palavras, um elemento será considerado vazio se todas as densidades dentro dele forem

próximas de zero e se as variáveis de projeto nesse elemento não aumentarem demasiadamente quando se anda na direção de menos gradiente.

De maneira similar, um elemento de deslocamento será considerado *sólido* se satisfizer as condições:

- $\rho_{e_i} \geq \rho^u_{fix}$, para todo $i = 1, 2, ..., N_n$,
- $\nabla_{e_j} f \leq \varepsilon_{grad}$, para todo $j = 1, 2, ..., N_d$,

em que ρ^u_{fix} é uma tolerância com valor próximo de 1. Em outras palavras, um elemento será considerado sólido se todas as densidades dentro dele forem próximas de 1 e se as variáveis de projeto nesse elemento não diminuírem demasiadamente quando se anda na direção de menos gradiente.

As salvaguardas relacionadas ao gradiente da função objetivo garantem que, ao menos no início do processo de otimização, os valores das variáveis de projeto nos elementos vazios não aumentem e os valores das variáveis de projeto nos elementos sólidos não diminuam. Isso indica que podemos fixar os valores dessas variáveis.

Em nossos experimentos, utilizamos as tolerâncias $\rho^l_{fix} = 10^{-6}$, $\rho^u_{fix} = 0{,}9$ e $\varepsilon_{grad} = 10^{-6}$. Observamos que a tolerância ρ^l_{fix}, escolhida para encontrar densidades próximas de zero, deve ser mais restrita para evitar erros numéricos, pois desejamos suprimir do cálculo do gradiente as componentes relacionadas aos elementos vazios, com base na observação de que essas componentes têm valores próximos de zero.

No exemplo bidimensional, cada elemento de deslocamento é composto por $N_n = 2^2 = 4$ elementos de densidade. Além disso, como $n_{mr} = d_{mr}$, as densidades correspondem às variáveis de projeto. A Figura 6.3 exibe a estrutura com os elementos de deslocamento vazios destacados em azul com marcador "$\circ$" e os elementos sólidos destacados em vermelho com marcador "$\times$".

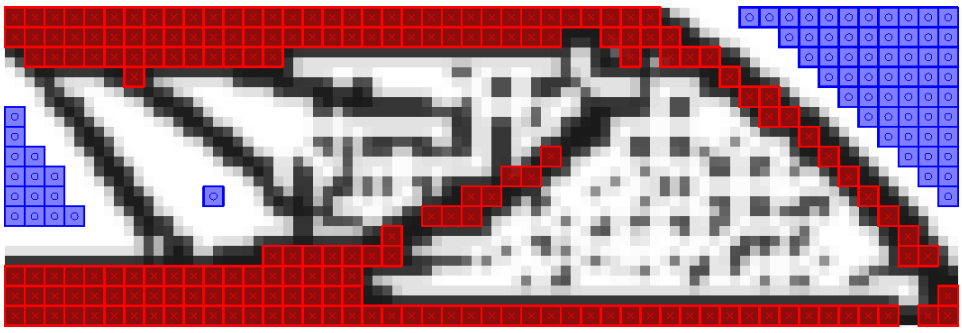


Figura 6.3: Estrutura com elementos de deslocamento vazios (em azul com marcador "$\circ$") e elementos sólidos (em vermelho com marcador "$\times$").

Nas iterações finais do processo de otimização topológica, a mudança na distribuição de material entre um iterando e o próximo geralmente ocorre nas regiões do contorno da estrutura, ou seja, onde há o encontro de uma parte sólida com uma parte vazia do domínio. Por esse motivo, podemos fixar as variáveis em elementos que são completamente vazios ou sólidos e manter no problema apenas as variáveis em elementos que possuem densidades mistas ou intermediárias. Para assegurar que a região do domínio na qual as variáveis podem ser atualizadas seja suficientemente grande, em nossa implementação consideramos que apenas elementos de deslocamento vazios cercados por vazios ou

elementos sólidos cercados por sólidos tenham variáveis fixadas. A Figura 6.4 exibe esses elementos para o exemplo bidimensional.

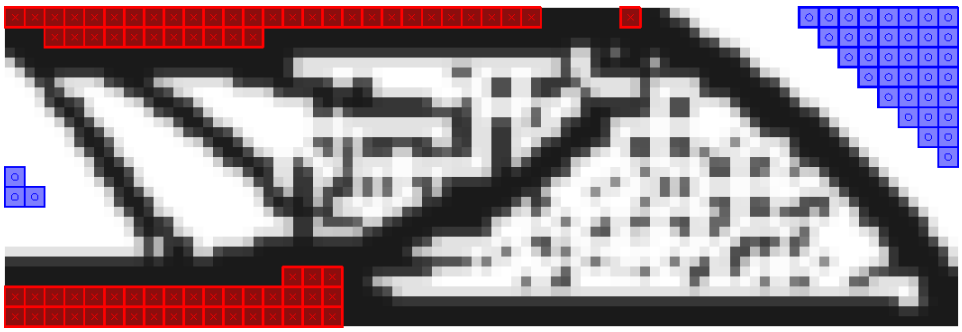


Figura 6.4: Estrutura com elementos vazios cercados por vazios (em azul com marcador "$\circ$") e elementos sólidos cercados por sólidos (em vermelho com marcador "$\times$").

Após determinarmos esses elementos, mantemos fixos os valores das variáveis de projeto neles contidas. Na Figura 6.5, destacamos em laranja com marcador "$*$" as variáveis de projeto (que também são densidades) escolhidas para serem fixadas no exemplo bidimensional.

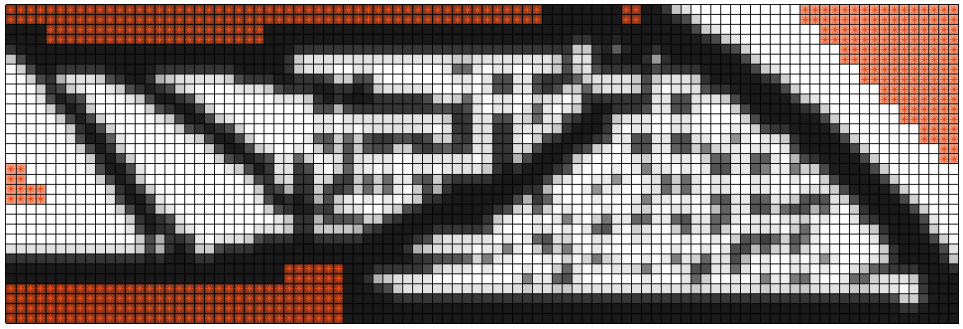


Figura 6.5: Estrutura com variáveis de projeto fixadas (em laranja com marcador "$*$").

As componentes correspondentes às variáveis de projeto fixadas podem ser retiradas da resolução dos subproblemas de programação linear. Como resolvemos o PL para obter o passo que será usado para atualizar as variáveis de projeto, podemos fazer com que os limitantes inferior e superior das componentes do passo correspondentes às variáveis fixadas sejam ambos iguais a zero. Assim, essas componentes serão nulas e as variáveis não serão alteradas.

Note que as densidades nos elementos escolhidos podem mudar com a aplicação do filtro e é importante que as variáveis de projeto nos elementos sólidos, de fato, façam parte do cálculo das densidades ao redor desses elementos. Na prática, como escolhemos elementos vazios (ou sólidos) cercados por vazios (ou sólidos) e utilizamos um raio de filtro pequeno (menor que o tamanho do elemento de deslocamento), além de fixar as variáveis de projeto, os valores das densidades nos elementos escolhidos não devem variar muito.

Como todas as densidades contidas nos elementos vazios cercados por vazios são próximas de zero, as componentes do gradiente correspondentes a esses elementos também terão valor próximo de zero, de modo que podemos suprimir essas componentes do cálculo do gradiente e atribuir valor 0 a elas.

Ademais, podemos fixar em zero os deslocamentos nodais internos aos elementos vazios, uma vez que a contribuição da matriz de rigidez desses elementos, calculada pela expressão (5.15), será aproximadamente nula, já que todas as densidades internas são aproximadamente iguais a zero. Contudo, não podemos fixar os deslocamentos dos elementos sólidos. Além disso, os graus de liberdade dos nós que pertencem a elementos vizinhos que não foram escolhidos para serem fixados devem permanecer no problema. Assim, fixamos apenas os deslocamentos dos nós que são compartilhados somente por elementos vazios cercados por vazios. Na Figura 6.6, exibimos a estrutura do exemplo bidimensional com a malha de deslocamentos e destacamos em verde com marcador "$*$" os nós cujos graus de liberdade serão suprimidos do problema. Note que a escolha desses nós deve ser feita após aumentarmos o grau dos elementos. Neste exemplo, utilizamos elementos do tipo Lagrange de grau 2.

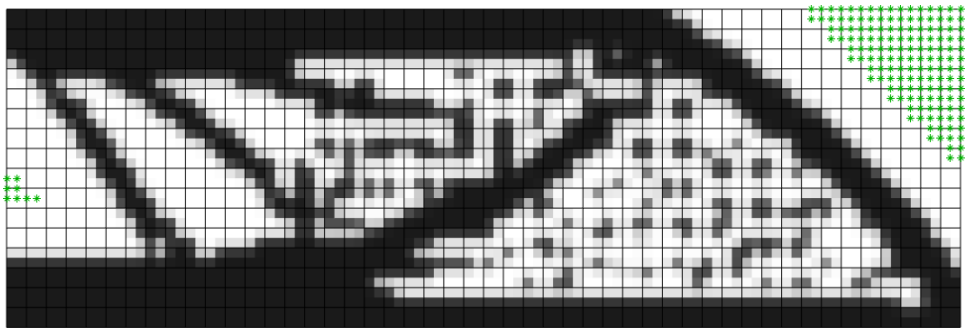


Figura 6.6: Estrutura com nós cujos graus de liberdade serão suprimidos do problema (em verde, com marcador "$*$").

Na prática, retiramos as componentes correspondentes aos graus de liberdade desses nós do vetor de cargas nodais, bem como as linhas e colunas da matriz de rigidez global, e fixamos os valores dos deslocamentos nodais em zero, de maneira similar àquela que fazemos com os graus de liberdade restringidos por apoios. Com isso, reduzimos a dimensão dos sistemas lineares de equilíbrio.

Isso pode ser feito pois as linhas (e colunas) da matriz de rigidez global correspondentes aos graus de liberdade dos nós internos aos elementos vazios são aproximadamente nulas, de modo que as equações podem ser removidas do sistema de equilíbrio e os valores das incógnitas (deslocamentos) podem ser fixados. Entretanto, na prática, as componentes dessas linhas da matriz têm a ordem de grandeza do valor E_{min} (módulo de Young do elemento vazio), utilizado justamente para evitar que a matriz seja singular devido à presença de linhas ou colunas nulas. Assim, para que não ocorram erros numéricos com a estratégia proposta, devemos utilizar um valor E_{min} próximo de zero, mas ainda grande o bastante para que a matriz não seja singular. Em nossos testes, obtivemos bons resultados utilizando $E_{min} = 10^{-9}E_0$, sendo E_0 o módulo de Young do material sólido.

Observamos também que após retirarmos as linhas e colunas da matriz de rigidez global, se estivermos utilizando o método dos gradientes conjugados com o *multigrid* como precondicionador na resolução dos sistemas lineares, será necessário refazer a fase de *setup*. Além disso, se o método utilizado for a fatoração de Cholesky, será necessário recalcular o vetor de permutação de mínimo grau da matriz.

Na Figura 6.7, apresentamos a estrutura final obtida para o exemplo desta seção, após eliminarmos as variáveis escolhidas, resolvermos o problema novamente utilizando elementos do tipo Lagrange de grau 2 e aplicarmos a estratégia de arredondamento

de densidades. Essa estrutura não contém artefatos aparentes, o que sugere que o aumento do grau dos elementos para 2 foi suficiente para melhorar a qualidade da solução.

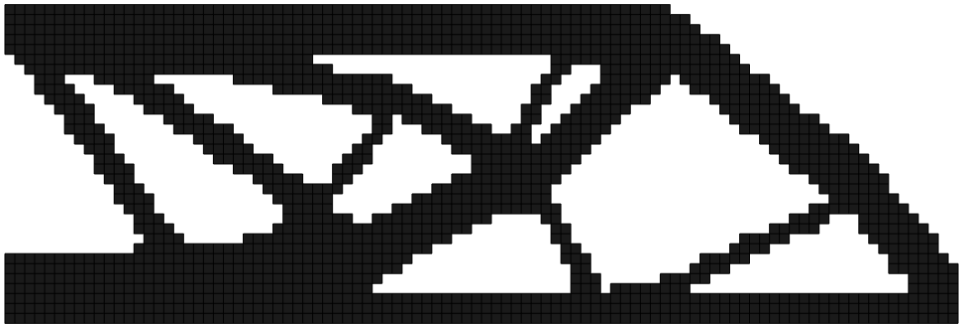


Figura 6.7: Estrutura final obtida após a resolução do problema com elementos de grau 2 e o arredondamento de densidades.

6.2.2 Adaptação das variáveis fixadas

No processo descrito anteriormente, escolhemos as variáveis que serão fixadas e suprimidas do problema uma única vez, após a resolução com elementos lineares. Contudo, no decorrer do processo de otimização topológica, podem surgir novas regiões vazias ou sólidas na estrutura. Dessa forma, a quantidade de variáveis escolhidas para serem fixadas pode aumentar ou até diminuir ao longo das iterações.

Além disso, escolher as variáveis que serão fixadas no início do processo, com base na solução obtida com grau 1, pode ser perigoso. Por exemplo, a estrutura pode conter uma barra incompleta com uma região vazia no meio. Se escolhermos fixar as variáveis de projeto dessa região em zero, o algoritmo pode encontrar dificuldades para convergir caso deseje preencher a barra com material.

Tendo isso em vista, propomos algumas estratégias de adaptação das variáveis que serão fixadas, refazendo a escolha ao longo das iterações.

- **Estratégia 1:** escolher as variáveis apenas uma vez, antes da resolução do problema com grau maior.
- **Estratégia 2:** escolher as variáveis em todas as iterações externas da PLS para o problema com grau maior.
- **Estratégia 3:** escolher as variáveis a cada n iterações externas da PLS.
- **Estratégia 4:** escolher as variáveis no início e na n-ésima iteração externa da PLS.

As escolhas são sempre feitas em iterações externas da PLS, isto é, em iterações nas quais o passo foi aceito. Nos resultados computacionais, comparamos cada uma dessas estratégias, com o objetivo de determinar qual delas é a mais adequada, isto é, a que torna o algoritmo mais eficiente.

Capítulo 7

Resultados Computacionais

Neste capítulo, apresentamos os resultados computacionais obtidos por meio de uma implementação desenvolvida para a resolução de problemas de otimização topológica de estruturas tridimensionais, apresentados no Capítulo 1. Analisamos o desempenho e a eficiência do *multigrid* (descrito no Capítulo 3) como precondicionador do método dos gradientes conjugados para a resolução dos sistemas lineares de equilíbrio e, posteriormente, testamos a técnica de multirresolução, explicada no Capítulo 5.

Para a resolução dos problemas de otimização, utilizamos um algoritmo de programação linear sequencial (PLS), conforme descrito no Capítulo 2. Aplicamos o método dos elementos finitos, utilizando inicialmente elementos prismáticos retangulares de grau 1. Com o objetivo de evitar os artefatos que surgem na multirresolução, testamos a utilização de elementos com grau maior e estratégias adaptativas para suprimir variáveis, propostas no Capítulo 6, obtendo estruturas mais detalhadas sem prejudicar demais a eficiência conquistada com a multirresolução. Além disso, testamos a estratégia de arredondamento das densidades, para obter soluções com poucas densidades intermediárias.

A implementação foi feita em MATLAB©. Os resultados da Seção 7.3 foram obtidos em um computador com processador Intel® Core™ i7 - 8700U (3.2GHz), 32GB de memória RAM, sob o sistema operacional Ubuntu 18.04.1 LTS, utilizando a versão R2019a do Matlab. Já os resultados das demais seções foram obtidos em um computador com processador AMD Ryzen™ Threadripper™ 1950X, com 64GB de memória RAM, utilizando a versão R2021b do Matlab.

7.1 Detalhes da implementação

Por simplicidade, consideramos problemas artificiais em que os elementos finitos possuem as três dimensões medindo 1 unidade de comprimento (mm, por exemplo), o módulo de Young do material sólido que compõe as estruturas é $E_0 = 1\ N/mm^2$, o módulo de Young do vazio é $E_{min} = 10^{-6}E_0$, o coeficiente de Poisson é $\nu = 0{,}3$ e a intensidade das forças aplicadas é de $1\ N$ (ou $1\ N/mm^2$ no caso de forças distribuídas). O parâmetro de penalização do modelo SIMP utilizado foi $p = 3$.

Para a resolução dos sistemas lineares de equilíbrio, comparamos a rotina `pcg` do Matlab com precondicionador `ichol` e uma nova rotina de gradientes conjugados que utiliza o método *multigrid* como precondicionador. Na Seção 7.3 apresentamos testes de

parâmetros do *multigrid*.

Os subproblemas de programação linear (PL) foram resolvidos utilizando a rotina interna `linprog` do Matlab, da qual aumentamos o número máximo de iterações para garantir a resolução precisa dos PL, pelo método Dual Simplex. Seguindo a notação dos parâmetros da PLS no Capítulo 2, o raio da região de confiança inicial utilizado é $\delta_0 = 0{,}1$ e o raio mínimo é $\delta_{min} = 10^{-4}$. Os parâmetros da função de mérito são $\theta_0 = 1$ e $\theta_{max} = 1$. Outros parâmetros, como os de aceitação do passo, redução e aumento do raio da região de confiança, têm os valores numéricos que aparecem no Algoritmo 1.

Ademais, escolhemos o vetor das variáveis de projeto inicial como sendo um vetor constante com todas as componentes iguais à fração de volume máximo permitido. Por exemplo, se a estrutura deve ocupar no máximo 40% do volume total do domínio, tomamos um vetor com dimensão igual ao número de elementos na malha e com todas as componentes iguais a 0,4. Com isso, o ponto inicial é factível para o problema.

7.2 Estruturas testadas

Nesta seção, mostramos os exemplos de problemas de otimização topológica estrutural testados. Nos testes, consideramos diferentes discretizações do domínio de cada problema, ou seja, consideramos malhas com diferentes quantidades de elementos. Representamos a viga em balanço pelas iniciais `cb`, a viga MBB por `mbb`, a torre de transmissão por `tt`, o meio cubo por `hc`, o prédio com torção por `bt`, a viga em L por `ls` e a ponte por `bd`, seguido do número de elementos em cada direção x, y e z, nesta ordem. Por exemplo, `cb60x20x20` representa a viga em balanço com o refinamento da malha contendo 60 camadas de elementos na direção x, 20 na direção y e 20 na direção z.

7.2.1 Exemplo 1 - Viga em balanço

Este tipo de estrutura é frequentemente empregado na literatura de otimização topológica. Uma das extremidades da viga é engastada, isto é, fixada em outro corpo rígido (impedindo os deslocamentos em todas as direções), e a extremidade oposta recebe uma carga, motivo pelo qual dizemos que ela está em balanço. A Figura 7.1 ilustra o domínio do problema. Uma força concentrada é aplicada para baixo no ponto médio da aresta inferior direita. A estrutura deve ocupar, no máximo, 20% do volume total do domínio.

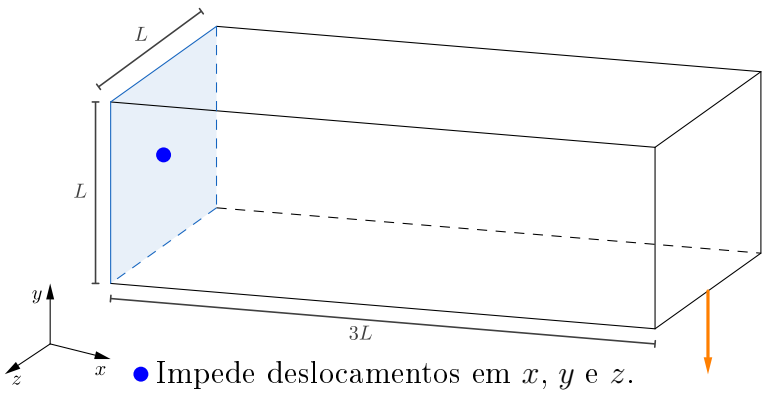
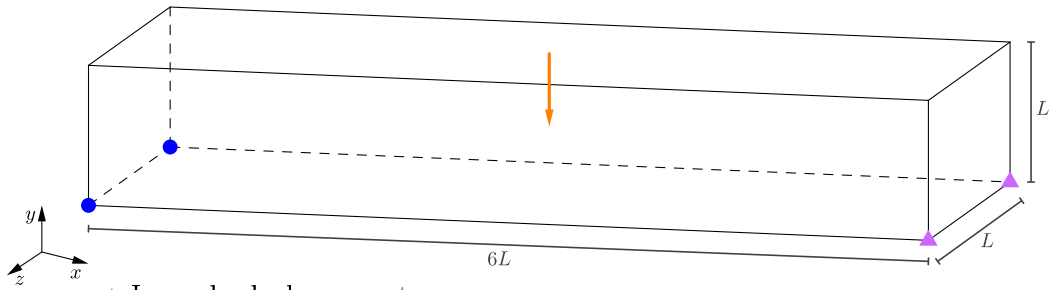


Figura 7.1: Domínio inicial do problema da viga em balanço.

7.2.2 Exemplo 2 - Viga MBB

A viga originalmente produzida pela empresa alemã Messerschmitt-Bolkow-Blohm, mais conhecida como viga MBB, se tornou outro clássico da otimização topológica estrutural. Ela possui apoios nos quatro cantos da parte inferior e uma carga concentrada é aplicada no centro da parte superior. A Figura 7.2 ilustra o domínio do problema. A estrutura deve ocupar, no máximo, 20% do volume total do domínio. Neste e em outros exemplos, consideramos que os apoios estão aplicados em todos os nós da face inferior dos elementos nos cantos inferiores da malha.



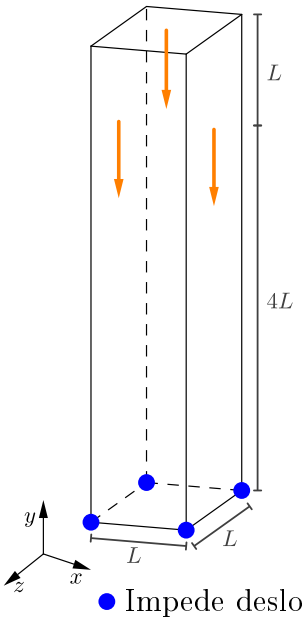
▲ Impede deslocamentos em y.

● Impede deslocamentos em x, y e z.

Figura 7.2: Domínio inicial do problema da viga MBB.

7.2.3 Exemplo 3 - Torre de transmissão

Neste exemplo, tentamos reproduzir uma torre de transmissão de energia elétrica. Para isso, consideramos o domínio da Figura 7.3.



● Impede deslocamentos em x, y e z.

Figura 7.3: Domínio inicial do problema da torre de transmissão.

Há apoios nos quatro cantos inferiores. Uma carga concentrada é aplicada para baixo no centro da face superior e outras duas cargas são aplicadas para baixo em duas

faces laterais opostas, na largura média e a 4/5 da altura do domínio (onde estariam os braços da torre, que sustentam os cabos). A estrutura deve ocupar, no máximo, 15% do volume total do domínio.

7.2.4 Exemplo 4 - Meio cubo com carga concentrada

A Figura 7.4 ilustra o domínio deste exemplo, que consiste em metade de um cubo, com apoios nos quatro cantos inferiores. Uma carga concentrada é aplicada para baixo no centro da face inferior. A estrutura deve ocupar, no máximo, 16% do volume.

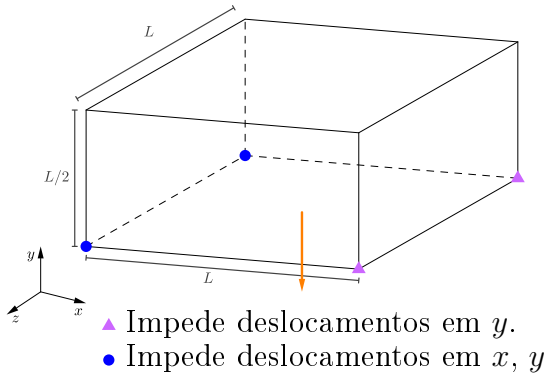


Figura 7.4: Domínio inicial do problema do meio cubo.

7.2.5 Exemplo 5 - Prédio com torção

Este exemplo consiste em um prédio sujeito a uma força de torção. A Figura 7.5 ilustra o domínio do problema. A face inferior é engastada, isto é, possui um apoio que impede os deslocamentos nas três direções. Para simular a torção, consideramos cargas concentradas aplicadas nos pontos médios das arestas da face superior, com direções e sentidos de acordo com a Figura 7.5. A estrutura deve ocupar, no máximo, 10% do volume total do domínio.

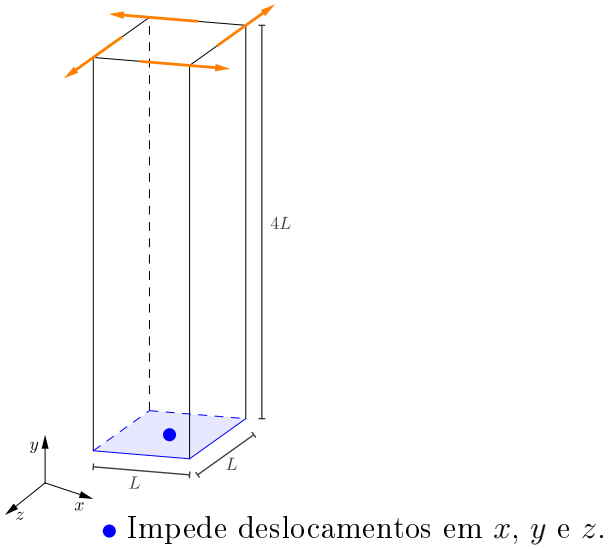


Figura 7.5: Domínio inicial do problema do prédio com torção.

7.2.6 Exemplo 6 - Viga em L

O domínio deste exemplo é constituído pela união de dois prismas retangulares retos, que se juntam no formato da letra "L", conforme ilustra a Figura 7.6. Podemos considerar o domínio como sendo um único prisma retangular que possui uma região vazia, fixando as densidades dessa região em zero. Uma carga concentrada é aplicada para baixo no centro da extremidade lateral direita. A estrutura é engastada na face superior do domínio e deve ocupar, no máximo, 18% do volume total.

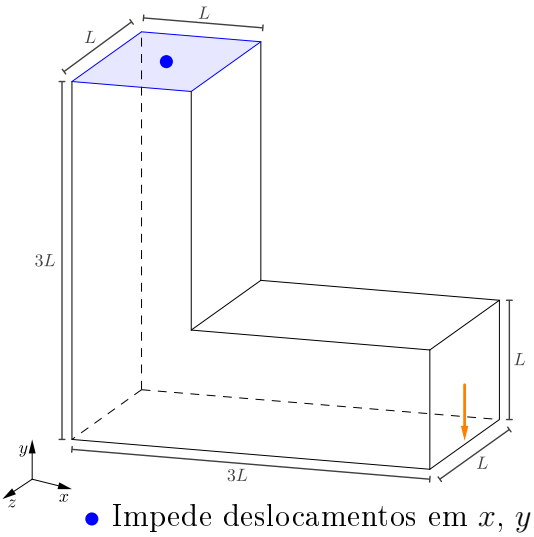


Figura 7.6: Domínio inicial do problema da viga em L.

7.2.7 Exemplo 7 - Ponte

A Figura 7.7 ilustra o domínio do problema da ponte. Os apoios estão na face inferior a uma distância L dos vértices. Consideramos que há uma camada que deve conter material, ou seja, que contém elementos de densidades fixas em 1, na parte central do domínio, paralela ao plano xz (a passarela da ponte). Uma carga distribuída é aplicada para baixo por toda parte superior dessa camada. A estrutura deve ocupar, no máximo, 15% do volume total do domínio.

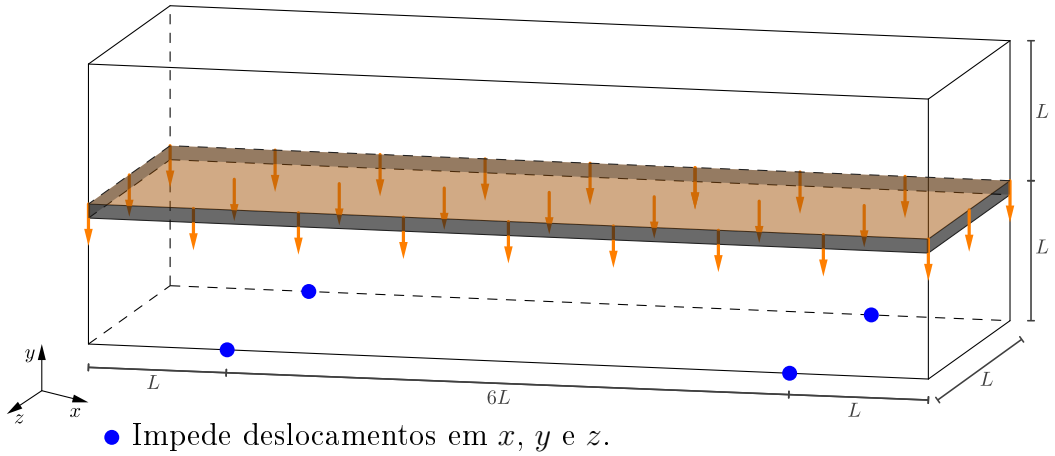


Figura 7.7: Domínio inicial do problema da ponte.

7.3 Testes de parâmetros do *multigrid*

Nesta seção, apresentamos os resultados dos experimentos realizados com uma implementação do *multigrid* como precondicionador do método dos gradientes conjugados para a resolução de sistemas lineares provenientes das discretizações dos problemas de otimização topológica estrutural tridimensionais. Analisamos o desempenho do *multigrid* em comparação com a fatoração incompleta de Cholesky, que é comumente utilizada como precondicionador na resolução dos sistemas lineares de grande porte.

7.3.1 *Multigrid* geométrico

Para os testes do *multigrid* geométrico, trabalhamos com três estruturas distintas (Exemplos 1, 2 e 3 da Seção 7.2) e dois refinamentos para cada uma: `cb48x16x16`, `cb96x32x32`, `mbb96x16x16`, `mbb192x32x32`, `tt16x80x16` e `tt32x160x32`. Para algumas análises da influência da dimensão do problema no tempo de solução, também consideramos outros dois refinamentos da viga em balanço: `cb144x48x48` e `cb192x64x64`. Guardamos as informações de um sistema linear, Ku = f, obtido na 5^a iteração externa da PLS, para cada problema e analisamos a resolução desse sistema com diferentes parâmetros do *multigrid* geométrico.

Testamos o método *multigrid* alterando a quantidade de malhas, o tipo de ciclo, o suavizador e as quantidades de iterações da pré e da pós-suavização (η_1 e η_2). Consideramos como padrão o método com 4 níveis de malhas, ciclo V, suavizador Jacobi com relaxação $\omega = 0{,}5$ e $\eta_1 = \eta_2 = 1$. Nas tabelas, utilizamos a seguinte legenda:

- N_{mg}: número de iterações para convergência do método dos gradientes conjugados com *multigrid* geométrico como precondicionador.
- N_{ic}: número de iterações para convergência do método dos gradientes conjugados com Cholesky incompleto como precondicionador.
- TS_{mg}: tempo (em segundos) para a fase de *setup* do *multigrid* geométrico.
- TS_{ic}: tempo (em segundos) para o cálculo do fator de Cholesky incompleto.
- T_{mg}: tempo (em segundos) para resolver o sistema linear utilizando o método dos gradientes conjugados com *multigrid* geométrico como precondicionador.
- T_{ic}: tempo (em segundos) para resolver o sistema linear utilizando o método dos gradientes conjugados com Cholesky incompleto como precondicionador.

Consideramos como critério de parada para convergência do método dos gradientes conjugados a norma relativa do resíduo menor do que 10^{-8}, para ambos os precondicionadores. A fase de *setup* do *multigrid* geométrico envolve a preparação do suavizador, a construção das malhas, das matrizes de prolongação e das matrizes do sistema em cada malha, além da fatoração de Cholesky da matriz na malha mais grossa.

Inicialmente, resolvemos os sistemas lineares utilizando os parâmetros padrão do *multigrid* geométrico, conforme descritos anteriormente: 4 malhas, ciclo V, suavizador Jacobi com $\omega = 0{,}5$ e $\eta_1 = \eta_2 = 1$. Comparamos os resultados com a resolução utilizando o

precondicionador Cholesky incompleto, obtendo os valores da Tabela 7.1. Note que tanto o número de iterações quanto os tempos de *setup* e de solução foram menores para o *multigrid* geométrico em todos os sistemas testados, o que sugere que ele é um excelente precondicionador. A redução no tempo de solução foi bastante significativa, principalmente nos problemas maiores, chegando a um tempo de solução aproximadamente 32 vezes menor no caso do problema `mbb192x32x32`.

Tabela 7.1: Resultados com os parâmetros padrão do *multigrid* geométrico.

Sistema	N_{mg}	N_{ic}	TS_{mg}	TS_{ic}	T_{mg}	T_{ic}	T_{ic}/T_{mg}
`cb48x16x16`	31	294	0,07	0,09	0,44	2,46	5,59
`cb96x32x32`	24	597	0,57	0,77	2,58	35,54	13,78
`mbb96x16x16`	30	582	0,14	0,19	0,86	9,94	11,56
`mbb192x32x32`	21	1217	1,29	1,56	4,11	131,69	32,04
`tt16x80x16`	33	653	0,12	0,16	0,81	9,14	11,28
`tt32x160x32`	27	1388	1,00	1,28	4,94	138,59	28,05

Observando os dois refinamentos de um mesmo tipo de estrutura, notamos que, no caso do precondicionador Cholesky incompleto, o número de iterações necessárias para convergência do método dos gradientes conjugados cresce conforme o número de variáveis aumenta, fazendo com que o tempo de solução também fique cada vez maior e dificultando a utilização de malhas mais finas. Por outro lado, isso não parece acontecer com o *multigrid* geométrico, em que o número de iterações foi até menor para os problemas com mais variáveis.

Para reforçar esse resultado, resolvemos outros dois sistemas da viga em balanço com refinamentos maiores e analisamos como a dimensão do sistema influencia no tempo de solução, obtendo a Tabela 7.2 e o gráfico da Figura 7.8. Os pontos para o *multigrid* geométrico foram bem aproximados por uma curva de tendência linear (com coeficiente de determinação R^2 aproximadamente igual a 0,9964) e, apesar de haver crescimento do tempo com a dimensão, ele é mais controlado. Já no caso do Cholesky incompleto, os pontos são bem aproximados por uma curva quadrática (com R^2 aproximadamente igual a 0,9999) e vemos que o tempo de solução cresce bastante com a dimensão dos sistemas.

Tabela 7.2: Resultados para sistemas da viga em balanço com diferentes refinamentos.

Sistema	**Dim.**	N_{mg}	N_{ic}	T_{mg}	T_{ic}
`cb48x16x16`	41616	31	294	0,44	2,46
`cb96x32x32`	313632	24	597	2,58	35,54
`cb144x48x48`	1037232	22	879	6,57	142,55
`cb192x64x64`	2433600	20	1178	13,66	467,42

Em seguida, analisamos o desempenho do *multigrid* alterando alguns parâmetros e mantendo os demais na forma padrão. Na Tabela 7.3, mostramos os resultados obtidos variando a quantidade de malhas entre 3, 4 e 5. Note que, para todos os sistemas lineares, o número de iterações e o tempo de solução foram menores ao utilizarmos 3 níveis de malhas e maiores ao utilizarmos 5 níveis. Ou seja, a utilização de 3 malhas foi suficiente para um bom desempenho do método, mas usar 5 malhas foi exagero.

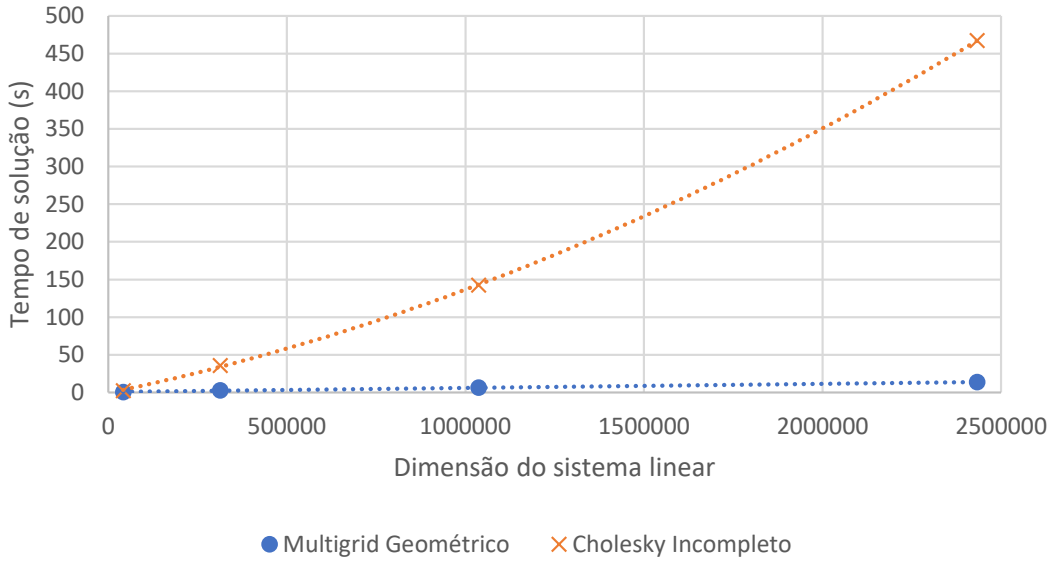


Figura 7.8: Tempo de solução em função da dimensão do sistema linear para problemas da viga em balanço, com os precondicionadores *multigrid* geométrico e Cholesky incompleto.

Tabela 7.3: Resultados variando a quantidade de malhas do *multigrid* geométrico.

Sistema	**Malhas**	N_{mg}	TS_{mg}	T_{mg}
`cb48x16x16`	3	25	0,07	**0,37**
	4	31	0,07	0,44
	5	33	**0,07**	0,47
`cb96x32x32`	3	19	0,64	**2,16**
	4	24	**0,57**	2,58
	5	30	0,58	3,26
`mbb96x16x16`	3	23	0,15	**0,69**
	4	30	0,14	0,86
	5	32	**0,14**	0,93
`mbb192x32x32`	3	17	1,73	**3,70**
	4	21	1,29	4,11
	5	28	**1,27**	5,51
`tt16x80x16`	3	23	0,13	**0,59**
	4	33	0,12	0,81
	5	41	**0,11**	1,02
`tt32x160x32`	3	23	1,34	**4,62**
	4	27	1,00	4,94
	5	35	**1,00**	6,44

Acreditamos que essas conclusões possam ser diferentes se resolvermos sistemas maiores do que os considerados. Para sistemas com dimensões maiores, possivelmente a utilização de mais do que 3 níveis de malhas seja mais eficaz. Para verificar isto, resolvemos outros dois sistemas da viga em balanço com refinamentos maiores e analisamos o

crescimento do tempo de solução em função da dimensão do sistema utilizando 3, 4 e 5 níveis de malhas, obtendo os resultados da Tabela 7.4 e plotando os gráficos da Figura 7.9. Observe que, para as três quantidades de malhas, o crescimento do tempo de solução em função da dimensão é bem aproximado por uma curva de tendência linear (com valores R^2 aproximadamente iguais a 0,9995; 0,9964 e 0,9969 para 3, 4 e 5 níveis, respectivamente) e que, a partir de uma certa dimensão, a curva para 4 malhas começa a ficar abaixo da curva para 3 malhas, confirmando que a utilização de 4 malhas se torna mais eficaz.

Tabela 7.4: Resultados variando a quantidade de malhas do *multigrid* geométrico para sistemas da viga em balanço com diferentes refinamentos.

Sistema	**Dim.**	**Malhas**	N_{mg}	TS_{mg}	T_{mg}
`cb48x16x16`	41616	3	25	0,07	**0,37**
		4	31	0,07	0,44
		5	33	**0,07**	0,47
`cb96x32x32`	313632	3	19	0,64	**2,16**
		4	24	**0,57**	2,58
		5	30	0,58	3,26
`cb144x48x48`	1037232	3	18	3,00	**6,05**
		4	22	2,23	6,57
		5	25	**2,21**	7,46
`cb192x64x64`	2433600	3	17	9,06	14,39
		4	20	5,51	**13,66**
		5	25	**5,40**	17,07

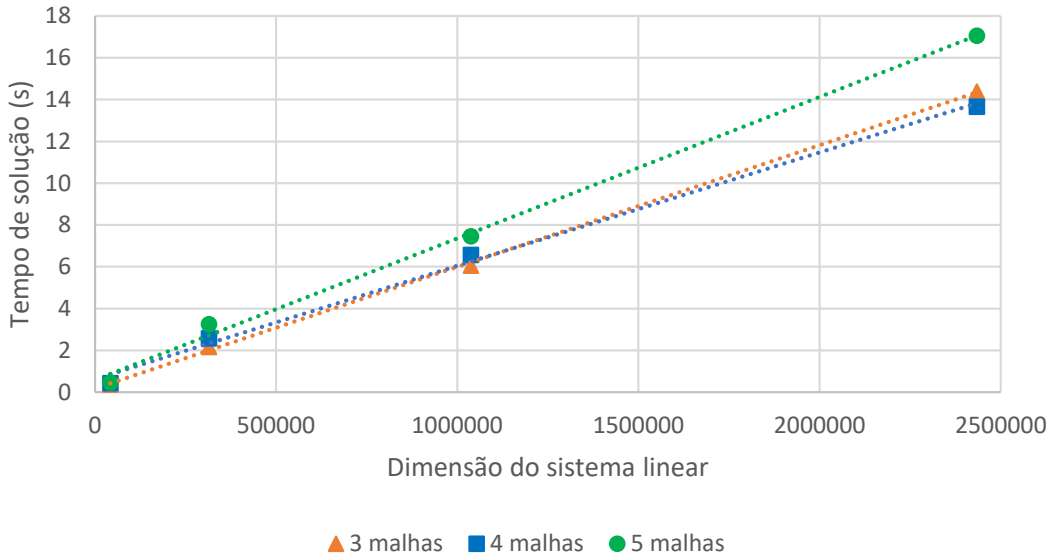


Figura 7.9: Tempo de solução em função da dimensão do sistema linear para problemas da viga em balanço, com diferentes quantidades de malhas do *multigrid* geométrico.

Já o tempo de *setup*, em geral, foi maior com 3 níveis de malhas, devido ao cálculo do fator de Cholesky na malha mais grossa, que é feito para uma matriz com dimensão maior quando menos níveis são utilizados. Em sistemas muito grandes, esse

tempo de *setup* pode ficar significativo e indicar que é necessário utilizar mais malhas para melhorar o desempenho.

Na Tabela 7.5, apresentamos os resultados obtidos variando o tipo de ciclo do *multigrid*, considerando o ciclo V e o ciclo W. Para todos os sistemas resolvidos, o ciclo W foi mais eficiente, reduzindo o número de iterações e, consequentemente, também o tempo de solução. O tipo de ciclo não influencia em nada a etapa de *setup*, motivo pelo qual não mostramos os tempos dessa etapa na tabela.

Tabela 7.5: Resultados variando o tipo de ciclo do *multigrid* geométrico.

Sistema	**Ciclo**	N_{mg}	T_{mg}
`cb48x16x16`	V	31	0,44
	W	23	**0,35**
`cb96x32x32`	V	24	2,58
	W	17	**2,05**
`mbb96x16x16`	V	30	0,86
	W	21	**0,73**
`mbb192x32x32`	V	21	4,11
	W	16	**3,65**
`tt16x80x16`	V	33	0,81
	W	24	**0,66**
`tt32x160x32`	V	27	4,94
	W	19	**3,99**

Agora, analisamos alguns métodos estacionários que são frequentemente utilizados como suavizadores no *multigrid*: Jacobi (J) com relaxação, Gauss-Seidel (GS) ou *Successive Over Relaxation* (SOR) e *Symmetric Successive Over Relaxation* (SSOR).

Testamos a resolução dos sistemas para os três tipos de estruturas, variando o parâmetro de relaxação ω para cada um dos suavizadores. Neste caso, consideramos um número máximo de 200 iterações para convergência do método dos gradientes conjugados, uma vez que, dependendo do valor de ω, o método estacionário pode não ser um bom suavizador e prejudicar a convergência. Como todos os sistemas testados com os parâmetros padrão foram resolvidos com menos de 50 iterações de gradientes conjugados, consideramos que um número de iterações maior do que 200 já é indício suficiente de que o método convergiu muito lentamente. Na Tabela 7.6, apresentamos os resultados para o método de Jacobi. Note que a convergência é lenta para $\omega > 0{,}8$, ultrapassando as 200 iterações. O valor $\omega = 2/3$, que geralmente é o melhor para Jacobi, não foi tão bom quando utilizamos o método como suavizador neste tipo de problema, já que os tempos de solução foram menores para outros valores de ω. É importante ressaltar que o melhor valor de ω depende de cada sistema que está sendo resolvido. Durante a resolução do problema de otimização topológica, nos deparamos com vários sistemas lineares diferentes, e aqui estamos testando somente um desses sistemas para cada estrutura. Apesar dos resultados indicarem que $\omega = 0{,}6$ foi melhor, alguns testes mostraram que esse valor pode não ser tão bom em certas iterações da otimização topológica, sendo necessário utilizar um valor menor. Assim, por segurança, decidimos manter $\omega = 0{,}5$ para o método de Jacobi, que

apresentou bons resultados e funcionou bem para todos os problemas considerados.

Tabela 7.6: Resultados variando o parâmetro ω do suavizador Jacobi.

Sistema	ω	N_{mg}	T_{mg}
cb96x32x32	0,1	45	4,96
	0,3	30	3,29
	0,5	24	2,58
	0,6	23	**2,50**
	2/3	42	4,62
	0,8	>200	>22,41
	1	>200	>22,44
mbb192x32x32	0,1	39	7,74
	0,3	26	5,11
	0,5	21	4,11
	0,6	20	**3,90**
	2/3	42	8,38
	0,8	>200	>40,64
	1	>200	>40,72
tt32x160x32	0,1	47	8,74
	0,3	31	5,71
	0,5	27	4,94
	0,6	26	**4,77**
	2/3	48	8,96
	0,8	>200	>37,96
	1	>200	>37,97

Os resultados variando o parâmetro ω do método SOR estão na Tabela 7.7. Observe que o método dos gradientes conjugados só conseguiu convergir antes das 200 iterações para valores de ω muito pequenos, abaixo de 0,2, e mesmo assim o tempo de solução não foi melhor do que com o suavizador Jacobi. Como a matriz de rigidez é simétrica, acreditamos que esse inconveniente tenha ocorrido devido à perda da simetria na matriz de iteração do método SOR. Por esse motivo, nos problemas de otimização topológica, é melhor utilizarmos o método SSOR, com o qual obtivemos os resultados da Tabela 7.8. Neste caso, a convergência ocorreu em poucas iterações para todos os valores de ω testados. Além disso, notamos que o tempo de solução fica pior conforme ω se aproxima dos valores extremos, 0 ou 2, e os melhores resultados aconteceram para ω no intervalo [0,8; 1,2]. Novamente, é difícil dizer qual é o melhor valor de ω para todos os sistemas lineares que aparecem no decorrer da resolução de cada problema de otimização topológica. É possível elaborar estratégias para adaptar ω durante o processo iterativo, mas neste trabalho decidimos utilizar valores fixos para o parâmetro.

Para o método de Jacobi, vimos que é necessária a utilização do parâmetro de relaxação e utilizamos $\omega = 0{,}5$, que funcionou bem para todos os sistemas considerados. Já para o método SSOR decidimos utilizar $\omega = 1{,}2$. Na Tabela 7.9, apresentamos os

resultados obtidos variando o suavizador e as quantidades de iterações na pré e na pós-suavização (η_1 e η_2).

Tabela 7.7: Resultados variando o parâmetro ω do suavizador SOR.

Sistema	ω	N_{mg}	T_{mg}
`cb96x32x32`	0,2	41	**5,81**
	0,4	>200	>28,84
	0,6	>200	>28,93
	0,8	>200	>28,88
	1	>200	>28,94
	1,2	>200	>28,85
	1,4	>200	>28,97
	1,6	>200	>28,89
	1,8	>200	>28,96
`mbb192x32x32`	0,2	38	**9,76**
	0,4	>200	>52,60
	0,6	>200	>52,64
	0,8	>200	>52,57
	1	>200	>52,58
	1,2	>200	>52,79
	1,4	>200	>52,60
	1,6	>200	>52,63
	1,8	>200	>52,51
`tt32x160x32`	0,2	60	**14,29**
	0,4	>200	>48,32
	0,6	>200	>48,10
	0,8	>200	>48,21
	1	>200	>48,21
	1,2	>200	>48,37
	1,4	>200	>48,36
	1,6	>200	>48,56
	1,8	>200	>48,35

Não reportamos os resultados para o método SOR porque, conforme vimos, ele não funcionou bem como suavizador e o método dos gradientes conjugados convergiu de forma bastante lenta, atingindo a quantidade máxima de iterações, sendo necessário aplicarmos um parâmetro de relaxação ω muito pequeno para melhorar a convergência. Contudo, notamos que o método funciona melhor para valores maiores de η_1 e η_2, embora, mesmo assim, não consiga obter resultados tão bons quanto os outros suavizadores.

Para os suavizadores considerados na Tabela 7.9, notamos que o tempo de solução do sistema foi menor utilizando $\eta_1 = \eta_2 = 1$, apesar de ter sido necessário realizar mais iterações para convergência. Ou seja, é suficiente aplicarmos apenas uma iteração na

pré e na pós-suavização. Acreditamos que, quanto maiores os valores de η_1 e η_2, menos eficaz será o método e, por esse motivo, não consideramos quantidades maiores do que 2.

Tabela 7.8: Resultados variando o parâmetro ω do suavizador SSOR.

Sistema	ω	N_{mg}	T_{mg}
`cb96x32x32`	0,2	27	4,88
	0,4	21	3,76
	0,6	18	3,21
	0,8	18	3,19
	1	17	3,07
	1,2	17	**3,01**
	1,4	17	3,02
	1,6	18	3,21
	1,8	21	3,77
`mbb192x32x32`	0,2	24	7,89
	0,4	19	6,21
	0,6	16	5,17
	0,8	15	4,82
	1	15	4,92
	1,2	15	**4,82**
	1,4	15	4,82
	1,6	17	5,52
	1,8	22	7,23
`tt32x160x32`	0,2	35	10,74
	0,4	29	8,77
	0,6	27	8,17
	0,8	25	**7,56**
	1	25	7,69
	1,2	25	7,58
	1,4	26	7,87
	1,6	28	8,52
	1,8	35	10,71

Comparando os suavizadores, observamos que o método de Jacobi apresentou melhores resultados com relação ao tempo de solução dos sistemas lineares. Além disso, o tempo de *setup* é maior quanto mais complexas são as matrizes que precisam ser extraídas para a aplicação do suavizador. Assim, o *setup* fica mais barato com o método de Jacobi e mais caro com o SSOR. Portanto, o método de Jacobi com $\eta_1 = \eta_2 = 1$ foi o suavizador mais eficiente para os sistemas testados.

Para finalizar, testamos algumas combinações dos parâmetros do *multigrid* geométrico que apresentaram melhores resultados. Conforme vimos nas Tabelas 7.3 e 7.5, a utilização de 3 malhas foi melhor do que 4 malhas e o ciclo W foi mais eficiente do que

o ciclo V para os sistemas considerados. Mas será que a combinação desses parâmetros é melhor ainda? Os resultados na Tabela 7.10 sugerem que isso não é sempre verdade.

Tabela 7.9: Resultados variando o suavizador do *multigrid* geométrico.

Sistema	**Suavizador** (η_1, η_2)	N_{mg}	TS_{mg}	T_{mg}
`cb48x16x16`	J (1, 1)	31	**0,07**	**0,44**
	J (2, 2)	26	0,07	0,56
	SSOR (1, 1)	21	0,10	0,51
	SSOR (2, 2)	18	0,10	0,72
`cb96x32x32`	J (1, 1)	24	**0,57**	**2,58**
	J (2, 2)	20	0,58	3,25
	SSOR (1, 1)	17	1,00	3,01
	SSOR (2, 2)	15	1,03	4,50
`mbb96x16x16`	J (1, 1)	30	**0,14**	**0,86**
	J (2, 2)	23	0,14	0,99
	SSOR (1, 1)	20	0,20	0,97
	SSOR (2, 2)	18	0,20	1,49
`mbb192x32x32`	J (1, 1)	21	**1,29**	**4,11**
	J (2, 2)	17	1,30	4,98
	SSOR (1, 1)	15	2,06	4,82
	SSOR (2, 2)	13	2,07	7,11
`tt16x80x16`	J (1, 1)	33	**0,12**	**0,81**
	J (2, 2)	26	0,12	0,94
	SSOR (1, 1)	34	0,17	1,46
	SSOR (2, 2)	30	0,17	2,11
`tt32x160x32`	J (1, 1)	27	**1,00**	**4,94**
	J (2, 2)	23	1,02	6,38
	SSOR (1, 1)	25	1,74	7,58
	SSOR (2, 2)	22	1,75	11,42

Utilizar menos níveis de malhas e o ciclo W parece reduzir o número de iterações, porém, com relação ao tempo de solução, nos sistemas maiores foi um pouco mais eficaz utilizar 4 malhas e o ciclo W do que 3 malhas e o ciclo W. Novamente, ressaltamos que isso depende das dimensões dos problemas considerados. Para sistemas muito grandes, talvez seja melhor a utilização de mais níveis de malhas. Além disso, existem outros tipos de ciclos do *multigrid* que podem ser analisados em trabalhos futuros.

De modo geral, para os problemas que estamos resolvendo, consideramos que o melhor para o método *multigrid* geométrico é utilizar o ciclo W, suavizador Jacobi com relaxação $\omega = 0{,}5$ e $\eta_1 = \eta_2 = 1$. Porém, a quantidade de malhas pode variar conforme a dimensão do problema.

Além disso, estamos utilizando a fatoração de Cholesky para resolver o sistema na malha mais grossa, o que pode ficar caro dependendo da dimensão e da quantidade

de malhas utilizada. Neste caso, uma opção é utilizar um método iterativo (gradientes conjugados, por exemplo) para resolver o sistema na última malha. Outra estratégia é aumentar a quantidade de malhas com base em alguma dimensão máxima para a qual seja possível utilizar Cholesky. Assim, comparamos as seguintes estratégias:

- **Estratégia 1:** utilizar 3 malhas e Cholesky para resolver o sistema linear na malha mais grossa.
- **Estratégia 2:** utilizar 3 malhas e gradientes conjugados com precondicionador Cholesky incompleto para resolver o sistema na malha mais grossa.
- **Estratégia 3:** começar com 3 malhas, mas aumentar esse número caso a dimensão do sistema na última malha seja maior que 7500.
- **Estratégia 4:** utilizar o máximo de malhas possível e Cholesky para resolver o sistema na malha mais grossa.

A dimensão 7500, utilizada na Estratégia 3, foi obtida experimentalmente.

Tabela 7.10: Resultados variando parâmetros combinados do *multigrid* geométrico.

Sistema	**Parâmetros**	N_{mg}	T_{mg}
`cb48x16x16`	4 malhas e ciclo V	31	0,44
	3 malhas e ciclo V	25	0,37
	4 malhas e ciclo W	23	0,35
	3 malhas e ciclo W	21	**0,34**
`cb96x32x32`	4 malhas e ciclo V	24	2,58
	3 malhas e ciclo V	19	2,16
	4 malhas e ciclo W	17	**2,05**
	3 malhas e ciclo W	17	2,20
`mbb96x16x16`	4 malhas e ciclo V	30	0,86
	3 malhas e ciclo V	23	**0,69**
	4 malhas e ciclo W	21	0,73
	3 malhas e ciclo W	20	0,72
`mbb192x32x32`	4 malhas e ciclo V	21	4,11
	3 malhas e ciclo V	17	3,70
	4 malhas e ciclo W	16	**3,65**
	3 malhas e ciclo W	16	4,24
`tt16x80x16`	4 malhas e ciclo V	33	0,81
	3 malhas e ciclo V	23	**0,59**
	4 malhas e ciclo W	24	0,66
	3 malhas e ciclo W	21	**0,59**
`tt32x160x32`	4 malhas e ciclo V	27	4,94
	3 malhas e ciclo V	23	4,62
	4 malhas e ciclo W	19	**3,99**
	3 malhas e ciclo W	19	4,48

Pensando na resolução de problemas maiores, testamos essas estratégias para os sistemas da viga em balanço com diferentes refinamentos, obtendo os resultados da Tabela 7.11. Na Estratégia 3, para os problemas `cb144x48x48` e `cb192x64x64`, o número de malhas foi aumentado para 4. Na Estratégia 4, para os problemas `cb48x16x16` e `cb144x48x48`, é possível utilizar no máximo 5 malhas, para o problema `cb96x32x32` é possível utilizar 6 malhas e para `cb192x64x64` é possível utilizar 7 malhas. As Estratégias 3 e 4 tiveram melhor desempenho, pois a resolução do sistema na malha mais grossa é mais eficiente com a fatoração de Cholesky quando a dimensão dessa malha é pequena.

Tabela 7.11: Resultados utilizando diversas estratégias do *multigrid* geométrico para sistemas da viga em balanço com diferentes refinamentos.

Sistema	**Dim.**	**Estratégia**	N_{mg}	TS_{mg}	T_{mg}
`cb48x16x16`	41616	1	21	0,08	0,34
		2	21	0,08	0,57
		3	21	**0,08**	**0,31**
		4	23	0,08	0,38
`cb96x32x32`	313632	1	17	0,64	2,20
		2	17	0,59	5,95
		3	17	0,62	2,09
		4	17	**0,59**	**2,06**
`cb144x48x48`	1037232	1	16	2,63	5,73
		2	16	2,29	26,73
		3	16	**2,26**	**5,37**
		4	16	2,27	5,39
`cb192x64x64`	2433600	1	16	6,91	14,19
		2	16	5,51	82,03
		3	16	5,45	12,42
		4	16	**5,42**	**12,40**

Independentemente dos parâmetros utilizados, vimos que o *multigrid* geométrico é um ótimo precondicionador para o método dos gradientes conjugados na resolução dos sistemas lineares provenientes da otimização topológica, permitindo a resolução desses sistemas em tempos muito menores do que com o precondicionador Cholesky incompleto.

7.3.2 *Multigrid* algébrico

Apresentamos agora os resultados dos experimentos realizados com uma implementação do *multigrid* algébrico adaptativo como precondicionador do método dos gradientes conjugados, para a resolução de sistemas lineares da otimização topológica. Trabalhamos com três estruturas distintas e dois refinamentos para cada uma: `cb60x20x20`, `cb90x30x30`, `mbb120x20x20`, `mbb180x30x30`, `tt20x100x20` e `tt30x150x30`. Para algumas análises sobre a influência da dimensão do problema no tempo de solução, também consideramos outros refinamentos da viga em balanço: `cb120x40x40`, `cb150x50x50`, `cb180x60x60` e `cb210x70x70`. Guardamos as informações de um sistema linear, Ku = f, obtido na 5ª

iteração externa da PLS, para cada problema e analisamos a resolução desse sistema com diferentes parâmetros do *multigrid* algébrico. Também analisamos o desempenho do precondicionador em comparação com a fatoração incompleta de Cholesky.

Além dos parâmetros comuns ao *multigrid* geométrico – quantidade de malhas, tipo de ciclo, suavizador e quantidades de iterações na pré e na pós-suavização (η_1 e η_2) –, também precisamos analisar parâmetros presentes nos algoritmos da fase de *setup* do *multigrid* algébrico: o parâmetro de conexões fortes (θ_s), a quantidade de vetores na base do espaço de teste (n_{tv}), as tolerâncias para geração do espaço de teste (ε_r e ε_q), o número máximo de iterações na prolongação DPLS (it_p) e a tolerância para o número de condição na prolongação DPLS (κ). Consideramos como padrão o método com 4 níveis de malhas, ciclo V, suavizador Jacobi com relaxação $\omega = 0{,}5$, $\eta_1 = \eta_2 = 1$ e os parâmetros $\theta_s = 0{,}5$, $n_{tv} = 10$, $\varepsilon_r = 10^{-2}$, $\varepsilon_q = 10^{-4}$, $it_p = 5$ e $\kappa = 50$. Nas tabelas, utilizamos a legenda:

- N_{amg}: número de iterações para convergência do método dos gradientes conjugados com *multigrid* algébrico como precondicionador.
- N_{ic}: número de iterações para convergência do método dos gradientes conjugados com Cholesky incompleto como precondicionador.
- T_{ts}: tempo (em segundos) para a geração do espaço de teste.
- T_{sc}: tempo (em segundos) para o cálculo das forças de conexões.
- T_c: tempo (em segundos) para a construção das malhas grossas (*coarsening*).
- T_p: tempo (em segundos) para a construção das matrizes de prolongação.
- TS_{amg}: tempo total (em segundos) para a fase de *setup* do *multigrid* algébrico.
- TS_{ic}: tempo (em segundos) para o cálculo do fator de Cholesky incompleto.
- T_{amg}: tempo (em segundos) para resolver o sistema linear utilizando o método dos gradientes conjugados com *multigrid* algébrico como precondicionador.
- T_{ic}: tempo (em segundos) para resolver o sistema linear utilizando o método dos gradientes conjugados com Cholesky incompleto como precondicionador.

Consideramos como critério de parada para convergência do método dos gradientes conjugados a norma relativa do resíduo menor que 10^{-8}. A fase de *setup* do *multigrid* algébrico é mais complicada do que a do *multigrid* geométrico, de modo que analisamos separadamente os tempos de cada uma das etapas mais complexas: geração do espaço de teste, cálculo das forças de conexões, construção das malhas grossas e construção das matrizes de prolongação. Além dessas quatro etapas, o tempo total de *setup* também envolve a preparação do suavizador, a construção das matrizes do sistema em cada malha e a fatoração de Cholesky da matriz do sistema na malha mais grossa.

Resolvendo os sistemas lineares com os parâmetros padrão do *multigrid* algébrico, conforme descritos anteriormente, e comparando os resultados com a resolução utilizando o precondicionador Cholesky incompleto, obtivemos os valores da Tabela 7.12. Os resultados indicam que o *multigrid* algébrico também é um bom precondicionador, pois reduziu o número de iterações e o tempo de solução em comparação com o Cholesky

incompleto. Todavia, essas reduções não foram tão grandes quanto no caso geométrico, chegando-se a um tempo médio de solução apenas 1,74 vezes menor, aproximadamente. Ademais, diferentemente do *multigrid* geométrico, no caso algébrico o número de iterações parece estar crescendo com a dimensão do sistema linear, e o tempo gasto na fase de *setup* é bastante grande, o que exige maior cuidado.

Uma das maiores vantagens do *multigrid* algébrico sobre o geométrico é a sua aplicabilidade. No caso geométrico, é necessário que as quantidades de elementos nas três direções da malha inicial sejam divisíveis por $2^{ngrids-1}$, sendo *ngrids* a quantidade de malhas utilizada, para que seja possível construir as malhas mais grossas dividindo as quantidades de elementos por 2 ao passar de um nível para o outro. Por exemplo, se utilizamos 4 níveis de malhas, é necessário que as quantidades de elementos nas três direções da malha inicial sejam divisíveis por 2^3. Assim, podemos resolver o problema `cb48x16x16`, mas ocorre um erro ao tentarmos resolver o problema `cb60x20x20`. O *multigrid* algébrico trabalha apenas com a matriz do sistema linear, sem se preocupar com a malha, podendo ser aplicado em uma variedade muito maior de problemas.

Tabela 7.12: Resultados com os parâmetros padrão do *multigrid* algébrico.

Sistema	N_{amg}	N_{ic}	TS_{amg}	TS_{ic}	T_{amg}	T_{ic}	T_{ic}/T_{amg}
`cb60x20x20`	96	360	24,40	0,20	2,71	5,66	2,09
`cb90x30x30`	134	551	109,85	0,64	12,25	27,29	2,23
`mbb120x20x20`	256	742	86,21	0,40	14,36	22,46	1,56
`mbb180x30x30`	333	1152	418,86	1,33	60,38	113,27	1,88
`tt20x100x20`	355	888	91,99	0,33	16,55	22,80	1,38
`tt30x150x30`	546	1301	316,08	1,06	83,70	107,45	1,28

Olhando para os tempos de cada etapa do *setup* no *multigrid* algébrico, fornecidos na Tabela 7.13, observamos que a etapa mais cara é a de geração do espaço de teste (T_{ts}), que envolve a resolução de um problema de autovalores generalizado pelo método de minimização do quociente de Rayleigh. Para os sistemas com dimensões menores, o tempo gasto para construir as matrizes de prolongação pelo método DPLS (T_p) aparece em segundo lugar. Já quando aumentamos a dimensão, o tempo de construção das malhas grossas (T_c) passa a ser maior. O tempo de cálculo das forças de conexões (T_{sc}) é menor que os demais, mas também é significativo. Em geral, o tempo total de *setup* (TS_{amg}) cresce bastante com a dimensão do problema.

Tabela 7.13: Tempos de *setup* do *multigrid* algébrico.

Sistema	T_{ts}	T_{sc}	T_c	T_p	TS_{amg}
`cb60x20x20`	18,17	0,80	1,75	3,52	24,40
`cb90x30x30`	88,19	2,65	7,39	11,05	109,85
`mbb120x20x20`	74,30	1,59	3,63	6,38	86,21
`mbb180x30x30`	363,20	5,34	28,58	20,61	418,86
`tt20x100x20`	82,22	1,32	2,73	5,47	91,99
`tt30x150x30`	274,74	4,50	18,49	17,54	316,08

Para analisar melhor como a dimensão do sistema influencia os tempos de solução e de *setup*, resolvemos outros quatro sistemas da viga em balanço com refinamentos maiores, obtendo os resultados da Tabela 7.14.

Tabela 7.14: Resultados para sistemas da viga em balanço com diferentes refinamentos.

Sistema	**Dim.**	N_{amg}	N_{ic}	TS_{amg}	TS_{ic}	T_{amg}	T_{ic}
`cb60x20x20`	79380	96	360	24,40	0,20	2,71	5,66
`cb90x30x30`	259470	134	551	109,85	0,64	12,25	27,29
`cb120x40x40`	605160	133	723	296,66	1,43	22,66	68,35
`cb150x50x50`	1170450	176	918	732,90	2,73	56,29	167,17
`cb180x60x60`	2009340	192	1097	1629,31	5,01	111,38	356,48
`cb210x70x70`	3175830	254	1279	3135,92	9,14	221,46	648,40

Na Figura 7.10, plotamos o tempo de solução em função da dimensão do sistema. Tanto para o *multigrid* algébrico quanto para o Cholesky incompleto, os pontos foram bem aproximados por curvas de tendência quadráticas, com valores do coeficiente de determinação aproximadamente iguais a 0,9996 e 0,9993, respectivamente. Contudo, notamos que o crescimento do tempo de solução com a dimensão do sistema é mais rápido no caso do Cholesky incompleto. Uma maneira de compararmos esses crescimentos é olhando para o coeficiente do termo quadrático em cada curva de ajuste. Para isso, dividimos cada uma das dimensões pela menor delas (79380) e obtivemos gráficos parecidos, mas agora com os coeficientes das curvas de ajuste em ordem de grandeza menor. Nesse caso, a curva para o Cholesky incompleto possui o coeficiente do termo quadrático igual a 0,1659, enquanto para o *multigrid* algébrico esse coeficiente é 0,075.

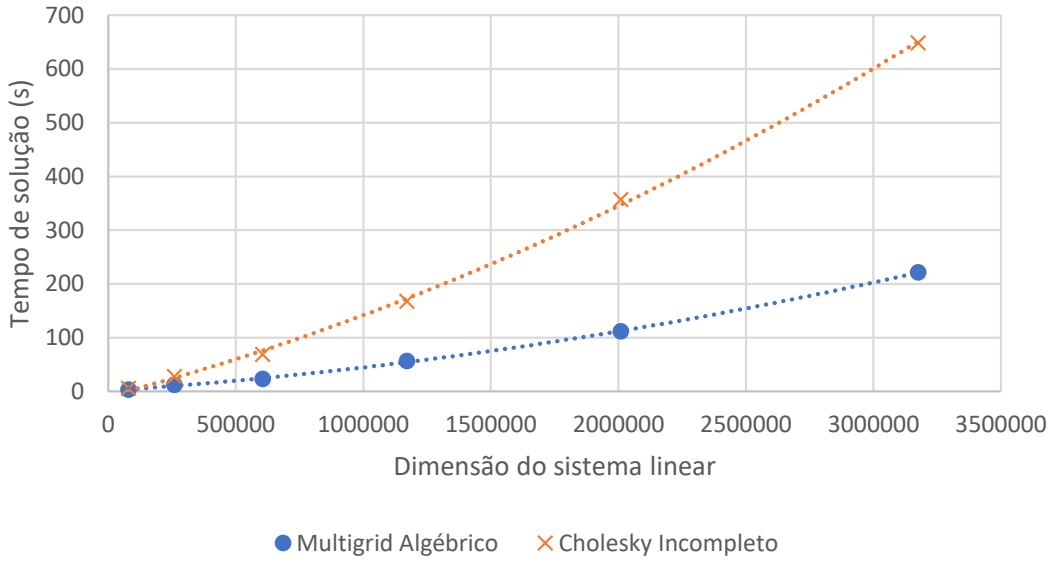


Figura 7.10: Tempo de solução em função da dimensão do sistema linear para problemas da viga em balanço, com os precondicionadores *multigrid* algébrico e Cholesky incompleto.

Na Figura 7.11, plotamos o tempo total de *setup* em função da dimensão do sistema. Neste caso, a situação se inverte e o tempo de preparação para o *multigrid* algébrico cresce mais rapidamente que o do Cholesky incompleto. Os pontos para o *multigrid* foram bem aproximados por uma curva quadrática com coeficiente de determinação 0,9996, já

os pontos para o Cholesky incompleto foram aproximados por uma curva linear com coeficiente de determinação 0,9908. Apesar da fase de preparação do *multigrid* algébrico ser bastante cara, veremos que, na resolução do problema completo de otimização topológica, felizmente conseguimos realizar a fase de *setup* uma única vez, antes da primeira iteração da PLS, e utilizar as componentes obtidas para resolver os sistemas em todas as demais iterações.

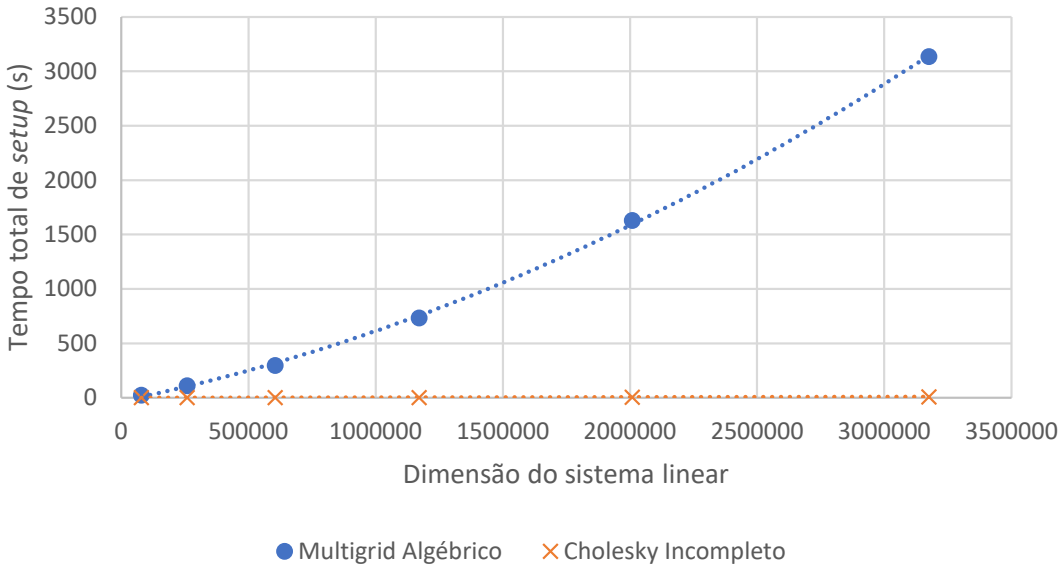


Figura 7.11: Tempo de *setup* em função da dimensão do sistema linear para problemas da viga em balanço, com os precondicionadores *multigrid* algébrico e Cholesky incompleto.

Em seguida, analisamos o desempenho do *multigrid* algébrico alterando alguns parâmetros e mantendo os demais na forma padrão, assim como fizemos para o caso geométrico. Comparamos o número de iterações e o tempo de solução do sistema linear, bem como o tempo gasto em cada etapa de *setup* do *multigrid*.

Na Tabela 7.15, mostramos os resultados obtidos variando a quantidade de malhas entre 3, 4 e 5. Na Tabela 7.16, apresentamos os resultados obtidos comparando os ciclos V e W. Note que os resultados nessas duas tabelas foram compatíveis com aqueles apresentados para o *multigrid* geométrico nas Tabelas 7.3 e 7.5. Ou seja, novamente constatamos que o uso de apenas 3 níveis de malhas foi melhor e que usar 5 malhas foi um exagero para os problemas com as dimensões consideradas. Além disso, o ciclo W apresentou resultados melhores do que o ciclo V, com relação ao número de iterações e ao tempo de solução para todos os sistemas lineares resolvidos. Com relação ao *setup*, a quantidade de malhas não parece alterar significativamente o tempo total e sabemos que o tipo de ciclo não influencia em nada esta etapa.

Estudamos também o crescimento do tempo de solução em função da dimensão do sistema linear, utilizando diferentes quantidades de malhas. Para isso, resolvemos outros quatro sistemas da viga em balanço, com diferentes refinamentos, obtendo a Tabela 7.17 e os gráficos da Figura 7.12.

Para as três quantidades de malhas, o crescimento do tempo de solução em função da dimensão foi bem aproximado por uma curva de tendência quadrática (com coeficientes de determinação aproximadamente iguais a 0,9986; 0,9996 e 0,9998 para 3, 4 e 5 níveis, respectivamente). Para as dimensões consideradas, a curva para 3 malhas está

sempre abaixo da curva para 4 malhas, que, por sua vez, está sempre abaixo da curva para 5 malhas. Nesse caso, não pudemos constatar se é melhor utilizar mais do que 3 níveis de malhas quando a dimensão aumenta muito.

Tabela 7.15: Resultados variando a quantidade de malhas do *multigrid* algébrico.

Sistema	Malhas	N_{amg}	T_{ts}	T_{sc}	T_c	T_p	TS_{amg}	T_{amg}
cb60x20x20	3	72	17,80	0,79	1,74	3,41	**23,90**	**2,04**
	4	96	18,17	0,80	1,75	3,52	24,40	2,71
	5	107	17,84	0,80	1,75	3,44	23,98	2,98
cb90x30x30	3	92	89,95	2,66	7,41	10,88	111,51	**8,88**
	4	134	88,19	2,65	7,39	11,05	**109,85**	12,25
	5	156	89,84	2,69	7,43	11,06	111,57	14,30
mbb120x20x20	3	225	74,03	1,58	3,59	6,33	85,86	**13,10**
	4	256	74,30	1,59	3,63	6,38	86,21	14,36
	5	301	73,84	1,59	3,60	6,34	**85,68**	16,66
mbb180x30x30	3	269	363,59	5,33	28,59	20,33	419,33	**54,10**
	4	333	363,20	5,34	28,58	20,61	**418,86**	60,38
	5	383	367,85	5,38	28,83	20,68	423,87	69,36
tt20x100x20	3	265	79,91	1,31	2,72	5,36	89,56	**12,73**
	4	355	82,22	1,32	2,73	5,47	91,99	16,55
	5	393	79,74	1,33	2,75	5,47	**89,55**	18,40
tt30x150x30	3	479	275,49	4,45	18,25	17,31	316,50	**78,88**
	4	546	274,74	4,50	18,49	17,54	316,08	83,70
	5	585	272,35	4,51	18,40	17,52	**313,60**	89,09

Tabela 7.16: Resultados variando o tipo de ciclo do *multigrid* algébrico.

Sistema	Ciclo	N_{amg}	T_{amg}
cb60x20x20	V	96	2,71
	W	71	**2,19**
cb90x30x30	V	134	12,25
	W	93	**9,45**
mbb120x20x20	V	256	14,36
	W	209	**12,86**
mbb180x30x30	V	333	60,38
	W	263	**53,95**
tt20x100x20	V	355	16,55
	W	259	**13,40**
tt30x150x30	V	546	83,70
	W	467	**79,49**

Tabela 7.17: Resultados variando a quantidade de malhas do *multigrid* algébrico para sistemas da viga em balanço com diferentes refinamentos.

Sistema	Dim.	Malhas	N_{amg}	T_{amg}
cb60x20x20	79380	3	72	**2,04**
		4	96	2,71
		5	107	2,98
cb90x30x30	259470	3	92	**8,88**
		4	134	12,25
		5	156	14,30
cb120x40x40	605160	3	92	**17,43**
		4	133	22,66
		5	174	29,81
cb150x50x50	1170450	3	112	**42,16**
		4	176	56,29
		5	222	70,54
cb180x60x60	2009340	3	135	**98,21**
		4	192	111,38
		5	244	140,71
cb210x70x70	3175830	3	156	**183,58**
		4	254	221,46
		5	318	271,63

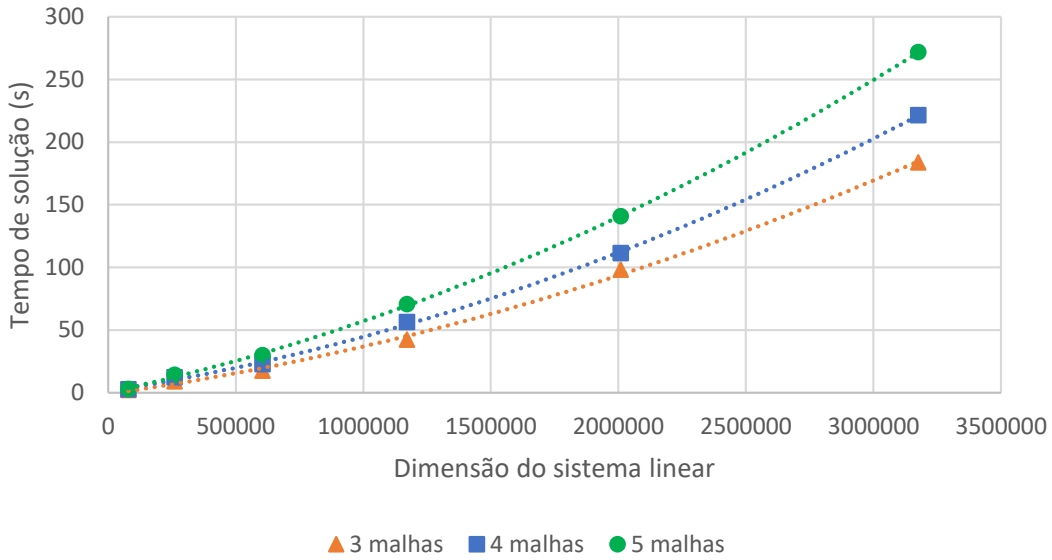


Figura 7.12: Tempo de solução em função da dimensão do sistema linear para problemas da viga em balanço, com diferentes quantidades de malhas do *multigrid* algébrico.

Conforme vimos nas Tabelas 7.15 e 7.16, a utilização de 3 malhas foi melhor do que 4 malhas e o ciclo W foi mais eficiente do que o ciclo V para os sistemas considerados. Testamos também a combinação desses parâmetros para os sistemas com dimensões

maiores, obtendo os resultados da Tabela 7.18. Observe que utilizar 3 níveis de malhas e o ciclo W reduz o número de iterações. Porém, essa não parece ser sempre a combinação mais eficiente com relação ao tempo de solução.

Tabela 7.18: Resultados variando a quantidade de malhas e o ciclo do *multigrid* algébrico.

Sistema	**Parâmetros**	N_{amg}	T_{amg}
`cb180x60x60`	4 malhas e ciclo V	192	111,38
	3 malhas e ciclo V	135	98,21
	4 malhas e ciclo W	134	**89,88**
	3 malhas e ciclo W	112	106,72
`mbb180x30x30`	4 malhas e ciclo V	333	60,38
	3 malhas e ciclo V	269	54,10
	4 malhas e ciclo W	263	**53,95**
	3 malhas e ciclo W	242	58,27
`tt30x150x30`	4 malhas e ciclo V	546	83,70
	3 malhas e ciclo V	479	**78,88**
	4 malhas e ciclo W	467	79,49
	3 malhas e ciclo W	437	83,98

Analisando agora o suavizador, primeiramente testamos diferentes fatores de relaxação ω. Neste caso, consideramos um número máximo de 1000 iterações para convergência do método dos gradientes conjugados. Para o método de Jacobi, obtivemos os resultados da Tabela 7.19. Note que a convergência é lenta para $\omega > 0{,}8$, ultrapassando as 1000 iterações. Os tempos de solução foram melhores para ω em torno de 0,6. Assim como no caso geométrico, por segurança, decidimos manter $\omega = 0{,}5$ para o Jacobi.

No caso do método SSOR, obtivemos os resultados da Tabela 7.20. Note que, em geral, o valor de ω mais eficiente com relação ao número de iterações e ao tempo de solução foi 1,4. Porém, observe também que há uma diferença significativa nos tempos de *setup* para cada valor de ω, principalmente nos problemas `mbb180x30x30` e `tt30x150x30`, por conta da geração do espaço de teste. Ademais, é difícil dizer qual é o melhor valor de ω para todos os sistemas lineares que aparecem no decorrer da resolução de cada problema de otimização topológica. Desse modo, escolhemos $\omega = 1{,}2$ para o método SSOR, pois apresentou bons resultados.

Utilizando os métodos de Jacobi (J) com relaxação $\omega = 0{,}5$ e o método SSOR com $\omega = 1{,}2$, na Tabela 7.21 apresentamos os resultados obtidos variando o suavizador e as quantidades de iterações na pré e na pós-suavização (η_1 e η_2). Para ambos os suavizadores, notamos que o tempo de solução do sistema foi menor utilizando $\eta_1 = \eta_2 = 1$, apesar dessa opção exigir mais iterações para convergência. Concluímos, portanto, que basta aplicar poucas vezes a suavização para que o método *multigrid* seja eficaz.

Agora, comparando os suavizadores com $\eta_1 = \eta_2 = 1$, notamos que, para todos os problemas, o método SSOR apresentou resultados melhores que o método de Jacobi com relação ao número de iterações e ao tempo de solução dos sistemas lineares, diferentemente do que ocorreu com o *multigrid* geométrico, no qual Jacobi obteve sempre os melhores resultados. Entretanto, o tempo de *setup* para o SSOR é muito maior do

que para o método de Jacobi, por conta da etapa de geração do espaço de teste e da preparação do suavizador. A matriz que precisa ser extraída e utilizada no problema de autovalores generalizado para o método de Jacobi é simplesmente a matriz diagonal com a mesma diagonal da matriz K do sistema. Já para o SSOR essa matriz é mais complexa.

Tabela 7.19: Resultados variando o parâmetro ω do suavizador Jacobi.

Sistema	ω	N_{amg}	T_{ts}	T_{sc}	T_c	T_p	TS_{amg}	T_{amg}
cb90x30x30	0,3	162	86,46	2,63	7,32	10,81	**107,76**	14,75
	0,5	134	88,19	2,65	7,39	11,05	109,85	12,25
	0,6	126	86,79	2,63	7,34	10,81	108,12	**11,45**
	2/3	193	87,73	2,63	7,32	10,87	109,10	17,66
	0,8	>1000	86,77	2,62	7,31	10,81	108,07	>92
mbb180x30x30	0,3	418	358,10	5,37	28,87	20,44	413,90	75,70
	0,5	333	363,20	5,34	28,58	20,61	418,86	60,38
	0,6	312	356,57	5,36	28,59	20,32	411,96	**56,44**
	2/3	421	355,47	5,33	28,75	20,26	**410,92**	75,86
	0,8	>1000	356,31	5,31	28,90	20,44	412,07	>181
tt30x150x30	0,3	685	269,79	4,46	18,56	17,45	**311,07**	104,96
	0,5	546	274,74	4,50	18,49	17,54	316,08	83,70
	0,6	507	271,72	4,47	18,68	17,58	313,25	**77,15**
	2/3	762	273,31	4,45	18,51	17,37	314,45	115,94
	0,8	>1000	273,57	4,49	18,15	17,38	314,38	>152

Tabela 7.20: Resultados variando o parâmetro ω do suavizador SSOR.

Sistema	ω	N_{amg}	T_{ts}	T_{sc}	T_c	T_p	TS_{amg}	T_{amg}
cb90x30x30	0,8	80	155,33	2,64	7,13	10,85	178,43	12,37
	1	76	153,17	2,63	7,19	10,89	176,34	11,62
	1,2	68	150,75	2,63	7,32	10,95	**174,18**	10,47
	1,4	67	151,85	2,63	7,05	11,00	175,03	**10,32**
	1,6	84	159,22	2,58	7,09	10,97	182,30	12,72
mbb180x30x30	0,8	178	631,08	5,34	29,34	20,78	**691,52**	54,45
	1	159	645,52	5,26	28,41	20,50	704,81	48,15
	1,2	146	671,60	5,33	28,87	20,66	731,67	44,98
	1,4	133	691,03	5,28	27,11	20,57	749,18	**40,64**
	1,6	141	727,14	5,29	27,61	20,79	785,98	43,22
tt30x150x30	0,8	349	488,30	4,49	18,86	17,80	**533,66**	90,63
	1	254	507,10	4,48	19,38	17,38	552,43	65,45
	1,2	227	543,72	4,46	19,37	17,80	589,58	58,77
	1,4	227	639,96	4,44	19,12	17,79	685,48	58,50
	1,6	215	619,74	4,36	18,60	17,58	664,34	**55,31**

Tabela 7.21: Resultados variando o suavizador do *multigrid* algébrico.

Sistema	**Suav.** (η_1, η_2)	N_{amg}	T_{ts}	T_{sc}	T_c	T_p	TS_{amg}	T_{amg}
cb60x20x20	J (1, 1)	96	18,17	0,80	1,75	3,52	**24,40**	2,71
	J (2, 2)	75	18,21	0,80	1,77	3,48	24,43	3,14
	SSOR (1, 1)	56	30,79	0,78	1,70	3,39	37,36	**2,67**
	SSOR (2, 2)	47	32,01	0,78	1,75	3,46	38,68	3,80
cb90x30x30	J (1, 1)	134	88,19	2,65	7,39	11,05	**109,85**	12,25
	J (2, 2)	106	88,99	2,63	7,40	10,77	110,35	14,41
	SSOR (1, 1)	68	150,75	2,63	7,32	10,95	174,18	**10,47**
	SSOR (2, 2)	57	154,81	2,59	7,36	11,00	178,26	14,97
mbb120x20x20	J (1, 1)	256	74,30	1,59	3,63	6,38	**86,21**	14,36
	J (2, 2)	196	75,32	1,58	3,62	6,24	87,07	16,28
	SSOR (1, 1)	113	140,34	1,59	3,59	6,49	153,47	**10,64**
	SSOR (2, 2)	93	140,20	1,54	3,59	6,45	153,16	14,95
mbb180x30x30	J (1, 1)	333	363,20	5,34	28,58	20,61	**418,86**	60,38
	J (2, 2)	255	364,02	5,34	28,84	20,37	419,69	69,29
	SSOR (1, 1)	146	671,60	5,33	28,87	20,66	731,67	**44,98**
	SSOR (2, 2)	118	673,57	5,19	28,20	20,51	732,67	61,70
tt20x100x20	J (1, 1)	355	82,22	1,32	2,73	5,47	91,99	16,55
	J (2, 2)	280	81,96	1,32	2,74	5,41	**91,65**	19,57
	SSOR (1, 1)	162	153,46	1,33	2,79	5,37	164,11	**12,83**
	SSOR (2, 2)	134	154,21	1,29	2,86	5,33	164,78	18,01
tt30x150x30	J (1, 1)	546	274,74	4,50	18,49	17,54	**316,08**	83,70
	J (2, 2)	395	280,70	4,49	18,37	17,38	321,74	90,56
	SSOR (1, 1)	227	543,72	4,46	19,37	17,80	589,58	**58,77**
	SSOR (2, 2)	175	542,28	4,42	19,86	17,93	588,70	78,45

Apesar da fase de preparação do *multigrid* algébrico ser bastante cara, na resolução do problema completo de otimização topológica felizmente conseguimos realizar a fase de *setup* uma única vez, antes da primeira iteração da PLS, e utilizar as componentes obtidas para resolver os sistemas em todas as demais iterações. Nesta situação, fica difícil perceber qual é o melhor suavizador, pois precisamos levar em consideração o tempo de *setup* e os tempos de solução dos sistemas. Para tentar obter resultados mais conclusivos, resolvemos os problemas completos de otimização topológica variando o suavizador, obtendo os resultados da Tabela 7.22. Com isso, observamos que o tempo total gasto para resolver o problema (T_{total}) é sempre menor para o suavizador Jacobi. Note que o número de sistemas resolvidos (N_s) fica igual mesmo trocando o suavizador.

Na Tabela 7.23, variamos o parâmetro utilizado para decidir se as conexões entre os nós da malha são fortes ou não. Lembramos que o valor de θ_s deve estar no intervalo $[0, 1)$. Além disso, quanto mais próximo de zero está este valor, mais conexões são consideradas como fortes e, quanto mais próximo de 1, menos conexões são consideradas

fortes. Este parâmetro não parece influenciar significativamente os tempos de preparação das componentes do *multigrid* algébrico, isto é, as etapas de *setup*. Já o número de iterações e o tempo de solução dos sistemas lineares, em geral, foram melhores para θ_s mais próximo de 1. Ou seja, o método é mais eficaz quando consideramos menos conexões como fortes. Note que não podemos utilizar $\theta_s = 1$, pois, neste caso, não teríamos nenhuma conexão forte e não seria possível construir as malhas grossas da maneira como estamos fazendo.

Tabela 7.22: Resultados variando o suavizador do *multigrid* algébrico resolvendo o problema completo de otimização topológica.

Problema	**Suav.** (η_1, η_2)	N_s	T_{total}
`cb60x20x20`	J (1, 1)	39	**224,81**
	SSOR (1, 1)	39	244,38
`cb90x30x30`	J (1, 1)	23	**776,12**
	SSOR (1, 1)	23	796,41
`mbb120x20x20`	J (1, 1)	46	**963,33**
	SSOR (1, 1)	46	1113,94
`mbb180x30x30`	J (1, 1)	25	**2540,69**
	SSOR (1, 1)	25	2887,15
`tt20x100x20`	J (1, 1)	56	**1224,11**
	SSOR (1, 1)	56	1241,72
`tt30x150x30`	J (1, 1)	38	**3210,50**
	SSOR (1, 1)	38	3372,58

Segundo Franceschini *et al.* [20], esta forma de cálculo das conexões fortes, de fato, gera valores muito próximos de 1 e é difícil escolher a melhor tolerância para decidir quais delas serão fortes. Os autores utilizam uma estratégia um pouco diferente, na qual escolhem um valor θ_f inteiro como sendo a quantidade de conexões fortes que cada nó pode possuir. Testamos essa estratégia, variando o parâmetro θ_f, e obtivemos os resultados da Tabela 7.24. Note que o número de iterações cresce conforme θ_f aumenta e que o tempo de solução do sistema, em geral, é menor para $\theta_f = 20$. Além disso, observe que os tempos de todas as etapas de *setup* sofrem alterações significativas conforme variamos θ_f, o que não ocorre na outra estratégia que testamos. Em geral, o tempo total de *setup* parece diminuir conforme aumentamos θ_f.

Variando a quantidade n_{tv} de vetores na matriz do espaço de teste, obtivemos os resultados da Tabela 7.25. Como esperado, quanto maior for essa quantidade, mais cara fica a etapa de geração do espaço de teste, pois precisamos encontrar mais autovetores. Além disso, o tempo gasto no cálculo das forças de conexões e na prolongação também cresce. Porém, utilizar uma quantidade muito pequena não parece ser uma boa ideia quando olhamos para o tempo de solução do sistema linear, o qual, na maioria dos casos, diminui conforme n_{tv} aumenta. Assim, precisamos escolher um valor intermediário, para o qual o método seja eficaz, mas sem prejudicar demais o tempo de *setup*.

Para tentar obter resultados mais conclusivos, resolvemos os problemas completos de otimização topológica variando o parâmetro n_{tv}, obtendo a Tabela 7.26. Observe que o valor $n_{tv} = 30$ forneceu os menores tempos em três dos seis problemas testados e

ficou em segundo lugar em outros dois problemas. Portanto, acreditamos que esse seja um bom valor para o parâmetro. Todavia, é complicado dizer exatamente qual é o melhor valor, pois os resultados são sensíveis ao problema que está sendo resolvido.

Tabela 7.23: Resultados variando o parâmetro θ_s do *multigrid* algébrico.

Sistema	θ_s	N_{amg}	T_{ts}	T_{sc}	T_c	T_p	TS_{amg}	T_{amg}
cb60x20x20	0,2	257	18,18	0,81	1,62	3,59	24,35	7,02
	0,3	148	17,86	0,79	1,65	3,56	24,01	4,12
	0,5	96	18,17	0,80	1,75	3,52	24,40	2,71
	0,7	89	17,87	0,79	1,77	3,43	24,01	2,46
	0,9	89	17,82	0,78	1,75	3,37	**23,88**	2,49
	0,95	77	18,01	0,83	1,89	3,46	24,39	**2,25**
cb90x30x30	0,2	373	85,74	2,64	6,64	11,29	**106,81**	33,41
	0,3	290	86,64	2,65	7,09	11,25	108,15	26,30
	0,5	134	88,19	2,65	7,39	11,05	109,85	12,25
	0,7	100	86,08	2,63	7,17	10,90	107,34	**9,09**
	0,9	99	86,48	2,65	7,51	10,97	108,19	**9,09**
	0,95	98	87,98	2,60	7,32	10,75	109,23	9,15
mbb120x20x20	0,2	506	74,75	1,59	3,44	6,55	86,62	27,88
	0,3	274	74,77	1,61	3,72	6,54	86,96	15,35
	0,5	256	74,30	1,59	3,63	6,38	86,21	14,36
	0,7	206	73,87	1,59	3,70	6,54	86,01	11,55
	0,9	174	74,43	1,60	3,82	6,43	86,58	9,77
	0,95	165	72,67	1,60	3,72	6,43	**84,75**	**9,39**
mbb180x30x30	0,2	833	357,88	5,35	29,85	20,62	414,72	147,91
	0,3	411	359,20	5,36	29,31	20,74	415,69	73,69
	0,5	333	363,20	5,34	28,58	20,61	418,86	60,38
	0,7	250	354,04	5,30	26,48	20,31	**407,11**	44,76
	0,9	208	357,16	5,31	27,78	20,35	411,56	37,27
	0,95	192	356,33	5,22	27,48	19,79	409,84	**34,56**
tt20x100x20	0,2	722	80,16	1,33	2,66	5,51	89,90	33,27
	0,3	524	80,75	1,32	2,72	5,42	90,44	24,14
	0,5	355	82,22	1,32	2,73	5,47	91,99	16,55
	0,7	271	80,58	1,31	2,72	5,30	90,15	12,55
	0,9	260	80,59	1,32	2,75	5,24	90,14	12,11
	0,95	244	79,95	1,28	2,82	5,17	**89,49**	**11,51**
tt30x150x30	0,2	1031	270,90	4,47	17,91	17,98	312,06	155,63
	0,3	747	271,93	4,45	18,55	17,77	313,48	113,02
	0,5	546	274,74	4,50	18,49	17,54	316,08	83,70
	0,7	426	270,32	4,48	19,65	17,69	313,00	65,29
	0,9	275	272,39	4,43	18,96	17,87	314,51	42,10
	0,95	265	269,42	4,37	19,01	17,23	**310,97**	**40,84**

Tabela 7.24: Resultados variando o parâmetro de conexões fortes θ_f do *multigrid* algébrico conforme a estratégia de Franceschini *et al.* [20].

Sistema	θ_f	N_{amg}	T_{ts}	T_{sc}	T_c	T_p	TS_{amg}	T_{amg}
cb60x20x20	5	43	26,51	1,59	6,63	3,09	38,78	2,85
	10	49	23,12	1,42	4,57	3,70	33,34	**2,14**
	20	61	21,27	1,27	3,18	4,26	30,34	2,18
	30	67	19,96	1,22	2,96	4,50	28,95	2,18
	40	91	19,46	1,19	2,70	4,70	**28,32**	2,82
cb90x30x30	5	48	124,39	4,98	28,16	8,84	172,47	14,36
	10	58	108,70	4,55	21,66	10,97	148,15	9,14
	20	70	99,75	4,17	13,65	12,49	131,29	**8,25**
	30	82	94,99	4,03	11,63	13,40	125,04	8,90
	40	103	94,02	3,95	10,00	14,13	**123,01**	10,61
mbb120x20x20	5	80	98,14	3,00	15,91	5,23	125,16	13,20
	10	90	89,81	2,75	8,65	6,57	108,96	8,45
	20	112	82,31	2,54	6,45	7,70	99,68	**8,08**
	30	152	79,50	2,40	5,23	8,10	95,75	9,94
	40	206	78,13	2,37	4,73	8,62	**94,35**	12,86
mbb180x30x30	5	77	449,89	9,61	111,56	15,69	614,77	69,15
	10	101	416,75	9,04	60,71	19,95	513,05	38,65
	20	139	393,33	8,23	45,54	21,98	471,63	**32,87**
	30	189	376,25	7,86	37,76	23,41	447,15	39,67
	40	261	376,78	7,93	33,70	25,07	**445,20**	53,26
tt20x100x20	5	86	113,95	2,52	11,14	4,43	134,47	11,86
	10	122	96,74	2,35	6,81	5,71	112,66	9,87
	20	158	89,98	2,08	4,59	6,58	103,80	**9,35**
	30	197	87,81	1,99	3,83	6,91	101,03	10,84
	40	235	85,88	1,95	3,50	7,45	**99,20**	12,38
tt30x150x30	5	86	377,72	8,16	91,12	13,63	510,90	62,51
	10	132	344,62	7,51	46,21	16,92	420,74	42,62
	20	186	297,81	6,70	31,46	19,17	357,20	**37,05**
	30	214	290,02	6,53	27,25	20,49	345,90	38,49
	40	294	279,44	6,50	24,80	21,73	**333,90**	50,06

No cálculo dos autovetores da geração do espaço de teste, consideramos como critérios de parada a norma do resíduo relativa menor que uma tolerância ε_r e a diferença relativa entre os valores do quociente de Rayleigh em duas iterações consecutivas menor que uma outra tolerância ε_q. Na Tabela 7.27, comparamos os resultados variando essas duas tolerâncias. Observe que o tempo gasto na geração do espaço de teste (T_{ts}) diminui bastante ao aumentarmos as tolerâncias. Porém, ao utilizar valores muito altos, como $\varepsilon_r = \varepsilon_q = 10^{-1}$, notamos um aumento significativo no tempo de solução do sistema linear,

uma vez que os vetores no espaço de teste perdem muita precisão. Felizmente, ainda podemos utilizar tolerâncias grandes sem comprometer a eficiência do *multigrid*.

Tabela 7.25: Resultados variando o parâmetro n_{tv} do *multigrid* algébrico.

Sistema	n_{tv}	N_{amg}	T_{ts}	T_{sc}	T_c	T_p	TS_{amg}	T_{amg}
cb60x20x20	5	250	9,81	0,68	1,62	3,09	**15,32**	6,69
	10	96	18,17	0,80	1,75	3,52	24,40	2,71
	20	92	32,22	0,89	1,76	3,88	38,91	2,62
	30	95	49,21	0,97	1,67	4,29	56,32	2,74
	40	85	65,97	1,02	1,59	4,43	73,17	**2,37**
cb90x30x30	5	414	47,32	2,43	6,74	9,55	**66,46**	36,62
	10	134	88,19	2,65	7,39	11,05	109,85	12,25
	20	109	160,15	2,92	7,28	12,36	183,29	10,02
	30	113	247,66	3,23	7,37	13,90	272,73	10,39
	40	108	329,64	3,42	7,12	14,32	355,07	**9,87**
mbb120x20x20	5	271	52,35	1,50	3,78	5,30	**63,17**	15,11
	10	256	74,30	1,59	3,63	6,38	86,21	14,36
	20	199	122,83	1,77	3,89	7,50	136,32	11,17
	30	208	178,43	1,95	3,73	8,36	192,83	11,80
	40	186	228,61	2,10	3,71	8,43	243,18	**10,51**
mbb180x30x30	5	352	251,35	4,91	29,14	16,63	**302,91**	63,37
	10	333	363,20	5,34	28,58	20,61	418,86	60,38
	20	225	582,70	5,91	26,99	23,81	640,55	40,42
	30	173	859,56	6,53	26,13	26,13	919,56	**31,33**
	40	195	1185,03	6,97	26,64	27,21	1247,00	35,41
tt20x100x20	5	568	55,05	1,25	2,77	4,78	**64,06**	26,40
	10	355	82,22	1,32	2,73	5,47	91,99	16,55
	20	334	123,49	1,48	2,79	6,33	134,35	15,75
	30	276	158,33	1,64	2,91	7,04	170,24	13,07
	40	242	195,38	1,75	2,72	7,06	207,16	**11,27**
tt30x150x30	5	418	163,13	4,06	19,20	14,43	**201,54**	64,06
	10	546	274,74	4,50	18,49	17,54	316,08	83,70
	20	498	507,63	4,93	19,01	19,78	552,19	75,89
	30	347	725,56	5,46	18,71	21,88	772,57	53,21
	40	278	886,80	5,84	18,24	22,92	934,73	**42,88**

Novamente, para tentar obter resultados mais indicativos de quais valores escolher, resolvemos os problemas completos de otimização topológica variando as tolerâncias ε_r e ε_q. Dentre os pares $(\varepsilon_r, \varepsilon_q)$ testados na Tabela 7.27, acreditamos que $(10^{-1}, 10^{-1})$ não forneceu bons resultados, pois o tempo de solução do sistema cresce bastante e a redução no tempo de *setup* com relação ao par $(10^{-1}, 10^{-2})$ pode não ser satisfatória. Além disso,

os pares $(10^{-1}, 10^{-4})$, $(10^{-2}, 10^{-4})$ e $(10^{-4}, 10^{-4})$ apresentaram resultados parecidos, sendo que $(10^{-1}, 10^{-4})$ foi um pouco melhor, em geral.

Tabela 7.26: Resultados variando o parâmetro n_{tv} do *multigrid* algébrico resolvendo o problema completo de otimização topológica.

Problema	n_{tv}	N_s	T_{total}
`cb60x20x20`	5	39	290,93
	10	39	224,81
	20	39	215,63
	30	39	**212,20**
	40	39	232,21
`cb90x30x30`	5	23	1049,70
	10	23	776,12
	20	23	**742,72**
	30	23	759,02
	40	23	790,83
`mbb120x20x20`	5	46	1224,42
	10	46	963,33
	20	46	990,48
	30	46	978,72
	40	46	**940,77**
`mbb180x30x30`	5	25	3034,73
	10	25	2540,69
	20	25	2638,36
	30	25	**2536,40**
	40	25	3214,16
`tt20x100x20`	5	56	1091,02
	10	56	1224,11
	20	56	1089,37
	30	56	1011,31
	40	56	**964,55**
`tt30x150x30`	5	38	3341,27
	10	38	3210,50
	20	38	2767,23
	30	38	**2592,52**
	40	38	2744,75

Assim, testamos a resolução dos problemas completos apenas para os pares $(10^{-1}, 10^{-2})$, $(10^{-1}, 10^{-3})$ e $(10^{-1}, 10^{-4})$, obtendo os resultados da Tabela 7.28. Observe que os menores tempos (T_{total}) foram obtidos para $(10^{-1}, 10^{-2})$ em quatro dos seis problemas testados. Apenas nos dois problemas da viga MBB os tempos para $(10^{-1}, 10^{-3})$ foram um pouco menores. Note também que a quantidade de sistemas resolvidos (N_s) fica igual mesmo trocando os parâmetros, com exceção do problema `tt30x150x30`, para o

qual essa quantidade com o par $(10^{-1}, 10^{-3})$ foi maior do que com os demais. Decidimos então escolher $\varepsilon_r = 10^{-1}$ e $\varepsilon_q = 10^{-2}$.

Tabela 7.27: Resultados variando os parâmetros ε_r e ε_q do *multigrid* algébrico.

Sistema	ε_r	ε_q	N_{amg}	T_{ts}	T_{sc}	T_c	T_p	TS_{amg}	T_{amg}
cb60x20x20	10^{-1}	10^{-1}	146	3,13	0,79	1,67	3,63	**9,39**	4,03
	10^{-1}	10^{-2}	88	8,07	0,77	1,67	3,44	14,12	**2,42**
	10^{-1}	10^{-3}	111	14,05	0,80	1,75	3,58	20,34	3,11
	10^{-1}	10^{-4}	99	17,86	0,78	1,75	3,43	23,98	2,75
	10^{-2}	10^{-4}	96	18,17	0,80	1,75	3,52	24,40	2,71
	10^{-4}	10^{-4}	96	17,96	0,77	1,75	3,41	24,05	2,66
cb90x30x30	10^{-1}	10^{-1}	311	10,68	2,63	7,45	12,20	**33,53**	28,18
	10^{-1}	10^{-2}	115	35,50	2,59	6,86	11,16	56,65	**10,43**
	10^{-1}	10^{-3}	132	66,09	2,66	7,23	10,99	87,50	11,97
	10^{-1}	10^{-4}	129	88,89	2,63	7,41	11,00	110,48	11,82
	10^{-2}	10^{-4}	134	88,19	2,65	7,39	11,05	109,85	12,25
	10^{-4}	10^{-4}	134	87,43	2,62	7,37	10,99	108,96	12,21
mbb120x20x20	10^{-1}	10^{-1}	392	7,14	1,60	3,83	7,48	**20,37**	21,84
	10^{-1}	10^{-2}	316	27,45	1,56	3,62	6,69	39,63	17,57
	10^{-1}	10^{-3}	253	54,06	1,59	3,62	6,43	66,01	14,12
	10^{-1}	10^{-4}	251	73,08	1,56	3,63	6,39	84,95	**13,98**
	10^{-2}	10^{-4}	256	74,30	1,59	3,63	6,38	86,21	14,36
	10^{-4}	10^{-4}	256	73,18	1,56	3,61	6,41	85,04	14,26
mbb180x30x30	10^{-1}	10^{-1}	675	24,49	5,31	33,87	23,60	**88,50**	121,12
	10^{-1}	10^{-2}	575	91,45	5,26	31,17	22,39	151,43	102,99
	10^{-1}	10^{-3}	353	223,69	5,36	27,72	20,57	278,47	63,21
	10^{-1}	10^{-4}	333	359,61	5,25	28,66	20,62	415,27	60,19
	10^{-2}	10^{-4}	333	363,20	5,34	28,58	20,61	418,86	60,38
	10^{-4}	10^{-4}	333	360,03	5,23	28,64	20,60	415,63	**60,16**
tt20x100x20	10^{-1}	10^{-1}	476	5,90	1,31	2,89	6,02	**16,38**	22,30
	10^{-1}	10^{-2}	319	22,44	1,30	2,78	5,57	32,34	**14,92**
	10^{-1}	10^{-3}	356	59,04	1,31	2,77	5,39	68,75	16,62
	10^{-1}	10^{-4}	343	79,50	1,31	2,74	5,42	89,20	15,95
	10^{-2}	10^{-4}	355	82,22	1,32	2,73	5,47	91,99	16,55
	10^{-4}	10^{-4}	355	79,81	1,30	2,73	5,41	89,48	16,59
tt30x150x30	10^{-1}	10^{-1}	786	17,60	4,38	18,58	19,54	**61,08**	119,49
	10^{-1}	10^{-2}	690	93,78	4,42	18,83	18,06	136,02	105,60
	10^{-1}	10^{-3}	616	183,74	4,53	19,26	18,35	226,74	94,37
	10^{-1}	10^{-4}	541	274,91	4,39	18,25	17,45	315,81	**82,92**
	10^{-2}	10^{-4}	546	274,74	4,50	18,49	17,54	316,08	83,70
	10^{-4}	10^{-4}	546	273,28	4,39	18,16	17,41	314,04	83,62

Tabela 7.28: Resultados variando os parâmetros ε_r e ε_q do *multigrid* algébrico resolvendo o problema completo de otimização topológica.

Problema	ε_r	ε_q	N_s	T_{total}
cb60x20x20	10^{-1}	10^{-2}	39	**224,89**
	10^{-1}	10^{-3}	39	226,25
	10^{-1}	10^{-4}	39	229,16
cb90x30x30	10^{-1}	10^{-2}	23	**713,86**
	10^{-1}	10^{-3}	23	725,60
	10^{-1}	10^{-4}	23	795,38
mbb120x20x20	10^{-1}	10^{-2}	46	1070,80
	10^{-1}	10^{-3}	46	**950,79**
	10^{-1}	10^{-4}	46	951,38
mbb180x30x30	10^{-1}	10^{-2}	25	2578,49
	10^{-1}	10^{-3}	25	**2449,28**
	10^{-1}	10^{-4}	25	2535,43
tt20x100x20	10^{-1}	10^{-2}	56	**1036,09**
	10^{-1}	10^{-3}	56	1144,36
	10^{-1}	10^{-4}	56	1205,96
tt30x150x30	10^{-1}	10^{-2}	38	**2813,58**
	10^{-1}	10^{-3}	57	4184,18
	10^{-1}	10^{-4}	38	3156,19

Na Tabela 7.29, variamos o número máximo de iterações it_p utilizado como critério de parada no algoritmo da prolongação DPLS. Esse parâmetro praticamente não teve influência nas etapas de *setup*, com exceção da etapa de construção das matrizes de prolongação, que ficou um pouco mais rápida para o valor $it_p = 2$. Em geral, o valor padrão $it_p = 5$ apresentou bons resultados com relação ao tempo de solução do sistema.

No algoritmo da prolongação DPLS, também utilizamos a tolerância κ para o número de condição. Variando essa tolerância, obtivemos os resultados da Tabela 7.30. Novamente, não houve mudança significativa nos tempos das etapas de *setup*, com exceção da construção das matrizes de prolongação, que fica mais rápida quanto menor é o valor de κ. Já em relação ao número de iterações e ao tempo de solução do sistema linear, observamos um crescimento quando usamos um valor muito grande ou muito pequeno para κ, de modo que os melhores resultados foram obtidos para $\kappa = 50$ e $\kappa = 100$. Escolhemos utilizar $\kappa = 100$.

Levando em consideração os resultados apresentados nesta seção para os problemas testados, consideramos que o melhor para a nossa implementação do método *multigrid* algébrico é utilizar 4 níveis de malhas, ciclo W, suavizador Jacobi com relaxação $\omega = 0{,}5$, $\eta_1 = \eta_2 = 1$ e os parâmetros $\theta_f = 20$, $n_{tv} = 30$, $\varepsilon_r = 10^{-1}$, $\varepsilon_q = 10^{-2}$, $it_p = 2$ e $\kappa = 100$. Na Tabela 7.31, apresentamos os resultados obtidos com esses "melhores" parâmetros e os comparamos com a resolução utilizando o precondicionador Cholesky incompleto, assim como fizemos no início da seção com os parâmetros padrão (Tabela 7.12).

Note que, agora, a redução do número de iterações e do tempo de solução do sistema linear foi maior, chegando-se a um tempo de solução 3 vezes melhor no caso do problema `mbb180x30x30`.

Tabela 7.29: Resultados variando o parâmetro it_p do *multigrid* algébrico.

Sistema	it_p	N_{amg}	T_{ts}	T_{sc}	T_c	T_p	TS_{amg}	T_{amg}
`cb60x20x20`	2	101	17,73	0,80	1,76	2,68	**23,11**	2,79
	5	96	18,17	0,80	1,75	3,52	24,40	2,71
	8	96	17,92	0,80	1,76	3,46	24,10	**2,67**
	10	96	17,94	0,80	1,75	3,44	24,09	2,67
`cb90x30x30`	2	118	86,63	2,65	7,37	8,68	**105,81**	**10,70**
	5	134	88,19	2,65	7,39	11,05	109,85	12,25
	8	134	86,79	2,66	7,35	10,79	108,15	12,15
	10	134	87,30	2,65	7,33	10,80	108,64	12,16
`mbb120x20x20`	2	251	73,12	1,58	3,59	5,26	**83,84**	**13,92**
	5	256	74,30	1,59	3,63	6,38	86,21	14,36
	8	256	73,50	1,59	3,60	6,33	85,32	14,28
	10	256	73,34	1,59	3,59	6,27	85,09	14,22
`mbb180x30x30`	2	336	353,09	5,35	28,57	17,22	**405,27**	60,24
	5	333	363,20	5,34	28,58	20,61	418,86	60,38
	8	333	354,91	5,38	28,57	20,42	410,39	59,88
	10	333	356,45	5,39	28,63	20,55	412,13	**59,86**
`tt20x100x20`	2	353	80,14	1,32	2,73	4,41	**88,81**	**16,40**
	5	355	82,22	1,32	2,73	5,47	91,99	16,55
	8	355	79,95	1,33	2,75	5,38	89,65	16,67
	10	355	80,25	1,32	2,76	5,37	89,95	16,60
`tt30x150x30`	2	545	269,52	4,48	18,34	14,47	**307,53**	**83,42**
	5	546	274,74	4,50	18,49	17,54	316,08	83,70
	8	546	274,69	4,52	18,40	17,52	315,93	83,73
	10	546	273,08	4,50	18,30	17,43	314,10	83,60

Pensando na resolução de problemas maiores, comparamos novamente o crescimento do tempo de solução em função da dimensão do sistema utilizando os "melhores" parâmetros com diferentes quantidades de malhas, obtendo os resultados da Tabela 7.32 e os gráficos da Figura 7.13. Observe que, para resolver os problemas grandes, foi melhor utilizar uma quantidade maior de malhas.

Os pontos para 3 malhas foram melhor aproximados por uma curva de tendência quadrática com coeficiente de determinação igual a 0,9998. Já os pontos para 4 e 5 malhas foram aproximados por curvas lineares, com coeficientes de determinação iguais a 0,9918 e 0,9984, respectivamente.

De um modo geral, obtivemos bons resultados para o *multigrid* algébrico com relação ao tempo de solução dos sistemas lineares. Em todos os casos, esse tempo foi

menor do que utilizando o precondicionador Cholesky incompleto. No entanto, as etapas de *setup* do AMG são bastante caras.

Tabela 7.30: Resultados variando o parâmetro κ do *multigrid* algébrico.

Sistema	κ	N_{amg}	T_{ts}	T_{sc}	T_c	T_p	TS_{amg}	T_{amg}
cb60x20x20	10	121	17,38	0,79	1,74	2,50	**22,52**	3,29
	50	96	18,17	0,80	1,75	3,52	24,40	2,71
	100	97	17,90	0,79	1,74	3,71	24,31	**2,69**
	500	104	18,78	0,81	1,80	4,21	25,79	2,96
cb90x30x30	10	147	85,03	2,56	7,36	7,91	**103,19**	13,22
	50	134	88,19	2,65	7,39	11,05	109,85	12,25
	100	124	87,96	2,62	7,35	11,80	110,30	**11,48**
	500	156	91,83	2,61	7,37	13,35	115,79	14,50
mbb120x20x20	10	280	73,27	1,56	3,63	4,83	**83,47**	15,25
	50	256	74,30	1,59	3,63	6,38	86,21	14,36
	100	258	75,50	1,58	3,60	6,92	87,90	**14,31**
	500	309	77,62	1,55	3,61	7,70	90,82	17,18
mbb180x30x30	10	367	353,62	5,16	29,07	15,71	**404,25**	64,89
	50	333	363,20	5,34	28,58	20,61	418,86	**60,38**
	100	341	360,63	5,32	28,72	22,47	418,35	61,98
	500	372	371,88	5,33	28,76	24,64	431,93	67,72
tt20x100x20	10	381	80,41	1,30	2,74	4,02	**88,62**	17,38
	50	355	82,22	1,32	2,73	5,47	91,99	**16,55**
	100	361	82,06	1,31	2,74	5,77	92,11	16,87
	500	434	82,96	1,30	2,73	6,29	93,56	20,52
tt30x150x30	10	572	263,66	4,34	18,40	13,13	**300,04**	85,21
	50	546	274,74	4,50	18,49	17,54	316,08	83,70
	100	540	270,79	4,39	18,07	18,72	312,81	**82,85**
	500	737	275,21	4,50	18,22	20,94	319,78	112,36

Tabela 7.31: Resultados com os "melhores" parâmetros do *multigrid* algébrico.

Sistema	N_{amg}	N_{ic}	TS_{amg}	TS_{ic}	T_{amg}	T_{ic}	T_{ic}/T_{amg}
cb60x20x20	50	360	32,70	0,20	2,36	5,66	2,40
cb90x30x30	55	551	144,01	0,64	9,64	27,29	2,83
mbb120x20x20	86	742	92,07	0,40	9,02	22,46	2,49
mbb180x30x30	91	1152	393,53	1,33	36,88	113,27	3,07
tt20x100x20	133	888	78,35	0,33	11,75	22,80	1,94
tt30x150x30	129	1301	340,72	1,06	40,90	107,45	2,63

Tabela 7.32: Resultados para sistemas da viga em balanço com diferentes refinamentos utilizando os "melhores" parâmetros e variando a quantidade de malhas do *multigrid* algébrico.

Sistema	Dim.	Malhas	N_{amg}	T_{amg}	T_{amg}
cb60x20x20	79380	3	46	**32,28**	2,69
		4	50	32,70	**2,36**
		5	52	33,22	2,43
cb90x30x30	259470	3	48	**143,41**	12,51
		4	55	144,01	9,64
		5	57	145,99	**9,05**
cb120x40x40	605160	3	49	391,63	40,77
		4	57	**389,48**	22,89
		5	60	391,06	**20,00**
cb150x50x50	1170450	3	50	**977,56**	114,46
		4	59	981,29	54,28
		5	63	980,45	**43,14**
cb180x60x60	2009340	3	51	2431,33	313,05
		4	58	**2211,74**	107,16
		5	64	2222,31	**77,23**

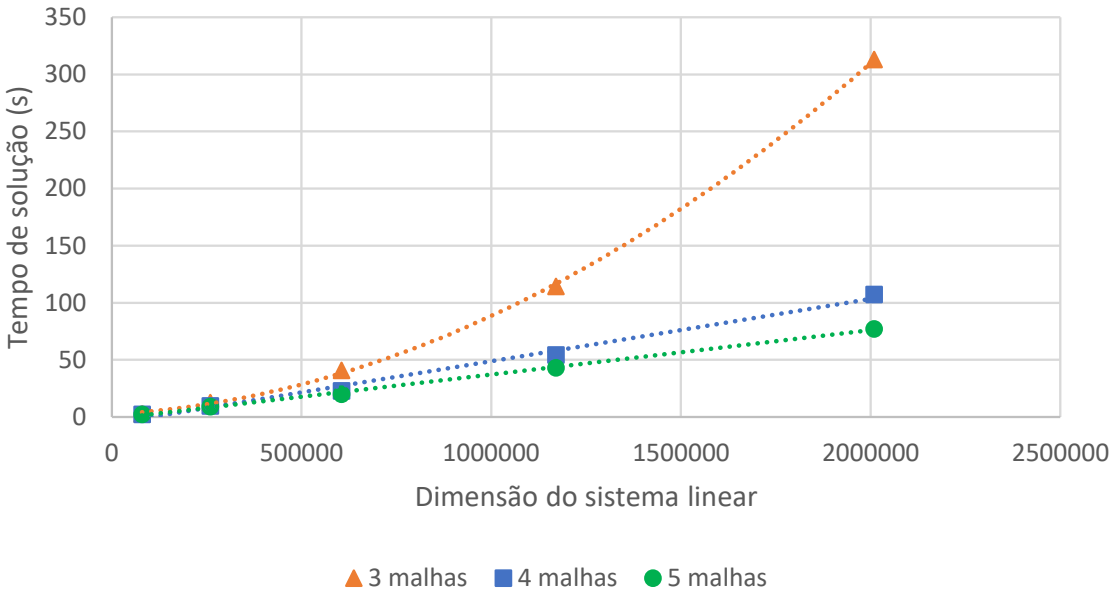


Figura 7.13: Tempo de solução em função da dimensão do sistema linear para problemas da viga em balanço, com diferentes quantidades de malhas do *multigrid* algébrico utilizando os "melhores" parâmetros.

7.4 *Multigrid* aplicado à otimização topológica

Após verificarmos o bom funcionamento do *multigrid* na resolução de alguns sistemas lineares provenientes da otimização topológica estrutural e ajustarmos os parâmetros do método, consideramos nesta seção uma análise da resolução dos problemas completos. Comparamos a utilização do método dos gradientes conjugados com precondicionador sendo a fatoração incompleta de Cholesky (ichol), o *multigrid* algébrico (AMG) e o *multigrid* geométrico (GMG) na resolução dos sistemas lineares.

Conforme vimos anteriormente, o *multigrid* algébrico aparenta ser eficiente para a resolução dos sistemas, porém com um custo de preparação elevado. Neste caso, tentamos realizar a fase de *setup* uma única vez, antes da primeira iteração da PLS, e utilizar as componentes obtidas do *multigrid* em todas as demais iterações, sendo necessário apenas recalcular as matrizes de iteração do suavizador, as matrizes do sistema em cada malha e o fator de Cholesky da matriz na malha mais grossa em cada iteração. Felizmente, isso funcionou bem e o método se mostrou eficaz na resolução dos sistemas lineares de todas as iterações, já que o custo para a realização do *setup* em mais de uma iteração poderia ser maior do que o ganho no tempo de solução.

O *multigrid* foi aplicado com 4 níveis de malhas, ciclo W, suavizador Jacobi com relaxação $\omega = 0{,}5$ e $\eta_1 = \eta_2 = 1$. No *multigrid* algébrico, utilizamos os parâmetros $\theta_f = 20$, $n_{tv} = 30$, $\varepsilon_r = 10^{-1}$, $\varepsilon_q = 10^{-2}$, $it_p = 2$ e $\kappa = 100$, que acreditamos ser a melhor combinação encontrada a partir das análises da Seção 7.3. Nas tabelas desta seção, utilizamos a seguinte legenda:

- N_{it}: número de iterações externas (com passo aceito) para convergência da PLS.
- N_s: quantidade de sistemas lineares resolvidos.
- F: valor ótimo da função objetivo.
- T_{sp}: tempo (em segundos) gasto na fase de *setup* do precondicionador.
- T_s: tempo (em segundos) gasto para resolver os sistemas lineares.
- T_{total}: tempo total (em segundos) gasto para resolver o problema.

Testamos os problemas da viga em balanço, da viga MBB e da torre de transmissão, com diferentes refinamentos da malha inicial. Na Tabela 7.33, apresentamos os resultados das resoluções completas de cada problema, comparando os diferentes precondicionadores. Observe que, para um mesmo problema, o número de iterações da PLS (N_{it}), o número de sistemas lineares resolvidos (N_s) e o valor ótimo da função objetivo (F), com quatro casas decimais de precisão, em geral foram iguais para os três precondicionadores, com exceção dos problemas da torre, para os quais o método com *multigrid* convergiu em menos iterações do que com ichol, obtendo, entretanto, um valor de função maior.

O tempo de *setup* do precondicionador (T_{sp}) foi sempre menor para o GMG e maior para o AMG. Os resultados do tempo de solução confirmam que o *multigrid*, tanto geométrico quanto algébrico, foi um precondicionador melhor que a fatoração incompleta de Cholesky, reduzindo consideravelmente o tempo de solução dos sistemas lineares (T_s) e, consequentemente, o tempo total de solução dos problemas (T_{total}).

Comparando os resultados entre o *multigrid* geométrico e o algébrico, notamos que o GMG é mais eficaz e também possui um *setup* muito rápido. A grande desvantagem do método é que ele só pode ser aplicado em problemas com domínios regulares e discretizações específicas, ou seja, é necessário que a quantidade de elementos em cada direção da malha inicial seja divisível por $2^{ngrids-1}$, sendo *ngrids* a quantidade de malhas utilizada. O *multigrid* algébrico é mais robusto, podendo ser utilizado em qualquer tipo de problema e, mesmo com um tempo de *setup* grande, se mostra mais eficiente do que a fatoração incompleta de Cholesky.

Tabela 7.33: Resolução dos problemas de otimização topológica, comparando os precondicionadores do método dos gradientes conjugados para resolver os sistemas lineares.

Problema	**Prec.**	N_{it}	N_s	F	T_{sp}	T_s	T_{total}
`cb96x32x32`	ichol	27	30	17,3682	27,70	2509,93	2887,59
	AMG	27	30	17,3682	171,69	522,81	1044,43
	GMG	27	30	17,3682	6,59	97,24	453,97
`cb120x40x40`	ichol	26	30	14,6952	53,83	6509,70	7628,74
	AMG	26	30	14,6952	371,04	1186,94	2625,93
	GMG	26	30	14,6952	12,57	175,63	1254,13
`mbb192x32x32`	ichol	32	36	12,7875	79,18	20272,02	21862,06
	AMG	32	36	12,7875	459,69	2594,27	4557,15
	GMG	32	36	12,7875	16,66	162,68	1704,05
`mbb240x40x40`	ichol	23	26	9,9553	106,67	32067,73	34809,10
	AMG	23	26	9,9553	1031,39	4490,94	8153,36
	GMG	23	26	9,9553	23,29	197,74	2864,92
`tt32x160x32`	ichol	33	35	31,6311	50,69	11052,08	11690,97
	AMG	25	27	31,8482	365,27	1791,43	2651,96
	GMG	25	27	31,8482	9,81	221,01	726,84
`tt40x200x40`	ichol	26	27	27,8722	73,72	19528,49	21125,10
	AMG	23	24	27,9789	783,20	3256,40	5473,88
	GMG	23	24	27,9789	16,81	212,55	1648,57

A comparação entre os tempos de resolução, apresentada na Tabela 7.33, também pode ser vista nos gráficos da Figura 7.14, para cada um dos problemas resolvidos. As barras azuis com marcador "×" representam a fatoração incompleta de Cholesky, as barras laranja com marcador "△" representam o *multigrid* algébrico e as barras verdes com marcador "○" representam o *multigrid* geométrico. Todos os gráficos mostram as mesmas conclusões: o tempo de *setup* é maior para o AMG e menor para o GMG, o tempo de resolução dos sistemas e o tempo total são maiores para o ichol, são reduzidos significativamente com o AMG e a redução é ainda maior com o GMG.

As Figuras 7.15-7.20 mostram as estruturas ótimas obtidas para cada um dos problemas testados. Notamos que não há diferença entre as estruturas obtidas com cada precondicionador, com exceção de pequenos detalhes nos problemas `tt32x160x32` e `tt40x200x40`, que são praticamente imperceptíveis. Nas legendas das figuras, informamos o raio do filtro, r, utilizado em cada problema.

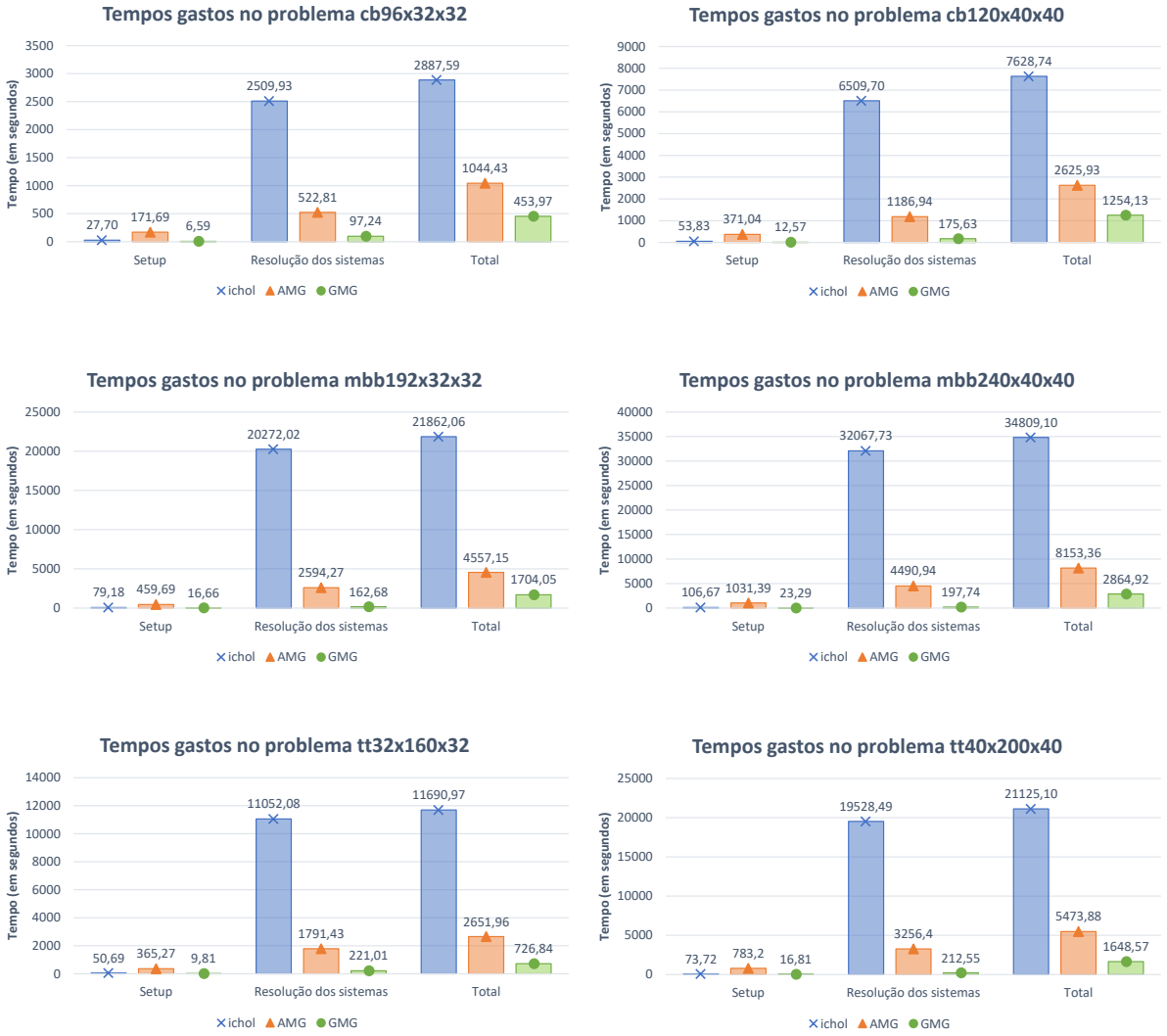
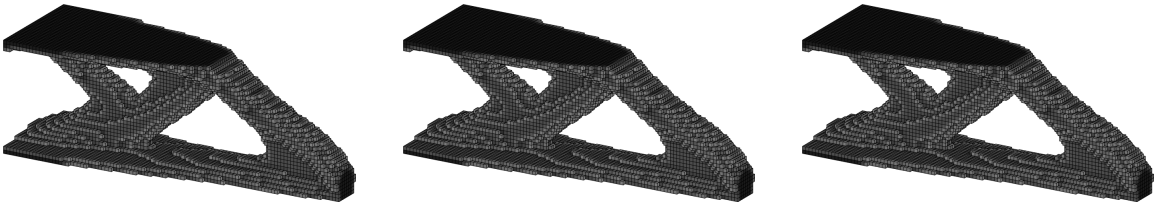
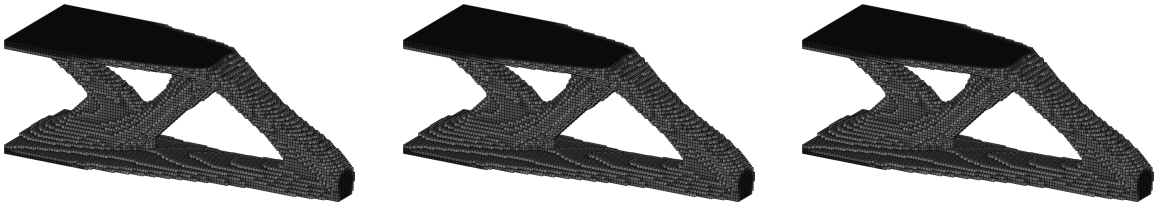


Figura 7.14: Comparação dos tempos de resolução dos problemas de otimização topológica utilizando diferentes precondicionadores do método dos gradientes conjugados.



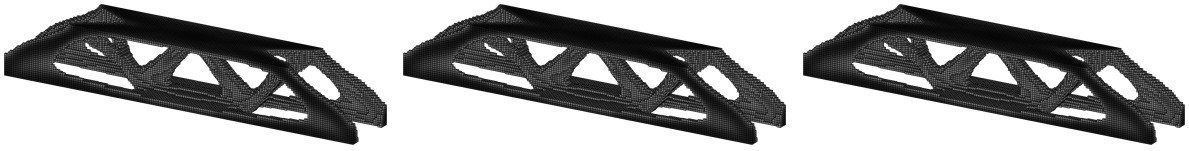
(a) ichol (b) AMG (c) GMG

Figura 7.15: Estruturas ótimas obtidas para o problema `cb96x32x32` ($r = 2{,}2$).



(a) ichol (b) AMG (c) GMG

Figura 7.16: Estruturas ótimas obtidas para o problema `cb120x40x40` ($r = 2{,}5$).



(a) ichol (b) AMG (c) GMG

Figura 7.17: Estruturas ótimas obtidas para o problema `mbb192x32x32` ($r = 5{,}0$).

(a) ichol (b) AMG (c) GMG

Figura 7.18: Estruturas ótimas obtidas para o problema `mbb240x40x40` ($r = 5{,}5$).

(a) ichol (b) AMG (c) GMG

Figura 7.19: Estruturas ótimas obtidas para o problema `tt32x160x32` ($r = 2{,}0$).

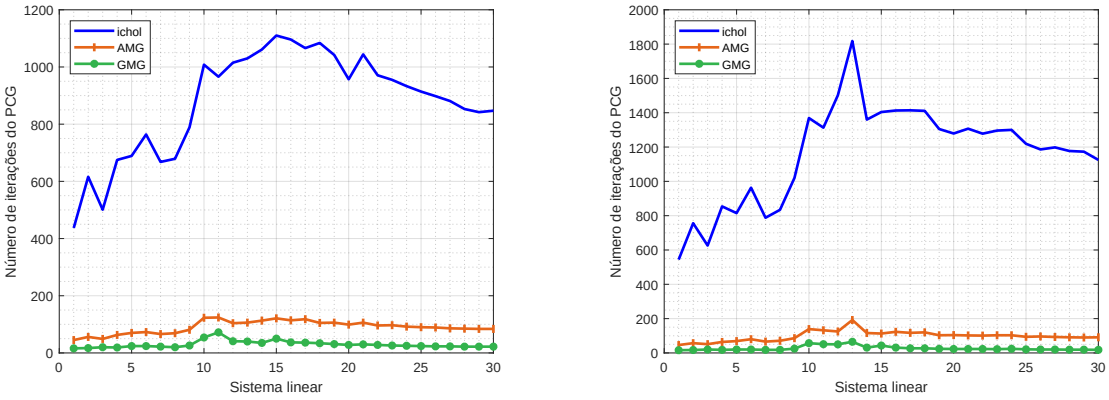
(a) ichol (b) AMG (c) GMG

Figura 7.20: Estruturas ótimas obtidas para o problema `tt40x200x40` ($r = 2{,}2$).

7.4.1 Número de iterações do método dos gradientes conjugados

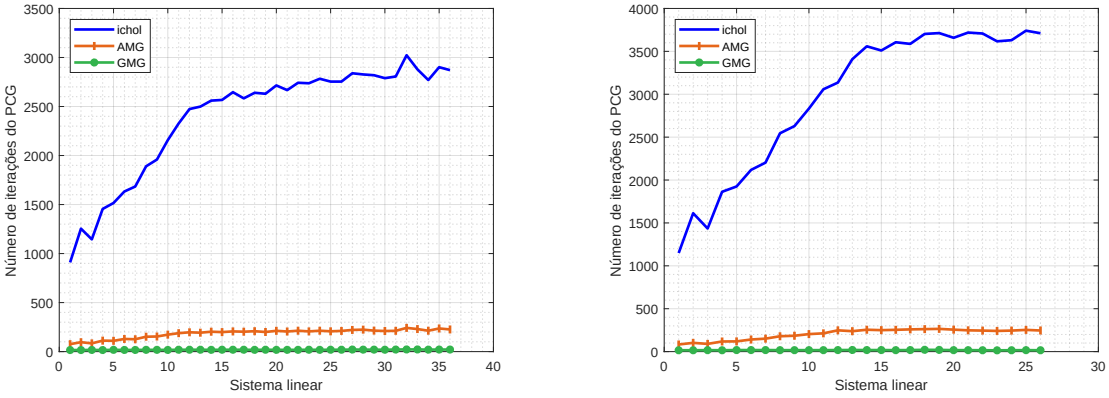
Nesta subseção, analisamos o número de iterações do método dos gradientes conjugados na resolução de cada um dos sistemas lineares durante o processo de otimização. As Figuras 7.21, 7.22 e 7.23 mostram gráficos da quantidade de iterações do PCG

para cada sistema linear dos problemas da viga em balanço, da viga MBB e da torre de transmissão, respectivamente, com cada um dos três precondicionadores. No problema `tt32x160x32`, consideramos apenas os primeiros 27 sistemas e, no problema `tt40x200x40`, os primeiros 24 sistemas, ainda que a resolução com ichol tenha exigido mais iterações.



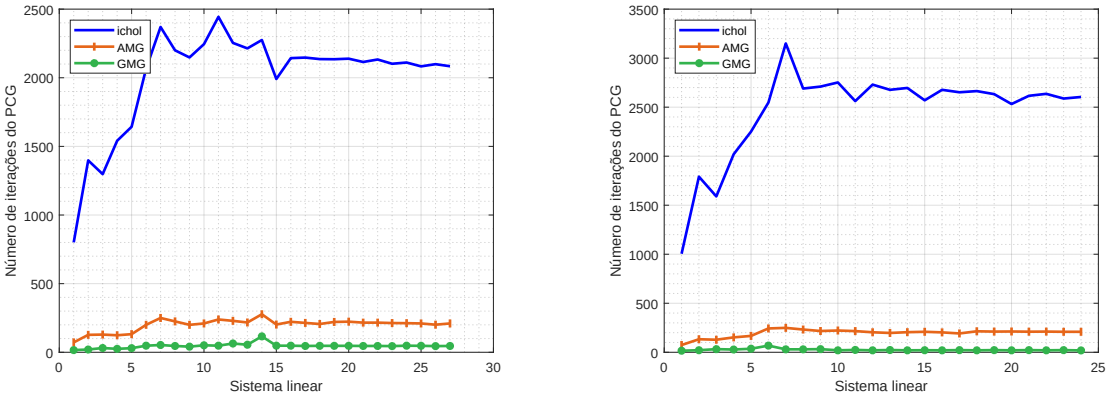
(a) `cb96x32x32` (b) `cb120x40x40`

Figura 7.21: Número de iterações do PCG para os problemas da viga em balanço.



(a) `mbb192x32x32` (b) `mbb240x40x40`

Figura 7.22: Número de iterações do PCG para os problemas da viga MBB.



(a) `tt32x160x32` (b) `tt40x200x40`

Figura 7.23: Número de iterações do PCG para os problemas da torre de transmissão.

Em geral, o método dos gradientes conjugados com o precondicionador ichol precisou de mais iterações para convergir e o precondicionador GMG foi o que gastou menos iterações. Notamos também que o comportamento do número de iterações do PCG no decorrer do processo de otimização topológica depende do problema resolvido.

Em todos os casos, há um crescimento grande do número de iterações no início. Isso provavelmente ocorre porque a distribuição das densidades de material muda bastante no início do processo e começam a surgir regiões vazias na estrutura, ou seja, elementos com densidade próxima de zero, o que aumenta o número de condição da matriz de rigidez. Como utilizamos a solução de um sistema como aproximação inicial para a resolução do próximo, a quantidade de iterações do PCG fica mais estável depois de um determinado momento.

No caso dos problemas da viga em balanço, notamos que há um pico no número de iterações entre o 10º e o 15º sistema linear. Neste momento, aparecem alguns sistemas difíceis de resolver, que gastam um número de iterações acima da média, o que acreditamos ser devido ao aumento no número de elementos com densidades próximas de zero. No problema `cb96x32x32`, por exemplo, o GMG gasta menos de 30 iterações, em geral, para resolver os sistemas, mas durante o pico ele ultrapassa 70 iterações. O número de elementos com densidade menor do que 0,01 é 33041 durante a resolução do 8º sistema e aumenta para 61428 no 11º sistema. Depois dessa elevação, o número de iterações do PCG volta a decrescer um pouco e se estabiliza. Algo parecido acontece com os problemas da torre de transmissão. No caso da viga MBB, depois do 10º sistema, a quantidade de iterações fica mais estável, porém não diminui como nos outros problemas.

7.4.2 Tempo gasto em cada etapa do algoritmo

Vimos que o *multigrid* apresenta ótimos resultados como precondicionador do método dos gradientes conjugados para a resolução dos sistemas lineares provenientes da otimização topológica, reduzindo significativamente o tempo gasto na resolução desses sistemas, que em geral é a etapa mais cara na resolução dos problemas. No problema `mbb240x40x40` da Tabela 7.33, por exemplo, observamos que o tempo gasto na resolução dos sistemas com o GMG foi menor que 24 segundos, enquanto o tempo total de resolução do problema foi aproximadamente 2865 segundos. Isso sugere que em alguns casos, principalmente para os problemas com dimensões grandes, o *multigrid* é tão eficaz que a resolução dos sistemas lineares deixa de ser a etapa que consome mais tempo no algoritmo de otimização topológica.

Tendo isso em vista, nesta subseção analisamos a distribuição do tempo gasto em cada etapa do algoritmo utilizando os diferentes precondicionadores (ichol, AMG e GMG), para cada um dos três tipos de estrutura testados. Dividimos os tempos das etapas principais do algoritmo da seguinte maneira:

- T_{pf}: tempo gasto na pré-filtragem.
- T_f: tempo gasto nas aplicações do filtro.
- T_k: tempo gasto para montar as matrizes de rigidez.
- T_{sp}: tempo gasto na fase de *setup* do precondicionador.

- T_s: tempo gasto para resolver os sistemas lineares.
- T_g: tempo gasto para os cálculos do gradiente da função objetivo.
- T_{lp}: tempo gasto na resolução dos subproblemas de programação linear.
- T_o: tempo gasto nas demais linhas de código.
- T_{total}: tempo total gasto para resolver o problema.

Todos os tempos são medidos em segundos e foram obtidos considerando a aplicação da PLS até a convergência. A etapa de pré-filtragem corresponde à obtenção dos vizinhos de cada elemento na malha inicial e ao cálculo dos fatores de peso, necessários para as aplicações do filtro de densidades. Essa etapa é realizada uma única vez, antes da primeira aplicação do filtro. O tempo T_k inclui a montagem da matriz de rigidez do elemento e da matriz global na malha mais fina, já o tempo para construir as matrizes em outros níveis de malha do *multigrid* é incluído em T_{sp}.

Nas Tabelas 7.34, 7.35 e 7.36, apresentamos os tempos gastos em cada etapa do algoritmo para os problemas `cb120x40x40`, `mbb240x40x40` e `tt40x200x40`, respectivamente. Ao lado de cada tempo, entre parênteses, mostramos a porcentagem do tempo gasto naquela etapa com relação ao tempo total de resolução.

Observe que apenas os tempos de *setup* (T_{sp}) e resolução dos sistemas (T_s) são efetivamente alterados quando mudamos o precondicionador. Para os três tipos de estruturas testados, quando usamos a fatoração incompleta de Cholesky as etapas que consumiram mais tempo foram a resolução dos sistemas lineares, a resolução dos subproblemas de programação linear e a construção das matrizes de rigidez, nesta ordem. Quando utilizamos o *multigrid* algébrico, a ordem entre essas etapas se manteve, mas o tempo de resolução dos sistemas lineares diminuiu e o tempo da etapa de *setup* entrou para os três maiores, ultrapassando a construção das matrizes de rigidez. Já no caso do *multigrid* geométrico, o tempo de resolução dos sistemas diminuiu bastante e a etapa que mais consumiu tempo foi a resolução dos subproblemas de PL. Agora, a resolução dos sistemas lineares ficou na segunda posição nos casos da viga em balanço e da torre de transmissão, e na terceira posição no caso da viga MBB.

Tabela 7.34: Tempos gastos em cada etapa do algoritmo para o problema `cb120x40x40`.

Tempo	ichol	AMG	GMG
T_{pf}	15,11 (0,20%)	15,49 (0,59%)	15,52 (1,24%)
T_f	0,46 (0,01%)	0,46 (0,02%)	0,45 (0,04%)
T_k	121,77 (1,60%)	122,15 (4,65%)	122,19 (9,74%)
T_{sp}	53,83 (0,71%)	371,04 (14,13%)	12,57 (1,00%)
T_s	6509,70 (85,33%)	1186,94 (45,20%)	175,63 (14,00%)
T_g	8,37 (0,11%)	8,35 (0,32%)	8,33 (0,66%)
T_{lp}	918,79 (12,04%)	921,18 (35,08%)	919,12 (73,29%)
T_o	0,71 (0,01%)	0,32 (0,01%)	0,32 (0,03%)
T_{total}	7628,74	2625,93	1254,13

Tabela 7.35: Tempos gastos em cada etapa do algoritmo para o problema `mbb240x40x40`.

Tempo	ichol	AMG	GMG
T_{pf}	130,31 (0,37%)	129,76 (1,59%)	132,18 (4,61%)
T_f	6,61 (0,02%)	6,69 (0,08%)	6,48 (0,23%)
T_k	215,77 (0,62%)	215,94 (2,65%)	215,05 (7,51%)
T_{sp}	106,67 (0,31%)	1031,39 (12,65%)	23,29 (0,81%)
T_s	32067,73 (92,12%)	4490,94 (55,08%)	197,74 (6,90%)
T_g	16,03 (0,05%)	16,13 (0,20%)	15,98 (0,56%)
T_{lp}	2264,75 (6,51%)	2261,84 (27,74%)	2273,56 (79,36%)
T_o	1,23 (0,00%)	0,67 (0,01%)	0,64 (0,02%)
T_{total}	34809,10	8153,36	2864,92

Tabela 7.36: Tempos gastos em cada etapa do algoritmo para o problema `tt40x200x40`.

Tempo	ichol	AMG	GMG
T_{pf}	16,47 (0,08%)	16,36 (0,30%)	16,64 (1,01%)
T_f	0,30 (0,00%)	0,28 (0,01%)	0,27 (0,02%)
T_k	184,35 (0,87%)	164,66 (3,01%)	163,82 (9,94%)
T_{sp}	73,72 (0,35%)	783,20 (14,31%)	16,81 (1,02%)
T_s	19528,49 (92,44%)	3256,40 (59,49%)	212,55 (12,89%)
T_g	13,75 (0,07%)	12,17 (0,22%)	12,10 (0,73%)
T_{lp}	1307,05 (6,19%)	1240,35 (22,66%)	1225,92 (74,36%)
T_o	0,97 (0,00%)	0,46 (0,01%)	0,46 (0,03%)
T_{total}	21125,10	5473,88	1648,57

A maior variação aconteceu no caso da viga MBB. Utilizando a fatoração incompleta de Cholesky como precondicionador, o tempo de resolução dos sistemas lineares representou 92,12% do tempo total. Com o AMG, essa porcentagem caiu para 55,08%, havendo uma redução no tempo T_s de aproximadamente 7,1 vezes. Já com o GMG, o tempo de resolução dos sistemas com relação ao precondicionador ichol foi aproximadamente 162 vezes menor e essa etapa representou apenas 6,90% do tempo total.

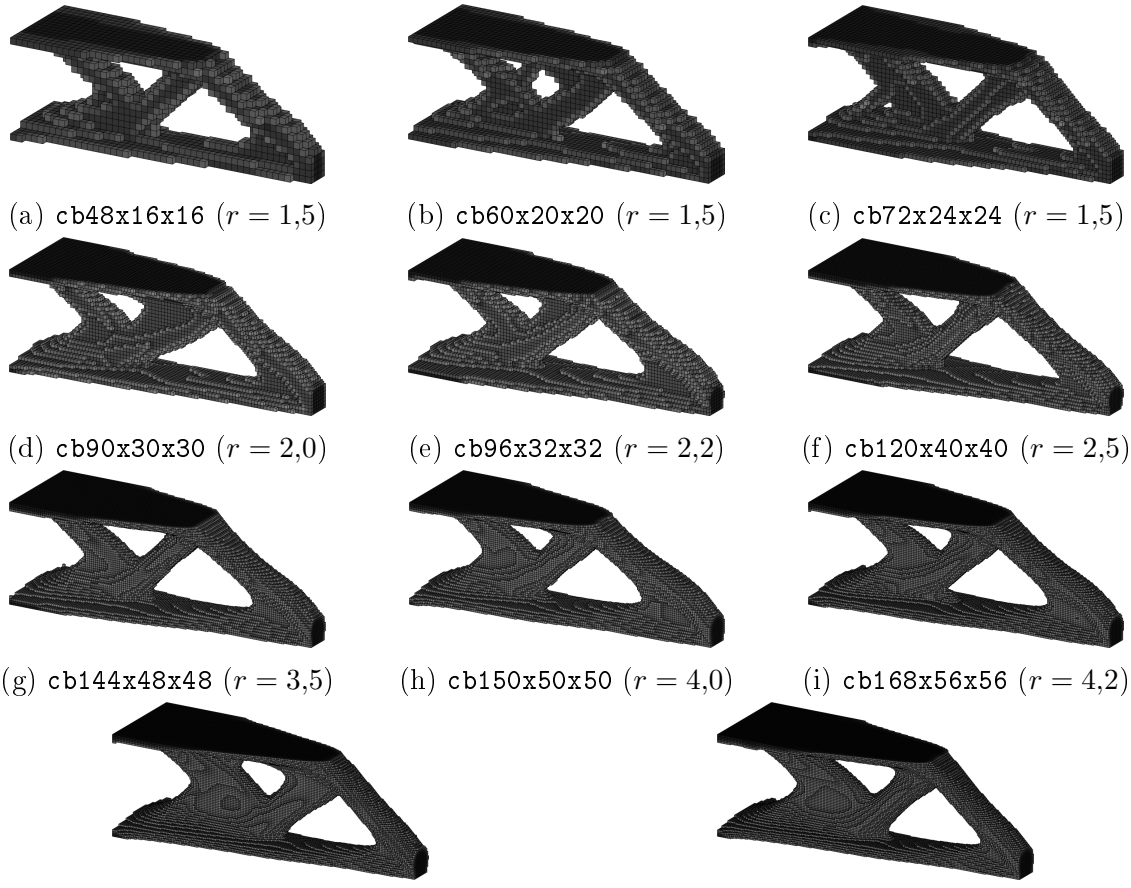
7.4.3 Crescimento do tempo com a dimensão

Nos resultados anteriores, vimos que a utilização do *multigrid* ajuda a reduzir o tempo de resolução dos sistemas lineares da otimização topológica. Entretanto, existem outras etapas no processo que consomem bastante tempo, fazendo com que o tempo total de resolução dos problemas com dimensões grandes ainda seja elevado. Pensando na resolução de problemas com malhas cada vez mais finas, ou seja, com mais elementos, nesta subseção analisamos o crescimento do tempo gasto em cada etapa do processo de otimização em função da quantidade de elementos na malha. Para tanto, resolvemos o problema da viga em balanço com diferentes refinamentos da malha.

Na Tabela 7.37, apresentamos os resultados das resoluções completas de cada um dos problemas com os diferentes precondicionadores. A coluna "Dim." da tabela apresenta o número total de elementos na malha inicial. Lembramos que nem todos os problemas podem ser resolvidos com o *multigrid* geométrico (GMG). Em alguns casos, seria possível utilizar apenas 2 ou 3 malhas e, consequentemente, a comparação dos tempos poderia ser injusta. Observamos novamente que, para um mesmo problema, o número de iterações da PLS (N_{it}), o número de sistemas lineares resolvidos (N_s) e o valor ótimo da função objetivo (F) foram iguais, independentemente do precondicionador utilizado. O tempo total de resolução foi maior para o precondicionador ichol e menor para o GMG. A Figura 7.24 mostra as estruturas ótimas obtidas para cada um dos refinamentos.

Tabela 7.37: Resolução dos problemas da viga em balanço com diferentes refinamentos.

Problema	**Dim.**	**Prec.**	N_{it}	N_s	F	T_{total}
`cb48x16x16`	12288	ichol	39	41	32,5918	213,14
		AMG	39	41	32,5918	106,44
		GMG	39	41	32,5918	45,73
`cb60x20x20`	24000	ichol	45	48	25,3198	653,20
		AMG	45	48	25,3198	281,79
`cb72x24x24`	41472	ichol	22	23	21,2219	694,05
		AMG	22	23	21,2219	284,40
		GMG	22	23	21,2219	129,19
`cb90x30x30`	81000	ichol	29	32	18,0806	2707,12
		AMG	29	32	18,0806	935,88
`cb96x32x32`	98304	ichol	27	30	17,3682	2887,59
		AMG	27	30	17,3682	1044,43
		GMG	27	30	17,3682	453,97
`cb120x40x40`	192000	ichol	26	30	14,6952	7628,74
		AMG	26	30	14,6952	2625,93
		GMG	26	30	14,6952	1254,13
`cb144x48x48`	331776	ichol	24	27	13,1159	13840,76
		AMG	24	27	13,1159	4771,34
		GMG	24	27	13,1159	2503,56
`cb150x50x50`	375000	ichol	27	32	12,8113	19198,90
		AMG	27	32	12,8113	6136,73
`cb168x56x56`	526848	ichol	27	33	11,7715	31052,36
		AMG	27	33	11,7715	9573,56
		GMG	27	33	11,7715	4706,57
`cb180x60x60`	648000	ichol	21	24	11,1901	28277,39
		AMG	21	24	11,1901	9838,11
`cb192x64x64`	786432	ichol	21	25	10,8924	38701,17
		AMG	21	25	10,8924	12956,46
		GMG	21	25	10,8924	5997,65



(a) `cb48x16x16` ($r = 1{,}5$) (b) `cb60x20x20` ($r = 1{,}5$) (c) `cb72x24x24` ($r = 1{,}5$)

(d) `cb90x30x30` ($r = 2{,}0$) (e) `cb96x32x32` ($r = 2{,}2$) (f) `cb120x40x40` ($r = 2{,}5$)

(g) `cb144x48x48` ($r = 3{,}5$) (h) `cb150x50x50` ($r = 4{,}0$) (i) `cb168x56x56` ($r = 4{,}2$)

(j) `cb180x60x60` ($r = 4{,}7$) (k) `cb192x64x64` ($r = 5{,}2$)

Figura 7.24: Estruturas ótimas obtidas para o problema da viga em balanço.

Estamos resolvendo problemas artificiais em que os elementos têm todos os lados de tamanho 1 e a intensidade da carga aplicada é 1 N, para qualquer refinamento da malha. Assim, quanto maior é a quantidade de nós, mais precisa fica a solução e os deslocamentos nodais aproximados são menores, uma vez que a intensidade da carga é sempre a mesma. Por esse motivo, o valor da função objetivo diminui conforme aumentamos o refinamento. Além disso, tentamos obter soluções independentes da malha, ou seja, o mesmo formato da estrutura para todos os refinamentos. Para tanto, como o tamanho dos elementos é sempre igual, precisamos aumentar o raio de aplicação do filtro de densidades e ajustar esse raio para cada problema. Observamos que a escolha de raios proporcionais às quantidades de elementos não foi capaz de eliminar o problema de dependência da malha em todos os refinamentos. Neste caso, para obter as estruturas da Figura 7.24, testamos diversos valores para r e escolhemos aqueles apresentados nas legendas da figura. Entretanto, para a análise de crescimento do tempo com a dimensão, é mais adequado utilizarmos raios que crescem de maneira proporcional, o que fizemos para obter os resultados apresentados a seguir.

Pela Tabela 7.37, notamos que, para cada refinamento, temos uma quantidade diferente de iterações e de sistemas resolvidos (N_{it} e N_s), de modo que não é possível comparar o tempo gasto na resolução dos sistemas lineares quando um problema precisa resolver 40 sistemas e o outro resolve apenas 20, por exemplo. Sendo assim, para uma análise mais efetiva, nos resultados apresentados a seguir paramos a resolução dos problemas em um número fixo de 10 repetições do laço principal do algoritmo, isto é, 10

iterações internas. Apenas no cálculo dos gradientes consideramos 10 iterações externas (quando o passo da PLS é aceito), pois o gradiente precisa ser recalculado apenas quando o passo é aceito. Esse valor de 10 iterações não ultrapassa a quantidade total de iterações necessárias para a resolução completa de nenhum dos problemas considerados.

Nas tabelas que seguem, comparamos o tempo gasto em cada etapa para os diferentes refinamentos da viga em balanço, utilizando os precondicionadores ichol, AMG e GMG na resolução dos sistemas lineares. Apresentamos o número de elementos na malha de cada problema (Dim.) e também o raio do filtro utilizado, pois este valor precisa ser aumentado conforme as dimensões crescem, influenciando os tempos gastos nas etapas de pré-filtragem, aplicações do filtro e cálculo dos gradientes. Para uma melhor análise, consideramos raios que crescem de maneira proporcional ao aumento na quantidade de elementos em cada direção da malha. Apresentamos também gráficos relacionados a cada tabela, que ilustram as tendências de crescimento do tempo em função da dimensão.

Tabela 7.38: Crescimento do tempo gasto na pré-filtragem.

Problema	**Dim.**	**Raio**	ichol	AMG	GMG
`cb48x16x16`	12288	1,2	0,30	0,27	0,26
`cb60x20x20`	24000	1,5	0,82	0,78	-
`cb72x24x24`	41472	1,8	1,49	1,45	1,47
`cb90x30x30`	81000	2,25	5,95	5,97	-
`cb96x32x32`	98304	2,4	7,28	7,19	7,29
`cb120x40x40`	192000	3,0	20,91	21,24	20,99
`cb144x48x48`	331776	3,6	46,36	45,22	46,21
`cb150x50x50`	375000	3,75	61,85	62,99	-
`cb168x56x56`	526848	4,2	109,33	107,79	108,76
`cb180x60x60`	648000	4,5	158,15	158,86	-
`cb192x64x64`	786432	4,8	200,71	199,93	202,20

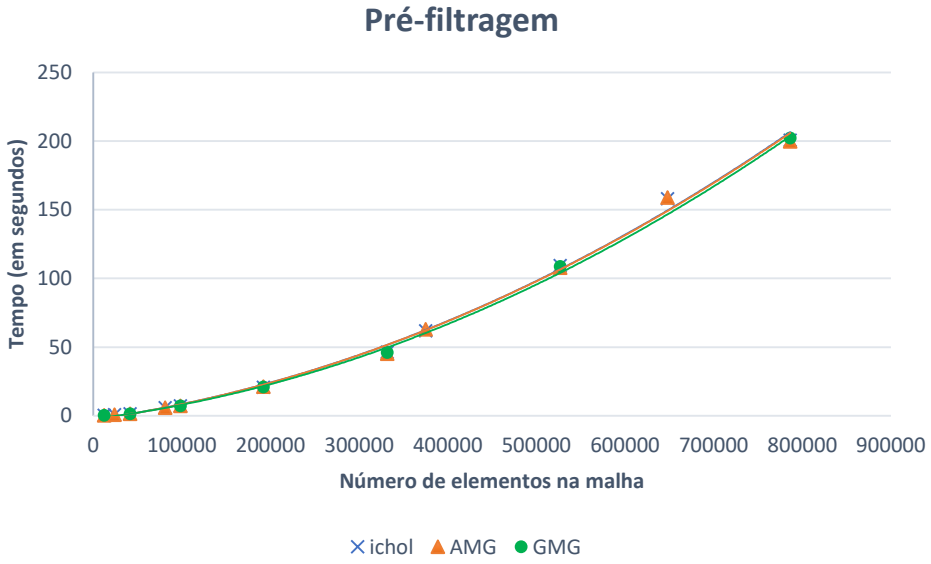


Figura 7.25: Crescimento do tempo gasto na pré-filtragem.

As tendências de crescimento dos tempos de pré-filtragem (Figura 7.25) e aplicações do filtro (Figura 7.26) são parecidas e ambas aparentam ser quadráticas. O valor de R^2 para a curva do AMG (em laranja, com marcador "△"), por exemplo, é igual a 0,9963 no caso da pré-filtragem e 0,9985 no caso das aplicações do filtro. Os tempos dessas etapas dependem também do raio do filtro utilizado, de modo que devemos tomar cuidado e evitar aumentar o raio demasiadamente. A pré-filtragem consome bastante tempo, porém ela é realizada apenas uma vez no início das iterações e, em compensação, torna baratas as aplicações do filtro.

Tabela 7.39: Crescimento do tempo gasto nas aplicações do filtro.

Problema	**Dim.**	**Raio**	ichol	AMG	GMG
`cb48x16x16`	12288	1,2	0,00	0,00	0,00
`cb60x20x20`	24000	1,5	0,01	0,01	-
`cb72x24x24`	41472	1,8	0,01	0,01	0,01
`cb90x30x30`	81000	2,25	0,04	0,04	-
`cb96x32x32`	98304	2,4	0,06	0,06	0,06
`cb120x40x40`	192000	3,0	0,25	0,26	0,25
`cb144x48x48`	331776	3,6	0,60	0,64	0,62
`cb150x50x50`	375000	3,75	0,97	0,97	-
`cb168x56x56`	526848	4,2	1,66	1,64	1,64
`cb180x60x60`	648000	4,5	2,40	2,37	-
`cb192x64x64`	786432	4,8	3,59	3,59	3,57

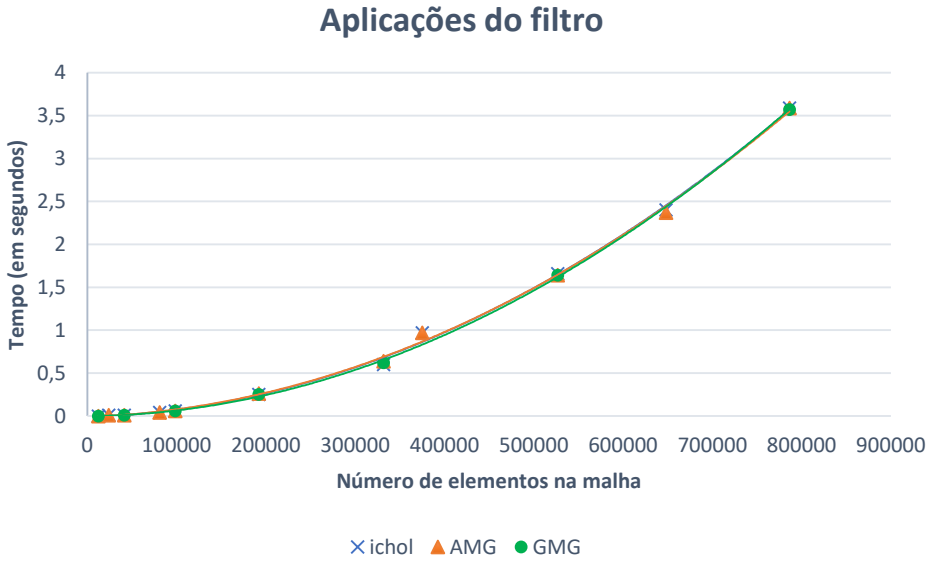


Figura 7.26: Crescimento do tempo gasto nas aplicações do filtro.

O tempo consumido na construção das matrizes de rigidez, apesar de ser grande, apresenta uma tendência de crescimento linear em função da dimensão do problema (Figura 7.27), com valor de R^2 igual a 0,9978 para o AMG, por exemplo. Além disso, a maior parte do tempo gasto nessa etapa corresponde ao uso da função `sparse` do Matlab, utilizada para armazenar a matriz na forma esparsa.

Tabela 7.40: Crescimento do tempo gasto para montar as matrizes de rigidez.

Problema	**Dim.**	**Raio**	ichol	AMG	GMG
cb48x16x16	12288	1,2	2,16	2,30	2,34
cb60x20x20	24000	1,5	4,51	4,51	-
cb72x24x24	41472	1,8	10,93	10,86	10,88
cb90x30x30	81000	2,25	21,00	21,23	-
cb96x32x32	98304	2,4	27,83	27,82	27,84
cb120x40x40	192000	3,0	54,84	54,78	54,89
cb144x48x48	331776	3,6	94,64	94,60	95,11
cb150x50x50	375000	3,75	107,61	108,29	-
cb168x56x56	526848	4,2	152,89	152,64	153,11
cb180x60x60	648000	4,5	172,33	172,92	-
cb192x64x64	786432	4,8	218,13	217,54	217,98

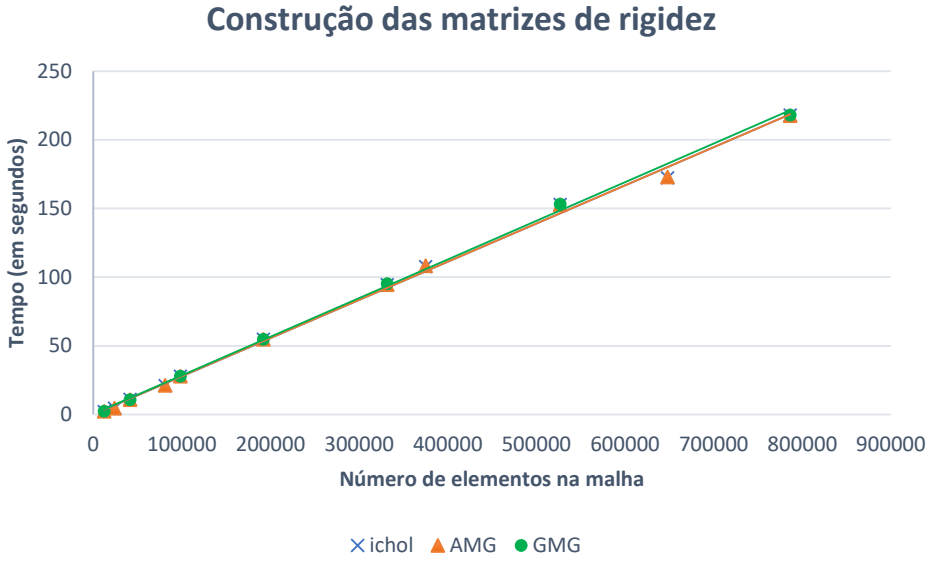
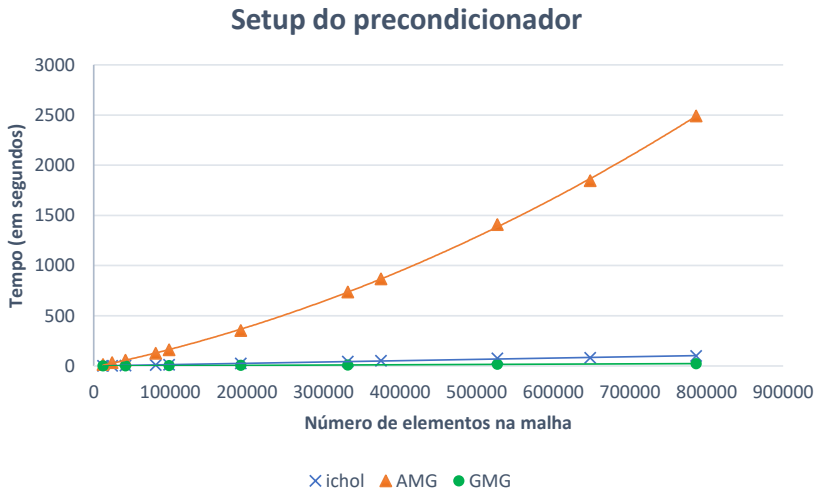


Figura 7.27: Crescimento do tempo gasto para montar as matrizes de rigidez.

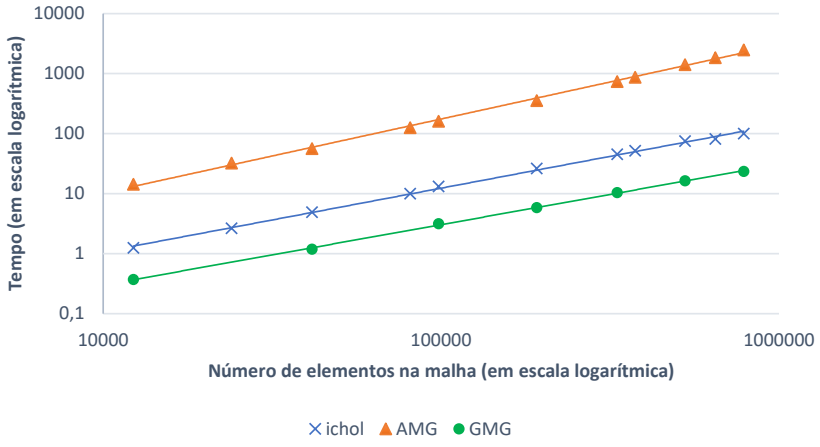
Os tempos de *setup* do precondicionador e de resolução dos sistemas lineares são os únicos que efetivamente se alteram quando mudamos o precondicionador. Pela Figura 7.28(a), podemos observar que a tendência de crescimento do tempo de *setup* para o precondicionador AMG é melhor aproximada por uma curva quadrática, com R^2 igual a 0,9998, e por uma reta para os precondicionadores ichol e GMG, com R^2 igual a 0,9950 e 0,9989, respectivamente. Para o GMG, o crescimento do tempo de *setup* é um pouco mais lento do que para a fatoração incompleta de Cholesky. O tempo gasto com o AMG é muito maior do que com os outros precondicionadores, de modo que fica difícil visualizar os gráficos de tendência para ichol e GMG. Para uma melhor análise, apresentamos também os gráficos com os eixos em escala logarítmica na Figura 7.28(b). Aproximando os pontos por curvas de tendência do tipo potência, o expoente obtido foi aproximadamente igual a 1,23 para o AMG, 1,06 para o ichol e 1,01 para o GMG.

Tabela 7.41: Crescimento do tempo gasto na fase de *setup* do precondicionador.

Problema	**Dim.**	**Raio**	ichol	AMG	GMG
cb48x16x16	12288	1,2	1,25	14,37	0,37
cb60x20x20	24000	1,5	2,64	32,43	-
cb72x24x24	41472	1,8	4,93	56,46	1,19
cb90x30x30	81000	2,25	9,98	126,07	-
cb96x32x32	98304	2,4	13,19	161,57	3,16
cb120x40x40	192000	3,0	26,20	353,78	5,84
cb144x48x48	331776	3,6	45,22	736,75	10,44
cb150x50x50	375000	3,75	51,72	868,05	-
cb168x56x56	526848	4,2	75,02	1407,93	16,37
cb180x60x60	648000	4,5	81,67	1847,60	-
cb192x64x64	786432	4,8	100,58	2491,92	23,37



(a) Escala linear.

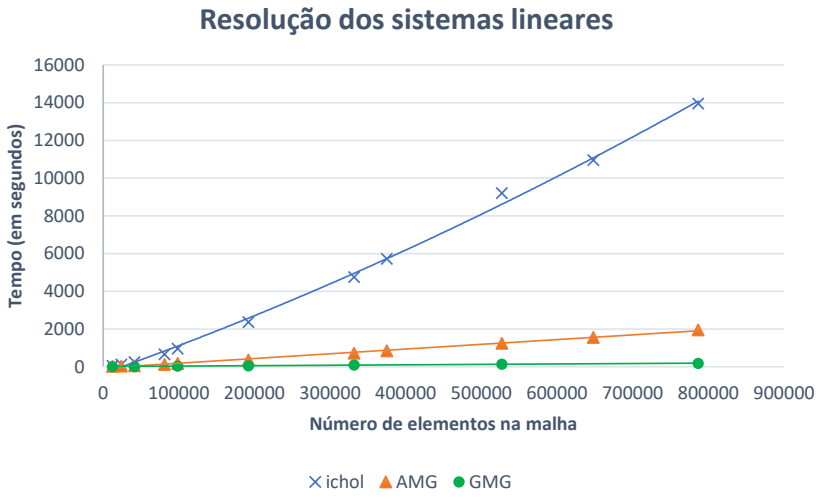


(b) Escala logarítmica.

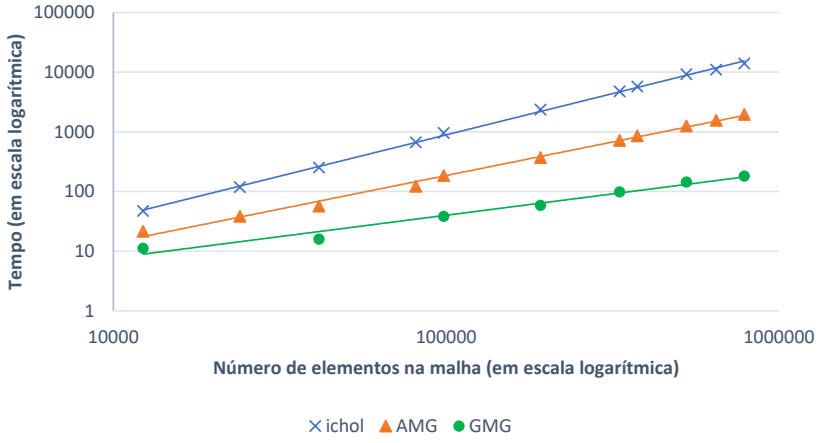
Figura 7.28: Crescimento do tempo gasto na fase de *setup* do precondicionador.

Tabela 7.42: Crescimento do tempo gasto para resolver os sistemas lineares.

Problema	**Dim.**	**Raio**	ichol	AMG	GMG
cb48x16x16	12288	1,2	47,38	21,40	11,21
cb60x20x20	24000	1,5	118,68	38,23	-
cb72x24x24	41472	1,8	252,24	57,04	15,93
cb90x30x30	81000	2,25	665,34	122,71	-
cb96x32x32	98304	2,4	962,17	184,41	38,48
cb120x40x40	192000	3,0	2367,17	370,14	58,74
cb144x48x48	331776	3,6	4755,66	713,11	98,64
cb150x50x50	375000	3,75	5722,97	852,67	-
cb168x56x56	526848	4,2	9201,93	1253,01	143,99
cb180x60x60	648000	4,5	10953,06	1557,87	-
cb192x64x64	786432	4,8	13956,77	1963,69	180,99



(a) Escala linear.



(b) Escala logarítmica.

Figura 7.29: Crescimento do tempo gasto para resolver os sistemas lineares.

No caso da resolução dos sistemas lineares, a tendência de crescimento do tempo (Figura 7.29(a)) aparenta ser quadrática para o ichol e linear para o AMG e o GMG, com valor de R^2 igual a 0,9976, 0,9969 e 0,9836, respectivamente. O consumo de tempo nesta etapa é maior para a fatoração incompleta de Cholesky e menor para o GMG. Apresentamos também os gráficos em escala logarítmica na Figura 7.29(b). Aproximando os pontos por curvas de tendência do tipo potência, o expoente obtido é aproximadamente igual a 1,38 para o ichol, 1,12 para o AMG e 0,72 para o GMG, reforçando que o *multigrid* é mais adequado para problemas de grande porte.

Os cálculos do gradiente possuem uma tendência de crescimento do tempo aproximadamente linear (Figura 7.30), com valor de R^2 igual a 0,9978 no caso do AMG. Essa etapa consome pouco tempo em relação ao total.

Tabela 7.43: Crescimento do tempo gasto no cálculo dos gradientes.

Problema	**Dim.**	**Raio**	ichol	AMG	GMG
`cb48x16x16`	12288	1,2	0,23	0,22	0,22
`cb60x20x20`	24000	1,5	0,41	0,42	-
`cb72x24x24`	41472	1,8	0,70	0,69	0,71
`cb90x30x30`	81000	2,25	1,38	1,38	-
`cb96x32x32`	98304	2,4	1,74	1,77	1,75
`cb120x40x40`	192000	3,0	3,34	3,35	3,34
`cb144x48x48`	331776	3,6	5,86	5,89	5,85
`cb150x50x50`	375000	3,75	6,55	6,54	-
`cb168x56x56`	526848	4,2	9,13	9,22	9,25
`cb180x60x60`	648000	4,5	11,73	11,59	-
`cb192x64x64`	786432	4,8	14,92	14,87	14,81

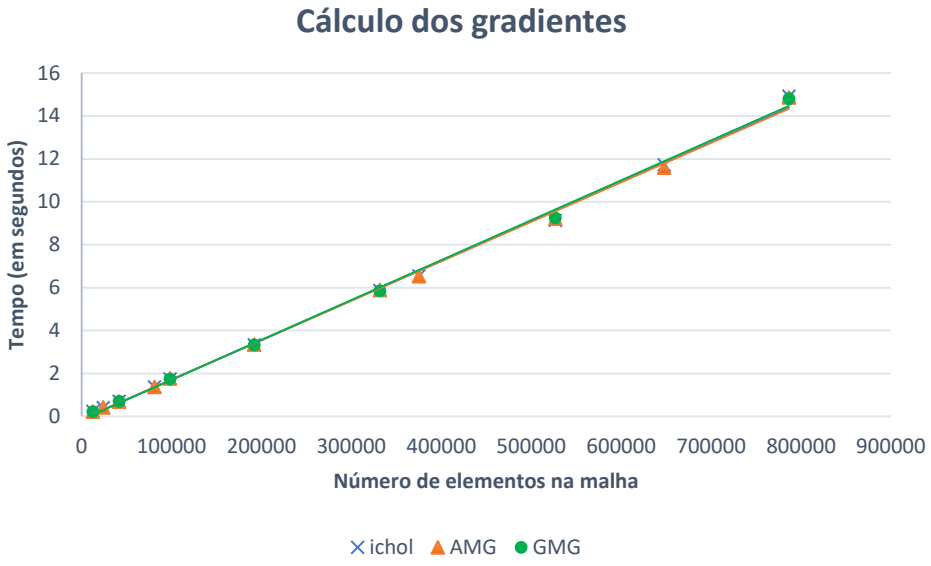


Figura 7.30: Crescimento do tempo gasto no cálculo dos gradientes.

Vimos que a resolução dos subproblemas de programação linear é uma das etapas mais caras do algoritmo, se tornando a mais cara de todas quando utilizamos o

multigrid geométrico. Além disso, pela Figura 7.31, notamos que o crescimento do tempo dessa etapa é melhor aproximado por uma curva de tendência quadrática, com valor de R^2 igual a 0,9943 para o AMG, por exemplo. O termo quadrático da curva de tendência é muito pequeno, o que nos dá a falsa impressão de que a curva seja praticamente linear. Contudo, ao aproximarmos os pontos por uma reta, notamos que a maioria deles fica relativamente distante da curva.

Tabela 7.44: Crescimento do tempo gasto na resolução dos subproblemas de PL.

Problema	**Dim.**	**Raio**	ichol	AMG	GMG
`cb48x16x16`	12288	1,2	3,55	2,88	2,86
`cb60x20x20`	24000	1,5	10,40	10,48	-
`cb72x24x24`	41472	1,8	31,42	31,31	31,34
`cb90x30x30`	81000	2,25	120,60	120,45	-
`cb96x32x32`	98304	2,4	191,26	191,60	191,16
`cb120x40x40`	192000	3,0	656,72	663,45	655,53
`cb144x48x48`	331776	3,6	1580,57	1585,56	1584,07
`cb150x50x50`	375000	3,75	1917,64	1901,46	-
`cb168x56x56`	526848	4,2	2868,39	2863,25	2881,37
`cb180x60x60`	648000	4,5	3440,88	3421,72	-
`cb192x64x64`	786432	4,8	4067,99	4069,07	4078,29

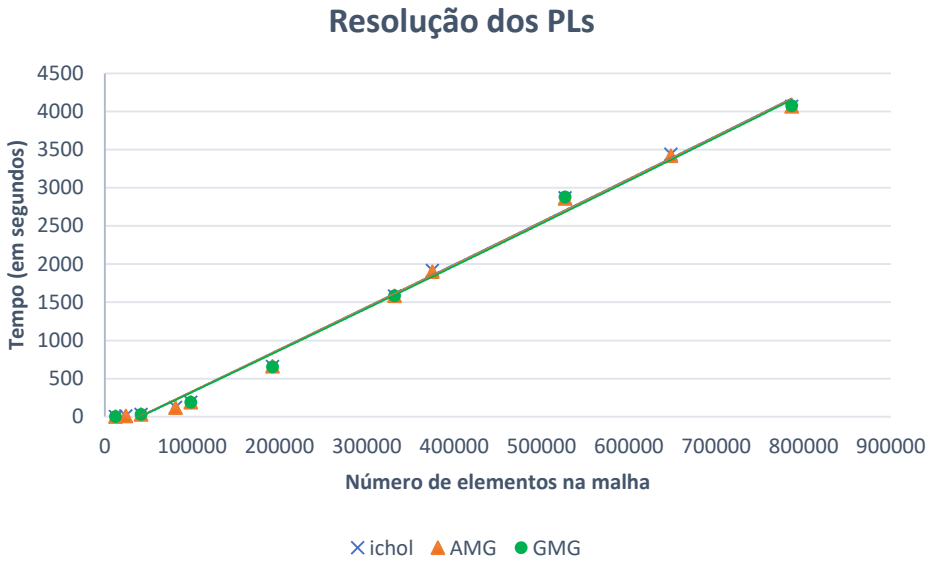


Figura 7.31: Crescimento do tempo gasto na resolução dos subproblemas de PL.

Em geral, o crescimento do tempo total de resolução dos problemas em função do número de elementos na malha segue uma tendência quadrática (Figura 7.32). A utilização do *multigrid* como precondicionador reduz significativamente o tempo total, sendo o GMG mais rápido do que o AMG. Para reduzir ainda mais os tempos da nossa implementação, podemos tentar acelerar a fase de *setup* do *multigrid* algébrico e, principalmente, voltar nossa atenção para a resolução dos subproblemas de programação linear, que se torna a etapa principal em termos de consumo de tempo.

Tabela 7.45: Crescimento do tempo total em função da dimensão do problema.

Problema	**Dim.**	**Raio**	ichol	AMG	GMG
cb48x16x16	12288	1,2	54,95	41,46	17,29
cb60x20x20	24000	1,5	137,51	86,90	-
cb72x24x24	41472	1,8	301,81	157,87	61,56
cb90x30x30	81000	2,25	824,42	397,93	-
cb96x32x32	98304	2,4	1203,69	574,52	269,83
cb120x40x40	192000	3,0	3129,73	1467,16	799,73
cb144x48x48	331776	3,6	6529,45	3182,05	1841,20
cb150x50x50	375000	3,75	7869,93	3801,31	-
cb168x56x56	526848	4,2	12419,23	5795,98	3314,95
cb180x60x60	648000	4,5	14821,27	7173,56	-
cb192x64x64	786432	4,8	18563,99	8961,35	4721,90

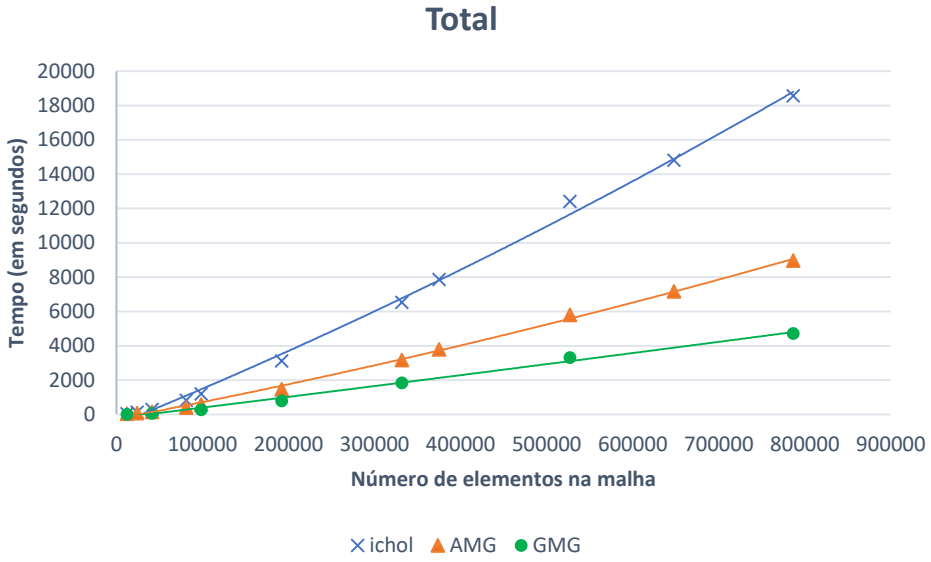


Figura 7.32: Crescimento do tempo total em função da dimensão do problema.

7.5 Simetrias das estruturas

Apesar do *multigrid* ser muito eficiente na resolução dos sistemas lineares de equilíbrio e, consequentemente, em reduzir o tempo total de resolução dos problemas de otimização topológica, a problemática do consumo de memória computacional continua ativa. Precisamos armazenar a matriz de rigidez global e, no caso do *multigrid*, também temos que armazenar matrizes em outras malhas e os operadores de transferência entre as malhas. Quanto maior for o número de elementos na malha, maior é a quantidade de entradas não nulas dessas matrizes, havendo um ponto a partir do qual não conseguimos mais refinar a malha por falta de memória computacional para resolver o problema.

Felizmente, em muitos casos, a estrutura possui simetrias que podemos explorar para reduzir o tempo e a memória consumidos. Embora na otimização topológica não saibamos qual é o formato da estrutura, podemos deduzir que ela seja simétrica por algum

plano que passa pelo domínio, desde que os apoios e as cargas externas sejam simétricos com relação a esse plano.

Assim, consideramos que as densidades de material em elementos simétricos devem ser iguais e, consequentemente, só precisamos obter uma delas. Com isso, reduzimos a quantidade de variáveis do problema e as dimensões das matrizes de rigidez.

Para levar em conta a simetria da estrutura, precisamos adaptar o domínio do problema. Primeiro, reduzimos a região do domínio, considerando somente a parte desejada e mantendo os apoios e as cargas nos pontos originais. Depois, acrescentamos novos apoios nos planos de simetria, impedindo os deslocamentos nas direções ortogonais a esses planos. Por exemplo, a viga em balanço possui uma simetria por um plano paralelo ao plano xy passando pela metade da largura do domínio. Nesse caso, cortamos o domínio por esse plano, conforme a Figura 7.33, e acrescentamos apoios por toda a nova face (destacada em verde, com marcador "◁"), impedindo o deslocamento na direção z.

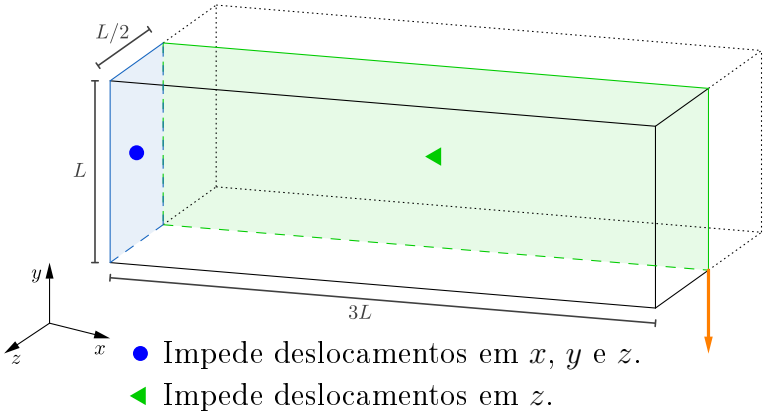


Figura 7.33: Domínio inicial do problema da viga em balanço com simetria.

A viga em balanço e a viga em L (Exemplos 1 e 6 da Seção 7.2) possuem simetria com relação a um plano paralelo ao plano xy. Já a viga MBB, a torre de transmissão, o meio cubo e a ponte (Exemplos 2, 3, 4 e 7 da Seção 7.2) possuem dois planos de simetria, um paralelo ao plano xy e outro paralelo ao plano yz, de modo que podemos trabalhar apenas com um quarto dessas estruturas. O prédio com torção (Exemplo 5) é o único problema deste trabalho em que não podemos considerar nenhuma simetria, por causa da disposição das cargas no domínio.

Testamos a resolução dos problemas da viga em balanço, da viga MBB e da torre de transmissão considerando as simetrias das estruturas. Na Tabela 7.46, comparamos os resultados obtidos com aqueles provenientes da resolução dos problemas completos (sem considerar as simetrias). Utilizamos o *multigrid* geométrico como precondicionador para a resolução dos sistemas lineares e o mesmo raio do filtro, r, para resolver o problema com ou sem as simetrias. As Figuras 7.34, 7.35 e 7.36 mostram as estruturas ótimas obtidas, nas quais espelhamos as densidades de material em relação aos planos de simetria para obter o formato completo das estruturas.

Na Tabela 7.46, a coluna "Dim." mostra o número de elementos na malha mais fina para cada problema. Considerando as simetrias, há uma redução significativa na dimensão, principalmente nos casos da viga MBB e da torre, em que resolvemos apenas um quarto das estruturas. A tabela também mostra o número de iterações para a convergência da PLS (N_{it}) e o número de passos recusados, entre parênteses. Em todos os problemas

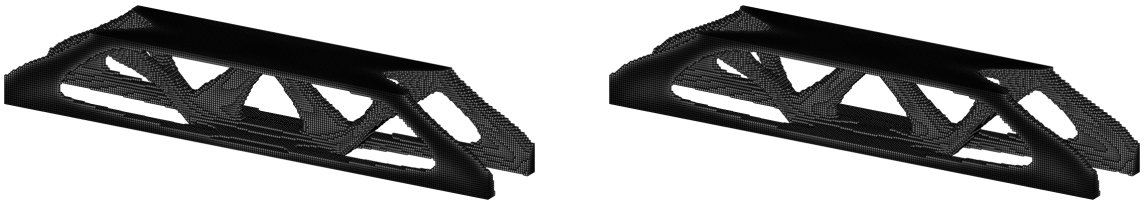
testados, o número de iterações foi maior quando consideramos as simetrias. Apesar disso, o tempo total de resolução é reduzido, além do consumo de memória ser menor, de modo que agora somos capazes de resolver problemas com resoluções maiores.

Tabela 7.46: Resultados para os problemas considerando as simetrias das estruturas.

Problema	**Simetrias**	**Dim.**	N_{it}	F	T_{total}
`cb192x64x64`	Não	786432	21 (+3)	10,8924	5997,65
	Sim	393216	25 (+1)	21,5657	3920,47
`mbb240x40x40`	Não	384000	23 (+2)	9,9553	2864,92
	Sim	96000	36 (+4)	39,5974	667,30
`tt40x200x40`	Não	320000	23 (+0)	27,9789	1648,57
	Sim	80000	52 (+5)	70,6786	414,28

(a) Sem simetrias (b) Com simetrias

Figura 7.34: Estruturas ótimas obtidas para o problema `cb192x64x64` ($r = 5{,}2$).



(a) Sem simetrias (b) Com simetrias

Figura 7.35: Estruturas ótimas obtidas para o problema `mbb240x40x40` ($r = 5{,}5$).

(a) Sem simetrias (b) Com simetrias

Figura 7.36: Estruturas ótimas obtidas para o problema `tt40x200x40` ($r = 2{,}2$).

O valor da função objetivo (F) aumenta quando consideramos as simetrias porque estamos aplicando as forças externas com as mesmas intensidades em apenas metade ou um quarto da estrutura, de modo que os deslocamentos calculados são maiores, fazendo com que a estrutura tenha uma flexibilidade maior. Para melhorar a comparação entre os valores da função objetivo, a partir da solução obtida com as simetrias, montamos um vetor de densidades para a estrutura completa, levando em conta que as densidades em elementos simétricos são iguais. Com esse novo vetor de densidades, calculamos o valor da função objetivo para o problema na malha sem simetrias. Nesse caso, o valor obtido para o problema `cb192x64x64` foi $F = 10{,}7828$, para o problema `mbb240x40x40` foi $F = 9{,}8982$ e para o problema `tt40x200x40` foi $F = 27{,}8052$. Esses valores estão mais próximos e são menores do que aqueles obtidos com a solução dos problemas sem considerar as simetrias.

Além disso, lembramos que os problemas de otimização topológica possuem muitos mínimos locais, de modo que a solução encontrada em cada caso pode variar. Todavia, os formatos obtidos para as estruturas com simetrias são bastante parecidos com aqueles sem considerar as simetrias e, em muitos casos, ficam até melhores, uma vez que são perfeitamente simétricos.

A seguir, apresentamos resultados utilizando a estratégia de arredondamento das densidades, que permite obter estruturas com poucas densidades intermediárias. A partir daqui, sempre que possível, utilizaremos o *multigrid* geométrico como precondicionador para resolver os sistemas de equilíbrio e exploraremos as simetrias das estruturas.

7.6 Arredondamento de densidades

Nos resultados apresentados anteriormente, o vetor de densidades obtido como solução possui várias componentes com valores no intervalo $(0, 1)$, embora isso não seja tão perceptível nas figuras com as estruturas obtidas, uma vez que estamos exibindo apenas os elementos com densidades maiores que 0,5, com o intuito de esconder os elementos brancos (ou cinza claro) e facilitar a visualização das estruturas tridimensionais. Contudo, a presença de densidades intermediárias pode dificultar a produção da estrutura obtida. Além disso, não sabemos o quanto as densidades com valores pequenos (menores que 0,5), que sequer aparecem na figura, contribuem para reduzir a flexibilidade da estrutura.

Conforme vimos, o valor da função objetivo do problema de otimização topológica se altera quando mudamos as condições nas quais resolvemos o problema (por exemplo, o refinamento da malha, a consideração de simetrias, o raio do filtro e os valores dos parâmetros do algoritmo), devido à presença de muitos mínimos locais. De acordo com Sigmund [46], para uma comparação mais efetiva dos resultados com métodos diferentes, devemos utilizar sempre os mesmos parâmetros, calcular o valor da função objetivo considerando o mesmo refinamento e utilizar condições de contorno independentes da malha. Ademais, o autor sugere que o ideal é arredondar as densidades para 0 ou 1, respeitando a restrição de volume, e recalcular o valor da função objetivo, pois assim eliminamos a influência das densidades intermediárias no valor da flexibilidade. Também é mais adequado utilizar a solução arredondada para comparar as figuras das estruturas obtidas.

Outra boa prática é fornecer o valor da função objetivo referente à estrutura totalmente sólida, isto é, com todas as densidades iguais a 1, ocupando 100% do volume do domínio. Esse é um caso extremo, no qual a estrutura é a mais rígida possível, formada

por um bloco totalmente sólido. Quanto menor for a fração de volume que a estrutura pode ocupar no domínio, maior será o valor ótimo da função objetivo. Assim, o valor da flexibilidade para a estrutura totalmente sólida nos fornece uma informação adicional sobre quão boa é a solução obtida.

Em [46], o autor utiliza uma técnica de arredondamento simples, que consiste em atribuir o valor 1 para as maiores densidades, até completar o volume desejado, e atribuir 0 para as demais densidades. Entretanto, ele sugere que é ainda melhor comparar os resultados utilizando uma estratégia de arredondamento mais elaborada, que envolva, por exemplo, a projeção de Heaviside. Neste trabalho, propomos uma nova estratégia de arredondamento, explicada na Seção 6.1, que leva em consideração as informações do vetor gradiente do Lagrangiano do problema.

Nesta seção, apresentamos resultados obtidos com essa estratégia de arredondamento. Os valores utilizados para os parâmetros do algoritmo foram apresentados na Seção 6.1. Comparamos as soluções obtidas sem arredondamento, com o arredondamento simples, com o novo arredondamento e com a estrutura totalmente sólida.

A Tabela 7.47 apresenta os valores ótimos da função objetivo (flexibilidade) calculados em cada caso, para os problemas da viga em balanço, da viga MBB e da torre de transmissão. Nesta tabela, utilizamos a seguinte notação:

- F: valor da flexibilidade para a solução original, sem arredondar as densidades.
- F_{rnd}: valor da flexibilidade para a solução obtida com o arredondamento simples.
- F_{prj}: valor da flexibilidade para a solução obtida com o novo arredondamento de densidades, baseado em projeção.
- F_{sol}: valor da flexibilidade para a estrutura totalmente sólida.

Tabela 7.47: Valores da flexibilidade obtidos com o arredondamento de densidades.

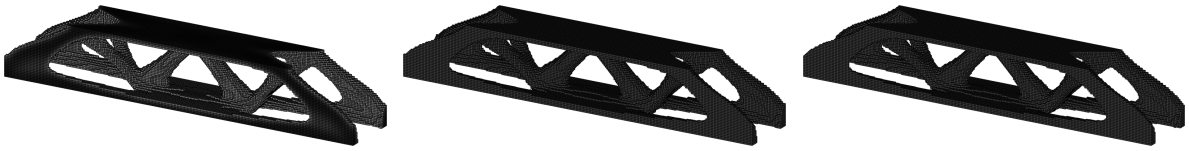
Problema	F	F_{rnd}	F_{prj}	F_{sol}
`cb192x64x64`	21,5657	18,5492	18,5464	11,1078
`mbb240x40x40`	39,5974	26,7998	26,7618	13,2849
`tt40x200x40`	70,6786	64,4782	64,4782	42,0986

Notamos que, em todos os problemas testados, o valor da função objetivo para a estrutura original (F) foi o maior, indicando que as densidades intermediárias influenciam no cálculo da flexibilidade, e o valor da função para a estrutura totalmente sólida (F_{sol}) foi sempre o menor, pois essa é a estrutura menos flexível. Além disso, nos problemas da viga em balanço e da viga MBB, o valor obtido para a estrutura com o novo arredondamento de densidades (F_{prj}) foi menor do que aquele encontrado para a estrutura com o arredondamento simples (F_{rnd}), indicando que a nossa estratégia é capaz de encontrar soluções mais próximas de um minimizador local. Já no caso da torre de transmissão, esses valores foram iguais, ao menos com 4 casas decimais de precisão.

As Figuras 7.37, 7.38 e 7.39 mostram as estruturas obtidas com as densidades originais (à esquerda), com as densidades arredondadas de forma simples (no meio) e com as densidades arredondadas pela nossa estratégia (à direita). Notamos que não há diferenças visíveis nos formatos das estruturas.

(a) Densidades originais (b) Arredondamento simples (c) Novo arredondamento

Figura 7.37: Estruturas ótimas obtidas para o problema `cb192x64x64` com o arredondamento de densidades ($r = 5{,}2$).



(a) Densidades originais (b) Arredondamento simples (c) Novo arredondamento

Figura 7.38: Estruturas ótimas obtidas para o problema `mbb240x40x40` com o arredondamento de densidades ($r = 5{,}5$).

(a) Densidades originais (b) Arredondamento simples (c) Novo arredondamento

Figura 7.39: Estruturas ótimas obtidas para o problema `tt40x200x40` com o arredondamento de densidades ($r = 2{,}2$).

Em nossa estratégia de arredondamento, controlamos a qualidade da solução obtida aplicando novamente o algoritmo da programação linear sequencial, usando a solução arredondada como vetor de densidades inicial. Repetimos esse processo até que o número máximo de 10 tentativas de arredondamento seja atingido ou até que a diferença entre duas soluções arredondadas consecutivas seja relativamente pequena e a violação na restrição de volume esteja dentro de uma tolerância. Nos três problemas apresentados aqui, foi necessária apenas 1 tentativa de arredondamento para atingir o critério de parada mencionado. Na Tabela 7.48, apresentamos o número de iterações da PLS (N_{it}), com o número de passos recusados (N_{rc}) entre parênteses, o tempo (em segundos) gasto para arredondar as densidades (T_{prj}) e o tempo total gasto para resolver o problema (T_{total}).

As chamadas da PLS após o arredondamento, em geral, gastam poucas iterações, uma vez que a solução arredondada já está próxima de um minimizador local. Muitas vezes, é necessário apenas realizar as 3 iterações consecutivas nas quais a norma do gradiente projetado é menor que a tolerância estabelecida no critério de parada da PLS.

Portanto, o tempo total de resolução não cresce demasiadamente com as novas chamadas da PLS, desde que não sejam feitas muitas tentativas de arredondamento. Ademais, o tempo gasto para arredondar as densidades cresce conforme a quantidade de elementos na malha aumenta, mas ele é praticamente insignificante em relação ao tempo total de resolução. O valor T_{prj} é um pouco maior após o arredondamento, pois é nesse momento que aplicamos a projeção de Heaviside.

Tabela 7.48: Resultados obtidos com o arredondamento de densidades.

Problema	**Chamada da PLS**	N_{it} (N_{rc})	T_{prj}	T_{total}
`cb192x64x64`	Solução inicial	25 (+1)	0,01	3992,92
	Após o arredondamento	3 (+0)	0,77	169,27
`mbb240x40x40`	Solução inicial	36 (+4)	0,01	681,04
	Após o arredondamento	7 (+0)	0,18	51,48
`tt40x200x40`	Solução inicial	52 (+5)	0,01	422,92
	Após o arredondamento	3 (+0)	0,16	17,36

Na Tabela 7.49, mostramos as quantidades de elementos vazios (com densidade $\rho = 0$), de elementos intermediários (com densidade $\rho \in (0, 1)$) e de elementos sólidos (com densidade $\rho = 1$), para as soluções original e arredondada de cada problema, considerando apenas a parte simétrica das estruturas. Notamos que as soluções originais possuem uma quantidade significativa de elementos intermediários, especialmente no problema `mbb240x40x40`, o que pode justificar a diferença entre os valores F e F_{prj} na Tabela 7.47. O arredondamento de densidades foi capaz de eliminar completamente os elementos intermediários nos problemas da viga MBB e da torre, mantendo o volume da estrutura em 20% e 15% do total, respectivamente. Já no problema `cb192x64x64`, restaram 54 elementos intermediários, que representam apenas 0,014% do total de elementos. Entretanto, o volume da estrutura foi mantido em 20% do total.

Tabela 7.49: Distribuição das densidades de material no domínio.

Problema	**Solução**	$\rho = 0$	$\rho \in (0, 1)$	$\rho = 1$
`cb192x64x64`	Original	218982	153938	20296
	Arredondada	314545	54	78617
`mbb240x40x40`	Original	43972	51765	263
	Arredondada	76800	0	19200
`tt40x200x40`	Original	59328	15694	4978
	Arredondada	68000	0	12000

A estratégia de arredondamento de densidades proposta se mostrou bastante eficaz, sendo capaz de reduzir significativamente a quantidade de elementos com densidades intermediárias e obter soluções com valores da flexibilidade menores, sem violar a restrição de volume, com um custo computacional relativamente baixo.

Na próxima seção, apresentaremos os resultados obtidos com a técnica de multirresolução, que permite obter estruturas com resoluções altas em um tempo menor do que o método tradicional. Utilizaremos o arredondamento de densidades para comparar os métodos de maneira mais efetiva.

7.7 Multirresolução

Nosso objetivo é encontrar estruturas cada vez mais detalhadas e com contornos mais suaves, o que exige que utilizemos malhas com uma quantidade grande de elementos. Com o intuito de tornar o algoritmo de resolução mais eficiente, além de utilizar a programação linear sequencial para resolver o problema de otimização, explorar as simetrias das estruturas e aplicar o método *multigrid* na resolução dos sistemas lineares de equilíbrio, neste trabalho combinamos essas estratégias com a multirresolução, descrita no Capítulo 5. Nesta seção, apresentamos resultados obtidos com a implementação dessa técnica para resolver problemas tridimensionais de otimização topológica estrutural.

7.7.1 Comparações com o método tradicional

Inicialmente, comparamos as soluções obtidas com a multirresolução e com a otimização topológica tradicional. Na multirresolução, as malhas de densidades e de variáveis de projeto são obtidas a partir da malha de deslocamentos, dividindo cada elemento maior em $(n_{mr})^3$ elementos de densidade e $(d_{mr})^3$ variáveis de projeto. Testamos algumas combinações de valores para os parâmetros n_{mr} e d_{mr}.

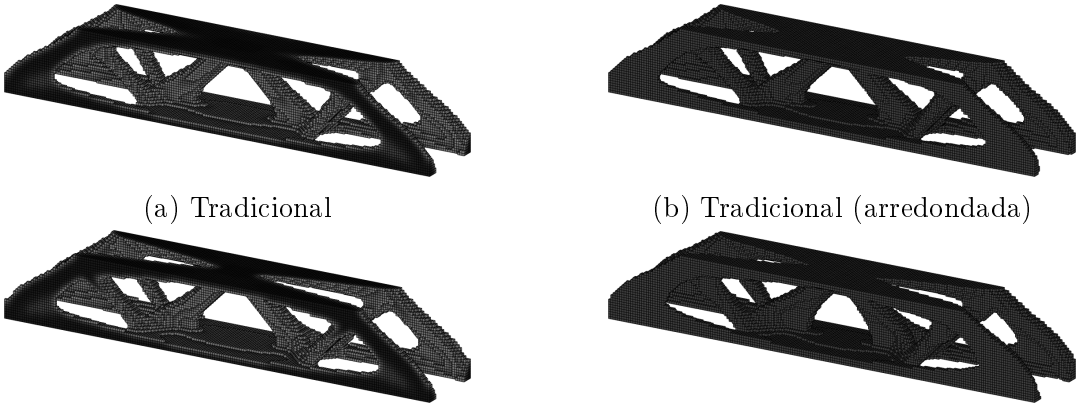
Consideramos que a malha de deslocamentos possui elementos que medem 1 unidade de comprimento em cada direção. Por sua vez, os elementos de densidade são menores e medem $1/n_{mr}$ unidade de comprimento em cada direção. Neste caso, podemos utilizar raios menores para o filtro (usado para a projeção das variáveis de projeto no espaço das densidades), em comparação com os raios utilizados no problema tradicional. Nos resultados desta subseção, para a multirresolução escolhemos um raio, r_{min}, e tentamos mantê-lo para todos os valores de n_{mr}. Já para as estruturas de referência, utilizamos um raio proporcional, isto é, igual a $n_{mr}r_{min}$.

Ademais, consideramos que as regiões de aplicação dos apoios em cada método são iguais. Por exemplo, se há um apoio que restringe os deslocamentos dos pontos da face de um elemento do problema base, na multirresolução com $n_{mr} = 2$ esse apoio passa a estar nas faces de 4 elementos de densidade. Além disso, no método tradicional com o dobro da resolução também colocamos os apoios na mesma região que engloba as 4 faces desses elementos. Com isso, a comparação entre os métodos fica mais precisa.

Observamos, contudo, que os cenários nos quais resolvemos o problema com a multirresolução e com o método tradicional ainda são diferentes, isto é, alguns parâmetros do algoritmo se alteram, como o tamanho dos elementos de densidade, o valor do raio do filtro, a quantidade de nós e de elementos na malha onde resolvemos o sistema linear etc. Como os problemas de otimização topológica possuem muitos mínimos locais próximos, naturalmente pode haver diferenças nos resultados numéricos e nos formatos das estruturas obtidas com os diferentes métodos. Para uma análise mais objetiva, apresentamos também as estruturas obtidas com a estratégia de arredondamento de densidades.

No problema da viga MBB, tomamos o refinamento com `96x16x16` elementos de deslocamento como base e obtivemos a viga com as resoluções contendo `192x32x32`, `288x48x48` e `384x64x64` elementos de densidade, utilizando n_{mr} igual a 2, 3 e 4, respectivamente. Também variamos os possíveis valores de d_{mr}, com $1 < d_{mr} \leq n_{mr}$. O raio do filtro utilizado foi $r_{min} = 2{,}0$. As Figuras 7.40, 7.41 e 7.42 mostram as estruturas ótimas obtidas com o método tradicional e com a multirresolução (MR).

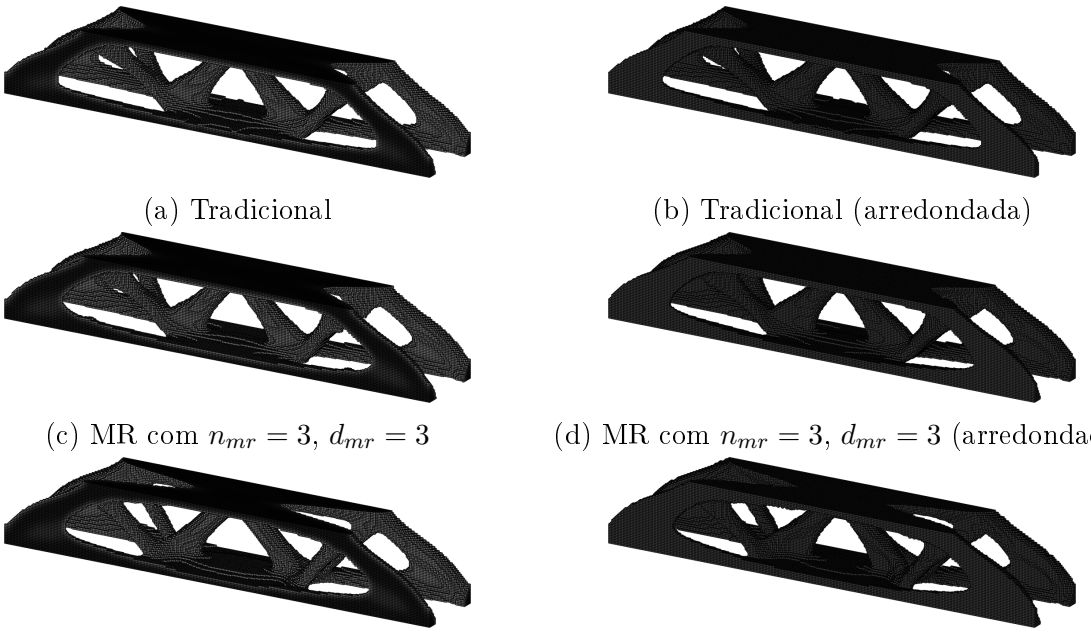
Em geral, o formato da estrutura com a multirresolução ficou parecido com o que encontramos no método tradicional, havendo apenas alguns detalhes diferentes. Na estrutura obtida com a multirresolução, $n_{mr} = 4$ e $d_{mr} = 3$, com o arredondamento de densidades podemos notar a presença de barras que não estão nas demais vigas encontradas. Na verdade, essas barras também estão "escondidas" nas estruturas com $n_{mr} = 3$, $d_{mr} = 3$ e com $n_{mr} = 4$, $d_{mr} = 3$ sem o arredondamento, pois os elementos nas novas barras possuem densidades próximas de 0,5, que não aparecem na figura. Aplicando o arredondamento simples, isto é, atribuindo 1 para as maiores densidades até completar o volume desejado e 0 para as demais, podemos ver que as barras aparecem incompletas na estrutura, como mostra a Figura 7.43, na qual exibimos a visão lateral das vigas. O uso da nossa estratégia de arredondamento de densidades foi capaz de eliminar os artefatos presentes na solução. Com $n_{mr} = 3$, as barras adicionais foram eliminadas, já com $n_{mr} = 4$ elas foram totalmente preenchidas.



(a) Tradicional (b) Tradicional (arredondada)

(c) MR com $n_{mr} = 2$, $d_{mr} = 2$ (d) MR com $n_{mr} = 2$, $d_{mr} = 2$ (arredondada)

Figura 7.40: Estruturas obtidas para a viga MBB com resolução `192x32x32` ($r_{min} = 2{,}0$).

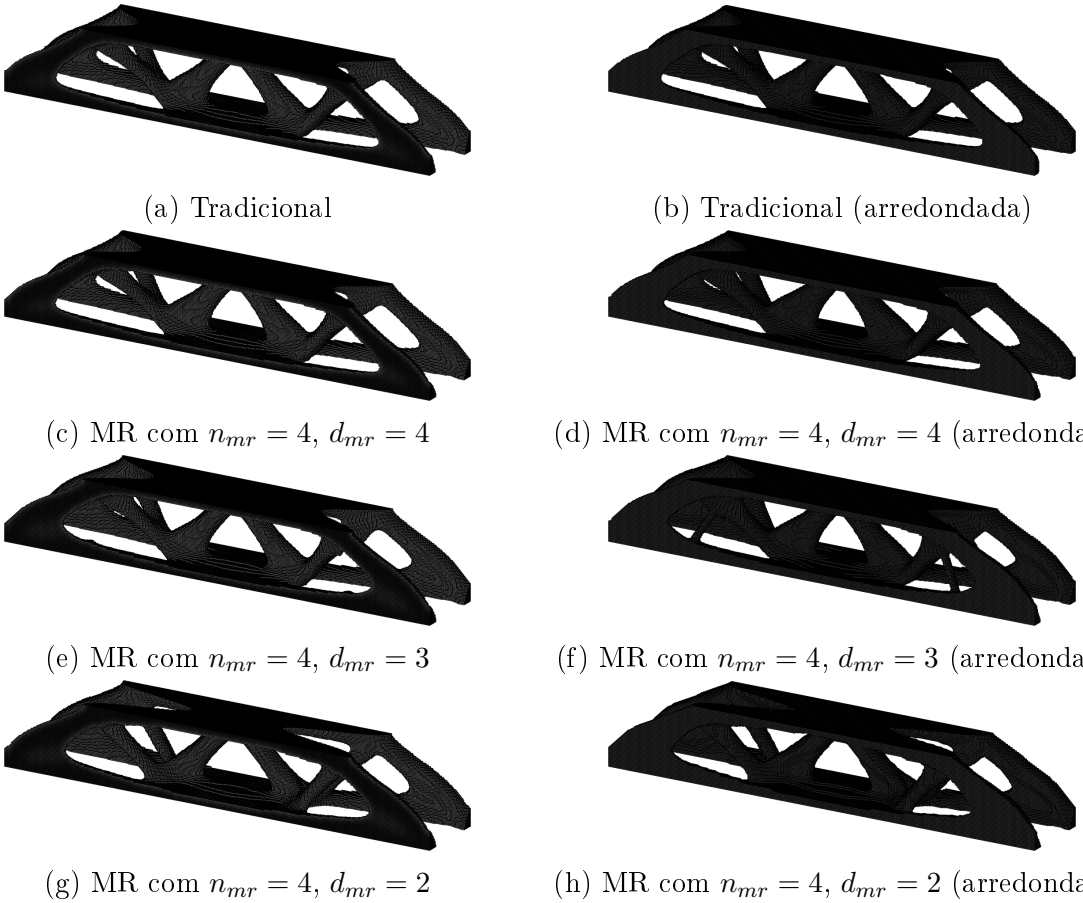


(a) Tradicional (b) Tradicional (arredondada)

(c) MR com $n_{mr} = 3$, $d_{mr} = 3$ (d) MR com $n_{mr} = 3$, $d_{mr} = 3$ (arredondada)

(e) MR com $n_{mr} = 3$, $d_{mr} = 2$ (f) MR com $n_{mr} = 3$, $d_{mr} = 2$ (arredondada)

Figura 7.41: Estruturas obtidas para a viga MBB com resolução `288x48x48` ($r_{min} = 2{,}0$).



(a) Tradicional

(b) Tradicional (arredondada)

(c) MR com $n_{mr} = 4$, $d_{mr} = 4$

(d) MR com $n_{mr} = 4$, $d_{mr} = 4$ (arredondada)

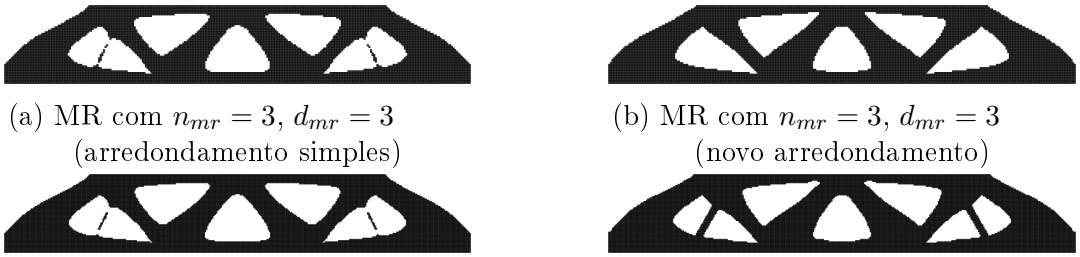
(e) MR com $n_{mr} = 4$, $d_{mr} = 3$

(f) MR com $n_{mr} = 4$, $d_{mr} = 3$ (arredondada)

(g) MR com $n_{mr} = 4$, $d_{mr} = 2$

(h) MR com $n_{mr} = 4$, $d_{mr} = 2$ (arredondada)

Figura 7.42: Estruturas obtidas para a viga MBB com resolução `384x64x64` ($r_{min} = 2{,}0$).



(a) MR com $n_{mr} = 3$, $d_{mr} = 3$ (arredondamento simples)

(b) MR com $n_{mr} = 3$, $d_{mr} = 3$ (novo arredondamento)

(c) MR com $n_{mr} = 4$, $d_{mr} = 3$ (arredondamento simples)

(d) MR com $n_{mr} = 4$, $d_{mr} = 3$ (novo arredondamento)

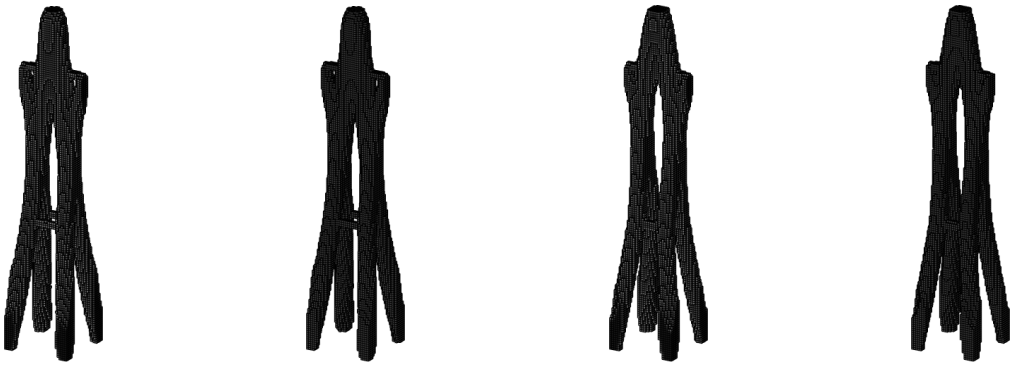
Figura 7.43: Visão lateral da viga MBB, nas quais aparecem artefatos ou barras adicionais.

Observamos que o uso do parâmetro d_{mr} com valor maior do que 2 ultrapassa o limitante superior imposto na Tabela 5.1, uma vez que estamos utilizando elementos finitos de grau 1. De acordo com Gupta *et al.* [27], isso pode causar a não unicidade do campo das variáveis de projeto, favorecendo o surgimento de artefatos na estrutura. Na Figura 7.43, notamos o aparecimento de artefatos ou barras adicionais na viga MBB quando utilizamos $d_{mr} = 3$. Todavia, em alguns casos conseguimos bons resultados, mesmo ultrapassando o limitante para d_{mr}, graças a aplicação do filtro de densidades.

Também resolvemos o problema da torre de transmissão considerando o refinamento com `16x80x16` elementos de deslocamento como base e obtivemos a estrutura com as resoluções contendo `32x160x32`, `48x240x48` e `64x320x64` elementos de densidade,

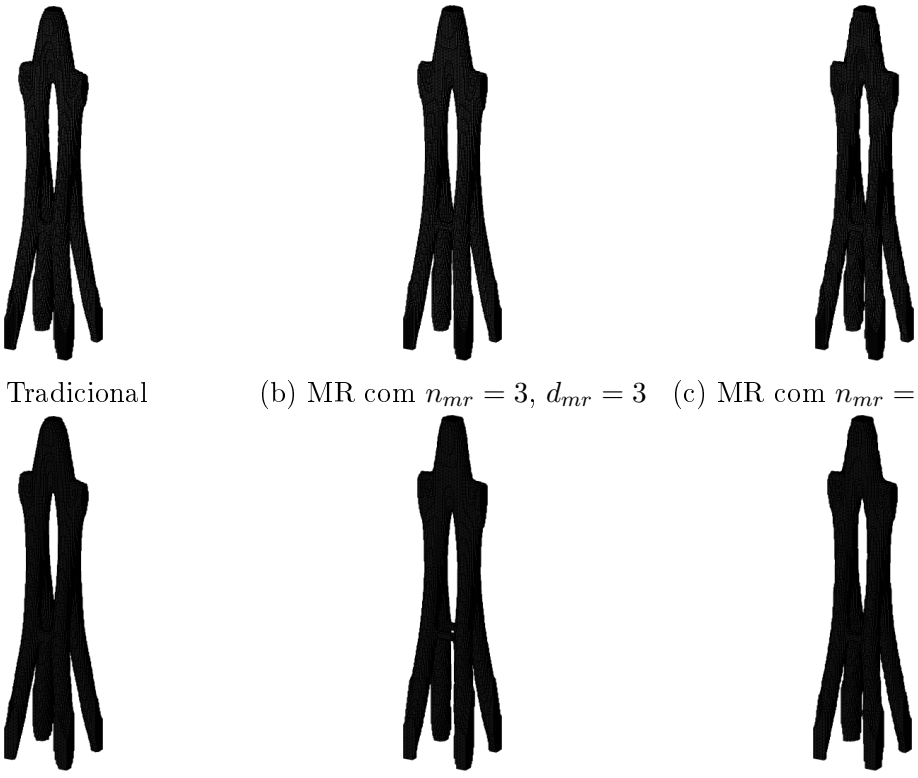
utilizando n_{mr} igual a 2, 3 e 4, respectivamente, e variando o parâmetro d_{mr}. O raio do filtro utilizado foi $r_{min} = 1{,}1$. As Figuras 7.44, 7.45 e 7.46 mostram as estruturas obtidas.

Neste caso, o formato da estrutura obtida com a multirresolução também ficou parecido com o que encontramos no método tradicional, havendo apenas pequenas diferenças. No caso da multirresolução com $n_{mr} = d_{mr} = 4$, a estrutura após o arredondamento de densidades apresentou um formato diferente, no qual as barras que conectam o centro da torre foram suprimidas e apareceram buracos na parte onde estão os braços que sustentam os cabos.



(a) Tradicional (b) Tradicional (arredondada) (c) MR com $n_{mr} = 2$, $d_{mr} = 2$ (d) MR com $n_{mr} = 2$, $d_{mr} = 2$ (arredondada)

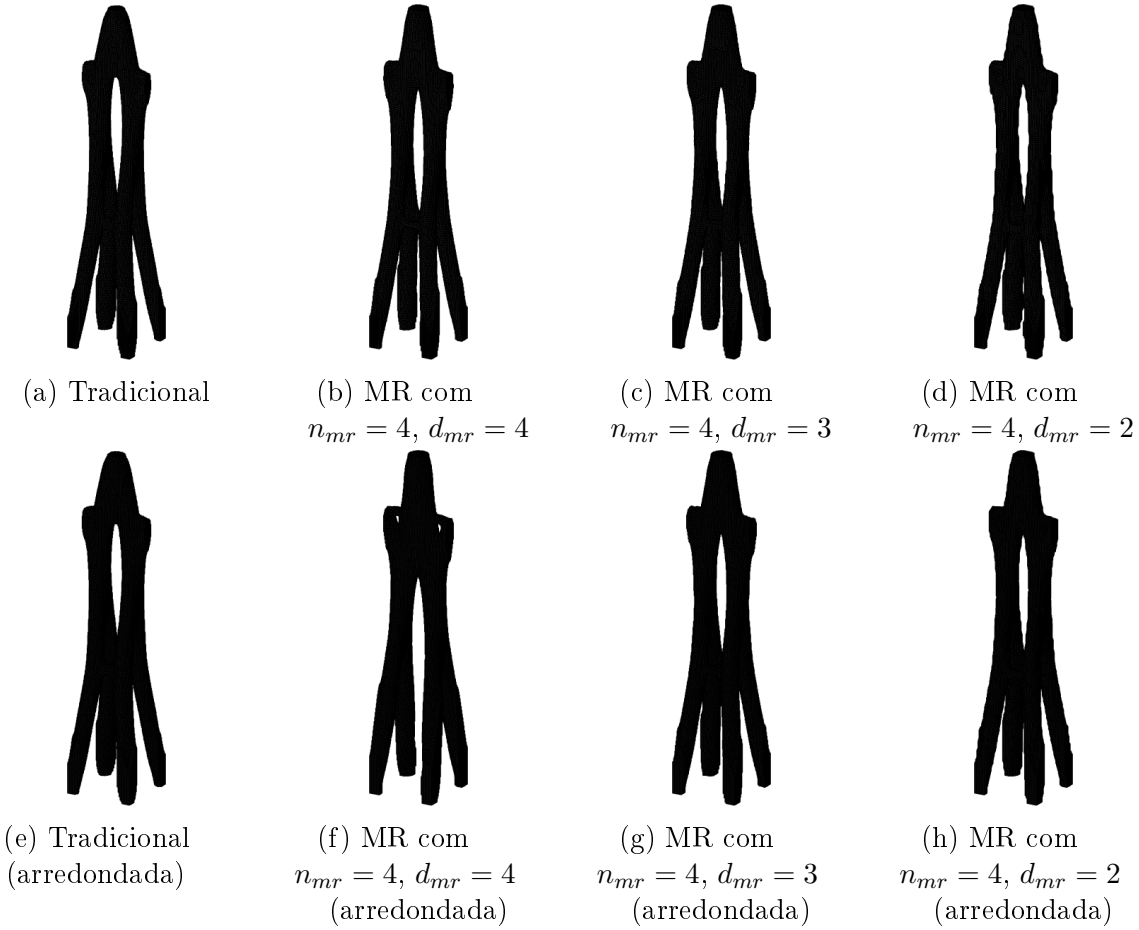
Figura 7.44: Estruturas obtidas para a torre com resolução `32x160x32` ($r_{min} = 1{,}1$).



(a) Tradicional (b) MR com $n_{mr} = 3$, $d_{mr} = 3$ (c) MR com $n_{mr} = 3$, $d_{mr} = 2$

(d) Tradicional (arredondada) (e) MR com $n_{mr} = 3$, $d_{mr} = 3$ (arredondada) (f) MR com $n_{mr} = 3$, $d_{mr} = 2$ (arredondada)

Figura 7.45: Estruturas obtidas para a torre com resolução `48x240x48` ($r_{min} = 1{,}1$).



(a) Tradicional

(b) MR com $n_{mr} = 4$, $d_{mr} = 4$

(c) MR com $n_{mr} = 4$, $d_{mr} = 3$

(d) MR com $n_{mr} = 4$, $d_{mr} = 2$

(e) Tradicional (arredondada)

(f) MR com $n_{mr} = 4$, $d_{mr} = 4$ (arredondada)

(g) MR com $n_{mr} = 4$, $d_{mr} = 3$ (arredondada)

(h) MR com $n_{mr} = 4$, $d_{mr} = 2$ (arredondada)

Figura 7.46: Estruturas obtidas para a torre com resolução `64x320x64` ($r_{min} = 1{,}1$).

Nas Tabelas 7.50 e 7.51 apresentamos alguns resultados numéricos para a viga MBB e a torre, respectivamente, comparando as soluções obtidas pelo método tradicional e pela multirresolução, variando os parâmetros n_{mr} e d_{mr}.

Para $n_{mr} = 2$, em ambos os problemas, o número de iterações da PLS (N_{it}) e o tempo total de resolução (T_{total}) diminuíram com a multirresolução. Para $n_{mr} = 3$, o número de iterações aumentou e, quando $d_{mr} = 3$, o tempo total acabou sendo maior do que com o método tradicional. A justificativa para isso é que, utilizando o *multigrid*, a maior parte do tempo gasto vem da resolução dos subproblemas de programação linear (PL), conforme vimos na Subseção 7.4.2. Logo, se $n_{mr} = d_{mr}$, os PL são resolvidos em uma malha com o mesmo refinamento do método tradicional e, uma vez que a multirresolução gastou mais iterações, o tempo total ficou maior. Diminuindo o valor de d_{mr}, reduzimos o tempo de resolução do problema de otimização e, consequentemente, o tempo total. Mesmo com um número de iterações consideravelmente maior, há uma redução significativa no tempo gasto quando $d_{mr} = 2$. Para $n_{mr} = 4$, as conclusões são similares.

Como já observamos acima, a resolução dos problemas com a multirresolução e com o método tradicional é feita em cenários diferentes, de modo que pode haver variações nos resultados numéricos e é difícil comparar as soluções. O valor da função objetivo calculado com a multirresolução, por exemplo, é bastante diferente daquele encontrado com o método tradicional. Para contornar esse problema, tentamos corrigir o valor da função utilizando o vetor de densidades ótimo encontrado com a multirresolução para

resolver o sistema de equilíbrio na mesma malha do método tradicional. Nas Tabelas 7.50 e 7.51, apresentamos os valores da função com a solução original (F) e com a solução arredondada (F_{prj}) corrigidos para a multirresolução. Os valores de F_{prj} foram levemente menores do que os do método tradicional nos problemas da viga MBB e levemente maiores nos problemas da torre. Entretanto, essa comparação não é perfeita porque os problemas de otimização resolvidos não são idênticos.

Tabela 7.50: Resultados para a viga MBB utilizando a multirresolução.

Problema	**Método**	N_{it} (N_{rc})	F	F_{prj}	T_{total}
`mbb192x32x32`	Tradicional	149 (+49)	42,2079	29,5838	1007,85
	MR ($n_{mr} = 2$, $d_{mr} = 2$)	99 (+28)	41,8844	29,0968	404,64
`mbb288x48x48`	Tradicional	31 (+1)	30,6031	21,8568	1310,43
	MR ($n_{mr} = 3$, $d_{mr} = 3$)	54 (+16)	30,7311	21,5573	2038,39
	MR ($n_{mr} = 3$, $d_{mr} = 2$)	119 (+35)	30,0526	21,5270	507,97
`mbb384x64x64`	Tradicional	23 (+4)	24,5260	17,9443	4172,88
	MR ($n_{mr} = 4$, $d_{mr} = 4$)	27 (+3)	24,5207	17,7055	5128,90
	MR ($n_{mr} = 4$, $d_{mr} = 3$)	62 (+19)	24,5949	17,6148	2289,97
	MR ($n_{mr} = 4$, $d_{mr} = 2$)	109 (+31)	24,1608	17,7139	497,78

Tabela 7.51: Resultados para a torre de transmissão utilizando a multirresolução.

Problema	**Método**	N_{it} (N_{rc})	F	F_{prj}	T_{total}
`tt32x160x32`	Tradicional	145 (+39)	74,1889	64,2686	584,69
	MR ($n_{mr} = 2$, $d_{mr} = 2$)	48 (+3)	75,0951	64,5759	85,62
`tt48x240x48`	Tradicional	38 (+9)	59,8312	52,3252	915,67
	MR ($n_{mr} = 3$, $d_{mr} = 3$)	56 (+10)	59,6810	52,3825	866,34
	MR ($n_{mr} = 3$, $d_{mr} = 2$)	70 (+15)	59,8071	52,4316	145,45
`tt64x320x64`	Tradicional	30 (+4)	52,0692	46,2967	2830,03
	MR ($n_{mr} = 4$, $d_{mr} = 4$)	65 (+16)	51,8031	46,3002	5372,10
	MR ($n_{mr} = 4$, $d_{mr} = 3$)	69 (+10)	51,8570	46,3242	1022,87
	MR ($n_{mr} = 4$, $d_{mr} = 2$)	79 (+19)	51,9503	46,3885	173,78

A fim de obter resultados mais conclusivos sobre a qualidade da solução encontrada com a multirresolução, resolvemos o problema pelo método tradicional utilizando o vetor de densidades obtido com a multirresolução como aproximação inicial. Se o valor da função objetivo (F_{pos}) calculado após essa nova resolução do problema estiver próximo do valor (F_{mr}) calculado para a solução inicial, significa que a distribuição de material encontrada com a multirresolução já era uma boa solução para o problema. Caso contrário, concluímos que a solução obtida pela multirresolução é imprecisa. Na Tabela 7.52, apresentamos os valores F_{mr} e F_{pos}, bem como o número de iterações da PLS (N_{pos}) necessárias para resolver o problema novamente usando a solução da multirresolução como aproximação inicial, obtidos para alguns problemas da viga MBB e da torre com $n_{mr} = d_{mr}$.

Em geral, o número de iterações (N_{pos}) foi pequeno, com exceção dos problemas `tt32x160x32` e `tt48x240x48`, o que fez com que o valor da razão F_{mr}/F_{pos} fosse um pouco

maior nesses dois casos. Contudo, o valor dessa razão foi aproximadamente igual a 1 em todos os problemas, indicando que as soluções que obtivemos com a multirresolução são consideravelmente boas.

Tabela 7.52: Resultados obtidos ao resolver o problema pelo método tradicional utilizando a solução encontrada pela multirresolução como aproximação inicial.

Problema	N_{pos}	F_{mr}	F_{pos}	F_{mr}/F_{pos}
`mbb192x32x32`	7 (+2)	41,8844	41,8522	1,0008
`mbb288x48x48`	3 (+0)	30,7311	30,7041	1,0009
`mbb384x64x64`	3 (+0)	24,5207	24,4960	1,0010
`tt32x160x32`	41 (+9)	75,0951	74,5537	1,0073
`tt48x240x48`	12 (+2)	59,6810	59,5096	1,0029
`tt64x320x64`	3 (+0)	51,8031	51,7492	1,0010

Para analisar o tempo gasto em cada etapa do algoritmo, resolvemos alguns problemas parando após 20 repetições do laço principal, isto é, após 20 iterações externas da PLS. Dividimos os tempos das etapas principais do algoritmo da seguinte maneira:

- T_{pf}: tempo gasto na pré-filtragem.
- T_f: tempo gasto nas aplicações do filtro.
- T_k: tempo gasto para montar as matrizes de rigidez.
- T_{sp}: tempo gasto na fase de *setup* do precondicionador.
- T_s: tempo gasto para resolver os sistemas lineares.
- T_g: tempo gasto para os cálculos do gradiente da função objetivo.
- T_{lp}: tempo gasto na resolução dos subproblemas de programação linear.
- T_o: tempo gasto nas demais linhas de código.
- T_{total}: tempo total gasto para resolver o problema.

Todos os tempos são medidos em segundos. Nas Tabelas 7.53 e 7.54, apresentamos os resultados para os problemas da viga MBB e da torre de transmissão, com resoluções `288x48x48` e `48x240x48`, respectivamente.

Os tempos de montagem da matriz de rigidez, *setup* do precondicionador e resolução dos sistemas lineares são sempre reduzidos com a multirresolução, uma vez que os sistemas de equilíbrio são resolvidos em uma malha mais grossa (malha de deslocamentos) se comparada com a malha do método tradicional. Quando $n_{mr} = d_{mr} = 3$, as malhas de variáveis de projeto e de densidades são iguais e ambas têm o mesmo refinamento daquela utilizada no método tradicional. Sendo assim, os tempos de pré-filtragem e aplicações do filtro ficam parecidos. Em contrapartida, o tempo de cálculo dos gradientes foi um pouco menor para a multirresolução, pois mais passos da PLS foram recusados, enquanto o tempo de resolução dos PL foi maior, provavelmente porque os subproblemas resolvidos

eram mais complicados. No total, o tempo gasto foi reduzido com a multirresolução. Essa redução é muito maior quando utilizamos $d_{mr} = 2$, uma vez que, neste caso, os tempos gastos em todas as etapas são menores do que os do método tradicional.

Tabela 7.53: Tempos gastos em cada etapa do algoritmo para o problema `mbb288x48x48`.

Tempo	Tradicional	MR ($n_{mr} = 3$, $d_{mr} = 3$)	MR ($n_{mr} = 3$, $d_{mr} = 2$)
T_{pf}	63,72	64,54	6,75
T_f	2,70	3,00	0,80
T_k	79,72	3,55	3,38
T_{sp}	8,53	0,37	0,37
T_s	68,14	7,47	7,33
T_g	6,52	2,74	2,40
T_{lp}	752,04	830,35	77,86
T_o	0,32	0,14	0,07
T_{total}	981,69	912,16	98,96

Tabela 7.54: Tempos gastos em cada etapa do algoritmo para o problema `tt48x240x48`.

Tempo	Tradicional	MR ($n_{mr} = 3$, $d_{mr} = 3$)	MR ($n_{mr} = 3$, $d_{mr} = 2$)
T_{pf}	17,27	17,18	2,32
T_f	0,44	0,43	0,24
T_k	77,69	3,14	3,21
T_{sp}	7,60	0,36	0,38
T_s	77,56	7,32	7,77
T_g	4,88	2,28	2,21
T_{lp}	390,80	427,72	38,63
T_o	0,22	0,12	0,06
T_{total}	576,46	458,56	54,81

7.7.2 Outras estruturas

O propósito da otimização topológica é ajudar na escolha de um formato inicial para uma estrutura previamente desconhecida. Vimos que a multirresolução consegue obter resultados em tempos menores do que o método tradicional e, embora existam algumas diferenças, o formato da estrutura fica parecido com aquele de referência e é bastante razoável. Nesta seção, testamos a resolução de outros problemas, incluindo alguns mais desafiadores, que envolvem cargas distribuídas ou regiões com densidades fixas no domínio.

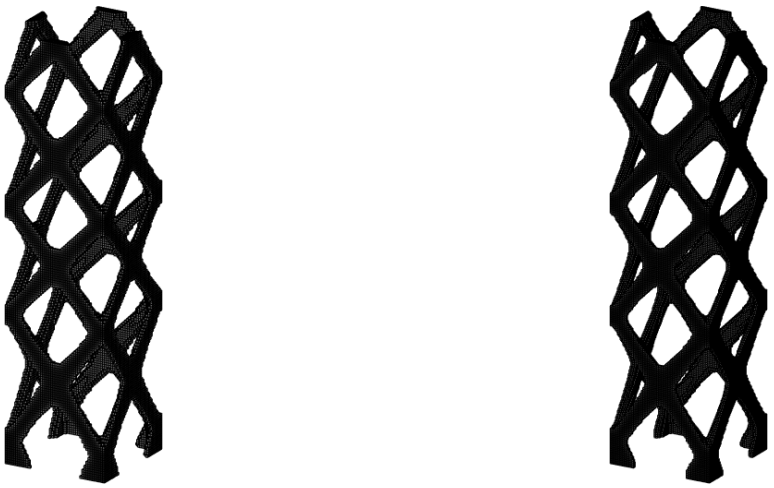
As Figuras 7.47, 7.48 e 7.49 mostram as estruturas ótimas obtidas para os problemas do meio cubo, do prédio com torção e da viga em L (Exemplos 4, 5 e 6 da Seção 7.2), respectivamente. Para esses problemas, consideramos como base os refinamentos com `32x16x32`, `16x64x16` e `48x48x16` elementos de deslocamento, respectivamente,

correspondendo à estrutura completa. Nesses três problemas, utilizamos $d_{mr} = 2$ e obtivemos duas resoluções diferentes, com $n_{mr} = 3$ e $n_{mr} = 4$. Na Figura 7.50, mostramos a estrutura obtida para o problema da ponte (Exemplo 7 da Seção 7.2) com $n_{mr} = 3$ e $d_{mr} = 2$, utilizando como base o refinamento com `192x48x24` elementos de deslocamento.

Observamos que alguns artigos consideram restrições de tensão no problema da viga em L, para evitar tensões excessivas na região que conecta as partes horizontal e vertical da viga, mas não fizemos isso neste trabalho. Ademais, no problema da ponte, normalizamos o vetor de cargas nodais para facilitar a resolução do problema.

(a) `hc96x48x96` (b) `hc128x64x128`

Figura 7.47: Estruturas obtidas para o problema do meio cubo ($r_{min} = 1{,}5$).



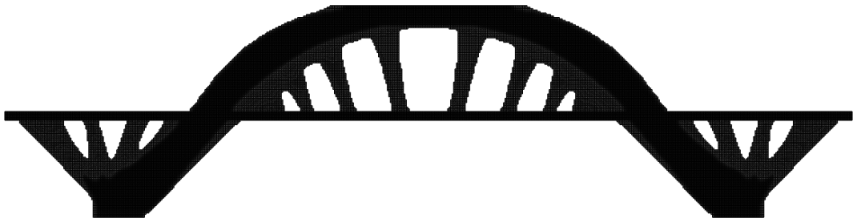
(a) `bt48x192x48` (b) `bt64x256x64`

Figura 7.48: Estruturas obtidas para o problema do prédio com torção ($r_{min} = 1{,}5$).

(a) `ls144x144x48` (b) `ls192x192x64`

Figura 7.49: Estruturas obtidas para o problema da viga em L ($r_{min} = 1{,}5$).

(a) Visão oblíqua da ponte



(b) Visão lateral da ponte

Figura 7.50: Estrutura obtida para o problema da ponte `bd576x144x72` ($r_{min} = 1{,}5$).

Todas as estruturas encontradas são adequadas e fazem sentido, do ponto de vista da engenharia. Portanto, conseguimos resolver os problemas de otimização topológica estrutural utilizando a multirresolução, incluindo problemas mais complicados, que envolvem múltiplas cargas (prédio com torção), regiões vazias fixas (viga em L), regiões sólidas fixas e cargas distribuídas (ponte).

Na Tabela 7.55, apresentamos os resultados obtidos para esses problemas, incluindo o número de iterações da PLS (N_{it}), com o número de passos recusados entre parênteses (N_{rc}), e o tempo total de resolução (T_{total}). A coluna "Dim." da tabela mostra a quantidade de elementos na malha de densidades, considerando apenas a parte simétrica das estruturas. Notamos que o tempo de resolução foi razoavelmente baixo, apesar do número de iterações ter sido grande em alguns casos.

Tabela 7.55: Resultados obtidos para outras estruturas com a multirresolução.

Problema	**Dim.**	N_{it} (N_{rc})	T_{total}
`hc96x48x96`	110592	43 (+9)	71,74
`hc128x64x128`	262144	42 (+10)	80,59
`bt48x192x48`	442368	334 (+66)	5225,36
`bt64x256x64`	1048576	164 (+30)	2703,75
`ls144x144x48`	497664	116 (+29)	1047,41
`ls192x192x64`	1179648	122 (+29)	1144,29
`bd576x144x72`	1492992	129 (+17)	9015,47

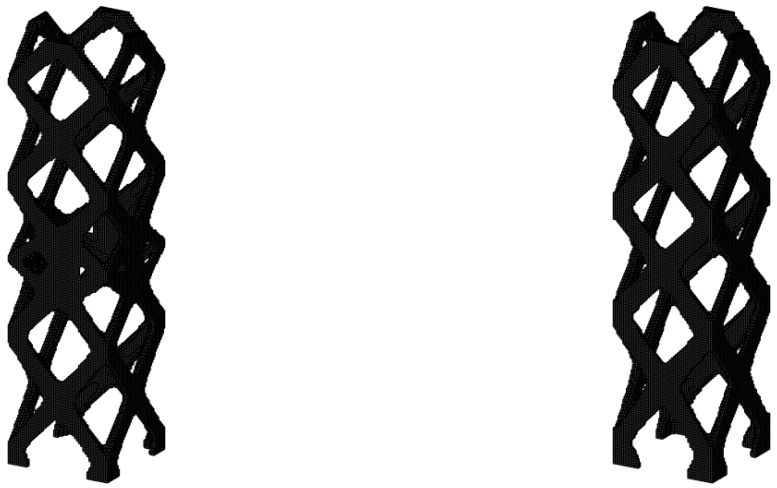
Obtivemos estruturas boas com resoluções altas, consumindo menos memória e tempo computacional. As dimensões dos problemas `bt64x256x64`, `ls192x192x64` e `bd576x144x72` ultrapassam 1 milhão de elementos, mesmo considerando apenas a parte

simétrica das estruturas. Portanto, conseguimos resolver de maneira eficiente problemas tridimensionais que podem ser considerados de grande porte.

Com o método tradicional, sequer conseguimos resolver alguns desses problemas com resoluções maiores, devido à falta de memória computacional. Nas Figuras 7.51, 7.52 e 7.53, comparamos as estruturas obtidas pelo método tradicional e pela multirresolução para os problemas com dimensões mais baixas, após o arredondamento de densidades. Nos três problemas, com a multirresolução utilizamos o raio do filtro $r_{min} = 1{,}5$ e com o método tradicional utilizamos um raio proporcional, igual a $n_{mr} r_{min} = 4{,}5$.

(a) Tradicional (b) MR com $n_{mr} = 3$, $d_{mr} = 2$

Figura 7.51: Estruturas obtidas para o problema `hc96x48x96` ($r_{min} = 1{,}5$).



(a) Tradicional (b) MR com $n_{mr} = 3$, $d_{mr} = 2$

Figura 7.52: Estruturas obtidas para o problema `bt48x192x48` ($r_{min} = 1{,}5$).

(a) Tradicional (b) MR com $n_{mr} = 3$, $d_{mr} = 2$

Figura 7.53: Estruturas obtidas para o problema `ls144x144x48` ($r_{min} = 1{,}5$).

Notamos que há diferenças nas estruturas encontradas pelo método tradicional, como um preenchimento maior na parte inferior do meio cubo, bem como na parte central do prédio com torção, e a substituição de duas barras laterais da viga em L por uma única barra central. É possível que consigamos encontrar estruturas mais parecidas com as obtidas pela multirresolução fazendo ajustes no valor do raio do filtro. Além disso, lembramos que os problemas resolvidos por cada método não são idênticos. Contudo, os formatos obtidos são bastante próximos e comprovam que a multirresolução é capaz de encontrar soluções adequadas.

Na Tabela 7.56, comparamos os resultados numéricos obtidos pelo método tradicional e pela multirresolução. O valor da função objetivo apresentado corresponde ao da solução com densidades arredondadas para 0 ou 1 (F_{prj}) e esse valor foi corrigido para a multirresolução, utilizando o vetor de densidades ótimo encontrado para resolver o sistema de equilíbrio na mesma malha do método tradicional.

Tabela 7.56: Comparações entre o método tradicional e a multirresolução.

Problema	**Método**	N_{it} (N_{rc})	F_{prj}	T_{total}
`hc96x48x96`	Tradicional	17 (+4)	8,6015	382,70
	MR ($n_{mr} = 3$, $d_{mr} = 2$)	43 (+9)	8,4205	71,74
`bt48x192x48`	Tradicional	126 (+54)	66,8234	24206,39
	MR ($n_{mr} = 3$, $d_{mr} = 2$)	334 (+66)	66,4727	5225,36
`ls144x144x48`	Tradicional	45 (+4)	35,5820	5842,83
	MR ($n_{mr} = 3$, $d_{mr} = 2$)	116 (+29)	39,0343	1047,41

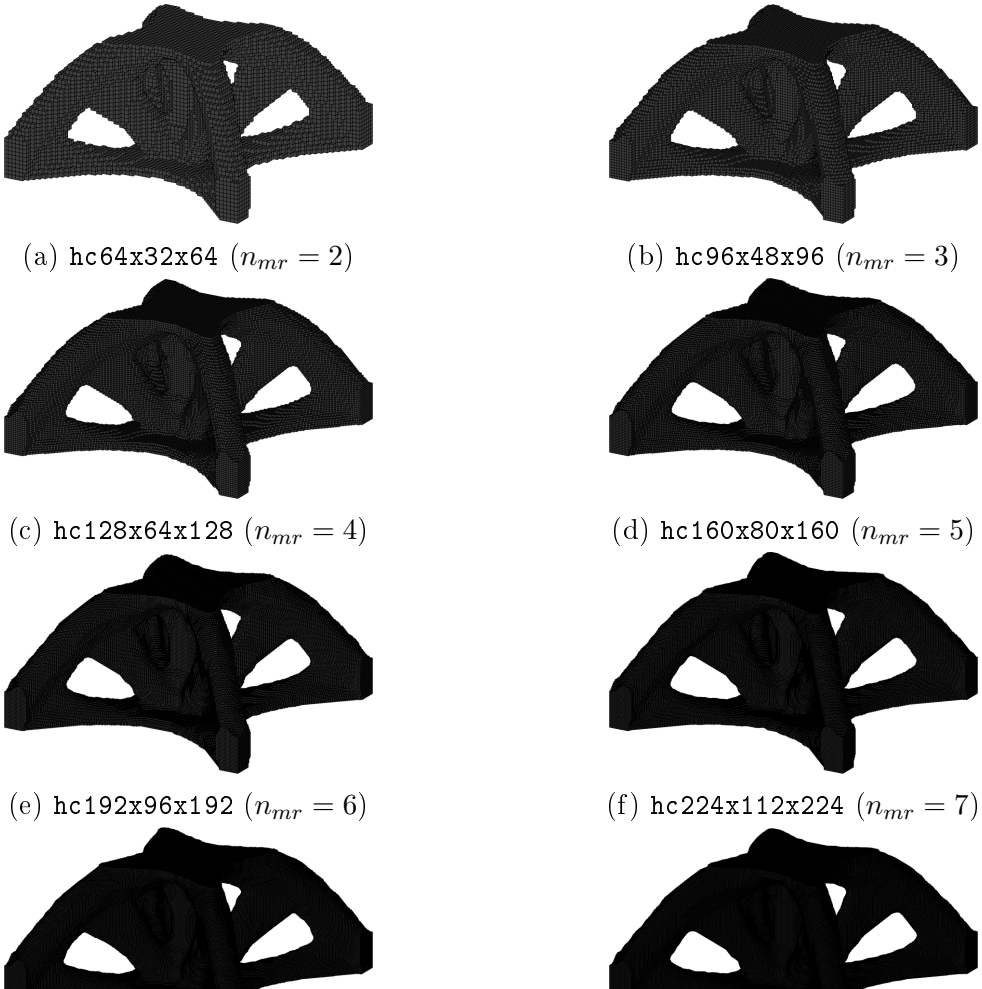
Apesar do número de iterações ter sido maior com a multirresolução em todos os problemas, o tempo total gasto foi significativamente menor. O valor da flexibilidade, calculado para a solução arredondada, foi levemente menor com a multirresolução nos casos do meio cubo e do prédio com torção, e maior no caso da viga em L.

7.7.3 Crescimento do tempo com a dimensão

Testamos a resolução de diversos tipos de problemas com a multirresolução, variando os valores dos parâmetros n_{mr} e d_{mr}. Em geral, obtivemos estruturas boas e soluções precisas com um baixo custo computacional. Entretanto, notamos que é necessário tomar cuidado com a escolha do parâmetro d_{mr}. É recomendado que esse valor esteja dentro do limite imposto na Tabela 5.1, para amenizar o surgimento de artefatos. Além disso, quanto maior for o valor de d_{mr}, maior será a quantidade de variáveis de projeto, o que faz com que o tempo de resolução dos subproblemas de PL cresça. Por outro lado, o valor de n_{mr} determina a resolução final da estrutura. Logo, desejamos aumentar o valor de n_{mr} e reduzir o valor de d_{mr}.

Nesta seção, analisamos a qualidade da solução obtida e o crescimento do tempo gasto em cada etapa do algoritmo quando aumentamos o valor de n_{mr} e mantemos o valor de d_{mr} fixo. Para tanto, resolvemos o problema do meio cubo com carga concentrada, considerando o refinamento com `32x16x32` elementos de deslocamento como base e variando o valor do parâmetro n_{mr} de 2 até 9. Mantivemos o parâmetro $d_{mr} = 2$ fixo, bem como o raio do filtro $r_{min} = 1{,}5$.

A Figura 7.54 mostra as estruturas obtidas para o problema do cubo com os diferentes valores de n_{mr}, após o arredondamento de densidades. Como estamos mantendo o valor do raio do filtro, as soluções são independentes da malha, de modo que não ganhamos novos detalhes no formato da estrutura com o aumento da resolução. Além disso, uma diferença grande entre os valores de n_{mr} e d_{mr} pode fazer com que a qualidade do contorno da estrutura seja reduzida, exigindo que aumentemos o raio do filtro, o que prejudica a eficiência do algoritmo.



(a) `hc64x32x64` ($n_{mr} = 2$) (b) `hc96x48x96` ($n_{mr} = 3$)

(c) `hc128x64x128` ($n_{mr} = 4$) (d) `hc160x80x160` ($n_{mr} = 5$)

(e) `hc192x96x192` ($n_{mr} = 6$) (f) `hc224x112x224` ($n_{mr} = 7$)

(g) `hc256x128x256` ($n_{mr} = 8$) (h) `hc288x144x288` ($n_{mr} = 9$)

Figura 7.54: Estruturas obtidas para o problema do meio cubo utilizando a multirresolução com $d_{mr} = 2$ e diferentes valores de n_{mr} ($r_{min} = 1{,}5$).

Para uma análise mais efetiva do crescimento do tempo, paramos a resolução dos problemas em um número fixo de 20 repetições do laço principal do algoritmo, isto é, 20 iterações externas da PLS. A Tabela 7.57 mostra os tempos gastos em cada etapa do algoritmo, em função do parâmetro n_{mr}.

As malhas de deslocamento e de variáveis de projeto são sempre as mesmas, uma vez que o problema base e o parâmetro d_{mr} estão fixos. Assim, os sistemas lineares, bem como os PLs, têm sempre as mesmas dimensões, independentemente do valor de n_{mr}. Por esse motivo, os tempos de *setup* do precondicionador (T_{sp}), resolução dos sistemas

lineares (T_s) e resolução dos PLs (T_{lp}) se mantêm aproximadamente constantes com o aumento de n_{mr}. Os tempos gastos nas demais etapas crescem conforme aumentamos a resolução final da estrutura, pois dependem da quantidade de elementos na malha de densidades. Embora a matriz de rigidez tenha sempre a mesma dimensão, o tempo T_k cresce quando aumentamos n_{mr} porque a matriz de cada elemento de deslocamento é calculada pela equação (5.15), que envolve uma soma de termos calculados para cada elemento de densidade contido no elemento de deslocamento.

Tabela 7.57: Crescimento dos tempos gastos em cada etapa do algoritmo utilizando a multirresolução, em função do parâmetro n_{mr}, para o problema do meio cubo.

n_{mr}	T_{pf}	T_f	T_k	T_{sp}	T_s	T_g	T_{lp}	T_{total}
2	3,32	0,09	2,56	0,35	4,08	0,81	30,11	41,48
3	2,61	0,28	2,62	0,34	4,40	1,77	31,65	43,74
4	5,89	0,65	3,11	0,34	4,19	3,69	30,78	48,71
5	11,64	1,39	4,20	0,35	4,44	6,82	32,77	61,67
6	19,97	2,31	5,43	0,34	4,40	11,45	31,21	75,22
7	32,44	3,69	7,30	0,34	4,56	18,03	32,02	98,48
8	48,06	5,77	12,94	0,34	4,92	27,10	32,88	132,15
9	72,36	8,07	15,93	0,36	4,66	40,78	36,06	180,33

O crescimento dos tempos gastos na pré-filtragem (T_{pf}), nas aplicações do filtro (T_f), na montagem das matrizes de rigidez (T_k) e no cálculo dos gradientes (T_g) exige que tenhamos certo cuidado ao aumentar o valor de n_{mr} demasiadamente.

Com base nos resultados que obtivemos até aqui, consideramos que o mais adequado é utilizar um valor pequeno para o parâmetro d_{mr}, respeitando os limites indicados em [27], como $d_{mr} = 2$ quando usamos elementos finitos de grau 1, por exemplo. Ademais, se o objetivo é reduzir o raio do filtro para obter mais detalhes na estrutura com o aumento da resolução, o melhor é utilizar valores de n_{mr} próximos de d_{mr} (por exemplo, $d_{mr} \leq n_{mr} \leq d_{mr} + 3$), para que a qualidade da solução não seja perdida.

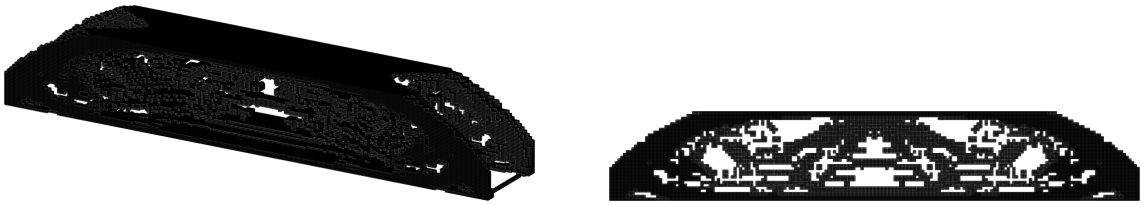
7.7.4 O problema dos artefatos

Na multirresolução, quando aumentamos o refinamento da malha de densidades, ou seja, quando aumentamos o valor de n_{mr}, o tamanho dos elementos de densidade diminui. Se mantivermos o mesmo raio do filtro, a região de aplicação do filtro ficará igual para todas as resoluções, de modo que obteremos resultados independentes da malha, conforme vimos anteriormente.

Dessa maneira, o formato da estrutura em alta resolução é o mesmo daquele encontrado com uma resolução mais baixa, com a vantagem de possuir uma qualidade melhor e contornos mais suaves. Todavia, com o aumento da resolução, muitas vezes desejamos observar mais detalhes no formato da estrutura, incluindo o surgimento de novos buracos e novas barras, por exemplo. Nesse caso, para atingir esse objetivo, precisamos reduzir o raio de aplicação do filtro.

Contudo, a redução do raio do filtro torna mais aparente um infortúnio da

multirresolução, que é o surgimento de regiões com o material distribuído em um padrão aleatório, com partes "flutuando" ou barras incompletas, como ilustra a Figura 7.55 para a viga MBB com raio $r_{min} = 0{,}6$. Conforme vimos na Subseção 5.5, de acordo com Gupta *et al.* [26], o aparecimento dessas regiões (denominadas *artefatos*) é decorrente da baixa precisão utilizada na aproximação dos deslocamentos da estrutura, que é feita em uma malha muito mais grossa do que a malha de densidades.



(a) Visão oblíqua da viga (b) Visão lateral da viga

Figura 7.55: Estrutura obtida para a viga MBB, utilizando a multirresolução com $n_{mr} = 4$ e $d_{mr} = 2$, raio do filtro $r_{min} = 0{,}6$ e elementos finitos de grau 1.

Uma maneira de amenizar o problema dos artefatos é a aplicação de um filtro de densidades. Entretanto, para que a estrutura não tenha artefatos, devemos utilizar um raio de filtro relativamente grande, o que nos leva de volta ao problema da baixa resolução da estrutura obtida. Logo, se queremos reduzir o raio para encontrar mais detalhes da estrutura, é necessário aplicar outras estratégias para a correção dos artefatos. Neste trabalho, testamos a utilização de elementos finitos de grau maior do que 1, apresentados nas Subseções 1.3.2 e 1.3.3, com o intuito de melhorar a aproximação dos deslocamentos da estrutura.

Na próxima seção, apresentamos resultados obtidos com o aumento do grau dos elementos finitos. A desvantagem em se utilizar elementos de grau maior do que 1 é que a dimensão das matrizes de rigidez aumenta, de modo que a resolução dos sistemas lineares pode voltar a ser a etapa mais cara do algoritmo, mesmo com a utilização do *multigrid*. Neste caso, testamos as estratégias adaptativas para suprimir variáveis, propostas na Seção 6.2, reduzindo a dimensão dos sistemas lineares e a quantidade de variáveis do problema, de modo que podemos utilizar elementos de grau 2 ou 3 sem prejudicar demais a eficiência conquistada com a multirresolução.

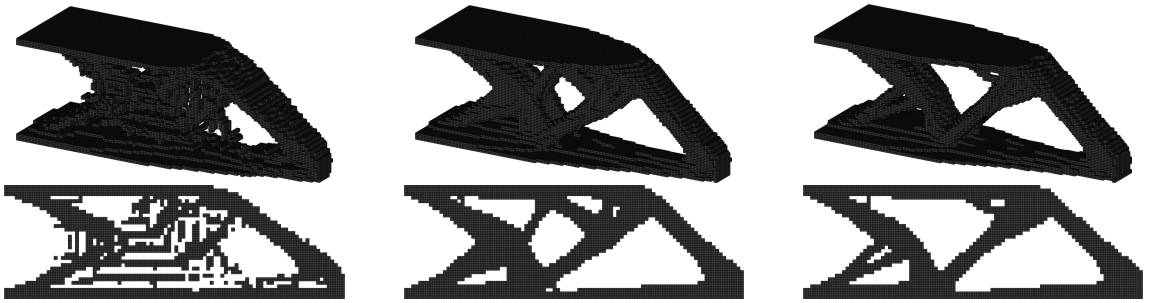
7.8 Aumento do grau dos elementos finitos

Para amenizar o surgimento de artefatos na estrutura quando reduzimos o raio de aplicação do filtro e melhorar a qualidade da solução obtida com a multirresolução, testamos a utilização de elementos finitos com grau maior do que 1, isto é, elementos cujas funções de forma são compostas por polinômios de grau maior do que 1, de modo que a aproximação dos deslocamentos da estrutura seja mais precisa.

Inicialmente, comparamos a utilização de elementos do tipo Lagrange e do tipo *serendipity*, apresentados nas Subseções 1.3.2 e 1.3.3, respectivamente, com grau 1, 2 e 3. Em seguida, testamos a estratégia adaptativa de aumento do grau dos elementos e fixação de variáveis, proposta na Seção 6.2. Para utilizar o *multigrid* geométrico com elementos do tipo *serendipity*, realizamos a prolongação como proposto na Subseção 3.2.4. Ademais, utilizamos o suavizador SSOR, que foi melhor para os elementos de grau maior que 1.

7.8.1 Comparações entre elementos Lagrange e *serendipity*

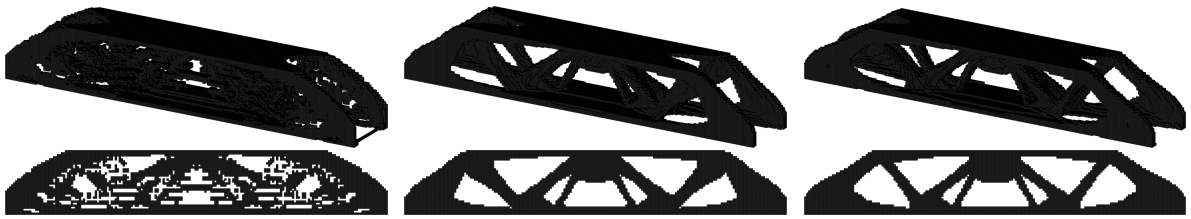
Resolvemos os problemas da viga em balanço e da viga MBB, utilizando a multirresolução com $n_{mr} = 4$ e $d_{mr} = 2$, raio do filtro $r_{min} = 0{,}6$ e comparamos as soluções obtidas com elementos do tipo Lagrange e do tipo *serendipity* de grau 1, 2 e 3. As Figuras 7.56 e 7.57 mostram as estruturas obtidas para os problemas `cb192x64x64` e `mbb384x64x64`, respectivamente. Nas figuras, exibimos a solução após o arredondamento de densidades, com a visão oblíqua e a visão lateral das estruturas, para permitir uma melhor visualização dos detalhes e da presença de artefatos.



(a) Grau 1 (b) Grau 2 Lagrange (c) Grau 2 *serendipity*

(d) Grau 3 Lagrange (e) Grau 3 *serendipity*

Figura 7.56: Estruturas obtidas para o problema `cb192x64x64`, utilizando a multirresolução com $n_{mr} = 4$, $d_{mr} = 2$ e raio do filtro $r_{min} = 0{,}6$.



(a) Grau 1 (b) Grau 2 Lagrange (c) Grau 2 *serendipity*

(d) Grau 3 Lagrange (e) Grau 3 *serendipity*

Figura 7.57: Estruturas obtidas para o problema `mbb384x64x64`, utilizando a multirresolução com $n_{mr} = 4$, $d_{mr} = 2$ e raio do filtro $r_{min} = 0{,}6$.

A distribuição de material encontrada com elementos de grau 1 conteve uma grande quantidade de artefatos, de modo que é difícil interpretar a estrutura obtida. Com o uso de elementos de grau maior, as partes sólidas ficaram mais definidas e não apareceram artefatos visíveis, sendo que com grau 3 a estrutura ficou levemente melhor do que com grau 2. Assim, verificamos que a melhoria na aproximação dos deslocamentos faz com que o surgimento dos artefatos seja amenizado. Para os problemas tridimensionais, parece ser suficiente utilizar elementos de grau 2 ou 3.

Em geral, a redução no raio do filtro fez com que surgissem novos detalhes na estrutura, como era desejado. Notamos, entretanto, que as soluções encontradas foram diferentes quando alteramos o tipo e o grau dos elementos. Isso provavelmente ocorreu devido ao problema de otimização topológica possuir muitos mínimos locais e estarmos encontrando apenas uma aproximação para a melhor distribuição de material. A variação na solução com a mudança no tipo de elemento foi mais aparente na viga em balanço, onde as barras centrais mudaram bastante. Já no caso da viga MBB, houve uma pequena mudança nas soluções obtidas com grau 3, onde surgiram algumas barras adicionais.

A Tabela 7.58 mostra os resultados obtidos para cada problema com os diferentes tipos de elementos. O valor $\tilde{F}_{prj}$ corresponde à flexibilidade ótima calculada pela multirresolução para a solução com densidades arredondadas para 0 ou 1. Já o valor F_{prj} corresponde à flexibilidade ótima corrigida, calculada com os deslocamentos obtidos resolvendo-se o sistema de equilíbrio na malha de densidades com elementos de grau 1.

Tabela 7.58: Resultados obtidos com diferentes tipos de elementos.

Problema	**Elemento**	N_{it} (N_{rc})	$\tilde{F}_{prj}$	F_{prj}	T_{total}
`cb192x64x64`	Lagrange de grau 1	146 (+3)	50,7310	32,9178	304,03
	Lagrange de grau 2	54 (+3)	65,4032	18,9579	964,84
	Lagrange de grau 3	53 (+5)	84,9460	18,9811	10179,56
	Serendipity de grau 2	38 (+0)	60,3249	19,2790	1094,98
	Serendipity de grau 3	50 (+0)	62,2680	19,1914	5853,88
`mbb384x64x64`	Lagrange de grau 1	177 (+0)	49,5897	88,6575	371,84
	Lagrange de grau 2	41 (+1)	62,2408	17,8114	713,69
	Lagrange de grau 3	86 (+27)	78,3640	17,9775	18518,54
	Serendipity de grau 2	43 (+1)	57,0394	18,0393	1136,08
	Serendipity de grau 3	90 (+30)	58,6524	18,0499	14199,92

Notamos que, em ambos os problemas, o valor $\tilde{F}_{prj}$ foi muito menor com elementos de grau 1 do que com elementos de grau maior. Isso indica que a estrutura com artefatos possui uma rigidez artificial, de modo que a sua flexibilidade aparenta ser menor. Além disso, o valor da função objetivo varia bastante de acordo com o tipo e o grau dos elementos, o que faz com que a comparação seja difícil e imprecisa. Quando corrigimos o valor da função, calculando F_{prj}, a flexibilidade ficou muito maior com elementos de grau 1. Ademais, os valores ficaram mais próximos e comparáveis utilizando elementos de grau 2 e 3. O valor F_{prj} para a solução com elementos do tipo Lagrange foi levemente menor do que com elementos do tipo *serendipity*. Para ambos os tipos de elementos, o valor com grau 2 foi levemente menor do que com grau 3, exceto para o problema `cb192x64x64` com elementos *serendipity*. Todavia, ressaltamos que a comparação entre os valores da função

objetivo ainda não é perfeita, uma vez que as soluções obtidas foram visivelmente diferentes e a mudança no tipo e no grau dos elementos faz com que os problemas resolvidos não sejam idênticos.

Quando mudamos do grau 2 para o grau 3, a diferença no valor da função objetivo (F_{prj}) foi relativamente pequena. Por outro lado, a diferença no tempo total de resolução (T_{total}) foi significativa. O aumento no grau dos elementos faz com que o tempo de resolução cresça bastante, uma vez que a dimensão dos sistemas lineares aumenta. Dessa forma, devemos ter cuidado com o aumento excessivo do grau dos elementos em problemas tridimensionais. Felizmente, pelos resultados obtidos até aqui, notamos que as soluções encontradas com elementos de grau 2 já são consideravelmente boas. Utilizar elementos de grau maior do que 2 ou 3 pode não ser vantajoso por conta do crescimento do custo computacional.

Para uma melhor comparação entre os tempos gastos com cada tipo de elemento, resolvemos os problemas até atingir 30 iterações externas da PLS. Na Tabela 7.59, apresentamos os tempos gastos nas etapas de montagem da matriz de rigidez (T_k), *setup* do precondicionador (T_{sp}), resolução dos sistemas lineares (T_s) e cálculo dos gradientes (T_g), bem como o tempo total (T_{total}), utilizando o *multigrid* geométrico como precondicionador do método dos gradientes conjugados na resolução dos sistemas.

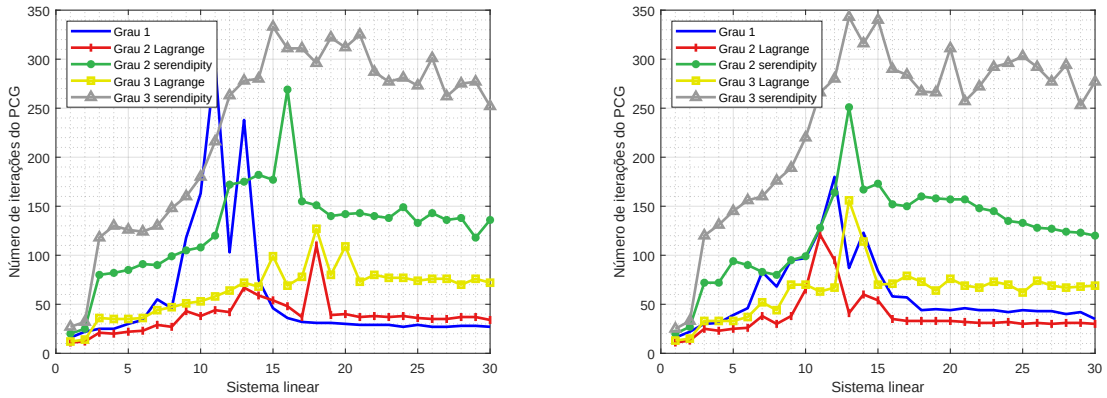
Tabela 7.59: Tempos gastos com diferentes tipos de elementos.

Problema	**Elemento**	T_k	T_{sp}	T_s	T_g	T_{total}
`cb192x64x64`	Lagrange de grau 1	5,84	0,59	16,32	8,28	103,27
	Lagrange de grau 2	67,69	20,36	403,45	22,17	583,47
	Serendipity de grau 2	30,14	12,40	758,75	14,46	883,91
	Lagrange de grau 3	529,27	121,44	4616,35	354,57	5691,31
	Serendipity de grau 3	90,87	34,51	3422,88	27,87	3645,09
`mbb384x64x64`	Lagrange de grau 1	5,50	0,56	16,24	8,13	98,72
	Lagrange de grau 2	65,70	19,76	393,90	22,35	563,82
	Serendipity de grau 2	30,02	12,67	735,98	14,29	857,53
	Lagrange de grau 3	642,87	148,90	5669,17	355,53	6885,23
	Serendipity de grau 3	103,20	38,39	3945,65	30,53	4182,03

Naturalmente, os tempos gastos nas etapas relacionadas ao cálculo dos deslocamentos nodais (T_k, T_{sp} e T_s) e o tempo gasto no cálculo dos gradientes (T_g), que envolve os deslocamentos, crescem conforme aumentamos o grau dos elementos, uma vez que o número de nós da malha aumenta.

Comparando os elementos do tipo Lagrange e do tipo *serendipity* com grau 2 ou 3, notamos que os tempos T_k, T_{sp} e T_g foram menores para os elementos *serendipity*. Isso era esperado, já que os elementos do tipo *serendipity* possuem uma quantidade menor de nós em relação aos elementos do tipo Lagrange de mesmo grau. Entretanto, o tempo de resolução dos sistemas (T_s) e, consequentemente, o tempo total, foram maiores para os elementos *serendipity* com grau 2 e menores apenas com grau 3. Para entender melhor o que aconteceu nesse caso, analisamos o número de iterações do método dos gradientes conjugados. A Figura 7.58 mostra os gráficos da quantidade de iterações do PCG para

cada sistema linear dos problemas da viga em balanço e da viga MBB, com cada um dos tipos de elementos finitos.



(a) `cb192x64x64` (b) `mbb384x64x64`

Figura 7.58: Número de iterações do PCG com diferentes tipos de elementos.

Observe que o número de iterações do PCG foi menor com os elementos do tipo Lagrange em comparação aos elementos do tipo *serendipity*. Acreditamos que isso ocorre pois o *multigrid* geométrico possui uma certa vantagem com os elementos de Lagrange. Os nós nesse tipo de elemento estão igualmente espaçados, de modo que é possível realizar a prolongação do *multigrid* de maneira usual, interpolando os valores para nós da malha fina com base nos valores dos nós vizinhos na malha grossa. No caso dos elementos do tipo *serendipity*, a posição dos nós faz com que a prolongação seja mais complicada. Neste caso, realizamos a prolongação utilizando a interpolação por meio das funções de forma do método dos elementos finitos, conforme descrito na Subseção 3.2.4. Essa estratégia funcionou bem, porém o *multigrid* geométrico não teve o mesmo desempenho como precondicionador quando utilizamos os elementos *serendipity*, de modo que o número de iterações do método dos gradientes conjugados aumentou. Com elementos de grau 2, apesar da dimensão dos sistemas lineares ser menor para os elementos *serendipity*, o aumento no número de iterações do PCG fez com que o tempo de resolução dos sistemas fosse maior do que com elementos Lagrange. Por outro lado, com grau 3, a redução na dimensão dos sistemas é mais significativa do que o aumento no número de iterações, de tal forma que o tempo de resolução dos sistemas ficou menor para os elementos *serendipity*.

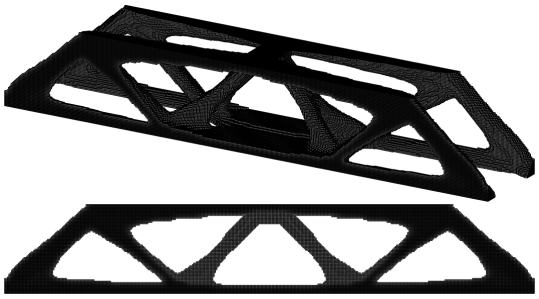
Com base nesses resultados, concluímos que é mais vantajoso utilizar elementos do tipo Lagrange até o grau 2 e do tipo *serendipity* com grau 3 ou mais quando usamos o *multigrid* geométrico na resolução dos sistemas lineares. As soluções obtidas foram boas para ambos os tipos de elementos e o uso de elementos com grau 2 ou 3 foi suficiente para reduzir a presença de artefatos nas estruturas tridimensionais.

7.8.2 Estratégia adaptativa de aumento do grau dos elementos

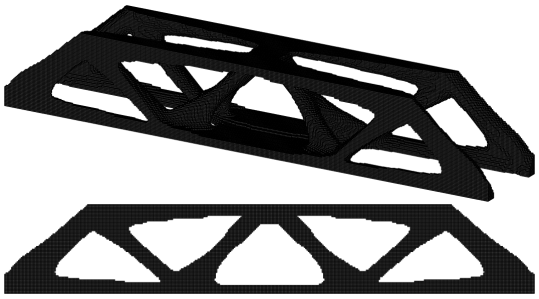
Vimos que o aumento no grau dos elementos finitos é capaz de melhorar a qualidade da solução do problema de otimização topológica estrutural com a multirresolução, mas provoca um crescimento no custo computacional. Nesta subseção, testamos a estratégia adaptativa de aumento do grau dos elementos, proposta na Seção 6.2, com o objetivo de obter estruturas detalhadas e sem artefatos, prejudicando o mínimo possível a eficiência conquistada com a multirresolução.

Conforme explicado na Seção 6.2, para evitar erros numéricos na estratégia aplicada ao fixar os deslocamentos em regiões vazias, utilizamos o valor $E_{min} = 10^{-9}E_0$ como módulo de Young do vazio, sendo $E_0 = 1N/mm^2$ o módulo de Young do material sólido. Ademais, utilizamos as tolerâncias $\rho^l_{fix} = 10^{-6}$ e $\rho^u_{fix} = 0{,}9$ para encontrar os elementos com densidades próximas de 0 ou 1, respectivamente, e a tolerância $\varepsilon_{grad} = 10^{-6}$ para as componentes do gradiente da função objetivo na escolha das variáveis de projeto que serão fixadas.

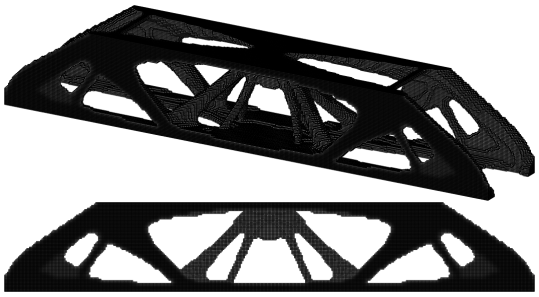
Resolvemos versões mais desafiadoras dos problemas da viga MBB e da viga em balanço, nas quais reduzimos o volume máximo das estruturas para 15% do volume total do domínio, e também testamos o problema da viga em L com 18% do volume. Nos três problemas, utilizamos a multirresolução com $n_{mr} = 4$ e $d_{mr} = 2$. Primeiramente, comparamos as estruturas obtidas na primeira etapa da estratégia adaptativa (com elementos de grau 1) e após a resolução com grau 2, sem fixar nenhuma variável. As Figuras 7.59, 7.60 e 7.61 mostram as estruturas obtidas para os problemas `mbb384x64x64`, `cb192x64x64` e `ls192x192x64`, respectivamente, com diferentes raios do filtro.



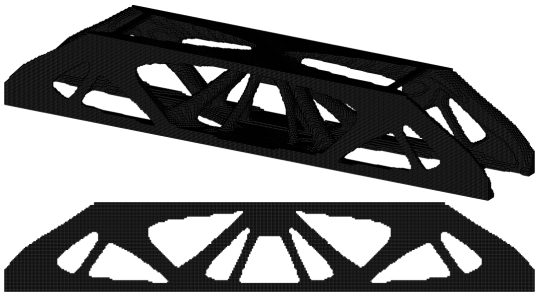
(a) Com elementos de grau 1 ($r_{min} = 1{,}5$)



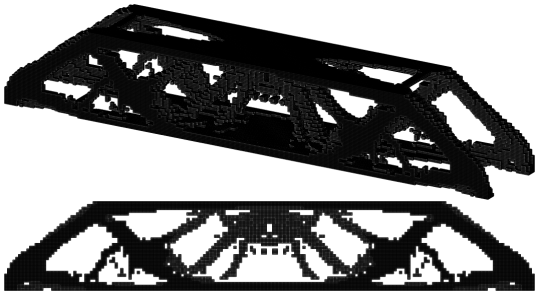
(b) Com a estratégia de aumento do grau 1 para grau 2 e o arredondamento ($r_{min} = 1{,}5$)



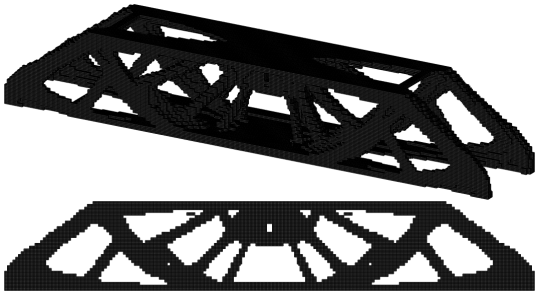
(c) Com elementos de grau 1 ($r_{min} = 1{,}1$)



(d) Com a estratégia de aumento do grau 1 para grau 2 e o arredondamento ($r_{min} = 1{,}1$)

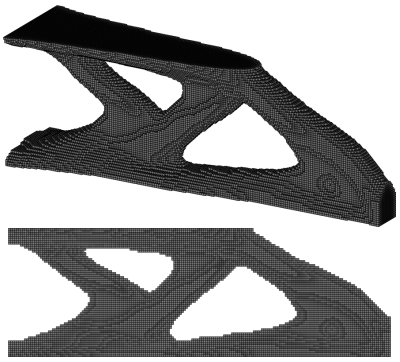


(e) Com elementos de grau 1 ($r_{min} = 0{,}6$)

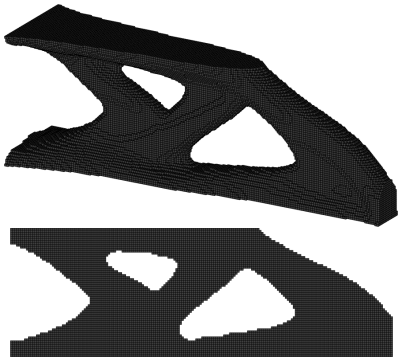


(f) Com a estratégia de aumento do grau 1 para grau 2 e o arredondamento ($r_{min} = 0{,}6$)

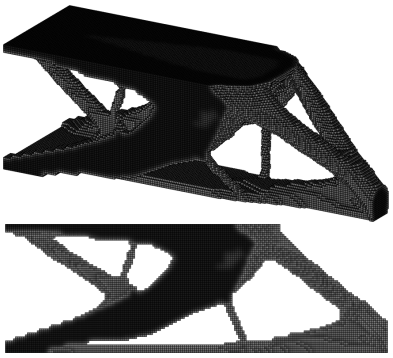
Figura 7.59: Estruturas obtidas para o problema `mbb384x64x64` com 15% do volume, utilizando a multirresolução com $n_{mr} = 4$ e $d_{mr} = 2$.



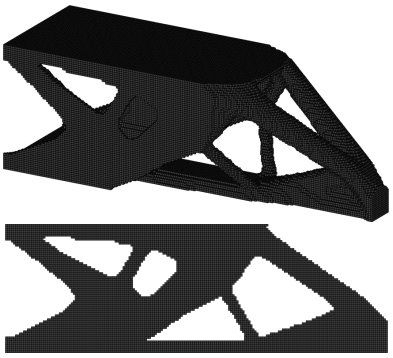
(a) Com elementos de grau 1 ($r_{min} = 1{,}5$)



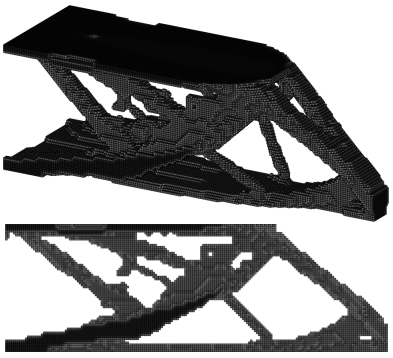
(b) Com a estratégia de aumento do grau 1 para grau 2 e o arredondamento ($r_{min} = 1{,}5$)



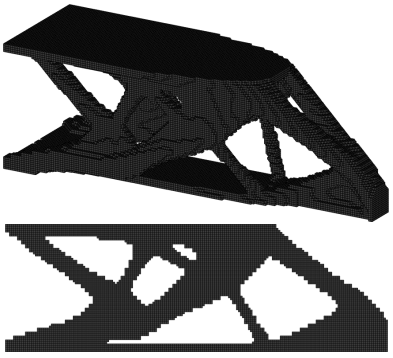
(c) Com elementos de grau 1 ($r_{min} = 1{,}1$)



(d) Com a estratégia de aumento do grau 1 para grau 2 e o arredondamento ($r_{min} = 1{,}1$)



(e) Com elementos de grau 1 ($r_{min} = 0{,}8$)



(f) Com a estratégia de aumento do grau 1 para grau 2 e o arredondamento ($r_{min} = 0{,}8$)

Figura 7.60: Estruturas obtidas para o problema `cb192x64x64` com 15% do volume, utilizando a multirresolução com $n_{mr} = 4$ e $d_{mr} = 2$.

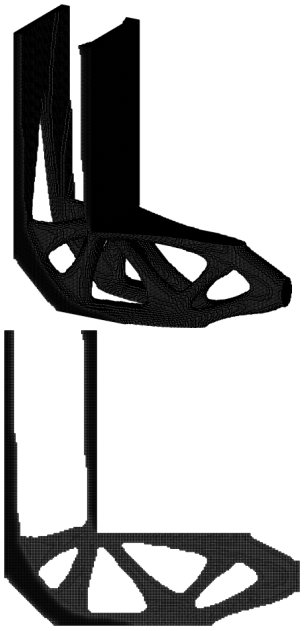
A redução no raio do filtro, de fato, fez com que as estruturas tivessem mais barras e buracos, gerando formatos mais detalhados. Contudo, quando reduzimos o raio excessivamente (geralmente quando $r_{min} < 1{,}0$), a solução obtida com elementos de grau 1 apresentou diversos artefatos. A estratégia de aumento do grau 1 para grau 2 foi capaz de eliminar a maioria desses artefatos, melhorando a qualidade da estrutura e mantendo a essência do formato encontrado na solução inicial.

Na Tabela 7.60, apresentamos os valores da flexibilidade obtidos para cada problema com os diferentes raios do filtro. Utilizamos a seguinte legenda:

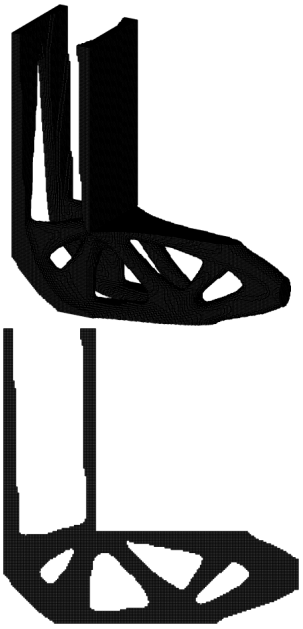
- F_0: valor ótimo da função objetivo para a solução inicial, com elementos de grau 1.

- F: valor ótimo da função objetivo para a solução obtida com a estratégia de aumento do grau 1 até grau 2.
- F_{prj}: valor ótimo da função objetivo para a solução obtida com a estratégia de aumento do grau 1 até grau 2 e densidades arredondadas.

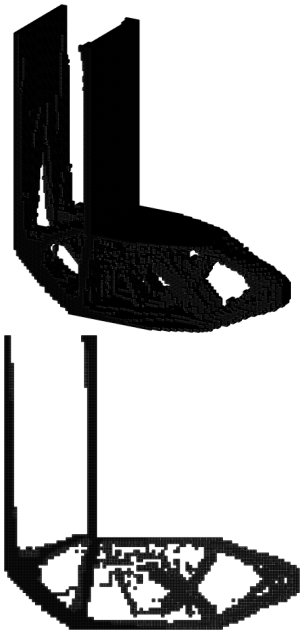
Para permitir uma comparação mais adequada, todos os valores da função foram corrigidos, isto é, recalculados na malha de densidades com elementos lineares.



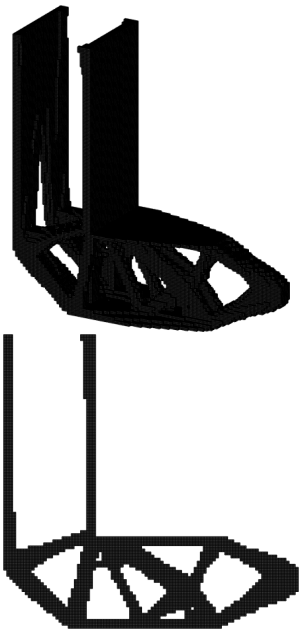
(a) Com elementos de grau 1 ($r_{min} = 1{,}5$)



(b) Com a estratégia de aumento do grau 1 para grau 2 e o arredondamento ($r_{min} = 1{,}5$)



(c) Com elementos de grau 1 ($r_{min} = 0{,}6$)



(d) Com a estratégia de aumento do grau 1 para grau 2 e o arredondamento ($r_{min} = 0{,}6$)

Figura 7.61: Estruturas obtidas para o problema `ls192x192x64` com 18% do volume, utilizando a multirresolução com $n_{mr} = 4$ e $d_{mr} = 2$.

Tabela 7.60: Valor ótimo da função objetivo obtido com a estratégia de aumento do grau.

Problema	r_{min}	F_0	F	F_{prj}
`mbb384x64x64`	1,5	28,9117	28,8230	22,0466
	1,1	26,4703	26,2478	21,5667
	0,6	28,2834	23,8943	21,9514
`cb192x64x64`	1,5	28,3227	28,2484	22,6476
	1,1	29,9472	28,9959	22,8144
	0,8	29,1993	26,9456	22,6302
`ls192x192x64`	1,5	41,4058	40,8706	30,1565
	0,6	60,8895	33,3053	30,4914

O valor da função objetivo para a solução obtida com a estratégia de aumento do grau (F) foi sempre menor do que aquele encontrado para a solução inicial (F_0). Quando usamos raios de filtro maiores que 1,0, a diferença entre esses valores foi relativamente pequena, indicando que a solução obtida com grau 2 se manteve próxima daquela encontrada inicialmente. Com raios menores que 1,0, para os quais a solução inicial continha artefatos, o aumento do grau melhorou bastante a qualidade da estrutura e o valor da flexibilidade ficou bem menor, sobretudo no problema `ls192x192x64`.

Ademais, o valor F foi menor para as estruturas obtidas com os raios de filtro menores que 1,0. Todavia, o mesmo não ocorreu sempre com o valor da função após o arredondamento de densidades (F_{prj}). Em geral, podemos dizer que as estruturas mais detalhadas são tão boas quanto aquelas mais simples e possuem formatos razoáveis. Nos próximos resultados, consideramos apenas as soluções obtidas com raios de filtro menores que 1,0 para os três problemas testados.

A seguir, testamos a estratégia proposta para fixar variáveis, com o intuito de reduzir o tempo gasto na resolução do problema com grau maior que 1. Conforme dito na Subseção 6.2.2, propomos algumas estratégias de adaptação das variáveis que serão fixadas. Apresentamos os resultados para as seguintes estratégias:

- **E0:** não escolher variáveis para serem fixadas.
- **E1:** escolher as variáveis uma vez, antes da resolução do problema com grau maior.
- **E2:** escolher as variáveis em todas as iterações externas da PLS para o problema com grau maior.
- **E3:** escolher as variáveis a cada 5 iterações externas da PLS.
- **E4:** escolher as variáveis no início e na $5^{\underline{a}}$ iteração externa da PLS.

A Tabela 7.61 mostra as quantidades de iterações da PLS e os valores ótimos da função objetivo obtidos com cada estratégia. O número de iterações (N_{it}) e o número de passos recusados (N_{rc}) com sobrescritos 1 e 2 correspondem aos necessários para obter a aproximação inicial com grau 1 e a solução com grau 2, respectivamente. Observamos também que, nos três problemas, foi necessária apenas uma tentativa de arredondamento de densidades, que gastou 3 iterações da PLS.

Tabela 7.61: Resultados obtidos com a estratégia de aumento do grau dos elementos.

Problema	**Estratégia**	N_{it}^1 (N_{rc}^1)	N_{it}^2 (N_{rc}^2)	F	F_{prj}
mbb384x64x64	E0	22 (+1)	32 (+5)	23,8943	21,9514
	E1	22 (+1)	32 (+5)	23,8943	21,9514
	E2	22 (+1)	32 (+5)	23,8943	21,9514
	E3	22 (+1)	32 (+5)	23,8943	21,9514
	E4	22 (+1)	32 (+5)	23,8943	21,9514
cb192x64x64	E0	56 (+11)	102 (+23)	26,9456	22,6302
	E1	56 (+11)	86 (+20)	26,9545	22,6251
	E2	56 (+11)	86 (+20)	26,9545	22,6251
	E3	56 (+11)	86 (+20)	26,9545	22,6251
	E4	56 (+11)	86 (+20)	26,9545	22,6251
ls192x192x64	E0	27 (+0)	32 (+1)	33,3053	30,4914
	E1	27 (+0)	32 (+1)	33,3053	30,4914
	E2	27 (+0)	32 (+1)	33,3053	30,4914
	E3	27 (+0)	32 (+1)	33,3053	30,4914
	E4	27 (+0)	32 (+1)	33,3053	30,4914

Em geral, o número de iterações e o valor da função obtidos com cada estratégia foram iguais, com exceção do problema `cb192x64x64` que gastou menos iterações quando fixamos as variáveis, obtendo, entretanto, um valor de função levemente maior para a solução original (F) e menor para a solução arredondada (F_{prj}). As estruturas obtidas foram praticamente idênticas com as diferentes estratégias.

Na Tabela 7.62, apresentamos os tempos gastos (em segundos) com cada estratégia. Os valores correspondem ao tempo total necessário para obter a solução final, somando-se os tempos gastos na resolução inicial com grau 1, na resolução com grau 2 e no arredondamento de densidades. Mostramos os tempos consumidos nas etapas do algoritmo que efetivamente se alteram quando mudamos a estratégia, utilizando a legenda:

- T_{sp}: tempo gasto na fase de *setup* do precondicionador.
- T_s: tempo gasto para resolver os sistemas lineares.
- T_g: tempo gasto para os cálculos do gradiente da função objetivo.
- T_{lp}: tempo gasto na resolução dos subproblemas de programação linear.
- T_{fx}: tempo gasto na escolha das variáveis fixadas.
- T_{total}: tempo total gasto para resolver o problema.

Observamos que os tempos gastos na pré-filtragem, aplicação do filtro, montagem da matriz de rigidez e demais linhas de código não se alteram efetivamente quando fixamos as variáveis, motivo pelo qual não apresentamos esses valores na tabela.

Tabela 7.62: Tempos gastos com a estratégia de aumento do grau dos elementos.

Problema	**Estratégia**	T_{sp}	T_s	T_g	T_{lp}	T_{fx}	T_{total}
`mbb384x64x64`	E0	28,38	1355,84	32,24	79,44	0,00	1601,05
	E1	18,66	887,60	21,85	67,50	0,26	1102,11
	E2	18,31	885,13	21,57	67,90	5,02	1098,37
	E3	18,73	883,41	21,75	68,38	1,03	1099,91
	E4	18,63	883,11	21,66	67,20	0,38	1096,96
`cb192x64x64`	E0	82,99	3499,81	90,09	230,58	0,00	4193,95
	E1	52,41	2222,71	59,97	197,62	0,31	2779,40
	E2	53,81	2231,21	60,06	196,22	10,86	2797,88
	E3	51,82	2213,69	59,24	196,18	2,29	2768,05
	E4	52,20	2231,68	59,43	197,06	0,35	2786,58
`ls192x192x64`	E0	71,35	3574,63	52,66	208,00	0,00	4196,82
	E1	30,69	1882,14	37,48	193,09	0,86	2421,91
	E2	33,08	1901,82	36,97	188,99	16,19	2455,50
	E3	30,80	1861,87	37,02	188,67	3,50	2399,72
	E4	30,47	1846,67	37,09	188,38	1,31	2381,19

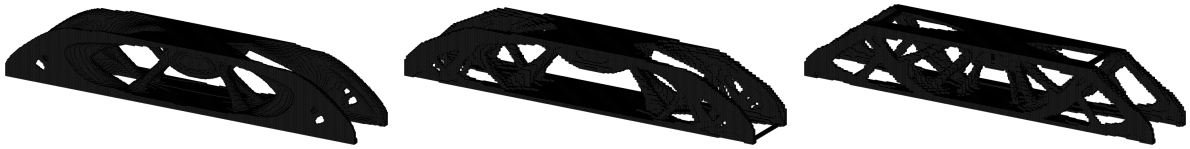
Quando fixamos alguns deslocamentos na resolução do problema com grau 2, as dimensões dos sistemas de equilíbrio diminuem, de modo que os tempos gastos nas etapas de *setup* do precondicionador (T_{sp}) e resolução dos sistemas lineares (T_s) são reduzidos. Além disso, ao fixarmos algumas variáveis de projeto, reduzimos os tempos gastos no cálculo dos gradientes (T_g) e na resolução dos PLs (T_{lp}). Por sua vez, o tempo gasto com a escolha das variáveis fixadas (T_{fx}) foi geralmente pequeno em relação ao tempo total. Logo, a estratégia proposta para fixar variáveis foi bastante eficaz em reduzir o tempo total de resolução dos problemas.

Comparando as diferentes estratégias de adaptação das variáveis fixadas, vemos que todas elas foram igualmente eficientes, sendo que a estratégia E4 foi levemente melhor nos problemas `mbb384x64x64` e `ls192x192x64`, e a estratégia E3 foi levemente melhor no problema `cb192x64x64`. A estratégia E1 pode ser perigosa, uma vez que escolher as variáveis com base apenas na solução inicial pode fazer com que o algoritmo encontre dificuldades para convergir, embora isso não tenha acontecido nos problemas testados. Por outro lado, a estratégia E2, na qual escolhemos as variáveis em todas as iterações, pode fazer com que os tempos T_{sp} e T_{fx} cresçam, prejudicando o tempo total. Com base nesses resultados, acreditamos que as estratégias E3 ou E4 sejam melhores. É possível que variações dessas estratégias sejam ainda mais eficientes. Podemos, por exemplo, refazer a escolha das variáveis em outros intervalos de iterações.

Agora, resolvemos os problemas utilizando o método tradicional com elementos de grau 1 e raios de filtro proporcionais aos utilizados na multirresolução, bem como a multirresolução com elementos de grau 2 desde o início, e comparamos os resultados com aqueles obtidos pela nossa estratégia adaptativa de aumento do grau dos elementos. Nessas comparações, utilizamos a estratégia E4 para a escolha das variáveis fixadas.

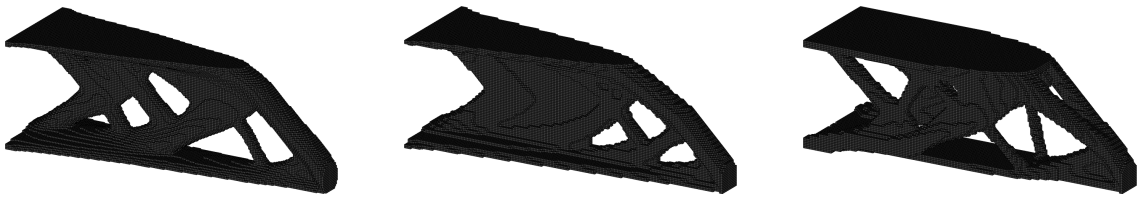
As Figuras 7.62, 7.63 e 7.64 mostram as estruturas obtidas com cada método

para os problemas `mbb384x64x64`, `cb192x64x64` e `ls192x192x64`, respectivamente. Notamos que cada método convergiu para um minimizador local diferente, de maneira que as distribuições de material obtidas não foram iguais. Em geral, as estruturas obtidas pela estratégia de aumento do grau dos elementos foram um pouco mais detalhadas, enquanto aquelas obtidas pela multirresolução com grau 2 tiveram formatos mais simplificados.



(a) Tradicional (b) MR com grau 2 (c) Estratégia proposta

Figura 7.62: Estruturas obtidas para o problema `mbb384x64x64` ($r_{min} = 0{,}6$).



(a) Tradicional (b) MR com grau 2 (c) Estratégia proposta

Figura 7.63: Estruturas obtidas para o problema `cb192x64x64` ($r_{min} = 0{,}8$).

(a) Tradicional (b) MR com grau 2 (c) Estratégia proposta

Figura 7.64: Estruturas obtidas para o problema `ls192x192x64` ($r_{min} = 0{,}6$).

Na Tabela 7.63, comparamos os resultados obtidos com cada método. Para a estratégia de aumento do grau, consideramos a soma do número de iterações necessárias para obter as soluções com grau 1 e com grau 2. Além do valor da função objetivo para a solução original (F), também apresentamos o valor da função para a solução após o arredondamento de densidades (F_{prj}). O tempo total corresponde ao tempo gasto para obter a solução final com densidades arredondadas para 0 ou 1.

O número de iterações foi sempre menor para o método tradicional. Com a multirresolução utilizando elementos de grau 2 desde o início, a quantidade de iterações foi, em geral, mais elevada. A estratégia proposta, na qual resolvemos o problema com grau 2 após obter uma aproximação inicial utilizando elementos de grau 1, conseguiu reduzir o número de iterações, exceto no caso da viga em balanço.

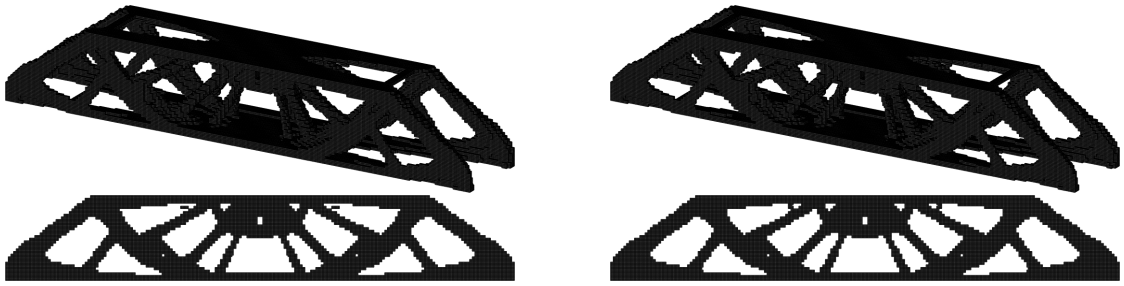
Apesar do número de iterações ser menor, o tempo total de resolução com o método tradicional foi sempre maior. Mesmo utilizando elementos de grau 2, a multirresolução se manteve mais eficiente do que o método tradicional. Com a estratégia proposta,

reduzimos ainda mais o tempo de resolução, sem perder a qualidade da solução. Entretanto, os valores da flexibilidade nem sempre foram menores para as estruturas obtidas com a estratégia proposta.

Tabela 7.63: Comparações entre o método tradicional, a multirresolução com grau 2 e a estratégia adaptativa de aumento do grau dos elementos.

Problema	**Método**	N_{it} (N_{rc})	F	F_{prj}	T_{total}
`mbb384x64x64`	Tradicional	32 (+2)	23,9889	21,3414	5178,87
	MR com grau 2	125 (+6)	24,3325	22,3007	4571,80
	Estratégia proposta	54 (+6)	23,8943	21,9514	1096,96
`cb192x64x64`	Tradicional	46 (+8)	24,7126	21,6733	7351,34
	MR com grau 2	95 (+9)	24,2809	21,5981	3250,48
	Estratégia proposta	142 (+31)	26,9545	22,6251	2786,58
`ls192x192x64`	Tradicional	28 (+1)	29,9054	26,4983	17337,37
	MR com grau 2	72 (+3)	33,9329	31,1160	11304,99
	Estratégia proposta	59 (+1)	33,3053	30,4914	2381,19

Até aqui, testamos a estratégia de aumento do grau dos elementos utilizando no máximo grau 2. Para finalizar, também testamos a estratégia aumentando o grau dos elementos de 1 até 3 para os problemas da viga MBB e da viga em L. Nas Figuras 7.65 e 7.66, comparamos as estruturas obtidas com a estratégia proposta utilizando grau máximo 2 e 3. A Tabela 7.64 apresenta os resultados para esses problemas.

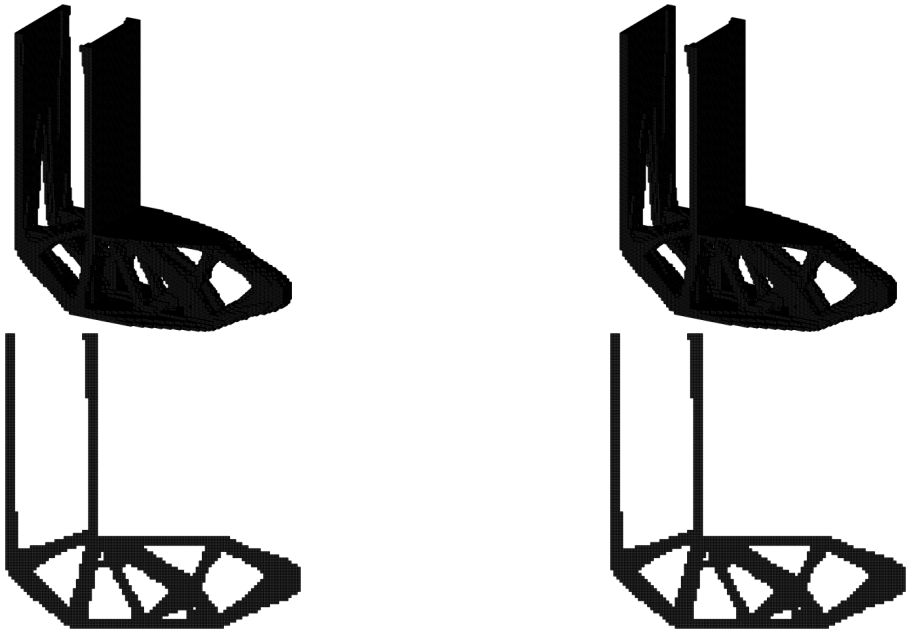


(a) Grau máximo 2 (b) Grau máximo 3

Figura 7.65: Estruturas obtidas para o problema `mbb384x64x64` com 15% do volume, utilizando a multirresolução, $n_{mr} = 4$, $d_{mr} = 2$, $r_{min} = 0{,}6$ e a estratégia adaptativa de aumento do grau dos elementos.

Tabela 7.64: Resultados obtidos com a estratégia de aumento do grau dos elementos.

Problema	**Grau máximo**	N_{it}^1 (N_{rc}^1)	N_{it}^2 (N_{rc}^2)	N_{it}^3 (N_{rc}^3)	F_{prj}	T_{total}
`mbb384x64x64`	2	22 (+1)	32 (+5)	-	21,9514	1096,96
	3	22 (+1)	16 (+0)	21 (+6)	21,9250	8119,33
`ls192x192x64`	2	27 (+0)	32 (+1)	-	30,4914	2381,19
	3	27 (+0)	16 (+0)	37 (+0)	30,4819	15197,55



(a) Grau máximo 2 (b) Grau máximo 3

Figura 7.66: Estruturas obtidas para o problema `ls192x192x64` com 18% do volume, utilizando a multirresolução, $n_{mr} = 4$, $d_{mr} = 2$, $r_{min} = 0{,}6$ e a estratégia adaptativa de aumento do grau dos elementos.

No caso da viga MBB, a resolução com grau 3 foi capaz de corrigir um pequeno artefato remanescente na parte superior central da viga. Contudo, no geral, as mudanças visíveis no formato da estrutura obtida com grau 3 foram pequenas em comparação com aquela obtida com grau máximo 2. Além disso, a redução no valor da flexibilidade foi pouco significativa. Por outro lado, o uso de elementos com grau 3 provocou um aumento significativo no tempo total de resolução, que ficou maior do que aquele obtido ao resolvermos o problema pelo método tradicional no caso da viga MBB.

É possível que para resoluções maiores, ou seja, para valores maiores de n_{mr}, a estratégia proposta com grau máximo 3 continue mais eficiente do que o método tradicional. Porém, conforme vimos, a melhoria na qualidade da solução com o aumento do grau 2 para o grau 3 é relativamente pequena. Em nossos testes, concluímos que o uso de elementos finitos de grau 2 foi suficiente para amenizar o problema dos artefatos nos problemas tridimensionais e para obter estruturas com formatos razoáveis usando raios de filtro consideravelmente pequenos.

A estratégia adaptativa de aumento do grau dos elementos se mostrou bastante eficaz. Conseguimos obter estruturas mais detalhadas e sem artefatos, conforme desejado, gastando menos memória e tempo computacional.

Conclusão

Neste trabalho, estudamos o problema de otimização topológica de estruturas tridimensionais e apresentamos diversos resultados computacionais obtidos com uma implementação feita em Matlab, na qual combinamos técnicas que aceleram a resolução do problema, reduzindo significativamente a memória computacional e o tempo consumidos.

Aplicamos o método dos elementos finitos para discretizar o domínio no qual a estrutura deve estar contida, utilizando elementos que têm o formato de um prisma retangular reto. Na forma discretizada, temos um problema de otimização não linear com restrições, que resolvemos aplicando um método de programação linear sequencial (PLS). Em nossa implementação, ajustamos os parâmetros do algoritmo da PLS e utilizamos um critério de parada baseado nas condições KKT do problema. Os resultados computacionais mostram que, em geral, o método foi capaz de resolver os problemas gastando poucas iterações, com poucas recusas de passo, obtendo topologias ótimas que condizem com a finalidade de cada estrutura, do ponto de vista da engenharia.

A aplicação do filtro da média ponderada das densidades foi eficaz em evitar a formação do chamado "tabuleiro de xadrez" na estrutura, que é uma instabilidade numérica decorrente do tipo de elemento finito utilizado. Além disso, com ajustes no raio de aplicação do filtro, as soluções obtidas foram independentes da malha, ou seja, o formato da estrutura se manteve parecido em diferentes refinamentos. Contudo, notamos que pequenas mudanças no raio do filtro ou nos demais parâmetros do algoritmo podem gerar soluções distintas, devido à presença de muitos mínimos locais próximos.

Analisamos os tempos gastos em cada etapa do algoritmo de resolução do problema e constatamos que o cálculo da função objetivo, que envolve a resolução de um sistema linear relacionado à condição de equilíbrio estático da estrutura, é a etapa mais cara, consumindo mais da metade do tempo total. A matriz desse sistema, denominada matriz de rigidez, é simétrica, definida positiva e esparsa. Em problemas tridimensionais, as dimensões dessa matriz crescem demasiadamente sem que aumentemos muito o número de elementos na malha. Nesse caso, evitamos utilizar a decomposição de Cholesky para resolver o sistema linear, pois o armazenamento do fator de Cholesky demanda muita memória computacional. Utilizamos então o método dos gradientes conjugados, que é mais conveniente em problemas de grande porte. Entretanto, para se obter uma melhor eficácia do método, decidimos combiná-lo com um precondicionador apropriado. Tendo isso em consideração, testamos a aplicação do método *multigrid* e conseguimos reduzir significativamente o tempo gasto na resolução dos sistemas lineares em comparação com a fatoração incompleta de Cholesky, que é comumente utilizada como precondicionador.

Implementamos o *multigrid* geométrico na forma clássica, bem como uma versão do *multigrid* algébrico para problemas estruturais. Realizamos vários testes computacionais alterando a quantidade de malhas, o tipo de ciclo, o suavizador, as quantidades

de iterações da pré e da pós-suavização, além de outros parâmetros no caso do *multigrid* algébrico. Ajustamos cada um desses parâmetros, com a intenção de obter a melhor eficácia possível ao aplicarmos o *multigrid* como precondicionador do método dos gradientes conjugados na resolução dos sistemas lineares de equilíbrio. Tanto o *multigrid* geométrico quanto o algébrico foram bastante eficientes para esse propósito, sendo que a versão geométrica foi melhor. A desvantagem do *multigrid* geométrico é que ele só pode ser aplicado em problemas com malhas regulares e discretizações específicas. Já o *multigrid* algébrico é mais robusto, podendo ser utilizado em qualquer tipo de problema, ainda que o custo de preparação (*setup*) desse método seja elevado.

Com a utilização do *multigrid*, a etapa que mais consome tempo passa a ser a resolução dos subproblemas de programação linear. Considerando as análises de crescimento do tempo conforme aumentamos o número de elementos na malha, constatamos que a resolução dos PLs inspira cautela quando se resolve problemas de grande porte.

Em alguns casos, exploramos as simetrias das estruturas, restringindo as dimensões dos domínios, e conseguimos reduzir a memória consumida no armazenamento da matriz de rigidez, bem como o custo de resolução dos problemas.

Ainda com a intenção de reduzir o custo do cálculo da função objetivo, estudamos os modelos de ordem reduzida, cuja ideia é obter aproximações para as soluções dos sistemas lineares resolvendo sistemas com dimensões menores. Todavia, não conseguimos combinar essa estratégia com a programação linear sequencial, pois o uso das soluções aproximadas no meio do processo de otimização pode tornar incompatível a comparação entre as reduções real e prevista da função de mérito, de modo que o algoritmo encontra dificuldades para convergir.

Em compensação, conseguimos combinar a PLS com uma técnica de multirresolução, utilizando três malhas com refinamentos distintos, de tal forma que conseguimos obter estruturas com resoluções altas reduzindo os custos de todas as etapas da resolução do problema de otimização topológica estrutural. Mantendo os mesmos parâmetros do algoritmo, em geral, obtivemos estruturas parecidas com aquelas encontradas pelo método tradicional. Além disso, os valores da flexibilidade ficaram próximos quando calculados no mesmo cenário, isto é, considerando a mesma malha e as mesmas condições de contorno. Observamos, entretanto, que a comparação entre os métodos não é perfeita, já que os problemas resolvidos não são idênticos. Utilizando uma malha mais grossa para a resolução dos sistemas lineares, uma malha intermediária para a resolução dos subproblemas de programação linear e uma malha mais fina para a distribuição das densidades de material, fomos capazes de obter estruturas tridimensionais com mais de 1 milhão de elementos na malha de densidades, consumindo menos memória e tempo computacional.

Por outro lado, mantendo o raio de aplicação do filtro (usado para a projeção das variáveis de projeto no espaço das densidades), obtivemos soluções independentes da malha, de modo que o aumento na resolução não apresentou novos detalhes no formato da estrutura. Para obter estruturas mais detalhadas com a multirresolução, fomos obrigados a diminuir o raio do filtro. Contudo, ao fazer isso observamos o surgimento de "artefatos", isto é, regiões com o material distribuído de maneira aleatória, contendo partes "flutuando" ou barras incompletas na estrutura. Verificamos que esse inconveniente é decorrente da baixa precisão na aproximação dos deslocamentos e conseguimos amenizá-lo utilizando elementos finitos de grau maior do que 1. Recorrendo à interpolação por meio das funções de forma do método dos elementos finitos, conseguimos construir o operador

de prolongação e aplicar o método *multigrid* geométrico com elementos do tipo Lagrange e do tipo *serendipity* de grau 2 e 3. Atestamos que é mais vantajoso utilizar elementos do tipo Lagrange até o grau 2 e do tipo *serendipity* com grau 3 ou maior. Independentemente do tipo de elemento, o aumento no grau faz com que a resolução dos sistemas lineares volte a ser a etapa mais cara do algoritmo, causando um crescimento considerável no tempo de resolução do problema. Felizmente, nos problemas tridimensionais, o uso de elementos com grau 2 ou 3 foi satisfatório para eliminar os artefatos na estrutura.

Para amenizar o crescimento do tempo de resolução quando utilizamos elementos de grau maior do que 1, propusemos uma estratégia adaptativa de aumento do grau dos elementos, na qual suprimimos algumas variáveis do problema, mantendo seus valores fixados. Essa estratégia se mostrou bastante eficiente e conseguimos obter estruturas com mais detalhes, sem a presença de artefatos, mantendo a multirresolução mais eficiente do que o método tradicional mesmo com o uso de elementos de grau 2 ou, em alguns casos, até de grau 3. Salientamos, entretanto, que o uso de elementos com grau maior do que 3 pode tornar o algoritmo ineficiente.

Ademais, propusemos uma nova estratégia de arredondamento de densidades, baseada em informações do gradiente do Lagrangiano do problema, com a qual conseguimos reduzir substancialmente a quantidade de elementos com densidades intermediárias e obter soluções binárias que estão próximas de um minimizador local, sem violar a restrição de volume, com um acréscimo no custo computacional relativamente pequeno.

Comprovamos que nossa implementação é robusta e permite obter uma boa solução para o problema de otimização topológica estrutural, cumprindo o objetivo de fornecer um formato inicial adequado para a estrutura. Combinando a programação linear sequencial, o *multigrid*, a multirresolução e as novas estratégias propostas, conseguimos obter estruturas tridimensionais em alta resolução, contendo mais detalhes, com um baixo consumo de memória e de tempo computacional.

Por fim, sugerimos alguns possíveis desdobramentos deste trabalho:

- Combinar as técnicas propostas para resolver problemas com restrições de manufatura aditiva [60, 63], permitindo que seja possível reproduzir modelos das estruturas com a impressão 3D para a realização de testes físicos;
- Resolver outros tipos de problemas de otimização topológica, como os de mecanismos flexíveis [23, 44, 62];
- Resolver problemas com malhas irregulares ou elementos com outros formatos (tetraédricos, por exemplo), aplicando o *multigrid* algébrico;
- Comparar outros tipos de suavizadores do *multigrid* (Gauss-Seidel *red-black*, por exemplo), bem como outros tipos de ciclo (*full multigrid*, por exemplo);
- Persistir na tentativa de combinar a PLS com os métodos de redução de ordem, fazendo as ressalvas necessárias;
- Utilizar alternativas para as funções `sparse` (como proposto em [19]) e `linprog` do Matlab, com o objetivo de acelerar a montagem da matriz de rigidez e a resolução dos subproblemas de programação linear, respectivamente. Como opção, implementar o algoritmo utilizando outra linguagem de programação de alto nível.

Referências Bibliográficas

[1] AMIR, O.; AAGE, N.; LAZAROV, B. S. On multigrid-CG for efficient topology optimization. *Structural and Multidisciplinary Optimization*, v. 49, n. 5, p. 815–829, 2014. DOI: 10.1007/s00158-013-1015-5.

[2] ANDREASSEN, E.; CLAUSEN, A.; SCHEVENELS, M.; LAZAROV, B. S.; SIGMUND, O. Efficient topology optimization in MATLAB using 88 lines of code. *Structural and Multidisciplinary Optimization*, v. 43, n. 1, p. 1–16, 2011. DOI: 10.1007/s00158-010-0594-7.

[3] ASSAN, A. E. *Método dos Elementos Finitos - Primeiros Passos*. 2. ed. Campinas: Editora da Unicamp, 2003. ISBN 85-268-0623-8.

[4] BENDSØE, M. P. Optimal shape design as a material distribution problem. *Structural Optimization*, v. 1, n. 4, p. 193–202, 1989. DOI: 10.1007/BF01650949.

[5] BENDSØE, M. P.; KIKUCHI, N. Generating optimal topologies in structural design using a homogenization method. *Computer Methods in Applied Mechanics and Engineering*, v. 71, n. 2, p. 197–224, 1988. DOI: 10.1016/0045-7825(88)90086-2.

[6] BENDSØE, M. P.; SIGMUND, O. *Topology Optimization: Theory, Methods and Applications*. Berlin: Springer-Verlag, 2004. ISBN 978-3-662-05086-6.

[7] BERGAMASCHI, L.; GAMBOLATI, G.; PINI, G. Spectral analysis of large finite element problems by optimization methods. *Shock and Vibration*, v. 1, n. 6, p. 529–540, 1994. DOI: 10.3233/SAV-1994-1603.

[8] BERGAMASCHI, L.; MARTÍNEZ, A.; PINI, G. Parallel Rayleigh quotient optimization with FSAI-based preconditioning. *Journal of Applied Mathematics*, v. 2012, p. 1–14, 2012. DOI: 10.1155/2012/872901.

[9] BOURDIN, B. Filters in topology optimization. *International Journal for Numerical Methods in Engineering*, v. 50, n. 9, p. 2143–2158, 2001. DOI: 10.1002/nme.116.

[10] BRIGGS, W. L.; HENSON, V. E.; MCCORMICK, S. F. *A Multigrid Tutorial*. 2. ed. Philadelphia: Society for Industrial and Applied Mathematics, 2000. ISBN 978-0-89871-462-3.

[11] BRUNS, T. E. A reevaluation of the SIMP method with filtering and an alternative formulation for solid-void topology optimization. *Structural and Multidisciplinary Optimization*, v. 30, n. 6, p. 428–436, 2005. DOI: 10.1007/s00158-005-0537-x.

[12] BRUNS, T. E.; TORTORELLI, D. A. An element removal and reintroduction strategy for the topology optimization of structures and compliant mechanisms. *International Journal for Numerical Methods in Engineering*, v. 57, n. 10, p. 1413–1430, 2003. DOI: 10.1002/nme.783.

[13] CHOI, Y.; OXBERRY, G.; WHITE, D.; KIRCHDOERFER, T. Accelerating design optimization using reduced order models. *arXiv preprint*, 2019. DOI: 10.48550/ARXIV.1909.11320.

[14] COOK, R. D.; MALKUS, D. S.; PLESHA, M. E.; WITT, R. J. *Concepts and Applications of Finite Element Analysis*. 4. ed. New York: Wiley, 2001. ISBN 978-0-471-35605-9.

[15] DEATON, J. D.; GRANDHI, R. V. A survey of structural and multidisciplinary continuum topology optimization: Post 2000. *Structural and Multidisciplinary Optimization*, v. 49, n. 1, p. 1–38, 2014. DOI: 10.1007/s00158-013-0956-z.

[16] DÍAZ, A.; SIGMUND, O. Checkerboard patterns in layout optimization. *Structural Optimization*, v. 10, n. 1, p. 40–45, 1995. DOI: 10.1007/BF01743693.

[17] DIJK, N. P. van; MAUTE, K.; LANGELAAR, M.; KEULEN, F. van. Level-set methods for structural topology optimization: A review. *Structural and Multidisciplinary Optimization*, v. 48, n. 3, p. 437–472, 2013. DOI: 10.1007/s00158-013-0912-y.

[18] ESCHENAUER, H. A.; OLHOFF, N. Topology optimization of continuum structures: A review. *Applied Mechanics Reviews*, v. 54, n. 4, p. 331–390, 2001. DOI: 10.1115/1.1388075.

[19] FERRARI, F.; SIGMUND, O. A new generation 99 line Matlab code for compliance topology optimization and its extension to 3D. *Structural and Multidisciplinary Optimization*, v. 62, n. 4, p. 2211–2228, 2020. DOI: 10.1007/s00158-020-02629-w.

[20] FRANCESCHINI, A.; MAGRI, V. A. P.; MAZZUCCO, G.; SPIEZIA, N.; JANNA, C. A robust adaptive algebraic multigrid linear solver for structural mechanics. *Computer Methods in Applied Mechanics and Engineering*, v. 352, p. 389–416, 2019. DOI: 10.1016/j.cma.2019.04.034.

[21] GOGU, C. Improving the efficiency of large scale topology optimization through on-the-fly reduced order model construction. *International Journal for Numerical Methods in Engineering*, v. 101, n. 4, p. 281–304, 2015. DOI: 10.1002/nme.4797.

[22] GOMES, F. A. M.; SENNE, T. A. An SLP algorithm and its application to topology optimization. *Computational and Applied Mathematics*, v. 30, n. 1, p. 53–89, 2011. DOI: 10.1590/S1807-03022011000100001.

[23] GOMES, F. A. M.; SENNE, T. A. An algorithm for the topology optimization of geometrically nonlinear structures. *International Journal for Numerical Methods in Engineering*, v. 99, n. 6, p. 391–409, 2014. DOI: 10.1002/nme.4686.

[24] GROEN, J. P.; LANGELAAR, M.; SIGMUND, O.; RUESS, M. Higher-order multiresolution topology optimization using the finite cell method. *International Journal for Numerical Methods in Engineering*, v. 110, n. 10, p. 903–920, 2017. DOI: 10.1002/nme.5432.

[25] GUPTA, D. K.; KEULEN, F. van; LANGELAAR, M. Design and analysis adaptivity in multiresolution topology optimization. *International Journal for Numerical Methods in Engineering*, v. 121, n. 3, p. 450–476, 2020. DOI: 10.1002/nme.6217.

[26] GUPTA, D. K.; LANGELAAR, M.; KEULEN, F. van. QR-patterns: Artefacts in multiresolution topology optimization. *Structural and Multidisciplinary Optimization*, v. 58, n. 4, p. 1335–1350, 2018. DOI: 10.1007/s00158-018-2048-6.

[27] GUPTA, D. K.; VEEN, G. J. van der; ARAGÓN, A. M.; LANGELAAR, M.; KEULEN, F. van. Bounds for decoupled design and analysis discretizations in topology optimization. *International Journal for Numerical Methods in Engineering*, v. 111, n. 1, p. 88–100, 2017. DOI: 10.1002/nme.5455.

[28] HASLINGER, J.; MÄKINEN, R. A. E. *Introduction to Shape Optimization: Theory, Approximation, and Computation*. Philadelphia: Society for Industrial and Applied Mathematics, 2003. ISBN 978-0-89871-536-1.

[29] HIBBELER, R. C. *Resistência dos Materiais*. 7. ed. São Paulo: Pearson Education do Brasil, 2010. ISBN 978-85-7605-373-6.

[30] LIU, K.; TOVAR, A. An efficient 3D topology optimization code written in Matlab. *Structural and Multidisciplinary Optimization*, v. 50, n. 6, p. 1175–1196, 2014. DOI: 10.1007/s00158-014-1107-x.

[31] LONGSINE, D. E.; MCCORMICK, S. F. Simultaneous Rayleigh-quotient minimization methods for Ax=λBx. *Linear Algebra and its Applications*, v. 34, p. 195–234, 1980. DOI: 10.1016/0024-3795(80)90166-4.

[32] LUENBERGER, D. G.; YE, Y. *Linear and Nonlinear Programming*. 3. ed. New York: Springer, 2008. ISBN 978-0-387-74503-9.

[33] MAGRI, V. A. P.; FRANCESCHINI, A.; JANNA, C. A novel algebraic multigrid approach based on adaptive smoothing and prolongation for ill-conditioned systems. *SIAM Journal on Scientific Computing*, v. 41, n. 1, p. 190–219, 2019. DOI: 10.1137/17M1161178.

[34] MARTÍNEZ, J. M. A note on the theoretical convergence properties of the SIMP method. *Structural and Multidisciplinary Optimization*, v. 29, n. 4, p. 319–323, 2005. DOI: 10.1007/s00158-004-0479-8.

[35] MUKHERJEE, S.; LU, D.; RAGHAVAN, B.; BREITKOPF, P.; DUTTA, S.; XIAO, M.; ZHANG, W. Accelerating large-scale topology optimization: State-of-the-art and challenges. *Archives of Computational Methods in Engineering*, v. 28, n. 7, p. 4549–4571, 2021. DOI: 10.1007/s11831-021-09544-3.

[36] NGUYEN, T. H.; LE, C. H.; HAJJAR, J. F. Topology optimization using the p-version of the finite element method. *Structural and Multidisciplinary Optimization*, v. 56, n. 3, p. 571–586, 2017. DOI: 10.1007/s00158-017-1675-7.

[37] NGUYEN, T. H.; PAULINO, G. H.; SONG, J.; LE, C. H. A computational paradigm for multiresolution topology optimization (MTOP). *Structural and Multidisciplinary Optimization*, v. 41, n. 4, p. 525–539, 2010. DOI: 10.1007/s00158-009-0443-8.

[38] NGUYEN, T. H.; PAULINO, G. H.; SONG, J.; LE, C. H. Improving multiresolution topology optimization via multiple discretizations. *International Journal for Numerical Methods in Engineering*, v. 92, n. 6, p. 507–530, 2012. DOI: 10.1002/nme.4344.

[39] PARK, J.; SUTRADHAR, A. A multi-resolution method for 3D multi-material topology optimization. *Computer Methods in Applied Mechanics and Engineering*, v. 285, p. 571–586, 2015. DOI: 10.1016/j.cma.2014.10.011.

[40] PEETZ, D.; ELBANNA, A. On the use of multigrid preconditioners for topology optimization. *Structural and Multidisciplinary Optimization*, v. 63, n. 2, p. 835–853, 2021. DOI: 10.1007/s00158-020-02750-w.

[41] RIBEIRO, A. A.; KARAS, E. W. *Otimização Contínua*. São Paulo: Cengage Learning, 2013. ISBN 978-85-221-2002-4.

[42] RIETZ, A. Sufficiency of a finite exponent in SIMP (power law) methods. *Structural and Multidisciplinary Optimization*, v. 21, n. 2, p. 159–163, 2001. DOI: 10.1007/s001580050180.

[43] SENNE, T. A.; GOMES, F. A. M.; SANTOS, S. A. On the approximate reanalysis technique in topology optimization. *Optimization and Engineering*, v. 20, n. 1, p. 251–275, 2019. DOI: 10.1007/s11081-018-9408-3.

[44] SIGMUND, O. On the design of compliant mechanisms using topology optimization. *Mechanics of Structures and Machines*, v. 25, n. 4, p. 493–524, 1997. DOI: 10.1080/08905459708945415.

[45] SIGMUND, O. A 99 line topology optimization code written in Matlab. *Structural and Multidisciplinary Optimization*, v. 21, n. 2, p. 120–127, 2001. DOI: 10.1007/s001580050176.

[46] SIGMUND, O. On benchmarking and good scientific practise in topology optimization. *Structural and Multidisciplinary Optimization*, v. 65, n. 11, p. 315–325, 2022. DOI: 10.1007/s00158-022-03427-2.

[47] SIGMUND, O.; MAUTE, K. Topology optimization approaches: A comparative review. *Structural and Multidisciplinary Optimization*, v. 48, n. 6, p. 1031–1055, 2013. DOI: 10.1007/s00158-013-0978-6.

[48] SIGMUND, O.; PETERSSON, J. Numerical instabilities in topology optimization: A survey on procedures dealing with checkerboards, mesh-dependencies and local minima. *Structural Optimization*, v. 16, n. 1, p. 68–75, 1998. DOI: 10.1007/BF01214002.

[49] STÜBEN, K. An introduction to algebraic multigrid. In: TROTTENBERG, U.; OOSTERLEE, C. W.; SCHÜLLER, A. (Ed.). *Multigrid*. San Diego: Academic Press, 2001. ISBN 978-0-12-701070-0.

[50] SUNDAR, H.; STADLER, G.; BIROS, G. Comparison of multigrid algorithms for high-order continuous finite element discretizations. *Numerical Linear Algebra with Applications*, v. 22, n. 4, p. 664–680, 2015. DOI: 10.1002/nla.1979.

[51] SVANBERG, K. The method of moving asymptotes - a new method for structural optimization. *International Journal for Numerical Methods in Engineering*, v. 24, n. 2, p. 359–373, 1987. DOI: 10.1002/nme.1620240207.

[52] TROTTENBERG, U.; OOSTERLEE, C. W.; SCHÜLLER, A. *Multigrid*. San Diego: Academic Press, 2001. ISBN 978-0-12-701070-0.

[53] VITORINO, A. *Otimização Topológica de Estruturas Tridimensionais*. Dissertação de Mestrado em Matemática Aplicada — Universidade Estadual de Campinas, Campinas, 2019. DOI: 10.47749/T/UNICAMP.2019.1088809.

[54] WANG, F.; LAZAROV, B. S.; SIGMUND, O. On projection methods, convergence and robust formulations in topology optimization. *Structural and Multidisciplinary Optimization*, v. 43, n. 6, p. 767–784, 2011. DOI: 10.1007/s00158-010-0602-y.

[55] WANG, Y.; LIAO, Z.; YE, M.; ZHANG, Y.; LI, W.; XIA, Z. An efficient isogeometric topology optimization using multilevel mesh, MGCG and local-update strategy. *Advances in Engineering Software*, v. 139, p. 102733, 2020. DOI: 10.1016/j.advengsoft.2019.102733.

[56] WATKINS, D. S. *Fundamentals of Matrix Computations*. 2. ed. New York: Wiley-Interscience, 2002. ISBN 978-0-471-21394-9.

[57] XIAO, M.; LU, D.; BREITKOPF, P.; RAGHAVAN, B.; DUTTA, S.; ZHANG, W. On-the-fly model reduction for large-scale structural topology optimization using principal components analysis. *Structural and Multidisciplinary Optimization*, v. 62, n. 1, p. 209–230, 2020. DOI: 10.1007/s00158-019-02485-3.

[58] XU, S.; CAI, Y.; CHENG, G. Volume preserving nonlinear density filter based on heaviside functions. *Structural and Multidisciplinary Optimization*, v. 41, n. 4, p. 495–505, 2010. DOI: 10.1007/s00158-009-0452-7.

[59] YOO, J.; JANG, I. G.; LEE, I. Multi-resolution topology optimization using adaptive isosurface variable grouping (MTOP-aIVG) for enhanced computational efficiency. *Structural and Multidisciplinary Optimization*, v. 63, n. 4, p. 1743–1766, 2021. DOI: 10.1007/s00158-020-02774-2.

[60] ZEGARD, T.; PAULINO, G. H. Bridging topology optimization and additive manufacturing. *Structural and Multidisciplinary Optimization*, v. 53, n. 1, p. 175–192, 2016. DOI: 10.1007/s00158-015-1274-4.

[61] ZHANG, W.; LI, D.; YUAN, J.; SONG, J.; GUO, X. A new three-dimensional topology optimization method based on moving morphable components (MMCs). *Computational Mechanics*, v. 59, n. 4, p. 647–665, 2017. DOI: 10.1007/s00466-016-1365-0.

[62] ZHU, B.; ZHANG, X.; ZHANG, H.; LIANG, J.; ZANG, H.; LI, H.; WANG, R. Design of compliant mechanisms using continuum topology optimization: A review. *Mechanism and Machine Theory*, v. 143, p. 103622, 2020. DOI: 10.1016/j.mechmachtheory.2019.103622.

[63] ZHU, J.; ZHOU, H.; WANG, C.; ZHOU, L.; YUAN, S.; ZHANG, W. A review of topology optimization for additive manufacturing: Status and challenges. *Chinese Journal of Aeronautics*, v. 34, n. 1, p. 91–110, 2021. DOI: 10.1016/j.cja.2020.09.020.