UNIVERSIDADE DE SÃO PAULO ESCOLA POLITÉCNICA
DEPARTAMENTO DE ENGENHARIA MECÂNICA

Desenvolvimento de um software de Otimização Topológica bi-dimensional aplicado ao projeto de estruturas rotativas.

Fernando Viegas Stump

Orientador: Emílio Carlos Nelli Silva

São Paulo

2003

UNIVERSIDADE DE SÃO PAULO ESCOLA POLITÉCNICA
DEPARTAMENTO DE ENGENHARIA MECÂNICA

Desenvolvimento de um software de Otimização Topológica bi-dimensional aplicado ao projeto de estruturas rotativas.

Trabalho de formatura apresentado à Escola Politécnica da Universidade de São Paulo para a obtenção do título de Graduação em Engenharia

Fernando Viegas Stump

Emílio Carlos Nelli Silva

Orientador: Emílio Carlos Nelli Silva

São Paulo
2003

FICHA CATALOGRÁFICA

Stump, Fernando Viegas
Desenvolvimento de um software de Otimização Topológica bi-dimensional aplicado ao projeto de estruturas rotativas
F. V. Stump. -- São Paulo, 2003.
49p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecânica

1. Otimização Estrutural 2. Otimização Topológica
3. Forças de campo 4. Rotores

À minha família e aos meus amigos.

AGRADECIMENTOS

Agradeço a todas as pessoas que me apoiaram durante estes anos de faculdade e permitindo que chegasse até aqui. Em especial gostaria de agradecer.

A meus pais, Patrick e Marisdalva, pelo exemplo e apoio incondicional a todas minhas decisões.

A meus irmãos, Gabi e Deco e familiares, pelo apoio mesmo sem entender muito bem o que eu faço.

Ao meu orientador, Professor Emílio, por todas recomendações e atenção dedicada ao desenvolvimento não só desse trabalho mas de minha vida acadêmica.

Aos meus amigos e amigas, Mauro, German, Meps, Pedro Sansão, Thalita, Nathalia e Luciana, que com seus conselhos tanto incentivaram como desvirtuaram minha vida acadêmica. Sem estes dificilmente chegaria aqui (são).

A Ruri, esta pessoa muito especial que me apoiou desde o início da faculdade, a quem devo grande parte de minhas conquistas, inclusive esta.

SUMÁRIO

LISTA DE FIGURAS

Lista de Abreviaturas ou siglas

OT Otimização Topológica

MEF Método dos Elementos Finitos

SIMP " Simple Isotropic Material with Penaliztion"

MGBSE Método dos gradients bi-conjugados para sistemas esparsos

PL Programação Linear

PLS Programação Linear Seqüencial.

Resumo

Este trabalho apresenta o desenvolvimento de um software de otimização topológica bi-dimensional aplicado ao projeto de estruturas rotativas.O Método de Otimização Topológica consiste na combinação do Método de Elementos Finitos com métodos de otimização com a finalidade de sintetizar estruturas de geometria ótima segundo um dado critério.Neste trabalho tem se como objetivo sintetizar estruturas de máxima rigidez com uma dada quantidade de material sujeito a forças externas e forças de campo, como a gravidade e aceleração centrípeta.

É apresentada a formulação teórica do problema é discutida as características da solução do problema de máxima rigidez considerando forças inerciais. Um exemplo de caráter pratico é apresentado evidenciando a potencialidade da aplicação do método no meio industrial.

ABSTRACT.

This work presents the development of software for two-dimensional topological optimization applied to the project of rotating structures. The Topology Optimization Method consists in the combination of the Finite Elements Methods with optimization methods with the purpose of synthesize structures of optimum geometry for a given criteria. This work has as objective to synthesize structures of maximum stiffness for a given amount of material subject to external forces and body forces, as the gravity and centripetal acceleration. The theoretical formulation of the problem is explained and the characteristics of the solution of the problem of maximum stiffness considering inertial forces are presented. A practical example of synthesis is presented showing the potentiality of the method for industry projects.

1 Introdução.

Otimização pode ser definida como o ato de obter o melhor resultado sob dadas circunstâncias. Na prática de um projeto de engenharia várias decisões são tomadas com o objetivo de melhorar o desempenho do produto, ou seja, minimizar os esforços e maximizar o desempenho gerado.

O conceito de otimização é um ponto comum no dia a dia de um engenheiro projetista. A procura por soluções ótimas para a solução dos problemas enfrentados sempre será a meta do engenheiro mecânico. Apesar dessa obviedade em se procurar a solução ótima, hoje em dia o engenheiro, em geral, tem pouco conhecimento à cerca dos métodos de otimização.

Focando no projeto de estruturas mecânicas, que é o tema desse trabalho, observa-se uma grande evolução e difusão dos métodos de análise estrutural. Por exemplo, a utilização do Método de Elementos Finitos é uma realidade em grande parte das empresas de projeto, entretanto os métodos de otimização ainda são pouco difundidos e aceitos fora do meio acadêmico. Essa situação se dá por dois principais fatores: o primeiro é a falta de dialogo entre os meios a industrial e o acadêmico, o segundo é o fato de que os problemas enfrentados pelos engenheiros da indústria muitas vezes excedem a capacidade dos métodos de otimização estrutural até hoje desenvolvidos.

A década de 90 observou uma ligeira mudança nessa cultura, a difusão dos métodos de otimização pôde ser notada pelo aumento do número de publicações sobre o tema e o surgimento de disciplinas de otimização em alguns cursos de pós-graduação no país.

Entre os métodos mais comuns de otimização estrutural o Método de Otimização Topológica, tema deste trabalho, ganhou destaque justamente por se mostrar capaz de solucionar grande parte dos problemas enfrentados pela indústria.

Hoje, indústria mecânica começa a se utilizar de métodos de otimização para auxiliar o projeto de suas estruturas. Por exemplo, no mercado nacional de unidades hidrogeradoras, que consiste na turbina hidráulica mais o gerador elétrico a ela acoplado, a redução do custo dos equipamentos é uma busca constante entre os fabricantes. Devido à longa historia de

desenvolvimento desse tipo de equipamento, mais de um século de projeto, as soluções estruturais já foram exaustivamente investigadas e poucos são os avanços obtidos ao se utilizar métodos convencionais de projeto. Dessa forma o Método de Otimização Topológica tem se mostrado uma alternativa promissora para a redução de custo desses equipamentos.

Esse trabalho contribui para demonstrar a viabilidade de aplicação desse método em projetos reais.

Para exemplificar aplicação do método de utilização topológica na industria Abaixo segue um exemplo mostrando o desenvolvimento da estrutura de um ônibus urbano através de métodos do MOT.

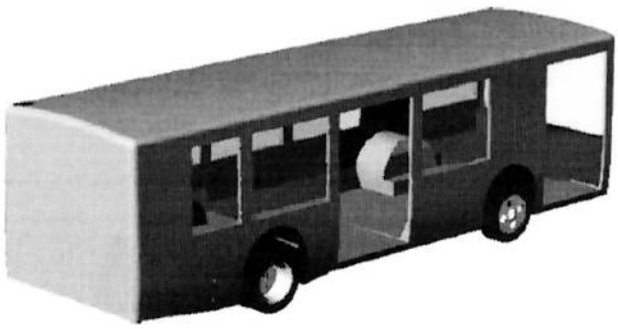

Figura 1. Domínio fixo de projeto (Região cinza) da carroceria do ônibus, região na qual o método irá distribuir material.

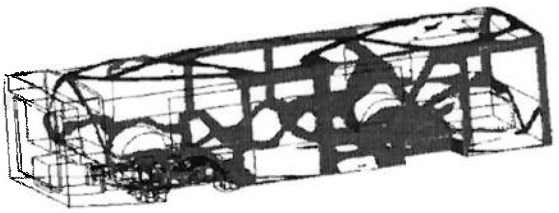

Figura 2. Resultado obtido através do MOT, distribuição ótima de material

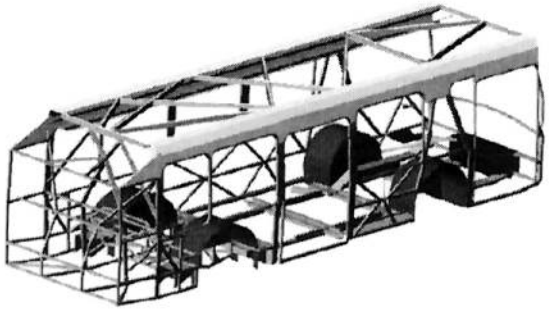

Figura 3. Interpretação do resultado obtido

Figura 4. Projeto final da estrutura.

Na industria mecânica pode se dizer que o custo da estrutura é proporcional ao seu peso de tal forma que o projeto ótimo dessas estruturas em geral será a estrutura mais leve

capaz de atender aos critérios de projeto. Em geral, dois são os critérios que norteiam o projeto de peças mecânicas, sua rigidez e sua resistência.

Este trabalho visa atender as situações de projeto onde o objetivo é maximizar a rigidez da estrutura mantendo o peso dentro de um limite aceitável, pré-determinado.

Para isso foi desenvolvido nesse trabalho, um software de Otimização Topológica, capaz de sintetizar estruturas de máxima rigidez com restrição de volume considerando tanto forças externas aplicadas à estrutura como forças inerciais devido a uma aceleração aplicada a peça.

A consideração de forças inerciais no problema de OT, curiosamente foi pouco explorada tanto em artigos científicos como em sua aplicação industrial, apesar de sua grande importância, como verá-se nesse trabalho.

O *software* foi implementado em linguagem C, e é composto basicamente de dois módulos, sendo um contendo o Método de Elementos Finitos e o algoritmo de solução do sistema de equações e o outro contendo o algoritmo de otimização, para fazer a conexão entre estes dois módulos há a análise de sensibilidade.

Nos capítulos que seguem será feita uma apresentação da teoria necessária para o entendimento do MOT e conseqüentemente do *software* desenvolvido. Ao final será apresentado um conjunto de estruturas sintetizadas pelo *software.*

2 Fundamentos teóricos

Nessa seção serão apresentadas as formulações utilizadas do Método dos Elementos Finitos e do Método de Otimização Topológica.

2.1 Método dos Elementos Finitos

Aqui será descrita a formulação do Método de Elementos Finitos, e do tipo de elemento utilizado.

Neste trabalho foi utilizada a aproximação de estado plano de tensão, esta consideração é utilizada em corpos que possuem uma geometria com a largura (direção *1*) e o comprimento

(direção *3*) em dimensões comparáveis e, no entanto, muito maiores que a espessura (direção *2*), como mostra a figura abaixo. As cargas mecânicas são aplicadas uniformemente sobre a espessura da peça.

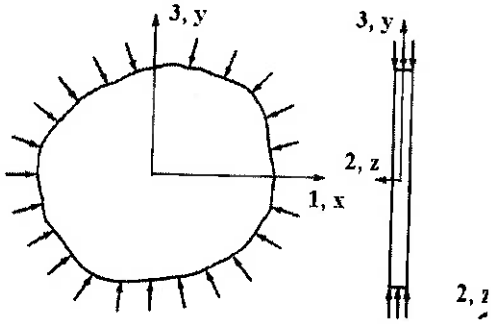

Figura 5. Exemplo de situação onde o caso de estado plano de tensão é valido.

Dessa forma a matriz da lei de Hooke para um material isotrópico torna-se:

$$\begin{Bmatrix} T_{xx} \\ T_{yy} \\ T_{xy} \end{Bmatrix} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \begin{Bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{xy} \end{Bmatrix} \qquad (1)$$

Onde E representa o módulo de Young e ν o coeficiente de Poisson

2.1.1 Formulação do elemento isoparamétrico de quatro nós

Neste trabalho foi utilizado elemento isoparamétrico bi-linear de quatro nós.

Considere a configuração bidimensional do elemento quadrilátero de quatro nós, no sistema de coordenadas locais (ξ,η) para um sistema de coordenadas globais (x, y), Figura 6.

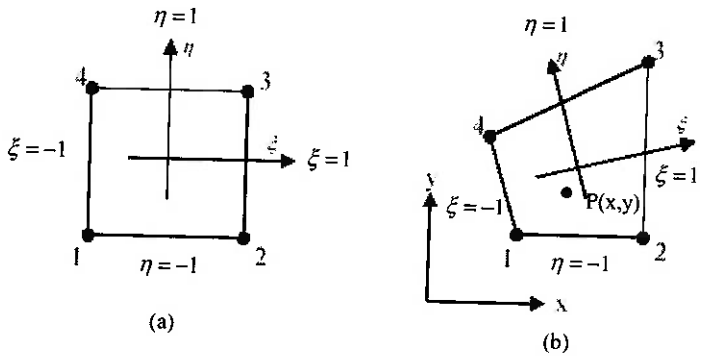

Figura 6. Mapa de coordenadas isoparamétricas. (a) elemento em coordenadas locais, (b) elemento em coordenadas globais.

No sistema de coordenadas locais, o elemento é quadrado (dimensão *2 x 2),* com o eixo de coordenadas no centro do elemento. Já no sistema de coordenadas globais, o elemento é distorcido da forma retangular. Sendo a função de forma (***N***) função do sistema de coordenadas locais, as coordenadas de qualquer ponto *P*, podem ser expressas em termos das coordenadas (*x, y*) dos nós, assim:

$$\begin{Bmatrix} x(\xi,\eta) \\ y(\xi,\eta) \end{Bmatrix} = [\mathbf{N}]\{\mathbf{x}\}_e \qquad (2)$$

onde $\{\mathbf{x}\}_e$ é o vetor de coordenadas nodais do elemento, ou seja:

$$\{\mathbf{x}\}_e = \{x_1\ y_1\ x_2\ y_2\ x_3\ y_3\ x_4\ y_4\} \qquad (3)$$

e $[\mathbf{N}]$ é a função de forma expressa na forma matricial, como:

$$[\mathbf{N}] = \begin{bmatrix} N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 & 0 \\ 0 & N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 \end{bmatrix} \qquad (4)$$

A função de forma são funções de interpolação de Lagrange e são expressas em coordenadas (ξ,η):

$$N_1(\xi,\eta)=\frac{(1-\xi)(1-\eta)}{4}$$
$$N_2(\xi,\eta)=\frac{(1+\xi)(1-\eta)}{4}$$
$$N_3(\xi,\eta)=\frac{(1+\xi)(1+\eta)}{4} \qquad (5)$$
$$N_4(\xi,\eta)=\frac{(1-\xi)(1+\eta)}{4}$$

As coordenadas de um ponto genérico (*x, y*) do elemento isoparamétrico, são obtidas em função das coordenadas nodais do elemento, usando os mesmos polinômios bi-lineares do elemento retangular, dessa forma:

$$x=N_1x_1+N_2x_2+N_3x_3+N_4x_4$$
$$y=N_1y_1+N_2y_2+N_3y_3+N_4y_4 \qquad (6)$$

Esta equação representa as aproximações para as componentes horizontal e vertical, respectivamente, do deslocamento mecânico de cada ponto no domínio do elemento.

No item seguinte veremos como determinar a matriz de rigidez do elemento.

2.1.2 Formulação da matriz de rigidez elemento

A expressão geral da matriz do elemento é escrita em termos das coordenadas globais (*x, y*). Portanto, o diferencial de comprimento *dx* e *dy* devem ser expressos em termos do diferencial de coordenadas locais $d\xi$ e $d\eta$. Além do que, a deformação é definida em termos da derivada da função de forma em coordenadas globais, e estas derivadas são os elementos da matriz [**B**] e eles devem ser transformados em derivada na respectiva coordenada local. Portanto, matriz [**B**] é uma matriz de operadores diferenciais dos polinômios bi-lineares.

$$[\mathbf{B}]=\begin{bmatrix} \frac{\partial N_1}{\partial x} & 0 & \frac{\partial N_2}{\partial y} & 0 & \frac{\partial N_3}{\partial y} & 0 & \frac{\partial N_4}{\partial y} & 0 \\ 0 & \frac{\partial N_1}{\partial y} & 0 & \frac{\partial N_2}{\partial x} & 0 & \frac{\partial N_3}{\partial x} & 0 & \frac{\partial N_4}{\partial x} \\ \frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial y} & \frac{\partial N_2}{\partial x} & \frac{\partial N_3}{\partial y} & \frac{\partial N_3}{\partial x} & \frac{\partial N_4}{\partial y} & \frac{\partial N_4}{\partial x} \end{bmatrix} \quad (7)$$

Os diferenciais (*dx, dy*) são relacionados com os diferenciais $d\xi$ e $d\eta$ por meio da equação abaixo. Assim:

$$dx=\frac{\partial x}{\partial \xi}d\xi+\frac{\partial x}{\partial \eta}d\eta$$
$$dy=\frac{\partial y}{\partial \xi}d\xi+\frac{\partial y}{\partial \eta}d\eta \quad (8)$$

onde:

$$\frac{\partial x}{\partial \xi}=\sum\frac{\partial N_i}{\partial \xi}x_i \;;\quad \frac{\partial y}{\partial \xi}=\sum\frac{\partial N_i}{\partial \xi}y_i$$
$$\frac{\partial x}{\partial \eta}=\sum\frac{\partial N_i}{\partial \eta}x_i \;;\quad \frac{\partial y}{\partial \eta}=\sum\frac{\partial N_i}{\partial \eta}y_i \quad (9)$$

As derivadas das coordenadas são combinadas na forma matricial como:

$$[\mathbf{j}]=\begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \quad (10)$$

onde [j] é a matriz Jacobiana de transformação.

Relacionando estas equações, os diferenciais de dois sistemas de coordenadas podem ser expressos por:

$$\begin{Bmatrix} dx \\ dy \end{Bmatrix} = [\mathbf{j}]^t \begin{Bmatrix} d\xi \\ d\eta \end{Bmatrix}, \tag{11}$$

De maneira semelhante, as derivadas da função de forma dos nós "*i*" são relacionadas por:

$$\begin{Bmatrix} \frac{\partial \mathbf{N}_i}{\partial x} \\ \frac{\partial \mathbf{N}_i}{\partial y} \end{Bmatrix} = [\mathbf{j}]^{-1} \begin{Bmatrix} \frac{\partial \mathbf{N}_i}{\partial \xi} \\ \frac{\partial \mathbf{N}_i}{\partial \eta} \end{Bmatrix} \tag{12}$$

A partir desses resultados é possível obter a matriz de rigidez de um elemento.

$$[\mathbf{K}_{uu}]_e = \int_{V_e} [\mathbf{B}_u]^t [\mathbf{c}^E] [\mathbf{B}_u] dV \tag{13}$$

Usando o Jacobiano como um operador para transformar as coordenadas locais (ξ,η) em coordenadas globais (x, y), A equação acima pode ser escrita da seguinte forma:

$$[\mathbf{K}]_e = \int_{-1}^{1}\int_{-1}^{1} [\mathbf{B}]^t [\mathbf{C}] [\mathbf{B}] \det(\mathbf{J}) d\eta d\xi \tag{14}$$

A matriz de rigidez local é inserida numa matriz global $[\mathbf{K}]$, através da conectividade de cada elemento, que associa um número a cada elemento e o número de nós a que está conectado.

Para a solução das equações de equilíbrio dada pelo sistema:

$$\mathbf{Ku} = \mathbf{F} \tag{15}$$

É necessária a determinação do termo à direita $\mathbf{F}$ da equação acima que representa as cargas aplicadas ao modelo.

O vetor de força $\mathbf{F}$ é a somatória dos esforços nodais, o seja as cargas aplicadas diretamente sobre o nó e as forças de campo dada por

$$\mathbf{F_c} = \int_{V_e} \mathbf{N}^t \mathbf{f_c} dV \tag{16}$$

Onde $\mathbf{N}$ representa a matriz de interpolação de deslocamentos e é dada pela equação 5.

No caso desse trabalho, o vetor $\mathbf{f_c}$ é dado por:

$$\mathbf{f_c} = \begin{bmatrix} \rho_{real}\left[g_x + \omega^2(X_p - X_0)\right] \\ \rho_{real}\left[g_y + \omega^2(Y_p - Y_0)\right] \end{bmatrix} \tag{17}$$

Onde ρ_{real} representa a densidade do material, aqui o sobrescrito real é utilizado para diferenciar da variável de projeto ρ_e que será apresentada a diante.

Os valores g_x e g_y representam a decomposição da aceleração da gravidade nos componentes x e y do sistema de coordenadas, ω é o valor da velocidade angular perpendicular ao plano Oxy, X_0, Y_0 é o centro de rotação do sistema e X_p, Y_p é o ponto onde se está calculando o vetor $\mathbf{f_c}$.

A integral 17 é avaliada em coordenadas locais, utilizando o operador Jacobinano temos:

$$\mathbf{f_b} = \int_{-1}^{1}\int_{-1}^{1} \mathbf{N}^t \mathbf{f_c} |\mathbf{J}| d\eta d\xi \tag{18}$$

Avaliação das integrais da equação 19 e 15 é feita numericamente utilizando a Quadratura Gaussiana, (Método de Gauss-Legendre). É provado que o valor somatória obtida por este método representa a integral exata para um polinômio de

grau igual a, no máximo três, atendendo as necessidades desse trabalho.Uma explicação mais detalhada do método pode ser obtida na referência [3].

Neste momento é possível resolver o sistema $\mathbf{Ku} = \mathbf{F}$ e dessa forma se possui o valor dos deslocamentos nodais da estrutura. A método de solução desse sistema está exposto no 2.4 Implementação Numérica

2.2 Otimização Topológica

A Otimização Topológica (OT) oferece um método sistemático e eficiente para o projeto otimizado de peças mecânicas, que consiste em redistribuir material em um dado volume, de maneira iterativa, a fim de obter a topologia que maximize uma ou mais funções objetivo da estrutura, atendendo a restrições impostas. Esse método combina o método de elementos finitos (MEF) com métodos de otimização.

Com método otimização Topológica, a definição da forma básica de uma estrutura deixa de ser um processo de tentativa e erro (através de sucessivas análises para procura de uma estrutura com melhor desempenho) para se tornar um processo de síntese da estrutura, isto é, ao se aplicar o método de otimização Topológica (MOT) o engenheiro obtém uma topologia inicial ótima da estrutura. Dessa maneira, o MOT não é apenas uma ferramenta que permite um aprimoramento do produto mas, também, que reduz os custo de projeto da peça uma vez que sua aplicação é sistemática.

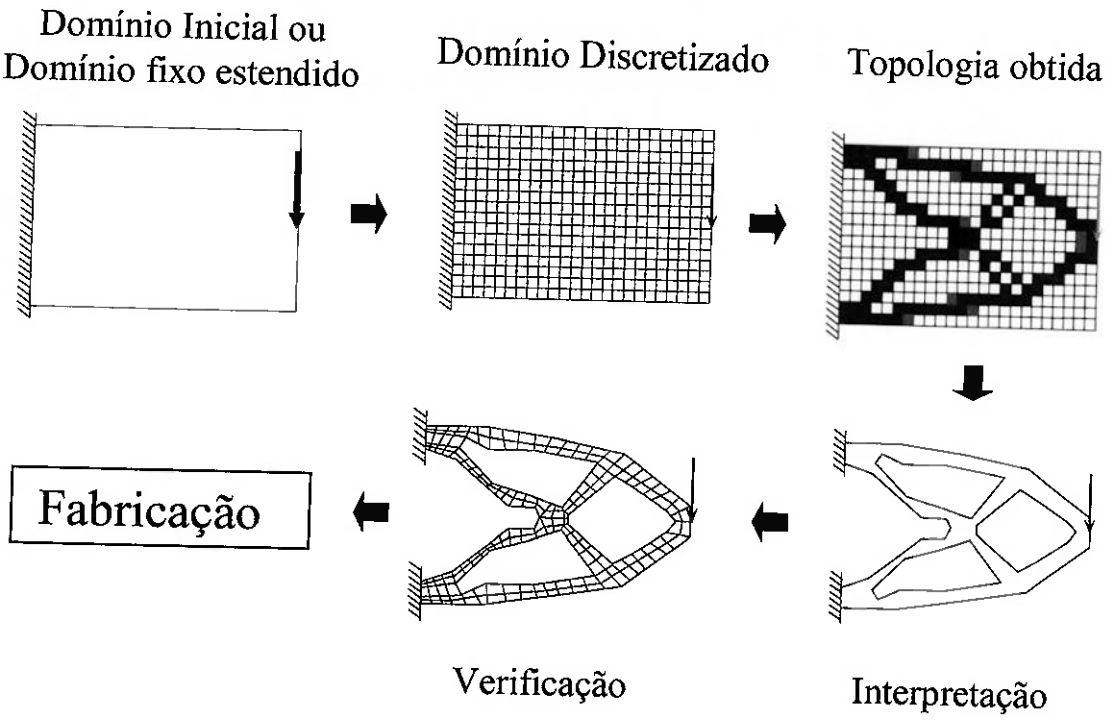

Figura 7. Processo de projeto de estruturas utilizando o Método de Otimização Topológica.

O problema de sintetizar a estrutura de uma viga engastada sujeita a um carregamento na extremidade com critério mínimo peso e máxima rigidez, está acima apresentado.

O primeiro passo para a solução desse problema consiste em definir o domínio no qual a estrutura pode existir. Esse domínio é limitado pelas condições de contorno da estrutura (pontos em que ela deve estar restrita) e pelos pontos de aplicação de carga.

No segundo passo o domínio é discretizado em elementos finitos e são aplicadas as condições de contorno. No terceiro passo, os dados do domínio são fornecidos ao *software* de Otimização Topológica que num processo iterativo distribui o material no domínio de forma a minimizar (ou maximizar) a função objetivo especificada, no caso, minimizar a flexibilidade. O resultado obtido é do tipo mostrado no item "topologia obtida" onde a cor escura indica a presença de material e a cor branca indica a ausência de material no ponto do domínio. Note que podem surgir pontos com cores intermediárias, denominados de escalas de cinza. Esses pontos indicam a presença de materiais intermediários que não podem ser implementados na prática e sempre ocorrem. Dessa forma, a imagem da estrutura obtida por OT representa um excelente ponto de partida que necessita ser

interpretado para se obter o projeto final da estrutura. Essa interpretação (quarta etapa) pode ser feita usando-se métodos de processamento de imagem ou simplesmente desenhando-se uma estrutura baseada na imagem obtida por OT sempre levando em conta as restrições impostas pelo método de manufatura.

A penúltima etapa consiste em se verificar o resultado final da estrutura. Em geral os resultados gerados por OT não são intuitivos e é interessante fazer uma verificação da estrutura final usando MEF, para criarmos confiança na solução através da comprovação da optimalidade do resultado.

Finalmente, a última etapa é a fabricação da estrutura. Hoje, existem várias técnicas de fabricação que permitem fabricar estruturas com formas complexas como prototipagem rápida, entre outras.

Esse trabalho se restringirá a discutir a otimização Topológica para o caso onde o objetivo é obter a estrutura de menor peso e máxima rigidez.

2.2.1 Conceitos teóricos do Método de Otimização Topológica.

Para a utilização de qualquer método de otimização é necessário inicialmente definir a formulação matemática do problema.

Nesta seção será apresentada a formulação do problema e os dois principais conceitos do MOT, o domínio fixo estendido (DFE) e modelo de material.

2.2.2 Formulação do Problema

Conforme dito anteriormente o objetivo desse trabalho é síntese de estruturas atendendo o critério de máxima rigidez para um dado volume de material considerando forças inerciais na formulação do problema.

Para atender a esse objetivo é necessário se formular o problema de tal forma a se quantificar matematicamente a função objetivo a ser extremaizada e as restrições do problema.

A formulação utilizada nesse trabalho é uma das mais tradicionais na síntese de componentes mecânicos e pode ser dada por:

Minimizar $C = \mathbf{F}^T\mathbf{U}$

(19)

Tal que

$$f \geq \frac{V(\boldsymbol{\rho})}{V_o}$$

(20)

$$0 < \rho_{e\min} \leq \rho_e \leq \rho_{e\max}$$

(21)

$\mathbf{K} \cdot \mathbf{U} = \mathbf{F}$ onde $\mathbf{K} = \mathbf{f}(\boldsymbol{\rho})$ e $\mathbf{F} = \mathbf{f}(\boldsymbol{\rho})$

Nessa formulação é possível identificar a função objetivo $C = \mathbf{F}^T\mathbf{U}$ que quantifica a flexibilidade da estrutura, A função objetivo que depende das variáveis de projeto. No caso exposto as variáveis de projeto, definidas como ρ_e, representam a quantidade de material em cada elemento, os valores de ρ_e compõem a matriz de rigidez ($\mathbf{K}$) do modelo de elementos finitos, que por sua vez permite obter os deslocamentos da estrutura através da equação de equilíbrio estático $\mathbf{K} \cdot \mathbf{U} = \mathbf{F}$. Assim temos a flexibilidade dada também por $C = \mathbf{U}^T\mathbf{K}\mathbf{U}$,

A forma que a quantidade de material é representada pelas variáveis de projeto (ρ_e) é definida pelo modelo de material que será utilizado, entretanto é importante notar que independentemente do modelo de material a variável de projeto representa a quantidade de material presente em cada elemento, é comum denominar essa quantidade com sendo a pseudo-densidade do elemento, que no caso pode variar de zero, (ausência de material), a um (presença de material).

A restrição imposta ao problema é definida por:

$$f \geq \frac{V(\mathbf{X})}{V_o},$$

onde $V(\mathbf{X}) = \sum_{\Omega} \rho_e v_e$

onde f é a fração de material que se deseja que permaneça no domínio fixo estendido de volume V_o.

2.2.3 Domínio Fixo Estendido (DFE)

O domínio estendido fixo (DFE) de projeto consiste num domínio de forma fixa limitado pelos pontos de apoio da estrutura e pontos de aplicação de carregamento, e que conterá a estrutura desconhecida. O objetivo da otimização Topológica é determinar os espaços sem material ("buracos") e a conectividade da estrutura através da remoção e adição de material nesse domínio. O problema de otimização consiste em se encontrar a distribuição ótima de propriedades de materiais no domínio fixo estendido (DFE).

Na implementação numérica, o DFE é discretizado em elementos finitos. Portanto, o modelo de elementos finitos do domínio não é alterado durante o processo de otimização, sendo alterada somente a distribuição de material nos elementos, Assim, as variáveis de projeto estão associadas a cada elemento da malha de MEF. Como a malha não varia durante o processo de otimização, o cálculo da sensibilidade da função objetivo se torna um processo mais simples, pois não é influenciado pela variação de forma do domínio.

2.2.4 Modelo de Material

O problema de otimização Topológica é em sua essência um problema discreto binário, considerando como variável de projeto a quantidade de material dos elementos, os únicos valores admissíveis em uma solução seriam zero (ausência de material), ou um, (presença de material), entretanto a formulações criadas utilizando variáveis discretas não foram bem sucedidas. Em geral o custo computacional se torna proibitivo para a solução de problemas com grande número de variáveis. Alem do que a formulação discreta insere ao problema múltiplos mínimos globais Para contornar esse problema foram criados modelos de material que se baseiam em variáveis contínuas

O modelo de material é uma equação que define a mistura em microescala de dois ou mais materiais (um deles pode ser ar) permitindo que haja estágios intermediários ao se passar da condição de zero material ("buraco") a sólido em cada ponto do

domínio. Isso garante a relaxação do problema de otimização evitando que se passe de ar para sólido de forma brusca, o que torna difícil o tratamento numérico do problema. É portanto um ponto chave na formulação da OT. A princípio, os estágios intermediários de materiais não têm significado físico sendo apenas decorrentes de um recurso matemático para relaxação do problema. A definição do modelo de material define o grau de relaxação do problema.

Existem vários modelos de materiais descritos na literatura que podem ser utilizados. O mais comum é o chamado Método das Densidades ou SIMP (Solid Isotropic Material with Penalization), que consiste numa equação matemática que define o valor da propriedade E da equação 2 em cada ponto do domínio em função da pseudo-densidade ρ_e do material usado no projeto (que varia de zero a um valor máximo) e a propriedade básica (E_0) do material a ser distribuído, como descrito na equação abaixo.

$$E_e = \rho_e^p . E_0 \tag{22}$$

Dessa forma tem-se uma variável em cada ponto do domínio, representada pelo valor da densidade naquele ponto. No caso do problema discretizado, define-se uma variável por elemento finito.

O valor p que aparece na equação 23 insere ao problema uma penalização da rigidez para as pseudo-densidades intermediárias, ou seja valores de ρ_e entre 0 e 1.

O valor atribuído à penalização p é amplamente discutido na literatura e uma discussão mais detalhada pode ser encontrada nas referências [1],[2]. Nesse trabalho foi utilizado o método da continuação que consiste em se alterar o valor de p ao longo das iterações de modo a auxiliar a obtenção de um bom resultado.Conforme se discutirá adiante.

2.2.5 Qualidade da Solução

Um fator importante para a utilização do MOT é a qualidade do resultado obtido. Em um caso prático se está interessado em obter um estrutura relativamente simples e principalmente bem definida. Entretanto ao se resolver problemas de OT é

relativamente freqüente se deparar com problemas numéricos que prejudicam a qualidade da solução. Estes problemas dependem de vários fatores entre eles: o tipo e ordem do elemento utilizado, densidade da malha de elementos finitos, algoritmo de otimização e o modelo de material.

Os problemas encontrados mais relevantes são instabilidade de tabuleiro e escalas de cinza.

Instabilidade de tabuleiro

O problema de instabilidade de tabuleiro é conhecido pela formação de regiões na estrutura final onde elementos vizinhos possuem densidades zero ou um, de maneira intercalada definindo um padrão semelhante a um tabuleiro de damas, Figura 8. Abaixo está um exemplo de uma estrutura que apresenta instabilidade de tabuleiro, também conhecido na literatura como tabuleiro de damas ou xadrez.

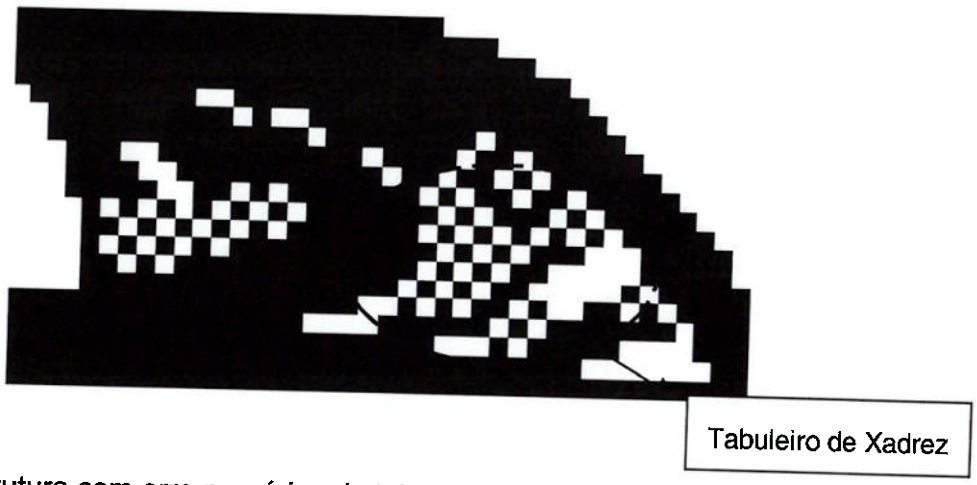

Figura 8. Estrutura com erro numérico de tabuleiro de xadrez

Acreditava-se que estas regiões representavam uma distribuição de material ótima, mas provou-se que este problema aparece devido a uma modelagem fraca [2], inerente à utilização de elementos de baixa ordem (quatro nós, como é o caso desse trabalho), na malha de elementos finitos para a modelagem do domínio. Esses elementos causam uma rigidez artificial maior que aquela dada por uma distribuição uniforme de material.

A precaução mais simples para esse problema é a modelagem do domínio com elementos de alta ordem (elementos de 8 ou 9 nós), entretanto essa solução aumenta o número de

graus de liberdade da malha de elementos finitos acarretando um aumento de tempo computacional.

Outros métodos para prevenir a formação de tabuleiro de damas baseiam-se na aplicação de filtros que são utilizados para o tratamento de imagens digitais. Basicamente os filtros tendem a amenizar a distribuição de densidade [4]. Neste trabalho foi utilizado o Método de Controle dos Gradientes (MCG).

O MCG pode ser aplicado na forma de filtro ou na forma de restrição ao problema de otimização, e consiste em se restringir à variação espacial da variável de projeto de tal forma que se evite o tabuleiro de damas. Neste trabalho será aplicado o filtro sobre os limites móveis

Escalas de Cinza

Ao se utilizar à formulação com variáveis contínuas, a solução pode apresentar regiões com elementos de pseudo-densidades intermediárias, chamadas de regiões com escala de cinza. A formação desse tipo de região esta intimamente ligada com o modelo de material utilizado. No modelo de material chamado de método das densidades o controle de escala de cinza é feita pelo valor da penalização p Valores de p maiores que 1 penalizam os elementos com densidades intermediárias

O controle de escala de cinzas é uma ferramenta importante quando se trata de um problema prático de engenharia uma vez que elementos que não estão bem definidos deverão ser mantidos ou retirados da estrutura conforme a interpretação do engenheiro projetista, pois existirão duvidas quanto a quais elementos se devem retirar ou manter e como isso afetará a rigidez global da estrutura.

A penalização p apesar de ser útil na redução da escala de cinza retira a convexidade do problema original. Ou seja. Na formulação com p igual a um é provado que o problema de otimização formulado nesse trabalho possui um único mínimo global. Dessa forma se garante matematicamente que a solução ótima será obtida. Quando se insere o valor p diferente de 1 o problema deixa de ser convexo e a estrutura final pode ser um mínimo local da função objetivo que não apresenta as a características desejadas.

Para amenizar este problema foi utilizado o Método da Continuação cujo o objetivo é iniciar o problema mais relaxado, ou seja com p igual a um e somente após algumas iterações

elevar o valor para p igual a dois e finalmente 3, que é um valor usualmente utilizado na literatura [1]. Dessa forma a solução tende a caminhar para a proximidade do mínimo global para então penalizar as densidades intermediárias e assim obter uma resposta de boa qualidade, sem escala de cinza.

Neste ponto é importante notar que o filtro e a penalização p competem entre si na definição da geometria ótima. O filtro tende a criar escala de cinza, pois não permite a variação espacial brusca da pseudo-densidade e, por outro lado, a penalização tende a levar todos os elementos a zero ou um, inserindo necessariamente uma variação espacial brusca.

Para se obter resultados de boa qualidade foi utilizada a seguinte estratégia. O *software* faz dez iterações com p igual a 1, seguido de mais dez com p igual a 2, então se ligar o filtro e elevar o valor de p para 3, são feitas n iterações necessárias para se atender ao critério de convergência, então o filtro é desligado e ocorre no mínimo mais cinco iterações para então se verificar o critério de convergência. Quando este é atingido considera-se o fim da otimização.

2.3 Método de otimização

Um ponto de extrema importância na solução numérica de um problema de otimização é o algoritmo de otimização que esta sendo utilizado.

Existem vários algoritmos de otimização disponíveis para solução de problemas de otimização não-lineares com restrições. A maior parte deles é baseado nas chamadas técnicas de programação matemática e métodos de aproximação seqüencial. Uma revisão e descrição detalhada desses algoritmos pode ser encontrada em [6].

Entre os algoritmos baseados em métodos de aproximação seqüencial, a programação linear seqüencial (PLS, ou "SLP",em inglês) tem sido aplicada com sucesso na solução de problemas de otimização topológica. Neste algoritmo, o problema de otimização não-linear é aproximado por uma seqüência de subproblemas de otimização lineares obtidos através da aproximação local da função objetivo e restrições na solução corrente da iteração, usando termos de primeira ordem da expansão em série de Taylor. Em cada iteração, o subproblema linear é resolvido usando um método de programação linear

(Simplex ou Kamarkar)[6],[7], para encontrar o ótimo da aproximação linear. O resultado do último subproblema de aproximação é atualizado como ponto inicial para o próximo subproblema de aproximação e este procedimento é repetido ao longo das iterações. Em cada iteração, são definidos limites móveis para as variáveis de projeto. Em geral, durante uma iteração, as variáveis de projeto são permitidas variar de 5 a 15% de seus valores iniciais na iteração. O método prossegue até que o ótimo final para o problema não-linear é atingido. Dessa forma, o "PLS" consiste na solução seqüencial de subproblemas lineares de otimização. Sendo recomendado quando há um grande número de variáveis de projeto e várias restrições são consideradas. Dessa forma, a princípio, o "PLS" se aplica ao problema em questão com a grande vantagem de permitir incluir à vontade restrições no problema de otimização.

Para se utilizar a Programação Linear Seqüencial é necessário, como já foi dito, linearizar a função objetivo, usando termos de primeira ordem da expansão em série de Taylor. Como se observa abaixo podemos representar a função objetivo pela seguinte função de primeiro grau:

$$C = C_0 + \frac{\partial C}{\partial \rho_1}_{\rho=\rho^0}(\rho_1 - \rho_1^0) + \frac{\partial C}{\partial \rho_2}_{\rho=\rho^0}(\rho_2 - \rho_2^0) + \ldots + \frac{\partial C}{\partial \rho_N}_{\rho=\rho^0}(\rho_N - \rho_N^0) \qquad (23)$$

Separando os termos constantes dos termos dependentes das variáveis, temos:

$$C = \underbrace{C_0 - \frac{\partial C}{\partial \rho_1}_{\rho=\rho^0}\rho_1^0 - \frac{\partial C}{\partial \rho_2}_{\rho=\rho^0}\rho_2^0 - \ldots - \frac{\partial C}{\partial \rho_N}_{\rho=\rho^0}\rho_N^0}_{constantes} + \underbrace{\frac{\partial C}{\partial \rho_1}_{\rho=\rho^0}\rho_1 + \frac{\partial C}{\partial \rho_2}_{\rho=\rho^0}\rho_2 + \ldots + \frac{\partial C}{\partial \rho_N}_{\rho=\rho^0}\rho_N}_{variáveis} \qquad (24)$$

As constantes podem ser retiradas da equação, pois não influenciam o processo de minimização da função objetivo. A equação a ser minimizada, portanto, assume uma forma linear cujas variáveis são as pseudo-densidades ρ_e. O coeficiente da variável ρ_e é a derivada da flexibilidade em relação à própria variável ρ_e, cujo cálculo segue abaixo

Sendo a função objetivo dada por:

$$C(\boldsymbol{\rho}) = \mathbf{f}^{\mathrm{T}} \cdot \mathbf{u} \tag{25}$$

Aplicando a regra da cadeia:

$$\frac{\partial C(\boldsymbol{\rho})}{\partial \rho_e} = \frac{\partial \mathbf{f}^{\mathrm{t}}}{\partial \rho_e} \cdot \mathbf{u} + \mathbf{f}^{\mathrm{T}} \cdot \frac{\partial \mathbf{u}}{\partial \rho_e} \tag{26}$$

Necessitamos obter $\frac{\partial \mathbf{u}}{\partial \rho_e}$

Sendo

$$\mathbf{K} \cdot \mathbf{u} = \mathbf{f}$$

Derivando dos dois lados

$$\frac{\partial (\mathbf{K} \cdot \mathbf{u})}{\partial \rho_e} = \frac{\partial \mathbf{f}}{\partial \rho_e}$$

Aplicando a regra da cadeia

$$\frac{\partial \mathbf{K}}{\partial \rho_e} \cdot u + \mathbf{K} \cdot \frac{\partial \mathbf{u}}{\partial \rho_e} = \frac{\partial \mathbf{f}}{\partial \rho_e} \Rightarrow \frac{\partial \mathbf{u}}{\partial \rho_e} = \mathbf{K}^{-1} \cdot \frac{\partial \mathbf{f}}{\partial \rho_e} - \mathbf{K}^{-1} \cdot \frac{\partial \mathbf{K}}{\partial \rho_e} \cdot \mathbf{u} \tag{27}$$

Substituindo em $\mathbf{f}^{\mathrm{T}} \cdot \frac{\partial \mathbf{u}}{\partial \rho_e}$

$$\mathbf{f}^{\mathrm{T}} \cdot \mathbf{K}^{-1} \cdot \frac{\partial \mathbf{f}}{\partial \rho_e} - \mathbf{f}^{\mathrm{T}} \cdot \mathbf{K}^{-1} \cdot \frac{\partial \mathbf{K}}{\partial \rho_e} \cdot \mathbf{u} \tag{28}$$

no caso desse trabalho onde $\mathbf{K}$ é simétrica

$$\mathbf{u}^{\mathrm{T}} = \mathbf{f}^{\mathrm{T}} \cdot \mathbf{K}^{-1} \tag{29}$$

Substituindo (30) (31) em (27)

$$\frac{\partial C(\boldsymbol{\rho})}{\partial \rho_e} = \frac{\partial \mathbf{f}^{\mathrm{t}}}{\partial \rho_e} \cdot \mathbf{u} + \mathbf{u}^{\mathrm{T}} \cdot \frac{\partial \mathbf{f}}{\partial \rho_e} - \mathbf{u}^{\mathrm{T}} \cdot \frac{\partial \mathbf{K}}{\partial \rho_e} \cdot \mathbf{u} \tag{30}$$

Sendo $\frac{\partial \mathbf{f}^{\mathrm{t}}}{\partial \rho_e} \cdot \mathbf{u} = \mathbf{u}^{\mathrm{T}} \cdot \frac{\partial \mathbf{f}}{\partial \rho_e}$

Temos

$$\frac{\partial C(\boldsymbol{\rho})}{\partial \rho_e} = 2 \cdot \mathbf{u}^{\mathrm{T}} \cdot \frac{\partial \mathbf{f}}{\partial \rho_e} - \mathbf{u}^{\mathrm{T}} \cdot \frac{\partial \mathbf{K}}{\partial \rho_e} \cdot \mathbf{u} \tag{31}$$

Dessa forma o problema de programação linear toma a forma:

Minimizar $C = \frac{\partial C}{\partial \rho_N}_{\rho=\rho^0} \rho_N^0 + \frac{\partial C}{\partial \rho_1}_{\rho=\rho^0} \rho_1 + \frac{\partial C}{\partial \rho_2}_{\rho=\rho^0} \rho_2 + \ldots + \frac{\partial C}{\partial \rho_N}_{\rho=\rho^0} \rho_N$ (32)

Tal que

$$\sum_{e=1}^{M} v_e \rho_e \leq f V_0 \tag{33}$$

$$\rho_{e\min} \leq \rho_e \leq \rho_{e\max} \tag{34}$$

$\mathbf{K} \cdot \mathbf{U} = \mathbf{F}$ onde $\mathbf{K} = \mathbf{f}(\boldsymbol{\rho})$ e $\mathbf{F} = \mathbf{f}(\boldsymbol{\rho})$

Os valores de ρ_{emin} e ρ_{emax} estão limitados entre 5% a 15% abaixo e acima, respectivamente, do valor de ρ_e, e são variáveis conforme um algoritmo que verifica se está ocorrendo oscilações da variável ρ_e, caso sim, os limites móveis tem seus valores percentuais diminuídos. Conforme será descrito no item implementação numérica

2.4 Implementação Numérica

Na implementação numérica a OT combina o método de elementos finitos (MEF) com algoritmos de otimização. Os principais passos de um programa de OT, estão representados no fluxograma da Figura 9.

Além do modelo de material é necessária a implementação conjunta de um módulo de elementos finitos, que deve ser capaz de calcular o valor da função objetivo, e um algoritmo de otimização que irá atualizar as variáveis de projeto de modo a minimizar a função objetivo.

O algoritmo de otimização necessita como entrada o valor do gradiente da função objetivo e restrições em relação as variáveis de projeto, Esse cálculo é feito durante a análise de sensibilidade.

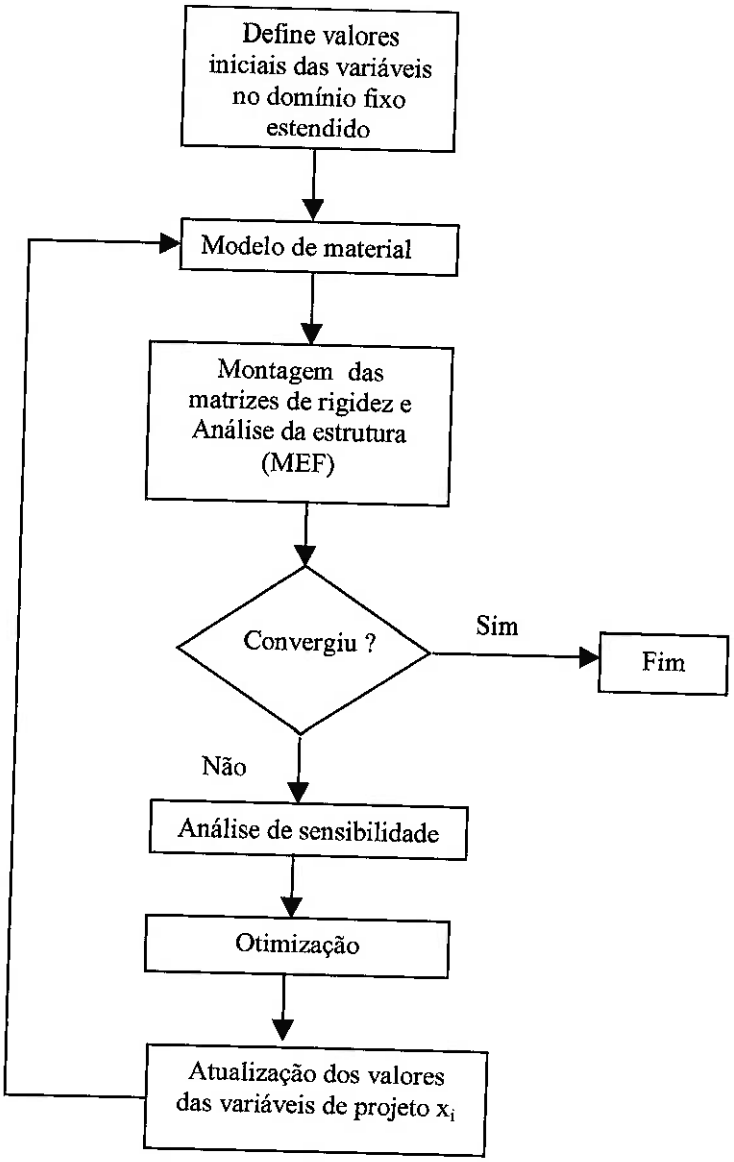

Figura 9. Fluxograma de implementação do MOT.

Abaixo está apresentado o processo de convergência de uma estrutura bidimensional projetada pelo método de otimização topológica.

O problema apresentado consiste em sintetizar uma viga bi-apoiada com carregamento a meio vão. Com o intuito de reduzir o número de elementos e tempo computacional foi sintetizada apenas metade da viga utilizando-se o conceito de simetria.

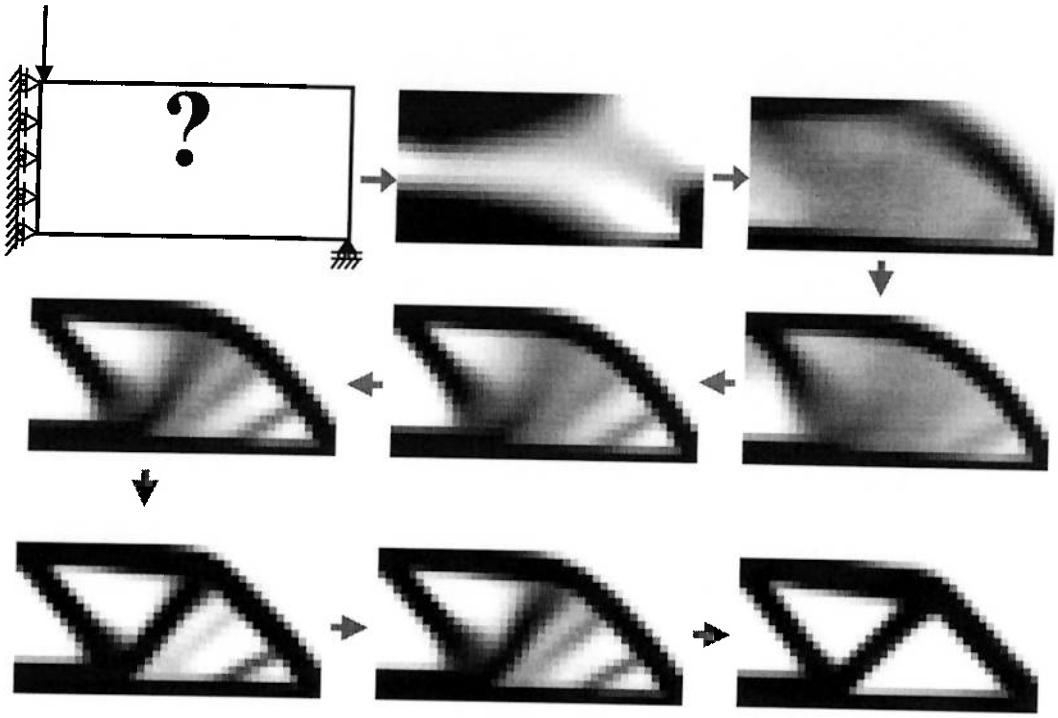

Figura 10. Convergência da solução no método de otimização topológica

Para se criar os dados de entrada do *software* desenvolvido, foi utilizado o *software* comercial Ansys 6.1, onde é gerada a geometria e aplicado os carregamentos, tanto externo (forças aplicadas aos nós) quanto os esforços de inércia (rotação e aceleração). A partir do modelo em Ansys é gerado um arquivo em formato ASCII, através do comando Archive Model -> Write.

Este arquivo é então lido pelo *software* desenvolvido, ele contém: as coordenadas de todos os nós, a conectividade dos elementos, propriedades mecânicas do material, o vetor de força externa aplicada, a indicação dos graus de liberdade restritos, o número de elementos (*M*), o número de nós (*N*).

A conectividade do elemento define os nós que compõem cada elemento, no caso de elementos de quatro nós são os vértices do quadrilátero.

A partir desses valores o *software* cria as matrizes fixas *ID*, matriz dos graus de liberdade não restritos já enumerados, a matriz *edof* formada com a conectividade dos elementos, a *fext* com as forças externas nodais aplicadas a estrutura, a *coord* matriz de coordenadas nodais. Obtém se o valor de *ksize*,número de graus de liberdade livres do modelo.

Com a matriz *ID* e *edof*, desenvolve-se a matriz *LM*. Apenas *fext,LM, ksize*, *coord* e *conect* serão utilizadas diretamente no algoritmo do MEF.

É necessário então a criação da matriz de rigidez local dos elementos, para isso utiliza-se os valor de *conect* e *coord*. Cada elemento possui uma matriz 8 x 8 que relaciona as forças aplicadas aos nós com os respectivos deslocamentos. Calculada a matriz local deve-se então inserir seus valores na matriz global *K* que possui tamanho *ksize* x *ksize*.

Para completar o sistema $\mathbf{K}\cdot\mathbf{u}=\mathbf{f}$ é necessário calcular o vetor *fb*, forças de campo aplicadas a estrutura. Tal qual a matriz de rigidez local o vetor *fb* é calculado utilizando a geometria do elemento e os valores de densidade e aceleração do elemento.

Com a montagem da matriz de rigidez global *K* e o vetor de forças composto por *fb* mais *fext*. Deve-se então resolver o sistema linear $\mathbf{K}\cdot\mathbf{u}=\mathbf{f}$, para isso utilizou-se a rotina *linbcg* disponível na referência [9]. Esta rotina resolve sistemas lineares utilizando o método MGBSE, Método dos Gradientes Bi-Conjugados para Sistemas Esparsos, para utilizar essa rotina é necessário transformar a matriz *K*, que foi gerada em sua forma completa para uma representação esparsa. Para isso utiliza-se a rotina, *dsprsin* também disponível na referência [9].

Dessa forma o valor dos deslocamentos são determinados no vetor *ub* de tamanho *ksize*.

A resposta de deslocamentos obtida pelo MEF implementado nesse *software* foi checada com a do *software* comercial Ansys 6.1. e os resultados se mostraram corretos.

Aqui vale a pena ressaltar de que apesar do *software* ser desenvolvido para a utilização de elementos quadriláteros não há problema em se utilizar malhas mistas com elementos de três ou quatro nós. Em uma malha mista o Ansys 6.1 respresenta o elemento triangular como elemento quadrilátero com dois nós coincidentes, essa representação não acarreta problemas para a formulação apresentada.

Outros valores necessários para a otimização são fornecidos dentro do corpo do programa, e são eles, *Vfrac* representa o limite máximo de material representado na forma de fração do volume inicial, a pseudo-densidade inicial de cada elemento é definida como *Vfrac* de tal modo que a restrição de volume inicia o processo de otimização ativa.

O controle do valor da penalização p utilizada no Método da Continuação é feita dentro do corpo do programa. O *software* esta preparado para ter até cinco etapas onde se pode controlar dentro de cada uma: o número máximo de iterações, o critério de parada ou saída da etapa e a ativação ou não do filtro.

O loop de otimização se inicia com o cálculo da matriz de rigidez global, dado os valores de pseudo-densidade inicial, para essa tarefa há um loop de *M* (número de elementos) voltas para se calcular a matriz de rigidez local e o vetor de forças de campo de cada elemento. É então formada a matriz de rigidez global e o vetor de forças, o sistema linear é resolvido.

Em posse dos deslocamentos nodais são calculados a função objetivo e os gradientes desta. Neste momento deve-se iniciar o módulo de otimização, para isso são calculados os limites móveis de cada variável.

Os limites móveis são calculados a partir de um valor inicial e são limitados por valores superiores e inferiores. Existem três variáveis, *sign1, sign2* e *sign3,* que representam se o valor da pseudo-densidade de cada elemento está aumentando ou diminuindo ao longo das três ultimas iterações. Dessa forma caso a pseudo-densidade de um elemento esteja oscilando, o que é comum quando se está próximo da solução final, os limites móveis são diminuídos auxiliando a convergência da solução.

O processo iterativo irá seguir a programação feita pelo método da continuação, ao final de cada etapa é gerado um arquivo texto em formato de APDL, que é a linguagem de programação do *software* Ansys onde está apresentada a distribuição de pseudo-densidadas corrente naquela iteração. Ao se ler este arquivo em Ansys são geradas as figuras que permitem visualizar o resultado.

Ao final do processo de otimização é gerado um arquivo de extenção *.m* para ser lido em Matlab. Este arquivo irá gerar os gráficos de convergência da função objetivo e restrição de volume ao longo das iterações.

A listagem parcial do *software* desenvolvido está apresentada no Anexo A apenas as partes mais importantes são apresentadas.

3 Resultados

Neste capítulo serão apresentados os resultados obtidos com o *software* desenvolvido. Inicialmente serão apresentados os resultados para uma viga em balanço, onde poder-se-á discutir as particularidades do MOT considerando forças inerciais, em seguida será abordado o problema de síntese de um rotor de alta velocidade de uma máquina rotativa, no caso uma turbina a vapor.

3.1 Viga engastada

De modo a validar o *software* e compreender as características da implementação considerando forças de campo no MOT, foi estuda a síntese de uma viga em balanço.

O domínio fixo estendido utilizado consiste na geometria abaixo apresentada.

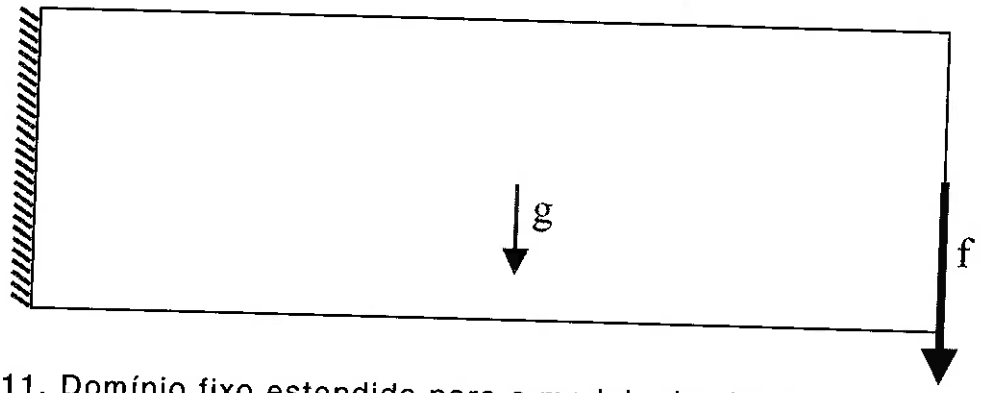

Figura 11. Domínio fixo estendido para o modelo de viga em balanço.

Inicialmente será apresentada a influência do método da continuação na qualidade do resultado obtido. A título de exemplo será sintetizada a estrutura da viga em balanço sujeita apenas a força externa **f**. No gráfico de convergência da Figura 12 serão definidos os pontos , *A*, *B*, *C* e *D*, de alteração dos parâmetros de otimização.

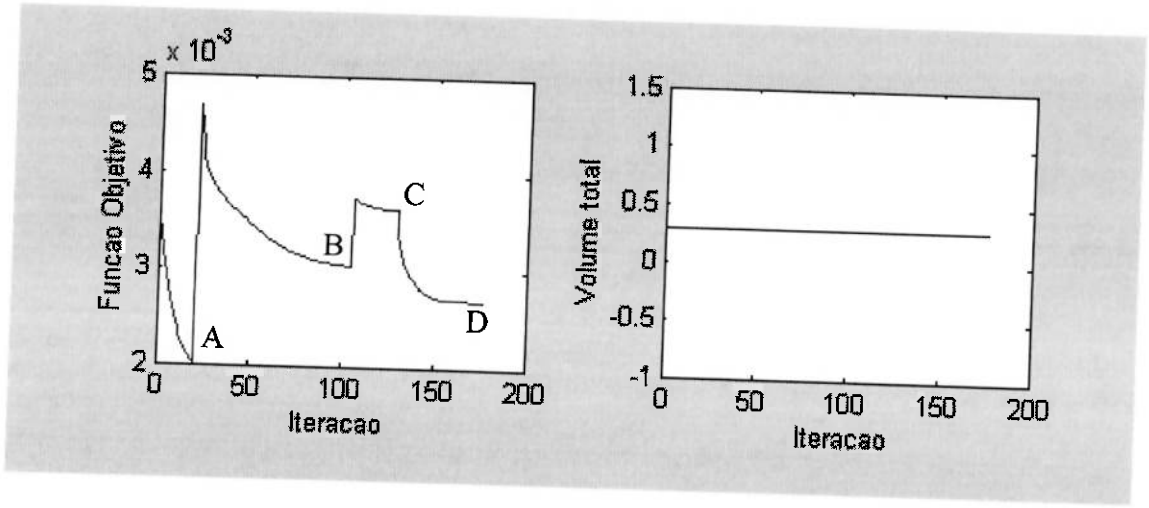

Figura 12. Curvas de convergência para uma viga engastada convergência sujeita apenas à força externa.

O trecho definido entre o início das iterações e o ponto *A*, representa a etapa do processo iterativo onde a penalização p igual a um e o filtro permanece desligado. Neste instante estrutura obtida esta apresentada na Figura 13

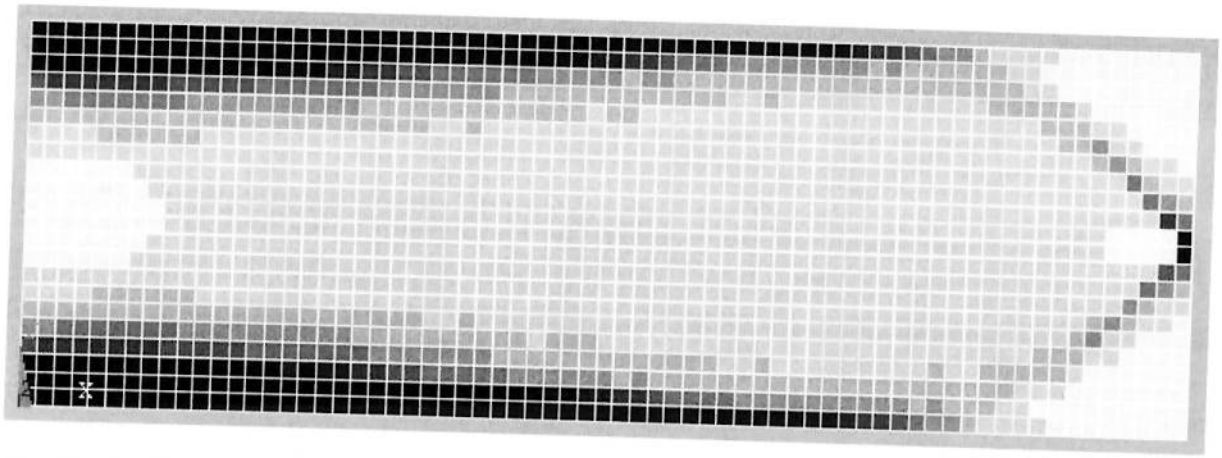

Figura 13. Estrutura obtida na instante representado pelo ponto *A*, no processo iterativo.

Era de se esperar, como se observa acima, que neste momento a estrutura ainda não estivesse bem definida, pois não foi permitida a convergência da solução.

No trecho entre os pontos *A* e *B*, penalização p é elevada ao valor 2 e o filtro é ligado, obtém-se então a estrutura apresentada na Figura 14.

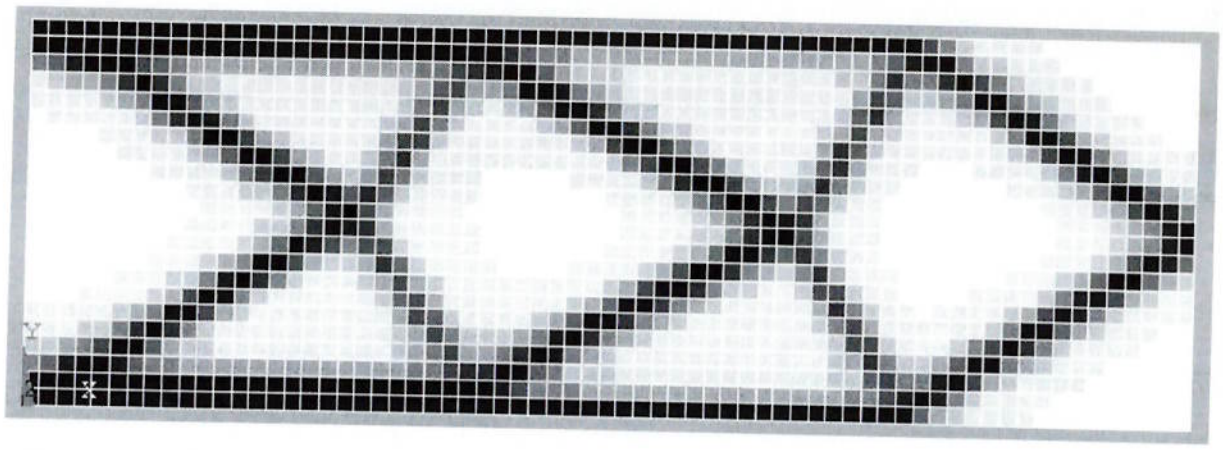

Figura 14. Estrutura obtida na instante representado pelo ponto B, no processo iterativo.

Observa-se neste momento que a estrutura já esta bem mais definida, porem ainda há um grande número de elementos com pseudo-densidades intermediárias, isso ocorre pois o filtro não permite a variação espacial brusca de pseudo-densidade.

Na próxima etapa a penalização é elevada a seu valor final 3, e são feitas mais *n* iterações necessárias para obter a convergência, atingindo então o ponto representado pela letra *C*. A estrutura obtida está apresentada na Figura 15.

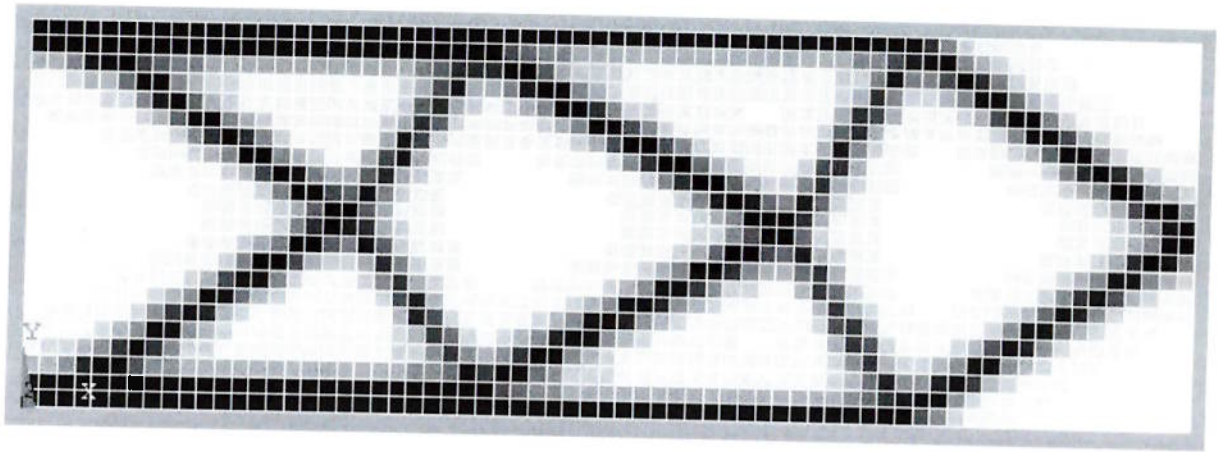

Figura 15. Estrutura obtida na instante representado pelo ponto C, no processo iterativo.

Observa-se que há pouca alteração nas estruturas referentes aos pontos *B* e *C*, para retirar escala de cinza é necessário a desativação do filtro, permitido que existam

elementos adjacentes com valores de pseudo-densidade próximos de zero e um. Entre o ponto *C* e *D* a penalização p é mantida igual a 3 e o filtro é desligado, observa-se nesta etapa uma queda significativa da função objetivo devido a retirada dos elementos cinza que possuem sua rigidez penalizada. O resultado final é apresentado na Figura 16.

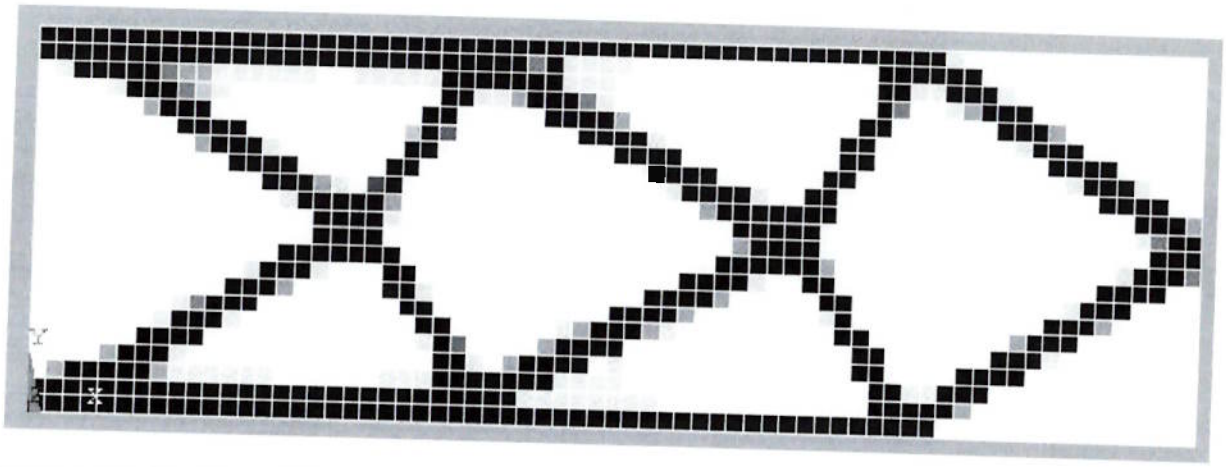

Figura 16. Estrutura obtida na instante representado pelo ponto C, no processo iterativo.

Apresentada a capacidade do Método da Continuação em melhorar a qualidade do resultado final será discutido agora a influência da força peso na solução obtida. Para tanto foram sintetizadas quatro estruturas com diferentes relações entre força peso do domínio fixo estendido devido à aceleração da gravidade e a força externa.

Inicialmente é apresentada na Figura 17 a estrutura ótima tal que a aceleração da gravidade gere um carregamento igual a dez vezes a carga externa aplicada a estrutura. Em um caso prático essa situação pode existir em um componente que sofre grande acelerações porem suporta uma carga não muito grande.

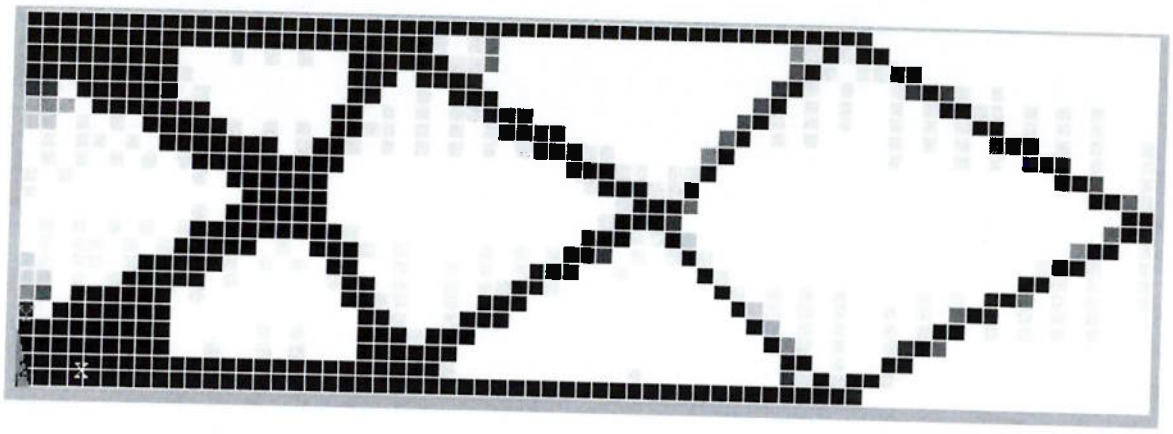

Figura 17. Estrutura obtida com o peso próprio do domínio fixo estendido igual a 10 vezes força externa aplicada.

Observa-se nesse resultado que o método concentra material na região próxima ao engaste, como era de se esperar, minimizando assim o momento fletor exercido pelo peso próprio da estrutura.

A figuras 18 e 19 apresentam outras relações de magnitude entre força peso e carregamento externo.

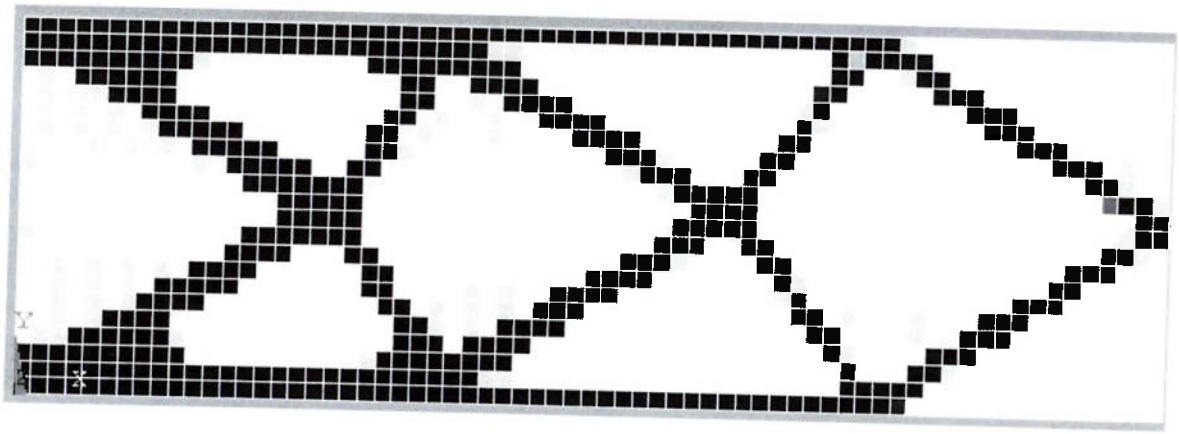

Figura 18. Estrutura obtida com o peso próprio do domínio fixo estendido igual a 2 vezes força externa aplicada.

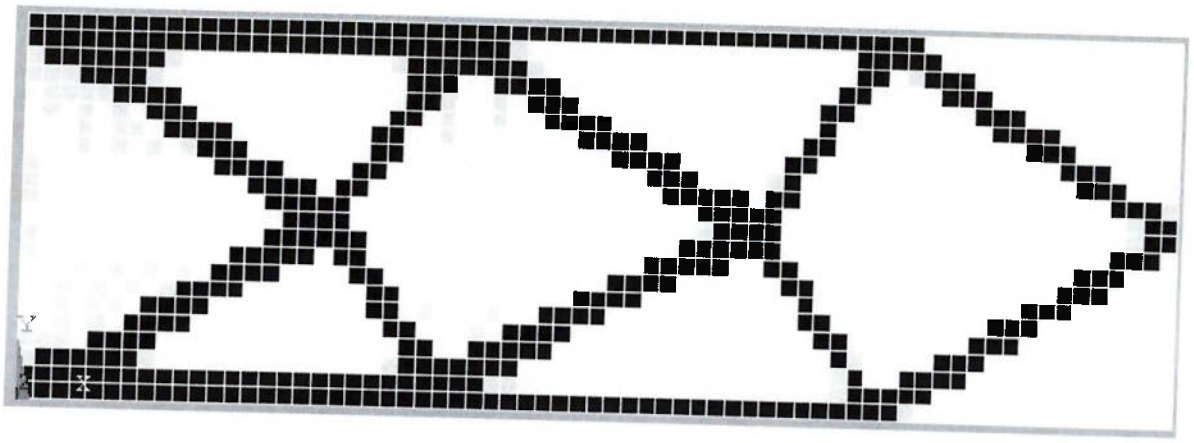

Figura 19. Estrutura obtida com o peso próprio do domínio fixo estendido igual a força externa aplicada.

A título de comparação a Figura 20 apresenta o resultado desconsiderando o peso próprio da estrutura.

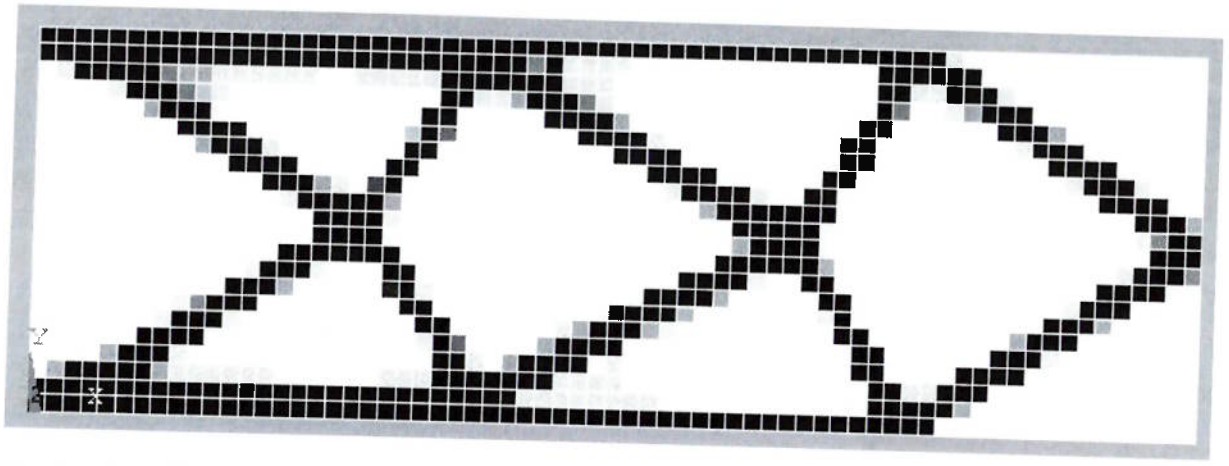

Figura 20. Estrutura obtida apenas com a aplicação da força externa.

Com os resultados acima apresentado, foi possível validar o *software* uma vez que estes se apresentaram similar aos da referência [1] bem como adquirir um sentimento melhor do problema de OT considerando peso próprio. Assim partiu-se para resolver um problema de caráter prático, o cubo do rotor de uma turbina a gás.

3.2 Rotor de uma turbina a gás

O problema de projeto de um cubo de rotor de uma turbina a gás foi extraído do artigo da referência [8], onde o problema é tratado utilizando métodos evolucionários de otimização.

Conforme apresentado nesse artigo ao se projetar um rotor de uma turbina a gás deve-se analisar a resposta da estrutura frente a três tipos básicos de carregamento, que são eles carga térmica, forças centrifuga devido à alta rotação e forças externas aplicadas devido ao acoplamento cubo eixo e a interação pá fluido. As principais respostas a serem analisadas são: a distribuição de tensão na estrutura e sua deformação. Outro fator de grande importância no projeto desse tipo de componente é sua massa, pode se dizer que a redução de massa é o principal objetivo no projeto de rotores de turbinas a gás [8].

No artigo o problema de otimização é formulado como:

Minimizar a flexibilidade e/ou a massa

Tal que: Restrições geométricas e de tensão fossem atendidas.

Nesse trabalho não foi considerada a tensão como restrição do problema, entretanto quando se usa a flexibilidade como função objetivo é provado matematicamente que as tensões médias da estrutura tendem a diminuir, mas não é garantido que a tensão local não atingirá um dado valor.

Primeiro passo para a síntese da estrutura por OT é a definição do domínio fixo estendido e do carregamento da estrutura.

De modo a adaptar o problema para o estado plano de tensão foi proposto o domínio fixo estendido apresentado na Figura 21, onde o centro do cubo é fixo e a carga das pás é distribuída ao cubo apenas em quatro pontos da circunferência, foram consideradas também forças tangenciais de mesma magnitude que as forças radiais de modo a simular o torque gerado pelas pás.

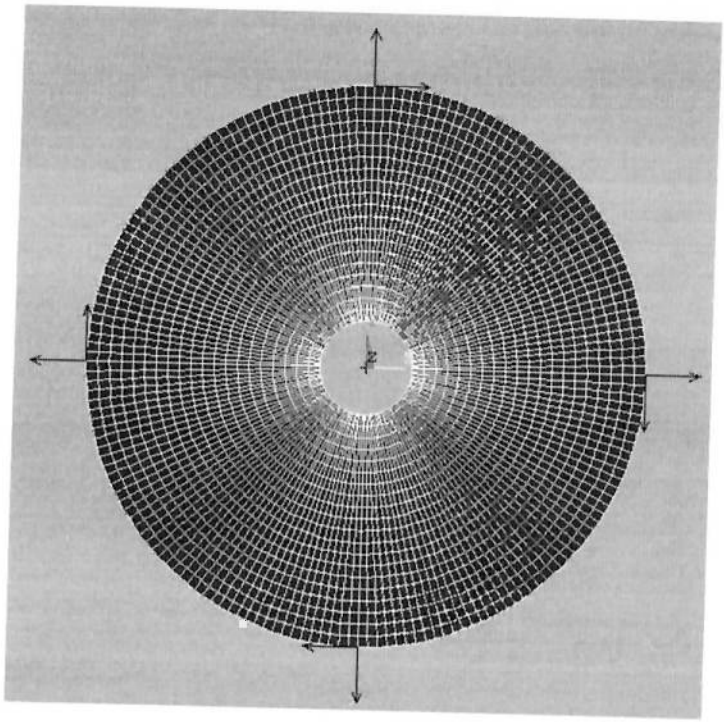

Figura 21. Domínio fixo estendido e condições de contorno para o cubo do rotor de uma turbina a gás.

Para este problema foram geradas duas famílias de resultados, sendo uma com restrição de volume em 50% e outra com 30% do volume inicial.

Com a restrição de 30% foram feitas sínteses com malhas de diferentes tamanhos. Na figuras que seguem é possível observar os resultados obtidos.

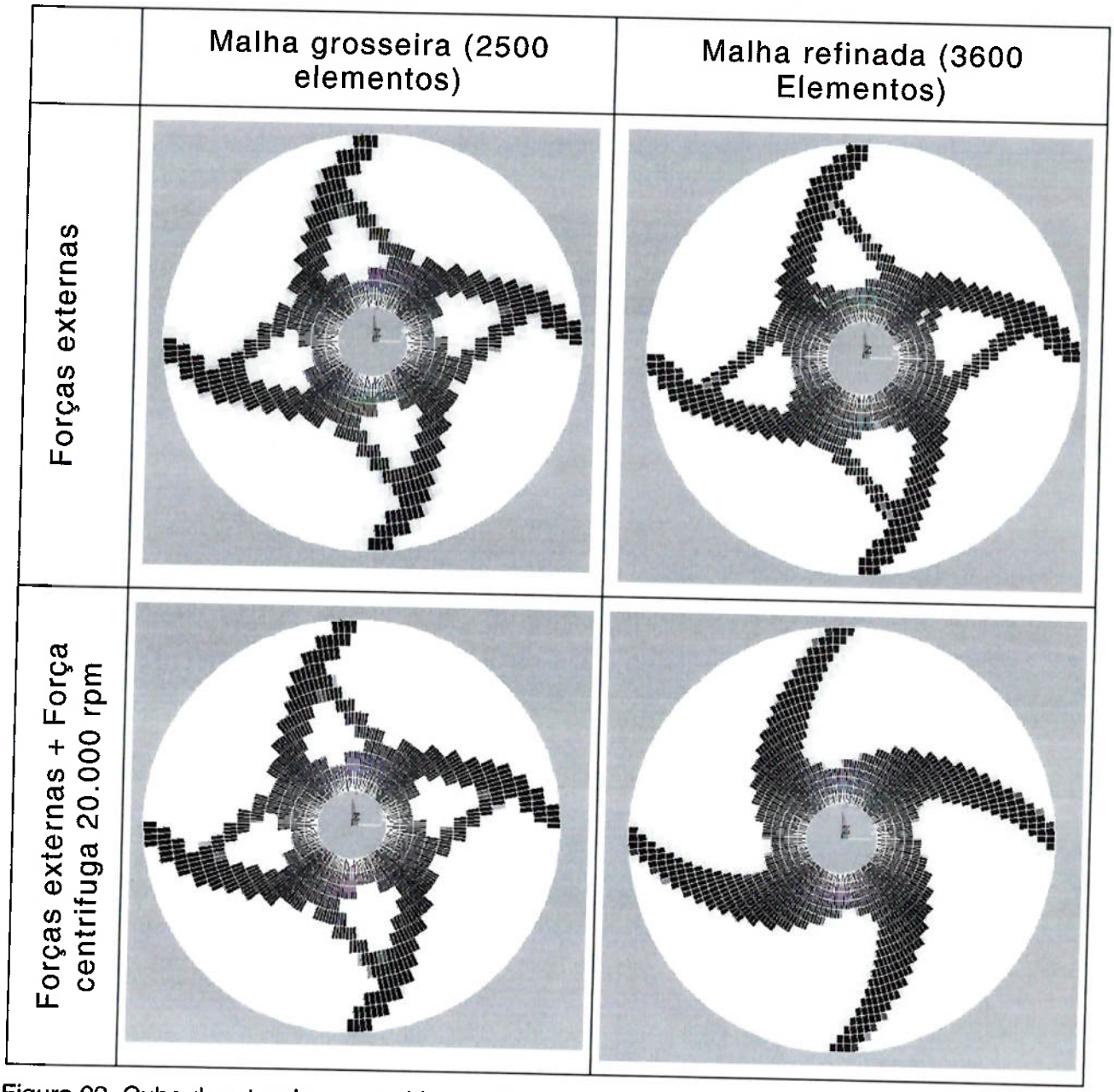

Figura 22. Cubo do rotor de uma turbina a gás, resultados obtidos com restrição de volume igual a 30%

Para avaliar a influência da rotação foi sintetizada uma estrutura com 30% de restrição de volume e força centrifuga referente à rotação de 50.000 rpm, o resultado esta abaixo apresentado.

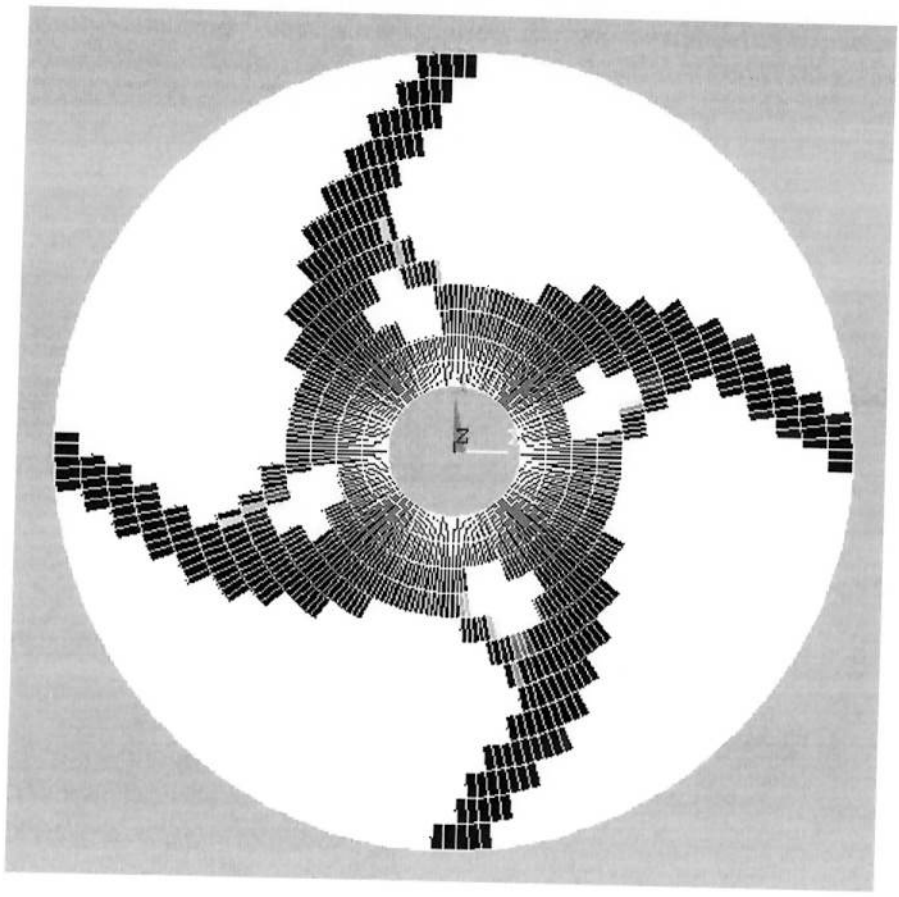

Figura 23. Estrutura obtida com 30% de restrição de volume e rotação igual a 50.000 rpm

A partir desses resultados podemos verificar que a discretização da malha tem influência no resultado obtido como se observa na Figura 22 os resultados obtidos com a força centrifuga apresentam diferença significativa entre a malha grosseira e a refinada. A questão de dependência da malha é um é um fato que vem sendo discutido na literatura específica a algum tempo, e não constitui exatamente um problema do MOT, mas apenas um característica que pode ser contornada utilizando filtros.

Outro ponto observado é a influência da força centrifuga no resultado obtido, como era de se esperar.

Dessa forma ao se otimizar uma estrutura com forças dependentes da massa é importante que a relação de magnitude entre estas forças e as forças externas representem com precisão os carregamentos aos quais a estrutura estará realmente

submetida, pois se observa uma razoável sensibilidade da geometria em relação a forças inerciais.

Em todos estes resultados a otimização convergiu e a restrição de volume não foi violada. Abaixo é apresentado um o gráfico de uma das sínteses, apenas para ilustrar a forma geral das curvas de convergência.

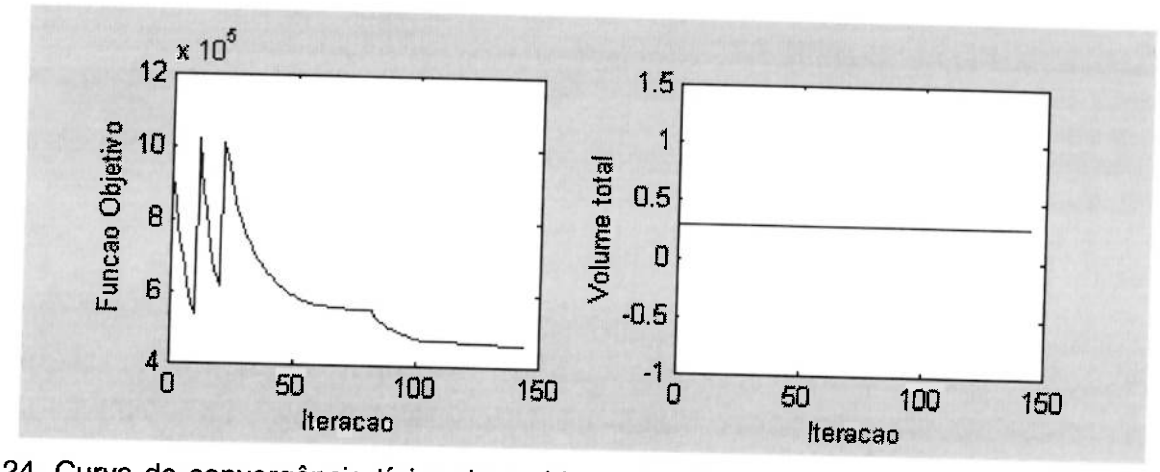

Figura 24. Curva de convergência típica do problema de síntese do rotor com restrição de 30%.

As curvas de convergência da maioria das sínteses do cubo da turbina a gás apresentam o comportamento observado na Figura 24. As descontinuidades da função objetivo se devem ao fato do método da continuidade ser utilizado. Como se pode observar há três trechos suaves, um para cada valor de penalização p. As descontinuidades ocorrem devido a alteração do valor de p, que ao se elevar penaliza os elementos de pseudo-densidade intermediária aumentando a flexibilidade da estrutura.

A ultima alteração de parâmetro do método da continuidade é apenas a desativação do filtro, no gráfico da Figura 24, observa-se uma descontinuidade na derivada da flexibilidade, em torno da iteração 80, isto ocorre pois com a desativação do filtro é permitido que os elementos próximos a estrutura tenha suas pseudo-densidades diminuídas, criando uma variação brusca de pseudo densidade, porém a função objetivo é minimizada mais rapidamente.

Os resultados obtidos com a restrição de 50% do volume serão abaixo apresentados.

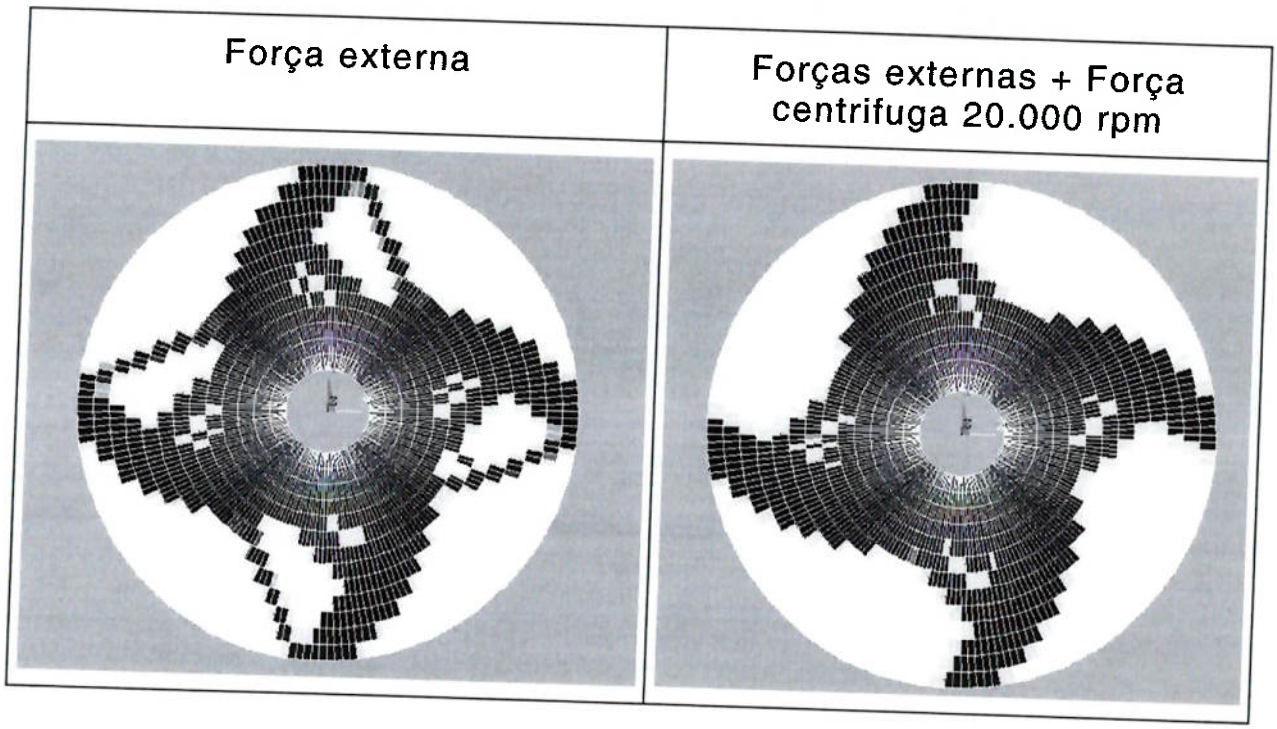

Figura 25 Cubo do rotor de uma turbina a gás resultados obtidos com restrição de volume igual à 50%

Os resultados obtidos para com a restrição de volume igual a 50% apresentam, a mesma tendência que o problema com restrição de 30%, como que era de se esperar.

Com estes resultados, fica demonstrado que a aplicação do MOT em um caso pratico é viável.

4 Conclusão

Neste trabalho foi apresentado o Método de Otimização Topológica aplicado ao projeto de estruturas rotativas

A formulação considerando forças inerciais foi apresentada e implementada.

Foi discutida a importância dessa formulação no projeto de componentes mecânicos de máquinas rotativas que operam em alta velocidade, e a influência dessas considerações tanto no resultado obtido como no próprio MOT.

O *software* desenvolvido nesse trabalho se mostrou robusto e rápido, capaz de sintetizar estruturas cujo domínio fixo estendido possui malhas relativamente grandes, de até 4000 elementos, em um tempo de no máximo 4 horas em um computador *Pentiun 4 Intel.*

Esse trabalho apesar de seu interesse mais acadêmico, de desenvolvimento e validação do Método de Otimização Topológica, não deixou de lado a visão prática de projeto de engenharia, e por se tratar de um projeto de formatura, esteve sempre preocupado com a viabilidade técnica.

A viabilidade técnica da utilização do MOT no meio industrial se traduz em obter resultados de boa qualidade em tempo hábil.

Um problema de caráter prático foi resolvido, a despeito das simplificações, uma vez que não há grande familiaridade com o problema, o método se mostrou capaz de fornecer soluções estruturais interessantes.

Como continuação desse trabalho na área de projeto de rotores de turbina a gás, propõe-se, de modo a tornar o *software* mais próximo do problema real, a alteração do módulo de elementos finitos de modo a sintetizar estruturas axissimétricas. Tal alteração não acarreta grandes alterações no *software,* logo é de fácil implementação, porém os resultados obtidos podem ser facilmente comparados com outros resultados presentes na literatura.

Este trabalho deixa para as gerações futuras um *software* capaz de atender a problemas de projeto de estruturas rotativas auxiliando, dessa forma, na disseminação dos métodos de otimização entre os alunos da Escola Politécnica e outros interessados.

5 Bibliografia

[1] Bendsoe, M. P., Sigmund O., "Topology Optimization, Theory, Methods and Aplications" Springer – Verlag, Berlin, Germany, 2003.

[2] Zhou M., Shyy Y.K. e Thomas H.L., "Checkerboard and minimum member size control in topology optimization", Struct Multidisc Optim 21, 2001, págs. 152-158.

[3] Bathe, K., "Finite Elements Procedures", Prentice Hall, 1996.

[4] Fujii D., Kikuchi N., "Improvement of numerical instabilities in topology optimization using the SLP method", Structure Multidisciplinary Optimization 21, 2001, págs 159-163.

[5] Cardoso, E. L., "Controle da Complexidade na Otimização topológica de estruturas continuas" Porto Alegre 200. 120p Dissertação (Mestrado) Escola de Engenharia, Universidade Federal do Rio Grande do Sul.

[6] Vanderplatz, G. "Numerical Optimization Techiniques for Engineering Design: with Applications". McGraw-Hill, New York, EUA, 1984

[7] Haftka, R. T., Gürdal, Z. e Kamat, M. P. "Element of Structural Optimization". Kluwer Academic Publishers, 1990

[8] Liu JS, Parks GT, Clarkson PJ., 2002 "Optimization of turbine disk profiles by metamorphic development" Journal of Mechanical Design, 124 (2) pp 192-200

[9] Press, W. H., Teukolsky, S. A., Vetterling, W. T., Flannery, B. P., " Numerical Recipes in C – The Art of Scientific Computing", págs 430-444, Cambridge University Press, 1999.

6 Anexos

6.1 Anexo A

```
#include <stdio.h>
#include <malloc.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include "nrutil.h"
//#include "func_r1.h"

#define Nome "Vibal_fi05_f0_g1_1564_v30"

#define Caminho ""

#define Entrada  Caminho""Nome".txt" /*input*/

#define SaidaP1          Caminho""Nome"_P1.ans"              /*output*/
#define SaidaP2          Caminho""Nome"_P2.ans"              /*output*/
#define SaidaP3          Caminho""Nome"_P3.ans"              /*output*/
#define SaidaP4          Caminho""Nome"_Filtro.ans"  /*output*/
#define SaidaP5          Caminho""Nome"_Cont.ans"    /*output*/
#define arqANSYS         Caminho""Nome"_Final.ans"   /*output*/
#define arqAux           Caminho""Nome"_Aux.txt"             /*output*/
#define arqMATLAB        Caminho""Nome".m"                   /*output*/
#define arqhist          Caminho""Nome".out"                 /*output*/

#define RAIO 0.005
#define FracVol 0.3

#define ITERTOL 0.5e-3

#define MAXITER 400

#define THRESH 1.0e-12
#define PI 3.1415192

// Valores de Filtro
#define minf 0.95
#define msup 1.05

#define mllower 0.03
#define mlupper 0.07

// Limites da pseudo-densidade
#define ZERO 1.0e-3
#define HUM 1

// Defini?es do linbcg
#define TOL 1.0e-7
#define ITOL 1
#define ITMAX 500
#define TAM 150
#define NMAX 1000000

extern void dusrmt_();
extern void dsplp_();
```

```
void output(char *arqANS, int M, int N, int **conect, double **coord, double *xmax, double
*xmin, double *xold);
void nrerror(char error_text[]);
void dsprsin(double **a, unsigned long n, double thresh, unsigned long nmax, double sa[],
unsigned long ija[]);
void linbcg(unsigned long n, double b[], double x[], int itol, double tol, int itmax, int *iter,
double *err);
void simplx (float **a, int m, int n, int m1, int m2, int m3, int *icase, int izrov[], int iposv[]);
void filter(int M, int N, float **coord, int **conect, double *b, double *filt_b, float radius);
double dist(float *xei, float *yei, float *xej, float *yej);
double size_elem(float *xe, float *ye);
void AuxMatF(double *xe, double *ye, double s, double t, double *detJ, double *N);
void AuxMat(double *xe, double *ye, double s, double t, double *detJ, double *dNdx, double
*dNdy);
void AuxMatV(double *xe, double *ye, double s, double t, double *detJ);
void Ke(double *xe, double *ye, double E, double v, double *Kel,double ro, double p);
void Fe(double *xe,double*ye,double ro_elem,double g_x,double g_y,double omg_x,double
omg_y,double omg_z,double *Fel, double ro, double p);
double Vol(double *xe,double*ye, double ro, double p);


// VARIAVEIS GLOBAIS

unsigned long *ija;
double *sa;

void main () {

        //VARIAVEIS AUXILIARES DE LEITURA DE DADOS E CONTROLE  DE LOOP
        int i, j, k,  count, iter, size, ELE;
        int   d2, f2, compara,  q[8];
        int b0, b1, b2, b3, b4, b5, b6, b7, b8, b9, b10, dof1, dof2, dof3, dof4;
        unsigned long ii, jj;
        char *temp, aux[15], d1[5], d3[5], e1, e2, e3;
        char f1[5], f3[5], f4[5], f6, f7, f8, c1[14], c2[14], c3[15], c4[5], c5, c6[5], c7[5];
        char g1[15], g2[10], g3[5], g4[5], Saida[50];
        float a4, a5, f5;
        double teste, dv;
        FILE *Auxiliar, *arquivo, *historia;
        FILE *Entr, *arq3;

        //VARIAVEI DO MEF
        int **conect, **spc1, **ID, **LM, M, N;
        double **coord, **loada3, *fb, *fbext;
        double *Kel, **K, *ye, *xe,  *ml;
        double g_x, g_y, omg_x, omg_y, omg_z, ni, E, real_dens;
        double *ub, *bf, *Fel, err ;
        unsigned long ksize;

        //VARIAVEIS DO MOT
        double *xmin, *xmax, *xnew, *xold, *xlower, *xupper, *area; // *xupper, *xlower,,
*xold, ;
        float x0, Vfrac;
        double *compliancia, *difer, *volume, *gradF, penalK,penalV,penalF, Vol_0, Vol_1;
// *objetivo, ,
```

```
        int it,  opcao, filter_en, itcontrole;
        double *ube, tempmat[8], dcompl, dfdensF, dfdensK;
        double *sign1, *sign2, *sign3, *filt_xupper, *filt_xlower, radius, complianciaold;
        int maxiterP1, maxiterP2, maxiterP3, maxiterFiltr, maxiterCont;

        /* Variaveis necessarias para o dsplp */
        int nvars,mrelas;
        double *prgopt,*dattrv,*primal,*duals,*work,*bl,*bu,*costs;
        int  *ind,*ibasis,*iwork;
        int  lw,liw,info,lamat,lbm;

        ////////////////////ALOCA?O DE MEMORIA//////////////////////////////////

        sa = dvector(1,NMAX);
        ija = lvector(1,NMAX);

        ////////////////////////////INICIO DA LEITURA DE DADOS////////////////////////////
        Auxiliar = fopen(arqAux,"w");                                   // Abertura do arquivo
auxiliar para Debbug

        Entr=fopen(Entrada,"r");                                                // Abre o arquivo
de entrada

        temp = (char *)malloc(TAM*sizeof(char ));        // A leitura ?sempre feita no arquivo
temp

        ////////////////////////////////////////////////////////////////////////////////

        /* VALOR DE "N" (NUMERO DE NOS) */

        compara = -1;
        while ( compara > 0 || compara < 0) {
                fgets(temp, TAM, Entr);
                compara = strncmp( temp, "NUMOFF,NODE,", 12);
        }
        sscanf( temp, "%s %d", c1, &N ); /*le a string do temp e atribui a N como inteiro*/

        // IMPRIME AUXILIAR
        fprintf(Auxiliar," Numero de ns = %d \n",N);

        ////////////////////////////////////////////////////////////////////////////////

        /* VALOR DE "M" (NUMERO DE ELEMENTOS) */

        fgets(temp, TAM, Entr);
        sscanf( temp, "%s %d", c2, &M );

        // IMPRIME AUXILIAR
        fprintf(Auxiliar," Numero de elementos = %d \n",M);

        ////////////////////////////////////////////////////////////////////////////////

        /* VETOR "COORD" (COORDENADAS DOS NOS) */

        // IMPRIME AUXILIAR
```

```
        ////////////////////////////////////////////////////////////////////////////

        /* VETOR "CONECT" (GRAUS DE LIBERDADE DOS NOS) */

        conect = imatrix(1,M,1,4);

        compara = -1;
        while ( compara > 0 || compara < 0) {

                fgets(temp, TAM, Entr);
                compara = strncmp( temp, "(19i8)", 6);
        }

        k=1;

        while (k<=M){
                fgets(temp, TAM, Entr);
//              printf("%s\n", temp);
                sscanf( temp, "%d %d %d %d %d %d %d %d %d %d %d %d %d %d %d
%d",
                        &b0, &b1, &b2, &b3, &b4, &b5, &b6, &b7, &b8, &b9, &b10,
&dof1, &dof2,
                        &dof3, &dof4 );

                conect[k][1] = dof1;
                conect[k][2] = dof2;
                conect[k][3] = dof3;
                conect[k][4] = dof4;
                k++;
        }

        ////////////////////////////////////////////////////////////////////////////

        /* MODULO DE ELASTICIDADE "E" */

        compara = -1;
        while ( compara > 0 || compara < 0) {

                fgets(temp, TAM, Entr);
                compara = strncmp( temp, "MPDATA,R5.0, 1,EX", 17);
        }
        sscanf( temp, "%s %s %c %s %s %lf", c3, c4, &c5, c6, c7, &E );


        ////////////////////////////////////////////////////////////////////////////

        /* DENSIDADE "real_dens" */

        compara = -1;
        while ( compara > 0 || compara < 0) {

                fgets(temp, TAM, Entr);
                compara = strncmp( temp, "MPDATA,R5.0, 1,DENS,", 20);
        }
        sscanf( temp, "%s %s %s %s %lf", g1, g2, g3, g4, &real_dens );
```

```
////////////////////////////////////////////////////////////////////////////////

/* COEFICIENTE DE POISSON "NI" */

compara = -1;
while ( compara > 0 || compara < 0) {

        fgets(temp, TAM, Entr);
        compara = strncmp( temp, "MPDATA,R5.0, 1,PRXY,", 20);
}
sscanf( temp, "%s %s %s %s %lf", g1, g2, g3, g4, &ni );

// IMPRIME AUXILIAR
fprintf(Auxiliar,"Coeficiente de Poisson %lf \n",ni);


////////////////////////////////////////////////////////////////////////////////
/* ACELERACAO DA GRAVIDADE */

compara = -1;
while ( compara > 0 || compara < 0) {
        fgets(temp, TAM, Entr);
        compara = strncmp( temp, "ACEL,", 5);
}
sscanf( temp, "%s %lf %s %lf %s %s", g1, &g_x, g3, &g_y, g4, g1);

// IMPRIME AUXILIAR
fprintf(Auxiliar,"Aceleração da gravidade x = %lf ; y = % lf \n", g_x, g_y);

////////////////////////////////////////////////////////////////////////////////

/* ROTACAO DA ESTRUTURA */

compara = -1;
while ( compara > 0 || compara < 0) {
        fgets(temp, TAM, Entr);
        compara = strncmp( temp, "OMEGA,", 6);
}
sscanf( temp, "%s %lf %s %lf %s %lf %s %s", g1, &omg_x, g2 ,&omg_y, g3,
&omg_z, g4, g1);

// IMPRIME AUXILIAR
fprintf(Auxiliar,"Velocidade angular de rotacao Omega_x = %lf ; Omega_y = % lf ;
Omega_z = % lf \n", omg_x, omg_y, omg_z);

////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////

/* MATRIZ DE RESTRI?ES "SPC1" (GRAUS DE LIBERDADE FIXOS) */

spc1 = imatrix(1,N,1,2);

for (k=1; k<=N; k++) {
        spc1[k][1]=1;
        spc1[k][2]=1;
```

```
}
.
.
.
////////////////////////////////////////////////////////////////////////////////

/* MATRIZ DE FOR?S "LOADA3" */

loada3=dmatrix(1,N,1,2);

for (k=1; k<=N; k++) {
        loada3[k][1]=0;
        loada3[k][2]=0;
}

.
.
.
//////////////////////////LEITURA DOS DADOS AUXILIARES////////////////////////////////

Vfrac = FracVol;                ////////////////////////////////////////////////////////////
x0=Vfrac;                       ////////////////////////////////////////////////////////////
radius=RAIO;                    ////////////////////////////////////////////////////////////
maxiterP1=10;                   ////////////////////////////////////////////////////////////
maxiterP2=10;                   ////////////////////////////////////////////////////////////
maxiterP3=200;                  ////////////////////////////////////////////////////////////
maxiterFiltr=30;                ////////////////////////////////////////////////////////////
maxiterCont=100;                ////////////////////////////////////////////////////////////

penalK=1;                       ////////////////////////////////////////////////////////////
penalV=1;
penalF=1;

filter_en=0;                    ////////////////////////////////////////////////////////////


////////////////////////////////MATRIZES FIXAS/////////////////////////////////////

/* VETOR "ID" (MATRIZ DE DOF's NAO RESTRITOS ENUMERADOS) */

ID =imatrix(1,N,1,2);
count=1;

for (i=1; i<=N; i++){
        for (j=1;j<=2;j++){

                if (spc1[i][j]==0) {
                        ID[i][j]=0;
                }
                else {
                        ID[i][j]=count;
                        count++;
                }
```

```
            }
        }

        ////////////////////////////////////////////////////////////////////////////////

        /* TAMANHO "ksize" DA MATRIZ REDUZIDA */

        ksize = count-1;

        ////////////////////////////////////////////////////////////////////////////////


        /* VETOR DE CARGAS "FB" */


        fb=dvector(1,ksize);
        fbext=dvector(1,ksize);

        for (ii=1; ii<=ksize; ii++) {
                fb[ii]=0;
                fbext[ii]=0;
        }

        for (i=1; i<=N; i++) {
                for(j=1;j<=2;j++)
                        if (ID[i][j]!=0) {
                                fbext[ID[i][j]]=loada3[i][j];

                }

        }


        ////////////////////////////////////////////////////////////////////////////////

        /* VETOR "LM" */

        LM=imatrix(1,M,1,8);

        for (i=1; i<=M; i++) for (j=1; j<=4; j++) LM[i][j]=0;/*para zerar o vetor
inicial*/

        for (i=1; i<=M; i++) {
                for (j=1; j<=4; j++) {
                                LM[i][2*(j-1)+1]=ID[conect[i][j]][1];
                                LM[i][2*(j-1)+2]=ID[conect[i][j]][2];
                }
        }


        ////////////////////////////ALOCA?O DE MEM?IA////////////////////////////////////
```

```
xmin = dvector(1,M);
xmax = dvector(1,M);
xnew = dvector(1,M);
xold = dvector(1,M);
xupper = dvector(1,M);
xlower = dvector(1,M);
sign1 = dvector(1,M);
sign2 = dvector(1,M);
sign3 = dvector(1,M);
ml = dvector(1,M);
area = dvector(1,M);

volume=dvector(1,MAXITER);
compliancia=dvector(1,MAXITER);
difer=dvector(1,MAXITER);

bf = dvector(1,ksize);
ub= dvector(1,ksize);
ube = dvector(1,8);
gradF=dvector(1,M);
K=dmatrix(1,ksize,1,ksize);

xe = (double *)malloc(4*sizeof(double));
ye = (double *)malloc(4*sizeof(double));
Kel = (double *)malloc(64*sizeof(double ));
Fel = (double *)malloc(8*sizeof(double ));


filt_xupper = (double *)malloc(M*sizeof(double ));
filt_xlower = (double *)malloc(M*sizeof(double ));

//nmeros de v?iaveis (nvars) e nmero de rstri?es (mrelas)
nvars = M;
mrelas = 1;
lamat=4*nvars+7;
lbm=8*mrelas;
lw=4*nvars+8*mrelas+lamat+lbm;
liw=nvars+11*mrelas+lamat+2*lbm;

//Dimensiona arrays que serao utilizados aqui
bl=dvector(1,mrelas+nvars); //restri?es inferiores
bu=dvector(1,mrelas+nvars); //restri?es superiores
prgopt=dvector(1,2);
dattrv=dvector(1,(2*mrelas*nvars)+1+nvars);
ind=ivector(1,mrelas+nvars);
primal=dvector(1,mrelas+nvars);
duals=dvector(1,mrelas+nvars);
ibasis=ivector(1,mrelas+nvars);
costs = dvector(1,M);
work=dvector(1,lw);
iwork=ivector(1,liw);
////////////////////////// INICIALIZA?O DOS VETORES DO LOOP//////////////////

for (i=1; i<=MAXITER; i++) {
        difer[i]=1;
}
```

```
for (i=1; i<=MAXITER; i++) {
        volume[i]=0;
}

for (i=1; i<=MAXITER; i++) {
        compliancia[i]=0;
}

//vetor das densidades iniciais
for (i=1; i<=M; i++) {
        xmin[i]=ZERO;
        xmax[i]=HUM;
}

for (i=1; i<=M; i++) {
        xnew[i]=x0;
}

//inicializa?o das vari?eis auxiliares para c?culo dos limites
//mveis
for (i=1; i<=M; i++) {

        sign1[i] = 1;
        sign2[i] = 1;
        sign3[i] = 1;
        ml[i] = mlupper;

}

// Deslocamentos iniciais
for (ii=1; ii<=ksize; ii++){
        ub[ii]=0.0;
}


// Calculo do Volume Inicial

Vol_0=0.0;
for(k=1; k<=M; k++) {

        for(i = 0; i < 4; i++){
                xe[i] = coord[conect[k][i+1]][1];
                ye[i] = coord[conect[k][i+1]][2];
        }

        area[k] = Vol(xe, ye, 1,1);
        Vol_0 = Vol_0 + Vol(xe, ye, 1,1);
}

Vol_1=Vol_0*Vfrac;

//Normaliza?o do volume
for(k=1; k<=M; k++) {
        area[k]=area[k]/Vol_0;
}
```

```
printf("Volume total %f \n", Vol_0);
printf("Volume maximo final %f \n", Vol_1);


historia = fopen(arqhist,"w");
//////////////////////////Escrita do cabe?lho do Arquivo de Saida//////////////////
fprintf(historia,"Arquivo de Saida: %s \n",arqANSYS);

fprintf(historia,"\nOPCOES DA OTIMIZACAO\n");

.
.
.


//////////////////////////INICIO DO LOOP DE OTIMIZA?O////////////////////////////

it=0;
itcontrole=0;
opcao = 1;
printf("Inicio da Otimiza•o");
while (opcao) {

        it = it+1;
        itcontrole=itcontrole + 1;

        fprintf(historia,"\n\nIteracao = %d \n",it);
        printf("\n\n\nITERACAO %d \n", it);

        //atribui a xold o valor de xnew
        for (i=1; i<=M; i++){
                xold[i] = xnew[i];
        }

        for (ii=1; ii<=ksize; ii++) {
                fb[ii]=fbext[ii];
        }

        for (ii=1; ii<=ksize; ii++) {
                for (i=1; i<=ksize; i++) {
                        K[ii][i]=0.0;
                }

        }

        ////////////////////// LOOP POR ELEMENTOS PARA O PROBLEMA K.U=F
//////////////////////
        for(k=1; k<=M; k++) {

                ////////////// CRIA MATRIZ DE RIGIDEZ LOCAL DO
ELEMENTO K//////////////
                for(i = 0; i < 4; i++){
                        xe[i] = coord[conect[k][i+1]][1];
                        ye[i] = coord[conect[k][i+1]][2];
                }
```

```
                                Ke(xe,ye,E,ni,Kel,xold[k],penalK);
                                ////////////////////////MATRIZ DE RIGIDEZ GLOBAL
K////////////////////////////////////
                                for(j = 1; j <=8; j++){
                                        q[j-1] = LM[k][j];
                                }
                                for (i=1; i<=8; i++) {
                                        if (q[i-1]!=0) {
                                                for (j=1; j<=8; j++) {
                                                        if (q[j-1]!=0) {
                                                                K[q[i-1]][q[j-1]] = K[q[i-
1]][q[j-1]] + Kel[(i-1)*8+(j-1)];
                                                        }
                                                }
                                        }
                                }
                                ////////////////////////////// CRIA VETOR DE FOR? LOCAL DO
ELEMENTO K ////////////////////
                                // Calculo da forca no n

        Fe(xe,ye,real_dens,g_x,g_y,omg_x,omg_y,omg_z,Fel,xold[k],penalF);
                                ////////////////////////////COMPOEM O VETOR DE FOR?
GLOBAL////////////////////////////////
                                for (i=0; i<=7; i++) {
                                        if (q[i]!=0) {
                                                fb[q[i]] =fb[q[i]]+Fel[i];
                                        }
                                }
                                // Calcula o volume total da estrutura
                                volume[it] = volume[it]+ Vol(xe, ye, xold[k],penalV);
                }////////////////////////FIM DO LOOP POR ELEMENTOS PARA O
PROBLEMA K.U=F ////////////////////////

                printf("Volume %f \n", volume[it]);
                printf("Volume / Volume Total %f \n", volume[it]/Vol_0);

                fprintf(historia,"Volume = %f \n",volume[it]);

                ////////////////////////////CALCULO DO DESLOCAMENTOS
UB////////////////////////////////

                //montagem da matriz esparsa

//              for(i=1;i<=NMAX;i++){
//                      ija[i]=0;
//                      sa[i]=0.0;
//              }

                dsprsin(K, ksize, THRESH, NMAX, sa, ija);

                //resolu?o do sistema K.UB = fb
                iter=0;
                err=0;
                linbcg(ksize, fb, ub, ITOL, TOL, ITMAX, &iter, &err);
                fprintf(historia,"Solucao do sistema linear: iter: %d, err: %f \n", iter,err);
```

```
                ////////////////////////CALCULO DE
FLEXIBILIDADE////////////////////////////////////
                compliancia[it]=0.0;
                for (ii=1; ii<=ksize; ii++)      compliancia[it] = compliancia[it] +
ub[ii]*fb[ii];


                printf("Flexibilidade %e \n",compliancia[it]);
                fprintf(historia,"Flexibilidade = %e \n",compliancia[it]);

                ////////////////////////GRADIENTE DA FUNCAO
OBJETIVO//////////////////////////

                for (k=1; k<=M; k++) gradF[k]=0;

                for (k=1; k<=M; k++) {

                        for (i=0; i<8; i++) {
                                tempmat[i]=0;
                        }
                        for(j = 1; j <=8; j++){
                                ube[j]=0.0;
                        }

                        //extrai de "ub" o vetor de deslocamento de cada elemento
                        for(j = 1; j <=8; j++){
                                if(LM[k][j] != 0){
                                        ube[j]=ub[LM[k][j]];
                                }
                        }

                        //calcula a derivada da compliancia de cada elemento

                        for(i = 0; i < 4; i++){
                                xe[i] = coord[conect[k][i+1]][1];
                                ye[i] = coord[conect[k][i+1]][2];
                        }

                        Ke(xe,ye,E,ni,Kel,1,1);
                        Fe(xe,ye,real_dens,g_x,g_y,omg_x,omg_y,omg_z,Fel,1,1);

                        dfdensF = penalF*pow(xold[k],(penalF-1));
                        dfdensK = penalK*pow(xold[k],(penalK-1));

                        // Calculo do gradiente
                        for (j=0; j<8; j++) {
                                for (i=0; i<8; i++) {
                                        tempmat[j]=tempmat[j] + ube[i+1]*Kel[(j*8)+i];
                                }
                        }

                        dcompl=0;


                        for (j=0; j<8; j++){
```

```
                                dcompl = dcompl + 2 * dfdensF * ube[j+1] * Fel[j] + -
dfdensK * ube[j+1]*tempmat[j];
                        }

                        gradF[k] = dcompl;

//                      dv=0.01;
                ////////////////////////////////ROTINA LP/////////////////////////////////////////

                for (i=1; i<=M; i++) {

                        if (sign1[i] > 0 && sign2[i] <= 0 && sign3[i] >= 0)
        ml[i] = ml[i] * minf;

                        else if (sign1[i] < 0 && sign2[i] >= 0 && sign3[i] <= 0)
        ml[i] = ml[i] * minf;
                        else
        ml[i] = ml[i] * msup;

                        if (ml[i] > mlupper)
        ml[i] = mlupper;
                        if (ml[i] < mllower)
        ml[i] = mllower;

                        xlower[i] = xold[i] * (1 - ml[i]);
                        xupper[i] = xold[i] * (1 + ml[i]);

                        if (xlower[i] < xmin[i]) xlower[i] = xmin[i];
                        if (xupper[i] > xmax[i]) xupper[i] = xmax[i];

                }

                if (filter_en) {

                        filter(M, N, coord, conect, xupper, filt_xupper, radius);
                        filter(M, N, coord, conect, xlower, filt_xlower, radius);

                }


        ////////////////////////////////ROTINA LP/////////////////////////////////////////
        ////////////////////////////////////////////////////////////////////////////////

                //Prepara?o dos dados para DSPLP (Programa?o Linear)

                        /* Rotina "DSPLP" que roda o LP em fortran
                        Recebe:
                        nvars  -> numero de variaveis de projeto
                        mrelas -> numero de restricoes
                        flag   -> minimizacao (0) ou maximizacao (1)
                        bu e bl -> limites moveis superior e inferior (arrays com nvars
posicoes)
                        costs  -> derivadas da funcao objetivo

                                Retorna:
                                info  -> flag de sucesso do LP
```

```
                densnew   -> nova distribuicao das variaveis de projeto
            */


        //copia as derivadas de "delF" para "costs" (input da rotina fortran)

        for (i=1; i<=M; i++) {
                costs[i] = gradF[i];
        }


        //Copia as informacoes de restricao lateral
        if (filter_en){
                for (i=1;i<=nvars;i++) {
                        bu[i]=filt_xupper[i-1];
                        bl[i]=filt_xlower[i-1];
                        ind[i]=3;
                }
        }
        else{

                for (i=1;i<=nvars;i++) {
                        bu[i]=xupper[i];
                        bl[i]=xlower[i];
                        ind[i]=3;
                }
        }
        //Significa que a variavel e maior ou igual ao limite inferior e menor ou igual
ao
        //limite superior


        // Copia o vetor com os valores das restricoes (b)
        // Aqui estamos considerando Ax<=b
        for (k=1;k<=mrelas;k++){
                bu[nvars+k] = Vfrac;
                ind[nvars+k] = 2;
        }

        //Copia dos valores de A para o vetor Ax<=b
        count=1;
for (i=1;i<=nvars;i++) {
                dattrv[count]=-i;
                dattrv[count+1]=1;
                dattrv[count+2]=area[i];
                count=count+3;
        }
                dattrv[count]=0;

        //Minimiza?o sem op?es
        prgopt[1]=1;
        prgopt[2]=1;


        //Dados auxiliares para dsplp
```

```
                //inicializa?o da variavel info

                info=1;

                //Chama o otimizador

                dsplp_( dusrmt_,&mrelas, &nvars, &costs[1], &prgopt[1], &dattrv[1],
                        &bl[1], &bu[1], &ind[1], &info, &primal[1], &duals[1],
                        &ibasis[1], &work[1], &lw, &iwork[1], &liw);

                //Resultado da otimiza?o (info>0 -> OK)



                if(info> =0){
                        printf("Otimizacao OK\n");
                }
                else
                        printf("Otimizacao Falhou\n");


                //Copia os valores de saida do otimizador
                if(info> =0){
                for (i=1;i< =nvars;i++)
                        xnew[i]=primal[i];
                }

                ////////////////////////////////////CONTROLE DO LOOP DE
OTIMIZA?O//////////////////////

                // Controle dos limites  moveis
                for (i=1; i< =M; i++) {

                        sign3[i] = sign2[i];
                        sign2[i] = sign1[i];
                        sign1[i] = xnew[i] - xold[i];

                }

                if (it == 1) {
                        complianciaold = compliancia[it];
                }

                else {
                        difer[it] = fabs((compliancia[it]-complianciaold)/complianciaold);
                        complianciaold = compliancia[it-1];
                }

                printf("\niteracao %d\t obj: %e\t vol: %f\t difer: %f\n", it, compliancia[it],
volume[it], difer[it]);
                printf("Status: PenalK %f\t  PenalF: %f\t PenalV: %f\t Filtro: %d\n", penalK,
penalF, penalV, filter_en);

                if (it>1){
```

```
                        if (difer[it]<=ITERTOL){
                                if(itcontrole<maxiterP1){
                                        itcontrole=maxiterP1;
                                        fprintf(historia,"Convergencia: Diferenca normalizada
= %f \n",difer[it]);
                                        printf("Convergencia: Diferenca normalizada = %f
\n",difer[it]);
                                }
                                else if (itcontrole<maxiterP2+maxiterP1){
                                        itcontrole=maxiterP2+maxiterP1;
                                        fprintf(historia,"Convergencia: Diferenca normalizada
= %f \n",difer[it]);
                                        printf("Convergencia: Diferenca normalizada = %f
\n",difer[it]);
                                }
                                else if(itcontrole<maxiterP3+maxiterP2+maxiterP1){
                                        itcontrole=maxiterP3+maxiterP2+maxiterP1;
                                        fprintf(historia,"Convergencia: Diferenca normalizada
= %f \n",difer[it]);
                                        printf("Convergencia: Diferenca normalizada = %f
\n",difer[it]);
                                }
                                else
if(itcontrole<maxiterFiltr+maxiterP3+maxiterP2+maxiterP1){

        //itcontrole=maxiterFiltr+maxiterP3+maxiterP2+maxiterP1;
                                        fprintf(historia,"Retirando instabilidade de tabuleiro
\n");
                                        fprintf(historia,"Convergencia: Diferenca normalizada
= %f \n",difer[it]);
                                        printf("Convergencia: Diferenca normalizada = %f
\n",difer[it]);
                                }
                                else
if(itcontrole<maxiterCont+maxiterFiltr+maxiterP3+maxiterP2+maxiterP1){

        itcontrole=maxiterCont+maxiterFiltr+maxiterP3+maxiterP2+maxiterP1;
                                        fprintf(historia,"Retirando escala de cinza \n");
                                        fprintf(historia,"Convergencia: Diferenca normalizada
= %f \n",difer[it]);
                                        printf("Convergencia: Diferenca normalizada = %f
\n",difer[it]);
                                }
                        }

                        if(itcontrole==maxiterP1){
                                penalK=2;
                                penalF=2;
                                penalV=1;
                                filter_en=0;
                                fprintf(historia,"MUDANCA DE  STATUS: MODO DA
CONTINUACAO \n");
                                output(SaidaP1,M,N,conect,coord,xmax,xmin,xold);

                        }
                        if(itcontrole==maxiterP2+maxiterP1){
```

```
                penalK=3;
                penalF=3;
                penalV=1;
                filter_en=0;
                fprintf(historia,"MUDANCA DE  STATUS: MODO DA
CONTINUACAO \n");
                output(SaidaP2,M,N,conect,coord,xmax,xmin,xold);
            }
            if(itcontrole==maxiterP3+maxiterP2+maxiterP1){
                penalK=3;
                penalV=1;
                penalF=3;
                filter_en=0;
                fprintf(historia,"MUDANCA DE  STATUS: MODO DA
CONTINUACAO \n");
                output(SaidaP3,M,N,conect,coord,xmax,xmin,xold);
            }
            if(itcontrole==maxiterFiltr+maxiterP3+maxiterP2+maxiterP1){
                penalK=3;
                penalV=1;
                penalF=3;
                filter_en=0;
                fprintf(historia,"MUDANCA DE  STATUS: MODO DA
CONTINUACAO \n");
                output(SaidaP4,M,N,conect,coord,xmax,xmin,xold);
            }

        if(itcontrole==maxiterCont+maxiterFiltr+maxiterP3+maxiterP2+maxiterP1){
                opcao=0;
                fprintf(historia,"FINAL DA OTIMIZACAO \n");
                output(SaidaP5,M,N,conect,coord,xmax,xmin,xold);
            }


        }
        if (it==MAXITER){
            opcao=0;
        }

        fclose(historia);
        historia = fopen(arqhist,"a");
    }

    ////////////////////////////FIM DO LOOP DE
OTIMIZA?O////////////////////////////////////////////
    // IMPRIME ARQUIVO

    fclose(historia);

//////////////////////GERA?O DOS VETORES PARA PLOTAGEM DE GRAFICOS//////////////////
////////////////////////////////////////////////////////////////////////////////


//////////////////////////CRIA?O DO ARQUIVO MATLAB////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
```

```
//Abre o arquivo em MATLAB
arquivo = fopen(arqMATLAB,"w");

//fun?o objetivo
fprintf(arquivo, "obj = [");
.
.
.vo MATLAB

////////////////////////////////////////////////////////////////////////////////////////////

//fornecimento de dados para arquivo em ANSYS

arq3=fopen(arqANSYS,"w");

fprintf(arq3, "/ClEAR\n/TITLE,Modelo Otimizado\n/PREP7\nET,1,42\n\n");
fprintf(arq3, "\nMP,EX,1,100\nNUXY=0.3\n\n");
for (i=1; i<=N; i++) fprintf(arq3, "N,%d,%f,%f\n", i, coord[i][1], coord[i][2]);

fprintf(arq3, "\n");
count=1;
for (i=1; i<=M; i++) {
        fprintf(arq3, "REAL,%d\n", count);
        fprintf(arq3, "E,%d,%d,%d,%d\n", conect[i][1], conect[i][2], conect[i][3],
conect[i][4]);
        count++;
}
.
.
.

fprintf(arq3, "\nFINISH");

fclose(arq3);

fclose (Auxiliar); // fecha o arquivo auxiliar para debbug

//////////////////////////LIBERA?O DE MEM?IA////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////

//libera?o de memria

        free_dvector(xmin,1,M);
        free_dvector(xmax,1,M);
        free_dvector(xnew,1,M);
.
.
.
```