ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

Nota final
9,8 (nove e oito)
22/01/04

DESENVOLVIMENTO DE SOFTWARE DE OTIMIZAÇÃO TOPOLÓGICA PARA ESTRUTURAS TRIDIMENSIONAIS

Aluno: *James Edward Shooter*
Orientador: *Prof. Dr. Emílio Carlos Nelli Silva*

PMR-2500
Trabalho de Formatura
2º Semestre de 2003

RESUMO

A otimização topológica é uma ciência que vêm sendo estudada cada vez mais, e embora ainda seja um campo de pesquisa recente, já apresenta resultados promissores. O presente trabalho vêm a acrescentar uma ferramenta para o estudo de estruturas ótimas, obtidas através da teoria de otimização topológica que não somente permite a obtenção de estruturas tridimensionais com alta rigidez e baixo peso (estruturas obtidas neste trabalho) como também permite a síntese de materiais com propriedades especificadas através da criação de microestruturas. Apesar do último tema não ser abordado neste trabalho, ele consiste em uma extensão da teoria aqui apresentada e pode, juntamente com o programa aqui desenvolvido, formar uma plataforma de otimização em trabalhos futuros. Este trabalho se limita à síntese de estruturas tridimensionais com elevada relação rigidez/volume. Os resultados obtidos mostram que estruturas utilizando muito menos material e que apresentam alta rigidez podem ser fabricadas, o que juntamente com a necessidade cada vez maior de redução de custos e de novos materiais justifica esse e outros trabalhos na área de otimização topológica.

O programa desenvolvido neste trabalho utiliza para a síntese das estruturas ótimas um algoritmo heurístico baseado nas condições de optimalidade do problema proposto. O programa também utiliza a teoria de elementos finitos para a discretização do domínio de projeto. O número de elementos utilizados em tal discretização é um fator determinante da qualidade dos resultados obtidos, sendo que uma estrutura de dados visando maximizar o uso de memória para o armazenamento da matriz de rigidez global foi desenvolvido para que fosse possível o uso de malhas com um maior número de elementos.

ABSTRACT

Topology optimization is a field that has been studied more and more, and although still a very recent research field, the results obtained are promising. This work is a tool in studying optimal structures obtained trough topology optimization, which can be used not only in obtaining high stiffness and low weight structures (as in this work) but also in obtaining materials with tailored physical properties such as thermal conductivity, the homogenization method. Although not present in this work, the homogenization method is used in obtaining optimal microstructures, and may be used together with the software developed in this work in a optimization platform. The results obtained in this work limit themselves to high stiffness and low weight tridimensional structures. The obtained results show that very light and stiff tridimensional structures can be designed. This fact, together with the increasing need for materials that are cheap and that have specified physical properties justify this and other works in topology optimization.

The developed program in this work uses a heuristic algorithm based in the problems optimality criteria. The program also uses the finite element method in determining the meshes nodal displacements. The number of finite elements used in the domain is a limiting factor in the quality of results. A data structure seeking to maximize computer memory usage in storing the global stiffness matrix was developed, permitting the use of meshes with a large number of finite elements.

SUMÁRIO

LISTA DE FIGURAS 5
1.Introdução 6
1.1.Aplicações da Otimização Topológica 9
1.1.1.Síntese de Estruturas Mecânicas 9
1.1.2.Síntese de Microeletromecanismos (MicroEletroMechanisms - MEMS) 9
1.1.3.Síntese de materiais com propriedades extremas 10
2.OBJETIVOS 12
3.FUNDAMENTOS TEÓRICOS 14
3.1.Formulação do problema de Otimização 14
3.1.1.Energia Mútua e Flexibilidade Média 14
3.1.2. Formulação da Função Objetivo para um Caso de Carregamento 16
3.1.3.Modelo de Material Utilizado 17
3.2.Critério de Optimalidade 20
3.3.Formulação para casos de múltiplos carregamentos 22
3.4.Método dos Elementos Finitos 24
3.5.Formulação Discreta 26
3.6.Filtro de Densidades e o "Tabuleiro de Xadrez" 27
4.IMPLEMENTAÇÃO 30
5.RESULTADOS 35
6.CONCLUSÕES 40
ANEXO 41
7.REFERÊNCIAS BIBLIOGRÁFICAS 53

LISTA DE FIGURAS

Figura 1 - Estrutura típica de um programa de otimização estrutural.....7
Figura 2 - Categorias de Otimização.....8
Figura 3 - Otimização do braço de suspensão de um caminhão.....9
Figura 4 - Síntese de um MEM.....10
Figura 5 - CAO (Computer Aided Optimizer).....12
Figura 6 - Etapas na obtenção de uma estrutura ótima.....13
Figura 7 - Corpo submetido a forças de superfície e forças de volume.....14
Figura 8 - Domínio de projeto sujeito a diferentes casos de carregamento.....23
Figura 9 - Elemento isoparamétrico de 8 nós.....26
Figura 10 - Tabuleiro de xadrez e filtro de sensibilidades.....28
Figura 11 - Estrutura de dados para geração da matriz esparsa.....31
Figura 12 - Fluxo principal de dados no programa.....34
Figura 13 - Fluxo de dados na subrotina de MEF.....34
Figura 14 - Viga engastada sujeita a carregamento externo.....35
Figura 15 - Topologia obtida para o exemplo 1..... 35
Figura 16 - Gráficos de convergência para o exemplo 1.....36
Figura 17 - Viga engastada sujeita a 2 carregamentos diferentes.....36
Figura 18 - Topologia obtida para o exemplo 2.....36
Figura 19 - Gráficos de convergência para o problema 2.....37
Figura 20 - Problema da "ponte".....37
Figura 21 - Topologia obtida para o exemplo 3.....37
Figura 22 - Gráficos de convergência para o problema 3.....37
Figura 20 - Problema 4.....38
Figura 21 - Topologia obtida para o exemplo 4..... 38
Figura 22 - Gráficos de convergência para o problema 4.....38
Figura 23 - Problema da solda.....39
Figura 24 - Topologia obtida para o problema da solda.....39
Figura 25 - Gráficos de convergência para o problema da solda.....39

1.Introdução

Devido à crescente necessidade de redução de custos e de melhor aproveitamento de recursos disponíveis para a produção de estruturas mecânicas, ou seja, melhor aproveitamento do material disponível, as indústrias cada vez mais vêm necessitando soluções de compromisso que otimizem um determinado aspecto da produção e que ao mesmo tempo não violem restrições impostas ao processo. Tendo em vista a complexidade envolvida na obtenção de tal solução quando estruturas mecânicas de grandes proporçõs e/ou de geometria complexa estão envolvidas, a necessidade do uso de programas e sistemas computacionais de otimização se torna evidente. Consideremos por exemplo a tarefa de projetar a estrutura mecânica de uma aeronave;nesse caso, o peso da estrutura deve ser o menor possível para que ela seja economicamente viável (menor uso de material e combustível). Ao mesmo tempo, no entanto, a estrutura deve ser resistente o suficiente para que possa carregar mais passageiros e oferecer-lhes maior segurança, e deve também satisfazer diversas restrições impostas por motivos estéticos, resposta da estrutura a diversos tipos de excitação, custo e manufatura. A tarefa descrita não é trivial e necessita, além da experiência de engenheiros bem treinados, de uma ferramenta computacional que auxilie na obtenção da solução. Consequentemente, a *Otimização Estrutural* é um campo de pesquisa que vêm sofrendo avanços e crescendo significativamente nos últimos anos.

Os programas de otimização estrutural são tipicamente compostos de um *módulo de análise* e de um *módulo de otimização*. O módulo de análise é utilizado para o cálculo da resposta da estrutura, fornecendo informações como deflexões, deformações e tensões no caso de uma análise estática ou frequências naturais e modos de vibrar da estrutura no caso de uma análise dinâmica. O módulo de análise também é utiliado para efetuar uma análise de sensibilidades, onde é calculada a alteração da resposta de uma estrutura devido à alteração de um ou mais parâmetros envolvidos no processo de otimização.Baseado nas informações a respeito das sensibilidades, o módulo de otimização calcula uma variação nos parâmetros que melhore a resposta da estrutura.Esse processo consistindo na análise, cálculo de sensibilidades e otimização ilustrado no

flufograma da figura 1.1 é então repetido várias vezes, até que nenhuma melhora possa ser obtida na resposta da estrutura, indicando que os parâmetros ótimos foram obtidos.

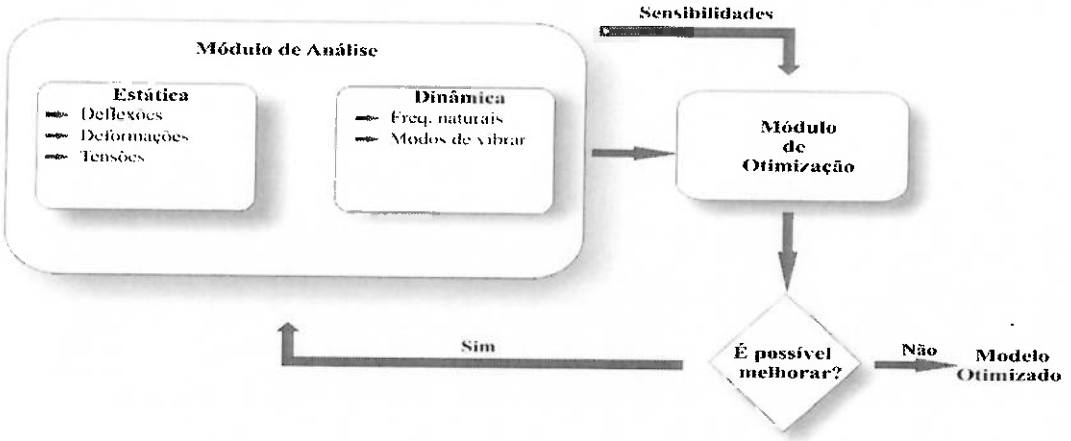


Figura 1.1. Estrutura típica de um programa de otimização estrutural

A otimização estrutural pode ser dividida em quatro categorias principais, a otimização de tamanho, otimização de forma, otimização de material e otimização topológica, descritos brevemente a seguir:

- **Otimização de Tamanho ou Paramétrica** → Na otimização paramétrica, a configuração da estrutura é pré-determinada. Um exemplo de otimização de tamanho é o problema de maximizar a rigidez de uma estrutura de treliças onde as variáveis de interesse são as áreas das secções transversais das barras que à compõem. É a forma mais simples de se otimizar uma estrutura.Na figura 1.2. A a estrutura de treliças é otimizada, sendo que a área de cada barra é uma variável de projeto.A otimização resulta numa estrutura em que várias das barras consideradas inicialmente são descartadas, ou seja, a otimização determinou uma área nula para as suas secções transversais.
- Otimização de Forma → Otimização de forma. Na otimixação de forma, a topologia da estrutura é pré-determinada, de forma que não há a possibilidade de se alterá-la introduzindo novos buracos na estrutura. A forma da estrutura é a variável de interesse nesse caso, podendo se alterar por exemplo a forma de buracos já existentes na estrutura. A otimização de forma é ilustrada na figura 1.2.B, onde a forma dos buracos existentes na estrutura inicial são alterados pelo processo.

- Otimização de Material →Na otimização de material, o objetivo é encontrar uma composição de materiais que otimize a função objetivo. Um exemplo de otimização de material é o problema de maximizar a rigidez de uma viga, ilustrada na figura 1.2.C. As variáveis de interesse nesse caso são a espessura e orientação das camadas material utilizados na viga.
- Otimização Topológica → Finalmente, a otimização topológica é a categoria de otimização que produz os melhores resultados, o que é evidente se considerarmos que em todas as outras categorias a topologia da estrutura é fixa, ou seja, no caso da otimização de tamanho o número de barras e a conectividade das barras da estrutura de treliças não pode ser alterada, no caso da otimização de material a estrutura é uma viga simples e não pode ser alterada e no caso da otimização de forma o número de buracos na estrutura não pode ser alterado. Já no caso da otimização topológica, além de ser possível alterar a forma dos buracos pré-existentes, também é possível introduzir novos buracos ou remover buracos já existentes, fornecendo uma gama maior de estruturas possíveis. A figura 1.2.D ilustra o resultado da otimização topológica de uma viga simplesmente apoiada.

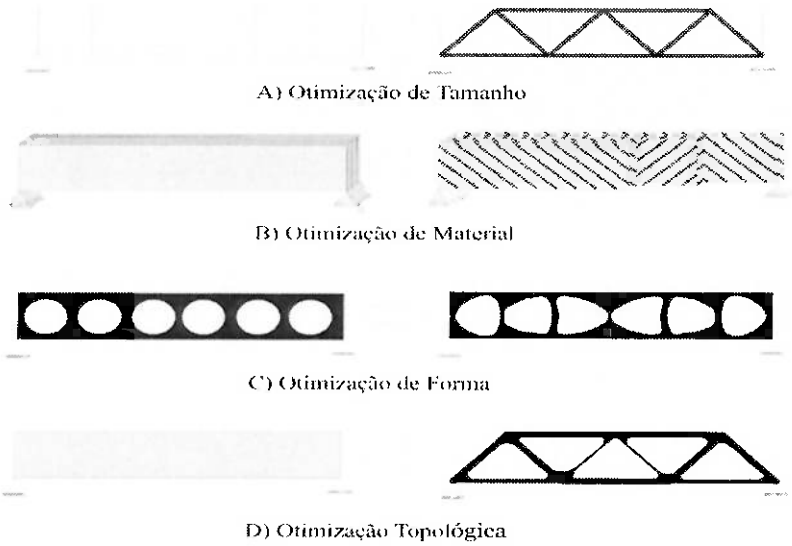


Figura 1.2. Categorias de Otimização

1.1.Aplicações da Otimização Topológica

A otimização topológica de estruturas mecânicas através do método de otimização topológica tem aplicações nas mais diversas áreas da ciência. Entre elas podemos destacar as mais frequentemente encontradas.

1.1.1.Síntese de Estruturas Mecânicas

O método da otimização topológica vêm sendo extensivamente empregado na síntese de estruturas mecânicas que tenham máxima rigidez respeitando uma restrição de volume máximo e na otimização de estruturas já existentes para que atendam às características mencionadas. A maximização da rigidez de estruturas mecânicas com restrição de volume máximo em particular é o assunto deste trabalho e a figura 1.1.1.1 mostra a otimização do braço de uma suspensão dianteira de um caminhão.

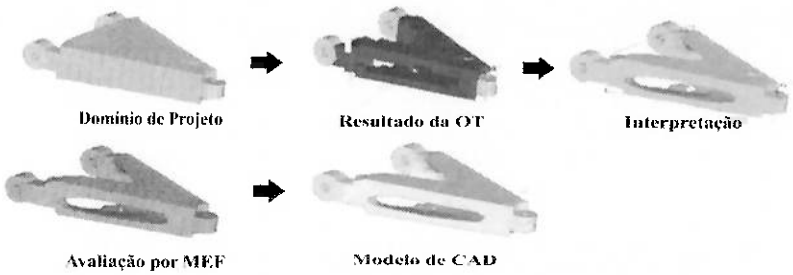


Figura 1.1.1.1.Otimização do braço de suspensão de um caminhão

1.1.2.Síntese de Microeletromecanismos (MicroEletroMechanisms - MEMS)

Mecanismos flexíveis são uma classe de mecanismos que não contém elos ou mancais (*i.e.*,juntas, dobradiças, solda, rolamentos), ou seja, são fabricados a partir de um único pedaço de material, sendo que a mobilidade desses mecanismos resulta das propriedades elásticas do próprio material. Um tipo importante de mecanismos flexíveis são os chamados MicroEletroMecanismos, os MEMS.Os MEMS são peças mecânicas de tamanho reduzido acoplados a dispositivos eletrônicos, que ao aplicarem uma tensão no material fazem com que este se deforme (essa deformação pode ser causada por efeito *Joule* ou por efeito piezoelétrico). A forma com que a tensão é aplicada e da topologia da peça, ambas determinadas pelo projetista, determinam qual será o movimento resultante

da peça. Devido ao tamanho reduzido desses mecanismos, eles devem ser feitos a partir de um único pedaço de material, sendo que a otimização topológica é uma ferramenta importante na determinação da topologia da peça a ser fabricada. A figura 1.1.2.1 mostra um exemplo de aplicação do método de otimização topológica ao projeto de MEMS.O objetivo nesse caso é obter um mecanismo que mediante a aplicação de tensões elétricas em dois pontos diferentes se deforma resultando em movimentos horizontais e verticais do cabeçote (um atuador com dois graus de liberdade). Como uma aplicação de mecanismos desse tipo podemos citar o desenvolvimento de dispositivos de armazenamento de dados. Já é possível armazenar dados em áreas de material da mesma ordem de grandeza de átomos ou moléculas. Consequentemente, o cabeçote de leitura de um dispositivo desse tipo deve ser pequeno o suficiente para poder varrer essas áreas minúsculas, tornando necessária a utilização de um MEM como o descrito.

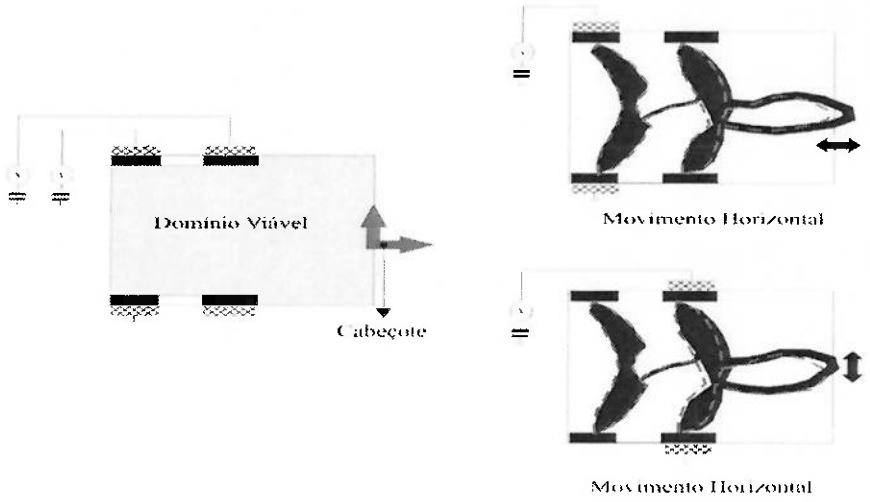


Figura 1.1.1.1.Síntese de um MEM

1.1.3.Síntese de materiais com propriedades extremas

Uma aplicação muito interessante da otimização topológica consiste na síntese de materiais que apresentem um comportamento deseado pelo projetista, seja ele mecânico, térmico ou elétrico. O método da otimização topológica altera a microestrutura do material de modo que ele tenha as propriedades desejadas. O método da otimização

topológica permite que se obtenham microestruturas que conferem ao material propriedades até não-intuitivas, como um coeficiente de *Poisson* negativo, ou seja, materiais que expandem na direção transversal quando tracionados na direção longitudinal e vice-versa. Outros exemplos são a síntese de materiais com condutividades térmica e elétrica desejadas.

2.OBJETIVOS

O objetivo deste trabalho é o desenvolvimento de uma ferramenta de síntese de estruturas mecânicas tridimensionais com o menor peso e maior rigidez possíveis que possa ser disponibilizada no meio acadêmico, possivelemente via Internet. Já existem programas disponíveis na rede mundial de computadores que sintetizam estruturas mecânicas ótimas, porém em domínios bidimensionais (www.topopt.dtu.dk). Dessa forma, este trabalho consiste no desenvolvimento de uma ferramenta de CAO (Computer Aided Optimizer),que integra as ferramentas de CAD (Conputer Aisded Design) e CAE (Computer Aided Engineering), sendo que o CAM (Computer Aided Manufacturing) seria utilizado posteriormente, para a fabricação da estrutura obtida. A figura **2.1** mostra essa relação.

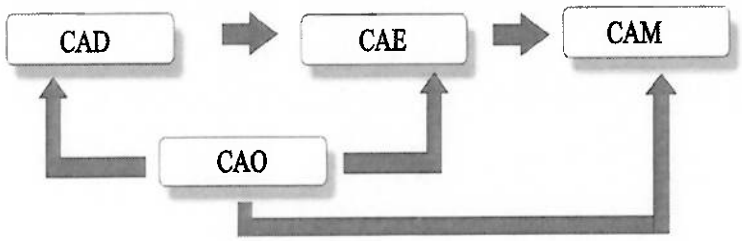


Figura 2.1. CAO (Computer Aided Optimizer)

A sequência básica de etapas que resulta na estrutura ótima está ilustrada na figura 2.2 para um domínio bidimensional. A primeira etapa consiste na determinação do domínio viável, dentro do qual a estrutura ótima deverá ser sintetizada e na determinação das cargas atuantes na mesma. A segunda etapa consiste na discrteização do domínio utilizando elementos finitos. A terceira etapa consiste na obtenção da topologia discretizada. A quarta etapa consiste na interpretação dos resutados obtidos para a obtenção de um modelo possível de ser fabricado.A quinta etapa consiste na verificação do comportamento da estrutura quando sujeita aos carregamentos de projeto e a última etapa consiste na fabricação da estrutura.

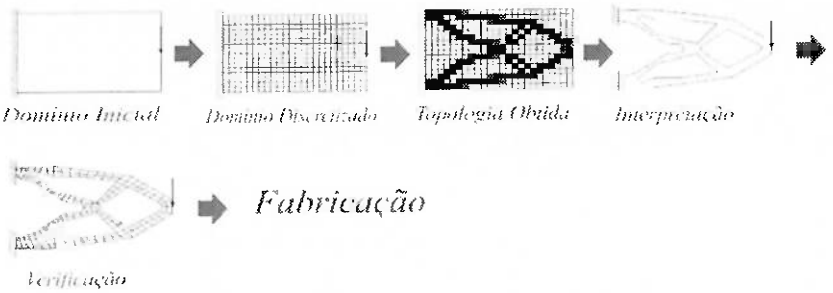


Figura 2.2.Etapas na obtenção de uma estrutura ótima

Neste trabalho serão implementadas a etapa 3 deste processo (obtenção da topologia ótima da estrutura na forma discreta), sendo que as informações a respeito da malha de elementos finitos e dos carregamentos atuantes na estrutura deverão ser fornecidos pelo usuário a partir de um arquivo de entrada gerado no software comercial ANSYS.

3.FUNDAMENTOS TEÓRICOS

3.1.Formulação do problema de Otimização

3.1.1.Energia Mútua e Flexibilidade Média

Considere um corpo tridimensional constituído de um material elástico linear, com domínio Ω e fixo na região Γ_d, como mostrado na figura 3.1.1.1. Suponha agora que este corpo está sujeito a uma força distribuída t aplicada numa porção de Γ_t de sua superfície e a uma força de volume *f*, como seu peso próprio. Esses carregamentos geram ao corpo um campo de deslocamentos $\mathbf{u}=\{u_1, u_2, u_3\}$.

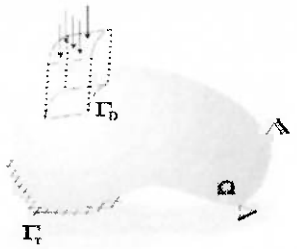


Figura 3.1.1.1.Corpo submetido a forças de superfície e forças de volume

Pela teoria da mecânica do contínuo sabe-se que a forma linear da flexibilidade média (ou "mean compliance") para esse corpo pode ser escrita da seguinte maneira :

$$l(\mathbf{u}) = \int_{\Omega} f\mathbf{u}d\Omega + \int_{\Gamma_T} t\mathbf{u}ds \quad \text{para } \mathbf{u} \in \mathbf{V} \tag{3.1.1.1}$$

onde **V** é o espaço linear admissível, tal que $\mathbf{V}=\{v=v_i\mathbf{e}_i\}$; $\mathbf{t}=\mathbf{T}\cdot\mathbf{n}$, sendo **T** o tensor de tensões (definição do estado de tensões num dado ponto) e **n** um vetor normal à superfície Γ_t . A quantidade l(**u**) é a flexibilidade média do corpo e pode ser interpretada como sendo a energia armazenada pelo corpo devido à deformação causada no mesmo pela aplicação da força de volume *f* em Ω e da força distribuída de superfície **t** em Γ_t. Na realidade, a flexibilidade média de um corpo elástico é igual ao dobro de sua energia potencial total. De fato, sabemos que o funcional Energia Potencial Total ($\prod$) é dado pela seguinte expressão :

$$\prod = \frac{1}{2}\int_{\Omega} \varepsilon(\mathbf{u})^T E\varepsilon(\mathbf{v})d\Omega - \int_{\Gamma_T} \mathbf{t}\cdot\mathbf{v}\, dS - \int_{\Omega} f\cdot\mathbf{v}\, d\Omega \quad \text{para } \mathbf{v} \in \mathbf{V} \tag{3.1.1.2}$$

Nessa expressão, o termo $\frac{1}{2}\int_{\Omega}\varepsilon(\mathbf{u})^{T}E\varepsilon(\mathbf{v})d\Omega$ é o trabalho virtual interno do corpo elástico no equilíbrio **u** devido a um campo de deslocamentos virtuais **v** e o termo $\int_{\Gamma_T}\mathbf{t}\cdot\mathbf{v}\,dS+\int_{\Omega}f\cdot\mathbf{v}\,d\Omega$ é o trabalho das forças externas (forças de superfície e de corpo) devido à **v**. **C** é a matriz de elasticidade tridimensional e **ε** o vetor de deformações linearizadas, dados pelas experssões abaixo:

$$E=\begin{pmatrix} 1\text{-}\nu & \nu & \nu & 0 & 0 & 0 \\ & 1\text{-}\nu & \nu & 0 & 0 & 0 \\ & & 1\text{-}\nu & 0 & 0 & 0 \\ & SIM. & & (1\text{-}2\nu)/2 & 0 & 0 \\ & & & & (1\text{-}2\nu)/2 & 0 \\ & & & & & (1\text{-}2\nu)/2 \end{pmatrix} \quad \varepsilon=\begin{Bmatrix}\varepsilon_x\\ \varepsilon_y\\ \varepsilon_z\\ \gamma_{yz}\\ \gamma_{zx}\\ \gamma_{xy}\end{Bmatrix} \tag{3.1.1.3}$$

Utilizando a notação tensorial, a expresão acima pode ser reescrita da seguinte maneira:

$$\Pi=\frac{1}{2}\int_{\Omega}E_{ijkl}\varepsilon_{kl}(\mathbf{u})\varepsilon_{ij}(\mathbf{v})d\Omega-\int_{\Gamma_T}\mathbf{t}\cdot\mathbf{v}\,dS-\int_{\Omega}f\cdot\mathbf{v}\,d\Omega \quad \text{para } \mathbf{v}\in V \tag{3.1.1.4}$$

Nessa expressão, E_{ijkl} é o tensor de elasticidade do material e $\varepsilon_{ij}(\mathbf{u})=\frac{1}{2}\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right)$ é a deformação linearizada devido ao campo de deslocamentos **u**.

Lembrando que no equilíbrio estático a condição de mínima energia potencial total deve ser satisfeita, devemos ter no equilíbrio estático a condição $\delta\prod=0$, que é obtida minimizando-se o funcional dado pela expressão **3.1.1.4**.Obtemos dessa forma a expressão **3.1.1.5**:

$$\int_{\Omega}\left[E_{ijkl}\varepsilon_{kl}(\mathbf{u})\varepsilon_{ij}(\delta\mathbf{v})-f\delta\mathbf{v}\right]d\Omega-\int_{\Gamma_T}\mathbf{t}\delta\,\mathbf{v}dS=0\Rightarrow\int_{\Omega}E_{ijkl}\varepsilon_{kl}(\mathbf{u})\varepsilon_{ij}(\delta\mathbf{v})d\Omega=\int_{\Gamma_T}\mathbf{t}\delta\mathbf{v}dS+\int_{\Omega}\mathbf{f}\delta\mathbf{v}\Omega \quad \text{para } \mathbf{v}\in V$$

onde $\delta\mathbf{v}$ é arbitrário entre as funções admissíveis.A expressão acima pode ser escrita numa forma reduzida, utilizando os operadores $a(\boldsymbol{u},\boldsymbol{v})$ e $l(u)$:

$$a(\mathbf{u},\delta\mathbf{v})=l(\mathbf{v}) \quad \text{para } \mathbf{v}\in V \text{ onde}$$
$$a(\mathbf{u},\delta\mathbf{v})=\int_{\Omega}E_{ijkl}\varepsilon_{kl}(\mathbf{v})\varepsilon_{ij}(\delta\mathbf{v})d\Omega \quad \text{e} \quad l(\mathbf{v})=\int_{\Gamma_T}\mathbf{t}\cdot\mathbf{v}\,dS+\int_{\Omega}f\cdot\mathbf{v}\,d\Omega \tag{3.1.1.6}$$

O operador $a(\mathbf{u},\mathbf{v})$ é denominado energia mútua e o operador $l(v)$ é denominado flexibilidade média. Como na expressão **3.1.1.5** , o campo de deslocamentos $\delta\mathbf{v}$ é arbitrário entre as funções admissíveis, podemos escrever, desde que **u** seja solução para o equilíbrio estático:

$$\int_{\Omega} E_{ijkl}\varepsilon_{kl}(\mathbf{u})\varepsilon_{ij}(\mathbf{u})d\Omega = \int_{\Gamma_T} \mathbf{t}\cdot\mathbf{u}\, dS + \int_{\Omega} f\cdot\mathbf{u}\, d\Omega \qquad (3.1.1.7)$$

Desta maneira, substituindo a equação acima na expressão **3.1.1.2** , concluímos que

$$\textit{flexibilidade média} = \int_{\Gamma_T} \mathbf{t}\cdot\mathbf{u}\, dS + \int_{\Omega} f\cdot\mathbf{u}\, d\Omega \;\; = -2\Pi_{equil.} \qquad (3.1.1.8)$$

3.1.2. Formulação da Função Objetivo para um Caso de Carregamento

Neste item será feita a formulação matemática do problema de maximização da rigidez (minimização da flexibilidade) de uma estrutura mecânica sujeita a uma restrição de volume máximo.Considere um corpo deformável ocupando um domínio Ω^{mat} no espaço, parte de um domínio maior Ω em $\mathbf{R^2}$ ou $\mathbf{R^3}$. O problema de otimização topológica pode então ser definido como o problema da determinação do tensor de elasticidade $E_{ijkl}(x)$, variável ao longo do domínio Ω e pertencente a um conjunto de tensores admissíveis, que minimize a função *flexibilidade média* no equilíbrio **u** dada pela expressão **3.1.1.6** . O tensor deve ser tal que a condição de equilíbrio estático no domínio seja satisfeita.Dessa forma, a expressão **3.1.2.1** define o problema de otimização:

$$\begin{array}{l} \min\limits_{u\in U,E} \quad l(\mathbf{u}) \\ \textit{Sujeito a} \quad a_E(\mathbf{u},\mathbf{v}) = l(\mathbf{v}) \textit{ para todo } v \in U \textit{ e } E \in E_{ad} \end{array} \qquad (3.1.2.1)$$

Nessa expressão, U é o espaço admissível para os campos de deslocamento no domínio $\boldsymbol{\Omega}$ e a restrição de equilíbrio estático no domínio é dada pela primeira das expressões **3.1.1.5**. O conjunto de tensores admissíveis E_{ad} pode variar, dependendo do modelo de material empregado. O modelo utilizado neste trabalho será descrito no próximo item.Note que a restrição de volume máximo não se encontra na expressão **3.1.2.1**. Isso acontece porque ela é considerada na formulação do modelo de material a

ser utilizado no domínio **Ω**, sendo satisfeita pelo conjunto de tensores admissíveis E_{ad} .É importante ressaltar que a restrição de equilíbrio é satisfeita na resolução do problema através do método de elementos finitos (Item **4.**).

3.1.2.1.Outras Formulações para a Função Objetivo

A função objetivo do problema de minimização da flexibilidade média de um corpo elástico sujeito a uma restrição de volume máximo pode ser escrita de outras formas. Considerando a expressão **3.1.1.2** e o princípio da mínima energia potencial total ,podemos expressar o problema de otimização da seguinte forma:

$$\min_{\mathbf{u}\in U, E\in E_{ad}} \left\{ \frac{1}{2} a_E(\mathbf{u},\mathbf{u}) - l(\mathbf{u}) \right\}$$

Considerando agora a expressão **3.1.1.8** , podemos concluir que o termo entre chaves é igual à metade da flexibilidade média do corpo elástico no equilíbrio **u** (com sinal negativo). Como o valor desse termo é negativo, devemos portanto maximizá-lo a fim de minimizar a flexibilidade média do corpo elástico. A expressão acima toma portanto a forma

$$\max_{E\in E_{ad}} \min_{\mathbf{u}\in U} \left\{ \frac{1}{2} a_E(\mathbf{u},\mathbf{u}) - l(\mathbf{u}) \right\} \qquad (3.1.2.1.1)$$

O problema de otimização também pode ser formulado em termos de tensão, levando em consideração o princípio da energia complementar, da seguinte maaneira:

$$\min_{E\in E_{ad}} \min_{\sigma\in \mathbf{S}} \left\{ \frac{1}{2} \int_{\Omega} C_{ijkl}\sigma_{ij}\sigma_{kl} \right\} \qquad (3.1.2.1.2)$$

onde $C_{ijkl} = (E^{-1})_{ijkl}$ é o tensor de rigidez do material e $S = \{\sigma \mid div\sigma + f = 0 \text{ em } \Omega,\ \sigma.n = t \text{ em } \Gamma_T\}$ é o conjunto de campos de tensão admissíveis.

3.1.3.Modelo de Material Utilizado

A parametrização do tensor de elasticidade (que define o conjunto de tensores admissíveis E_{ad} , ver item **3.1.2**) no domínio de projeto **Ω** deve ser tal que permita no

domínio apenas a presença ou ausência de material em cada ponto do domínio. Um possível conjunto E_{ad} seria, portanto, o conjunto composto de dois tensores: um nulo (um tensor com todas as suas propriedades mecânicas nulas representa ausência de material) e um determinado pelo tensor de elasticidade de um material base determinado a *priori*. Podemos definir então um conjunto de tensores em função de uma variável x, que representaria uma espécie de "densidade" do material e que pode assumir os valores 0 e 1 apenas. Considerando também a restricão de volume máximo imposta no problema de otimização, podemos definir o conjunto de tensores admissíveis E_{ad} como o conjunto

$$E_{ijkl}(x) = 1_{\Omega^{mat}} E^0_{ijkl}, 1_{\Omega^{mat}} = \begin{cases} 1 \; se \; x \in \Omega^{mat} \\ 0 \; se \; x \in \Omega / \Omega^{mat} \end{cases}$$
$$\int_\Omega 1_{\Omega^{mat}} \, d\Omega = Vol\left(\Omega^{mat}\right) \leq V \qquad (3.1.3.1)$$

Na expressão **3.1.3.1** $\mathbf{\Omega^{mat}}$ representa os pontos do domínio em que há material e $\mathbf{\Omega/\Omega^{mat}}$ os pontos com ausência de material. O tensor E^0_{ijkl} representa o material base. A parametrização do tensor de elasticidade na forma acima torna o problema de otimização inerentemente discreto, na medida em que em cada ponto só é possível a escolha de dois tensores de elasticidade, um que representa a ausência de material e outro que representando a presença de material. Como a utilização de algoritmos baseada na teoria de programação inteira para a resolução de problemas discretos de otimização gera problemas como a existência de vários mínmos locais, neste trabalho será utilizado o modelo de material SIMP (Simple Isotropic Material with Penalization), que contorna esse problema parametrizando o tensor de elasticidade $E_{ijkl}(x)$ da seguinte maneira:

$$E_{ijkl}(x) = \rho(x)^p E^0_{ijkl}, \; p \geq 1,$$
$$\int_\Omega \rho(x) d\Omega \leq V, \quad 0 \leq \rho(x) \leq 1, \quad x \in \Omega \qquad (3.1.3.2)$$

A utilização dessa parametrização do tensor de elasticidade por sua vez, permite a utilização de algoritmos para a solução de problemas de otimização contínuos. A parametrização na forma **3.1.3.2** consiste na interpolação do tensor entre 0 e o tensor de

elasticidade E°_{ijkl} do material base. O parâmetro p em **3.1.3.2** é um coeficiente penalizador de densidades intermediárias, que faz com que o problema se aproxime do problema discreto, para que na solução final o domínio não contenha muitos pontos com um material cujas propriedades sejam intermediárias entre 0 e o material base, ou seja, propriedades que seriam fisicamente incoerentes com as do material base. O valor do parâmetro p deve, no entanto, ser escolhido com cautela, já que quanto menor o seu valor, mais pontos com densidades intermediárias estarão presentes na solução final e um valor muito alto de p faz com que o problema volte à sua forma discreta.

3.1.3.1.Limites Numéricos para o fator de penalização

O expoente p na expressão **3.1.3.2** utilizado para penalizar as densidades intermediárias obtidas no processo de otimização deve ser selecionado de tal forma que não seja tão pequeno a ponto do resultado da otimização conter demasiado escalas de cinza (pontos com propriedades físicas intermediárias às do material base) nem tão grande a ponto de gerar instabilidades numéricas na resolução do problema, devido a semelhança com um problema de otimização discreto. Por outro lado, existe um limite inferior para o expoente p que assegura que as propriedades determinadas pelo tensor $\rho(x)^{p} E^{0}_{ijkl}$ sejam fisicamente viáveis se considerarmos a formação de microestruturas constituídas pelo material base. Ele é deduzido a partir dos limites de *Hashin-Shtrikman* e *Hashin-Shtrikman-Walpole*, que os valores máximo e mínimo que podem ser atingidos pelas propriedades efetivas de um material com ima microestrutura constituída de dois tipos diferentes de materiais (no presente problema os dois materiais seriam o materal base e o ar, com propriedades nulas). Dessa forma, pode-se mostrar que os limites inferiores para o expoente p são dados pela expressão **3.1.3.1.1**, onde ν_0 é o coeficiente de Poisson do material base.

$$\begin{aligned} p &> \max\left\{\frac{2}{1-\nu_0}, \frac{4}{1+\nu_0}\right\} \quad \textit{para problemas bidimensionais} \\ p &\geq \max\left\{15\frac{1-\nu_0}{7-5\nu_0}, \frac{3}{2}\frac{1-\nu_0}{1-2\nu_0}\right\} \quad \textit{para problemas tridimensionais} \end{aligned} \tag{3.1.3.1.1}$$

É interessante notar que para um coeficiente de Poisson de 1/3, a expreesão acima resulta em um limite inferior de 3 para o expoente p. Os valores mencionados são exatamente os valores utilizados para geração dos resultados deste trabalho (veja item 5).

3.2.Critério de Optimalidade

O problema de minimização da flexibilidade média com restrição de volume máximo na forma **3.1.2.1** (para um caso de carregamento) pode ser resolvido linearizando-se a função objetivo em relação às variáveis de projeto (densidades dos elementos finitos utilizados na discretização do domínio de projeto) e utilizando algoritmos de programação linear como o SIMPLEX.Neste trabalho será utilizado um algoritmo heurístico baseado nos critérios de optimalidade da função objetivo. A expresão **3.2.1** traz a formulação do problema de otimização para um caso de carregamento no espaço contínuo:

$$\begin{aligned} &\min_{u \in U, E} \quad l(\mathbf{u}) \\ &\text{Sujeito a} \quad a_E(\mathbf{u},\mathbf{v}) = l(\mathbf{v}) \text{ para todo } v \in U \\ &\qquad E_{ijkl}(x) = \rho(x)^p E^0_{ijkl}, \\ &\qquad \int_\Omega \rho(x) d\Omega \le V; 0 \le \rho \le 1 \end{aligned} \tag{3.2.1}$$

Associando ao problema os multiplicadores de Lagrange Λ, λ^+ e λ^-, podemos escrever o Lagrangeano Λ do problema na forma

$$L = l(\mathbf{u}) - \{a_E(\mathbf{u},\ddot{\mathbf{u}}) - l(\ddot{\mathbf{u}})\} + \Lambda\left(\int_\Omega \rho(x) d\Omega - V\right) + \int_\Omega \lambda^+(x)(\rho(x)-1) d\Omega + \int_\Omega \lambda^-(x)(0-\rho(x)) d\Omega \tag{3.2.2}$$

Na expressão **3.2.2**, **u** é o campo de deslocamentos do corpo no equilíbrio, **ü** é o multiplicador de Lagrange da restrição de equilíbrio, Λ é o multiplicador de Lagrange relativo à restrição de volume máximo, sendo essa uma restrição integral e λ^+ e λ^- são, respectivamente, os multiplicadores de Lagrange relativos às restrições impostas pelos limites superior e inferior do campo de densidades $\rho(x)$, sendo essas restrições ponto-a-ponto, ou seja, que devem ser satisfeitas em cada ponto do domínio. O campo de deslocamentos do corpo sujeito a restrição de equilíbrio é determinado pelo método dos elementos finitos (item **3.4**).A parcela $l(\mathbf{u})$ e $l(\ddot{\mathbf{u}})$ se cancelam e a parcela $a_E(\mathbf{u},\ddot{\mathbf{u}})$ se torna $a_E(\mathbf{u},\mathbf{u})$. Dessa forma, o problema da determinação do campo de deslocamentos e do

campo de densidades no corpo é resolvido em duas etapas diferentes, sendo a determinação do campo de deslocamentos feito pelo método dos elementos finitos e a determinação do campo de densidades pelo critério de optimalidade que será apresentado a seguir.

Minimizando Λ em relação à variável de projeto (campo de densidades ρ(x)) obtemos a expressão **3.2.3** e as condições de complementaridade **3.2.4**:

$$\frac{\partial L}{\partial \rho}=0 \Rightarrow -\frac{\partial a_E(\mathbf{u},\mathbf{u})}{\partial \rho}+\Lambda\int_\Omega d\Omega+\int_\Omega \lambda^+(x)d\Omega-\int_\Omega \lambda^-(x)d\Omega=0 \qquad (3.2.3)$$

$$\lambda^+\geq 0,\lambda^-\geq 0,\lambda\left(0-\rho(x)\right)=0,\lambda\left(\rho(x)-1\right)=0 \qquad (3.2.4)$$

Considerando por sua vez a expressão **3.1.1.6** para a energia mútua, podemos reescrever a expressão **3.2.3** da seguinte maneira:

$$\int_\Omega\left(-\frac{\partial\left(E_{ijkl}\varepsilon_{kl}(\mathbf{u})\varepsilon_{ij}(\mathbf{u})\right)}{\partial\rho(x)}+\Lambda+\lambda^+-\lambda^-\right)d\Omega=0 \qquad (3.2.5)$$

Utilizando agora as informações referentes ao modelo de material utilizado, mais precisamente a expressão **3.1.3.2**, podemos avaliar a derivada da flexiblidade média na expressão **3.2.5** , resultando em

$$-p\rho^{p-1}E^0_{ijkl}\varepsilon_{kl}(\mathbf{u})\varepsilon_{ij}(\mathbf{u})+\Lambda+\lambda^+-\lambda^-=0 \qquad (3.2.6)$$

Para pontos onde a densidade é intermediária ($0<\rho<1$) as restrições relativas aos limites superior e inferior do campo de densidades são inativas e consequentemente os respectivos multiplicadores de Lagrange são nulos ($\lambda^+=0$ e $\lambda^-=0$).Para densidades intermediárias a expressão **3.2.6** se torna

$$p\rho^{p-1}E^0_{ijkl}\varepsilon_{kl}(\mathbf{u})\varepsilon_{ij}(\mathbf{u})=\Lambda \qquad (3.2.7)$$

A expressão **3.2.7** nos diz que a energia específica de deformação em pontos de densidade intermediária é constante e igual a Λ Em pontos de densidade igual a 0 as

condições **3.2.4** nos dizem que a energia específica será menor que Λ e em pontos de densidade igual a 1 maior que Λ. Baseado nessas informações podemos desenvolver o seguinte algoritmo de atualização de densidades para cada ponto do domínio da estrutura:

$$\rho_{K+1} = \begin{cases} \max\{(1-\zeta)\rho_K, 0\} \text{ se } \rho_K B_K^{\eta} \leq \max\{(1-\zeta)\rho_K, 0\}, \\ \min\{(1+\zeta)\rho_K, 1\} \text{se } \min\{(1+\zeta)\rho_K, 1\} \leq \rho_K B_K^{\eta}, \text{ onde} \\ \rho_K B_K^{\eta} \text{ caso contrário} \end{cases} \tag{3.2.8}$$

$$B_K = \Lambda_K^{-1} p\rho(x)^{p-1} E_{ijkl}^0 \varepsilon_{kl}(\mathbf{u}_K)\varepsilon_{ij}(\mathbf{u}_K) \tag{3.2.9}$$

e $\mathbf{u}_K$ é o campo de deslocamentos da estrutura no equilíbrio e na iteração K. Λ_K é o valor do multiplicador de Lagrange Λ na iteração K. O algoritmo atualiza o valor da densidade em cada ponto do domínio independentemente dos outros, adicionando material em pontos com uma energia específica de deformação maior que Λ_K (isto é, com B_K>1) e retirando material nos pontos onde a energia específica de deformação é menor que Λ_K (isto é, com B_K<1) sendo que essas mudanças somente são realizadas se os novos valores de densidade não ultrapassarem os limites (0<ρ<1). Para pontos com densidades intermediárias um ótimo local é atingido quando a energia específica de deformação for igual à Λ_K (isto é, com B_K=1). A variável η em **3.2.8** é um parâmetro de sintonia e ζ um limite móvel. Ambos os parâmetros podem ser ajustados para aumentar a eficiência do algoritmo. Neste trabalho são utilizados respectivamente os valores 0.5 e 0.05. Em cada etapa do algoritmo é necessário o cálculo do multiplicador de Lagrange Λ_K. Como o volume da estrutura é uma função contínua e estritamente decrescente de Λ e como o volume é uma função monotônica nos intervalos onde um ou mais pontos possuem densidade internediária, o valor de Λ_K pode ser determinado através de um algoritmo de dicotomia ou método de Newton. Neste trabalho é utilizado um algoritmo de dicotomia.

3.3.Formulação para casos de múltiplos carregamentos

As formulações da função objetivo no item **3.2.1** e o critério de optimalidade no item **3.2** foram determinados considerando apenas um caso de carreGamento, ou seja,

que o corpo estará submetido a apenas um carregamento, podendo este ser composto de forças e/ou momentos. O problema de otimização pode também ser formulado considerando vários casos de carregamento, ou seja, que o corpo estará submetido a carregamentos diferentes em instantes de tempo diferentes. Obviamente a solução do problema para o caso de um carregamento não será igual a solução para outro caso de carregamento, sendo necessária uma solução de compromisso entre os diversos casos de carregamento. Isso é feito considerando cada caso de carregamento ponderado por um respectivo peso determinado a priori. A figura 3.3.1 ilustra um domínio inicial de projeto sujeito a diferentes casos de carregamento:

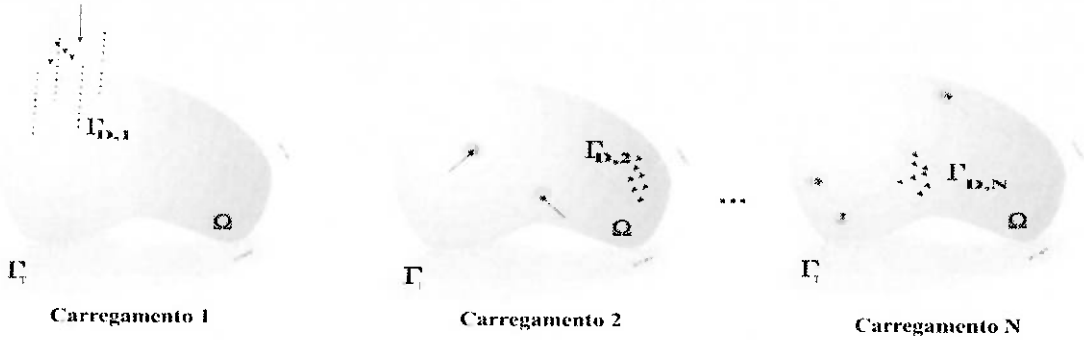


Fig. 3.3.1.Domínio de projeto sujeito a diferentes casos de carregamento

A flexibilidade média do corpo para *N* casos de carregamento pode então ser expressa pelo somatório das flexibilidades médias de cada caso de carregamento ponderados pelos respectivos pesos. A formulação do problema de otimização na forma **3.1.2.1** para *N* casos de carregamento se torna:

$$\begin{aligned} &\min_{\mathbf{u}_i \in U_i, E} \ \sum_{i=1}^{N} f_i l_i(\mathbf{u}_i) \\ &\text{Sujeito a } \ a_E(\mathbf{u}_i, \mathbf{v}_i) - l(\mathbf{v}_i) \text{ para todo } \mathbf{v}_i \in U_i \text{ e } E \in E_{ad} \end{aligned} \tag{3.3.1}$$

onde f_i é o peso relativo a cada caso de carregamento, u_i é o campo de deslocamentos no equilíbrio para cada caso de carregamento , v_i são campos de deslocamentos virtuais e U_i é o espaço linear admissível para cada caso de carregamento. Analogamente, a expressão **3.1.2.1.1** toma a forma

$$\max_{E\in E_{ad}} \min_{\hat{u}=\{u_1,\cdots,u_N\}} \left\{ \int_\Omega W(E,\hat{\mathbf{u}})d\Omega - l(\hat{\mathbf{u}}) \right\} \text{ onde}$$
$$W\left(E,\hat{\mathbf{u}}=\{\mathbf{u}_1,\cdots,\mathbf{u}_N\}\right)=\frac{1}{2}\sum_{i=1}^{M} f_i E_{pjkl}(x)\varepsilon_{pj}(\mathbf{u}_p)\varepsilon(\mathbf{u}_k) \quad \text{e} \tag{3.3.2}$$
$$l(\hat{\mathbf{u}}=\{\mathbf{u}_1,\cdots,\mathbf{u}_N\})=\sum_{i=1}^{M} f_i l_i(\mathbf{u}_i)$$

e a expressão **3.1.2.1.2** toma a forma

$$\min_{E\in E_{ad}} \min_{\substack{div\ \sigma_i+f_i=0\ em\ \Omega \\ \sigma_i\cdot n=t_i\ em\ \Gamma_T^i \\ i=1,\cdots,M}} \left\{ \frac{1}{2}\int_\Omega \sum_{i=1}^{N} f_i C_{pjkl}\sigma_{pj}^i\sigma_{kl}^i d\Omega \right\} \tag{3.3.3}$$

A expressão **3.2.9** para o critério de optimalidade por sua vez toma a seguinte forma para múltiplos casos de carregamento:

$$B_K = \Lambda_K^{-1} p\rho(x)^{p-1}\sum_{i=1}^{N} E_{pjkl}^0 \varepsilon_{pj}(\mathbf{u}_K^i)\varepsilon_{kl}(\mathbf{u}_K^i) \tag{3.3.4}$$

3.4.Método dos Elementos Finitos

A derivação das equações do método dos elementos finitos para elasticidade tridimensional pode ser feita através do príncípio da mínima energia potencial (PMPE) que nos diz que no equilíbrio o funcional de energia potencial total deve ter valor mínimo. O funcional de energia potencial total pode ser dado por:

$$\Pi = a(\mathbf{u},\mathbf{v}) = b(\mathbf{v}) \tag{3.4.1}$$

onde é a forma bilinear correspondente a energia de deformação , dada por

$$a(\mathbf{u},\mathbf{v}) = \frac{1}{2}\int_\Omega \varepsilon(\mathbf{u})E\varepsilon(\mathbf{v})d\Omega \tag{3.4.2}$$

e b(u) é a forma linear referente ao trabalho das forças externas, dada por

$$b(u) = \int_\Gamma t\cdot u d\Gamma + \int_\Omega p\cdot u d\Omega \tag{3.4.3}$$

Nas expressões acima, E é o tensor constitutivo de quarta ordem de elasticidade linear, ε é o tensor de segunda ordem das deformações, t é o vetor de forças externas, Γ é a superfície de aplicação das forças externas , p é o vetor de forças de corpoe u e v são

vetores de deslocamento. Considere a discretização do domínio em *ne* elementos finitos, a relação entre o vetor de deformações ε e o vetor de deslocamentos u dada pelo operador diferencial L e interpolando os deslocamentos u em cada elemento baseado nos valores nodais de deslocamento q^e , ou seja,

$$\Omega = \sum_{i=1}^{ne} \Omega_i^e$$

$$\mathbf{u}^e = \mathbf{N}^e \mathbf{q}^e$$

onde o índice e indica que estamos trabalhando no nível de um elemento e N^e é a matriz que contém as funções de interpolação do elemento. A relação entre o vetor de deslocamentos e o vetor de deformações pode ser dado por

$$L(\mathbf{u}) = \varepsilon \text{ onde } L = \mathbf{u} = \begin{Bmatrix} \partial/\partial x & 0 & 0 & 0 & \partial/\partial z & \partial/\partial y \\ 0 & \partial/\partial y & 0 & \partial/\partial z & 0 & \partial/\partial x \\ 0 & 0 & \partial/\partial z & \partial/\partial y & \partial/\partial x & 0 \end{Bmatrix}^T \begin{Bmatrix} u \\ v \\ w \end{Bmatrix} \qquad (3.4.4)$$

$$\varepsilon = \begin{Bmatrix} \varepsilon_x & \varepsilon_y & \varepsilon_z & \gamma_{yz} & \gamma_{zx} & \gamma_{xy} \end{Bmatrix}^T$$

O funcional de energia potencial total pode então ser dado por

$$\Pi = \frac{1}{2} \sum_{i=1}^{ne} \int_{\Omega^e} L(N^e) q^e E^e L(N^e) q^e d\Omega^e - \sum_{i=1}^{ne} \int_{\Gamma^e} t \cdot N^e q^e d\Gamma^e - \sum_{i=1}^{ne} \int_{\Omega^e} p \cdot N^e q^e d\Omega^e \qquad (3.4.5)$$

Extremizando então esse funcional em relação ao campo de deslocamenros e igualando a zero (mínima energia potencial total), obtemos:

$$\frac{d\Pi}{dq^e} = \sum_{i=1}^{ne} \int_{\Omega^e} L(N^e) E^e L(N^e) d\Omega^e q^e - \sum_{i=1}^{ne} \int_{\Gamma^e} t \cdot N^e d\Gamma^e - \sum_{i=1}^{ne} \int_{\Omega^e} p \cdot N^e d\Omega^e = 0 \qquad (3.4.6)$$

Obtendo

$$\sum_{i=1}^{ne} \int_{\Omega^e} L(N^e) E^e L(N^e) d\Omega^e q^e = \sum_{i=1}^{ne} \int_{\Gamma^e} t \cdot N^e d\Gamma^e + \sum_{i=1}^{ne} \int_{\Omega^e} p \cdot N^e d\Omega^e \text{ ou} \qquad (3.4.7)$$

$$\mathbf{Kq} = \mathbf{f} \qquad (3.4.8)$$

O sistema **3.4.8** resolvido fornece os valores dos deslocamentos nodais, sendo que o algoritmo utilizado deve ser eficiente devido ao tamanho das matrizes envolvidas. Neste trabalho está sendo utilizado um algoritmo baseado no método dos gradientes conjugados para solução de problemas de otimização. Além disso, as matrizes envolvidas são esparsas, ou seja, muitos de seus elementos são nulos; para que a alocação de memória no computador utilizado para a resolução do sistema linear seja mais eficiente, optou-se pelo armazenamento na forma esparsa, onde só os elementos não-nulos são armazenados. Neste trabalho será utilizado o elemento isoparamétrico de oito nós ilustrado na figura **3.4.1**, que consiste de um paralelepípedo com um nó em cada aresta, sendo que cada nó possui três graus de liberdade (translação nas três direções). O elemento possui, consequentemente, 8x3=24 graus de liberdade.

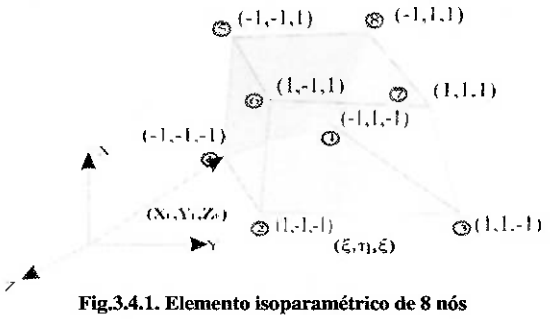


Fig.3.4.1. Elemento isoparamétrico de 8 nós

3.5.Formulação Discreta

Nessa seção as equações mais importantes para o problema de otimização topológica serão apresentadas em sua forma discreta, baseando-se na discretização do domínio de projeto em elementos finitos, cuja formulação foi feita no item anterior. A expressão **3.1.2.1** que equaciona o problema de otimização pode ser escrito na sua forma discreta como

$$\begin{aligned} &\min_{u,E_e,e=1,\cdots,N} \mathbf{f}^T\mathbf{u} \\ &tal\ que\ \mathbf{K}(E_e)\mathbf{u}=\mathbf{f}, \\ &E_e \in E_{ad}, e=1,\cdots,N \end{aligned} \tag{3.5.1}$$

onde N é o número de elementos na malha, E_e é o tensor de elasticidade no elemento e, f é o vetor de forças aplicado nos nós da malha, $\mathbf{u}$ é o vetor de deslocamentos nos nós da

malha e $\mathbf{K}(E_e)$ é a matriz de rigidez global, função do tensor de elasticidade de cada elemento da malha. Essa matriz pode ser expressa como função das matrizes de rigidez locais de cada elemento da malha da seguinte maneira:

$$\mathbf{K} = \sum_{e=1}^{N} \mathbf{K}_e(E_e) \tag{3.5.2}$$

onde $\mathbf{K}_e$ é a matriz de rigidez de cada elemento da malha. A expressão **3.1.3.2** para o modelo de material considera, na forma contínua, a existência de uma densidade variável ρ para a densidade em cada ponto do domínio. No caso discreto, cada elemento da malha é associado ao valor de uma variável de densidade, sendo que no interior do elemento a densidade é constante. Dessa forma, o critério de optimalidade **3.2.8** é aplicado no caso discreto para as N variáveis que representam os valores de densidade em cada elemento finito (no caso contínuo o critério teria de ser aplicado nos infinitos pontos do domínio). Note que a mesma malha de elementos finitos é utilizada para a daterminação dos campos de deslocamento e densidade no domínio de projeto.

3.6.Filtro de Densidades e o "Tabuleiro de Xadrez"

Um problema muito comum encontrado nos métodos de resolução do problema de minimização da flexibilidade média de uma estrutura mecânica sujeita a uma restrição de volume máximo é o "tabuleiro de xadrez", que se caracteriza pela formação de áreas com o aspecto de um tabuleiro de damas, com elementos de densidades próximas de 1 e próximas de 0 intercaladas. Esse problema também pode ser observado em análises de escoamento de *Stokes* pelo método dos elementos finitos e é associado à utilização de determinados tipos de elementos de baixa ordem na malha, como o elemento isoparamétrico de 8 nós que confere à configuração em forma de tabuleiro de xadrez uma rigidez alta. Uma maneira de eliminar o tabuleiro de xadrez é utilizar elementos de ordem maior, com um maior número de nós. Tal abordagem no entanto tornaria a solução do problema de otimização muito cara computacionalmente, optando-se geralmente pela utilização outros artifícios como restrições de perímetro, controles de gradiente e diversos tipos de filtros de densidade (filtros de vizinhança fixa, filtros espaciais). Neste trabalho optou-se pela utilização de um filtro de sensibilidades com um raio de varredura fixo.

Nesse filtro, a derivada da flexibilidade média em relação à densidade de cada elemento é obtida por uma média ponderada das derivadas da flexibilidade média em relação a densidade dos elementos cujos centróides estejam a uma distância menor ou igual ao raio de varredura em relação ao centróide do elemento central. Dessa forma, é evitada a ocorrência de mudanças bruscas de densidade em elementos vizinhos.Tal filtro também garante a unicidade da solução no domínio, ou seja, que a topologia final obtida não seja dependente do grau de discretização da malha de elementos finitos. O filtro pode ser expresso portanto pela seguinte expressão:

$$\frac{\partial f}{\partial \rho_k} = \frac{\sum_{i=1}^{N} \hat{H}_i \rho_i \frac{\partial f}{\partial \rho_i}}{\rho_k \sum_{i=1}^{N} \hat{H}_i} \tag{3.6.1}$$

onde N é o número de elementos cujo centróide se encontra a uma distância menor ou igual ao raio de varredura $r_{\min}$ do centróide do elemento k e f é a flexibilidade média da estrutura. O operador $\hat{H}_i$ atribui a sensibilidade da flexibilidade média em relação a cada elemento um peso, dado por

$$\hat{H}_i = r_{min} - dist(k,i), \{i \in N \mid dist(k,i) < r_{min}\}, k = 1, \cdots, N \tag{3.6.2}$$

A figura **3.6.1** ilustra esquematicamente a ocorrência do fenômeno do tabuleiro de xadrez e os elementos utilizados na filtragem de sensibilidades com um filtro de raio R.

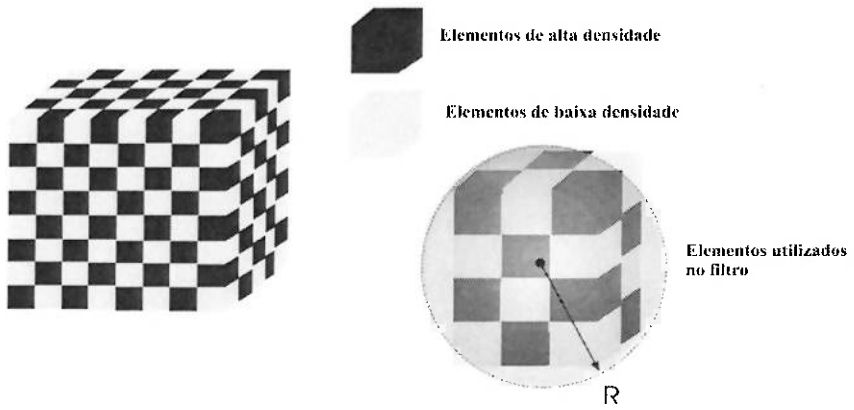


Fig. 3.6.1.Tabuleiro de xadrez e filtro de sensibilidades

Em todos os resultados obtidos neste trabalho foi utilizado um valor de r_{min} igual a uma vez e meia o tamanho da aresta de um elemento cúbico da malha de elementos finitos, ou seja, o filtro para um elemento engloba todos os elementos cujos centróides se encontrem dentro de uma esfera de raio igual a uma vez e meia a aresta de um elemento da malha.

4.IMPLEMENTAÇÃO

Neste item serão apresentados alguns aspectos relativos à implementação da rotina de otimização topológica para estruturas mecânicas tridimensionais. O fluxo de dados no programa pode ser resumido nas seguintes etapas:

1.Leitura do arquivo de entrada e montagem dos vetores e matrizes fixos: Nesta etapa o programa lê o arquivo de entrada, proveniente do programa comercial de elementos finitos ANSYS contendo todas as informações a respeito da malha de elementos finitos que deverá ser utilizada, como a matriz de conectividade, coordenadas nodais, forças e deslocamentos (condições de contorno). O programa também lê do arquivo as propriedades mecânicas do material base (módulo de elasticidade e coeficiente de Poisson), e o número de casos (para problemas multicaso, veja item **3.3**). O programa armazena todas essas informações em vetores e matrizes que serão utilizados ao longo do processamento. Nessa etapa também são montados vetores e matrizes de auxílio à rotina de MEF e montagem da matriz na forma esparsa e uma matriz que armazena informações a respeito dos elementos vizinhos para cada elemento da malha para o processo de filtragem de sensibilidades (veja item **3.6**).

2.Rotina de elementos finitos: Esta etapa consiste na determinação dos deslocamentos nodais da malha de elementos finitos através da resolução de um sistema de equações. Tal sistema de equações neste trabalho é resolvido utilizando um algoritmo de otimização baseado no método dos *Gradientes Conjugados*. O armazenamento da matriz de rigidez é feito na forma esparsa (somente são armazenados os elementos não-nulos da matriz), o que possibilita a discretização do domínio de projeto com malhas finas, da ordem de 100.000 elementos. A rotina de elementos finitos, mostrada no fluxograma **5.2** é composta das seguintes etapas:

1. Alocação de memória para a matriz de rigidez na forma esparsa. A memória é alocada na forma de um vetor de vetores de tamanho variável, sendo que o número de vetores é igual ao número de colunas da matriz de rigidez e o tamanho de cada vetor é igual ao número de elementos não-nulos na respectiva coluna na

matriz de rigidez global. A alocação de memória é feita com o auxílio de um vetor contendo o tamanho de cada coluna da matriz de rigidez. Esse vetor é construído na etapa 1 (Montagem de vetores e marizes fixos) e é passado à rotina de elementos finitos como argumento.A figura **4.1** ilustra os vetores citados e suas funções. O vetor COLSIZES contém o número de elementos não-nulos em cada coluna da matriz de rigidez global, sendo o seu tamanho igual ao número de colunas da matriz de rigidez global.Com o vetor COLSIZES e a matriz de conectividade da malha, é construído o vetor de vetores INDX, que é basicamente um mapeamento entre as matrizes de rigidez nas suas formas cheias e esparsa. O número de vetores nessa estrutura é igual ao número de colunas da matriz de rigidez e o tamanho de cada vetor é determinado pelo vetor COLSIZES.Cada um desses vetores contém o número do grau de liberdade correspondente ao elemento não-nulo em questão.

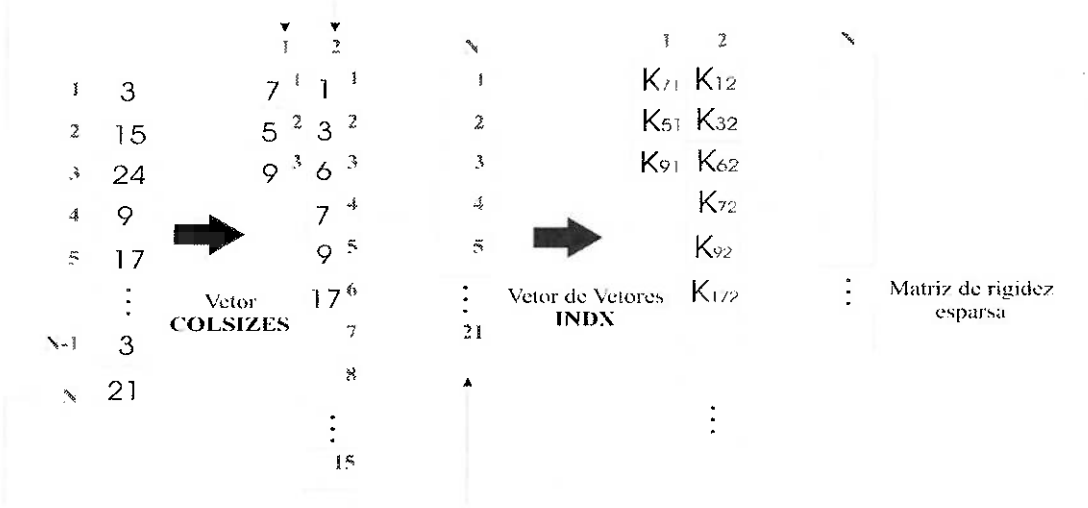


Figura 4.1. Estrutra de dados para geração da matriz esparsa

2. Determinação da matriz de rigidez local de um elemento da malha. A matriz de rigidez local de um elemento é determinada segundo a teoria de elementos finitos descrita no item **3.4**.

3. Inserção da matriz de rigidez local na matriz global na forma esparsa. A matriz local, calculada na etapa **2** é inserida na matriz global com o auxílio de uma matriz, montada na etapa **1**. Tal matriz é na verdade um vetor de vetores de tamanho variável (vetor de vetores INDX), de mesmo tamanho dos vetores da matriz de rigidez global e cada vetor indica quais elementos das matrizes de rigidez locais devem ser armazenados na respectiva coluna da matriz de rigidez global. Essa matriz é passada como argumento à rotina de elementos finitos. Dessa forma, a matriz de rigidez global é construída diretamente na sua forma esparsa utilizando um mapeamento dos elementos não-nulos da mesma, o que além de possibilitar o refinamento da malha utilizada permite a utilização do método dos gradientes conjugados para a solução do sistema de equações lineares resultante.

As etapas **2** e **3** são realizadas para cada elemento da malha (o que é feito através de um laço), o que permite obter a matriz de rigidez global diretamente na forma esparsa.
4.Solução do sistema de equações lineares para determinação dos deslocamentos nodais da malha. O sistema de equações $\boldsymbol{Ku=f}$, onde $\boldsymbol{K}$ é a matriz de rigidez global, $\boldsymbol{u}$ é o vetor de deslocamentos nodais a ser determinado e $\boldsymbol{f}$ é o vetor de forças nodais (construído na etapa **1**) é resolvido utilizando o método dos *Gradientes Conjugados*.

3.Cálculo da função objetivo: Nesta etapa é feito o cálculo da função objetivo (flexibilidade média), de acordo com a expressão **3.5.1** utilizando os valores dos deslocamentos nodais obtidos na etapa **2**. Os deslocamentos nodais para o cálculo da função objetivo são passados à rotina de cálculo da mesma na forma de um vetor, obtido na etapa anterior.

4.Cálculo das sensibilidades da função objetivo em relação à densidade de cada elemento: Nesta etapa é realizado o cálculo das sensibilidades da função objetivo em relação a cada variável de projeto (densidades dos elementos). Essas sensiblidades serão utilizadas na rotina que implementa o critério de optimalidade, de acordo com a expressão **3.2.8**, porém devem ser filtradas antes, o que é feito na próxima etapa.

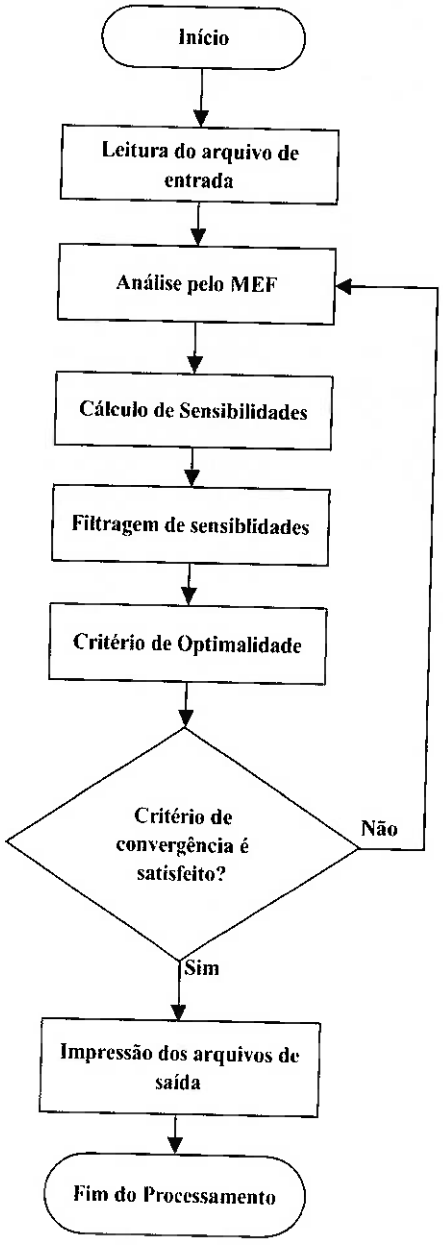
5.Filtragem das sensibilidades da função objetivo: Nesta etapa é realizada a filtragem das sensibilidades da função ojetivo em relação às variáveis de projeto de acordo com a expressão **3.6.1**.Para tal, a subrotina em questão utiliza os valores das sensibilidades originais, obtidas na etapa anterior e a matriz contendo os elementos vizinhos de cada elemento, obtido na etapa **1**.

6.Rotina de otimização topológica: Nesta etapa é realizada a determinação dos valores de densidade dos elementos da malha que minimizem a função objetivo (flexiblidade média), respeitando a restrição de volume imposta no problema.A subrotina em questão é baseada no critério de optimalidade descrito com maiores detalhes no item **3.2**.

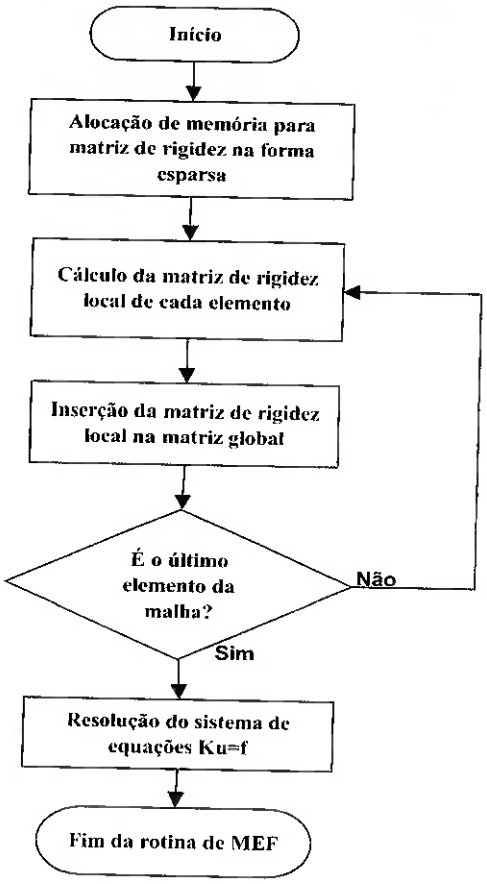
As etapas **2** até **6** fazem parte de um laço. O critério de convergência para saída desse laço é baseado no valor do módulo da diferença entre os valores da função objetivo (flexibilidade média) entre a última iteração e a penúltima iteração.

7.Impressão dos arquivos de saída: Nesta etapa o programa imprime um arquivo de saída que deve ser aberto no programa comercial de elementos finitos ANSYS para visualização e pós-processamento do modelo otimizado e um arquivo que deve ser aberto no programa de análise numérica MATLAB para visualização dos gráficos de convergência da função objetivo e do volume de material do modelo.

O fluxograma **5.1** mostra os passos **1** a **7**, ilustrando o fluxo de dados entre as partes principais do programa.



Fluxograma 5.1.Fluxo principal de dados no programa MEF



Fluxograma 5.2.Fluxo de dados na subrotina de

5.RESULTADOS

Para a obtenção de todos os resultados neste trabalho foram utilizados os valores de 100 e 1/3 respectivamente para o módulo de elasticidade (E_0) e coeficiente de *Poisson* (ν_0) do material base. Para os parâmetros do critério de optimalidade foram utilizados, respectivamente, os valores de 0,5 e 0,05 e para o raio de varredura do filtro de sensibilidades foi utilizado o valor de armin, onde a é o tamanho da aresta de um elemento da malha. Para cada exemplo são apresentadas três vistas da topologia obtida e os gráficos de convergência da função objetivo (flexibilidade média) e do volume da estrutura no domínio de projeto.

1. As figuras **5.1.B** e **5.1.C** ilustram a topologia obtida e os gráficos de convergência para o problema de uma viga 4x4x2 engastada sujeita a um carregamento na sua extremidade livre inferior direcionada para baixo. A figura **5.1.A** ilustra o problema. O domínio de projeto neste problema foi discretizado em aproximadamente 65.000 elementos e o tempo de processamento foi de aproximadamente 37 horas.

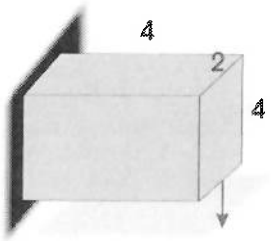


Fig. 5.1.A.Viga engastada sujeita a carregamento externo

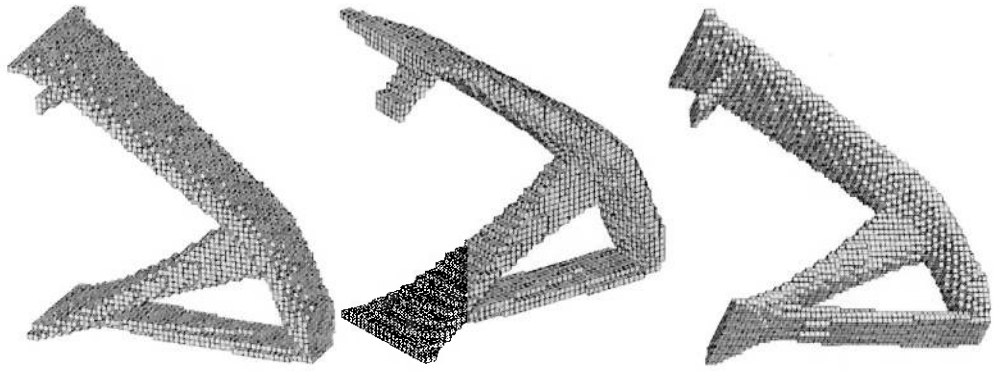


Fig. 5.1.B. Topologia obtida para o exemplo 1

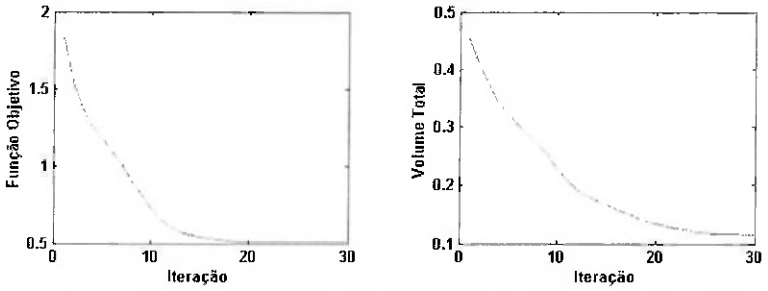


Fig. 5.1.C. Gráficos de convergência para o exemplo 2

2. As figuras **5.2.B** e **5.2.C** ilustram a topologia obtida e os gráficos de convergência para o problema de uma viga 6x2x2 engastada sujeita a dois casos distintos de carregamento (problema de múltiplos carregamentos). Um dos carregamentos se encontra na extremidade livre inferior da viga e é direcionada para baixo. O outro carregamento se encontra na extremidade livre superior da viga e é direcionada para cima. A figura **4.2.A** ilustra o problema: Neste caso o domínio de projeto foi discretizado em aproximadamente 73.000 elementos e o tempo de processamento foi de aproximadamente 48 horas.

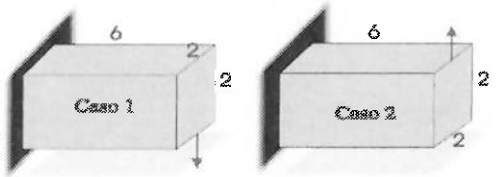


Fig. 5.2.A Viga engastada sujeita a 2 carregamentos diferentes

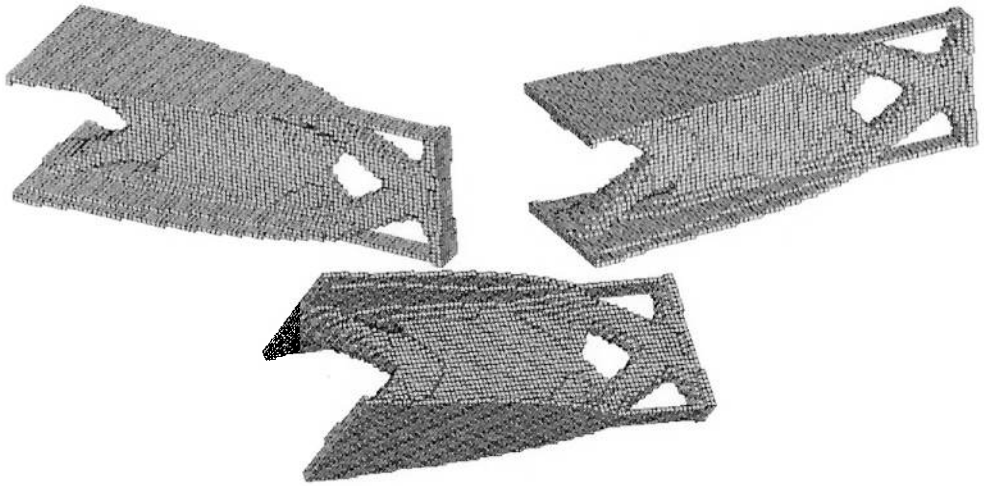


Fig. 5.2.C.Topologia obtida para oexemplo 2

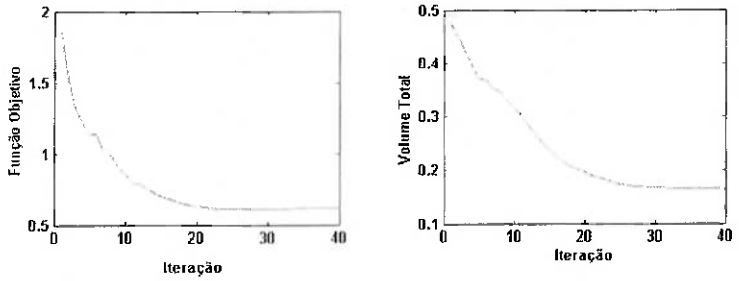


Fig. 5.2.C. Gráficos de convergência para o exemplo 2

3. As figuras **5.3.B** e **5.3.C** ilustram a topologia obtida e os gráficos de convergência para o "problema da ponte" , que consiste em uma viga 7x5x2 biapoiada sujeita a um carregamento direcionado para baixo na sua parte inferior. A figura **5.3.A** ilustra o problema: Neste caso o domínio de projeto foi discretizado em aproximadamente 70.000 elementos e o tempo de processamento foi de aproximadamente 42 horas.

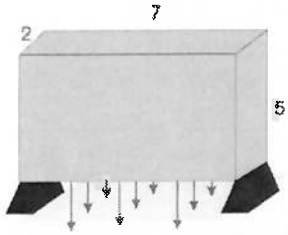


Fig. 5.3.A. Problema da "ponte"

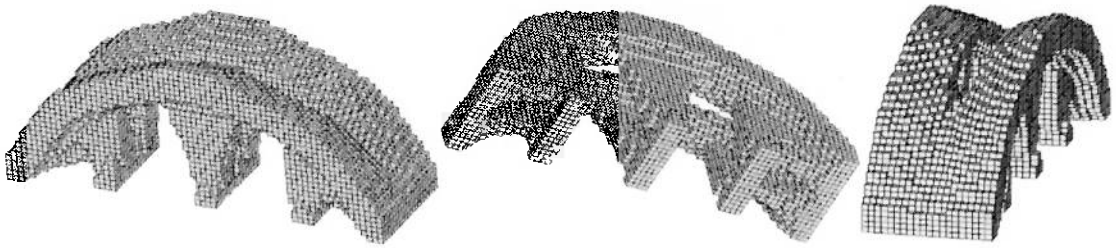


Fig. 5.3.B.Topologia obtida para o exemplo 3

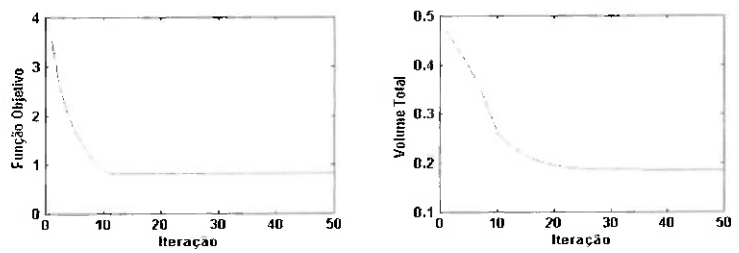


Fig. 5.3.C. Gráficos de convergência para o exemplo 3

4. As figuras **5.4.B** e **5.4.C** ilustram a topologia obtida e os gráficos de convergência para o problema de uma viga 6x2x2 biapoiada sujeita a um carregamento direcionado para baixo no centro e na sua parte superior.Para simplificar o problema e permitir a discretização do domínio em um número maior de elementos somente metade do problema foi resolvido. Para a obtenção da topologia final basta considerar a simetria do problema. A figura **5.4.A** ilustra o problema: Neste caso o domínio de projeto foi discretizado em aproximadamente 81.000 elementos e o tempo de processamento foi de aproximadamente 50 horas.

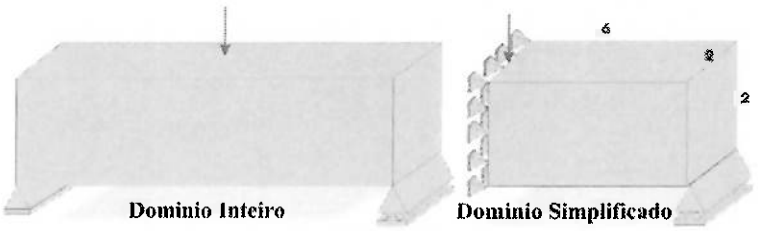


Fig. 5.4.A. Problema 4

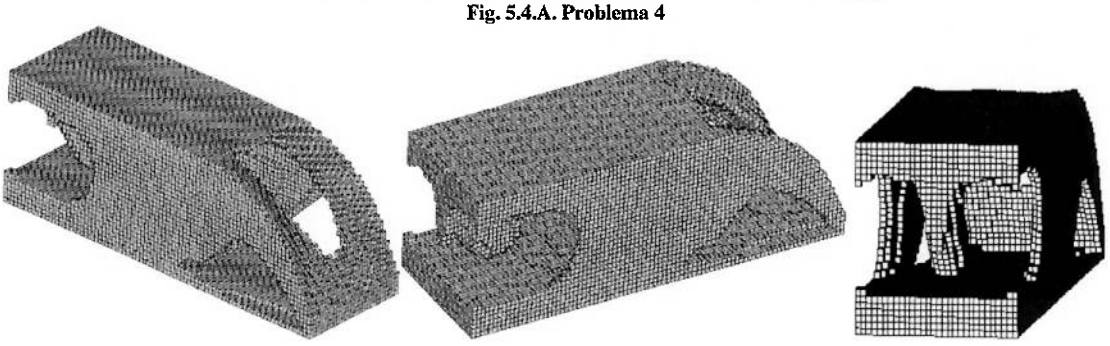


Fig.5.4.B. Topologia obtida para o exemplo 4

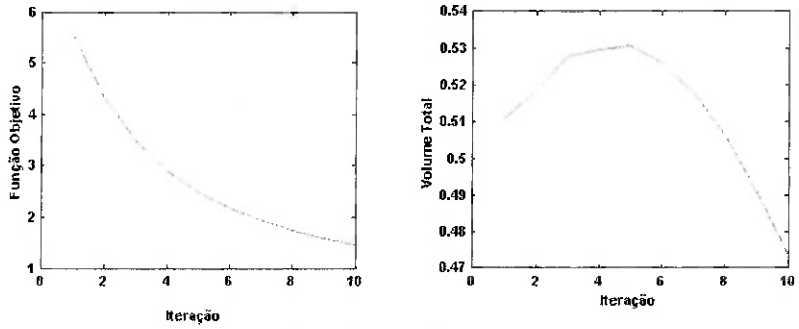


Fig. 5.4.C.Gráficos de Convergência para o exemplo 4

5.As figuras **5.5.B** e **5.5.C** ilustram a topologia obtida e os gráficos de convergência para o problema da otimização de um cordão de solda que une duas metades de uma viga engastada, de dimensões 5x5x2 sujeita a um carregamento unitário dirigido para baixo na sua extremidade livre superior. A espessura do cordão de solda é 1 e a sua altura 0.7. Neste exemplo, foi necessário assumir a existência de elementos passivos, ou seja, elementos que não participam do processo de otimização e portanto estão fora do domínio de projeto, apesar de serem utilizados no cálculo da função objetivo (flexibilidade média da viga soldada). Os elementos passivos neste caso são todos aqueles que fazem parte da viga, sendo fazem parte do domínio de projeto apenas os elementos que constituem o cordão de solda. Neste exemplo o domínio de projeto foi discretizado em aproximadamente elementos e o tempo de processamento foi de aproximadamente 3 horas. A figura **5.5.A** ilustra o problema.

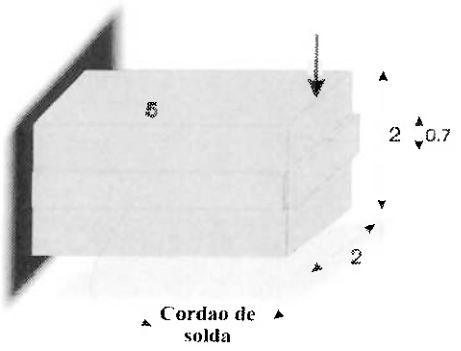


Fig.5.5.A. Problema da solda

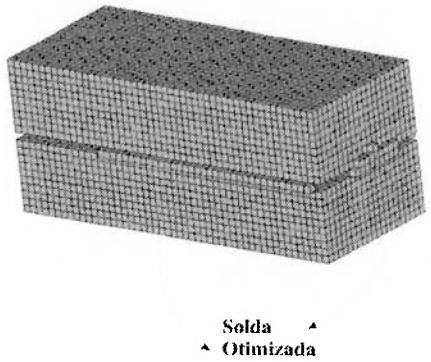


Fig. 5.5.B. Topologia obtida para o problema da solda

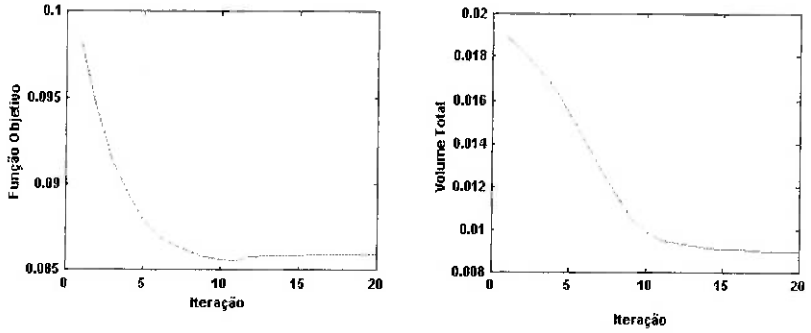


Fig. 5.4.C. Gráficos de convergência para o problema da solda

6.CONCLUSÕES

Foi desenvolvido neste trabalho um programa de otimização topológica para estruturas tridimensionais considerando casos com de um carregamento e casos de mais de um carregamento. A solução do problema de otimização em relação ao campo de densidades foi baseado em um algoritmo heurístico baseado em um critério de optimalidade para o problema de minimização da flexibilidade média de estruturas mecânicas. A determinação do campo de deslocamentos no domínio de projeto foi realizado através do método dos elementos finitos, utilizando elementos cúbicos isoparamétricos de 8 nós para a discretização e o método dos gradientes conjugados para a solução do sistema de equações lineares devido à grande dimensão de tal sistema. Um aspecto fundamental no trabalho foi a determinação da matriz de rigidez global de elementos finitos diretamante na foram esparsa, o que permitiu a obtenção da solução de problemas em domínios discretizados com malhas finas, da ordem de 100.000 elementos. Os resultados obtidos com a utilização do filtro de sensibilidades mostrou ser esse um método simples e eficaz para evitar o problema do tabuleiro de xadrez e garantir a unicidadeda solução, ou seja, garantir o mesmo resultado para graus diferentes de discretização do domínio. O critério de optimalidade motsrou ser um meio eficiente para a solução de problemas de otimização topológica. A utilização do critério de optimalidade em problemas onde a intuição física é limitada, como problemas onde restrições de natureza não-estrutural devem ser consideradas é limitada. A utilização de algoritmos baseados na teoria de programação matemática para a obtenção de resultados desse tipo é um tema a ser abordado em trabalhos futuros.

ANEXO

Abaixo se encontra uma listagem parcial do código fonte do programa desenvolvido, incluindo algumas passagens importantes.

```
**********INICIO DA LEITURA DE DADOS**********
/* Abertura do arquivo .txt *
arq_entrada=fopen(entrada,"r");
temp = (char *)malloc(TAM*sizeof(char ));
////////////////////* VALOR DE "NE" (NUMERO DE NOS) *////////////////////
compara = -1;
while ( compara > 0 || compara < 0) {
fgets(temp,TAM,arq_entrada);
        compara = strncmp( temp, "NUMOFF,NODE,", 12);
}
sscanf( temp, "%s %d", c1, &NE ); /*le a string do temp e atribui a N como inteiro*/
printf("O valor de NE e (numero de nos): %d\n", NE);
////////////////////* VALOR DE "M" (NUMERO DE ELEMENTOS) *////////////////////
fgets(temp, TAM, arq_entrada);
sscanf( temp, "%s %d", c2, &M ); /*le a string do temp e atribui a M como inteiro*/
PNP=ivector(1,M);
for(i=1;i<=M;i++) PNP[i]=0;
printf("O valor de M (numero de elementos ) e: %d\n", M);
////////////////////* MATRIZ "COORD" (COORDENADAS DOS NOS) *////////////////////
compara = -1;
while ( compara > 0 || compara < 0) {
        fgets(temp, TAM, arq_entrada);
        compara = strncmp( temp, "(3i8,6e16.9)", 12);
}
/* alocação de memoria para o vetor de coordenadas */
coord = dmatrix(1,3,1,NE);
if(dim==3){
        for(i=1;i<=NE;i++){
                fgets(temp, TAM, arq_entrada);
                a4=9999;        // como o ansys pode não escreve variavel nenhumas quando
                a5=9999;        // o valor desta é zero precisamos de uma condição para que
                a6=9999;        // a leitura identifique se o que foi lido é ou não uma coord valida
                sscanf( temp, "%d %d %d %lf %lf %lf", &a1, &a2, &a3, &a4, &a5, &a6 );
                if(a4==9999) a4 = 0;
                if(a5==9999) a5 = 0;
                if(a6==9999) a6 = 0;
                coord[1][i] = a4;
                coord[2][i] = a5;
                coord[3][i] = a6;
        }
}
**********Laço Principal e Chamada das Subrotinas**********
        printf("montando a matriz INDX e o vetor colsizes...\n");
        colsizes=ivector(1,ksize);
```

```
indx=(int **)malloc(ksize*sizeof(int *));
INDX(indx, colsizes,dim);  // criação de vetores e matrizes auxiliares para montagem da matriz de rigidez global
printf("montando o vetor diagindx...\n");
diagindx=ivector(1,ksize);
DiagINDX(diagindx, indx, colsizes); // criação vetor auxiliar para uso na subrotina de Gradientes Conjugados
int **femnts=FEMNTS(radius,dim);  // criação de matriz auxiliar para uso na filtragem de sensibilidades
obj=dvector(1,300);
vol=dvector(1,300);
double sum;
volfrac=0.5;
penal=3;
ub=dmatrix(1,ksize,1,NCASE);  // vetor de deslocamentos
x=dvector(1,M);
xold=dvector(1,M);
dcompl=dvector(1,M);
for(i=1;i<=M;i++){
        if(PNP[i]==0) x[i]=volfrac;
        else if(PNP[i]==1) x[i]=1;
}
int loop=0;
change=1;
while(change>0.01){  // laço principal do programa
        sum=0.0;
        loop++;
        for(i=1;i<=M;i++) xold[i]=x[i];
        MEF(indx,colsizes,ub,x,NCASE,dim);  // cálculo dos deslocamentos nodais
        compliance(ub, &compl);  // cálculo da função objetivo
        dcompliance(ub, dcompl,dim);  // cálculo das sensibilidades
        filter(x,dcompl,femnts,radius,dim);  // filtragem das sensibilidades
        OC(x,dcompl);  // critério de optimalidade
        change=fabs(x[1]-xold[1]);
        sum+=x[1];
        for(i=1;i<M;){
                i++;
                if(fabs(x[i]-xold[i])>change) change=fabs(x[i]-xold[i]); // critério de convergência
                sum+=x[i];
        }
        obj[loop]=compl;
        vol[loop]=sum/M;
        printf("it:%d  Obj:%lf  Vol:%lf  ch:%lf\n", loop, compl, sum/M, change);
        if(loop%10==0 && loop!=0) Files(loop);
}
        free_ivector(PNP,1,  M);  // liberação de memoria
        free_dvector(obj,1,300);
        free_dvector(vol,1,300);
        free_ivector(colsizes,1,ksize);
        free_ivector(diagindx,1,ksize);
        free_dvector(x,1,M);
        free_dvector(xold,1,M);
        free_dvector(dcompl,1,M);
        free_dmatrix(ub,1,ksize,1,NCASE;  // fim do programa
```

```
**********Critério de Optimalidade**********
void OC(double *x, double *dcompl){
        double move=0.05;
        double l1=0;
        double l2=100000.0;
        double lmid;
        double sum;
        double *xnew=dvector(1,M);
        int i;
        while(l2-l1>1.0e-4){
            sum=0.0;
            lmid=0.5*(l2+l1);
            for(i=1;i<=M;i++){
                if(PNP[i]==0){
                xnew[i]=__max(0.001,__max(x[i]-move,__min(1.0,__min(x[i]+move,x[i]*sqrt(-dcompl[i]/lmid)))));
                        sum+=xnew[i];
                }
                else xnew[i]=1;
            }
                if((sum-volfrac*M)>0.0) l1=lmid;
                else l2=lmid;
        }
        for(i=1;i<=M;i++)  x[i]=xnew[i];
        free_dvector(xnew,1,M);
}
**********Função que cria o vetor COLSIZES**********
void Colsizes(int *colsizes){
        for(int i=1;i<=ksize;i++) colsizes[i]=0;
        if(dim==3){
                int *dofs=ivector(1,81);
                for(i=1;i<=ksize;i++){
                        for(int j=1;j<=81;j++) dofs[j]=0;
                        int cont=1;
                        int *DOFneighbours=ivector(1,8);
                        for(int k=1;k<=8;k++) DOFneighbours[k]=0;
                        DOFNeighbours(i,DOFneighbours, dim);
                        for(k=1;k<=8 && DOFneighbours[k]!=0;k++){
                                for(int z=1;z<=24;z++){
                                        int num=IDconect[DOFneighbours[k]][z];
                                        if(num!=0 && Contains(dofs,num,81)==0 && num<=i){
                                                dofs[cont]=num;
                                                cont++;
                                        }
                                }
                        }
                        colsizes[i]=cont-1;
                        free_ivector(DOFneighbours,1,8);
                }
                free_ivector(dofs,1,81);
        }
        else if(dim==2){
```

```
                int *dofs=ivector(1,18);
                for(i=1;i<=ksize;i++){
                        for(int j=1;j<=18;j++) dofs[j]=0;
                        int cont=1;
                        int *DOFneighbours=ivector(1,4);
                        for(int k=1;k<=4;k++) DOFneighbours[k]=0;
                        DOFNeighbours(i,DOFneighbours, dim);
                        for(k=1;k<=8 && DOFneighbours[k]!=0;k++){
                                for(int z=1;z<=8;z++){
                                        int num=IDconect[DOFneighbours[k]][z];
                                        if(num!=0 && Contains(dofs,num,18)==0 && num<=i){
                                                dofs[cont]=num;
                                                cont++;
                                        }
                                }
                        }
                        colsizes[i]=cont-1;
                        free_ivector(DOFneighbours,1,4);
                }
                free_ivector(dofs,1,18);
        }
}

**********Função que cria o vetor de vetores INDX**********
void INDX(int **indx, int *colsizes, int dim){
        if(dim==3){
                int *dofs=ivector(1,81);
                for(int i=1;i<=ksize;i++){
                        for(int j=1;j<=81;j++) dofs[j]=0;
                        int cont=1;
                        int *DOFneighbours=ivector(1,8);
                        for(int k=1;k<=8;k++) DOFneighbours[k]=0;
                        DOFNeighbours(i,DOFneighbours,dim);
                        for(k=1;k<=8 && DOFneighbours[k]!=0;k++){
                                for(int z=1;z<=24;z++){
                                        int num=IDconect[DOFneighbours[k]][z];
                                        if(num!=0 && Contains(dofs,num,81)==0 && num<=i){
                                                dofs[cont]=num;
                                                cont++;
                                        }
                                }
                        }
                        indx[i-1]=(int *)malloc((cont-1)*sizeof(int));
                        for(k=0;k<cont-1;k++) indx[i-1][k]=dofs[k+1]
                        colsizes[i]=cont-1;
                        free_ivector(DOFneighbours,1,8);
                }
                free_ivector(dofs,1,81);
        }
        else if(dim==2){
                int *dofs=ivector(1,18);
```

```
                for(int i=1;i<=ksize;i++){
                        for(int j=1;j<=18;j++) dofs[j]=0;
                        int cont=1;
                        int *DOFneighbours=ivector(1,4);
                        for(int k=1;k<=4;k++) DOFneighbours[k]=0;
                        DOFNeighbours(i,DOFneighbours,dim);
                        for(k=1;k<=4 && DOFneighbours[k]!=0;k++){
                                for(int z=1;z<=8;z++){
                                        int num=IDconect[DOFneighbours[k]][z];
                                        if(num!=0 && Contains(dofs,num,18)==0 && num<=i){
                                                dofs[cont]=num;
                                                cont++;
                                        }
                                }
                        }
                        indx[i-1]=(int *)malloc((cont-1)*sizeof(int));
                        for(k=0;k<cont-1;k++) indx[i-1][k]=dofs[k+1];
                        colsizes[i]=cont-1;
                        free_ivector(DOFneighbours,1,4);
                }
                free_ivector(dofs,1,18);
        }
}

   ************************Função que prepara a matris FEMNTS para o filtro de sensibilidades ************************

int **FEMNTS(double r, int dim){
        int rceil=(int)ceil(r);
        int k=0;
        int i,j,cont,size;
        int **m;
        double xel1,yel1,xel2,yel2,zel1,zel2,dist,elsize;
        for(i=1;i<=rceil;i++) k+=i-1;
        if(dim==2){
                size=4*(k+rceil)+1;
                elsize=fabs(coord[1][conect[1][1]]-coord[1][conect[1][2]]);
                m=imatrix(1,M,1,size);
                for(i=1;i<=M;i++){
                        for(j=1;j<=size;j++) m[i][j]=0;
                }
                for(i=1;i<=M;i++){
                   xel1=(coord[1][conect[i][1]]+coord[1][conect[i][2]]+coord[1][conect[i][3]]+coord[1][conect[i][4]])/4;
                   yel1=(coord[2][conect[i][1]]+coord[2][conect[i][2]]+coord[2][conect[i][3]]+coord[2][conect[i][4]])/4;
                   cont=1;
                      for(j=1;j<=M;j++){
                        xel2=(coord[1][conect[j][1]]+coord[1][conect[j][2]]+coord[1][conect[j][3]]+coord[1][conect[j][4]])/4;
                        yel2=(coord[2][conect[j][1]]+coord[2][conect[j][2]]+coord[2][conect[j][3]]+coord[2][conect[j][4]])/4;
                        dist=sqrt(pow(fabs(xel1-xel2),2)+pow(fabs(yel1-yel2),2));
                        if(i!=j && dist<=r*elsize){
                             m[i][cont]=j;
                             cont++
```

```
                                        }
                                        if(cont==size) break;
                                }
                                m[i][cont]=i;
                        }
                }
                else if(dim==3){
                        size=6*rceil+12*k+8*(int)pow((rceil-1),2)+1;
                        elsize=fabs(coord[1][conect[1][1]]-coord[1][conect[1][2]]);
                        m=imatrix(1,M,1,size);
                        printf("alocou\n");
                        for(i=1;i<=M;i++) for(j=1;j<=size;j++) m[i][j]=0;
                        for(i=1;i<=M;i++){
                        xel1=(coord[1][conect[i][1]]+coord[1][conect[i][2]]+coord[1][conect[i][3]]+coord[1][conect[i][4]]+coord[1][co
                        nect[i][5]]+coord[1][conect[i][6]]+coord[1][conect[i][7]]+coord[1][conect[i][8]])/8;

                        yel1=(coord[2][conect[i][1]]+coord[2][conect[i][2]]+coord[2][conect[i][3]]+coord[2][conect[i][4]]+coord[2][co
                        nect[i][5]]+coord[2][conect[i][6]]+coord[2][conect[i][7]]+coord[2][conect[i][8]])/8;

                        zel1=(coord[3][conect[i][1]]+coord[3][conect[i][2]]+coord[3][conect[i][3]]+coord[3][conect[i][4]]+coord[3][co
                        nect[i][5]]+coord[3][conect[i][6]]+coord[3][conect[i][7]]+coord[3][conect[i][8]])/8;
                        cont=1;
                        for(j=1;j<=M;j++){
                        xel2=(coord[1][conect[j][1]]+coord[1][conect[j][2]]+coord[1][conect[j][3]]+coord[1][conect[j][4]]+coord[1][co
                        nect[j][5]]+coord[1][conect[j][6]]+coord[1][conect[j][7]]+coord[1][conect[j][8]])/8;
                        yel2=(coord[2][conect[j][1]]+coord[2][conect[j][2]]+coord[2][conect[j][3]]+coord[2][conect[j][4]]+coord[2][co
                        nect[j][5]]+coord[2][conect[j][6]]+coord[2][conect[j][7]]+coord[2][conect[j][8]])/8;
                        zel2=(coord[3][conect[j][1]]+coord[3][conect[j][2]]+coord[3][conect[j][3]]+coord[3][conect[j][4]]+coord[3][co
                        nect[j][5]]+coord[3][conect[j][6]]+coord[3][conect[j][7]]+coord[3][conect[j][8]])/8;
                        dist=sqrt(pow(fabs(xel1-xel2),2)+pow(fabs(yel1-yel2),2)+pow(fabs(zel1-zel2),2));
                        if(j!=i && dist<=r*elsize){
                                m[i][cont]=j;
                                cont++;
                        }
                        if(cont==size) break;
                        }
                        m[i][cont]=i;
                        }
                }
                return m;
}

    ****************************************Filtro de Sensibilidades ****************************************

void filter(double *x, double *dcompl, int **femnts, double r, int dim){
        int i,j;
        double *dcn=dvector(1,M);
        double sum1,sum2,hf,elsize, xel1, yel1, zel1, xel2, yel2, zel2;
        elsize=fabs(coord[1][conect[1][1]]-coord[1][conect[1][2]]);
        if(dim==3){
                for(i=1;i<=M;i++){
                        sum1=0.0;
                        sum2=0.0;
```

```
                xel1=(coord[1][conect[i][1]]+coord[1][conect[i][2]]+coord[1][conect[i][3]]+coord[1][conect[i][4]]+
                coord[1][conect[i][5]]+coord[1][conect[i][6]]+coord[1][conect[i][7]]+coord[1][conect[i][8]])/8;

                yel1=(coord[2][conect[i][1]]+coord[2][conect[i][2]]+coord[2][conect[i][3]]+coord[2][conect[i][4]]+
                coord[2][conect[i][5]]+coord[2][conect[i][6]]+coord[2][conect[i][7]]+coord[2][conect[i][8]])/8;

                zel1=(coord[3][conect[i][1]]+coord[3][conect[i][2]]+coord[3][conect[i][3]]+coord[3][conect[i][4]]+
                coord[3][conect[i][5]]+coord[3][conect[i][6]]+coord[3][conect[i][7]]+coord[3][conect[i][8]])/8;

                for(j=1;femnts[i][j]!=i;j++){
                        xel2=(coord[1][conect[femnts[i][j]][1]]+coord[1][conect[femnts[i][j]][2]]+coord[1][conect[femnts[i][j]][3]]+coord[1][conect[femnts[i][j]][4]]+coord[1][conect[femnts[i][j]][5]]+coord[1][conect[femnts[i][j]][6]]+coord[1][conect[femnts[i][j]][7]]+coord[1][conect[femnts[i][j]][8]])/8;

                        yel2=(coord[2][conect[femnts[i][j]][1]]+coord[2][conect[femnts[i][j]][2]]+coord[2][conect[femnts[i][j]][3]]+coord[2][conect[femnts[i][j]][4]]+coord[2][conect[femnts[i][j]][5]]+coord[2][conect[femnts[i][j]][6]]+coord[2][conect[femnts[i][j]][7]]+coord[2][conect[femnts[i][j]][8]])/8;

                        zel2=(coord[3][conect[femnts[i][j]][1]]+coord[3][conect[femnts[i][j]][2]]+coord[3][conect[femnts[i][j]][3]]+coord[3][conect[femnts[i][j]][4]]+coord[3][conect[femnts[i][j]][5]]+coord[3][conect[femnts[i][j]][6]]+coord[3][conect[femnts[i][j]][7]]+coord[3][conect[femnts[i][j]][8]])/8;

                        hf=(r*elsize)-sqrt(pow(fabs(xel2-xel1),2)+pow(fabs(yel2-yel1),2)+pow(fabs(zel2-zel1),2));

                        sum1+=hf;
                        sum2+=x[femnts[i][j]]*hf*dcompl[femnts[i][j]];
                }
                sum1+=r*elsize;
                sum2+=r*elsize*x[i]*dcompl[i];
                dcn[i]=sum2/(x[i]*sum1);
        }
}
else if(dim==2){
        for(i=1;i<=M;i++){
                sum1=0.0;
                sum2=0.0;
                xel1=(coord[1][conect[i][1]]+coord[1][conect[i][2]]+coord[1][conect[i][3]]+coord[1][conect[i][4]])/4;
                yel1=(coord[2][conect[i][1]]+coord[2][conect[i][2]]+coord[2][conect[i][3]]+coord[2][conect[i][4]])/4;
                for(j=1;femnts[i][j]!=i;j++){
                        xel2=(coord[1][conect[femnts[i][j]][1]]+coord[1][conect[femnts[i][j]][2]]+coord[1][conect[femnts[i][j]][3]]+coord[1][conect[femnts[i][j]][4]])/4;

                        yel2=(coord[2][conect[femnts[i][j]][1]]+coord[2][conect[femnts[i][j]][2]]+coord[2][conect[femnts[i][j]][3]]+coord[2][conect[femnts[i][j]][4]])/4;

                        hf=(r*elsize)-sqrt(pow(fabs(xel2-xel1),2)+pow(fabs(yel2-yel1),2));
                        sum1+=hf;
                        sum2+=x[femnts[i][j]]*hf*dcompl[femnts[i][j]];
                }
                sum1+=r*elsize;
                sum2+=r*elsize*x[i]*dcompl[i];
                dcn[i]=sum2/(x[i]*sum1);
```

```
                }
        }
        for(i=1;i<=M;i++){
                dcompl[i]=dcn[i];
        }
        free_dvector(dcn,1,M);
}


void compliance(double **ub, double *compl){
        *compl=0;
        for(int j=1;j<=NCASE;j++){
                for(int i=1;i<=ksize;i++){
                        *compl+=ub[i][j]*fb[i][j];
                }
        }
}

        ************************************Cálculo da Compliância************************************
void compliance(double **ub, double *compl){
        *compl=0;
        for(int j=1;j<=NCASE;j++){
                for(int i=1;i<=ksize;i++) *compl+=ub[i][j]*fb[i][j];
        }
}

        ************************************Cálculo das sensibilidades************************************

void dcompliance(double **ub, double *dcompl, int dim){
        if(dim==3){
                double **ube=dmatrix(1,24,1,NCASE);
                double **aux=dmatrix(1,24,1,NCASE);
                double **desloc=dmatrix(1,3*NE,1,NCASE);
                int *ideg=ivector(1,24);
                int ijkia;
                for(int i=1;i<=M;i++){
                        dcompl[i]=0;
                }
                for(i=1;i<=3*NE;i++){
                        for(int j=1;j<=NCASE;j++){
                                if(ID[i]!=0) desloc[i][j]=ub[ID[i]][j];
                                else desloc[i][j]=0;
                        }
                }
                for(int k=1;k<=M;k++) {
                        for(i=1;i<=8;i++){
                                ijkia=conect[k][i];
                                xe[i]=coord[1][ijkia];
                                ye[i]=coord[2][ijkia];
                                ze[i]=coord[3][ijkia];
                                ideg[3*i-2]=3*ijkia-2;
```

```
                        ideg[3*i-1]=3*ijkia-1;
                        ideg[3*i]=3*ijkia;
                }
                SKE(xe,ye,ze,E,V,ske,dim);
                for(i=1;i<=24;i++){
                        for(int j=1;j<=NCASE;j++) ube[i][j]=desloc[ideg[i]][j]
                }
                for (i=1;i<=24;i++){
                        for(int j=1;j<=NCASE;j++) aux[i][j]=0;
                }
                for(int z=1;z<=NCASE;z++){
                        for (i=1;i<=24;i++){
                                for (int j=1;j<=24;j++) aux[i][z]+=ube[j][z]*ske[i][j];
                        }
                }
                for(int j=1;j<=NCASE;j++){
                        for(i=1;i<=24;i++) dcompl[k]=dcompl[k]-penal*pow(x[k],penal-1)*ube[i][j]*aux[i][j];
                }
        }
        free_dmatrix(ube,1,24,1,NCASE);
        free_dmatrix(aux,1,24,1,NCASE);
        free_dmatrix(desloc,1,3*NE,1,NCASE);
        free_dvector(xe,1,8);
        free_dvector(ye,1,8);
        free_dvector(ze,1,8);
        free_ivector(ideg,1,24);
        free_dmatrix(ske,1,24,1,24);
}
else if(dim==2){
        double **ube=dmatrix(1,8,1,NCASE);
        double **aux=dmatrix(1,8,1,NCASE);
        double **desloc=dmatrix(1,2*NE,1,NCASE);
        int *ideg=ivector(1,8);
        int ijkia;
        for(int i=1;i<=M;i++) dcompl[i]=0;
        for(i=1;i<=2*NE;i++){
                for(int j=1;j<=NCASE;j++){
                        if(ID[i]!=0) desloc[i][j]=ub[ID[i]][j];
                        else desloc[i][j]=0;
                }
        }
        for(int k=1;k<=M;k++) {
                for(i=1;i<=4;i++){
                        ijkia=conect[k][i];
                        xe[i]=coord[1][ijkia];
                        ye[i]=coord[2][ijkia];
                        ideg[2*i-1]=2*ijkia-1;
                        ideg[2*i]=2*ijkia;
                }
                SKE(xe,ye,ze,E,V,ske,dim);
```

```
                    for(i=1;i<=8;i++){
                        for(int j=1;j<=NCASE;j++) ube[i][j]=desloc[ideg[i]][j];
                    }
                    for (i=1;i<=8;i++){
                        for(int j=1;j<=NCASE;j++) aux[i][j]=0;
                    }
                    for(int z=1;z<=NCASE;z++){
                        for (i=1;i<=8;i++){
                            for (int j=1;j<=8;j++) aux[i][z]+=ube[j][z]*ske[i][j];
                        }
                    }
                    for(int j=1;j<=NCASE;j++){
                        for(i=1;i<=8;i++) dcompl[k]=dcompl[k]-penal*pow(x[k],penal-1)*ube[i][j]*aux[i][j];
                    }
                }
                free_dmatrix(ube,1,8,1,NCASE);
                free_dmatrix(aux,1,8,1,NCASE);
                free_dmatrix(desloc,1,2*NE,1,NCASE);
                free_dvector(xe,1,4);
                free_dvector(ye,1,4);
                free_dvector(ze,1,4);
                free_ivector(ideg,1,8);
                free_dmatrix(ske,1,8,1,8);
        }
}

        ************************************Rotina de MEF ************************************
void MEF (int **indx, int *colsizes, double **ub, double *x,int NCASE, int dim){
        if(dim==3){
                ske=dmatrix(1,24,1,24);
                xe=dvector(1,8);
                ye=dvector(1,8);
                ze=dvector(1,8);
                int *ideg=ivector(1,24);
                double *ubaux=dvector(1,ksize);
                double *fbaux=dvector(1,ksize);
                int i,j,k,jbia1,ijkia;
                K=(double **)malloc(ksize*sizeof(double *));
                for(i=1;i<=ksize;i++) K[i-1]=(double *)malloc((colsizes[i])*sizeof(double));
                for (i=1; i<=ksize;i++)
                for (j=1;j<=colsizes[i];j++) K[i-1][j-1]=0.0;
                printf("montando a matriz K...\n");
                for(k=1;k<=M;k++){
                        for(i=1;i<=8;i++){
                                ijkia=conect[k][i];
                                xe[i]=coord[1][ijkia];
                                ye[i]=coord[2][ijkia];
                                ze[i]=coord[3][ijkia];
                                ideg[3*i-2]=3*ijkia-2;
                                ideg[3*i-1]=3*ijkia-1;
                                ideg[3*i]=3*ijkia;
```

```
                    }
                    /* matriz de rigidez local */
                    SKE(xe,ye,ze,E,V,ske,dim);
                    /* formação da matriz global */
                    for (i=1;i<=24;i++){
                        ijkia=ideg[i];
                        if(ID[ijkia]!=0){
                                for (j=1;j<=24;j++){
                                        jbia1=ideg[j];
                                        if(ID[jbia1]!=0){
                                                for(int t=1;t<=colsizes[ID[jbia1]];t++){
                                                if(indx[ID[jbia1]-1][t-1]==ID[ijkia]){
                                                        K[ID[jbia1]-1][t-1]+=ske[i][j]*pow(x[k],penal)
                                                        break;
                                                        }
                                                }
                                        }
                                }
                            }
                    }
            } /* fim do loop da matriz global */
            free_ivector(ideg,1,8);
            printf("resolvendo o sistema\n");
            for(i=1;i<=NCASE;i++){
                    for(j=1;j<=ksize;j++){
                            ubaux[j]=0;
                            fbaux[j]=fb[j][i];
                    }
                    linbcg(ksize, fbaux, ubaux,ITOL,TOL, ITMAX, &iter, &err);
                    for(j=1;j<=ksize;j++)
                            ub[j][i]=ubaux[j];
            }

            free_dvector(ubaux,1,ksize);
            free_dvector(fbaux,1,ksize);
            for(i=1;i<=ksize;i++){
                    free(K[i-1]);
            }
}
else if(dim==2){
            ske=dmatrix(1,8,1,8);
            xe=dvector(1,4);
            ye=dvector(1,4);
            ze=dvector(1,4);
            int *ideg=ivector(1,8);
            double *ubaux=dvector(1,ksize);
            double *fbaux=dvector(1,ksize);
            int i,j,k,jbia1,ijkia;
            K=(double **)malloc(ksize*sizeof(double *));
            for(i=1;i<=ksize;i++) K[i-1]=(double *)malloc((colsizes[i])*sizeof(double));
```

```
//inicialização da matriz de rigidez global
for (i=1; i<=ksize;i++)
        for (j=1;j<=colsizes[i];j++)
                K[i-1][j-1]=0.0;
// matriz de rigidez global
printf("montando a matriz K...\n");
for(k=1;k<=M;k++){
        for(i=1;i<=4;i++){
                ijkia=conect[k][i];
                xe[i]=coord[1][ijkia];
                ye[i]=coord[2][ijkia];
                ideg[2*i-1]=2*ijkia-1;
                ideg[2*i]=2*ijkia;
        }
        /* matriz de rigidez local */
        SKE(xe,ye,ze,E,V,ske,dim);
        /* formação da matriz global */
        for (i=1;i<=8;i++){
            ijkia=ideg[i];
            if(ID[ijkia]!=0){
                for (j=1;j<=8;j++){
                        jbia1=ideg[j];
                        if(ID[jbia1]!=0){
                                for(int t=1;t<=colsizes[ID[jbia1]];t++){
                                        if(indx[ID[jbia1]-1][t-1]==ID[ijkia]){
                                            K[ID[jbia1]-1][t-1]+=ske[i][j]*pow(x[k],penal);
                                            break;
                                        }
                                }
                        }
                }
            }
        }
} /* fim do loop da matriz global */
free_ivector(ideg,1,8);
printf("resolvendo o sistema\n");
for(i=1;i<=NCASE;i++){
        for(j=1;j<=ksize;j++){
                ubaux[j]=0;
                fbaux[j]=fb[j][i];
        }
        linbcg(ksize, fbaux, ubaux,ITOL,TOL, ITMAX, &iter, &err);
        for(j=1;j<=ksize;j++) ub[j][i]=ubaux[j];
}
free_dvector(ubaux,1,ksize);
free_dvector(fbaux,1,ksize);
for(i=1;i<=ksize;i++) free(K[i-1]);
    }
}
```

7.REFERÊNCIAS BIBLIOGRÁFICAS

- Silva, E.C.N., 2003, "Otimização Aplicada ao Projeto de Sistemas Mecânicos",Departamento de Engenharia Mecatrônica e Sistemas Mecânicos da Escola Politécnica da USP.
- Lima, C.R., 2002, "Projeto de Mecanismos Flexíveis Utilizando o Método de Otimização Topológica", Escola Politécnica da Universidade de São Paulo, São Paulo.
- Cardoso. E.L., 2000, Conrtole de Complexidade na Otimização Topológica de Estruturas Contínuas", Universidade Federal do Rio Grande do Sul, Porto Alegre.
- Bendsoe, M.P., Sigmund, O., 2003, "Topology – Optimization – Theory, Methods And Applications", Springer, Lingby, Denmaek.

Desenvolvimento de software de Otimização Topológica para estruturas tridimensionais

Aluno:James Edward Shooter

Orientador: Prof.Dr.Emílio Carlos Nelli Silva

PMR-2500 Trabalho de Formatura

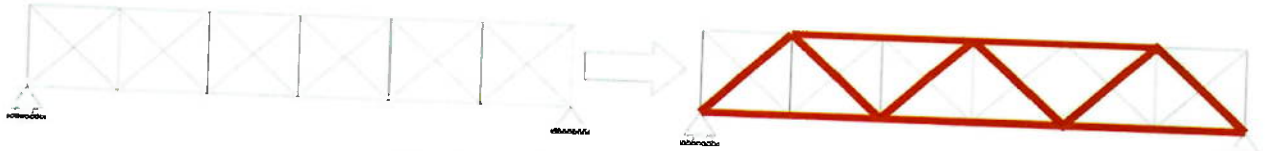
2º Semestre de 2003

Sequência da apresentação

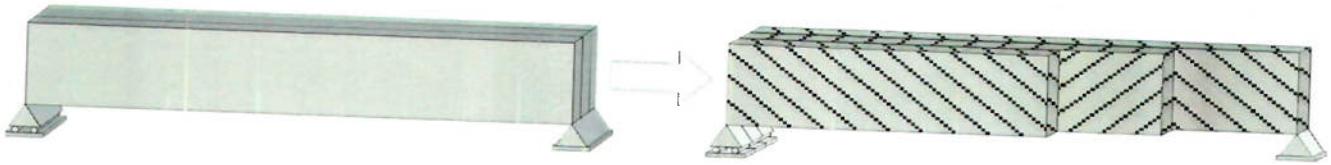
- Introdução
  - **Tipos de Otimização**
  - **Exemplos de aplicação do MOT**
- Objetivos
- Motivação
- Fundamentos Teóricos
  - **Otimização**
  - **Formulação de MEF utilizada**
  - **Critério de Optimalidade**
  - **Formulaçao Multi-Caso**
  - **Filtro**
- Implementação
  - **Fluxograma**
  - **Matriz esparsa**
- Resultados
- Conclusões

1.Introdução

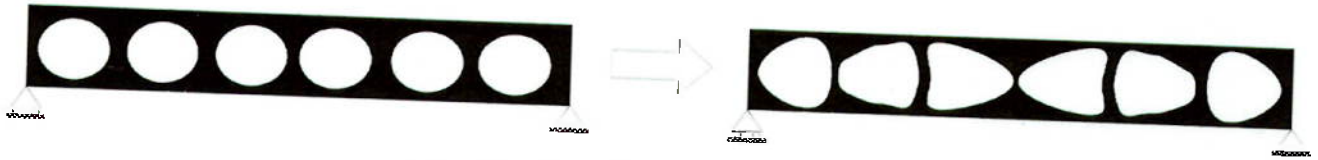
1.1Tipos de Otimização



A) Otimização de Tamanho



B) Otimização de Material



C) Otimização de Forma

D) Otimização Topológica

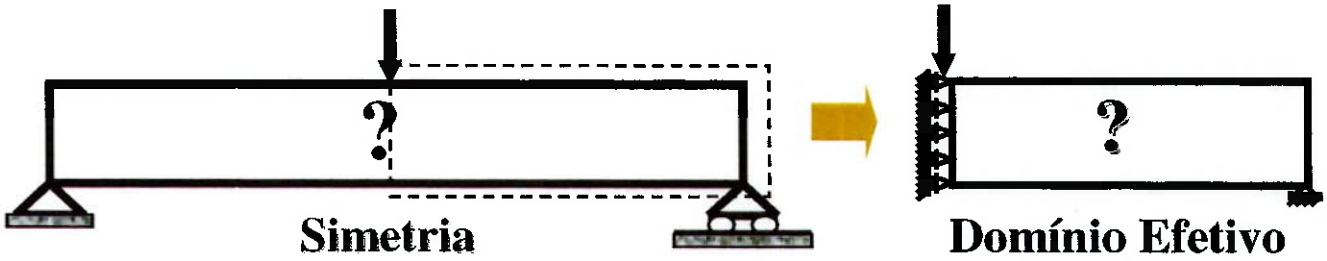
1.2.Exemplos de Aplicação do MOT

- Síntese de Estruturas mecânicas com menor peso e maior rigidez.
- Síntese de MEMS (MicroEletroMecanismos).
- Síntese de materiais com propriedades extremas.
- Maximização da frequëncia de ressonância em estruturas
- Resistência à flambagem
- Aplicações em diversas áreas da física (maximização da condução térmica, minimização da resistência ao escoamento).

2.Objetivos

- Obtenção de estruturas mecânicas tridimensionais leves e rígidas.
- Desenvolvimento de um software de otimização topológica para uso acadêmico.
- Solução de problemas multi-caso.
- Obter topologias bem definidas, com malhas refinadas.

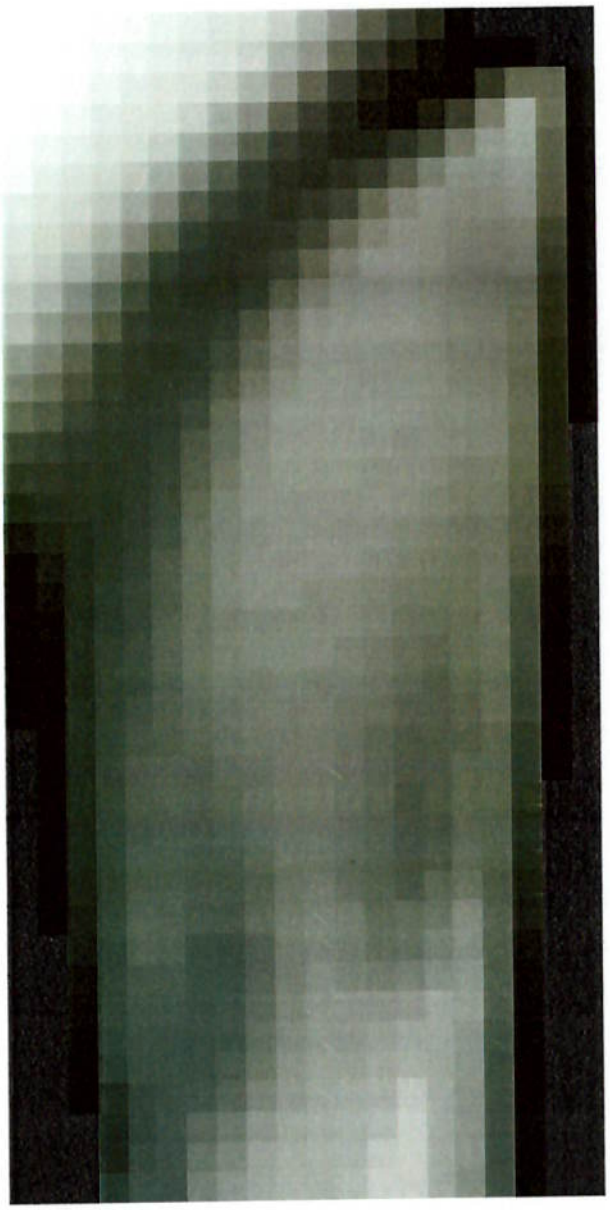
Exemplo de Otimização Topológica bidimensional

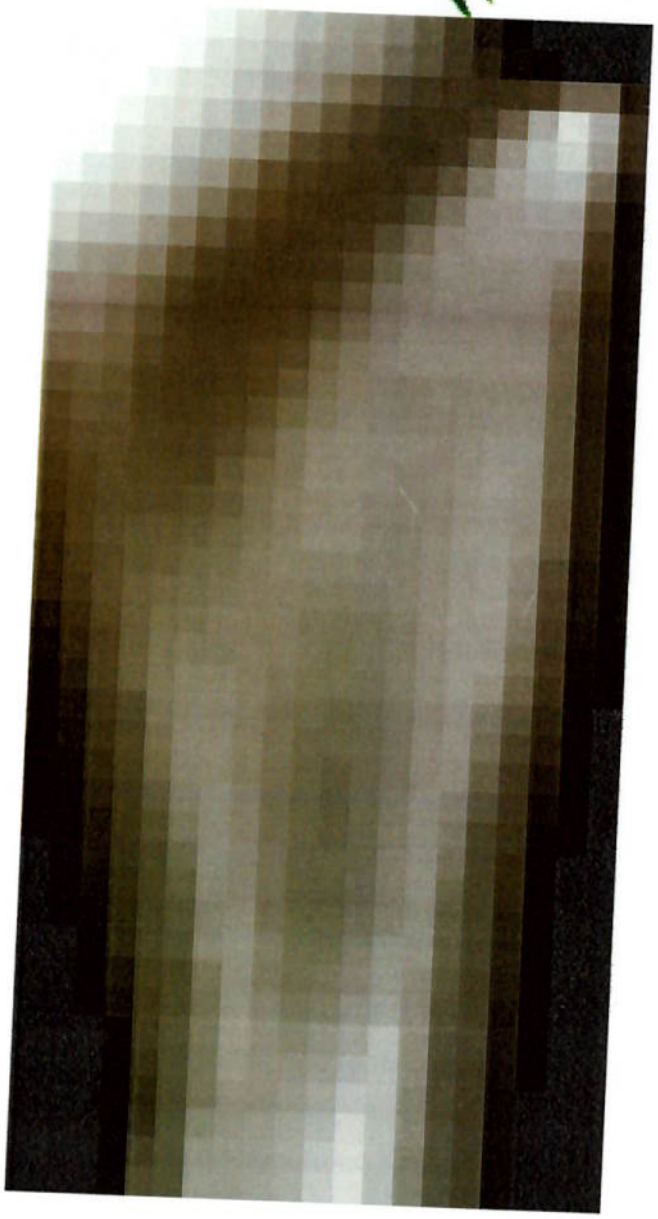


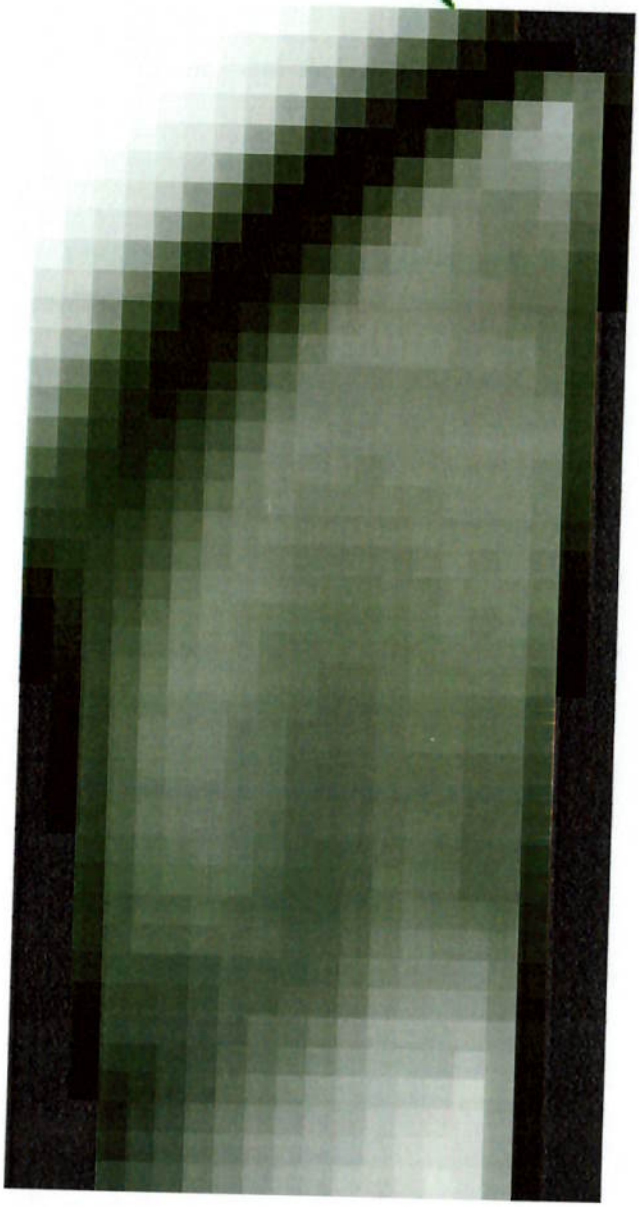
Simetria

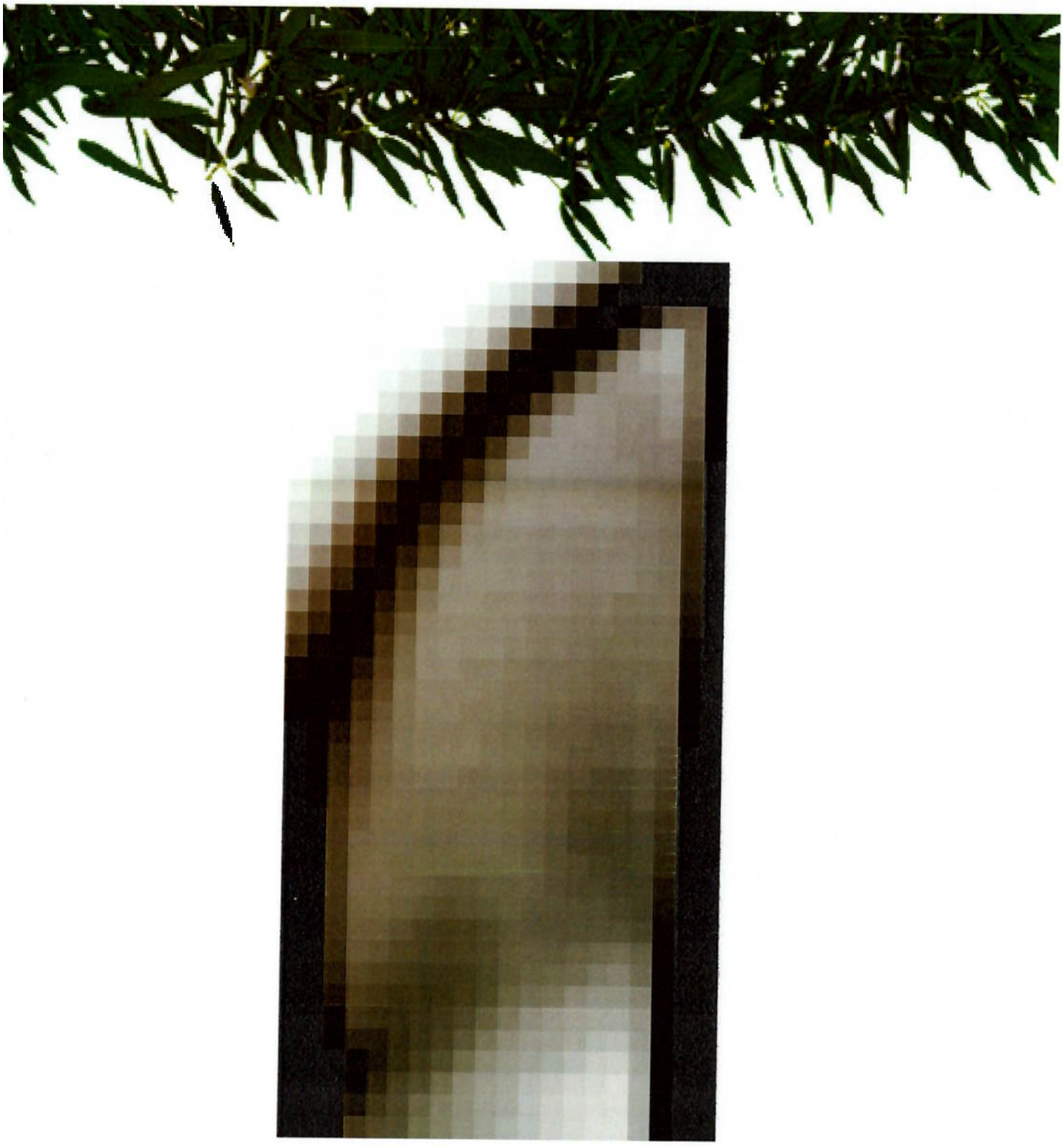
Domínio Efetivo

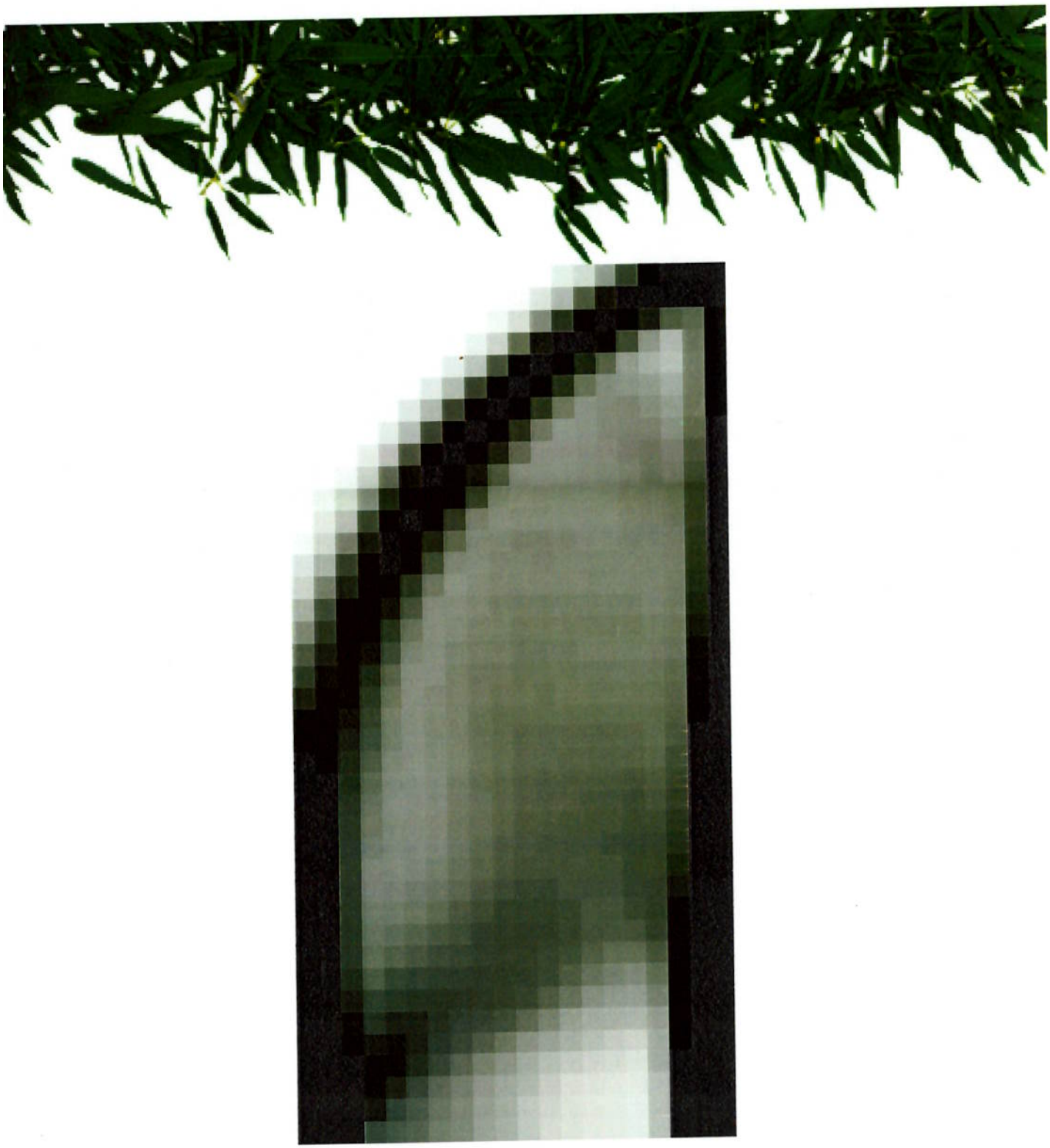
Resultado Obtido

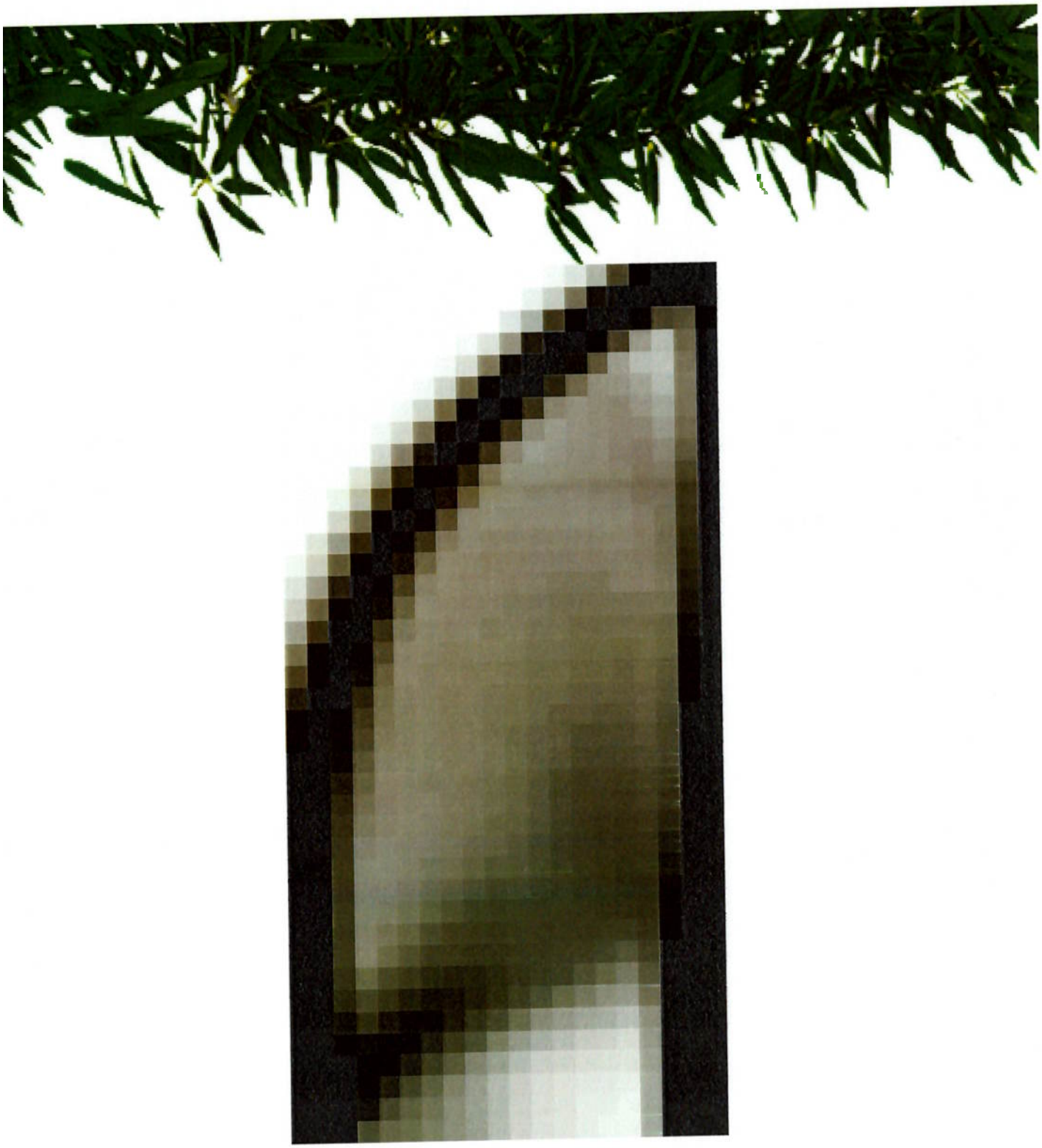


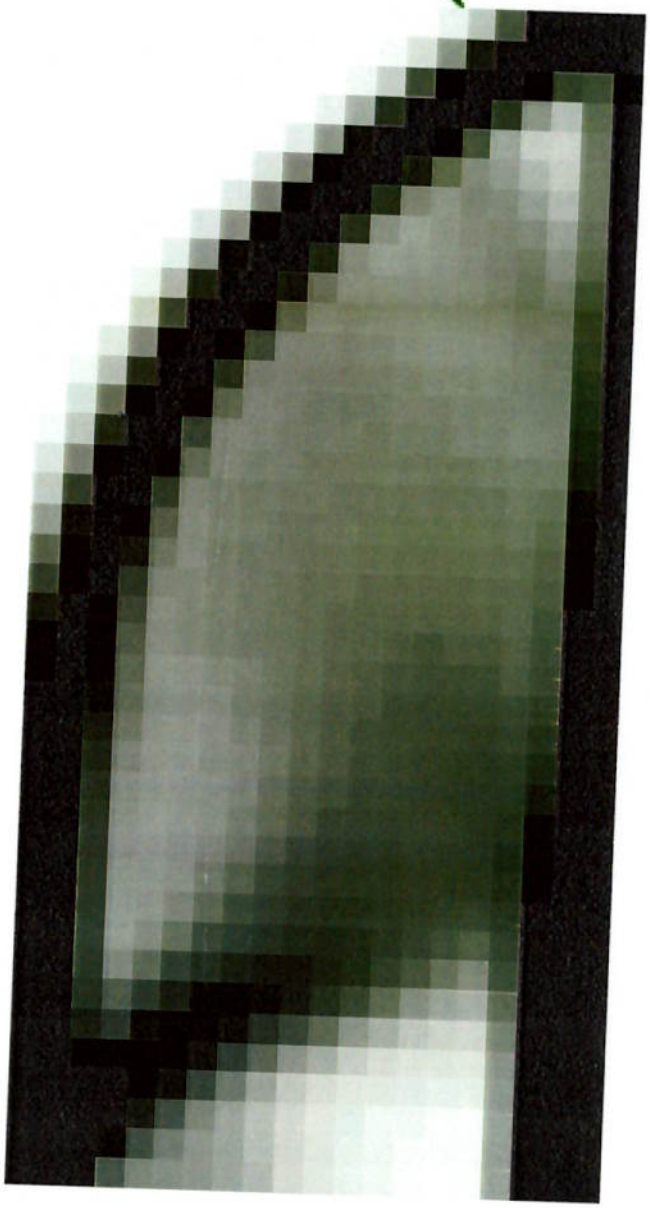


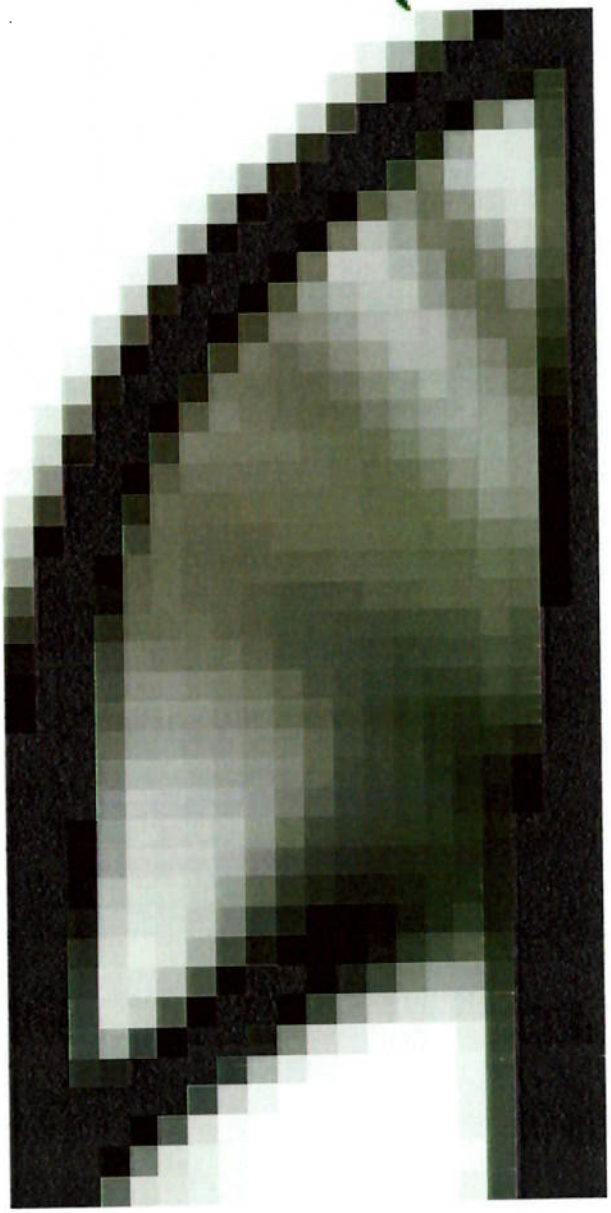


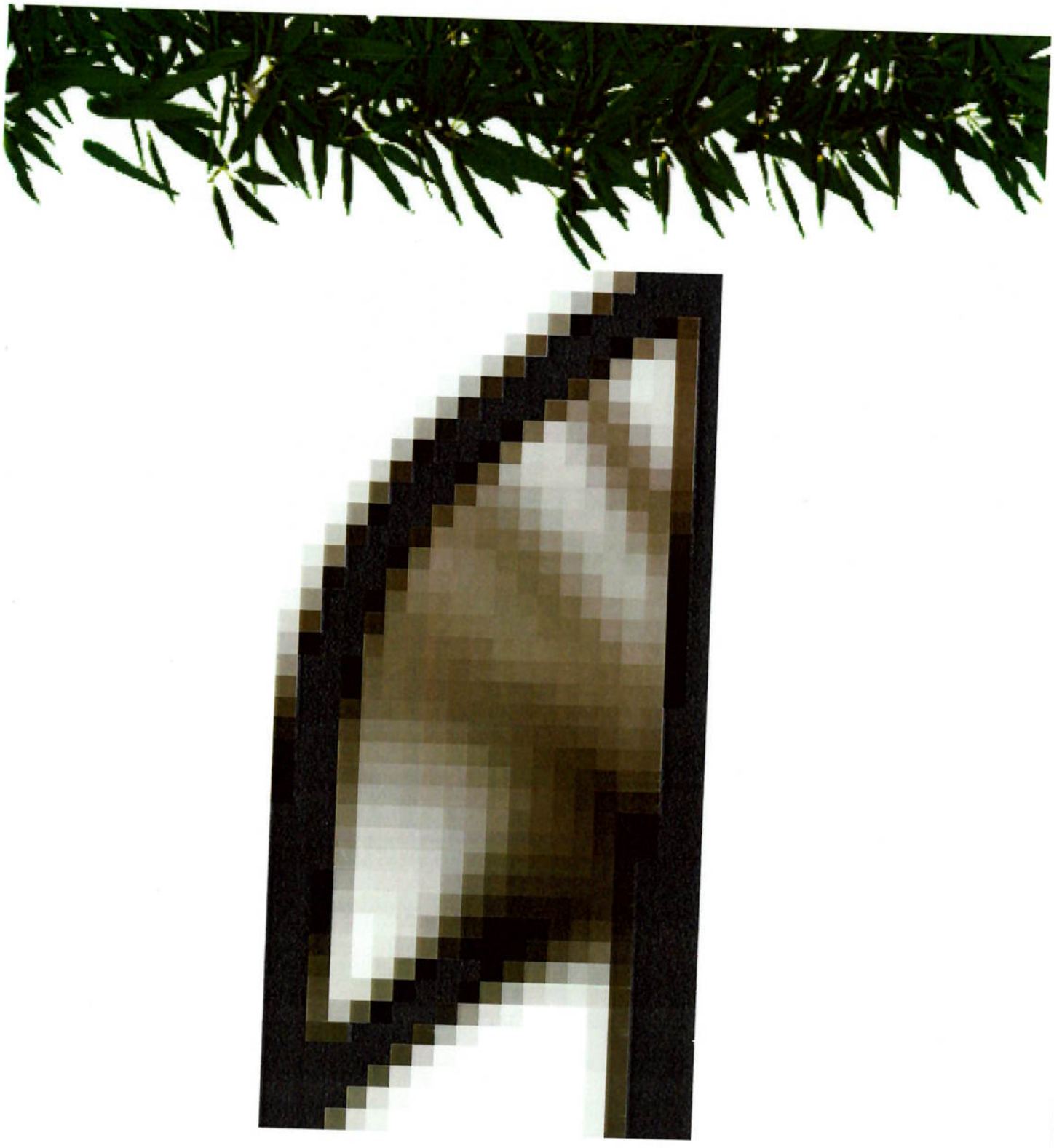


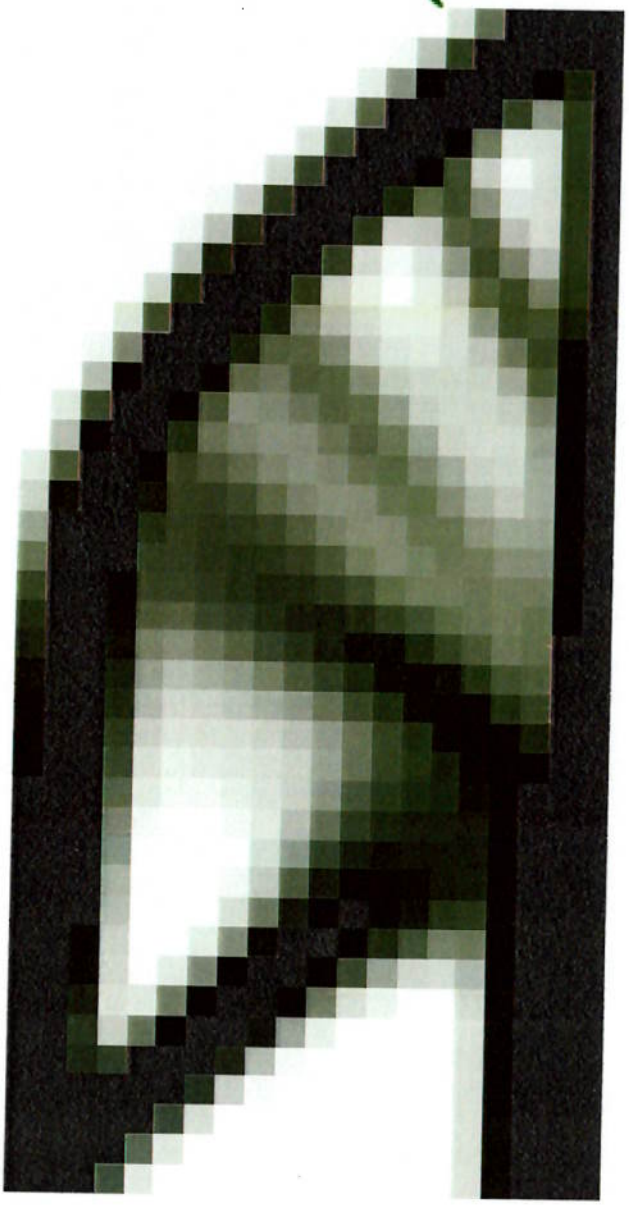


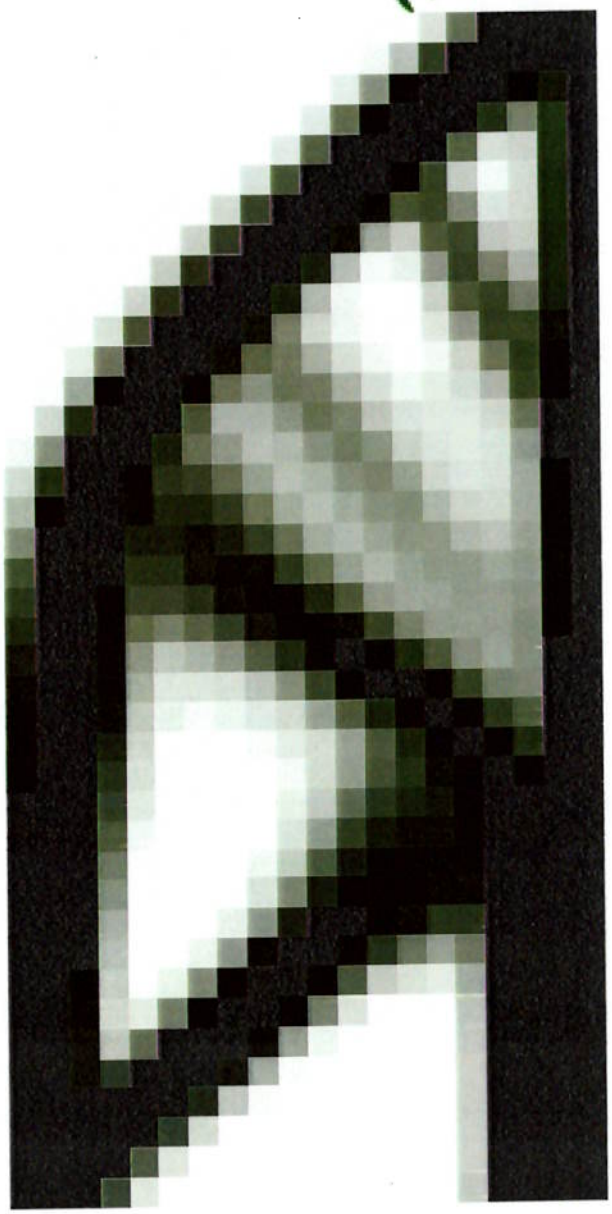


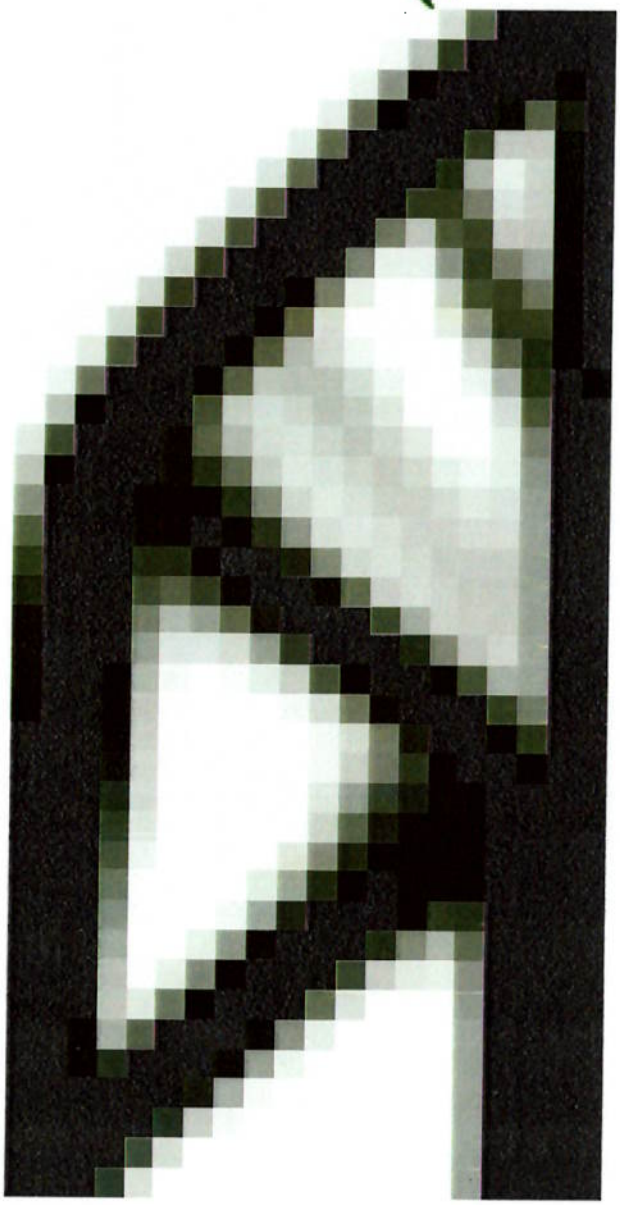


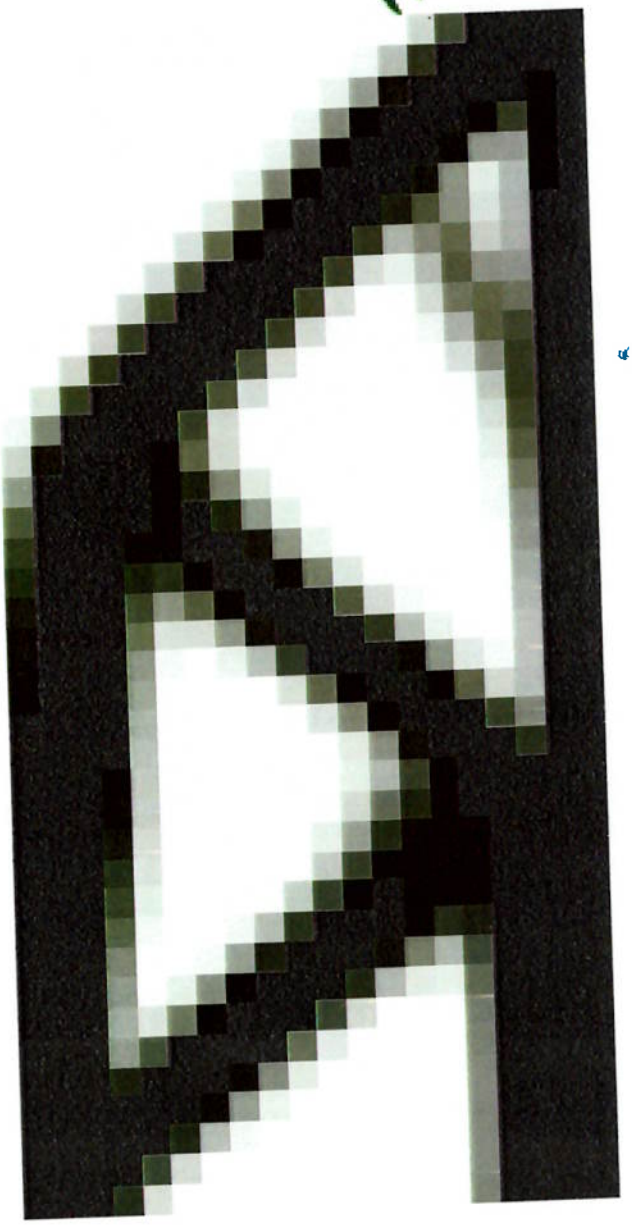


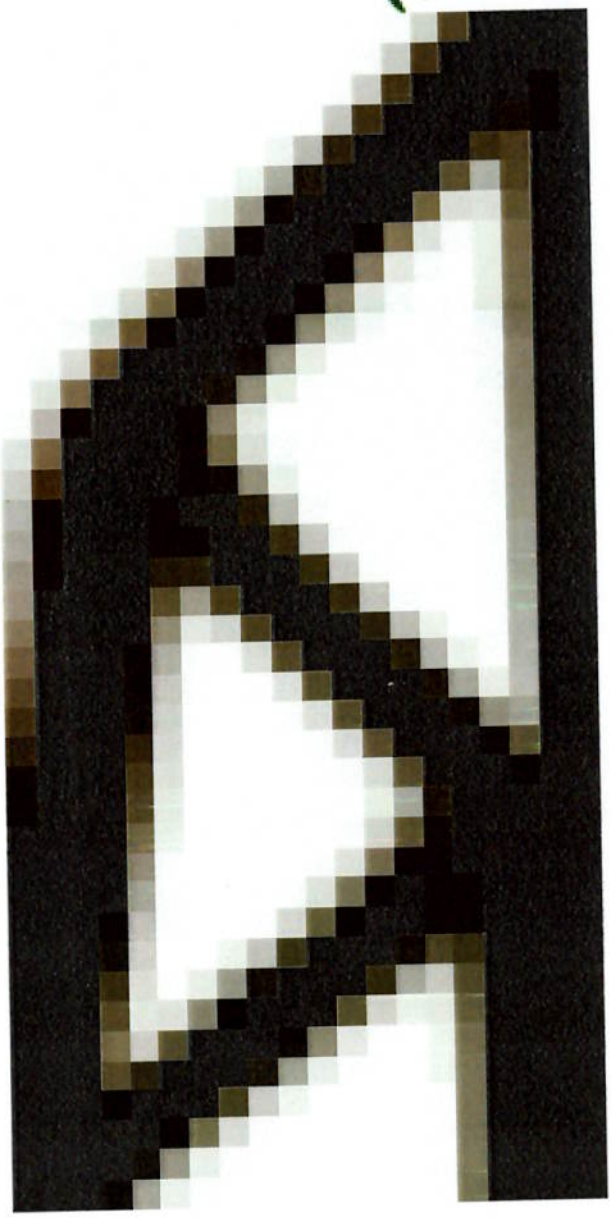






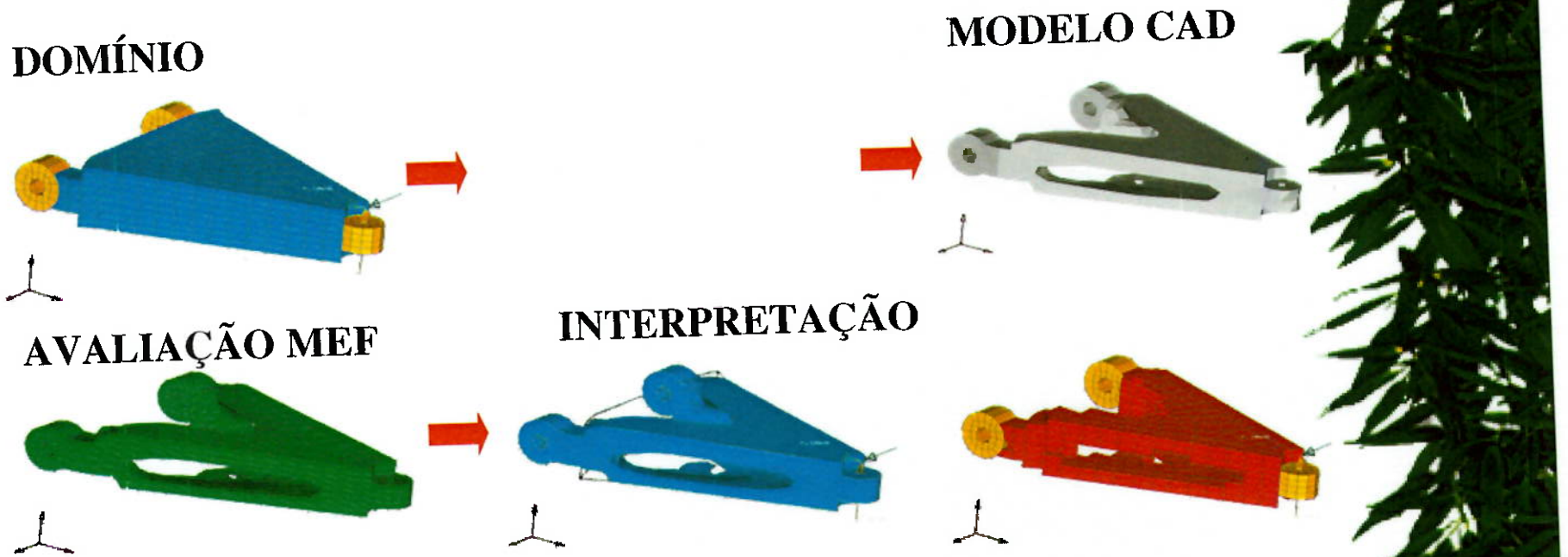






3.Motivação

- Necessidade de um software de uso acadêmico
- Uso cada vez mais frequente, na indústria, de sistemas de CAO (Computer Aided Optimizer)
- Economia de material na indústria em massa
- Melhor desempenho de estruturas na indústria aeronáutica e aeroespacial
- Melhor desempenho de estruturas na indústria automotiva



4.Fundamentação Teórica

4.1.Formulação do Problema de Otimização

$$Min \quad U^T KU$$

$$Suj \quad V = \sum_{I=1}^{ne} V_i \leq V_{\lim}$$

$$KU = F$$

U – vetor de deslocamentos dos nós da malha

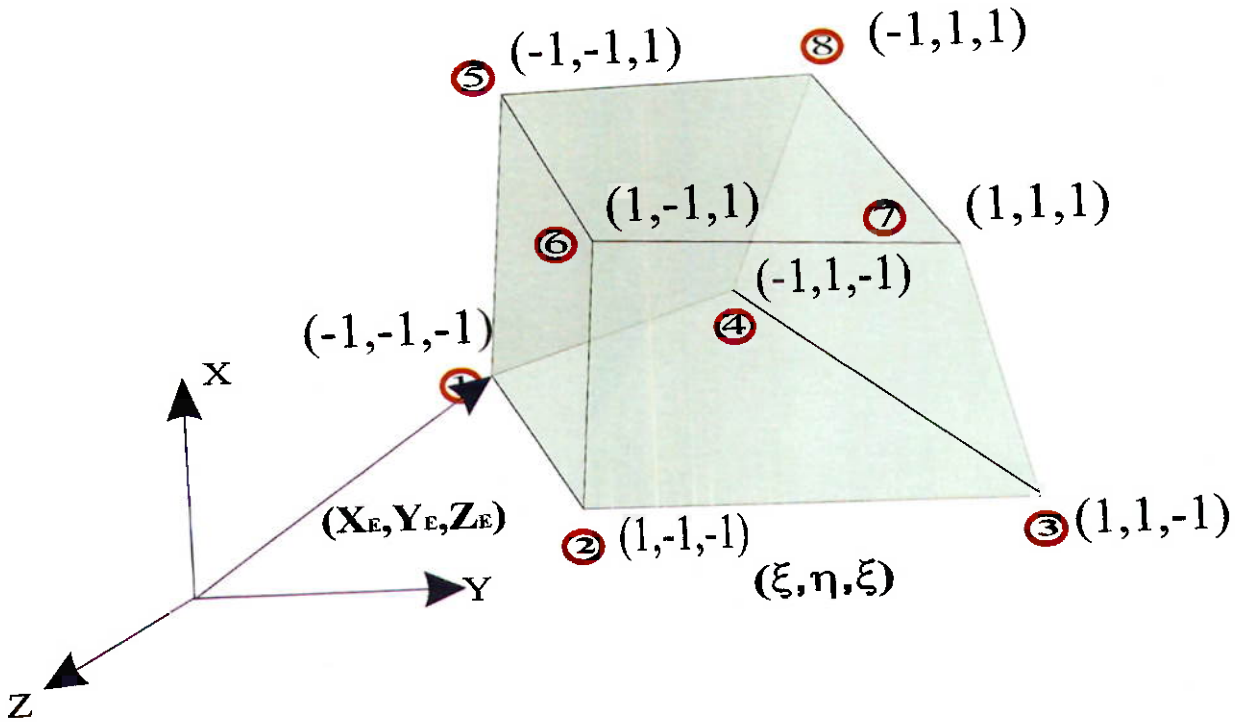
K – matriz de rigidez da malha

F – vator de carregamentos

V_i – volume do elemento i

- **As variáveis de projeto são as "densidades" em cada ponto (elemento) do domínio de projeto.**
- **Inicialmente, a densidade em cada elemento pode assumir os valores 0 (ausência de material) ou 1 (presença de material).**
- **Essa abordagem gera problemas devido à sua característica discreta.**
- **Solução: as variáveis de projeto podem assumir qualquer valor entre 0 e 1.**

4.2.Formulaçao de MEF utilizada

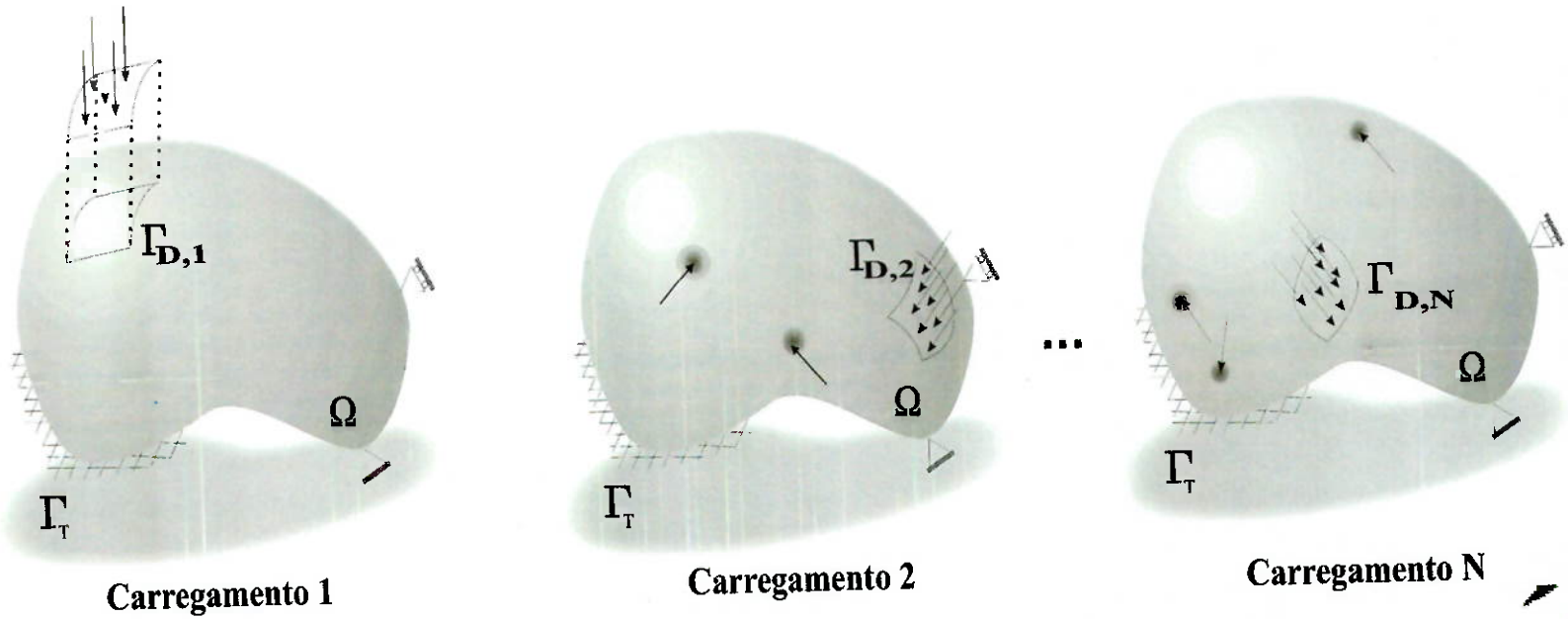


- **Elemento cúbico isoparamétrico de 8 nós**
- **Uso de integração seletiva**
- **Solução do sistema de equações lineares pelo método dos *Gradientes Co***

4.3.Critério de Optimalidade

- Cada variável de projeto é atualizada independentemente das outras
- Algoritmo heurístico baseado na optimalidade da função Lagrangeana.
- Muito usado na literatura de Otimização Topológica.
- Permite um tempo menor de processamento em relação a algoritmos baseados em programação matemática.

4.4 Função Objetivo Multi-caso

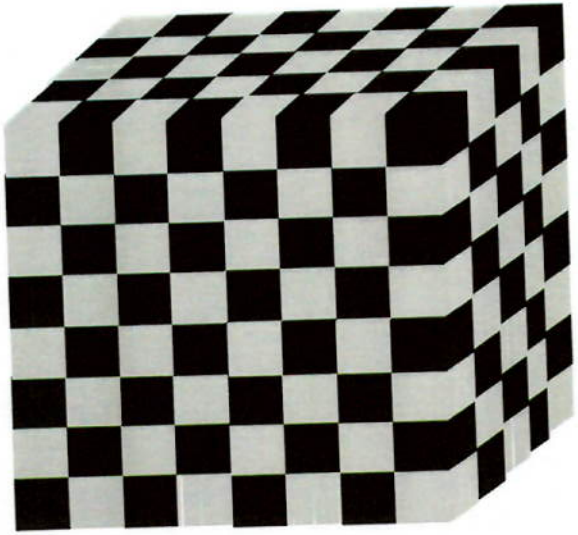


Carregamento 1 Carregamento 2 Carregamento N

- Função objetivo é função ponderada das flexibilidades nos casos
- Os carregamentos são não-simultâneos
- Solução para qualquer número de carregamentos

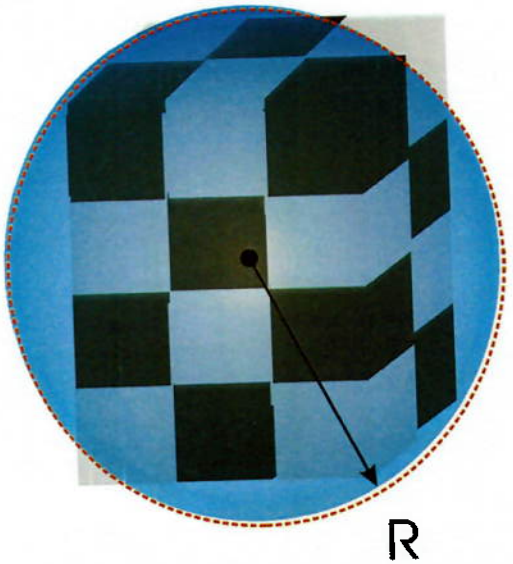
4.5.Filtro

- *Problema*: Formação do tabuleiro de xadrez
- *Solução*: Uso de um filtro de densidades



Elementos de alta densidade

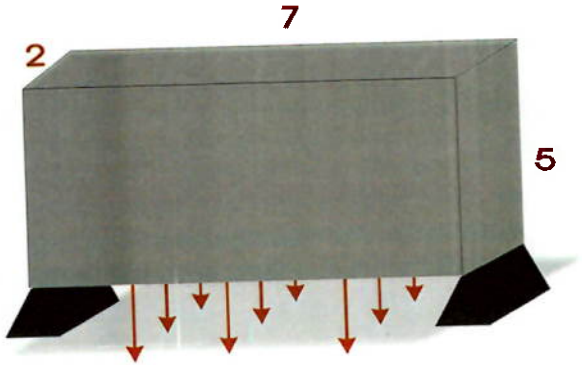
Elementos de baixa densidade

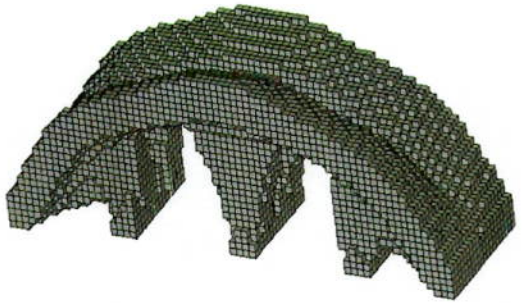


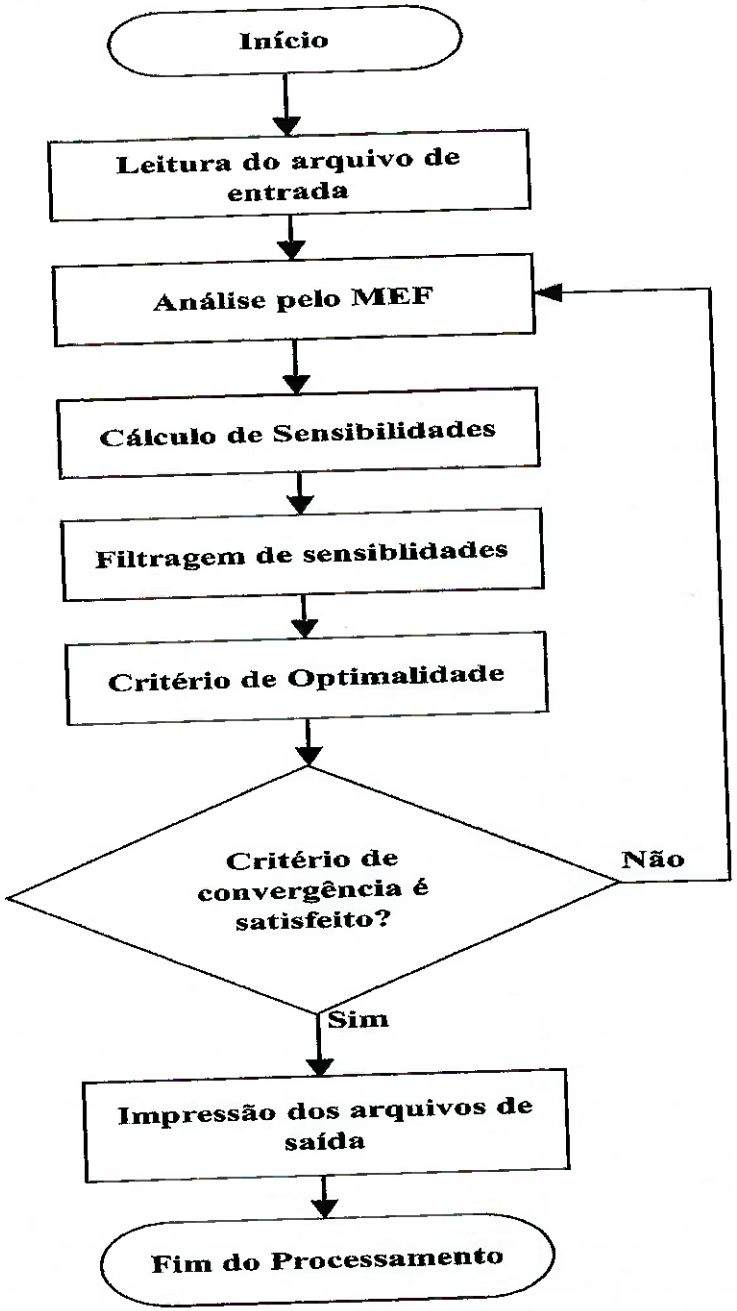
Elementos utilizados no filtro

5.Implementação

5.1.Fluxograma

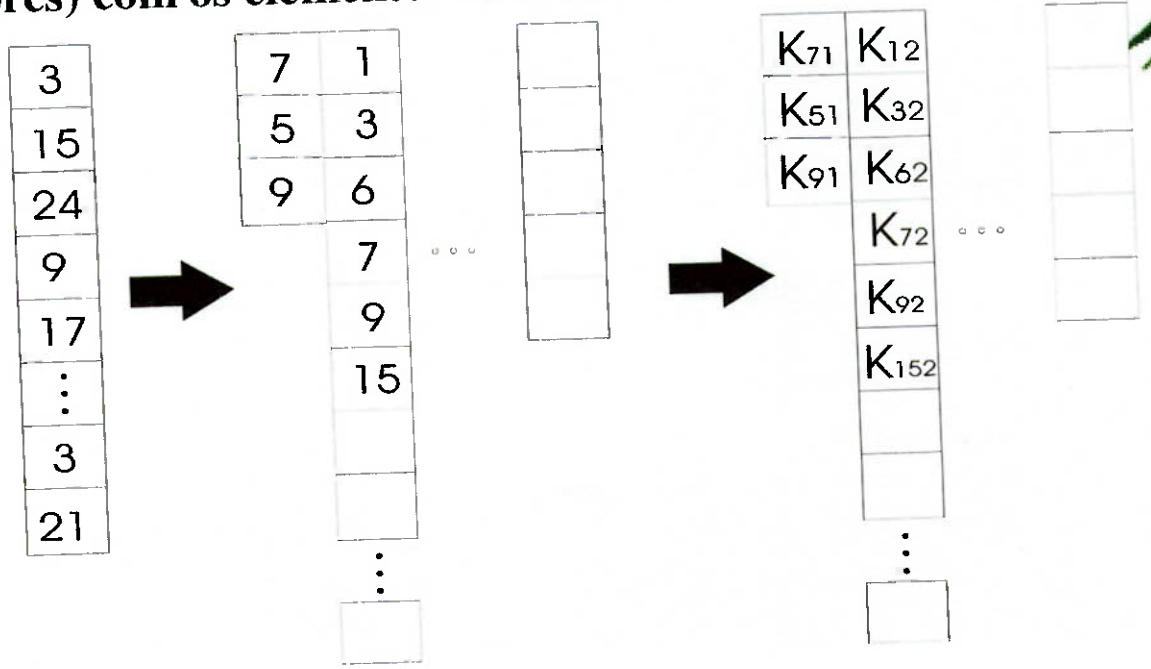






5.2.Matriz Esparsa

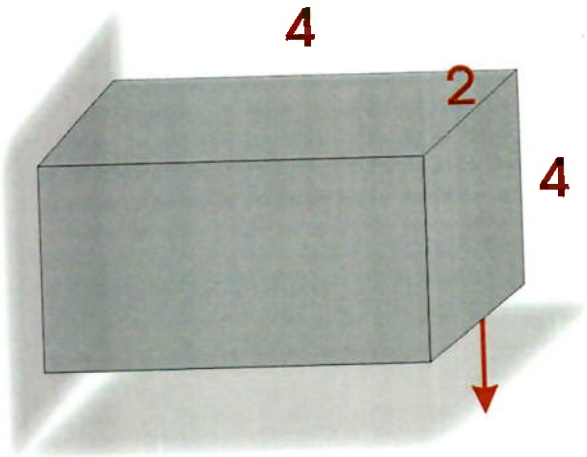
- *Problema*: matriz de rigidez para a malha de elementos cúbicos ocupa muito espaço na memória.
- *Solução*: armazenar somente os elementos não-nulos da matriz de rigidez global.
  - Cria-se um vetor contendo o número de elementos não-nulos em cada coluna da matriz de rigidez.
  - Cria-se um vetor de vetores de tamanho variável, cada vetor contendo a posição dos elementos não nulos da respectiva coluna Da matriz de rigidez
  - Com esse vetor de vetores é montada a matriz de rigidez (vetor De vetores) com os elementos não-nulos.

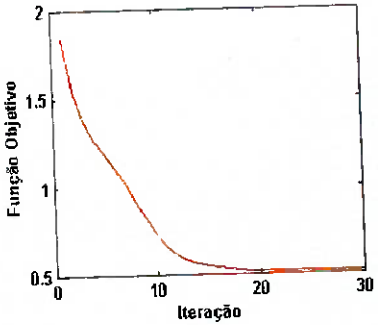


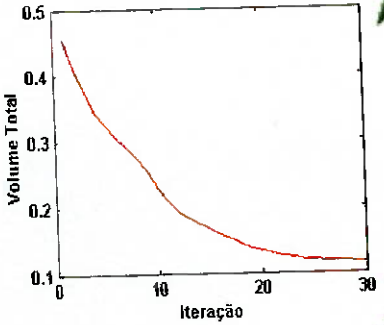
6.Resultados

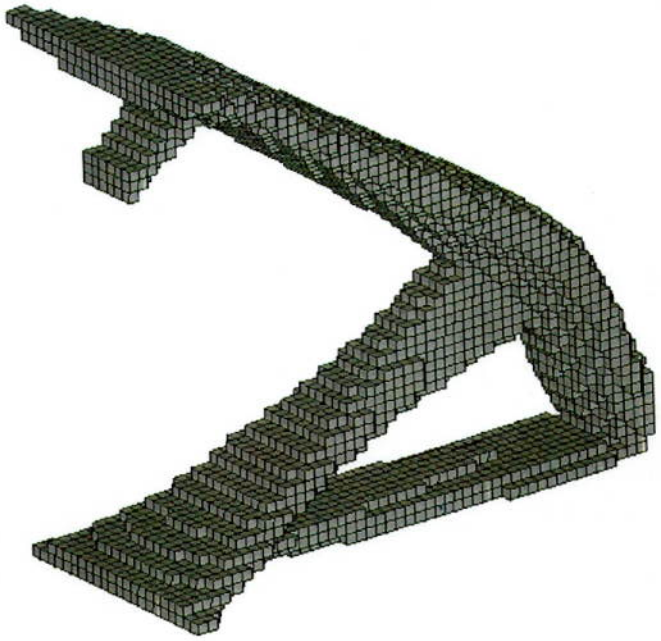
Discretização:65.000 elementos

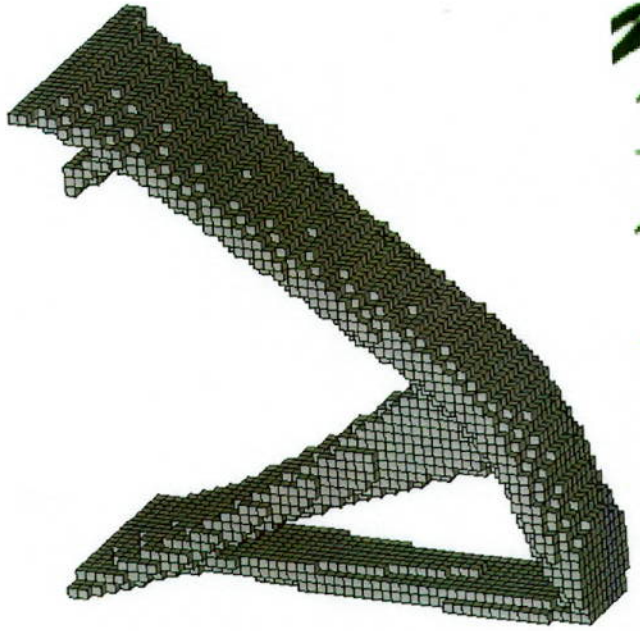
Tempo de Processamento:37 horas





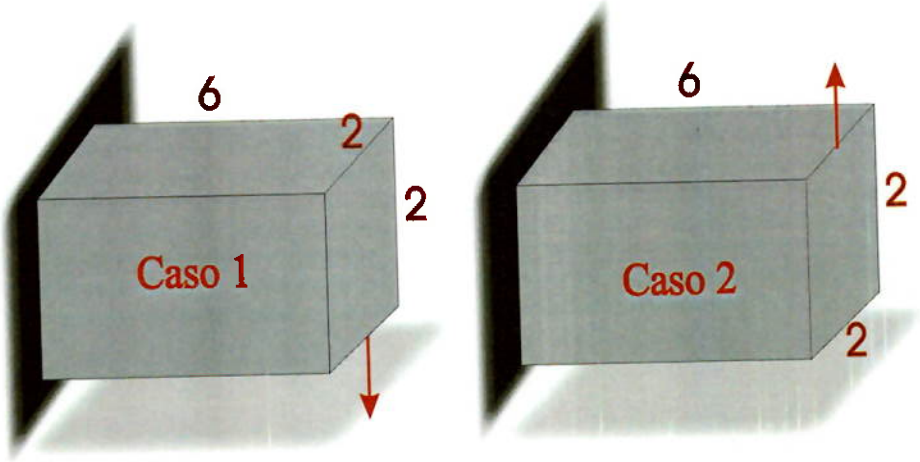


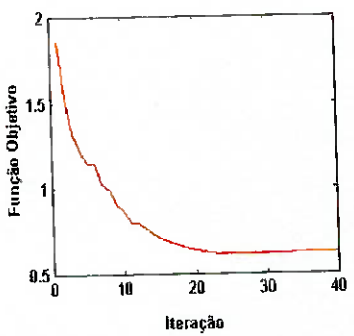


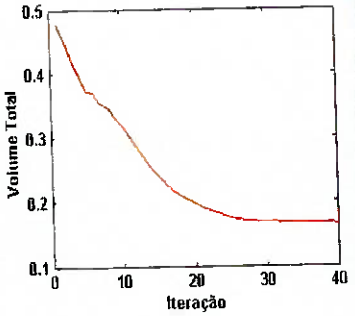


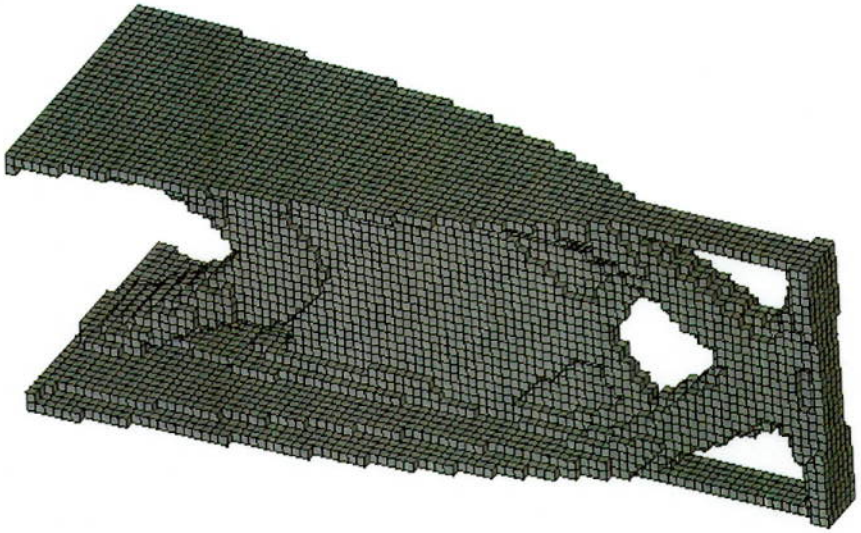
Discretização: 73.000 elementos

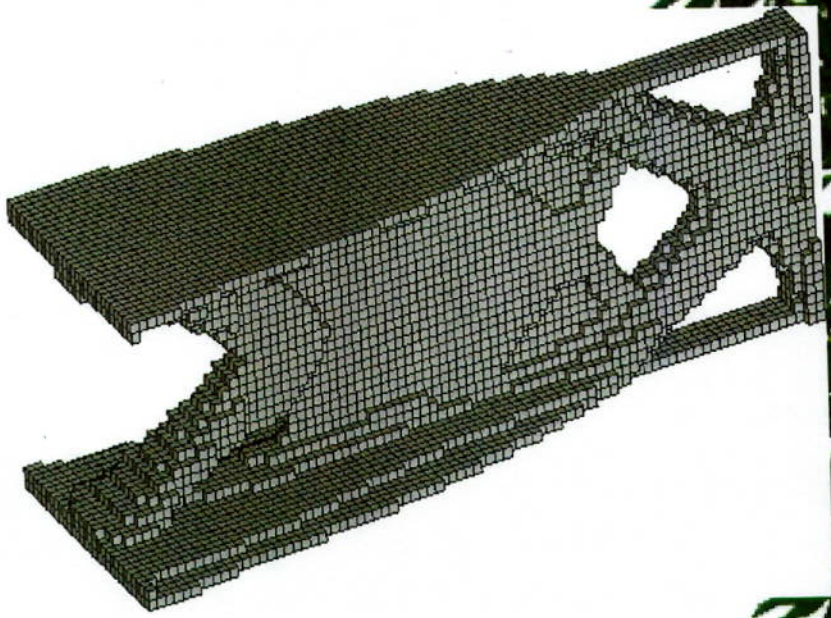
Tempo de Processamento: 48 horas





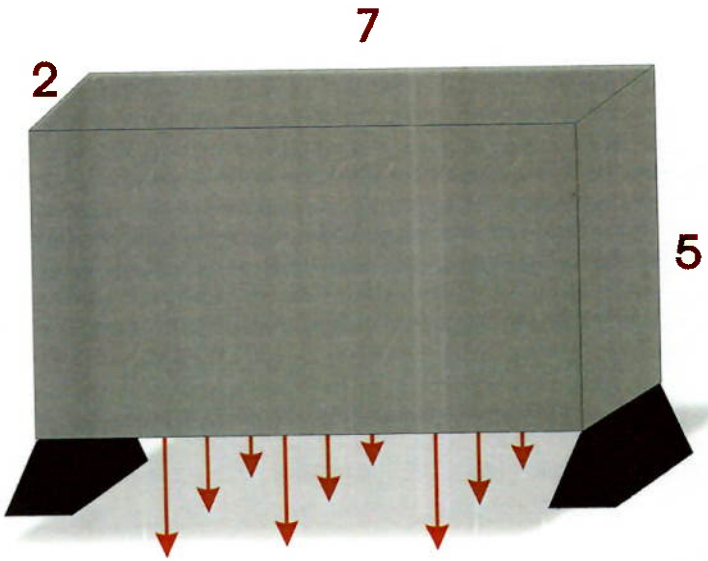


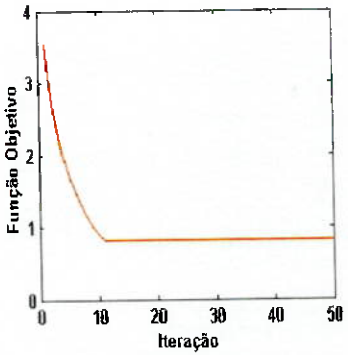


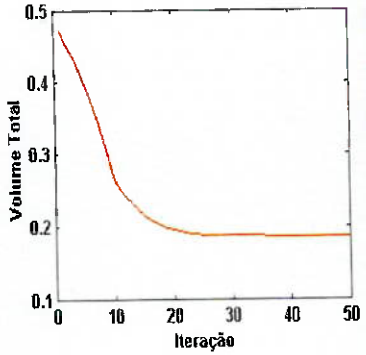


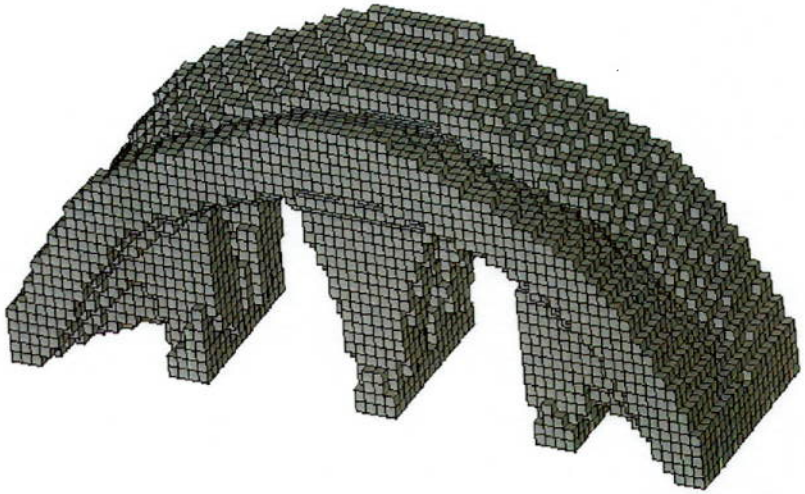
Discretização: 70.000 elementos

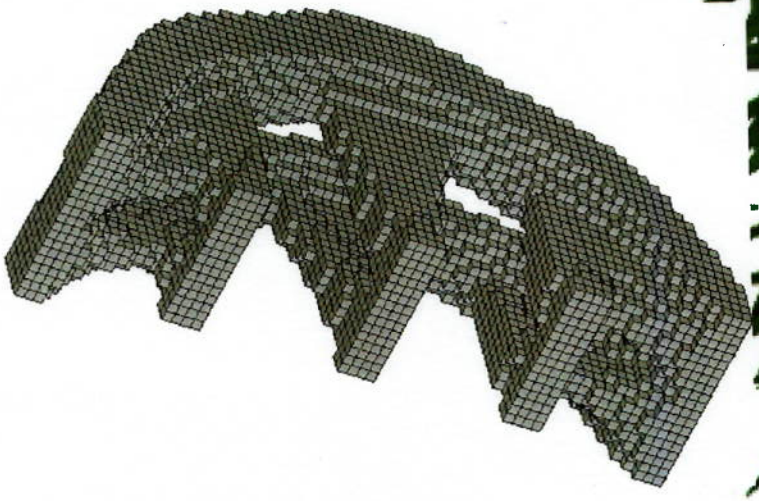
Tempo de Processamento: 42 horas





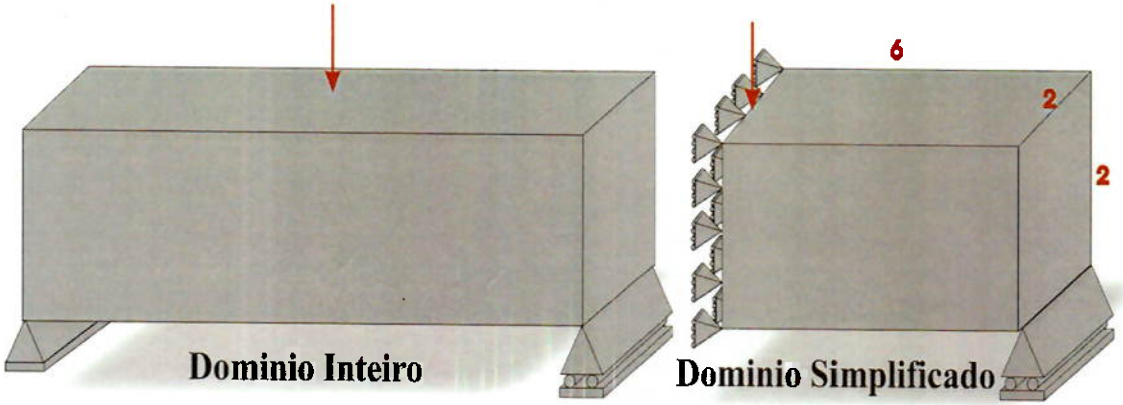


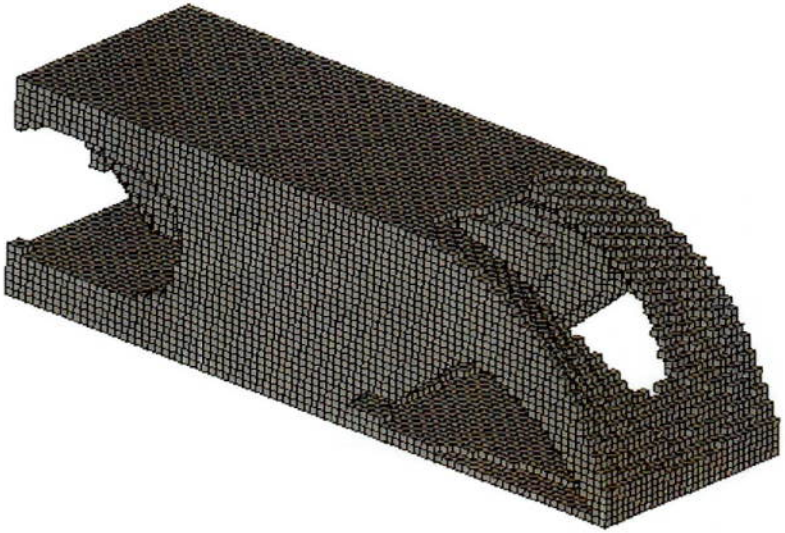


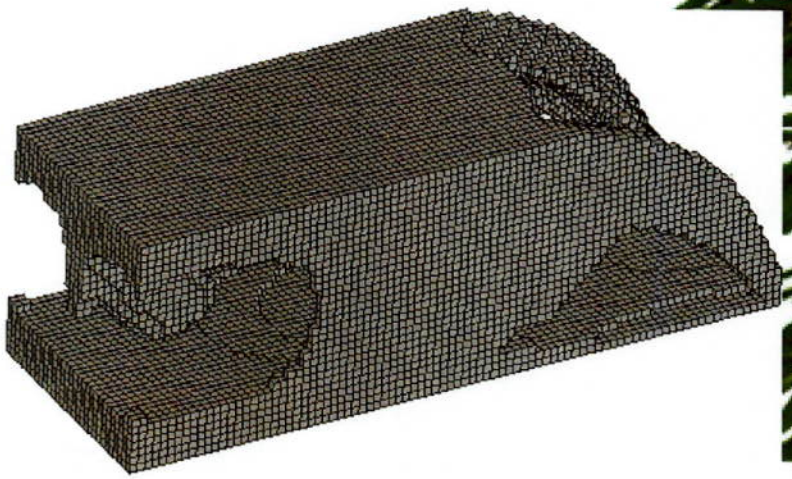


Discretização: 81.000 elementos

Tempo de Processamento: 50 horas







6.Conclusões

- Foram obtidos resultados com malhas da ordem de 100.000 elementos.
- O critério de optimalidade apresenta excelentes resultados para problemas estruturais.
- Os resultados obtidos não apresentaram o tabuleiro de xadrez.
- Foi desenvolvido um software de OT capaz de sintetizar estruturas ótimas com vários casos de carregamento e com a utilização de um filtro de densidades.
- O software desenvolvido mostrou-se eficiente, gerando topologias bem Definidas em malhas refinadas em um intervalo de tempo relativamente Curto.