Pontifícia Universidade Católica
do Rio de Janeiro

André Pimentel de Oliveira

Sistema Computacional p/ Otimização Topológica de Estruturas Bi dimensionais

Trabalho de Conclusão de Curso

Trabalho de conclusão de curso apresentada ao Programa de Graduação em Engenharia Mecânica do Departamento de Engenharia Mecânica da PUC–Rio como requisito parcial para obtenção do título de Bacharel em Engenharia Mecânica.

Orientador: Prof. Ivan de Menezes

Rio de Janeiro
Junho de 2016

Pontifícia Universidade Católica
do Rio de Janeiro

André Pimentel de Oliveira

Computational System to Topological Optimization of Bi dimensional Structures

Course's conclusion work presented by the program of graduation in Mechanical Engineering of the department of Mechanical Engineering in Scientific Technical Center of PUC-RIO with the partial requirement to obtain the title in Bachelor in Mechanical Engineering.

Professor Ivan de Menezes
Advisor
Department of Mechanical Engineering
PUC–Rio

June,
30th 2016

Agradecimentos

Aos meus pais, por todo apoio e incentivo ao longo da faculdade, a minha irmã, Natália, por estar sempre ao meu lado e a minha mãe, Marcia, por ser esta pessoa tão dedicada e especial em minha vida.

Ao meu orientador, Prof. Ivan Fabio Mota de Menezes, pela sua disponibilidade, bom humor e por ter sempre me incentivado e confiado em meu trabalho.

Aos meus amigos de faculdade, que com sua amizade verdadeira, transformaram mesmo os momentos mais difíceis, em grandes e boas recordações.

A minha equipe do Tecgraf pelo suporte e descontração.

E a Deus, por todas as muitas oportunidades e vitórias em minha vida.

Resumo

Pimentel, André; Menezes, Ivan. **Sistema Computacional p/ Otimização Topológica de Estruturas Bi dimensionais**. Rio de Janeiro, 2016. 34p. Trabalho de Conclusãoooo de Curso — Departamento de Engenharia Mecânica, Pontifícia Universidade Católica do Rio de Janeiro.

A otimização topológica se baseia na distribuição de material em um espaço, de forma que se gaste a menor quantidade deste para se produzir uma determinada peça. Definindo um domínio escolhido pelo projetista o desafio é achar a melhor distribuição de material de forma que possa ser solucionado o problema de minimização.

O objetivo deste trabalho é, através de códigos em MatLab e, juntamente a uma interface, projetar peças mecânicas que necessitem a menor quantidade de material possível gerando menores gastos em suas fabricações. Para isso serão apresentados parâmetros, explicados no decorrer deste trabalho, para serem definidos pelo usuário e com isso chegando ao melhor resultado.

O método usado para realização de cálculos será baseado na proposta de Bendsoe e Kikuchi (1988), isto é, considerar uma equação constitutiva homogenizada dependente apenas da densidade relativa do material. Também para a rotina de elementos finitos serão considerados para aproximação por quadrados de quatro nós.

Por fim ao final deste trabalho será apresentada uma análise comparando diferentes tipos de peças antes e depois da otimização ser feita, juntamente com o custo computacional e as informações de quão eficiente é o código desenvolvido.

Palavras–chave

; Otimização topológica; Suavização; Impressora 3D; Interface Gráfica.

Abstract

Pimentel, André; Menezes, Ivan (advisor). **Computational System to Topological Optimization of Bi dimensional Structures**. Rio de Janeiro, 2016. 34p. Conclusion course work — Mechanical Engineering Department, Pontifícia Universidade Católica do Rio de Janeiro.

The topological optimization is based on the distribution of material in a space, in a way that it will waste the less amount of this to yield a particular piece of a machine, a car, etc. By defining a domain chosen by the draftsman the challenge now is to find what's the best distribution of material trying to solve the problem of minimization.

The objective of this work is, by the help of MatLab codes, and, jointly with an interface, make projects of mechanical pieces that will consume the lesser amount of material, generating the fewest waste in his fabrication. To do this, it will be presented parameters, that will be explained in this work, to be defined by the user and, by doing this, we will get the best results.

The method used to make all the calculations will be based on the thesis of Bendsoe and Kikuchi (1988), that is, it will be considered an equation constitutive homogenised that will depend only on the relative density of the material. Also to the finite elements routine, it will be considered approaches of squares with four nodes.

In the end of this work it will be presented an analyses comparing different kinds of mechanical pieces before and after the code of optimization, jointly with the computational cost and other information of how useful the code developed is.

Keywords

; Topological Optimization ; Suavization ; Printer 3D ; Graphic Interface; .

Sumário

Lista de Figuras

1
Introdução

Um dos grandes dilemas da engenharia é: como projetar o melhor sistema possível, com a menor quantidade de material e o menor gasto, que consiga suportar todos os carregamentos em questão? Por causa disso cada vez mais e mais vem-se procurando técnicas para se conseguir reduções no tempo de projeto, sistemas que necessitem de materiais mais baratos e que garantam segurança, ou seja, um produto com maior funcionalidade, e esta é a parte mais difícil de todo o processo de produção.

Uma das ferramentas que ajudaram, e ainda estão ajudando o desenvolvimento destas estruturas, é o computador. Graças a este, houve um aumento considerável em pesquisas relacionada à análise estrutural, o que gerou métodos computacionais muito eficazes como o método dos elementos finitos, que será utilizado neste trabalho.

Mesmo com esses avanços computacionais não é suficiente criar um código que consiga calcular e gerar um melhor sistema, mas sim criar um código que projete um sistema corretamente e de maneira rápida, isto é, precisa-se de um sistema eficiente, distinto e que gere um melhor custo-benefício, ou seja, mesmo com uma máquina que não possua processadores de ultima geração, o código deve funcionar de maneira rápida e eficaz. E para isso ser possível o engenheiro deve procurar sempre empregar ferramentas analíticas, numéricas que permitam buscar soluções ótimas de maneira sistemática.

A teoria da otimização de estruturas permite que estes custos operacionais sejam reduzidos devido ao aumento de performance dos sistemas cria-

dos. Este processo pode ser definido como a obtenção de valores máximos ou mínimos de determinada função.

Nas próximas seções deste trabalho, serão mostrados cálculos de como é possível fazer estas reduções de material de maneira rápida e prática.

2
Formulação teórica

Quando se fala em otimização topológica, o primeiro tipo de estrutura que se imagina é a viga bi apoiada, e ela será usada como exemplo para a realização da formulação teórica. Neste caso se sabe que há dois apoios em ambas as extremidades e será adicionado uma força em seu ponto central, conforme mostrado na imagem.

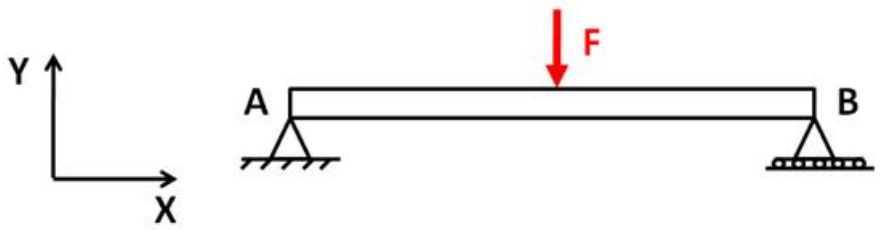

Figura 2.1: Viga bi apoiada com uma força no centro

Como já foi dito anteriormente, o foco dos problemas de otimização é achar a melhor distribuição de material, de forma a obter a menor *compliance* (flexibilidade), com uma limitação da quantidade total de material. Nesta formulação o domínio será considerado retangular e discretizado por elementos finitos quadrados, pois assim a numeração dos elementos e nós será mais simples. E ainda, o número de elementos na direção horizontal será denominado nelx e de elementos na direção vertical nely.

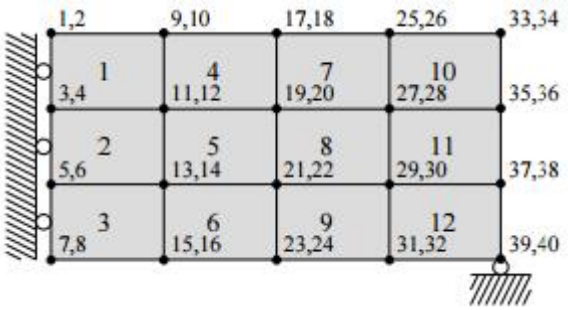

Figura 2.2: Domínio retangular com 12 elementos

A equação para minimizar a *compliance* da estrutura é dada por:

$$min_x \rightarrow c(x) = U^T KU = \sum_{e=1}^{N} (x_e)^p u_e^t k_0 u_e \tag{2-1}$$

Considerando:

$$\begin{cases} \frac{V_x}{V_0} = f \\ KU = F \\ 0 < x_{min} \leq x \leq 1 \end{cases} \tag{2-2}$$

Onde:

U	Deslocamento global
F	Vetor de forças
K	Matriz de rigidez global
u_e	Vetor dos elementos de deslocamento)
k	Matriz de rigidez
x	Vetor,que possui as variáveis de projeto
x_{min}	Vetor com as densidades relativas mínimas (diferente de 0 para evitar matriz singular)
N	Número de elementos usados para discretizar o volume do domínio projetado (nelx x nely)
P	Fator de penalização (normalmente p = 3)
V(x)	Volume do material
V_0	Volume desejado
f	A fração de volume final (0 <f <1)

Para resolver este problema será usado o método de *Optimality Criteria* (OC) pelo fato dele ser o mais simples. De acordo com Bendsoe (1995), pode-se fazer um regime de alterações para as variáveis de projeto de forma que:

$$x_e^{novo} = \begin{cases} max(x_{min}, x_e - m), & \text{se } x_e B_e^{\eta} \leq max(x_{min}, x_e - m) \\ x_e B_e^{\eta}, & \text{se } max(x_{min}, x_e - m) < x_e B_e^{\eta} \leq min(1, x_e + m) \\ min(1, x_e + m), & \text{se } min(1, x_e + m) \leq x_e B_e^{\eta} \end{cases} \tag{2-3}$$

$$B_e = -\frac{\frac{\partial c}{\partial x_e}}{\lambda \frac{\partial V}{\partial x_e}} \tag{2-4}$$

$$\frac{\partial c}{\partial x_e} = -p(x_e)^{p-1} u_e^T k_0 u_e \tag{2-5}$$

Onde:

m	Posição que delimita o movimento
η	Coeficiente de amortecimento (=1/2)
B_e	Valor encontrado a partir da equação (4)
λ	O multiplicador lagrangiano que pode ser calculado por um algorítimo de bi-seção

Para garantir a existência de uma solução que minimize a equação (2.1), Sigmund and Petersson (1998) afirmam que é necessária à existência de algum tipo de restrição no problema. Neste caso será implementada uma técnica de filtro, onde após todos os cálculos serem realizados, uma imagem preta e branca será feita de modo que as regiões pretas possuem material e as brancas não. Esta formulação mostrará como a estrutura deve ficar para que a redução de material seja a desejada pelo usuário, porém ela apresentará regiões cinzas, pois, na teoria, deve-se ter um valor de densidade $\rho = 1$ para regiões que devem ter material e densidade $\rho = 0$ para regiões que não devem ter material. Estes espaços possuem valores de densidade entre zero e um, o que não é possível de ser produzido, por isso, após a otimização ser realizada, o código irá passar um filtro na imagem para que não ocorra este problema.

Para fazer está distinção de onde deve e não deve ter material, o filtro utilizado irá modificar os elementos de sensibilidade da seguinte forma:

$$\frac{\partial c}{\partial x_e} = \frac{1}{x_e \sum_{f=1}^{N} x_f \frac{\partial c}{\partial x_f}} \tag{2-6}$$

$$H_f = r_{min} - dist(e,f), \left\{ f \epsilon N \mid dist(e,f) \leq rmin \right\}, e = 1,2,...,N \qquad (2\text{-}7)$$

Onde dist(e,f) é a distância entre o centro do elemento **e**, e o centro do elemento **f** e o coeficiente de convolução *Hf* possui valor zero fora da área em que o filtro está sendo aplicado. O coeficiente de coonvolução decresce linearmente como mostrado na equação (2.7) quando se aumenta a distância do elemento **f**.

Com essas informações já é possível, primeiramente, desenvolver um código que possa fazer a otimização topológica de uma determinada estrutura.

3
Implementação do código

Antes de começar a falar sobre o código, é preciso à apresentação de alguns comandos em Matlab que não são muito conhecidos, pois são especificamente feitos para o criador de interface MATLAB GUI. Estes serão mais utilizados quando for explicado o código da interface criada.

getappdata	Recupera o valor de uma variável definida em outro arquivo Matlab
setappdata	Cria uma variável global que pode ser acessada em qualquer arquivo Matlab

Sabendo-se disso, foi criado um programa baseado na formulação teórica apresentada na seção 2. Assim como linguagens C++, Java é possível criar um programa com diferentes funções que recebem alguns parâmetros. Baseado nessa ideia, o programa inicial será chamado de *otimcc.m*. Ele será responsável pela otimização do sistema, que recebe os seguintes parâmetros do usuário (todos os códigos estarão disponíveis na seção ANEXO deste trabalho):

nelx	É o número de elementos na direção horizontal
nely	Também já explicado na formulação teórica, é o número de elementos na direção vertical
volfrac	O volume final desejado, mostrado na equação (2.2) da formulação teórica.
penal	O fator de penalização
rmin	O tamanho do filtro, quanto maior seu valor, melhor a imagem, porém mais lento será o programa.

O sistema padrão usado será metade de uma viga bi apoiada com uma força localizada no meio (esta poderá ser alterada, e a explicação de como isto é feito, será apresentada), desta forma há uma simetria nas condições de contorno.

No começo do programa (linhas 1-55) há a implementação do arquivo principal, que irá fazer uso de todas as funções para o cálculo da redução de material. Antes de entrar na rotina de elementos finitos são adicionadas duas variáveis: *loop* e *change* (linhas 9 e 10), de tal forma que *loop* representará a quantidade de iterações necessárias para se chegar no volume final desejado e change o critério de parada dos cálculos, ou seja, as iterações irão parar caso as alterações do domínio possuam valor menor que 1

Dentro da condição de parada a rotina de elementos finitos retorna o vetor deslocamento (vetor U), e sabendo que a matriz rigidez para materiais sólidos é igual para todos os elementos sua chamada é feita apenas uma vez. Em seguida é feito um *loop* em todos os elementos (linhas 20-37) para determinar a função objetiva e sensibilidades. As variáveis n1 e n2 (linhas 22 e 23) representam os valores dos nós superior esquerdo e direito, respectivamente, dentro do número total de nós definidos pelo usuário e são usados para obter-se o valor do vetor deslocamento de um elemento (vetor Ue) do vetor deslocamento global U.

A análise de sensibilidade é feita através do filtro de malhas independentes (linha 39) e do critério de *Optimality Criteria optimizer* (linha 40). No final os valores do *compliance* e o número de iterações é salvo para usos futuros, se necessário ao usuário.

É importante ressaltar que, dependendo do número de forças aplicadas ao sistema, os vetores força e deslocamento serão definidos. Onde este número X é a quantidade de forças aplicadas (será explicado mais a frente), e a função "getappdata(0,'nforcas') irá recuperar o valor do número de forças fornecido pelo usuário.

3.1 Funções auxiliares

Como dito anteriormente, este programa faz uso de algumas funções auxiliares que são chamadas pelo programa principal, e nesta seção serão explicadas a importância de cada uma e o que elas irão fazer.

◦ Função OC (linha 59)

A atualização das variáveis é feita através desta função. Como o volume de material do sistema decresce com a função do multiplicados Lagrangiano, o valor do mesmo que irá satisfazer a limitação de volume pode ser encontrado através do método da bisseção.

Este método consiste em dividir o intervalo escolhido de forma iterativa (intervalo a,b). Para verificar se a raiz está contida na primeira ou na segunda metade do intervalo inicial, é utilizado o teorema de Bolzano. Em seguida, o processo é repetido para aquela metade que contém a raiz de f(x) = 0, ou seja, aquela em que a função, y = f(x), tem valores numéricos com sinais opostos nos seus extremos. Em cada iteração os valores dos valores de a ou b são alterados, dependendo de onde a raiz se encontra, de forma que

$$x = \frac{a+b}{2} \tag{3-1}$$

O processo tem fim quando se obtém um valor de tamanho menor ou igual a um definido, porém algumas vezes a escolha dos pontos [a,b] não irão chegar a resposta, o que fará com que a iteração nunca acabe, por isso no programa principal foi escolhido também um valor máximo de iterações para que isso não ocorra.

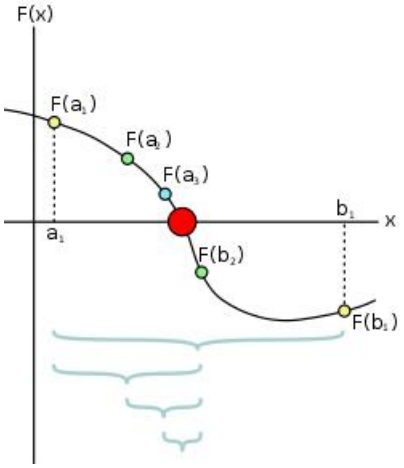

Figura 3.1: Método da bisseção, onde os limites são a1 e b1 e a raís b2

Sabendo do funcionamento deste método supõem-se um limite inferior e superior l1 e l2, respectivamente, os quais são analogamente, ao exemplo da figura (3.1) [a_1=l1,b_2=l2], para o multiplicador lagrangiano. Este processo terá fim quando o valor de l1 – l2 for menor que 0,0001, lembrando que há a possibilidade de nunca se chegar a este critério, por isso é necessário definir um valor máximo de iterações para evitar que o programa nunca tenha fim.

◦ Função check (Linha 71)

Observando a função, percebe-se que nem todos os elementos do domínio escolhido são procurados com finalidade de encontrar os que estão dentro do raio (rmin), mas apenas os elementos quadrados que possuem lado igual ou maior que o raio, ou seja, esta função tem a finalidade de pegar os elementos do sistemas e fazer um filtro para escolher onde deve e não deve ter material.

◦ Função FE (Linha 87)

É a função que realiza a rotina de elementos finitos. A matriz global de rigidez é criada a partir de um *loop* sobre todos os elementos do sistema, assim como, no caso da função principal que possuía as variáveis n1 e n2, que eram

usadas para se colocar os elementos da matriz de rigidez, nos locais corretos da matriz de rigidez global.

Ambos nós e elementos, são colunas numeradas da esquerda para a direita (ver figura (1.1) na seção introdução) e ainda, cada nó possui dois graus de liberdade, nas direções verticais e horizontais (como é feita a aplicação das forças será explicado após a apresentação desta ultima função). Os suportes aplicados à estrutura são implementados eliminando-se graus de liberdade fixo da equação linear.

◦ Aplicação das forças

No arquivo principal as linhas 24 e 28 possuem dois tipos diferentes de problemas:

◦ Caso haja apenas uma força sendo aplicada em todo o sistema (linha 24)
◦ Caso exista mais de uma força aplicada ao sistema (linha 30)

Quando se tem mais de uma força aplicada ao sistema, os vetores força e deslocamento devem ser definidos como um vetor de X colunas (já dito anteriormente), ou seja, primeiramente a função FE deve ser alterada de forma que o tamanho dos vetores força (F) e deslocamento (U) passem de (2*(nely+1)*(nelx+1),1), caso de uma força aplicada, para (2*(nely+1)*(nelx+1),getappdata(0,'nforcas')), para mais de uma força aplicada onde getappdata(0,'nforcas')) é o número de forças fornecidas pelo usuário, isto é, esta alteração diz que a função agora é a soma de X *compliances*:

$$c(x) = \sum_{i=1}^{x} U_i^T K U_i \tag{3-2}$$

Isso irá explicar os dois casos apresentados no começo deste tópico. Caso o número de forças seja igual a um, a matriz de rigidez será (a,1), caso contrário é necessário criar um comando 'for' que irá aplicar todas as forças passadas para a matriz de rigidez, ficando da forma (a,X).

3.2 Exemplos e utilização do programa

Agora que foi apresentado como o programa funciona, já é possível mostrar exemplos com explicações de seu significado. Para começar, será usado um exemplo clássico da mecânica, uma viga bi apoiada com uma força aplicada em seu centro, de tal forma que o volume final seja 40% do volume inicial ($\frac{V_f}{V_i}$=0,4) e com uma malha de 100 elementos na direção x e 20 na direção y conforme a figura:

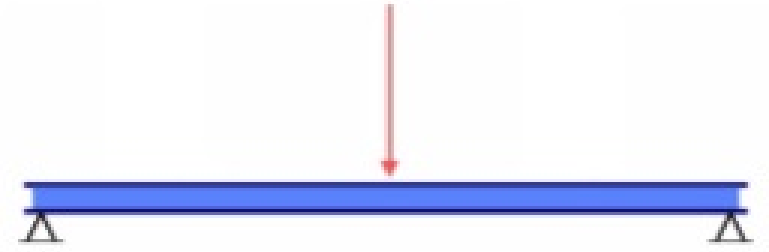

Figura 3.2: Viga bi-apoiada com uma força em seu centro

Realizando a otimização com e sem o filtro teremos:

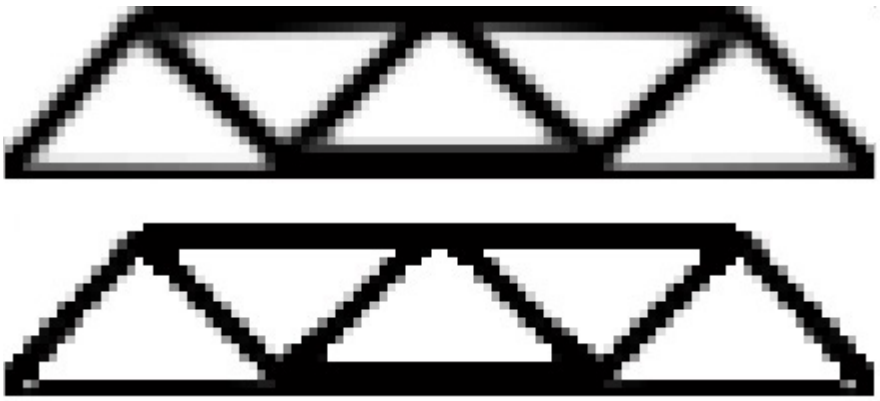

Figura 3.3: Otimização com e sem filtro

Destas duas imagens geradas pode-se tirar algumas informações. A primeira é que, como já foi dito na introdução foi gerado uma imagem com

tons de cinza. A região branca é onde não há necessidade de ter material (densidade igual a zero), normalmente em casos reais se adiciona um material mais barato para completar esta região. A região preta é onde há a necessidade de material, há vários tons do mesmo, devido ao fato de não haver um filtro na primeira imagem, diferentemente da segunda que possui um filtro que mostra claramente estas regiões.

Segundo, com essa estrutura a redução dos custos se tornará relativamente alta, por exemplo: Um aço SAE 1010 com valor entre R$350 R$800 por tonelada poderia sofrer uma redução drástica de preço, se fosse usado apenas 40% do material e com garantia de que a estrutura irá aguentar o carregamento. Para ficar mais claro, um exemplo da vida real pode ser um sinal de trânsito:

Figura 3.4: Suporte para sinais de trânsito

Se for utilizada a otimização topológica com o intuito de reduzir o material utilizado para suportar estes três sinais, com pesos considerados iguais, para 40% do inicial tem-se:

Figura 3.5: Otimização do suporte de sinais de trânsito com 40% de seu volume inicial

Para que essas imagens fossem geradas, foi necessário fazer várias alterações no código para cada um dos casos mostrados, e para que este método se torne mais convencional, uma interface foi criada a fim de tornar isto possível. Na próxima seção será mostrado o objetivo deste trabalho, juntamente a um pequeno exemplo para que o usuário não tenha nenhuma dúvida de como utilizar o programa.

4
A interface gráfica

4.1
Apresentação

O que a interface gráfica deste trabalho irá fazer é, utilizar o códigos de otimização para que o usuário possa entender, de maneira clara, como funciona a otimização topológica e, observar dados que irão mostrar os resultados e algumas informações referentes aos códigos.

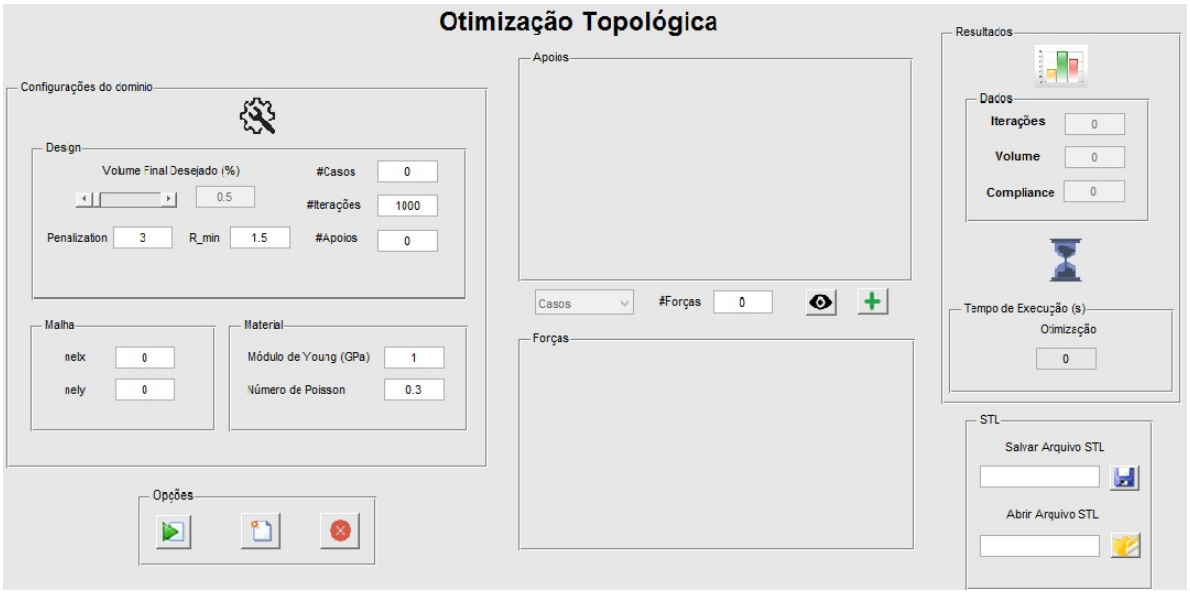

Figura 4.1: Interface grafica

Como pode ser visto na figura (4.1), o programa é dividido em seis seções que são:

◦ Configurações de domínio: onde o usuário irá escolher o tamanho da malha, propriedades referentes ao material, número de casos de força que estão agindo no sistema, pontos de fixação e o número de iterações máximo para evitar que o código possa entrar em um eventual *loop* infinito.

◦ Forças: onde após o preenchimento do número de forças, e a escolha do caso, uma janela irá aparecer nesta seção para que o usuário preencha em que posição do nó (X,Y) se encontram as mesmas e a intensidade de cada uma. (será mostrado na próxima seção).

◦ Fixação: analogamente as forças o usuário irá escolher os nós que estão fixos e se permitem deslocamento em x, y ou ambos.

◦ Opções: onde se encontram os botões de: Otimizar, Novo Projeto e Sair.

◦ Resultados: após ser feita a otimização, os valores da *compliance*, iterações necessárias, tempo de otimização e volume final serão exibidas, sendo que esta ultima apenas informará ao usuário que o necessitado foi realizado.

◦ STL: neste local será salvo um arquivo .STL para ser usado em impressoras 3D, também disponível uma opção para visualizar o mesmo.

Como foram utilizadas vários códigos diferentes, na pasta onde se encontra o arquivo há uma divisão para evitar qualquer dificuldade de compreensão, mostrada a seguir:

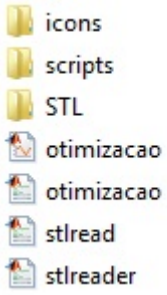

Figura 4.2: Ordenação do programa

Onde:

- icons: é a pasta em que se encontra os ícones do programa
- scripts: onde estão os códigos matlab
- STL: onde serão salvas as peças com formato .STL
- otimizacao: é o arquivo principal que irá rodar o programa.

4.2 Utilização do programa

O primeiro passo que o usuário deve seguir é preencher os campos referentes a Malha, Material e Design. Algum destes campos estão por *default*, valores do aço, pois é um dos materiais mais utilizados na engenharia, caso o usuário deseje, estes podem ser alterados.

Na malha deve-se escolher o número de elementos na direção X e Y, lembrando que os elementos são quadrados, neste exemplo será usado 100 elementos em X e 20 em Y, e lembrando que o número de nós total da malha é dado por 2(X+1)(Y+1), neste exemplo teremos 2(100+1)(20+1)=4242 nós.

Em design escolheremos o volume final que desejamos, 0.5 (50% do volume inicial), apenas um caso de força, dois pontos de fixação e no máximo mil iterações. Deste modo a janela ficará da seguinte maneira:

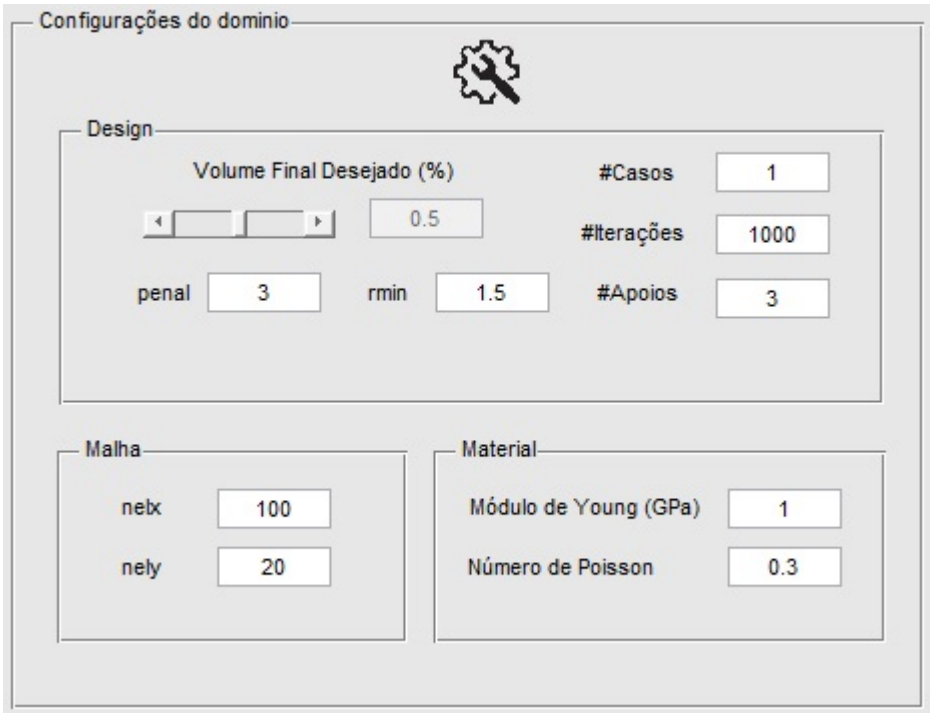

Figura 4.3: Exemplo de configurações de domínio

Após esta janela ser preenchida, a seções fixações agora apresentarão uma janela correspondente ao número escolhido pelo usuário e a janela PopUp agora terá o tamanho igual ao número de casos. Uma pequena observação, o MatLab possui um valor máximo de valores que podem ser passados que equivale a 3.000.000 (3.10^6), um valor mais do que aceitável, porém interessante de ser informado.

Variando o número de fixações esta janela irá deletar ou criar novos campos, dependendo do que o usuário decidir. No caso do número de casos o PopUp irá fazer a mesma coisa, aumentar ou diminuir seus itens.

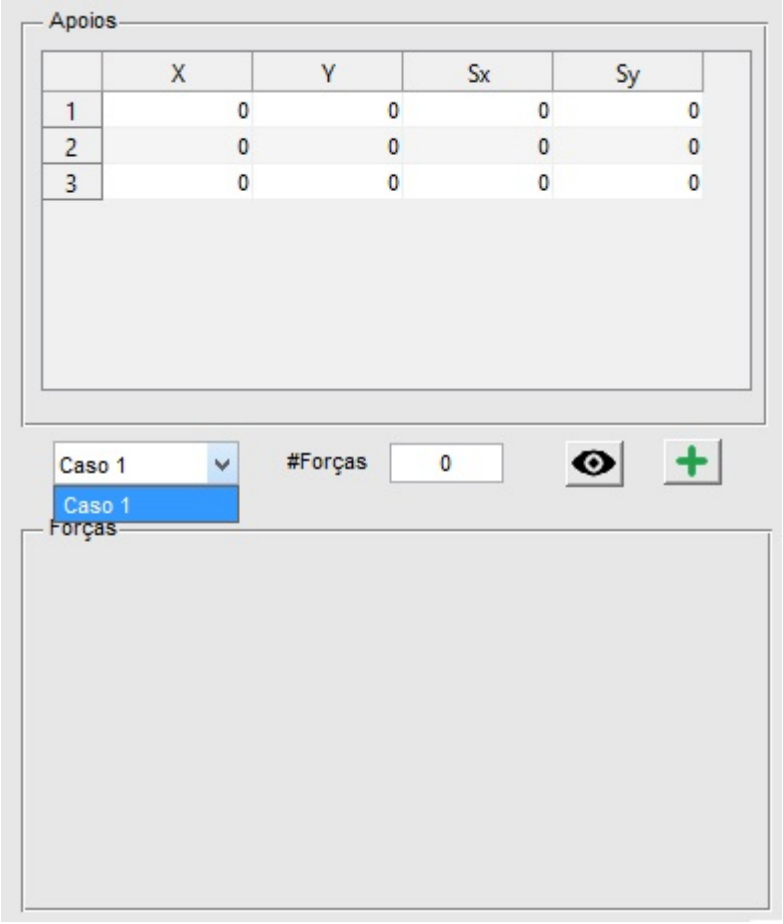

Figura 4.4: Janela de Forças e Fixações após estas serem definidas

Continuando o exemplo, a viga irá ter um apoio em seu centro, um em seu canto inferior direito e outro no canto inferior esquerdo, ambos não permitirão deslocamento nem em X nem em Y. Uma força atuará no ponto médio entre os apoios centrais e da direita e a outra entre os apoios centrais e da esquerda, também no ponto médio conforme a figura (4.5).

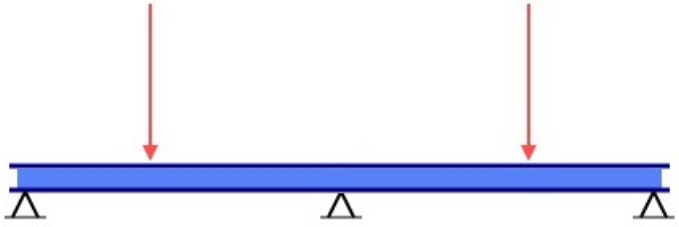

Figura 4.5: Viga com três apoios e duas forças atuantes

Para montarmos esta estrutura fixaremos o nó da ponta esquerda (X=0, Y=0, Sx=1, Sy=1), o nó da ponta direita (X=100, Y=0, Sx=1, Sy=1) e o do centro (X=50, Y=0, Sx=1, Sy=1)

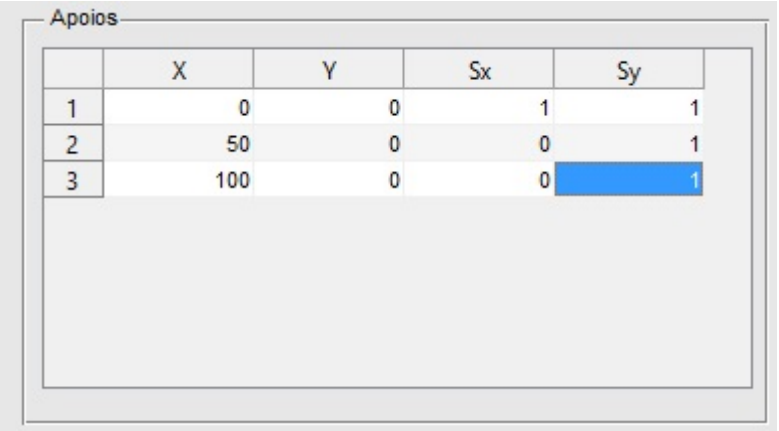

Figura 4.6: Apoios preenchidos

Para escolher as forças atuantes no sistema o usuário deve seguir os seguintes passos:

Figura 4.7: Preenchimento das forças atuantes para o caso N

1 → Escolher o número de forças para o caso N.

2 → Selecionar o caso N para que a janela de forças seja criada para o mesmo.

3 → Escolher pra cada força as coordenas X e Y do nó em que a mesma está sendo aplicada e as intensidades nas direções Fx e Fy.

4 → Clicar em adicionar para que as forças sejam salvas na memória do programa.

Caso o usuário queira ter certeza de que as forças e apoios estão nos lugares corretos, basta clicar no botão visualizar (ícone do olho) que a seguinte figura irá aparecer:

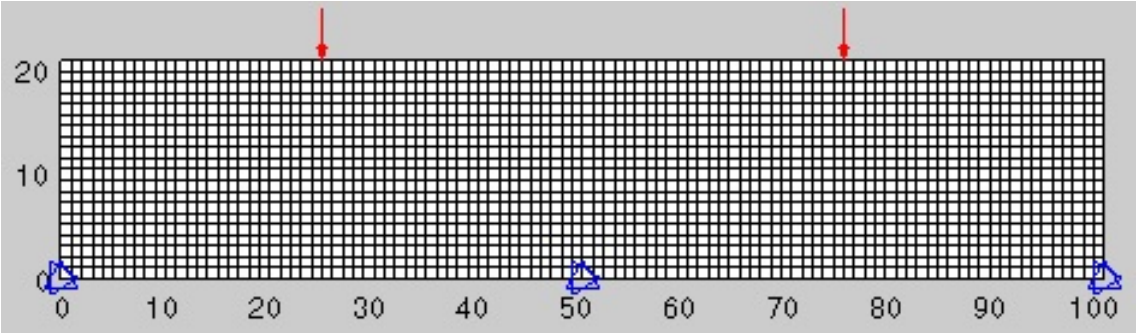

Figura 4.8: Malha com as forças e apoios nos pontos escolhidos

Para o exemplo utilizado as forças estão atuando corretamente entre os pontos de fixação e, este útimo, também está no lugar certo. Caso exista outros casos de força, basta troca-lo no PopUp e escolher a opção visualizar que a mesma será mostrada corretamente.

Por fim, com todos os blocos corretamente preenchidos basta clicar em Otimizar na seção Opções que será iniciada a otimização da peça. Uma barra de carregamento será mostrada para que o usuário esteja ciente do fim do programa.

Figura 4.9: Início e fim da otimização

Após o fim da otimização os campos da seção resultados serão preenchidos com as informações do programa.

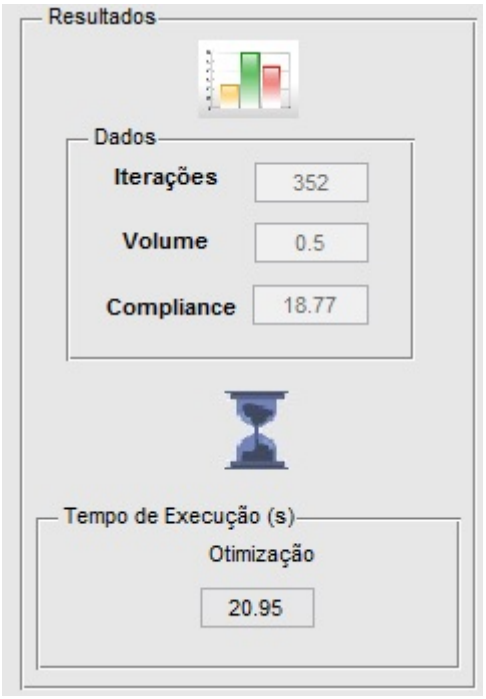

Figura 4.10: Resultados

Como a malha do exemplo não é muito refinada, o tempo de otimização foi relativamente baixo. Se o mesmo for muito grande (por exemplo 2000x500) o tempo de execução se tornará relativamente alto, algo importante de se ter em mente.

Por fim, caso o usuário queira criar um arquivo para imprimir em uma impressora 3D basta ir na seção STL, escolher o nome do arquivo a ser salvo e clciar em salvar. Para visualizar basta fazer o mesmo, porém no campo abrir.

Figura 4.11: Opção STL

Visualizando:

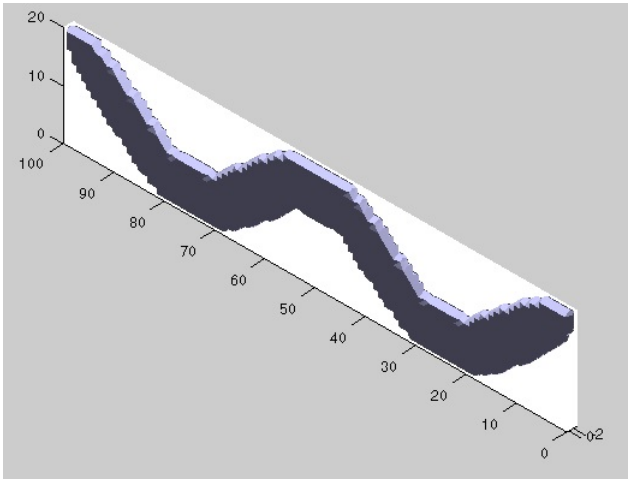

Figura 4.12: Visualização do STL

Várias peças diferentes podem ser criadas, dependendo de como as condições de contorno e as forças estão agindo sobre a mesma. Na próxima seção mais alguns exemplos serão mostrados rapidamente.

5
Outros exemplos

5.1
Exemplo 1

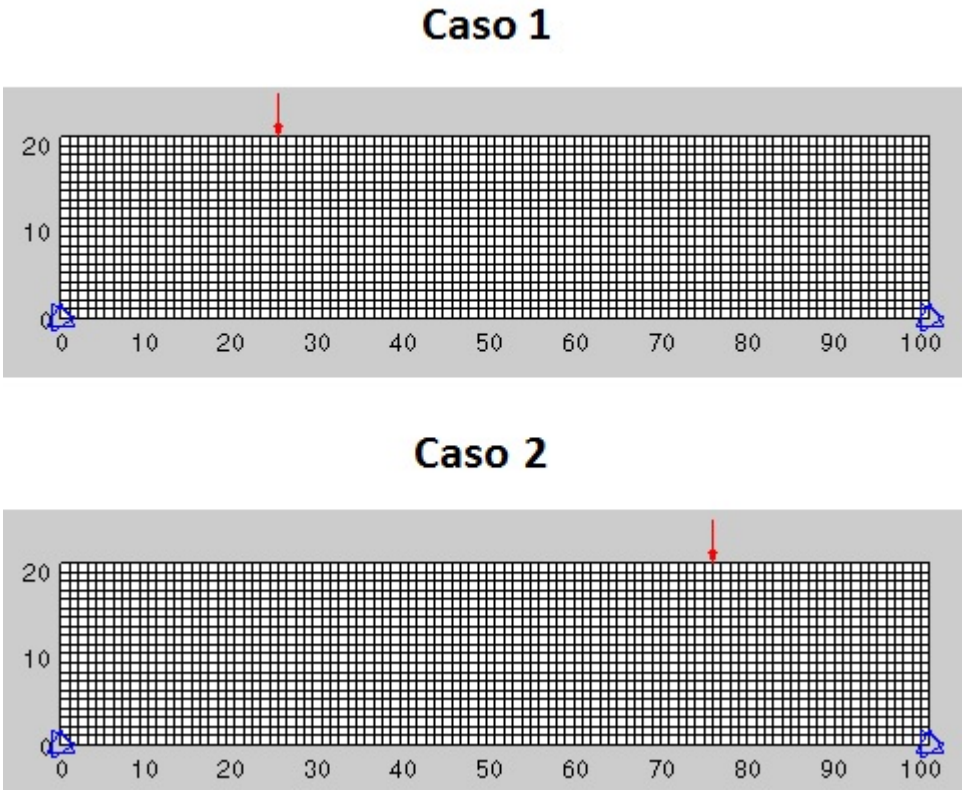

Figura 5.1: Ex 1. Forças atuantes

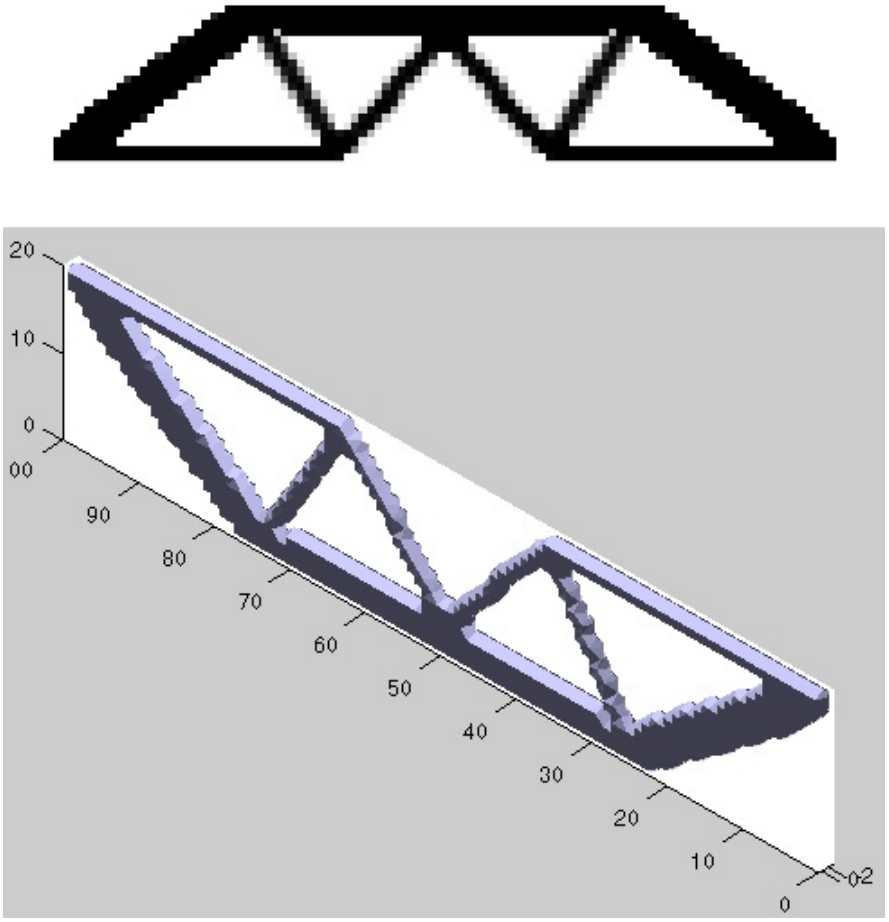

Figura 5.2: Ex 1. Otimização e peça 3D

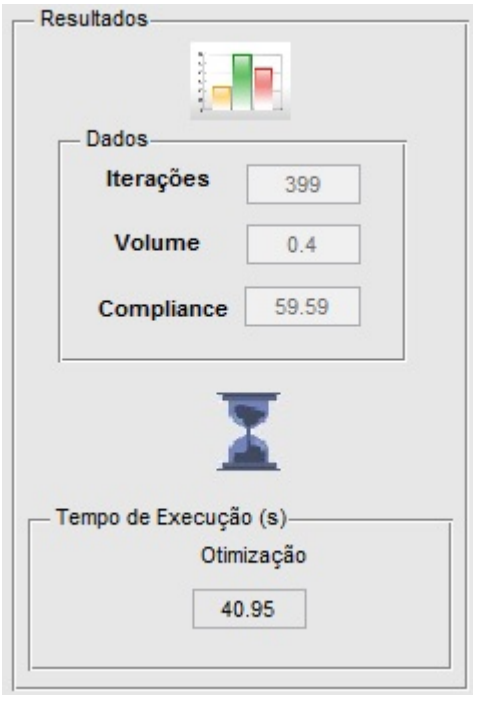

Figura 5.3: Ex 1. Resultados

5.2 Exemplo 2

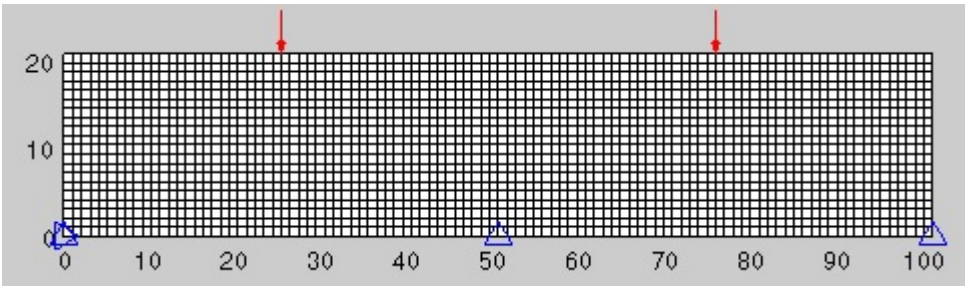

Figura 5.4: Ex 2. Forças atuantes

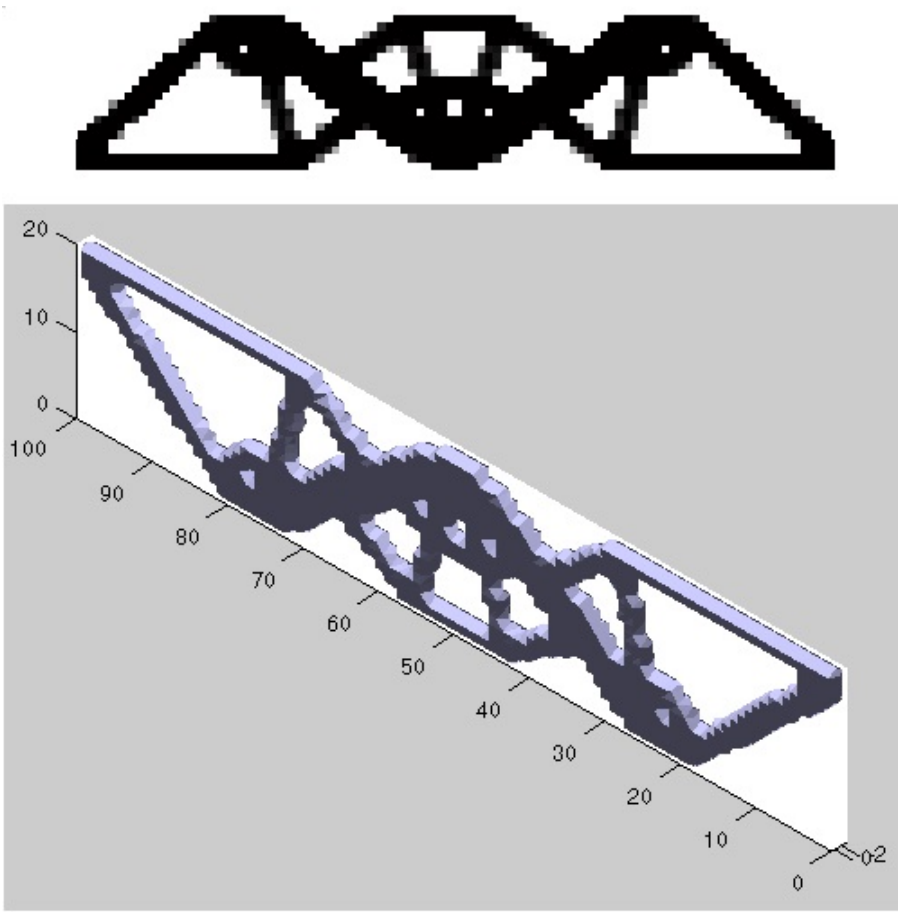

Figura 5.5: Ex 2. Otimização e peça 3D

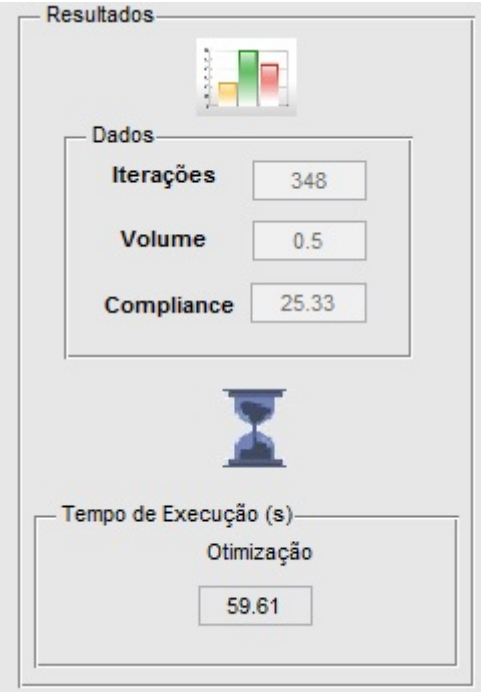

Figura 5.6: Ex 2. Resultados

5.3 Exemplo 3

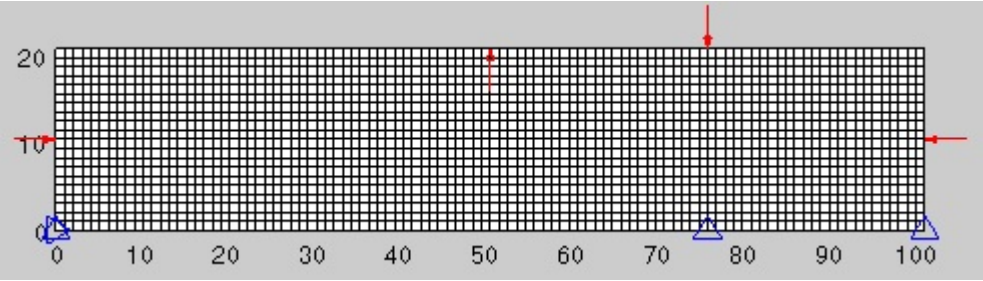

Figura 5.7: Ex 3. Forças atuantes

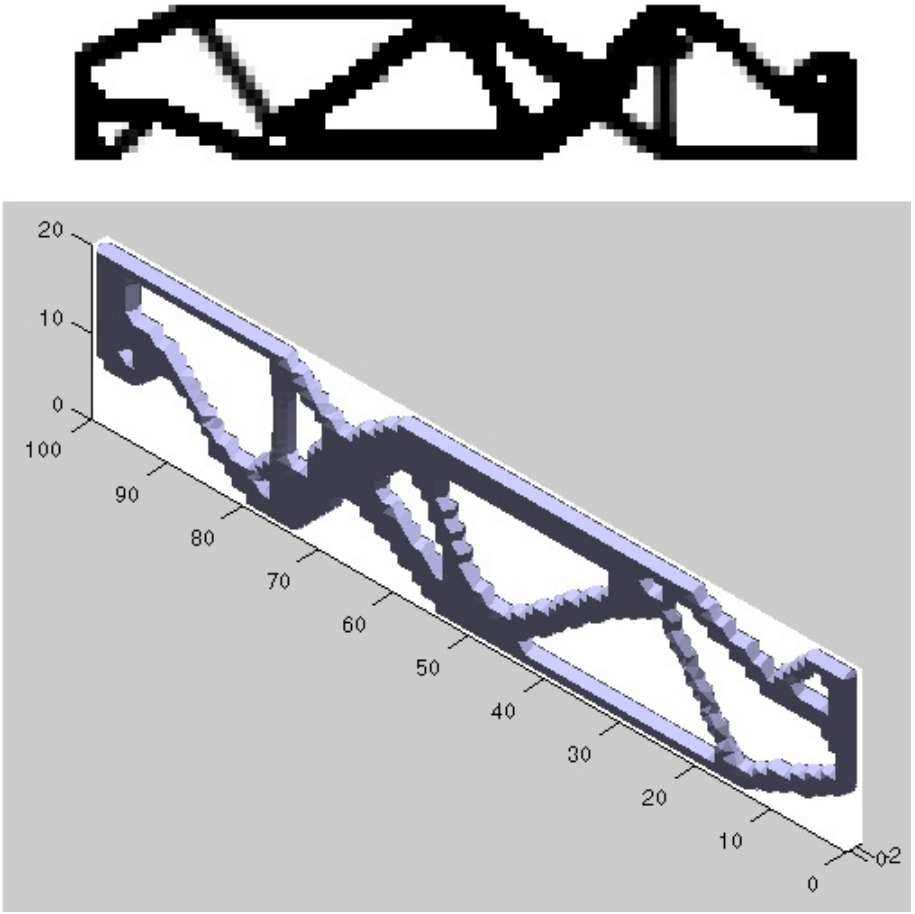

Figura 5.8: Ex 3. Otimização e peça 3D

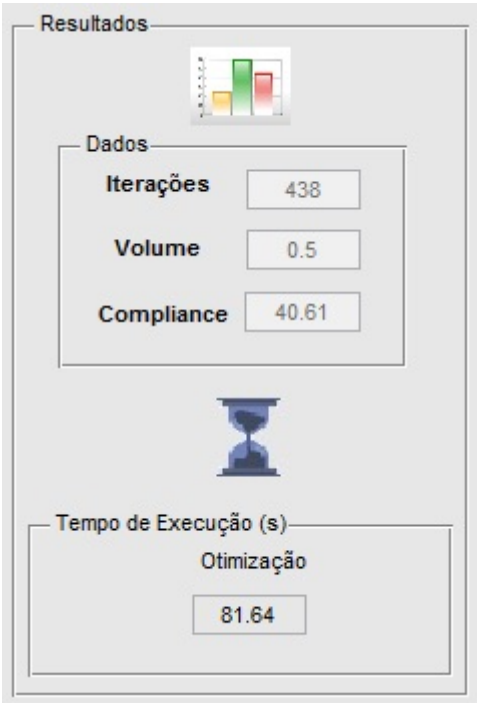

Figura 5.9: Ex 3. Resultados

6
Conclusão

Para pessoas que estão começando a estudar sobre a otimização topológica, o paper de Ole Sigmund (que foi utilizado na teoria deste trabalho) é um excelente guia de introdução. Com a utilização de uma interface gráfica, o entendimento de como o código funciona se torna muito mais claro.

Embora a criação de interfaces gráficas no Matlab se mostre menos interessante do que em programações como C++, Fortran, Java, dentre outros, ele é um excelente programa para possoas que possuem menos experiência nesta área. Para os que já possuem mais conhecimento sobre estes, as funções e lógicas do MatLab são similares as dos outros programas citados.

Quanto mais elementos a malha possui, mais refinado será o resultado, porém como foi visto, mais requisitado o computador será, o que torna a demanda de códigos mais rápidos e de maior eficiência.

Para trabalhos futuros sugere-se a implementação de um código de suavização, assim como a modificação em algumas linhas de código para que as funções utilizadas não apareçam multiplas vezes sem esta necessidade. Por exemplo, não realizar o mesmo cálculo caso haja a possibilidade da definição de uma variável global.

Referências Bibliográficas

[1] Haftka, Raphael T., - *"Structural and Multidisciplinary Optimization"* , Ed. Springer, 2011.

[2] Liu, Kai; Tovar Andrés., *"An efficient 3D topology optimization code written in Matlab"* , October 2013, p. 1-21.

[3] MatLab File Exchange
`https://www.mathworks.com/matlabcentral/fileexchange/`

[4] MathWorks, - *"Creating GUI with GUIDE"*
`https://www.mathworks.com/`

[5] Sigmund, Ole.; Bendsoe, M.P., - *"Topology Optimization - Theory, Methods and Applications"* , Ed. Springer, 2003.

A
Anexos

O codigo para o código 99 Lines escrito por Ole Sigmund pode ser acessado pelo link link:
`http://www.topopt.dtu.dk/files/matlab.pdf`

O codigo para o código 88 Lines escrito por Erik Andreassen, Anders Clausen, Mattias Schevenels, Boyan S. Lazarov e Ole Sigmund pode ser acessado pelo link:
`http://www.topopt.dtu.dk/files/TopOpt88.pdf`