UNIVERSIDADE DE SÃO PAULO ESCOLA POLITÉCNICA
DEPARTAMENTO DE ENGENHARIA MECÂNICA

Software Para Projeto de Mecanismos Flexíveis Tridimensionais Por Otimização Topológica.

Vitor Bellissimo Falleiros

Orientador: Prof. Dr. Emílio Carlos Nelli Silva.

São Paulo
2003

UNIVERSIDADE DE SÃO PAULO ESCOLA POLITÉCNICA
DEPARTAMENTO DE ENGENHARIA MECÂNICA

Software Para Projeto de Mecanismos Flexíveis Tridimensionais Por Otimização Topológica.

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

Vitor Bellissimo Falleiros

Orientador: Prof. Dr. Emílio Carlos Nelli Silva.

Área de Concentração:
Engenharia Mecânica

São Paulo
2003

FICHA CATALOGRÁFICA

Falleiros, Vitor Bellissimo
Software para projeto de mecanismos flexíveis tridimensionais pelo método da otimização topológica,
por V. B. Falleiros. São Paulo : EPUSP, 2003.
59P. + anexos

Trabalho de formatura – Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecânica.

1. Mecanismos 2. Otimização 3. Elementos Finitos
I. Falleiros, Vitor II. Universidade de São Paulo. Escola Poliécnica. Departamento de Engenharia Mecânica

Dedico este trabalho ao *Prof. Dr. Ivan Gilberto Sandoval Falleiros*, meu pai, que tanto me incentivou durante minha formação.

Agradeço ao *Prof. Dr. Emílio Carlos Nelli Silva* pela oportunidade e orientação neste trabalho e ao *Prof. Dr. Alberto Hernandez Neto* pelo apoio durante o curso todo.

Agradeço também aos meus colegas de curso, especialmente *André Izecson de Carvalho*, *Luís Augusto de Motta Mello*, *Taís Felipelli* e *Tiago Naviskas Lippi*, bem como a *Cícero Ribeiro de Lima*, cujos trabalhos precederam o meu na mesma área e cuja ajuda me foi de grande valia.

RESUMO

Diferente dos mecanismos tradicionais, os mecanismos flexíveis são constituídos geralmente de uma peça única, e seu funcionamento se baseia na sua deformação elástica, e não no movimento relativo entre diversas peças.

Esse tipo de mecanismo tem um vasta aplicação em campos que exijam precisão e/ou sistemas de tamanho reduzido. Pode-se citar como exemplo os cabeçotes de leitura de discos de computador, aparelhos de medição como transdutores piezo-elétricos e a área biomédica, onde a peça única facilita a esterilização de instrumentos.

O projeto desse tipo de sistema é uma tarefa bastante complexa, e até recentemente era sempre realizada por métodos de tentativa e erro. Com a difusão do conceito nas tecnologias modernas, abordagens sistemáticas começaram a ser propostas para o problema.

A razão deste trabalho é criar um software que realize o projeto de um mecanismo flexível otimizado de forma sistemática, utilizando-se do método dos elementos finitos e de conceitos modernos de otimização topológica.

Na otimização topológica, a preocupação é descobrir onde deve existir massa para aumentar a rigidez do sistema da forma mais eficiente possível (menor massa e maior rigidez). Ou seja, as variáveis de projeto da otimização serão densidades atribídas a cada elemento do MEF.

Dadas as funções de um mecanismo flexível, duas propriedades se apresentam claramente como essenciais: a rigidez da estrutura como um todo e a flexibilidade para permitir os movimentos desejados. O mecanismo flexível deverá prover máxima deformação, em um ponto e direção bem definidos quando sujeito a certos esforços, e ao mesmo tempo apresentar certa rigidez.

Para realizar essa análise, utiliza-se uma formulação matemática em termos de energia, dividida em dois conceitos básicos: a energia mútua e a flexibilidade média. A primeira está diretamente relacionada com a geração dos movimentos desejados nos pontos especificados, e deve ser maximizada. A segunda representa o inverso da rigidez estrutural do mecanismo e deve, portanto, ser minimizada.

ABSTRACT

Where traditional mechanisms work with relative movement between different parts, compliant mechanisms, usually composed of one single piece, work with elastic deformation of its structure.

This kind of mechanism has a vast application in fields that require precision and/or reduced size systems. For example, computer hard disk reading systems, measuring apparatus like piezo-electric transducers and even biomedicine, where the single piece structure is easier to sterilize than a complex mechanism.

However, the design for this kind of system is still a complex job, and, until recently, was done only by try-and-error. As the concept was more and more used in modern technology, systematic approaches were developed to the problem.

The reason for this work is to create a software that systematically designs an optimized compliant mechanism, working with the finite element method and modern concepts of topology optimization.

In topology optimization, the question posed is where should mass be concentrated to raise the structure´s stiffness in most efficient way (less mass, more stiffness). The design variables are, then, densities assigned to each element from the FEM discretization.

Given the functions of a compliant mechanism, two properties present themselves clearly as essential: structure´s stiffness as a whole, and the flexibility to perform the desired movements. The mechanism shall provide maximal movement in a chosen point and direction, when submitted to a specified effort, and, at the same time, it shall present some stiffness.

To perform this analysis, a mathematical formulation was developed in terms of energy, dividing itself in two basic concepts: mutual energy and mean compliance. The former is directly related to the desired movement generation in the specified points of the structure, and should be maximized. The later represents the inverse of the structure´s overall stiffness and should, therefore, be minimal.

SUMÁRIO

RESUMO

ABSTRACT

LISTA DE FIGURAS

LISTA DE SÍMBOLOS

1. INTRODUÇÃO 12

2. OBJETIVO 16

3. FUNDAMENTOS TEÓRICOS 17

3.1. O Método dos elementos finitos 17

3.1.1. Especificações dos elementos finitos implementados 18

3.2. Otimização topológica 22

3.3. Mecanismos flexíveis 26

3.3.1. Flexibilidade média 26

3.3.2. Energia mútua 27

4. FORMULAÇÃO MATEMÁTICA 29

4.1. Notação matricial 29

4.2. Função objetivo 30

4.3. Análise de sensibilidades 33

5. IMPLEMENTAÇÃO NUMÉRICA 34

5.1. Fluxograma 34

5.2. Modelo de entrada 34

5.3. Leitura de dados 35

5.4. Elementos finitos 35

5.4.1. Matrizes esparsas indexadas 37

5.5. Limites móveis 39

5.5.1. Filtro de malha 40

5.6. Otimização 43

5.6.1. Programação linear 43
5.6.1.1. Simplex 44
5.6.1.2. DSPLP 44
5.6.1.3. Linearização da função objetivo 44
5.6.2. Critério da optimalidade 45
5.7. Visualização de resultados 46

6. RESULTADOS 47
6.1. Validação 47
6.1.1. Descrição da análise realizada 47
6.1.2. Resultados esperados 48
6.1.3. Resultados obtidos 48
6.2. Resultados tridimensionais 49
6.2.1. Mecanismo de impressão 49
6.2.2. Pinça 53

7. COMENTÁRIOS E CONCLUSÕES 57
7.1. Continuidade do trabalho 57

LISTA DE REFERÊNCIAS 58

ANEXO A Exemplo de modelo de domínio para o software *AnSys* (mecanismo inversor) 60

ANEXO B Exemplo de arquivo *database* utilizado como entrada (mecanismo inversor) 61

ANEXO C Listagem parcial do programa 63

ANEXO D Exemplo de arquivo de saída a ser lido no *AnSys* (mecanismo inversor) 66

LISTA DE FIGURAS

Nº	Legenda	Página
1	Mecanismos clássicos: biela-manivela e alicate convencional	12
2	Mecanismo flexível equivalente ao alicate na figura acima	12
3	Tipos de mecanismos flexíveis	13
4	Arco e flecha: exemplo histórico de mecanismo flexível	14
5	O alicate de pesca: exemplo moderno de mecanismo flexível	14
6	Exemplo de domínio discretizado para análise por MEF	18
7	Elemento hexaédrico de 8 nós	19
8	Exemplo de otimização de barra apoiada pelas extremidades	24
9	Forma ótima para a rigidez no problema da viga bi-apoiada	24
10	Definição da estrutura analisada	27
11	Definição da estrutura para mecanismo flexível	27
12	Definição da energia mútua desejada	31
13	Flexibilidades médias que devem ser minimizadas	32
14	Fluxograma do programa implementado	34
15	Diversos tipos de matrizes esparsas	37
16	Limites móveis impedem que função linearizada se distancie demais da original	39
17	Exemplo de formação de tabuleiro de xadrez	40
18	Raio de abrangência do filtro	41
19	Domínio estudado	47
20	Resultado esperado na análise realizada	48
21	Resultado obtido com a análise tridimensional do mesmo problema	48
22	Comparação entre os resultados	49
23	Domínio para mecanismo de impressão	50
24	Resultado obtido para mecanismo de impressão	50
25	Curvas de convergência das funções analisadas para o mecanismo de impressão: função objetivo, volume (soma das pseudo-densidades), energia mútua e flexibilidades médias	51
26	Tensões internas (Von Mises) na estrutura deformada pelo esforço fornecido F_1	52

27	Sobreposição est. def. com a forma original, destacando o movimento obtido	53
28	Domínio para projeto da pinça	53
29	Pinça tridimensional otimizada, vista 1	54
30	Pinça tridimensional otimizada, vista 2	54
31	Curvas de convergência das funções analisadas para a pinça: função objetivo, volume (soma das pseudo-densidades), energia mútua e flexibilidades médias	55
32	Análise da deformação da pinça, comparada à forma original	56

LISTA DE SÍMBOLOS

σ	Vetor de tensões internas no elemento
ε	Vetor de deformações internas no elemento
E	Módulo de Young ou Módulo de Elasticidade
ν	Coeficiente de Poisson
K_e	Matriz de rigidez do elemento
B	Matriz de forma do elemento
D, D_0	Matriz de elasticidade do material
J	Jacobiano da transformação isoparamétrica
ξ, η, ς	Sistema local de coordenadas
D_D	Matriz de elasticidade – parcela relativa à deformação normal
D_V	Matriz de elasticidade – parcela relativa à variação volumétrica
D_{yz}	Matriz de elasticidade – parcela relativa à deformação por cisalhamento no plano yz
D_{zx}	Matriz de elasticidade – parcela relativa à deformação por cisalhamento no plano zx
D_{xy}	Matriz de elasticidade – parcela relativa à deformação por cisalhamento no plano xy
D(x), D_i	Tensor de elasticidade do elemento *x* (ou *i*)
$\rho(x)$, ρ_i	Pseudo-densidade do elemento *x*
p	Fator de penalização
t^i	Esforço externo aplicado à estrutura
u^i	Campo de deslocamentos gerado no corpo pelo esforço t^i
$L^i(u^i)$, L_{ii}	Flexibilidade média da estrutura, correspondente ao esforço t^i
Γ	Domínio da estrutura no espaço
$L^i(u^j)$, L_{ij}	Energia mútua da estrutura, ligada ao esforço gerador t^i e ao deslocamento gerado u^j
$\varepsilon(v)$	Tensor de deformações correspondente ao campo de deslocamentos *v*
$[u_i]$	Vetor de deslocamentos gerados pelo esforço t_i
[K]	Matriz de rigidez global da malha de elementos finitos
F	Função objetivo
w	Coeficiente de influência de cada parcela da função objetivo
r_{max}	Raio de abrangência do filtro espacial
r_{ij}	Distância entre o centro dos elementos *i* e *j*
s_j	Distância entre elementos *i* e *j*, relativa a r_{max}

$\bar{s}$	Média das distâncias relativas dos vizinhos *j* ao elemento central *i*
nv	Número de vizinhos de um elemento cuja distância é menor que r_{max}
ρ_i^{min}, ρ_i^{max}	Limites móveis inferior e superior para a pseudo-densidade do elemento *i*
V_i	Volume do elemento *i*
N	Número de variáveis de uma função linear estudada
x_1, x_2... x_N	Variáveis de uma função linear estudada
a_{01}, a_{02}... a_{0N}	Coeficientes de uma função linear estudada
z	Função linear estudada
a_{i1}, a_{i2}... a_{iN}	Coeficientes de funções de restrição às variáveis da função linear estudada
m_1, m_2, m_3	Quantidades de funções de restrição às variáveis, de acordo com a forma
Λ	Operador definido para o critério da optimalidade
B	Fator utilizado na otimização

1. Introdução

Define-se *mecanismo* como um sistema mecânico capaz de transformar e/ou transferir movimentos, esforços ou energia, de uma dada forma disponível para uma dada forma desejada. Como a energia se conserva no processo, o esforço obtido pode ser muito maior que o fornecido, mas os deslocamentos serão menores. A figura a seguir mostra alguns mecanismos clássicos.

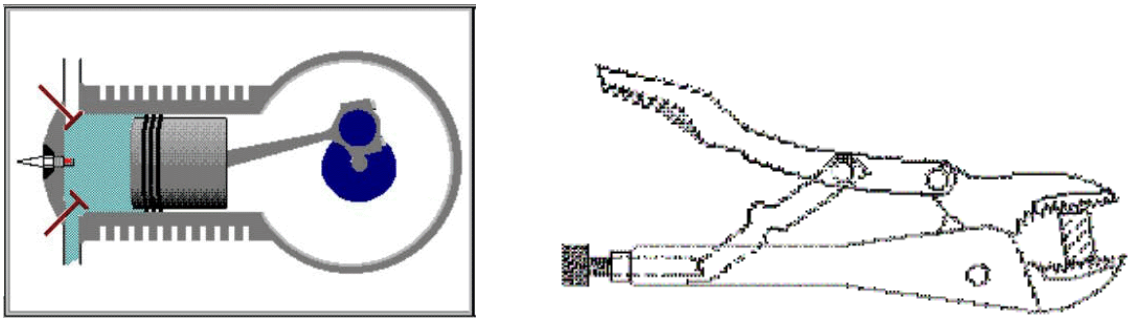


Figura 1 – Mecanismos clássicos: biela-manivela e alicate convencional

Diferente dos mecanismos tradicionais, os mecanismos flexíveis são constituídos geralmente de uma peça única, e seu funcionamento se baseia na sua deformação elástica, e não no movimento relativo entre diversas peças.

Enquanto nos mecanismos convencionais a flexibilidade dos componentes costuma ser um problema, o conceito dos mecanismos flexíveis está justamente em aproveitar essa flexibilidade, eliminando assim uma fonte de imprecisão e erros, proporcionando óbvias vantagens para o campo da mecânica de precisão.

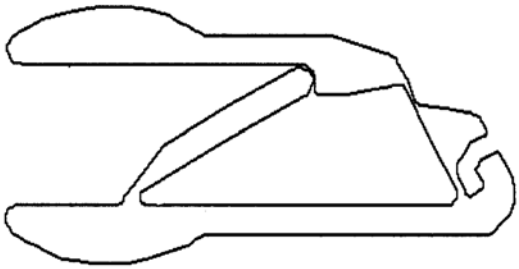


Figura 2 – Mecanismo flexível equivalente ao alicate na figura acima

Pode-se dividir os mecanismos flexíveis em duas classes distintas: mecanismos com flexibilidade distribuída ao longo da peça e mecanismos com pontos bem definidos de flexibilidade concentrada, chamados de pivôs. A diferença pode ser vista no exemplo dado pela figura a seguir:

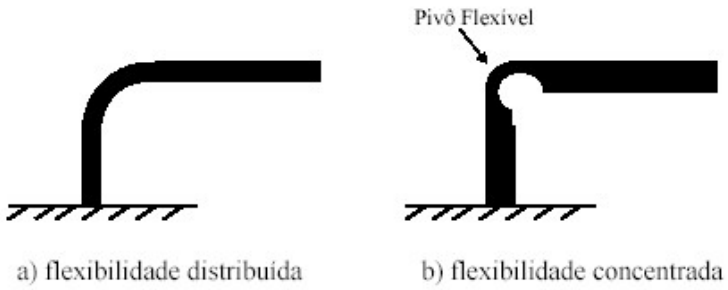


Figura 3 – Tipos de mecanismos flexíveis

Além das vantagens em precisão advindas da eliminação de juntas e articulações, a construção dos mecanismos flexíveis em uma única peça possibilita sua fabricação em dimensões menores que os mecanismos tradicionais. Pelo mesmo fator, podem-se também apreciar a redução de desgaste e de peso do sistema como um todo.

A adoção de mecanismos flexíveis pode também reduzir o custo de um projeto, ao diminuir o número de peças fabricadas, simplificar a montagem final, e, ainda, eliminar a necessidade de lubrificação entre peças. A construção em um único elemento possibilita também um aumento na eficiência do sistema, do ponto de vista energético, visto que as forças de atrito internas do material são sempre muito menores do que forças de atrito seco entre peças diversas.

Devido ao seu funcionamento, os mecanismos flexíveis armazenam energia durante seu funcionamento, na forma de energia de deformação. Dessa forma, eles podem incorporar o princípio de funcionamento das molas convencionais, armazenando energia propositadamente, para uso posterior. Essa utilização é bem exemplificada pela utilização de arco e flecha. A energia fornecida pelo braço do arqueiro é armazenada no arco quando este puxa a corda. Ao soltá-la, essa mesma energia é transferida à flecha, na forma de energia cinética.

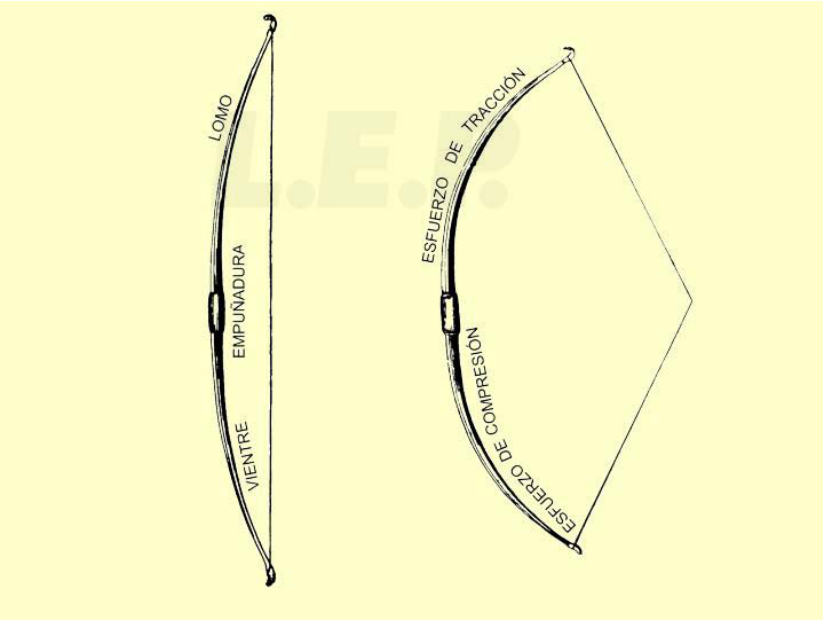


Figura 4 - Arco e flecha: exemplo histórico de mecanismo flexível

Esse tipo de mecanismo tem um vasta aplicação em campos que envolvam mecânica de precisão ou que exijam sistemas de tamanho reduzido. Pode-se citar como exemplo os cabeçotes de leitura de discos de computador, aparelhos de medição como transdutores piezo-elétricos, máquinas fotográficas e até mesmo a área biomédica, onde o fato de ser uma peça única torna-se ainda mais vantajoso ao facilitar a esterilização dos instrumentos. A possibilidade de redução de peso faz desse tipo de sistema bastante atraente para a indústria aeronáutica.

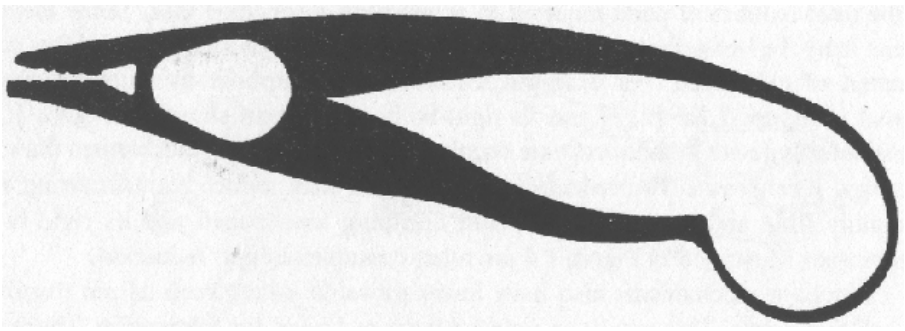


Figura 5 – O alicate de pesca: exemplo moderno de mecanismo flexível

Entretanto, os mecanismos flexíveis apresentam seus próprios problemas. Um dos mais críticos é a limitação da operação, em geral, a pequenos deslocamentos. Além disso, a fabricação de peças pequenas com precisão ainda é um desafio técnico. Outra desvantagem é a impossibilidade realizar rotações completas com mecanismos flexíveis.

Contudo, se, por um lado, o armazenamento de energia pode ser uma vantagem, por outro ele pode se apresentar como uma desvantagem. A energia armazenada na deformação de um sistema pode representar uma parcela da energia fornecida que não estará disponível na saída.

2. Objetivo

É provável que o maior desafio dos mecanismos flexíveis seja a sua criação. O projeto desse tipo de sistema é uma tarefa bastante complexa, e até recentemente era sempre realizada por métodos de tentativa e erro. Com a difusão do conceito nas tecnologias modernas, abordagens sistemáticas começaram a ser propostas para o problema.

Em especial, a utilização cada vez maior de computadores na área de projetos, permitindo a difusão de técnicas que exigem grande poder matemático, como o método dos elementos finitos, leva a abordagens mais sistemáticas desse desafio.

As técnicas de otimização também têm se aprimorado nas últimas décadas, mais uma vez aproveitando o aumento constante da capacidade computacional disponível.

Com isso, apareceu a possibilidade de desenvolvimento de algoritmos para a otimização de estruturas. Com os parâmetros corretos, pode-se dirigir esses algoritmos para a criação de mecanismos otimizados.

E essa é a proposta deste trabalho: desenvolver um software que, baseado em metodologias modernas de otimização, forneça a melhor forma para um mecanismo especificado, possibilitando assim o projeto sistemático de mecanismos flexíveis tridimensionais. O método dos elementos finitos provou ser uma ferramenta essencial nesse estudo.

3. Fundamentos utilizados

3.1. O método dos elementos finitos

Diversos problemas da engenharia envolvem equações de altíssima complexidade matemática, que muitas vezes só podem ser resolvidas de forma analítica para casos extremamente específicos, através da imposição de condições nem sempre desejáveis.

Para contornar situações desse tipo, foram criados métodos numéricos de resolução aproximada de equações complexas. Entre eles está o método dos elementos finitos (MEF), cada vez mais difundido como ferramenta de base entre os engenheiros.

Os conceitos envolvidos no MEF são bastante simples, mas a resolução de equações com precisão razoável costuma exigir uma capacidade computacional altíssima, envolvendo cálculos com matrizes da ordem de 100.000 elementos e até mais. Assim, esse método começou a se popularizar com o advento dos microprocessadores.

O MEF consiste em discretizar o domínio estudado (uma estrutura, um volume de fluido, etc), ou seja, dividi-lo em partes menores, chamadas de elementos finitos, e calcular, iterativamente, as grandezas desejadas nas extremidades dessas divisões, chamadas de nós. Esse cálculo é, em geral, mais simples do que o cálculo para o problema todo. A seguir, os valores são interpolados dentro de cada elemento finito, e obtém-se um resultado aproximado para todo o domínio. Sabe-se que o resultado fornecido não é exatamente igual ao real, mas existem muitas ferramentas para controlar esse erro.

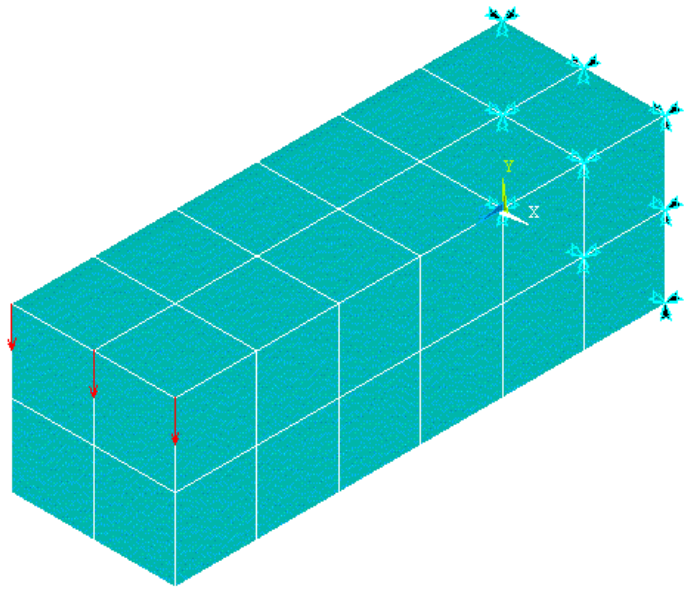


Figura 6 – Exemplo de domínio discretizado para análise por MEF

De acordo com a formulação do problema, os elementos finitos aplicados podem ser de diversas dimensões e tipos: barras, triângulos ou quadriláteros, tetraedros ou hexaedros, apenas para citar os mais utilizados. Pode-se ainda ter nós intermediários, para aumentar o grau de interpolação dos resultados dentro dos elementos, e assim se aproximar mais dos valores verdadeiros. Isso aumenta, contudo, a complexidade da implementação do método. Caso um resultado não seja satisfatório, pode-se ainda aumentar a discretização, ou seja, repartir o domínio em elementos menores.

Para reduzir a carga computacional da análise, recomenda-se sempre considerar condições que podem simplificar o problema. O maior exemplo disso são condições de simetria, sempre buscadas por aqueles que estão acostumados a trabalhar com modelos pesados do ponto de vista computacional.

3.1.1. Especificações dos elementos finitos implementados

Neste trabalho são utilizados elementos hexaédricos de 8 nós, ou seja, sem nós intermediários, como exemplificado abaixo.

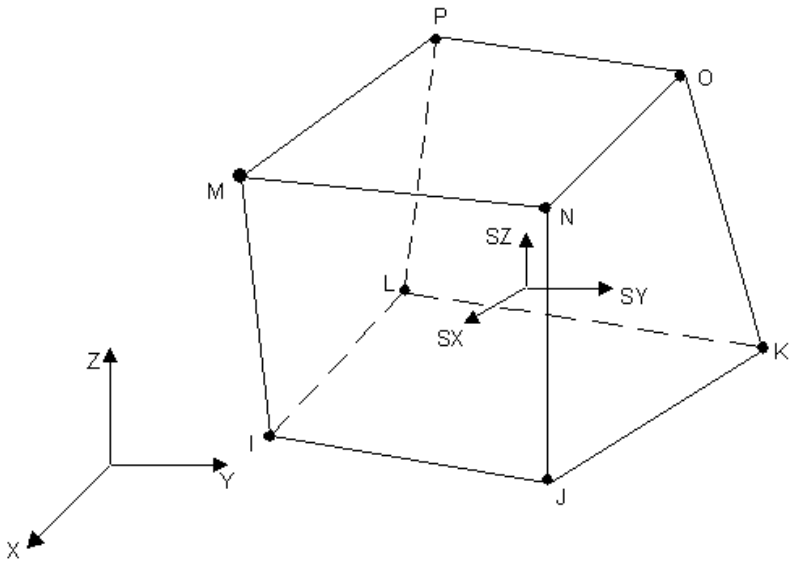


Figura 7 – Elemento hexaédrico de 8 nós

Além disso, utiliza-se uma formulação isoparamétrica, que garante a possibilidade do cálculo mesmo com grandes deformações nos elementos (LIPPI, [10]), e aumenta a probabilidade de convergência dos resultados.

No problema estudado, deve-se construir a matriz de rigidez de cada elemento. Para isso, foi utilizado o método da integração por redução seletiva. Segue o desenvolvimento matemático da matriz de rigidez de um elemento.

Assumindo que o material utilizado seja isotrópico, pode-se escrever:

$$\sigma = \begin{Bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{yz} \\ \tau_{zx} \\ \tau_{xy} \end{Bmatrix} = D \cdot \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{yz} \\ \gamma_{zx} \\ \gamma_{xy} \end{Bmatrix} = D \cdot \varepsilon$$

Onde:

- **σ** é o vetor de tensões internas no elemento;
- **ε** é o vetor de deformações internas no elemento;

- D é o tensor de elasticidade do material utilizado, definido por:

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

sendo E e ν o módulo de Young e o coeficiente de Poisson do material, respectivamente.

Sejam:

- $\mathbf{K_e}$: Matriz de rigidez do elemento
- **B**: Matriz de forma do elemento

A matriz de rigidez K_e é definida pela integração gaussiana como (KIKUCHI, [7]):

$$K_e = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} B^T \cdot D \cdot B \cdot J \, d\xi d\eta d\zeta$$

sendo J o operador Jacobiano, ou seja, o determinante da matriz Jacobiana.

Definem-se as seguintes parcelas da matriz de elasticidade:

$$D_D = \frac{E}{2\cdot(1-2\nu)\cdot(1+\nu)}\begin{bmatrix} 2-4\nu & -1+2\nu & -1+2\nu & 0 & 0 & 0 \\ -1+2\nu & 2-4\nu & -1+2\nu & 0 & 0 & 0 \\ -1+2\nu & -1+2\nu & 2-4\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(Parcela relativa à deformação normal)

$$D_V = \frac{E}{2\cdot(1-2\nu)}\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(Parcela relativa à variação volumétrica)

$$D_{yz} = \frac{E}{2\cdot(1+\nu)}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(Parcela relativa à deformação por cisalhamento no plano yz)

$$D_{zx} = \frac{E}{2\cdot(1+\nu)}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(Parcela relativa à deformação por cisalhamento no plano zx)

$$D_{xy} = \frac{E}{2\cdot(1+\nu)}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

(Parcela relativa à deformação por cisalhamento no plano xy)

Assim:

$$D = D_D + D_V + D_{yz} + D_{zx} + D_{xy} \quad \Rightarrow$$

$$\Rightarrow \quad K_e = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} B^T \cdot D_D \cdot B \cdot J \, d\xi d\eta d\zeta + \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} B^T \cdot D_V \cdot B \cdot J \, d\xi d\eta d\zeta +$$

$$+ \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} B^T \cdot D_{yz} \cdot B \cdot J \, d\xi d\eta d\zeta + \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} B^T \cdot D_{zx} \cdot B \cdot J \, d\xi d\eta d\zeta + \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} B^T \cdot D_{xy} \cdot B \cdot J \, d\xi d\eta d\zeta$$

$$\therefore \quad K_e = \sum_{i=1}^{2}\sum_{j=1}^{2}\sum_{z=1}^{2} B^T \cdot D_D \cdot B \cdot J\Big|_{(\xi,\eta,\zeta)=(ai,aj,az)} + \sum_{i=1}^{2}\sum_{j=1}^{2}\sum_{z=1}^{2} B^T \cdot D_V \cdot B \cdot J\Big|_{(\xi,\eta,\zeta)=(ai,aj,az)} +$$

$$+ \sum_{i=1}^{2} 4B^T \cdot D_{yz} \cdot B \cdot J\Big|_{(\xi,\eta,\zeta)=(ai,0,0)} + \sum_{j=1}^{2} 4B^T \cdot D_{zx} \cdot B \cdot J\Big|_{(\xi,\eta,\zeta)=(0,aj,0)} + \sum_{z=1}^{2} 4B^T \cdot D_{xy} \cdot B \cdot J\Big|_{(\xi,\eta,\zeta)=(0,aj,0)}$$

Essa separação das parcelas da matriz de rigidez aumenta a sensibilidade do cálculo e, portanto, a acuracidade dos valores obtidos.

Após realizar esses cálculos para cada elemento, cria-se a chamada *matriz de rigidez global*, onde todos os resultados são combinados, isto é, os valores de cada elemento referentes a um mesmo par de graus de liberdades são somados.

Com essa matriz em mãos, pode-se calcular em todos os pontos da estrutura qual o efeito dos esforços aplicados.

3.2. Otimização topológica

Otimização consiste em encontrar a melhor solução para um problema. Em problemas de engenharia, esse processo pode ser dividido em duas categorias: otimização de função e otimização de parâmetros.

Na otimização de função, o objetivo é encontrar a solução que proporcione a melhor forma para uma dada função (menor variação de temperatura, maior área sobre a curva, etc).

A otimização de parâmetros, utilizada neste trabalho, consiste em encontrar a solução que apresente os melhores valores de parâmetros desejados (menor quantidade de massa, menor deformação em certos pontos). Esse método começou a ganhar força com a difusão do MEF, que divide o domínio estudado logo no começo da análise, de modo que as variáveis do problema são valores discretos nos nós, e não mais funções.

Dá-se o nome de *variáveis de projeto* aos parâmetros que podem ser modificados na busca pela solução ótima ao problema apresentado. A função ou os valores a serem otimizados são chamados de *função objetivo*.

Na otimização topológica, a preocupação é a distribuição de massa num domínio, ou seja, descobrir onde deve existir massa para maximizar ou minimizar uma função que dependa dessa distribuição, como por exemplo a rigidez da estrutura.

Portanto, as variáveis de projeto da otimização topológica serão as densidades de cada elemento do MEF, e a função objetivo será a flexibilidade média, uma função da energia potencial de deformação existente na estrutura quando submetida a esforços. Em outras palavras, introduz-se o conceito de pseudo-densidade, um fator entre 0 e 1 aplicado à rigidez de cada elemento.

A figura a seguir exemplifica o processo de otimização de uma barra apoiada nas extremidades e carregada no centro, com o resultado final ao lado. Os métodos utilizados são semelhantes aos implementados neste trabalho.

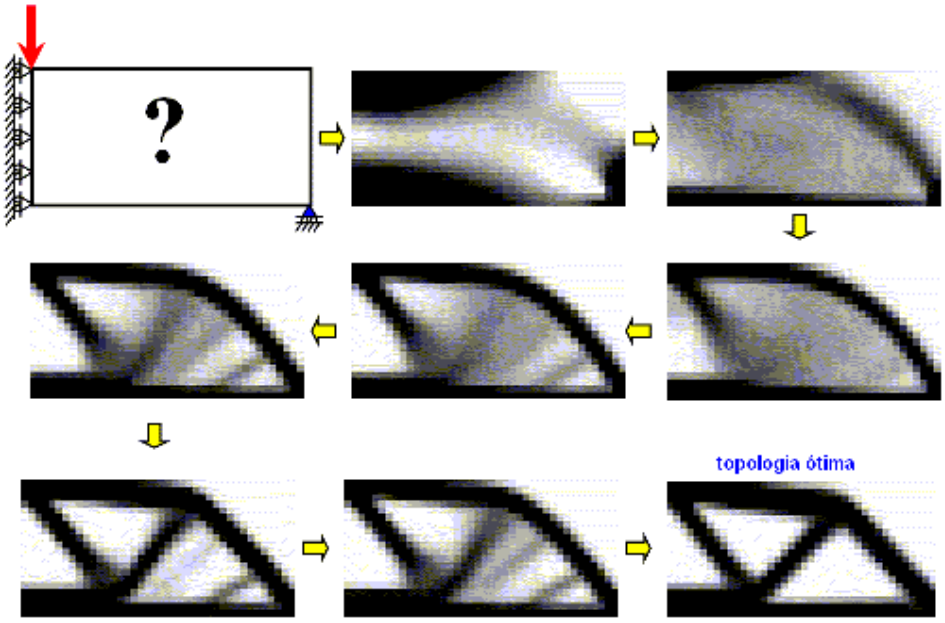


Figura 8 – Exemplo de otimização de barra apoiada pelas extremidades

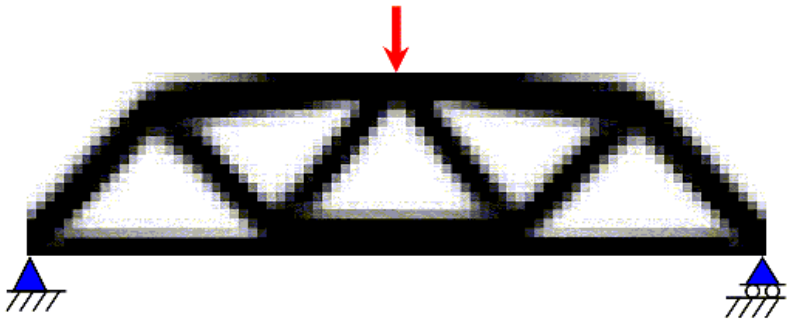


Figura 9 – Forma ótima para a rigidez no problema da viga bi-apoiada

Obviamente, não se pode fabricar algo com uma densidade intermediária – a matéria deve existir ou não, ou seja, a pseudo-densidade deve ser 0 ou 1. Porém, a literatura mostra (LIMA, [9]) que, trabalhando apenas com variáveis discretas, o processo de busca da forma ótima pode não convergir e, quando converge, a solução freqüentemente é uma estrutura desconexa, com a massa concentrada nos pontos de aplicação de esforços e restrições.

Assim, trabalha-se com uam variável **ρ** contínua entre 0 e 1, de acordo com a seguinte formulação:

$$D(x) = \rho(x) \cdot D_0$$

onde:

- **D(x)**: tensor de elasticidade do elemento
- **ρ(x)**: pseudo-densidade do elemento
- $\mathbf{D_0}$: tensor de elasticidade do material

Contudo, essa formulação, por si só, leva a uma estrutura com grande quantidade de elementos com densidades intermediárias entre 0 e 1, que também não são interessantes, dada a inviabilidade física.

Assim, após encontrar essa "solução intermediária", para reduzir a quantidade de elementos com densidades intermediárias, utiliza-se de um artifício simples, conhecido como *Simple Isotropic Materials With Penalization*, ou *SIMP*. Consiste em elevar a pseudo-densidade a uma potência **p**, chamada de fator de penalização. Dessa forma, os elementos com densidades entre 0 e 1 apresentam rigidez ainda menores. Ou seja, após a convergência do processo, este é retomado com a seguinte formulação:

$$D(x) = \rho(x)^p \cdot D_0$$

O valor de **p** é amplamente discutido na literatura, e é consenso que, para se obter uma estrutura fisicamente viável, este fator deve ser maior ou igual a 3. Um valor muito elevado, entretanto, tornaria o problema discreto, onde não haveria garantia de se obter a solução pelos métodos utilizados. Na implementação do software-base, foi definido **p = 3**. Isso previne o aparecimento das pseudo-densidades intermediárias, mas possibilita o aparecimento de mínimos locais nas funções-objetivo. Isso pode fazer com que o cálculo seja interrompido antes da completa otimização, parando em uma "otimização-local".

Na seqüência de figuras a seguir, apresenta-se como exemplo um processo de otimização topológica aplicado a uma estrutura de braço de suspensão para caminhões.

É importante notar que a otimização topológica por si só é um problema de otimização com uma única função objetivo, enquanto que o problema de mecanismos flexíveis apresenta duas funções-

objetivo, como veremos adiante, introduzindo a necessidade de artifícios para considerar as duas funções simultaneamente.

3.3. Mecanismos flexíveis

Dadas as funções de um mecanismo, duas propriedades se apresentam claramente como essenciais: a rigidez da estrutura como um todo e a flexibilidade para permitir os movimentos desejados. Ou seja, o mecanismo flexível deverá prover máxima deformação, em um ponto e direção bem definidos quando sujeito a esforços definidos, e ao mesmo tempo apresentar a rigidez necessária para suportar os trabalhos interno e externo.

Para realizar essa análise, utiliza-se uma formulação matemática em termos de energia, dividida em dois conceitos básicos: a energia mútua e a flexibilidade média. A primeira está diretamente relacionada com a geração dos movimentos desejados nos pontos especificados, e deve ser maximizada. A segunda representa o inverso da rigidez estrutural do mecanismo e deve, portanto, ser minimizada.

3.3.1. Flexibilidade média

Este é o conceito envolvido na otimização topológica. Consiste em analisar a energia potencial de deformação elástica gerada no corpo quando este é submetido a esforços externos. Pela teoria da mecânica do meio contínuo, define-se a flexibilidade média da seguinte forma:

$$L^1\left(u^1\right) = \int t^1 \cdot u^1 \, d\Gamma$$

sendo:

- $\mathbf{t^1}$: Esforço externo aplicado à estrutura
- $\mathbf{u^1}$: Campo de deslocamentos gerado no corpo pelo esforço $\mathbf{t^1}$
- $\mathbf{L^1\ (u^1)}$: Flexibilidade média da estrutura, correspondente à deformação $\mathbf{u^1}$ e ao esforço $\mathbf{t^1}$
- $\mathbf{\Gamma}$: Domínio do corpo no espaço

O esquema a seguir pode ajudar a compreender os conceitos:

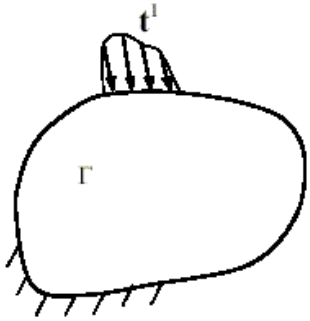


Figura 10 – Definição da estrutura analisada

Nota-se que a flexibilidade média tem dimensão de energia, e pode ser interpretada como sendo a quantidade de energia armazenada pelo corpo tridimensional quando submetido ao carregamento **t** numa dada superfície.

Na realidade, é fácil provar matematicamente que o módulo da flexibilidade média de um corpo tridimensional em equilíbrio é igual ao dobro da energia potencial total.

Essa é a primeira função-objetivo do problema.

3.3.2. Energia mútua

A energia mútua é analisada de forma semelhante à flexibilidade média. Porém, ela corresponde à aplicação de um esforço $\mathbf{t}^1$, e ao aparecimento de uma deformação específica desejada.

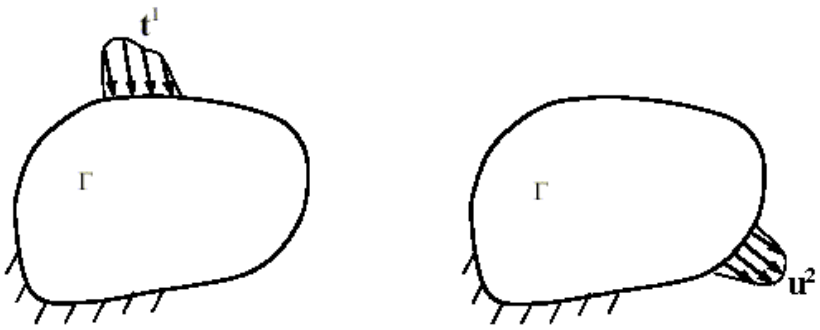


Figura 11 – Definição da estrutura para mecanismo flexível

Para estudar esse problema, define-se um esforço auxiliar $\mathbf{t}^2$, capaz de gerar o campo de deslocamentos $\mathbf{u}^2$ desejado. Ou seja, se o corpo fosse submetido apenas ao esforço $\mathbf{t}^2$, na superfície onde

é definido o vetor $\mathbf{u^2}$ desejado, ele sofreria o deslocamento $\mathbf{u^2}$. Assim, define-se a energia mútua do sistema:

$$L^1\left(u^2\right) = \int t^1 \cdot u^2 \, d\Gamma \quad \text{ou, pelo teorema da reciprocidade:} \quad L^2\left(u^1\right) = \int t^2 \cdot u^1 \, d\Gamma$$

O teorema de Betti prova que as duas formulações chegam ao mesmo valor (LIMA, [9]), ou seja, o trabalho realizado pela força $\mathbf{t^1}$, ao longo do campo de deslocamentos $\mathbf{u^2}$, é igual ao trabalho realizado pela força $\mathbf{t^2}$, ao longo do campo de deslocamentos $\mathbf{u^1}$. Em outras palavras, pode-se dizer que o movimento do mecanismo é recíproco: ao se aplicar o esforço no ponto onde o movimento era antes desejado, obtém-se um movimento naquele que, originalmente, era o ponto de aplicação.

Enfim, a energia mútua representa a deformação acumulada no corpo tridimensional entre as regiões de aplicação dos esforços $\mathbf{t^1}$ e $\mathbf{t^2}$. Assumindo como unitário o valor de $\mathbf{t^2}$, a energia mútua indica o deslocamento da região de aplicação dessa força quando o corpo é submetido ao esforço $\mathbf{t^1}$.

Essa é a segunda função-objetivo do problema.

4. Formulação matemática

4.1. Notação matricial

Com equações de equilíbrio estático, podemos provar que:

$$L^1\left(u^2\right) = \int_\Omega \varepsilon\left(u^1\right) \cdot D \cdot \varepsilon\left(u^2\right)$$
$$L^3\left(u^3\right) = \int_\Omega \varepsilon\left(u^3\right) \cdot D \cdot \varepsilon\left(u^3\right)$$

onde $\varepsilon(u)$ é o tensor de deformações correspondente ao campo de deslocamentos u, e D é o tensor de rigidez definido anteriormente, função das pseudo-densidades e do material. Assim, obviamente, essa relação é importante por destacar a variável de projeto utilizada.

Contudo, para a análise realizada neste trabalho, por elementos finitos, é mais conveniente utilizar uma notação matricial para as funções relacionadas. A partir das relações acima, pode-se escrever:

$$L^1\left(u^2\right) = \{u_1\} \cdot [K] \cdot \{u_2\}$$
$$L^3\left(u^3\right) = \{u_3\} \cdot [K] \cdot \{u_3\}$$

onde:

- $\{u_1\}$ é o vetor de deslocamentos em cada grau de liberdade da estrutura discretizada em elementos finitos, gerados pelo esforço t_1;
- $\{u_2\}$ é o vetor de deslocamentos em cada grau de liberdade da estrutura discretizada em elementos finitos, gerados pelo esforço t_2;
- $\{u_3\}$ é o vetor de deslocamentos em cada grau de liberdade da estrutura discretizada em elementos finitos, gerados pelo esforço t_3;
- [K] é a matriz de rigidez global da estrutura, função dos tensores de elasticidade de todos os elementos e, portanto, da pseudo-rigidez de cada um.

Essa será a notação utilizada a partir daqui.

4.2. Função objetivo

O mecanismo flexível deve ser projetado levando-se em conta simultaneamente as duas funções-objetivo apresentadas acima, ou seja, é um problema de otimização multifuncional.

Na literatura de otimização, algumas formas de se tratar otimizações desse tipo são sugeridas. Um método já aplicado com sucesso a mecanismos bidimensionais consiste em definir a seguinte *função multiobjetivo*:

$$\underset{\rho}{Maximizar}: \quad F_1 = \frac{Energia\ mútua}{Flexibilidade\ Média}$$

sujeito a:

$$\sum_{x=1}^{N} \rho(x) \leq V$$

onde V é uma limitação de volume imposta pelo usuário em termos de porcentagem do domínio inicial que pode ser ocupada por massa.

Sendo ρ a variável de projeto analizada, ou seja, as pseudo-densidades de cada elemento do domínio de elementos finitos. É intuitivo perceber que a função F_1 atingirá seu valor máximo quando a energia mútua for maximizada e a flexibilidade média minimizada, cumprindo assim as duas especificações do projeto.

Porém, para permitir melhor controle sobre a importância de cada uma das parcelas da energia no cálculo final, realiza-se uma generalização da função F_1, chegando à seguinte forma:

$$\underset{\rho}{Maximizar}: \quad F_2 = w \cdot \ln(Energia\ mútua) - (1-w) \cdot \ln(Flexibilidade\ Média)$$

sujeita à mesma restrição de volume. O fator **w**, contido entre 0 e 1, é um coeficiente de peso para cada parcela. Essa formulação permite flexibilizar a função objetivo original, F_1, de forma a priorizar uma ou outra característica do mecanismo flexível final.

Essa segunda formulação, implementada no início do trabalho, mostrou-se extremamente conveniente para os problemas de mecanismos flexíveis. Contudo, ela foi posteriormente substituída por outra mais elaborada, introduzindo outros fatores à função objetivo, com o intuito de minimizar movimentos que ocorrem nos mecanismos além dos especificados, como deslocamentos em outras direções quando solicitado o movimento verdadeiro do mecanismo.

Essa nova formulação, sugerida por NISHIWAKI et al. [8], é a seguinte:

$$F_3 = \frac{L_{21}}{w_1 \cdot L_{11} + w_2 \cdot L_{22} + w_3 \cdot L_{33} + w_4 \cdot L_{44}}$$

onde L_{21} é a energia mútua referente ao movimento desejado (índice 1 para entrada e 2 para saída), e os outros valores são flexibilidades médias referentes aos movimentos indesejados. Os diagramas a seguir exemplificam essas definições:

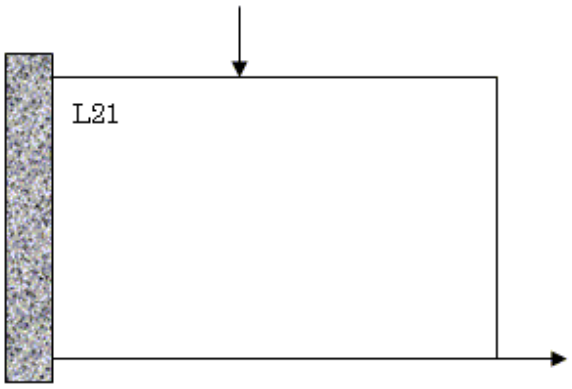


Figura 12 - Definição da energia mútua desejada

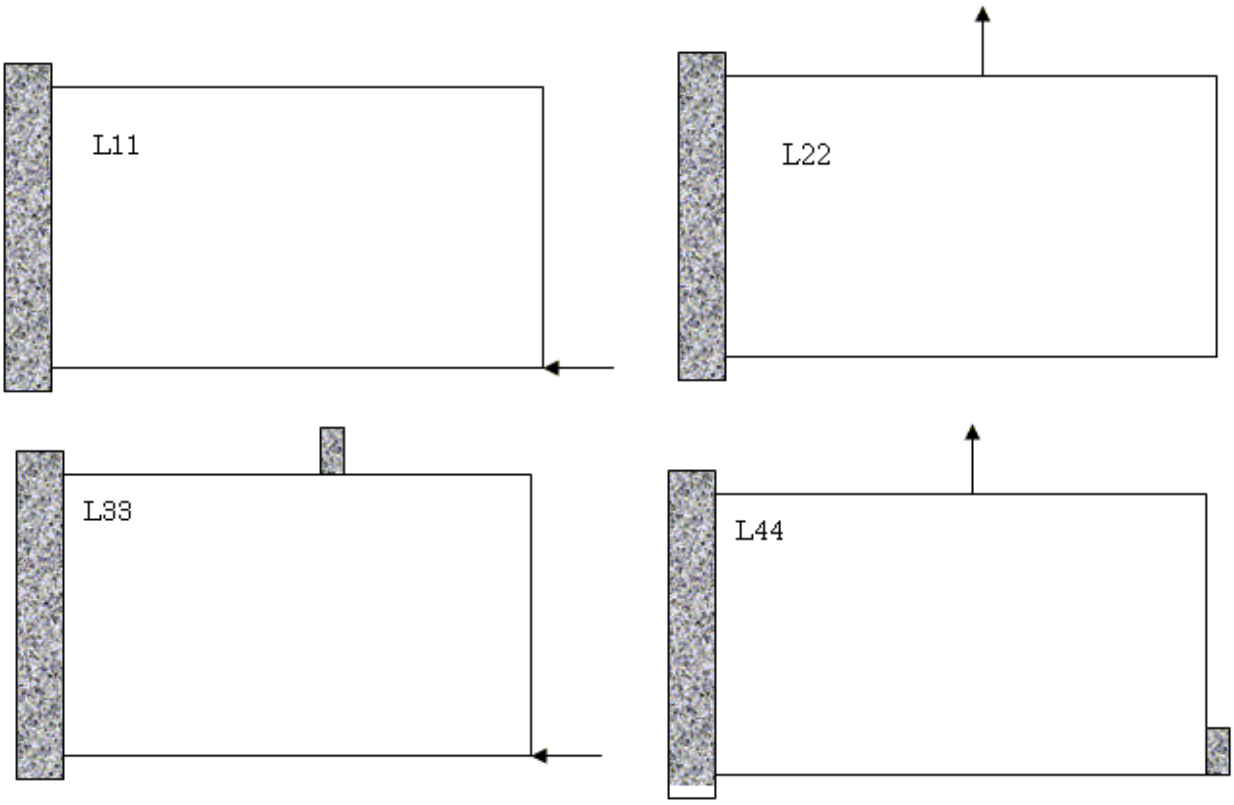


Figura 13 – Flexibilidades médias que devem ser minimizadas

Os fatores w_1, w_2, w_3 e w_4 podem ser ajustados, tendo em mente que:

- a soma dos quatro fatores não deve ficar distante de 1, para não desbalancear a função objetivo, aumentando ou diminuindo demais a importância relativa da energia mútua quanto às outras funções analisadas;

- as flexibilidades L_{11} e L_{22} referem-se aos movimentos indesejados que poderiam surgir durante a utilização do mecanismo, enquanto que;

- as flexibilidades L_{33} e L_{44} referem-se à rigidez da estrutura de um modo geral para resistir aos esforços a que será submetida.

Por exemplo, para dar prioridade à limitação dos movimentos indesejados, pode-se utilizar w_1 e w_2 iguais a 0,4 e w_3 e w_4 iguais a 0,1. Esses foram os valores utilizados para se obter os resultados apresentados neste trabalho.

A restrição de volume continua a mesma, definida pelo limite V.

4.3. Análise de sensibilidades

Conforme visto acima, é necessário conhecer a derivada parcial da função objetivo a cada variável de projeto, ou seja, à pseudo-densidade de cada elemento finito da estrutura discretizada.

Portanto, estamos interessados em:

$$\frac{\partial F_1}{\partial \rho_i} = \frac{1}{L^3\left(u^3\right)} \cdot \frac{\partial L^1\left(u^2\right)}{\partial \rho_i} - L^1\left(u^2\right) \cdot \frac{1}{\left[L^3\left(u^3\right)\right]^2} \cdot \frac{\partial L^3\left(u^3\right)}{\partial \rho_i}$$

$$\frac{\partial F_2}{\partial \rho_i} = \frac{w}{L^1\left(u^2\right)} \cdot \frac{\partial L^1\left(u^2\right)}{\partial \rho_i} - \frac{\left(1-w\right)}{L^3\left(u^3\right)} \cdot \frac{\partial L^3\left(u^3\right)}{\partial \rho_i}$$

Note que as equações acima envolvem os valores das derivadas parciais da energia mútua e da flexibilidade média.

Assim, as derivadas parciais podem ser escritas na forma:

$$\frac{\partial L^1\left(u^2\right)}{\partial \rho_i} = \left\{u_1\right\} \cdot \frac{\partial [K]}{\partial \rho_i} \cdot \left\{u_2\right\}$$

$$\frac{\partial L^3\left(u^3\right)}{\partial \rho_i} = \left\{u_3\right\} \cdot \frac{\partial [K]}{\partial \rho_i} \cdot \left\{u_3\right\}$$

Como definido previamente, *[K]* corresponde a uma combinação dos tensores de elasticidade de cada elemento, e, obviamente, a pseudo-densidade de um elemento não influenciará o tensor de outro. Assim, pode-se encontrar a dependência da função objetivo com relação a cada variável de projeto derivando cada tensor de elasticidade pela sua respectiva pseudo-densidade.

$$D_i = \rho_i^p \cdot D_0 \Rightarrow \frac{\partial D_i}{\partial \rho_i} = p \cdot \rho_i^{(p-1)} \cdot D_0$$

5. Implementação numérica

5.1. Fluxograma

O programa funciona segundo o seguinte fluxograma:

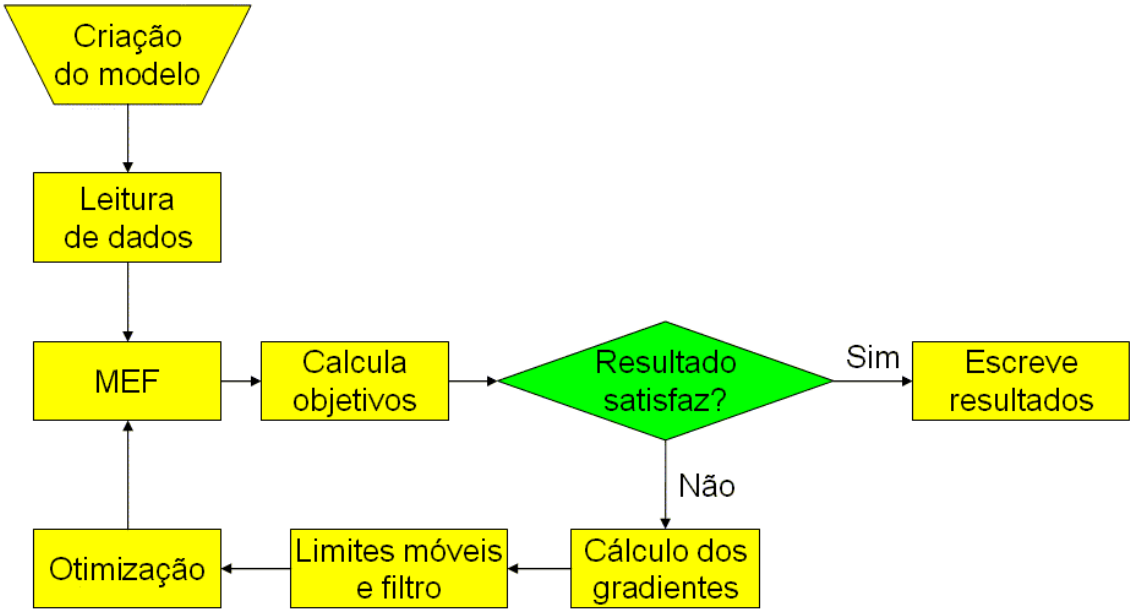


Figura 14 – Fluxograma do programa implementado

Cada passo do diagrama acima será apresentado em detalhe a seguir.

5.2. Modelo de entrada

A construção do modelo em elementos finitos é realizada no software comercial *AnSys*. Note que foi utilizada a versão 5.4 desse software. A utilização de versões diferentes pode ocasionar problemas de compatibilidade na leitura de dados. Estes, porém, podem ser facilmente solucionados pela alteração de algumas linhas-chave dos arquivos de entrada, utilizadas pelo software de mecanismos flexíveis para reconhecer os dados. Estas linhas estão destacadas no exemplo de arquivo de entrada disponível em anexo.

É necessário construir dois modelos: o primeiro com o esforço de entrada e o segundo com o esforço de saída, de modo que o programa possa diferenciar o que é fornecido do que é desejado.

Deve-se criar um domínio tridimensional com as proporções desejadas e especificar todas as restrições existentes, como fixações e condições de simetria. Obviamente, nos dois modelos o domínio e a malha devem ser iguais. Além disso, o software de mecanismos flexíveis assume que a malha possui apenas elementos cúbicos perfeitos e de volumes idênticos. É possível adaptá-lo para malhas mais complexas, com elementos irregulares, sendo que as rotinas do método de elementos finitos já estão preparadas para aceitar elementos com qualquer forma. Porém, isso exigiria mais cálculos e um aumento no tempo de processamento.

Uma vez feitos esses modelos, pode-se gravá-los em arquivos que o *AnSys* chama de *database*, através do comando **CDWRITE**. Estes devem ser chamados de *carga1.cdb* (esforços de entrada) e *carga2.cdb* (esforços de saída). Internamente, o programa de mecanismos flexíveis cria ainda dois outros casos de carregamento, com esforços opostos ao primeiro e ao segundo, para analizar a rigidez da estrutura. Esses arquivos de entrada contêm ainda informações sobre o material utilizado.

5.3. Leitura dos dados

A primeira tarefa executada pelo software é a leitura dos dados gerados pelo usuário através do *AnSys*. Esses dados são fornecidos a rotinas que fazem a construção da matriz de rigidez da malha de elementos finitos, com as pseudo-densidades iniciais fornecidas pelo usuário.

Além das pseudo-densidades iniciais, o usuário deve fornecer um limite de volume da estrutura, na forma de proporção do domínio inicial, ou seja, um valor entre 0 e 1.

Em seguida, são calculadas as condições iniciais dos deslocamentos de cada nó em cada direção para cada caso de carregamento, ou seja, a variação da estrutura em cada grau de liberdade irrestrito, bem como a energia mútua do sistema e todas as flexibilidades médias pertinentes.

5.4. Elementos finitos

A descrição do tipo de elemento finito utilizado no programa já foi feita acima.

A rotina utilizada para criar as matrizes de rigidez locais é uma adaptação para a linguagem C da rotina disponibilizada na literatura, originalmente na sintaxe do software *MatLab*. Essas matrizes são todas montadas no início do software e armazenas permanentemente, uma vez que seus valores não mudam com as pseudo-densidades e pode-se, assim, ganhar muito tempo de processamento. Uma rápida estimativa conclui que uma máquina com 256 MB de memória RAM deve ser capaz de armazenar por volta de 50.000 elementos dessa forma.

Contudo, aqui surge um problema de memória. Numa malha simples, o número de nós é um pouco maior que o número de elementos. Para cada nó de uma malha tridimensional, temos três graus de liberdade. A matriz global é uma matriz quadrada com uma posição para cada par de graus de liberdade, ou seja, o número de valores dessa matriz é igual ao quadrado do número de graus de liberdade. Em um modelo de 5.000 elementos, teríamos aproximadamente 17.000 graus de liberdade, já excluídos aqueles anulados pelas restrições (engastamentos e apoios). Isso significaria armazenar na memória $2{,}9 \cdot 10^8$ valores diferentes!

Todavia, a maioria desses valores é nula. Isso pode ser percebido de forma intuitiva pela definição do método dos elementos finitos. Quando se faz a matriz de um elemento, utilizam-se os graus de liberdade utilizados por aquele elemento, ou seja, nesse caso, os 24 graus referentes aos 3 movimentos de cada um dos 8 nós do elemento. Na montagem da matriz global, não vão aparecer, obviamente, os pares de graus de liberdade que não são conectados por um mesmo elemento, ou seja, referentes a nós de elementos diferentes.

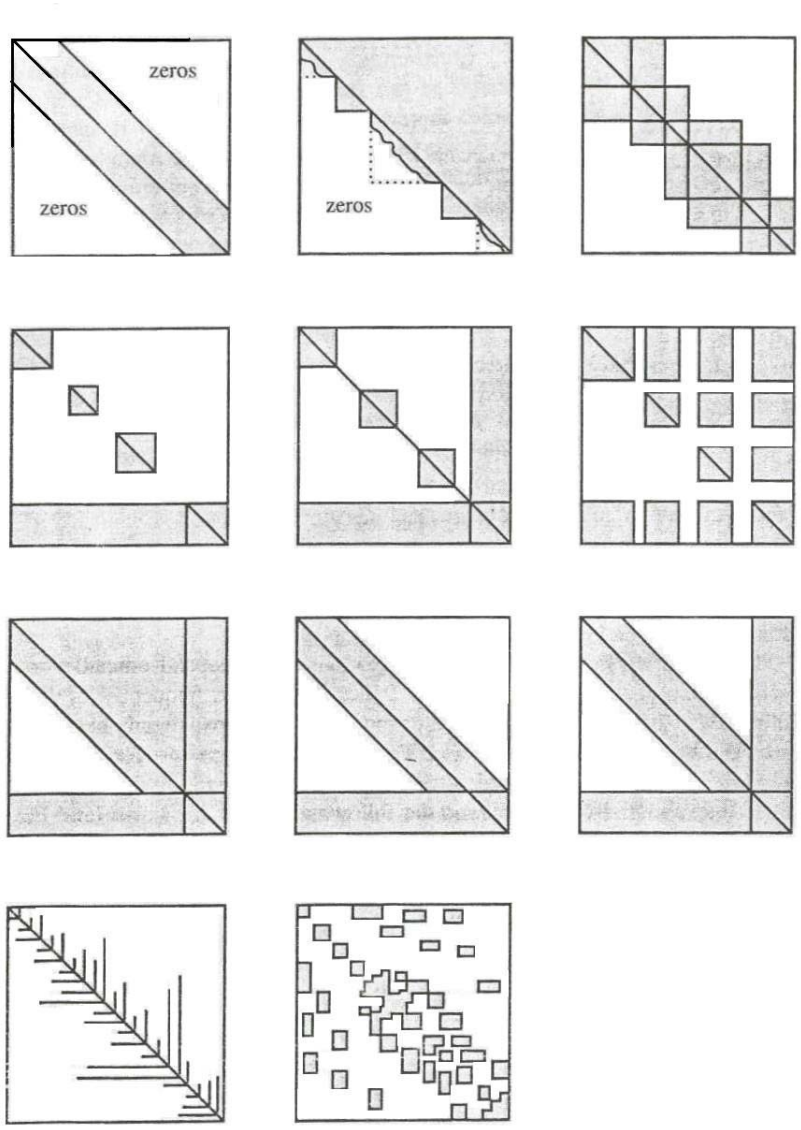


Figura 15 – Diversos tipos de matrizes esparsas

Assim, este trabalho utiliza uma estrutura de dados apresentada por PRESS et al [12], desenvolvida especialmente para tratamento das assim chamadas matrizes esparsas. Um conjunto de rotinas para realizar cálculos com matrizes armazenadas nessa forma está disponível na mesma fonte. Esse método de armazenamento é detalhado a seguir.

5.4.1. Matrizes esparsas indexadas

Existem na literatura diversas estruturas de dados para lidar com matrizes esparsas. Infelizmente, nenhuma delas é tomada como método padrão entre todos que trabalham com esse tipo de matriz.

Assim, para este trabalho, foi escolhida a estrutura sugerida por PRESS et al [12], que entre as mais eficientes disponíveis: armazena na memória uma quantidade de valores igual a pouco mais do que o dobro de valores não nulos na matriz – outros métodos podem chegar a cinco vezes esse número. Além disso, é disponibilizada uma gama de rotinas especial para esse método, em linguagem C, capaz de resolver os problemas de soluções de sistemas lineares tratados aqui. Essa estrutura tem a limitação de lidar apenas com matrizes quadradas, mas isso não é um problema visto que uma matriz de rigidez para o método dos elementos finitos não somente é sempre quadrada, mas também simétrica.

Para representar uma matriz *A* de dimensões *N x N*, o método da linha indexada utiliza dois vetores de uma dimensão, chamados de *sa* e *ija*. O primeiro guarda os valores da matriz em si, enquanto o segundo armazena índices desses valores, respeitando as seguintes regras:

- As primeiras *N* posições de *sa* guardam os elementos da diagonal da matriz *A*, em ordem. Mesmo valores nulos são armazenados. De forma genérica, isso representa uma leve ineficiência do método, mas nesse caso, os elementos da diagonal nunca são nulos. Um elemento nulo na diagonal corresponderia ao que se chama de movimento de corpo rígido, ou seja, uma análise do corpo livre no espaço, sem restrição alguma.
- Cada uma das primeiras *N* posições de *ija* guarda o índice do vetor *sa* que contém o primeiro elemento não-nulo **e** não-diagonal da linha correspondente da matriz *A*. Se a linha contiver apenas elementos nulos, o índice será igual ao índice da linha seguinte – ou igual ao índice do último elemento da linha anterior acrescido de 1.
- O valor *N+1* de *ija* é igual ao tamanho do vetor *sa*, acrescido de 1. Esse valor pode ser lido para determinar a quantidade de elementos não nulos na matriz *A*. A posição *N+1* de *sa* não é utilizada e pode receber um valor arbitrário qualquer.
- Por conseqüência, a primeira posição de *ija* sempre recebe o valor *N+2*.
- As posições posteriores a *N+1* em *sa* recebem os valores dos elementos não-nulos e não-diagonais da matriz *A*, ordenados por linha (em ordem crescente) e, em cada linha, ordenados por coluna (também em ordem crescente).
- Cada posição posterior a *N+1* em *ija* recebe o número da coluna do valor de mesma posição no vetor *sa*.

Um exemplo pode ajudar a entender melhor o esquema de armazenagem. Para isso, define-se a matriz *A* qualquer, de tamanho 5 x 5:

$$A = \begin{bmatrix} 3 & 0 & 1 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 0 & 7 & 5 & 9 & 0 \\ 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 6 & 5 \end{bmatrix}$$

Essa matriz será representada pelos seguintes vetores, de acordo com as regras apresentadas acima:

Índice k		1	2	3	4	5	6	7	8	9	10	11
ija[k]		7	8	8	10	11	12	3	2	4	5	4
sa[k]		3.0	4.0	5.0	0.0	5.0	*x*	1.0	7.0	9.0	2.0	6.0

5.5. Limites móveis

Os chamados *limites móveis* são criados como restrições para as rotinas de otimização, para que a solução não se afaste demais da realidade devido à linearização da função objetivo.

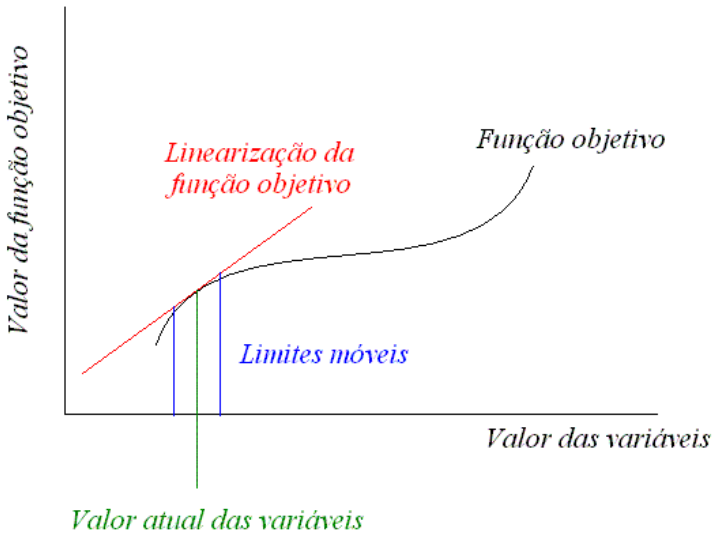


Figura 16 – Limites móveis impedem que função linearizada se distancie demais da original

Sobre esses limites aplica-se ainda o filtro que impede a geração de regiões de "tabuleiros de xadrez", como mostrado mais adiante. Só então pode-se passar os dados para a rotina de otimização escolhida, para maximizar a função objetivo.

A determinação desses limites a cada iteração se faz com base na variação de cada variável nas três últimas iterações. Caso a atualização da variável tenha se alternado entre um aumento e uma redução, conclui-se que a variável está oscilando e a gama onde ela pode variar (os limites móveis) é diminuída. Caso tenha sido um constante aumento ou constante redução, essa gama é ampliada, permitindo uma convergência mais rápida do processo.

A rotina *limites*, criada por CARDOSO e FONSECA [15], e disponível no **ANEXO C**, é responsável pelo cálculo desses limites móveis.

5.5.1. Filtro de malha

Um dos problemas dessa metodologia é a criação de áreas com alternância de massa entre os elementos, os chamados *tabuleiros de xadrez*. A figura a seguir exemplifica o caso num mecanismo flexível bidimensional desenvolvido utilizando o mesmo método (LIMA, [9]).

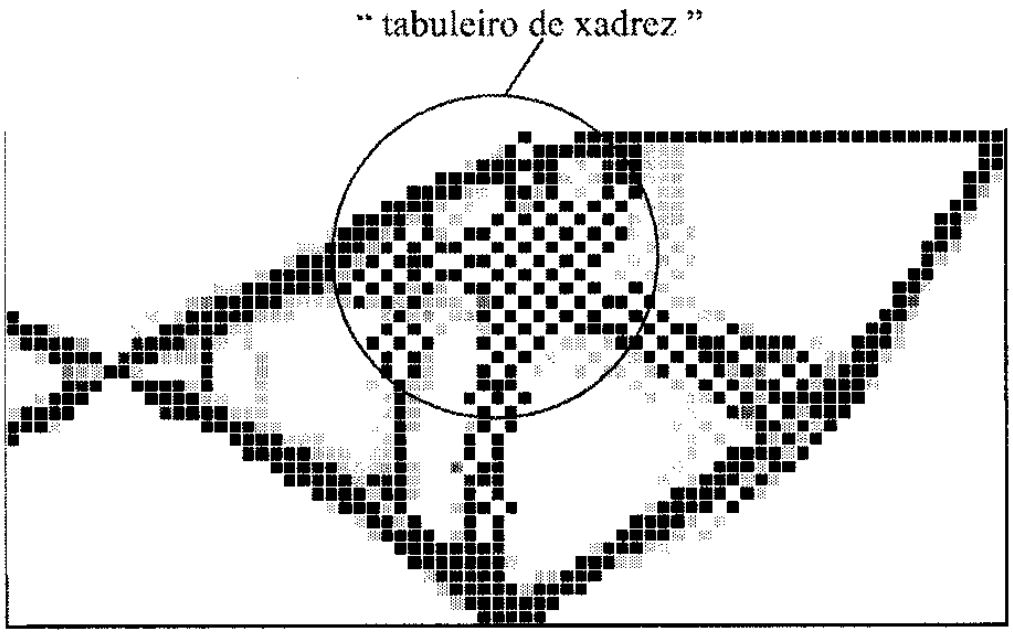


Figura 17 – Exemplo de formação de tabuleiro de xadrez

A explicação mais coerente para o fenômeno é que, segundo a formulação matemática utilizada, essa configuração apresenta uma relação extremamente favorável entre resistência e quantidade de massa (LIMA, [9]). Contudo, essas regiões são indesejáveis, visto que são fisicamente inviáveis e o intuito é o de projetar mecanismos reais.

Uma possível solução para o problema seria aumentar o grau do elemento finito utilizado, ou seja, o número de nós por elemento. Contudo, isso implicaria um aumento considerável do custo computacional e, portanto, do tempo de síntese. Além disso, não existe garantia de simplicidade numa solução deste tipo – poder-se-ia encontrar uma solução fisicamente viável, porém com uma estrutura tão complexa a ponto de se tornar impraticável.

A outra opção, escolhida neste trabalho, é a aplicação de um filtro sobre a malha de elementos finitos, limitando os valores das variáveis de projeto de acordo com os elementos vizinhos. Alguns autores sugerem a aplicação destes filtros diretamente sobre o campo de densidades da estrutura. Entretanto, como as densidades são aqui as variáveis de projeto, isto alteraria os valores encontrados como ótimos para a estrutura.

Seguindo a sugestão de LIMA [9], o filtro aqui é aplicado sobre os limites móveis passados à rotina de otimização, e apresenta a mesma formulação, descrita a seguir.

Primeiro, define-se um raio de abrangência r_{max}, que definirá quais elementos vão influenciar a variável analisada. Qualquer elemento cujo centro encontre-se a uma distância do centro do elemento analisado menor que o raio de abrangência será considerado um vizinho. O conceito é mostrado abaixo, para um elemento *i* com *nv* vizinhos *j*.

Figura 18 – Raio de abrangência do filtro

Em seguida, é calculado um peso para a influência dos vizinhos sobre o elemento central, de acordo com a distância destes vizinhos:

$$\overline{s} = \frac{\sum_{j=1}^{nv} s_j}{nv}, \quad sendo: \qquad s_j = \frac{r_{\max} - r_{ij}}{r_{\max}}$$

Com esse peso, calculam-se os limites inferior e superior do elemento *i*, segundo os respectivos valores dos vizinhos:

$$\rho_i^{\min} = \frac{\rho_i^{\min} \cdot V_i + \overline{s} \cdot \sum_{j=1}^{nv} \rho_j^{\min} \cdot V_j}{V_i + \overline{s} \cdot \sum_{j=1}^{nv} V_j} \qquad e \qquad \rho_i^{\max} = \frac{\rho_i^{\max} \cdot V_i + \overline{s} \cdot \sum_{j=1}^{nv} \rho_j^{\max} \cdot V_j}{V_i + \overline{s} \cdot \sum_{j=1}^{nv} V_j}$$

Como neste caso os volumes V_i de todos os elementos são iguais, as expressões acima podem ser simplificadas para:

$$\rho_i^{\min} = \frac{\rho_i^{\min} + \overline{s} \cdot \sum_{j=1}^{nv} \rho_j^{\min}}{1 + \overline{s}} \qquad e \qquad \rho_i^{\max} = \frac{\rho_i^{\max} + \overline{s} \cdot \sum_{j=1}^{nv} \rho_j^{\max}}{1 + \overline{s}}$$

Com esse filtro, é possível impedir a formação das regiões de tabuleiro de xadrez, sem alterar artificialmente as variáveis de projeto.

Entretanto, a utilização do filtro incorre no surgimento de algumas zonas de densidades intermediárias, também indesejadas. Assim, segundo sugestão de LIMA [9], após atingido certo valor de tolerância na função objetivo, o filtro é desligado e mais algumas iterações são realizadas. A experiência mostra que o número ideal dessas iterações extras está entre seis e oito.

A rotina responsável pela aplicação do filtro sobre os limites móveis foi chamada de *filtro* e está disponível no **ANEXO C**.

5.6. Otimização

A aplicação de qualquer um dos métodos de otimização aqui descritos exige a linearização da função objetivo analisada, para que se possa avaliar a influência de cada variável de projeto sobre seu valor. Isso foi feito por séries de Taylor, como visto anteriormente.

5.6.1. Programação linear

O objetivo da programação linear é resolver o seguinte tipo de problema: *para N variáveis independentes* $x_1, x_2, \ldots, x_N$, *maximizar a função linear:*

$$z = a_{01}x_1 + a_{02}x_2 + \ldots + a_{0N}x_N$$

respeitando as condições primárias:

$$x_1 \geq 0,\ x_2 \geq 0,\ \ldots,\ x_N \geq 0$$

e M condições adicionais, sendo m_1 na forma:

$$a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{iN}x_N \leq b_i \geq 0 \qquad i = 1, 2, \ldots, m_1$$

m_2 na forma:

$$a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{iN}x_N \geq b_i \geq 0 \qquad i = m_1 + 1, m_1 + 2, \ldots, m_1 + m_2$$

e m_3 na forma:

$$a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{iN}x_N = b_i \geq 0 \qquad i = m_1 + m_2 + 1, m_1 + m_2 + 2, \ldots, m_1 + m_2 + m_3$$

Existem diversos métodos conhecidos de programação linear, dos quais os mais citados na literatura são o *Simplex* e o *Kamarkar*.

5.6.1.1. Simplex

Em um primeiro instante, este projeto trabalhou com o método *simplex*, que apresenta eficiência razoável para uma quantidade de variáveis na ordem de centenas. As rotinas do método, em linguagem C, estão disponíveis na literatura (PRESS et al. [12]).

Este método consiste em, a cada iteração, partir de uma solução disponível para um problema, e mudar os valores das variáveis de projeto, sempre no limite de uma das restrições impostas, até que se encontre um valor mínimo ou máximo para a função objetivo. Os gradientes da função objetivo em relação às variáveis são utilizados para direcionar esse avanço.

5.6.1.2. DSPLP

Posteriormente, o método *simplex* no programa foi substituído pelas rotinas DSPLP, baseadas no método de *Kamarkar* (HAFTKA [18]). Trata-se de um conjunto de rotinas em Fortran, programadas por Hanson e Hiebert, do Sandia National Laboratory (EUA), e apresenta robustez e rapidez na solução de problemas com milhares de variáveis, possibilitando assim o estudo de uma malha mais refinada para obter resultados melhores do que com as rotinas Simplex.

Essas rotinas são distribuídas livremente pela internet no endereço *www.netlib.org/slatec*, juntamente com as instruções para sua utilização. Contudo, como as funções estão em Fortran, foi necessário transportar o programa para a plataforma Linux, onde é relativamente simples fazer a comunicação entre rotinas programadas em linguagens diferentes.

5.6.1.3. Linearização da função objetivo

Para resolução do problema proposto por qualquer método de Programação Linear, devemos linearizar a função objetiva para obter uma distinção clara da influência de cada variável sobre seu valor final. Assim, foi feita uma aproximação por séries de Taylor de primeira ordem:

$$F = F^0 + \sum_{i=1}^{N} \frac{\partial F}{\partial \rho_i} \cdot \left(\rho_i - \rho_i^0 \right)$$

Note que essa aproximação pode ser aplicada a qualquer formulação apresentada para a função objetivo. Contudo, a intenção dessa aproximação é maximizar a da função objetivo, e não conhecer seu valor (que pode ser feito de maneira mais precisa). Assim, foram descartadas as parcelas constantes da equação anterior, chegando finalmente a:

$$F = \sum_{i=1}^{N} \frac{\partial F}{\partial \rho_i} \cdot \rho_i$$

5.6.2. Critério da optimalidade

O critério da optimalidade é apresentado de forma extensa na literatura, com destaque para BENDSOE & SIGMUND, **Topology Optimization** [2], onde sua implementação está exemplificada tanto para a otimização de estruturas quanto para o desenvolvimento de mecanismos flexíveis.

Este método proporciona a atualização de cada variável de projeto independentemente das outras, aumentando a pseudo-densidade dos elementos nos pontos onde a energia de deformação é alta, e diminuindo-a onde este valor é baixo.

Essa atualização é feita com a ajuda do operador **Λ**, que, numa configuração ótima para a estrutura, será definido para cada variável da seguinte forma:

$$\Lambda = \frac{\partial F}{\partial \rho(x)}$$

sendo **F** a função objetivo do problema.

O critério da optimalidade define ainda o fator **B** de cada iteração para cada variável, que, numa condição de estrutura com aproveitamento ótimo de sua massa, será igual a 1, ou seja:

$$B = \Lambda^{-1} \frac{\partial F}{\partial \rho(x)}$$

Utilizando o método da bipartição, determina-se qual o fator **B** de cada variável na configuração atual. Caso ele seja menor que 1, a massa local é aumentada; caso seja maior, é reduzida, sempre dentro dos limites móveis já definidos. A alteração aplicada, caso esteja dentro dos limites móveis, é a seguinte:

$$\rho^{k+1}(x) = B^{0,3} \cdot \rho^{k}(x)$$

Uma vez implementado, este método mostrou bons resultados no desenvolvimento de mecanismos flexíveis, além de ser consideravelmente mais rápido e simples do que os métodos de programação linear.

A rotina *OC*, implementada em linguagem C, está disponível no **ANEXO C**.

5.7. Visualização de resultados

Finalmente, analiza-se a variação do valor da função objetivo na última iteração e, se esta for menor que um valor determinado pelo código do programa, o usuário tem a opção de interromper a busca pela solução. Caso isso aconteça, um novo arquivo é gerado, contendo as informações da malha. Este arquivo pode ser lido novamente pelo *AnSys*, com os tons de cinza dos elementos representando suas pseudo-densidades.

Dados sobre a evolução da solução são escritos num arquivo para ser executado no MatLab, de modo a gerar gráficos da função objetivo e do volume em função das iterações. Isso possibilita uma análise mais cuidadosa da convergência e da evolução da solução.

6. Resultados

6.1. Validação

Para validação do software implementado, foi realizada uma análise de um mecanismo inversor de movimento linear, já estudado por LIMA [9] com elementos bidimensionais, possibilitando a comparação dos resultados.

6.1.1. Descrição da análise realizada

A intenção é construir um mecanismo flexível com o domínio apresentado na figura a seguir:

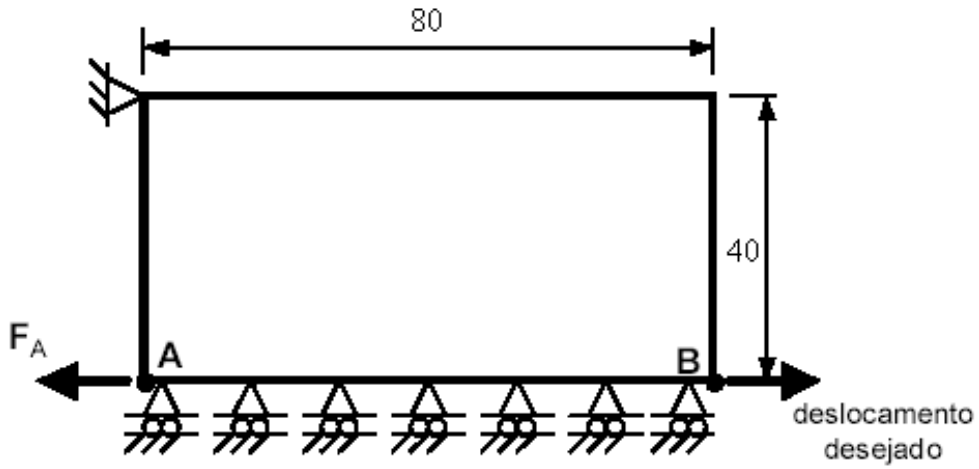


Figura 19 – Domínio estudado

A força F_A é o esforço a ser realizado pelo usuário, enquanto o deslocamento horizontal em B é o movimento desejado. Note que na borda inferior do domínio são aplicadas restrições que representam uma condição de simetria em relação a essa linha.

Os esforços de entrada e saída foram modelados com forças de valor unitário. As densidades iniciais de cada elementos foram 0,1 e o volume foi limitado em 20% do volume total do domínio. As características do material foram igualadas às do alumínio: módulo de elasticidade de 71 GPa e coeficiente de poisson de 0,34.

6.1.2. Resultados esperados

A mesma análise foi realizada por LIMA [9], em 2002, com um domínio bidimensional dividido em uma malha de elementos quadriláteros. A figura a seguir mostra o resultado por ele obtido.

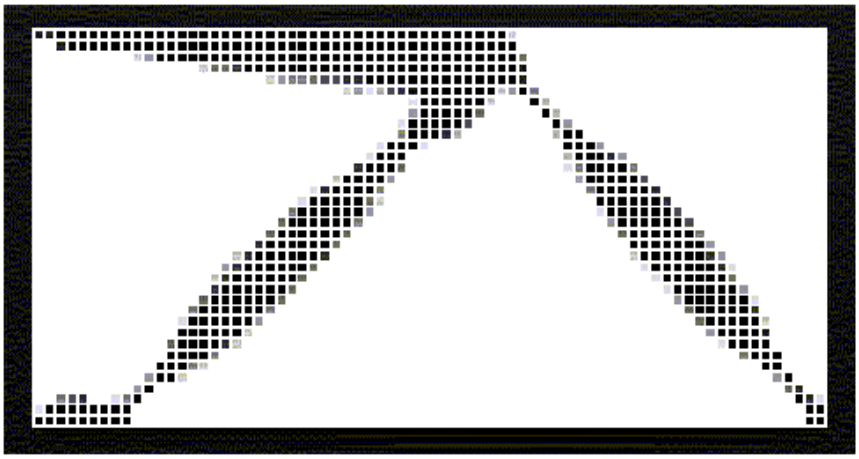


Figura 20 – Resultado esperado na análise realizada

6.1.3. Resultados obtidos

Aqui o domínio foi dividido numa malha de 80 x 40 elementos, mas foi introduzida uma espessura. Os elementos utilizados, portanto, não são mais bidimensionais. São tijolos de 8 nós cada um. A figura a seguir mostra o resultado obtido.

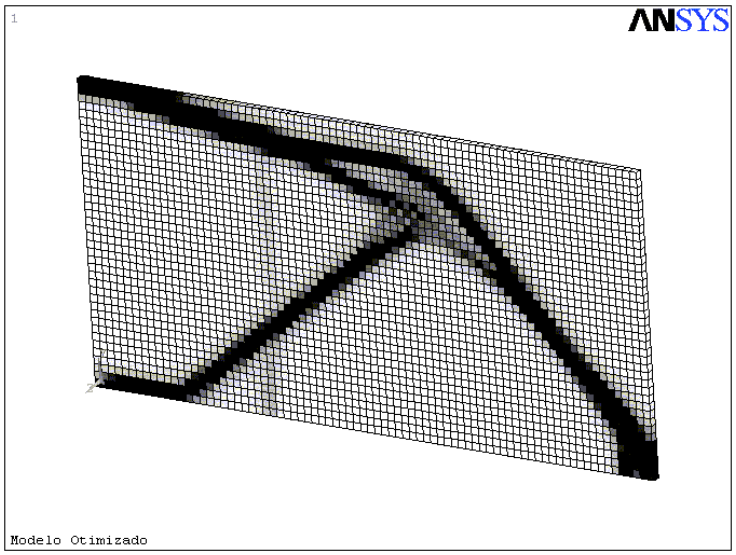
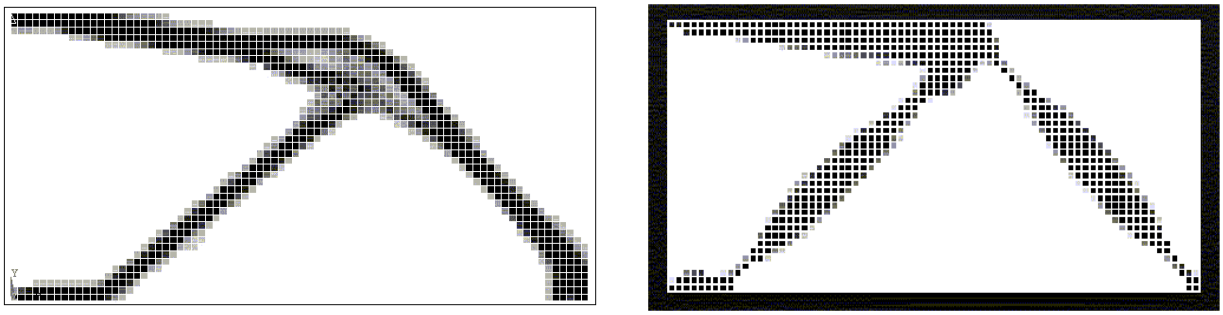


Figura 21 – Resultado obtido com a análise tridimensional do mesmo problema

A próxima figura possibilita uma comparação entre os 2 resultados.



a) Resultado obtido b) Resultado esperado

Figura 22 – Comparação entre os resultados

A semelhança entre os resultados apresentados comprova a eficácia dos métodos aqui utilizados.

Note que o filtro espacial gera uma quantidade maior de densidades intermediárias (elementos de cor cinza). Ao final da busca, portanto, deve-se deixar o programa correr por mais algumas (6 a 8) iterações com o filtro desligado para eliminar esse problema. O resultado obtido por LIMA [9] e mostrado na **figura 23b** mostra o resultado desse processo de correção. Os resultados mostrados a seguir também utilizam esse método.

6.2. Resultados tridimensionais

6.2.1. Mecanismo de impressão

Este mecanismo tem por função transformar um movimento rotativo limitado, como, por exemplo, de um motor de passo, em um movimento linear. Esse sistema é útil, por exemplo, para impressoras matriciais ou *"de ponto"*. A figura a seguir ilustra o domínio utilizado.

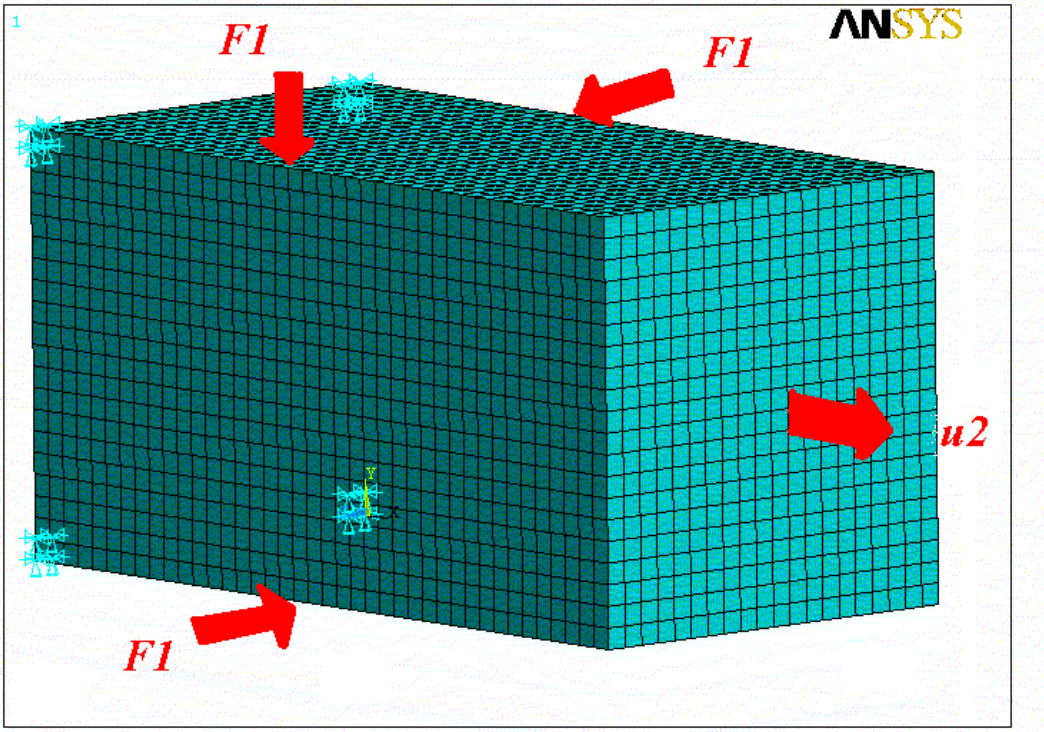


Figura 23 – Domínio para mecanismo de impressão

A malha tem 40x20x20 elementos, totalizando 16.000 elementos. O volume final da estrutura foi limitado em 13% do volume total do domínio inicial. As imagens abaixo ilustram o resultado obtido.

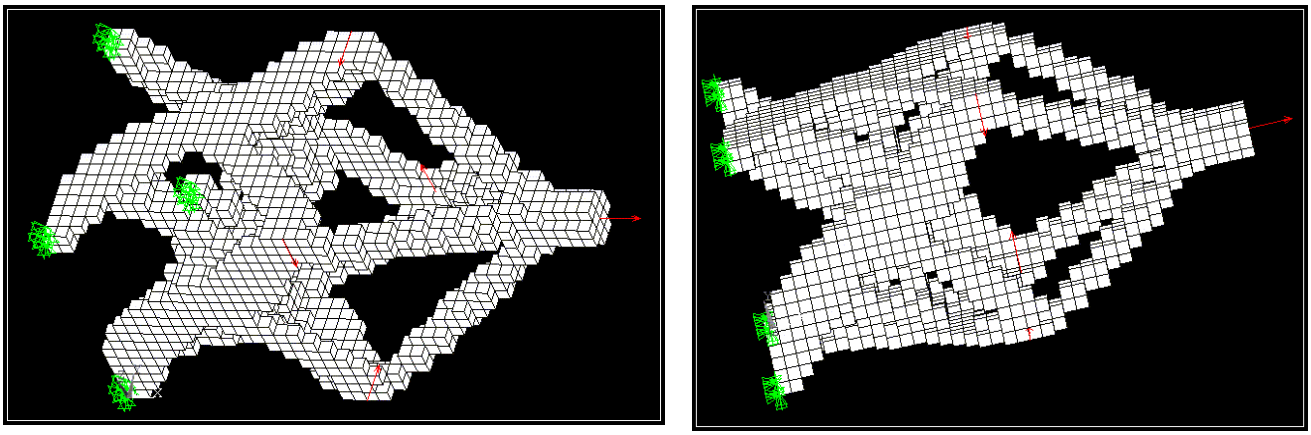


Figura 24 – Resultado obtido para mecanismo de impressão

O programa registra também dados para análise da convergência do processo, através da plotagem de gráficos pelo software comercial *MatLab*, como pode ser observado a seguir:

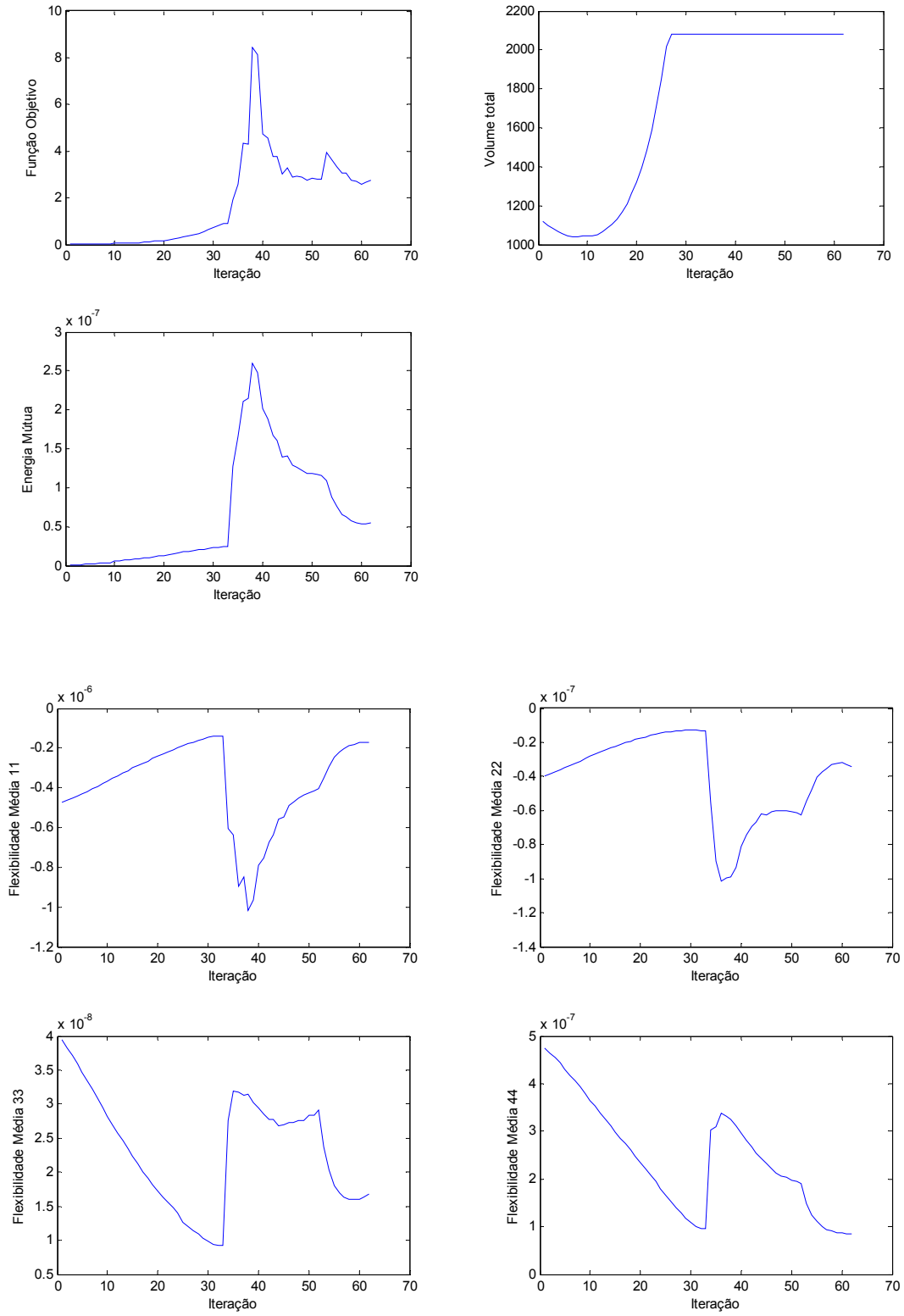


Figura 25 – Curvas de convergência das funções analisadas para o mecanismo de impressão: função objetivo, volume (soma das pseudo-densidades), energia mútua e flexibilidades médias

Percebem-se claramente nestes gráficos dois momentos de mudança brusca na evolução. O primeiro corresponde ao início da aplicação da penalização para pseduo-densidades intermediárias e, simultaneamente, do filtro de malha. O segundo corresponde à interrupção da aplicação do filtro.

A visualização através do *AnSys* permite, ainda, uma análise do funcionamento do mecanismo. As figuras a seguir mostram as distribuições de tensão interna, pelo critério de Von Mises, durante o funcionamento da estrutura.

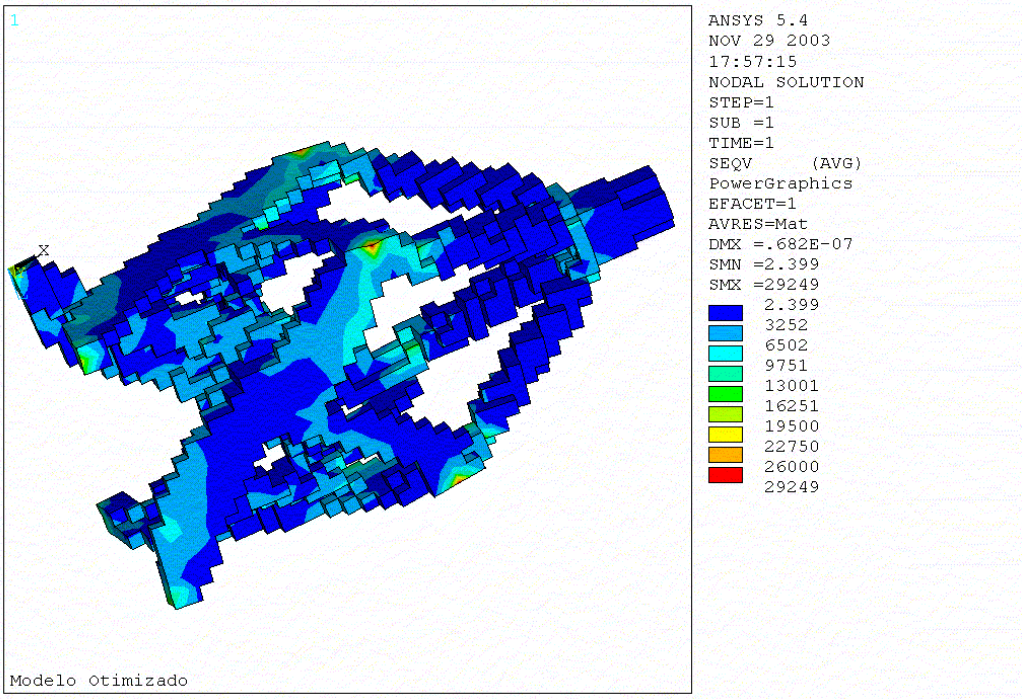


Figura 26 – Tensões internas (Von Mises) na estrutura deformada pelo esforço fornecido F_1

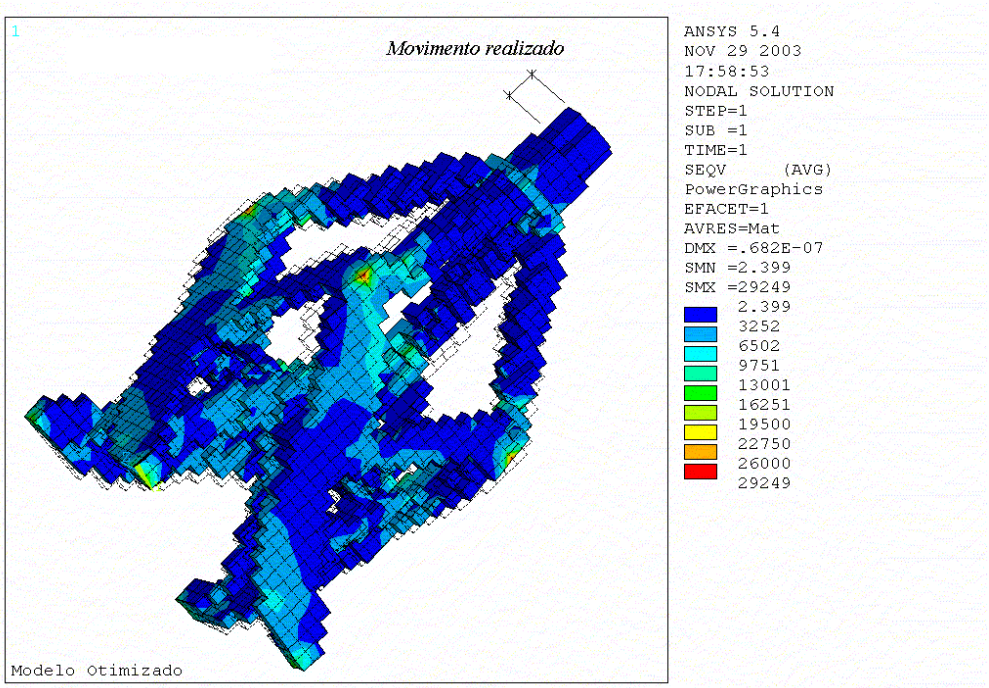


Figura 27 – Sobreposição da estrutura deformada com a forma original, destacando o movimento obtido

Apesar utilizar valores fictícios, análises como essa mostram o comportamento da estrutura, comprovando a eficiência para a função desejada.

6.2.2. Pinça tridimensional

Outro exemplo de simulação realizada é o de uma pinça tridimensional, definida de acordo com o domínio abaixo.

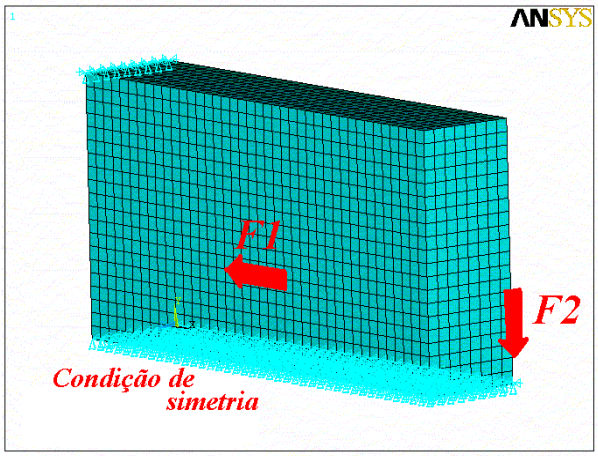


Figura 28 – Domínio para projeto da pinça

O domínio foi divido em 40 x 20 x 8 elementos, num total de 6400 cubos. O volume final foi limitado em 20% do domínio, enquanto a densidade inicial de cada elemento foi de 0,1.

Na estrutura obtida como ótima, percebem-se variações na espessura, de acordo com a posição dos esforços. O resultado da análise é mostrado a seguir.

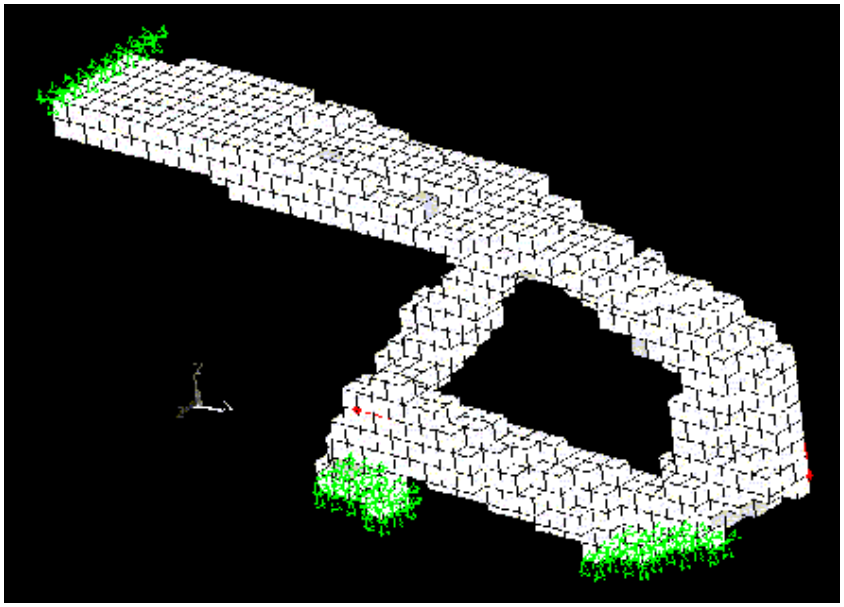


Figura 29 – Pinça tridimensional otimizada, vista 1

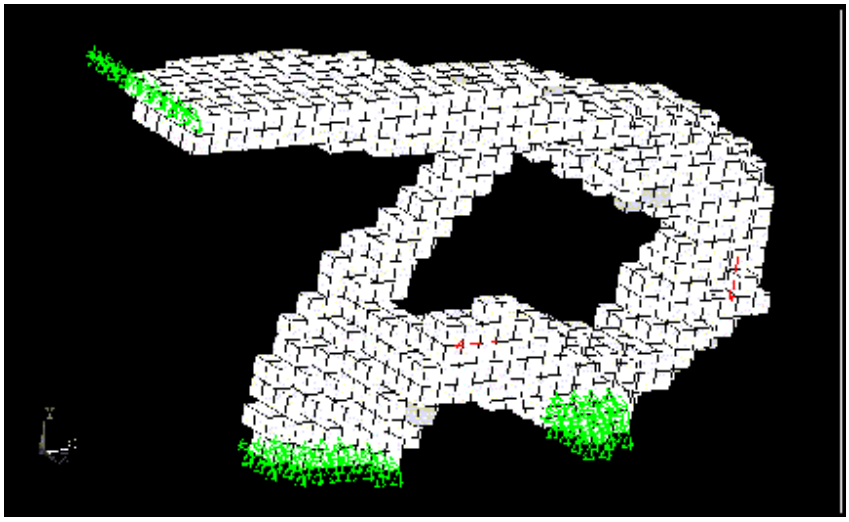


Figura 30 – Pinça tridimensional otimizada, vista 2

As curvas de convergência para essa análise são apresentadas a seguir:

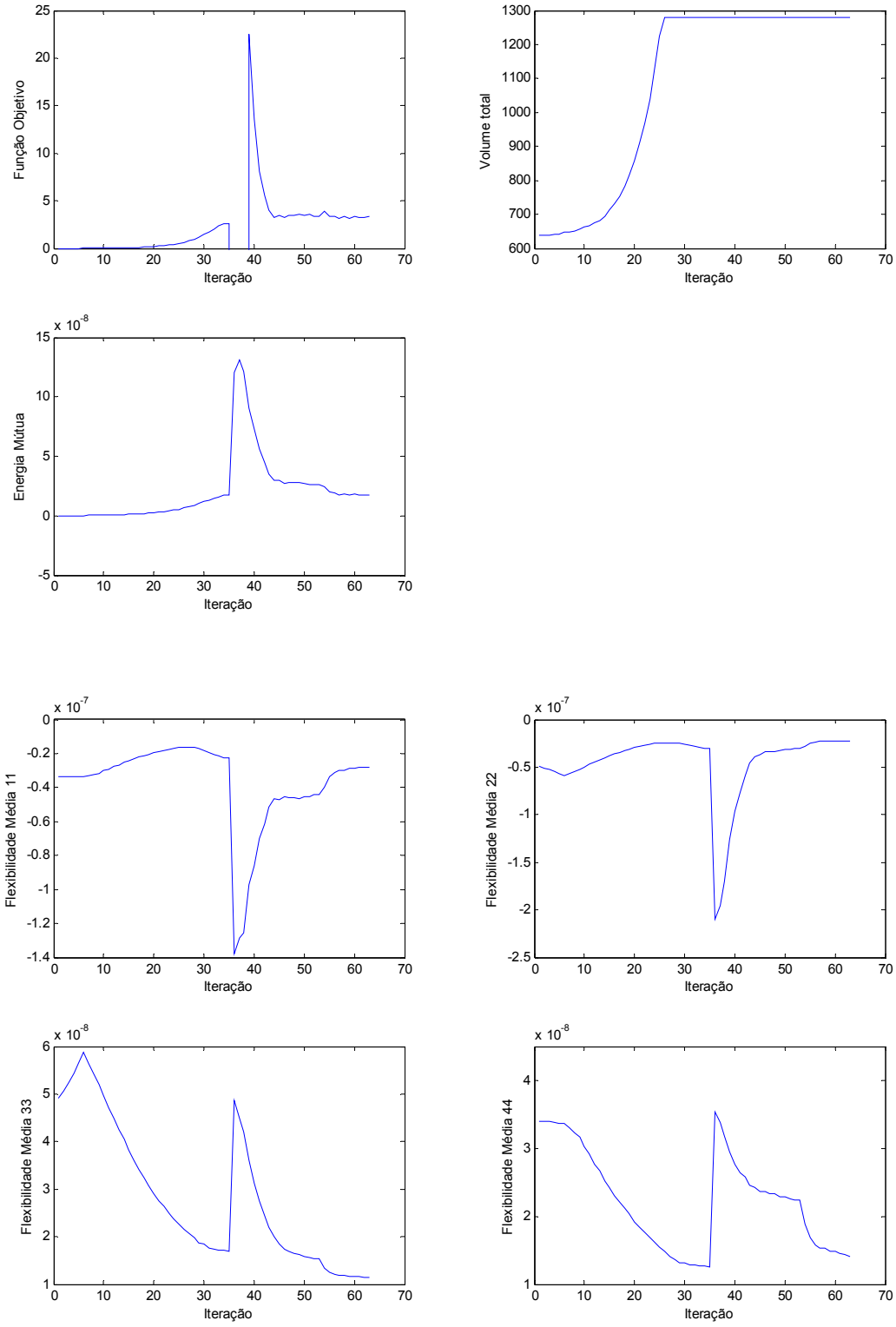


Figura 31 - Curvas de convergência das funções analisadas para a pinça: função objetivo, volume (soma das pseudo-densidades), energia mútua e flexibilidades médias

Mais uma vez, são claros os pontos de mudança no processo, ou seja, aplicação da penalização e do filtro e, no final, desligamento do filtro.

Mais uma vez, foi realizada uma análise do comportamento da estrutura quando submetida ao esforço disponível. O resultado é mostrado abaixo, comparado com a estrutura não deformada.

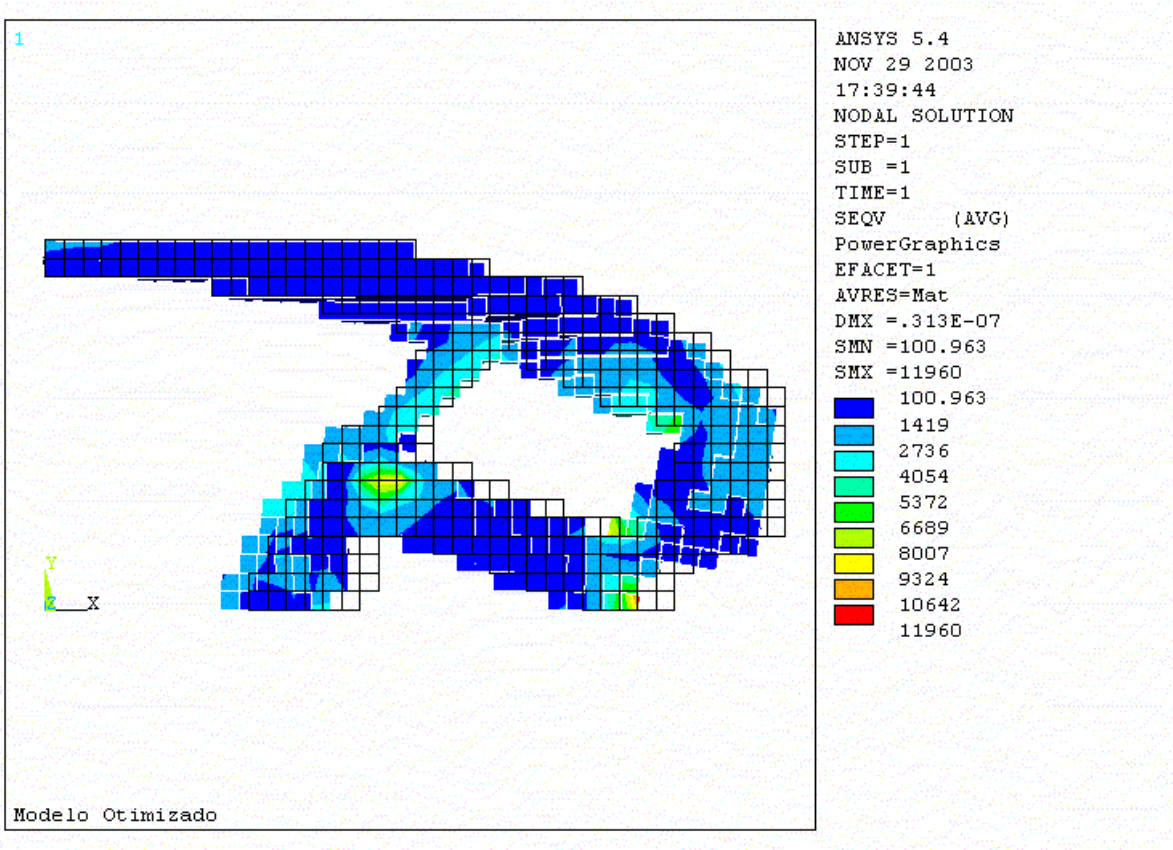


Figura 32 – Análise da deformação da pinça, comparada à forma original

7. Comentários e conclusões

O trabalho fio concluído com sucesso, porém está longe de realmente concluído. Os objetivos principais foram atingidos, comprovando a validade dos métodos utilizados. Métodos estes que vêm se desenvolvendo e se difundindo na comunidade científica nos últimos anos, graças à viabilização da computação como ferramenta para a Engenharia Mecânica.

Algumas ferramentas aqui utilizadas, como o método dos elementos finitos, já criaram suas raízes entre os engenheiros mecânicos, firmando-se como utensílios básicos para a profissão atualmente. Outras, como os conceitos de otimização topológica apresentados, ainda foram pouco exploradas, mas este trabalho mostra que possuem uma base sólida que deve ser aproveitada.

7.1. Continuidade do trabalho

O primeiro passo a seguir seria um estudo da influência, sobre o resultado final, de diversos parâmetros determinados arbitrariamente, como por exemplo os pesos atribuídos às flexibilidades na função objetivo, a amplitude dos limites móveis, a tolerância utilizada nas rotinas iterativas, o raio de abrangência do filtro espacial ou mesmo a limitação de volume.

Estudos como esses foram realizados para o caso bidimensional por LIMA [9], mas alguns efeitos podem ser diferentes numa malha tridimensional.

Alguns aprimoramentos mais imediatos que podem ser realizados são a introdução de recursos como a possibilidade de um mecanismo executar mais de uma função ou de maior arbitrariedade sobre o domínio de projeto, impedindo ou forçando que haja matéria em alguns pontos.

Outras possíveis mudanças que podem ser considerados, talvez em maior prazo, são a possibilidade de se utilizar mais de um material numa mesma estrutura, ou ainda os chamados *Functionaly Graded Materials*, cujas propriedades variam gradualmente. A utilização de variáveis de projeto nos nós da malha e não nos elementos em si também pode levar a resultados interessantes, principalmente por simplificar a análise em alguns pontos, eliminando, por exemplo, a necessidade do filtro espacial.

Lista de referências

1. AFONSO, H.D. **Relatório apresentado à FAPESP: Desenvolvimento de software para o proljeto de peças mecânicas otimizadas usando o método de otimização topológica**; São Paulo, 2000.

2. BENDSOE, M.P.; SIGMUND, O. **Topology Optimization – Theory, Methods and Applications**; Berlim, Springer, 2003.

3. CARVALHO, A.I. **Projeto de um novo tipo de espelho para telescópios – Um método para se obter espelhos esféricos de baixo custo para telescópios**; São Paulo, 2002. Dissertação (Graduação) – Escola Politécnica da Universidade de São Paulo.

4. FELIPELLI, T. **Desenvolvimento de software para projeto de micro-manipuladores**; São Paulo, 2002. Dissertação (Graduação) – Escola Politécnica da Universidade de São Paulo.

5. GALLAGHER, R.H. **Finite element analysis: fundamentals**; New Jersey, Prentice-Hall, Inc 1975.

6. HOWELL, L. L. **Compliant mechanisms**; New York, John Wiley & Sons, 2001.

7. KIKUCHI, N. **Finite Element Methods in Mechanics**; Cambridge, Cambridge University Press, 1985. 327 pp.

8. NISHIWAKI, S.; MIN, S.; YOO, J.; KIKUCHI, N **Optimal Structural Design Considering Flexibilitiy**; Michigan, 2000.

9. LIMA, C.R. **Projeto de mecanismos flexíveis usando o método de otimização topológica**; São Paulo, 2002. 146p. Dissertação (Mestrado) – Escola Politécnica da Universidade de São Paulo.

10. LIPPI, T.N. **Implementação de um software de otimização topológica para estruturas tridimensionais**, São Paulo, 2002. 53p. Dissertação (Graduação) – Escola Politécnica da Universidade de São Paulo.

11. MELLO, L.A.M **Desenvolvimento de um software de otimização topológica aplicado a peças e estruturas num domínio bidimensional**, São Paulo, 2002, 74p. Dissertação (Graduação) – Escola Politécnica da Universidade de São Paulo

12. PRESS, W.H.; TEUKOLSKY, S.A.; VETTERLING, W.T.; FLANNERY, B.P. **Numerical recipes in C**; New York, Cambridge University Press, Inc 1992.

13. SIGMUND, O. **A 99 line topology optimization code written in MatLab**; Dinamarca, 2000.

14. BATHE, K.J. **Finite Element Procedures**; New Jersey, Prentice Hall, 1996.

15. CARDOSO, E.L.; FONSECA, J.S.O.; **Intermediate density reduction and complexity control in the topology optimization**; 1999. 20th Iberian Latin-American Congress on Computational Methods in Engineering, Brasil.

16. COOK, R.D; YOUNG, W.C.; **Advanced Mechanics of Materials**, New York, Macmillan, 1985.

17. DANTZIG, G.B; **Linear Programming and Extensions**, New Jersey, Princeton, 1963.

18. HAFTKA, R.T.; **Element of Structural Optimization**, Boston, Kluwer Academic Publishers, 1996.

ANEXO A Exemplo de modelo de domínio para o software *AnSys 5.4* (mecanismo inversor)

Esse arquivo cria e malha o domínio a ser utilizado, aplica os esforços tidos como entrada do sistema, e escreve o arquivo *database* (*.cdb*) que será lido pelo software de mecanismos flexíveis.

```
!! Parâmetros de malha
taille = 0.01      !!! tamanho da borda dos elementos
young = 71e9
poisson = 0.34
densid = 3.8e3

/PREP7

ET,1,SOLID45

MPTEMP,,,,,,,,
MPTEMP,1,0
MPDATA,EX,1,,young
MPDATA,PRXY,1,,poisson
BLOCK,0,0.8,0,0.4,0,0.01,
TYPE,   1
MAT,      1
REAL,
ESYS,      0

ESIZE,taille,0,

DESIZE,3,2,15,15,28,taille,taille,1,4,

CM,_Y,VOLU
VSEL, , , ,      1
CM,_Y1,VOLU
CHKMSH,'VOLU'
CMSEL,S,_Y

MSHAPE,0,3d
MSHKEY,1
VMESH,_Y1
MSHKEY,0

CMDELE,_Y
CMDELE,_Y1
CMDELE,_Y2

!!! Restrições
NSEL,S,LOC,X,0,0,0
NSEL,R,LOC,Y,0.4,0.4,0
D,all,all,0
ALLSEL

NSEL,S,LOC,Y,0,0,0
D,all,UY,0
D,all,UZ,0
ALLSEL

!!! Esforços
NSEL,S,LOC,X,0,0,0
NSEL,R,LOC,Y,0,0,0
F,all,FX,-1
ALLSEL

CDWRITE,DB,'carga1','cdb',' ', , ,
```

ANEXO B Exemplo de arquivo *database* utilizado como entrada (mecanismo inversor)

Esse é o primeiro dos arquivos que devem ser lidos pelo software de mecanismos flexíveis. Os trechos destacados são pontos verificados para encontrar os dados desejados. Em versões diferentes do software *AnSys*, algumas dessas linhas podem ser escritas de forma diferente, ocasionando problemas de compatibilidade. Notadamente, a versão *AnSys 6.1*, escreve um valor a mais nas linhas referentes às definições dos elementos e suas conectividades, impossibilitando a leitura.

```
/COM,ANSYS RELEASE  5.4    UP19970828        11:08:54    11/16/2003
/PREP7
/NOPR
/TITLE,
*IF,_CDRDOFF,EQ,1,THEN     !if solid model was read in
_CDRDOFF=            !reset flag, numoffs already performed
*ELSE            !offset database for the following FE model
NUMOFF,NODE,  6642
NUMOFF,ELEM,  3200
NUMOFF,MAT ,     1
NUMOFF,TYPE,     1
*ENDIF
ET,     1, 45
NBLOCK,6,SOLID
(3i8,6e16.9)
      1       0       0  .000000000     .400000000
      2       0       0  .000000000
      3       0       0  .000000000     .390000000
      4       0       0  .000000000     .380000000
      5       0       0  .000000000     .370000000
      6       0       0  .000000000     .360000000
      7       0       0  .000000000     .350000000
      8       0       0  .000000000     .340000000
      9       0       0  .000000000     .330000000
     10       0       0  .000000000     .320000000

            (...)

   6632       0       0  .790000000     .290000000    1.000000000E-02
   6633       0       0  .790000000     .300000000    1.000000000E-02
   6634       0       0  .790000000     .310000000    1.000000000E-02
   6635       0       0  .790000000     .320000000    1.000000000E-02
   6636       0       0  .790000000     .330000000    1.000000000E-02
   6637       0       0  .790000000     .340000000    1.000000000E-02
   6638       0       0  .790000000     .350000000    1.000000000E-02
   6639       0       0  .790000000     .360000000    1.000000000E-02
   6640       0       0  .790000000     .370000000    1.000000000E-02
   6641       0       0  .790000000     .380000000    1.000000000E-02
   6642       0       0  .790000000     .390000000    1.000000000E-02
N,R5.3,LOC,     -1,
EBLOCK,18,SOLID
(18i7)
     1     1     1     1     0     0     0     0     8     0     2   162   241    41  3322  3324  3562  3561
     1     1     1     2     0     0     0     0     8     0   162   163   242   241  3324  3325  3601  3562
     1     1     1     3     0     0     0     0     8     0   163   164   243   242  3325  3326  3640  3601
     1     1     1     4     0     0     0     0     8     0   164   165   244   243  3326  3327  3679  3640
     1     1     1     5     0     0     0     0     8     0   165   166   245   244  3327  3328  3718  3679
     1     1     1     6     0     0     0     0     8     0   166   167   246   245  3328  3329  3757  3718
     1     1     1     7     0     0     0     0     8     0   167   168   247   246  3329  3330  3796  3757
     1     1     1     8     0     0     0     0     8     0   168   169   248   247  3330  3331  3835  3796
     1     1     1     9     0     0     0     0     8     0   169   170   249   248  3331  3332  3874  3835
     1     1     1    10     0     0     0     0     8     0   170   171   250   249  3332  3333  3913  3874

            (...)

     1     1     1  3190     0     0     0     0     8     0  3311  3312    52    53  6252  6291  3453  3454
     1     1     1  3191     0     0     0     0     8     0  3312  3313    51    52  6291  6330  3452  3453
```

```
     1     1     1  3192     0     0     0     0     8     0  3313  3314    50    51  6330  6369  3451  3452
     1     1     1  3193     0     0     0     0     8     0  3314  3315    49    50  6369  6408  3450  3451
     1     1     1  3194     0     0     0     0     8     0  3315  3316    48    49  6408  6447  3449  3450
     1     1     1  3195     0     0     0     0     8     0  3316  3317    47    48  6447  6486  3448  3449
     1     1     1  3196     0     0     0     0     8     0  3317  3318    46    47  6486  6525  3447  3448
     1     1     1  3197     0     0     0     0     8     0  3318  3319    45    46  6525  6564  3446  3447
     1     1     1  3198     0     0     0     0     8     0  3319  3320    44    45  6564  6603  3445  3446
     1     1     1  3199     0     0     0     0     8     0  3320  3321    43    44  6603  6642  3444  3445
     1     1     1  3200     0     0     0     0     8     0  3321   161    42    43  6642  3442  3403  3444
    -1
MPTEMP,R5.0, 1, 1,  .000000000    ,
MPDATA,R5.0, 1,EX  ,     1, 1, 7.100000000E+10,
MPTEMP,R5.0, 1, 1,  .000000000    ,
MPDATA,R5.0, 1,PRXY,     1, 1,  .340000000    ,
BFUNIF,TEMP,_TINY
AUTOTS,OFF
NSUBST,        1,        0,        0,OFF
KBC,     0
KUSE,     0
TIME,  .000000000
TREF,  .000000000
ALPHAD,  .000000000
BETAD,  .000000000
DMPRAT,  .000000000
TIMINT,OFF ,STRU
CNVTOL,F   ,-1
CNVTOL,U   ,-1
CNVTOL,M   ,-1
CNVTOL,ROT ,-1
CRPLIM,  .100000000
NCNV,     1,  .000000000    ,     0,  .000000000    ,  .000000000
LNSRCH,OFF
NEQIT,     0
PRED,OFF ,,OFF
ERESX,DEFA
ACEL,  .000000000    ,  .000000000    ,  .000000000
OMEGA,  .000000000    ,  .000000000    ,  .000000000    ,     0
DOMEGA,  .000000000    ,  .000000000    ,  .000000000
CGLOC,  .000000000    ,  .000000000    ,  .000000000
CGOMEGA,  .000000000    ,  .000000000    ,  .000000000
DCGOMG,  .000000000    ,  .000000000    ,  .000000000
IRLF,  0

D,     1,UX  ,  .000000000    ,  .000000000
D,     1,UY  ,  .000000000    ,  .000000000
D,     1,UZ  ,  .000000000    ,  .000000000
D,     2,UY  ,  .000000000    ,  .000000000
D,     2,UZ  ,  .000000000    ,  .000000000
D,   122,UY  ,  .000000000    ,  .000000000
D,   122,UZ  ,  .000000000    ,  .000000000

              (...)

D,   3400,UY  ,  .000000000    ,  .000000000
D,   3400,UZ  ,  .000000000    ,  .000000000
D,   3401,UY  ,  .000000000    ,  .000000000
D,   3401,UZ  ,  .000000000    ,  .000000000
D,   3402,UY  ,  .000000000    ,  .000000000
D,   3402,UZ  ,  .000000000    ,  .000000000
D,   3443,UX  ,  .000000000    ,  .000000000
D,   3443,UY  ,  .000000000    ,  .000000000
D,   3443,UZ  ,  .000000000    ,  .000000000
F,     2,FX  , -1.00000000    ,  .000000000
F,   3322,FX  , -1.00000000    ,  .000000000
/GO
FINISH
```

ANEXO C Listagem parcial do programa

São apresentadas aqui algumas rotinas desenvolvidas para esse software, em linguagem *C*.

```
#define NMAX            2.0e7       // Tamanho dos vetores de alocação esparsa p/ matrix de rigidez
#define RMAX            1.5e-2      // raio de abrangência do filtro
#define minf            0.95        // variação dos limites móveis
#define msup            1.05
#define mllower         0.04        // limites pros limites móveis (min 4% e max 15%)
#define mlupper         0.15

/* Variaves globais */
unsigned long *ija;
double *sa;


// soma valor na matriz esparsa representada por sa e ija
void insere(int i, int j, double v)
{
        unsigned int c1;

        if(i == j) sa[i] += v;    // fim if diagonal
        else
        {
                for(c1=ija[i];ija[c1]<j;c1++);  // posiciona
                sa[c1] += v;
        }
} // fim insere


// Prepara vetores ija e sa para montagem da matriz global
void Prepara(long ksize, int **vizinhos)
{
        long dof,i,n;

        n = ksize+2;
        for(dof=1;dof<=ksize;dof++)
        {
                ija[dof] = n;
                sa[dof] = 0.0;
                for(i=1;vizinhos[dof][i]!=-1;i++)
                {
                        sa[n] = 0.0;
                        ija[n] = vizinhos[dof][i];
                        n++;
                }
        } // fim for dof
        ija[ksize+1] = n;
} // fim PreparaInd


// monta matriz de rigidez global nos vetores externos sa e ija
void MontaGlobal(long ksize,int M,int N,int *LM,int **vizinhos,double *x,double p,float ***elementos)
{
        unsigned long i,j,k;
        double valor;
        int *dofs;

        dofs = ivector(1,24);

        //zera a matriz de rigidez global
        Prepara(ksize,vizinhos);

        for(k=0;k<M;k++) // para cada elemento
        {
                for(i=0;i<24;i++)
                        dofs[i+1] = LM[24*k+i];
                for(i=1;i<=24;i++)
                        if( dofs[i] != 0 )
```

```
                                        for(j=1;j<=24;j++)
                                                if( dofs[j] != 0 )
                                                {
                                                        valor = pow(x[k],p)*elementos[k][i][j];
                                                        insere(dofs[i],dofs[j],valor);
                                                } // fim if ambos os dofs não restritos
        } // fim for cada elemento

        free_ivector(dofs,1,24);
} // Fim MontaGlobal


// Função que atualiza as variáveis pelo critério da optimalidade e retorna o volume otimizado
double OC(double *x,double *xnew,double *grad,double *xlow,double *xup,float v,int n)
{
        int i;
        double volume;
        double b, l1, l2, lm;

        l1 = 0.0;
        l2 = 1.0e5;
        while( ( ((l2-l1)/(l2+l1))>EPS ) && (l2>EPS) )
        {
                volume = 0.0;
                lm = (l1+l2)/2;
                for(i=0;i<n;i++)
                {
                        b = grad[i]/lm;
                        if(b<=0) b = EPS;
                        else b = pow(b,0.3);            // b = min(x*b,x*EPS);
                        b = x[i]*b;

                        if(b>xup[i]) b = xup[i];
                        else if(b<xlow[i]) b = xlow[i];         // xlow < b < xup

                        if(b>EPS) xnew[i] = b;                  // xnew = max(b, EPS);
                        else xnew[i] = EPS;

                        volume += xnew[i];
                } // fim for cada elemento

                if( volume>(v*n) ) l1 = lm;
                else l2 = lm;
        } // fim while principal

        return(volume);
} // fim OC


// Função que calcula os limites móveis
void limites(int M, double *sign1, double *sign2, double *sign3, double *ml, double *x, double *xupper, double *xlower)
{
        int i;

        for (i=0; 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;
                else if (ml[i] < mllower) ml[i] = mllower;

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

                if (xlower[i] < EPS) xlower[i] = EPS;
                if (xupper[i] > 1.0) xupper[i] = 1.0;
        } // fim for i=1:M
} // fim limites
```

```
// função que aplica o filtro
void filtro(int M, float *cx, float *cy, float *cz, double *bu, double *bl, double radius)
{
	int i, j, k;
	double fac, sum, somatu, somatl, r;
	double *flow, *fup;

	flow = (double *)malloc(M*sizeof(double ));
	fup = (double *)malloc(M*sizeof(double ));

	for(i=0; i<M; i++)	// calcula filtro para cada elemento
	{
		sum = 0;
		somatu = 0;
		somatl = 0;

		for(j = 0; j < M; j++)
		{
			if(i != j)
			{
				r = distancia(cx[i],cy[i],cz[i],cx[j],cy[j],cz[j]);
				if(r < radius)
				{
					fac = (radius - r)/radius;

					sum += fac;
					somatu += (bu[j] * fac);
					somatl += (bl[j] * fac);
				} // fim if r< RMAX
			}// fim if i!=j
		} // fim for j

		fup[i] = (bu[i] + somatu) / (1 + sum);
		flow[i] = (bl[i] + somatl) / (1 + sum);
	} // fim for calcula filtro

	for(i=0;i<M;i++) // atualiza limites
	{
		if(flow[i]>=0.0) bl[i] = flow[i]; else bl[i] = 0.0;
		if(fup[i]<=1.0) bu[i] = fup[i]; else bu[i] = 1.0;
	} // fim for atualiza limites

	free(flow);
	free(fup);
} // fim filtro
```

ANEXO D Exemplo de arquivo de saída a ser lido no *AnSys* (mecanismo inversor)

Esse é um exemplo de arquivo gerado pelo software de mecanismos flexíveis para ser lido pelo AnSys e assim mostrar a solução encontrada.

```
/SHOW

/UNITS,CGS
/TITLE,Modelo Otimizado
/PREP7
ET,1,SOLID45

MPTEMP,,,,,,,,
MPTEMP,1,0
MPDATA,EX,1,,71000000000.000000
MPDATA,PRXY,1,,0.340000

MPDATA,EX,2,,71000000000.000000
MPDATA,PRXY,2,,0.340000

MPDATA,EX,3,,71000000000.000000
MPDATA,PRXY,3,,0.340000

MPDATA,EX,4,,71000000000.000000
MPDATA,PRXY,4,,0.340000

MPDATA,EX,5,,71000000000.000000
MPDATA,PRXY,5,,0.340000

MPDATA,EX,6,,71000000000.000000
MPDATA,PRXY,6,,0.340000

N,1,0.000000,0.400000,0.000000
N,2,0.000000,0.000000,0.000000
N,3,0.000000,0.390000,0.000000
N,4,0.000000,0.380000,0.000000
N,5,0.000000,0.370000,0.000000
N,6,0.000000,0.360000,0.000000                    !!! nós e coordenadas
N,7,0.000000,0.350000,0.000000
N,8,0.000000,0.340000,0.000000
N,9,0.000000,0.330000,0.000000

(...)

N,6632,0.790000,0.290000,0.010000
N,6633,0.790000,0.300000,0.010000
N,6634,0.790000,0.310000,0.010000
N,6635,0.790000,0.320000,0.010000
N,6636,0.790000,0.330000,0.010000
N,6637,0.790000,0.340000,0.010000
N,6638,0.790000,0.350000,0.010000
N,6639,0.790000,0.360000,0.010000
N,6640,0.790000,0.370000,0.010000
N,6641,0.790000,0.380000,0.010000
N,6642,0.790000,0.390000,0.010000
```

```
REAL,1

/CMAP,FILE,CMAP
/COLOR,OUTL,15
E,2,162,241,41,3322,3324,3562,3561
EMODIF,1,MAT,1
/COLOR,ELEM,15,1
E,162,163,242,241,3324,3325,3601,3562
EMODIF,2,MAT,1
/COLOR,ELEM,15,2
E,163,164,243,242,3325,3326,3640,3601
EMODIF,3,MAT,1
/COLOR,ELEM,15,3

(...)

E,3319,3320,44,45,6564,6603,3445,3446
EMODIF,3198,MAT,6
/COLOR,ELEM,0,3198
E,3320,3321,43,44,6603,6642,3444,3445
EMODIF,3199,MAT,6
/COLOR,ELEM,0,3199
E,3321,161,42,43,6642,3442,3403,3444          !!! Elementos, conectividades, materiais e cores
EMODIF,3200,MAT,6
/COLOR,ELEM,0,3200

FINISH                                        !!! Fim de programa
```