UNIVERSIDADE FEDERAL DO PARANÁ

DIEGO DOIN HOEPFNER

DETERMINAÇÃO DA CONFIGURAÇÂO ÓTIMA DE SENSORES PARA RECONSTRUÇÃO DE CARGAS UTILIZANDO ALGORITMOS GENÉTICOS

CURITIBA

2016

DIEGO DOIN HOEPFNER

DETERMINAÇÃO DA CONFIGURAÇÂO ÓTIMA DE SENSORES PARA RECONSTRUÇÃO DE CARGAS UTILIZANDO ALGORITMOS GENÉTICOS

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Mecânica, Área de Concentração em Mecânica dos Sólidos e Vibrações, Setor de Tecnologia, Universidade Federal do Paraná, como parte das exigências para a obtenção do título de Mestre em Engenharia Mecânica.

Orientador: Prof. Dr. Jucélio Tomás Pereira.

CURITIBA

2016

H694d Hoepfner, Diego Doin
Determinação da configuração ótima de sensores para reconstrução de cargas utilizando algoritmos genéticos / Diego Doin Hoepfner. – Curitiba, 2016.
105 f. : il. color. ; 30 cm.

Dissertação - Universidade Federal do Paraná, Setor de Tecnologia, Programa de Pós-Graduação em Engenharia Mecânica, 2016.

Orientador: Jucélio Tomás Pereira.
Bibliografia: p. 88-90.

1. Pressão. 2. Otimização estrutural. 3. Medidores de tensão. 4. Algoritmos genéticos. I. Universidade Federal do Paraná. II.Pereira, Jucélio Tomás. III. Título.

CDD: 624.172

TERMO DE APROVAÇÃO

DIEGO DOIN HOEPFNER

DETERMINAÇÃO DA CONFIGURAÇÂO ÓTIMA DE SENSORES PARA RECONSTRUÇÃO DE CARGAS UTILIZANDO ALGORÍTMOS GENÉTICOS

Dissertação aprovada como requisito parcial à obtenção do grau de Mestre em Engenharia Mecânica do Curso de Mestrado do Programa de Pós-Graduação em Engenharia Mecânica da Universidade Federal do Paraná, na área de concentração Mecânica dos Sólidos e Vibrações.

Banca Examinadora:

Prof. Dr. Marco Antonio Luersen
UTFPR

Prof. Dr. Carlos Alberto Bavastri
UFPR

Prof. Dr. Jorge Luiz Erthal
UFPR

Curitiba, 15 de março de 2016.

Dedicado à minha amada esposa Elenice.

AGRADECIMENTOS

Ao Grande Arquiteto Do Universo, fonte inesgotável de sabedoria, por direcionar meu caminho.

A minha esposa Elenice Cavalheiro da Silveira, pela motivação que me deu, por sua compreensão nos momentos de ausência e pela dedicação e companheirismo durante a realização deste trabalho.

Aos meus pais Bernardo e Elizabeth, pelo exemplo de retidão e pelo suporte que me deram.

Ao Prof. Dr. Jucélio Tomás Pereira pela orientação deste trabalho e por todos os ensinamentos e conhecimento compartilhados.

Ao meu chefe Bruno Santos Lucena, por compreender a importância deste projeto para a companhia e apoiar a sua execução.

A empresa CNH Industrial Latin America Ltda., por prover os meios para a realização deste trabalho, fornecendo os equipamentos, softwares e materiais necessários.

Ao Guilherme Escorsin Roque, pelo auxílio na execução dos testes físicos.

A todos que de alguma maneira contribuíram para a realização deste trabalho.

“O que sabemos é uma gota, o que ignoramos, um imenso oceano. A admirável disposição e harmonia do universo não pôde senão sair do plano de um ser omnisciente e onipotente. ”

(Isaac Newton).

RESUMO

O corrente trabalho busca elaborar uma metodologia simples, rápida e confiável para a identificação de carregamentos mecânicos em um componente em operação, utilizando-o como seu próprio transdutor. A metodologia é baseada na utilização de uma técnica de regressão linear a partir de um modelo de elementos finitos com carregamentos unitários e deformações medidas com extensômetros no componente de estudo. Um dos pontos chave para a obtenção da mais fiel reconstrução de carregamentos possível é a seleção do número de extensômetros e seu posicionamento no componente em estudo. Para a determinação desta localização otimizada, utiliza-se o conceito de projeto experimental ótimo, o qual busca a minimização da variância e covariância dos coeficientes a se determinar. Alguns dos critérios de projeto experimental ótimo mais citados na literatura são o A-ótimo, o D-ótimo e o E-ótimo. O critério adotado neste trabalho foi o citado pela literatura como o de maior relevância para o caso de reconstrução de cargas, o D-ótimo, o qual minimiza o determinante da matriz de dispersão do modelo. Esta minimização utiliza geralmente os algoritmos de troca de Fedorov ou multiplicativos, os mais usuais para projeto D-ótimo. Estes algoritmos, em geral, não são capazes de garantir um ótimo global para o projeto, além de muitas vezes sofrerem com problema de velocidade de convergência. Para contornar estes problemas decidiu-se adotar neste trabalho um algoritmo genético. Este algoritmo foi implementado na linguagem Python e com interface com o software nCode®. Outro algoritmo também desenvolvido neste projeto, em conjunto com um modelo de elementos finitos obtido com o software HyperMesh®, foi utilizado para gerar o domínio das variáveis de projeto, que consistem em posições e orientações de extensômetros virtuais posicionados sobre uma porção do contorno do componente em estudo. Desta forma, espera-se obter um conjunto de extensômetros em posições e orientações determinadas que minimizam o determinante da matriz de dispersão. Para validar a eficiência dos algoritmos gerados foram realizados testes virtuais para o estudo dos parâmetros do algoritmo genético. Testes físicos com dois componentes distintos foram também realizados com o objetivo de validar a metodologia. Ao final do trabalho verificou-se que o algoritmo genético proposto é capaz de atingir valores próximos ao mínimo global da função objetivo com maior constância e estabilidade do que o algoritmo de troca de Fedorov, apesar de sofrer com um custo computacional mais alto. Os testes físicos realizados evidenciaram que uma configuração de extensômetros otimizada pelo critério D-ótimo é capaz de reduzir a variância dos carregamentos reconstruídos.

Palavras-chave: Reconstrução de cargas. Projeto D-ótimo. Seleção ótima de extensômetros. Algoritmos genéticos. Algoritmo de troca de Fedorov.

ABSTRACT

The current work aims to develop a simple, fast and reliable methodology for the identification of mechanical loading on a component in operation, using it as its own transducer. The methodology is based on the usage of a linear regression technique from a finite element model with unitary loads and measured deformations with strain gauges in the component. One of the key points for obtaining the most accurate load reconstruction as possible is the selection of the number of strain gauges and their placement on the component under study. For the determination of the optimized location, the concept of optimal experimental design is used, which seeks to minimize the variance and covariance of the coefficients to be determined. Some of the optimal experimental design criteria most frequently cited in the literature are the A-optimal, D-optimal and E-optimal. The criterion used in this study was cited in the literature as the most relevant for the case of loads reconstruction, the D-optimal, which minimizes the determinant of the model´s scattering matrix. This minimization usually uses the Fedorov´s exchange algorithm or the multiplicative algorithms, the most common for D-optimal design. These algorithms generally are not able to secure a global optimum for the project, and often suffer with convergence speed problems. To work around these problems, it was decided to adopt in this work a genetic algorithm. This algorithm was implemented in Python and interfaced with the nCode® software. Another algorithm also developed in this study, in conjunction with a finite element model obtained with HyperMesh® software, was used to generate the design domain variables, which consist of certain positions and orientations of virtual gauges over a portion of the component's contour under study. Thus, it is expected to obtain a set of strain gages in certain positions and orientations that minimize the determinant of the scattering matrix. To validate the efficiency of the algorithms generated, virtual tests were performed to the study of the genetic algorithm parameters. Physical testing with two different components was also conducted to validate the method. At the end of the work it was found that the genetic algorithm proposed is able to achieve values close to the global minimum of the objective function with greater consistency and stability than the Fedorov exchange algorithm, although suffer from a higher computational cost. The physical tests carried out showed that an optimized strain gauge configuration by D-optimality criteria is able to reduce the variance of the reconstructed loads.

Keywords: Load reconstruction. D-optimal design. Optimal selection of strain gauges. Genetic algorithms. Fedorov´s exchange algorithm.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

®	Marca registrada
×	Produto vetorial

Alfabeto Latino

a_{ij}	Componente da matriz de influência
$[A]$	Matriz de influência
$[A]^T$	Matriz de influência transposta
$[A]^{-1}$	Inversa da matriz de influência
$[\mathrm{A}]^*$	Matriz de influência ótima
$[\mathrm{A}]_N$	Qualquer matriz de influência possível dentro do domínio de candidatos
$[A_+]$	Matriz de influência aumentada
C	Critério de convergência do algoritmo genético
D_j	Determinante da matriz de dispersão para o cromossomo j na geração corrente
$\overline{D_j}$	Determinante da matriz de dispersão para o cromossomo j na geração anterior
$det()$	Determinante de uma matriz
e	Vetor de orientação proveniente do algoritmo gerador de domínio
$\{f\}$	Vetor de carregamentos generalizados
$\hat{l}$	Componente na direção x do vetor normal elementar
$\lvert\hat{l}\rvert$	Valor absoluto da componente na direção x do vetor normal elementar
$\hat{m}$	Componente na direção y do vetor normal elementar
$\lvert\hat{m}\rvert$	Valor absoluto da componente na direção y do vetor normal elementar
$[M]$	Matriz de informação
$[M]^{-1}$	Matriz de dispersão
$[M_+]^{-1}$	Matriz de dispersão aumentada
$\lvert M^{-1}\rvert$	Determinante da matriz de dispersão
$\lvert {M_+}^{-1}\rvert$	Determinante da matriz de dispersão aumentada

$\hat{n}$	Componente na direção y do vetor normal elementar
$\lvert\hat{n}\rvert$	Valor absoluto da componente na direção y do vetor normal elementar
$R(j)$	Expectativa de seleção para o cromossomo no ordenamento j
$[S^2(f)]$	Matriz variância-covariância para as estimativas de carregamentos
$tr()$	Traço de uma matriz
U	Conjunto de posições de extensômetros
$\{y\}$	Vetor de um novo extensômetro candidato a ser adicionado na matriz de dispersão
$\{y\}^T$	Vetor transposto de um novo extensômetro candidato a ser adicionado na matriz de dispersão
$\{z\}$	Vetor de um extensômetro candidato a ser removido da matriz de dispersão
$\{z\}^T$	Vetor transposto de um extensômetro candidato a ser removido da matriz de dispersão

Alfabeto Grego

β	Domínio de candidatos
γ_{xy}	Deformação elementar cisalhante no plano xy
$\{\varepsilon\}$	Vetor de deformações medidas com extensômetros
ε_a	Deformação na superfície do elemento em uma determinada orientação
ε_x	Deformação elementar na direção x
ε_y	Deformação elementar na direção y
η	Vetor normal de um elemento
θ	Incremento angular
ϑ	Conjunto de orientações de extensômetros
σ^2	Variância na medida de deformações
φ	Função objetivo
ω	Vetor de orientação final para o extensômetro de ângulo 0º.

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos, na indústria, tornou-se um procedimento padrão de projeto a análise virtual do componente que se deseja projetar e construir. Para tanto, utilizam-se softwares implementados com métodos de solução aproximada de problemas de valores no contorno, como o Método de Elementos Finitos (MEF). O MEF é amplamente difundido para a estimativa de tensões e deformações em componentes de geometria complexa, possibilitando previsões de durabilidade dos produtos e auxiliando no dimensionamento.

O projeto de um componente estrutural utilizando ferramentas CAE *(Computer Aided Engineering)* é tão bom quanto a qualidade de seus dados de entrada. No caso de cálculos estruturais, as principais fontes de erro relacionadas à modelagem do problema físico para o modelo virtual são:

- Os erros introduzidos na solução do método de elementos finitos, divididos em 3 grupos principais por Reddy (2006), os erros de aproximação de domínio, erros de aritmética finita e quadratura e os erros de aproximação da solução;
- Erros associados à modelagem das condições de contorno;
- Erros de aproximação dos carregamentos, os quais nem sempre podem ser obtidos por cálculo (HUNTER 2012).

Desta forma, a habilidade de prever a durabilidade ou a resistência de um componente mecânico ou de uma estrutura depende fortemente do conhecimento dos carregamentos a serem aplicados. Estes carregamentos são, em geral, medidos introduzindo células de carga no sistema ou desenvolvendo transdutores especiais para cada aplicação (WICKHAM *et al.*, 1994).

Entretanto, estes métodos, em geral, alteram a física do problema, ou têm implementação muito complexa. Estas limitações podem ser superadas de uma forma muito barata ao utilizar-se a técnica de reconstrução de cargas, na qual o próprio componente que se deseja medir as cargas é usado como seu transdutor, apenas com a aplicação de extensômetros em pontos específicos da geometria. (ILANKAMBAN *et al.*, 1996)

Para que esta reconstrução de carregamentos seja eficiente, recai-se no problema da definição da posição dos pontos a serem instrumentados com extensômetros no componente, que é solucionado através da utilização do conceito de projeto D-ótimo, o qual minimiza a variância dos carregamentos ao otimizar a configuração de extensômetros (GUPTA, 2013).

Esta dissertação busca apresentar, através de discussão de vários métodos conhecidos, e implementar uma proposta de metodologia para obtenção da configuração ótima de instrumentação, utilizando a técnica de algoritmos genéticos, que produza a menor variância no vetor de carregamentos obtidos através da reconstrução das cargas.

1.1. OBJETIVOS

1.1.1. Objetivo geral

O objetivo deste estudo é desenvolver um algoritmo capaz de determinar os locais e orientações ótimos de instalação de extensômetros em uma estrutura qualquer que se comporta como um sistema linear e é submetida a um conjunto de carregamentos quase-estáticos. A combinação de extensômetros deve possibilitar que a estrutura se torne seu próprio transdutor de forças, permitindo a determinação dos carregamentos suportados durante o trabalho para o qual o componente foi projetado.

1.1.2. Objetivos Específicos

O objetivo geral pode ser detalhado nos seguintes objetivos secundários:

i. Propor uma metodologia para a seleção da configuração de extensômetros que permita reconstruir os carregamentos sofridos por um componente mecânico com a menor variância possível;
ii. Implementar computacionalmente o algoritmo e a metodologia apresentados;
iii. Elaborar e realizar testes físicos que permitam comprovar a eficácia do algoritmo e da metodologia.

1.2. ESTRUTURA DO TEXTO

No corrente capítulo é apresentado o problema de reconstrução de cargas utilizando o próprio componente como transdutor, sua importância e relevância para o desenvolvimento de projetos estruturais. Também são apresentados os objetivos gerais e específicos deste trabalho.

A partir do segundo capítulo têm-se a base teórica do trabalho, onde são explanadas as teorias e histórico de estudos sobre reconstrução de cargas, projeto experimental ótimo e sua aplicação na reconstrução de cargas, bem como os algoritmos de otimização para projeto D-ótimo.

No terceiro capítulo discute-se a metodologia adotada para a geração do domínio de extensômetros candidatos e para a seleção da configuração ótima. O método de validação do algoritmo, também apresentado neste capítulo, é composto por testes físicos e virtuais.

No quarto capítulo são apresentados os algoritmos finais implementados e os resultados dos testes virtuais e físicos. Os resultados virtuais apresentam uma comparação entre o algoritmo genético e o algoritmo de troca de Fedorov. Por sua vez, os resultados físicos de validação do algoritmo genético são apresentados para dois componentes mecânicos.

No quinto capítulo são apresentadas as conclusões do trabalho e discussões para futuros desenvolvimentos e estudos.

Por fim, são apresentadas nos apêndices as listagens dos algoritmos implementados.

2 REVISÃO DA LITERATURA

2.1. MÉTODO DE ELEMENTOS FINITOS

O Método de Elementos Finitos (MEF) é hoje uma das técnicas mais bem-sucedidas e estabelecidas para a solução de problemas complexos, que seriam muito difíceis de serem resolvidos analiticamente. Ele é aplicado com sucesso a diferentes áreas da engenharia como: engenharia civil, engenharia mecânica, engenharia nuclear, engenharia biomédica, hidrodinâmica, condução de calor, geomecânica *etc.* (BARKANOV 2001).

A formulação matemática da física dos problemas requer o conhecimento prévio dos fenômenos envolvidos (por exemplo, as leis da mecânica clássica), e certos artifícios matemáticos, os quais permitem obter relações matemáticas entre as grandezas físicas ou químicas envolvidas, na grande maioria das vezes, através de equações diferenciais.

Dependendo da complexidade do problema, a solução das equações diferenciais que regem o problema pode ser muito difícil de obter analiticamente (como por exemplo a solução de deslocamento de um ponto de uma estrutura de geometria complexa sob determinadas condições de contorno). Em muitos casos, os métodos aproximados de análise fornecem meios alternativos de obter a resposta. Entre os métodos mais populares estão o método das diferenças finitas, os métodos variacionais como os de Rayleigh-Ritz e de Galerkin e o Método de Elementos Finitos (MEF).

A solução da equação diferencial no MEF é realizada dividindo o domínio geometricamente complexo em subdomínios de geometria mais simples, chamados de *elementos finitos*. Para cada elemento, funções de aproximação são obtidas, usando a ideia básica de que qualquer função contínua pode ser representada por uma combinação linear, em geral, de funções polinomiais. Estas funções de aproximação são usualmente chamadas de *funções de interpolação*.

Relações algébricas dos parâmetros não determinados do problema (como o deslocamento em um problema de análise estrutural) são obtidas para o conjunto de elementos de forma a satisfazer as equações que regem a física do problema. Estes parâmetros não determinados são a solução do problema para um número finito e

específico de pontos do domínio, chamados de *nós*. Ao conjunto de nós e elementos que possuem relação de adjacência no domínio dá-se o nome de *malha* (REDDY 2006). A Figura 1 mostra um exemplo de discretização do domínio em nós e elementos.

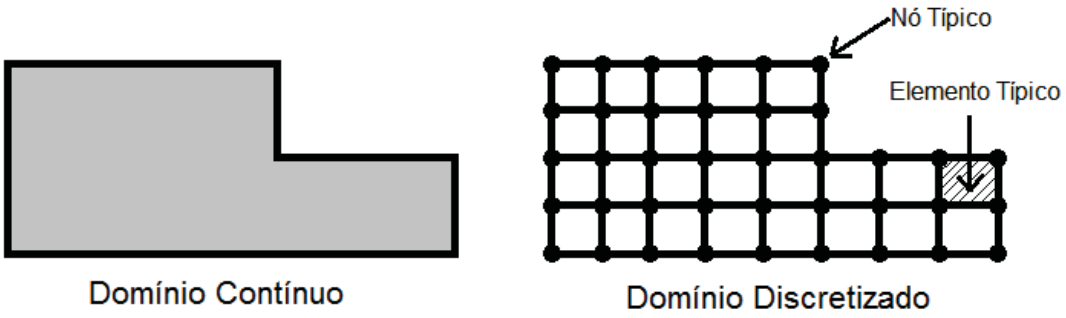


Figura 1 – Exemplo de domínio discretizado em elementos finitos.

O MEF produz muitas equações algébricas simultâneas, que são geradas e resolvidas com o auxílio de computadores. Os resultados são aproximados. Entretanto, a precisão melhora à medida que se aumenta o número de elementos utilizados para discretizar o domínio. Quanto maior o número de elementos maior será o número de equações a serem resolvidas, que exigem maior capacidade de processamento.

Com o aumento da capacidade de processamento dos computadores ao longo dos anos, problemas cada vez mais complexos puderam ser resolvidos, o que ajudou a tornar popular o método (COOK *et al*. 1989).

Existem muitos aplicativos comerciais que oferecem a solução completa para problemas de engenharia, permitindo a discretização da geometria inicial, atribuição de propriedades e condições de contorno para o problema, solução das equações algébricas obtidas pela discretização e pós-processamento dos resultados para que grandezas, como por exemplo a tensão em um problema de análise estrutural, possam ser visualizadas.

Como o intuito deste trabalho não é a solução do problema de análise de tensões/deformações, mas sim o de reconstruir carregamentos utilizando este resultado, optou-se pela utilização do software comercial HyperWorks® para gerar a matriz de influência relacionada a cada componente, como é apresentado nos capítulos seguintes.

2.2. TRANSDUTORES DE FORÇA E RECONSTRUÇÃO DE CARGAS

Enquanto a metodologia de análise por Elementos Finitos tornou-se muito mais sofisticada nos últimos anos, dando aos analistas maior fidelidade em seus modelos em termos de representação geométrica, o usuário ainda precisa fornecer ao modelo as informações de carregamento (HUNTER 2012).

Para muitos componentes, os carregamentos podem ser calculados através de princípios e equações básicas. Hunter (2012) cita alguns exemplos como o cálculo de esforços de uma transmissão, cargas devido a pressão de um gás, cargas de rolamentos, forças devido a desbalanceamentos *etc.*.

Embora estes possam ser cenários de carga muito complexos, eles são bem definidos. Entretanto, há uma grande quantidade de problemas da mecânica estrutural em que o carregamento vem de fontes externas de difícil, quando não impossível, quantificação. Domínios típicos desse fenômeno de carregamentos complexos incluem cargas de estrada de veículos (*on/off road*, aeroespacial, aquáticos). Outras estruturas como prédios e pontes também são submetidos a carregamentos complexos que normalmente são difíceis de quantificar (HUNTER 2012).

Para que um projeto e análise de engenharia possam ser confiáveis e de baixo custo, é imprescindível conhecer, na fase de projeto, os locais e magnitudes das forças transmitidas às estruturas.

O conhecimento das cargas no início do processo de projeto é essencial para aumentar a eficiência do processo de otimização do projeto e análise, o que garante a integridade estrutural do produto. A previsão exata das cargas conduz a uma maior confiabilidade na simulação numérica, como a análise pelo MEF, que, por sua vez, reduz de forma significativa a dependência de ensaios experimentais. Estes, frequentemente caros e demorados (GUPTA, 2013).

Uma vez que nem sempre é possível inserir uma célula de carga, é proposto que a própria estrutura e um conjunto de extensômetros possam ser usados para determinar cada uma das cargas aplicadas (forças e momentos) responsáveis pelas tensões medidas. Em essência, a estrutura torna-se o seu próprio transdutor (MASROOR e ZACHARY, 1991).

Neste método, conhecido como reconstrução de cargas, o componente pode ser utilizado na montagem completa, de forma inalterada. Exceto pela colocação de

extensômetros na superfície, não é necessária qualquer modificação estrutural no componente. As deformações são medidas em posições e orientações específicas pré-determinadas, e a carga é calculada a partir destas leituras de deformação (ILANKAMBAN *et al.*, 1996).

2.2.1. Metodologia de reconstrução de cargas

Uma das metodologias de reconstrução de cargas estáticas mais abordadas pela literatura assume que os carregamentos não ultrapassam o limite de proporcionalidade do material e que os deslocamentos da estrutura são pequenos o suficiente de forma que possa ser considerada linear a relação entre carregamentos aplicados e deformações medidas. Desta forma, o princípio da superposição linear, no qual estados de deformações para cada carregamento independente quando somados formam o estado de deformações final do componente, pode ser considerado válido (WICKHAM *et al.*, 1994).

Segundo Gupta (2013), a deformação em qualquer ponto de uma estrutura pode ser descrita idealmente como uma combinação linear dos carregamentos aplicados. Assim,

$$\{\varepsilon\} = [A]\{f\} \tag{2.1}$$

em que $\{\varepsilon\}$ é um vetor de dimensão $(g \times 1)$ de deformações longitudinais medidas em g diferentes posições na estrutura, $[A]$ é a matriz de influência de dimensão $(g \times n)$, sendo que a_{ij} representa a deformação na posição i devido ao carregamento generalizado e unitário aplicado na posição j, e $\{f\}$ é o vetor de dimensão $(n \times 1)$ das n forças generalizadas aplicadas na estrutura.

É importante salientar que as posições e direções das forças que se deseja reconstruir devem ser previamente determinadas. Isto exige um conhecimento prévio das possibilidades de forças e momentos que podem eventualmente acontecer em determinadas posições dos componentes.

No exemplo visualizado na Figura 2, o componente 1 é solicitado pelo componente 2 através de uma conexão com pino. Assim, o princípio de superposição linear indica que as forças e momentos que podem solicitar o

componente 1 serão sempre uma combinação linear entre as forças nos 3 eixos coordenados e os momentos fletor e torsor.

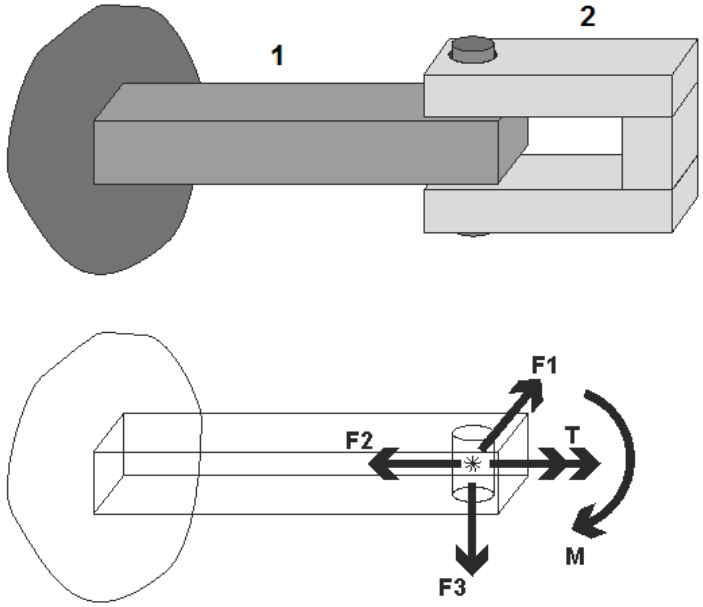


Figura 2 – Exemplo de determinação da posição e direção das possíveis forças que agem em um componente.

Partindo do princípio de que a matriz $[A]$ pode ser obtida por testes físicos ou virtuais, e que $\{\varepsilon\}$ é um conjunto de deformações medidas no teste físico para o qual se deseja reconstruir os carregamentos, o vetor de forças pode ser obtido invertendo-se a matriz de influência. Ou seja,

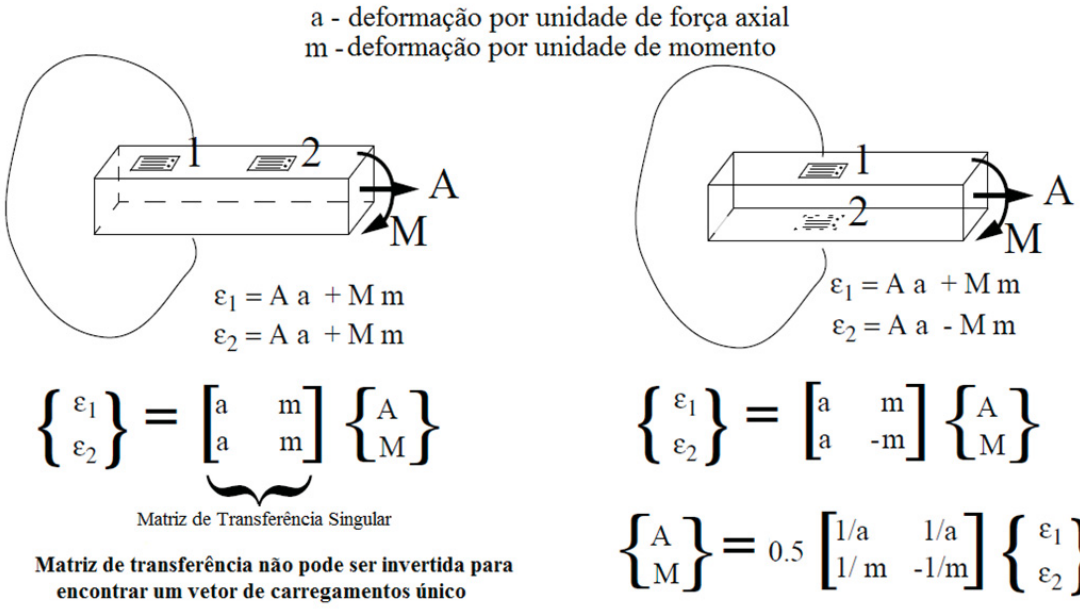
$$\{f\} = [A]^{-1}\{\varepsilon\}\,. \tag{2.2}$$

Entretanto, como a matriz $[A]$ é, em geral, retangular $(g > n)$, já que o número de posições de deformações medidas é, em geral, maior que o de forças, Masroor e Zachary (1990), Wickham ***et al***. (1994) e Bicchi (1992) utilizam o conceito de inversão generalizada à esquerda (BEN-ISRAEL e GREVILLE, 2003) para obter o vetor $\{f\}$:

$$\{f\} = ([A]^T[A])^{-1}[A]^T\{\varepsilon\}\,. \tag{2.3}$$

É fácil perceber, pelo fato de as deformações em cada extensômetro serem uma combinação linear das forças aplicadas na estrutura, que para que seja possível reconstruir um número n de carregamentos, são necessários $g \geq n$ extensômetros. Entretanto, somente esta restrição não é suficiente para garantir que todos os carregamentos desejados possam ser estimados, já que um ou mais extensômetros podem ser redundantes.

Ilankamban *et al* (1996) fornece alguns exemplos de aplicação desta metodologia para uma viga simples e uma combinação de carregamentos conhecidos. A Figura 3 mostra o caso de um carregamento axial e um momento fletor sendo aplicados simultaneamente a uma viga engastada. Na Figura 3.a, a escolha do posicionamento dos extensômetros gera uma matriz de influência singular, o que impossibilita a sua inversão, que significa que as posições dos dois extensômetros não são independentes para os carregamentos aplicados. Na Figura 3.b este problema é resolvido, já que o extensômetro inferior à viga passa a ter leitura oposta à do carregamento superior.



(a) (b)

Figura 3 – Carregamento axial e momento fletor em uma viga. a) matriz de influência singular. b) matriz de influência inversível.
Fonte: modificado de Ilankamban *et al* (1996).

Conforme mencionado, a matriz de influência $[A]$ pode ser obtida a partir de testes físicos ou através de um modelo de elementos finitos. Para tanto, aplica-se um carregamento unitário em cada uma das direções que se deseja reconstruir, e

medem-se as deformações resultantes nos pontos e orientações previamente determinados (configuração candidata). Estas deformações são obtidas utilizando extensômetros ou pela leitura direta após a análise com o modelo virtual. Desta forma, é possível obter para cada um dos carregamentos unitários, a coluna correspondente da matriz $[A]$ (HUNTER, 2012).

2.2.2. Estimativas dos erros e projeto ótimo de experimentos

Segundo Masroor e Zachary (1990) e Kutner *et al.* (2004), na prática, o vetor de deformações $\{\varepsilon\}$ contém erros de medição. Se os erros nas medições de deformação são independentes e identicamente distribuídos (o que normalmente ocorre em medições com extensômetros) e o desvio padrão de cada um deles é σ, a matriz de variância-covariância para as estimativas de carga, $[S^2(f)]$, pode ser escrita na forma (MASROOR e ZACHARY, 1990)

$$[S^2(f)] = \sigma^2([A]^T[A])^{-1}, \tag{2.4}$$

sendo $[S^2(f)]$ uma matriz $(n \times n)$ e σ^2 a variância na medida de deformações.

Os termos diagonais s_{ii} $(i = 1,2, \dots, n)$ de $[S^2(f)]$ são a variância da estimativa de força f_i; os demais termos s_{ij} $(i \neq j)$ da matriz são a covariância entre as estimativas de força f_i e f_j.

Como pode ser percebido na equação 2.4, inicialmente é possível reduzir a variância-covariância das estimativas de força minimizando a variância na medida de deformações. Entretanto, este erro está ligado diretamente aos extensômetros e dispositivos de aquisição de dados, e estará limitado à resolução e precisão destes.

Segundo Gupta (2013), outra forma de reduzir a variância-covariância nas estimativas de força é através da minimização dos elementos da matriz $([A]^T[A])^{-1}$, conhecida como matriz de dispersão (ANGELO, 2007), a qual depende apenas da configuração de extensômetros escolhida para se fazer a reconstrução dos carregamentos.

Este conceito de minimização da variância baseado na otimização da matriz de dispersão é conhecido como projeto ótimo de experimentos e, segundo O'Brien

(2003), foi desenvolvido por Smith (1918), seguido por Kiefer e Wolfowitz (1959) e Fedorov (1972).

Muitos são os critérios, ou funções objetivo, utilizados para a obtenção de projetos experimentais ótimos. Tendo como referência a bibliografia na área, alguns dos mais discutidos para regressão linear são (WONG, 1993):

- Critério D-ótimo: baseia-se na minimização do determinante da matriz de dispersão, $det\,(([A]^T[A])^{-1})$, ou na maximização do determinante da matriz de informação $det([A]^T[A])$. Este critério foi proposto por Wald (1943). Segundo Aguiar *et al*. (1995), quanto maior o determinante da matriz de informação, mais próximo da ortogonalidade estará a matriz de dispersão. Esta ortogonalidade garante a independência mútua dos coeficientes do modelo, e no caso específico da reconstrução de cargas significa a ortogonalidade entre as colunas da matriz $[A]$. Desta forma, quando o número de extensômetros for igual ao de carregamentos (matriz $[A]$ quadrada), cada extensômetro responderia a apenas um carregamento. A minimização da matriz de dispersão também reduz a variância e covariância média das estimativas de carregamento, como é discutido no próximo capítulo.

- Critério A-ótimo: baseado na minimização do traço da matriz de dispersão, $tr(([A]^T[A])^{-1})$, minimizando a dispersão média das estimativas dos parâmetros (ANGELO, 2007).

- Critério E-ótimo: minimiza o máximo autovalor da matriz de dispersão $\lambda_{max}\,(([A]^T[A])^{-1})$ (ANGELO, 2007).

A Tabela 1 sumariza os critérios para projeto experimental ótimo discutidos e suas funções objetivo, φ.

Gupta (2013) cita que o critério para a obtenção do projeto experimental ótimo que tem maior relevância para o caso de reconstrução de cargas é o D-ótimo. A utilização deste critério é praticamente unânime para esta aplicação como pode ser visto nos trabalhos de Hunter (2012), Wickham *et al.* (1994) e Gupta (2013).

Tabela 1 – Critérios para projeto experimental ótimo.

Critério	**Função objetivo** $(\boldsymbol{\varphi})$
D	$det\ (([A]^T[A])^{-1})$
A	$tr(([A]^T[A])^{-1})$
E	$\lambda_{max}\ (([A]^T[A])^{-1})$

Fonte: modificado de Angelo (2007).

Uma das vantagens do critério D-ótimo é a sua convergência monotônica, a qual é discutida por Yu (2010) e Gao *et al.* (2014), utilizando um algoritmo para o projeto D-ótimo.

Wickham *et al.* (1994) reportam que, em testes comparativos, aplicando os projetos A e D ótimos, o primeiro distribui mais uniformemente as variâncias entre as estimativas das cargas, mas o segundo atinge melhor o objetivo de otimização na precisão geral do vetor de carregamentos.

2.3. PROJETO ÓTIMO D-ÓTIMO

O conjunto de posições U e orientações ϑ de todos os extensômetros possíveis sobre a superfície de um componente constitui o domínio β do problema de otimização. A combinação de alguns destes extensômetros (número igual ou maior que o de forças que se deseja reconstruir) representa um projeto candidato possível. A partir de cada uma destas combinações possíveis, gera-se uma matriz de influência, $[\mathrm{A}]_N$, cujos componentes são as leituras de deformação de cada extensômetro deste conjunto candidato para cada carregamento que se deseja reconstruir. Esta matriz é utilizada para o cálculo da função objetivo, φ, a ser otimizada. Assim, o problema de otimização associado pode ser escrito como:

$$\text{Buscar } U \text{ e } \vartheta \text{ tal que, } \varphi([\mathrm{A}]^*) = min\left(\varphi([\mathrm{A}]_N)\right), \tag{2.5}$$

em que N representa a *N*-ésima matriz candidata $[\mathrm{A}]$, a_{ij} representa cada um dos seus elementos constituintes e $[\mathrm{A}]^*$ é a matriz de influência da configuração ótima de extensômetros.

Na prática, um número finito de pontos constitui o domínio do problema de otimização, já que as possíveis matrizes de influência, $[\mathrm{A}]_N$, são obtidas através de um número finito de extensômetros posicionados no contorno do componente. O problema de otimização se constitui, portanto, em um problema de otimização de variáveis discretas.

A obtenção de todos as deformações no contorno de um componente é muito difícil de se realizar por testes físicos, já que envolveria medir dados para centenas de extensômetros para cada direção de carregamento possível para a estrutura. Para facilitar a obtenção destas deformações, um modelo de elementos finitos do componente em estudo é utilizado. Um extensômetro é posicionado em cada nó ou centroide de cada face do elemento que pertence ao contorno do componente, em um ângulo discreto em relação ao sistema de coordenadas locais da superfície do elemento (ex. 0º, 15º, 30º, ... ,165º).

A combinação de todos os centroides e/ou nós multiplicados pelo número de ângulos possíveis fornece o tamanho do domínio discreto de busca do algoritmo de otimização (GUPTA, 2013).

Segundo Arora (2004) existem dois tipos básicos de métodos de otimização com variáveis discretas: os enumerativos e estocásticos. Em alguns métodos enumerativos existe a possibilidade de que todo o domínio de candidatos seja varrido, como o algoritmo de troca clássico proposto por Fedorov (1972) utilizado para a obtenção de projetos experimentais ótimos do tipo D-ótimo. Esta abordagem é muito custosa computacionalmente. Por outro lado, muitos algoritmos da classe dos enumerativos usam estratégias e regras heurísticas para reduzir o número de tentativas, como os algoritmos multiplicativos e do tipo *“Cocktail”,* também utilizados para obtenção de projetos experimentais ótimos. Já os métodos estocásticos são, em geral, baseados em fenômenos naturais, como o método de recozimento simulado (*simulated annealing*) ou algoritmos genéticos.

2.3.1. Algoritmos de otimização para projeto D-ótimo

Mandal *et al.* (2014) faz uma revisão bibliográfica dos algoritmos de otimização para projeto D-ótimo mais utilizados nos últimos anos. As metodologias de alguns destes algoritmos são apresentadas a seguir.

2.3.1.1. Algoritmos de troca

Os algoritmos de troca modificam iterativamente o projeto corrente, eliminando pontos de projeto existentes e adicionando novos pontos do espaço de projeto β, minimizando a função objetivo $\boldsymbol{\varphi}$. Algoritmos baseados neste método são apresentados nos trabalhos de Nguyen e Miller (1992), Smucker *et al.* (2011), Gupta (2013) e Liao (2001). Gupta (2013) apresenta o algoritmo de forma a reduzir o número de cálculos e que consiste em:

1. Escolher uma matriz inicial $[A]$;
2. Calcular $[M]^{-1} = ([A]^T[A])^{-1}$;
3. Calcular $|M^{-1}| = det\,(([A]^T[A])^{-1})$;
4. Adicionar uma linha $\{y\}^T$ de um novo candidato à matriz $[A]$ e calcular o novo determinante da matriz de dispersão $\left|{M_+}^{-1}\right|$ através da equação

$$\left|{M_+}^{-1}\right| = \frac{1}{|M^{-1}|\cdot\{y\}^T\cdot[M]^{-1}\cdot\{y\})} \tag{2.6}$$

e a nova matriz de dispersão $[M_+]^{-1}$ através da equação

$$[M_+]^{-1} = [M]^{-1} - \frac{([M]^{-1}\cdot\{y\})\cdot(([M]^{-1}\cdot\{y\})^T)}{((1+\{y\}^T)\cdot[M]^{-1}\cdot\{y\})}; \tag{2.7}$$

5. Remover uma linha $\{z\}^T$ de candidatos da matriz $[A_+]$ e calcular o novo determinante da matriz de dispersão $|M^{-1}|$ através da equação

$$|M^{-1}| = \frac{1}{|{M_+}^{-1}|\cdot(1-\{z\}^T)\cdot[M_+]^{-1}\cdot\{z\})} \tag{2.8}$$

e a nova matriz de dispersão $[M]^{-1}$ através da equação

$$[M]^{-1} = [M_+]^{-1} + \frac{([M_+]^{-1}\cdot\{z\})\cdot(([M_+]^{-1}\cdot\{z\})^T)}{((1-\{z\}^T)\cdot[M_+]^{-1}\cdot\{z\})}; \tag{2.9}$$

Desta forma, o determinante não necessita ser recalculado a cada nova iteração;

6. Se $|M^{-1}|$ for menor que o da iteração anterior, armazenar esta posição de extensômetro;
7. Repetir os passos 4 e 6 até que todo o domínio de candidatos tenha sido testado para todas as linhas da matriz $[A]$.

Mandal *et al.* (2014) cita que alguns problemas comuns deste tipo de algoritmo são sua baixa taxa de convergência e que, frequentemente, não existe garantia de que o resultado encontrado é o ótimo global. Miller e Nguyen (1994) também relatam que este tipo de algoritmo encontra ótimos locais e, quando executados repetidas vezes com inícios randômicos, eventualmente encontram um ótimo global. Além disto, conforme evidenciam Broudiscou *et al* (1996) e Ramos (2011), para um grande número de variáveis (ex. número de extensômetros maior que 30) pode ser muito difícil encontrar a resposta por este método.

2.3.1.2. Algoritmos multiplicativos e do tipo *"Cocktail"*

Nos algoritmos multiplicativos, pesos são atribuídos ao vetor das variáveis de projeto, e estes são atualizados a cada iteração por um fator multiplicativo, de forma que mais peso é adicionado ao candidato de projeto que traz maior ganho à função objetivo (MANDAL *et al.*, 2014). Dette *et al*. (2008) propõe dois algoritmos multiplicativos e provam sua convergência monotônica. Mandal e Torsney (2006) propõe um algoritmo multiplicativo baseado em agrupamentos (*Clusters*). Mandal *et al.* (2014) observa que estes algoritmos, apesar de simples, são em geral de lenta convergência.

Algoritmos do tipo *Cocktail* usam combinações de algoritmos de troca e multiplicativos. Yu (2011) propôs um algoritmo que é uma simples concatenação entre uma iteração do algoritmo de troca, seguido de uma iteração de um algoritmo multiplicativo e uma terceira do algoritmo chamado de Troca do Vizinho Mais Próximo (*Nearest Neighbor Exchange – NNE*). Mandal *et al.* (2014) afirma que este tipo de algoritmo é mais rápido que algoritmos tradicionais.

2.3.1.3. Algoritmos genéticos

Desenvolvido por John Holland e seus colegas da Universidade de Michigan na década de 70, os algoritmos genéticos (AG) imitam a teoria evolucionista proposta por Charles Darwin (MANDAL *et al.* 2014). Os mecanismos específicos do algoritmo usam a linguagem da biologia e sua implementação imita as operações genéticas (ARORA, 2004).

Para o entendimento do algoritmo, algumas definições são necessárias, as quais são adaptadas nesta seção a partir das definições de Mandal *et al.* (2014), Broudiscou *et al.* (1996) e Arora (2004). Estas definições são:

- Gene: é um escalar do vetor de projeto ótimo, no caso da reconstrução de cargas é um dos extensômetros do domínio β, em uma determinada posição e orientação.
- Cromossomo: representa um projeto do sistema (um indivíduo da população), neste caso, uma configuração de extensômetros.
- População: é o conjunto de cromossomos da iteração corrente. O tamanho da população é definido de acordo com a experiência de uso e do tipo de problema a ser resolvido.
- Geração: é cada iteração do algoritmo genético.

O procedimento geral de otimização através de AG consiste em:

1. Obtenção da população inicial através de algum método randômico, em que os genes (número atribuído aos extensômetros possíveis) são representados por números binários ou não, dependendo do tipo de codificação adotada e os cromossomos ou indivíduos são a combinação destes.
2. Calcula-se o valor da aptidão para cada cromossomo, seleciona-se o melhor indivíduo e remove-se o pior para gerar a próxima população. No caso da reconstrução de cargas, a aptidão é calculada de acordo com uma das funções objetivo para projetos experimentais ótimos conforme discutido na seção 2.2.2.

3. Realiza-se a reprodução para transformar a população atual em uma nova geração, através dos mecanismos de seleção, mutação e cruzamento.
    - Seleção: selecionam-se os pares de cromossomos que irão dar origem à nova geração de acordo com suas aptidões, quanto maior a aptidão de um cromossomo maior a probabilidade de ele ser selecionado.
    - Cruzamento: um par de cromossomos é dividido em um ou mais pontos aleatórios e são combinados a primeira parte de um com a segunda parte do outro e vice-versa, conforme exemplificado na Figura 4.
    - Mutação: seleciona-se um fator binário do cromossomo aleatoriamente e realiza-se a permuta de 0 para 1 e vice-versa. No caso de outros tipos de codificação, outros métodos de mutação são utilizados. As quantidades de mutação e cruzamento em cada reprodução podem ser ajustadas para melhorar a performance do algoritmo.
4. Repetem-se os passos 2 e 3 até que um critério de parada ou um número máximo de iterações seja atingido.

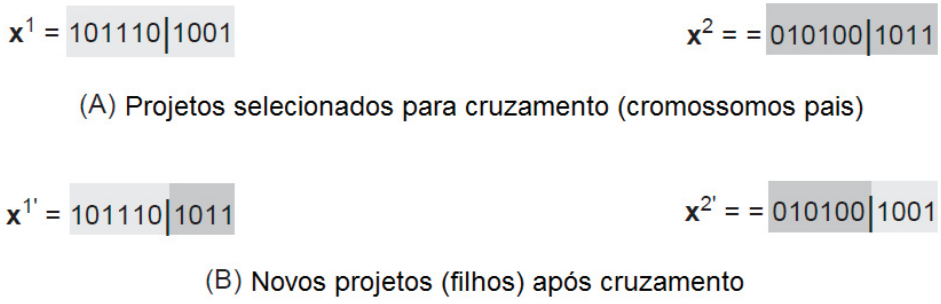


Figura 4 – Operação de cruzamento com um corte.
Fonte: modificado de Arora (2004).

Com base nos trabalhos de Xu (1999) e Arora (2014), a seguir são apresentados alguns pontos-chave na construção de algoritmos genéticos, que determinam sua eficiência na solução de problemas de otimização:

- Codificação - a forma como cada candidato é codificado para processamento no algoritmo. Alguns tipos citados são:
    1. Binária: Neste tipo de codificação cada gene é representado através da sua sequência binária de bits, é um dos mais utilizados e citados na literatura.
    2. Inteira: Quando o espaço de projeto é constituído por números inteiros, dois pontos consecutivos e próximos podem ter representações binárias

muito diferentes. Considere por exemplo que em uma dada iteração os pais selecionados foram os números 15 e 16, cujas representações binárias são 01111 e 10000 respectivamente. Se o cruzamento entre eles ocorrer apenas no primeiro bit, por exemplo, os filhos seriam os binários 11111 e 00000, que correspondem aos valores 31 e 0. Estes pontos são muito distantes da geração anterior, tornando difícil a convergência do algoritmo. Desta forma, para estes casos é mais interessante utilizar o próprio número inteiro ou uma codificação binária cinza.

3. Binária cinza: esta codificação adapta a codificação binária através de um algoritmo específico de forma que entre dois números inteiros consecutivos não exista mais do que 1 bit de diferença.

A Tabela 2 apresenta um comparativo de números inteiros nas 3 representações discutidas.

- Métodos de seleção: é o processo pelo qual os pais são selecionados de acordo com sua aptidão para dar origem à prole. Este passo tem especial importância na convergência do modelo uma vez que ele afeta a diversidade da população. Se o método impuser uma pressão seletiva muito alta sobre os indivíduos mais aptos, os melhores cromossomos serão selecionados com muito mais frequência para formar a nova população, reduzindo a diversidade da população e antecipando a satisfação do critério de parada. Por outro lado, com pouca pressão seletiva cromossomos com menor aptidão serão selecionados mais vezes, aumentando muito a diversidade da população e retardando a parada. Alguns métodos de seleção são (XU,1999):

  1. Seleção por roleta: é o método tradicional proposto por John Holland, que consiste em atribuir uma fatia de uma roleta imaginária para cada cromossomo da população, cujo tamanho do pedaço é proporcional à sua aptidão individual em relação aos outros elementos da população. A roleta é girada N (tamanho da população) vezes. Após cada rotação, o cromossomo apontado pelo marcador da roleta é selecionado. Na Figura 5 é apresentada uma representação gráfica deste processo.
  2. Seleção de acordo com a classificação (*rank*): neste método a aptidão é substituída por uma classificação associada a ela. Os indivíduos são

ordenados de acordo com a sua aptidão e recebem uma classificação de acordo com a ordem em que aparecem. São atribuídas probabilidades de seleção de cada cromossomo baseada no seu "*rank*" e, a partir daí, utiliza-se o método da roleta com as novas probabilidades para selecionar os cromossomos. Dois métodos que se enquadram nesta categoria são o linear de Baker e o de Reeves.

3. Seleção por competição: Este método seleciona um conjunto aleatório de cromossomos da população de dimensão m e seleciona o melhor deles. Em geral, esta dimensão m é 2 e a pressão seletiva aumenta a medida que se aumenta m.

- Modo de substituição da população - a forma como os cromossomos de uma geração são substituídos pelos novos. Alguns tipos citados são:
  1. Substituição de regime permanente: neste método apenas uma fração dos cromossomos menos aptos da população é substituída pelos novos cromossomos gerados durante a reprodução.
  2. Estratégia de evolução $(N + m)$: neste método os m filhos competem com os N pais em uma população intermediária de tamanho $(N + m)$.
  3. Elitismo: é possível que alguns cromossomos de gerações anteriores sejam mais aptos do que os da nova geração. Estes indivíduos seriam perdidos na nova geração ou poderiam ser arruinados nos processos de cruzamento e mutação. Para que esse efeito seja evitado, um número de cromossomos é guardado e inserido na nova geração sempre que estes forem mais aptos.

- Cruzamento - após o pareamento randômico dos pais, as suas cargas genéticas são combinadas para gerar um novo indivíduo. O cruzamento por um ponto que foi explicado anteriormente e exemplificado na Figura 4 é um dos mais comuns. Uma desvantagem deste método é que ele possui o chamado "efeito de cauda", que impossibilita que algumas combinações possam ser obtidas. Outras formas de cruzamento são:
  1. Cruzamento por dois pontos: é similar ao cruzamento por um ponto, mas neste caso 2 pontos são selecionados e o segmento resultante entre os

cortes são invertidos entre os dois pais. Este método permite uma maior possibilidade de combinações.

2. Cruzamento uniforme: neste método qualquer gene em qualquer posição pode ser trocado entre os pais de acordo com uma probabilidade. Apesar de o método ter a habilidade de gerar qualquer combinação possível entre dois pais, ele pode ser disruptivo para a evolução da aptidão no algoritmo.

- Mutação - após o cruzamento ter sido executado, alguns genes são transformados de acordo com uma probabilidade p. Este artifício introduz diversidade na população e previne a convergência prematura. Alguns métodos de mutação discutidos são:
  1. Mutação uniforme: este é o método mais convencional de mutação, em que cada gene tem igual probabilidade de sofrer mutação. Para cada gene, um número aleatório r é gerado e o gene é mutado se $r < p$. Para algoritmos com codificação binária o gene é alterado de 0 para 1 e vice-versa. Para as codificações por números inteiros ou reais, o gene é substituído por outro randomicamente escolhido do domínio de projeto.
  2. Mutação de contorno - este método é similar à mutação uniforme para codificações por números inteiros ou reais. Entretanto, o novo gene gerado será o elemento do contorno superior ou inferior do domínio, com igual probabilidade de ocorrência para ambos.

Tabela 2 – Representações de números de acordo com suas codificações.

Valor Inteiro	Representação binária	Representação binária cinza
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101

Fonte: Modificado de Xu (1999).

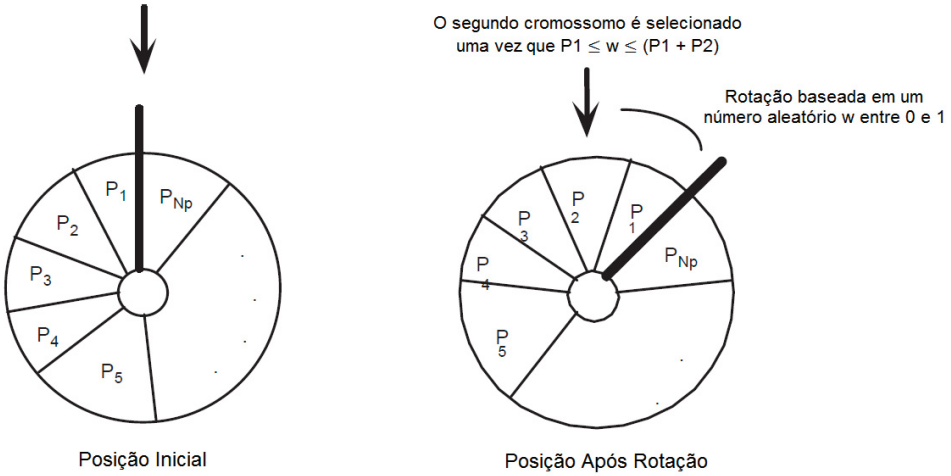


Figura 5 – Exemplo de seleção pelo método da roleta.
Fonte: modificado de Arora (2004).

3 MATERIAIS E MÉTODOS

3.1. METODOLOGIA DA SELEÇÃO DA CONFIGURAÇÃO ÓTIMA DE EXTENSÔMETROS

A metodologia para a seleção dos pontos ótimos para colocação de extensômetros em uma estrutura utilizando-o como o seu próprio transdutor de forças, proposta deste projeto, é similar àquela desenvolvida por Gupta (2013) e Hunter (2012), e é dividida em alguns passos principais. Estes passos são apresentados neste capítulo.

3.1.1. Geração do domínio de extensômetros candidatos

Um modelo de elementos finitos, construído com o auxílio do pré-processador HyperMesh® e analisado utilizando o solver RADIOSS®, permitirá a geração do domínio candidato.

O componente no qual se deseja otimizar as posições de extensômetros é discretizado, obtendo-se assim a malha de elementos finitos. O conjunto contendo os centroides das faces ou nó sobre o contorno dos elementos gerados compõe o domínio de posições possíveis para os extensômetros. Cada ângulo de orientação discreto em relação a um dos eixos do sistema de coordenadas local do elemento ou nó é um extensômetro candidato.

A análise é feita aplicando as devidas condições de contorno no componente de acordo com as interpretações da física do problema. São aplicados casos de carga com carregamentos unitários em cada uma das posições e direções das forças que se deseja reconstruir. Para cada um destes casos de carga o tensor de deformações é obtido para o centroide ou nós de todos os elementos do contorno.

Para o caso de modelos com elementos tridimensionais, as deformações superficiais do componente são obtidas adicionando-se uma camada de elementos de casca ou membrana com espessura muito fina (que não afete de forma significativa a rigidez e dimensões do componente) e com as mesmas propriedades dos elementos tridimensionais. Neste estudo foram adotados elementos lineares do

tipo casca e espessura de 0,01 mm como padrão para todos os modelos tridimensionais.

Na Figura 6 pode ser vista a viga engastada, exemplificada na Figura 2, modelada no pré-processador HyperMesh® com os carregamentos unitários conforme interpretado a partir da peça real. A análise foi executada com o processador RADIOSS® e o arquivo de saída da análise fornece o tensor de deformações centroidais para cada elemento conforme mostrado na Figura 7.

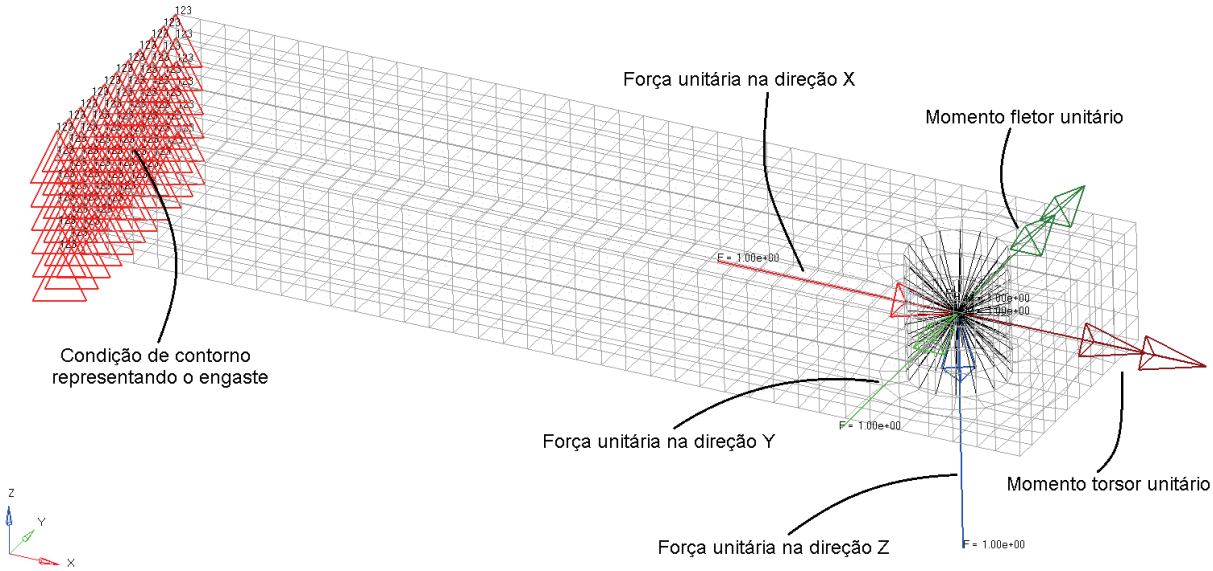


Figura 6 – Modelo de uma viga engastada com carregamentos unitários e condições de contorno.

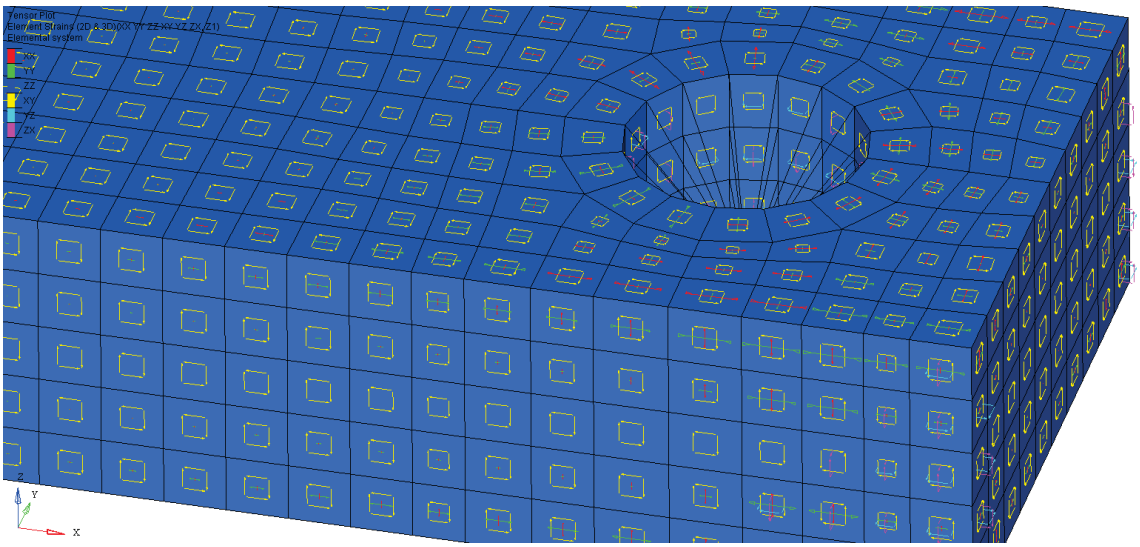


Figura 7 – Representação dos tensores de deformação nos elementos da membrana do componente para um carregamento unitário.

Para facilitar a obtenção das matrizes de influência candidatas, foi utilizado o software nCode®, o qual permite a localização de extensômetros virtuais no modelo. Estes fornecem o pós-processamento da resposta de deformação em uma posição e ângulo específico do modelo para um dado carregamento, permitindo a importação direta destes valores no algoritmo de otimização desenvolvido.

O cálculo das deformações $\varepsilon_a(\theta)$ em cada uma das direções discretas θ de extensômetros sobre cada elemento é calculado pelo nCode® através do tensor de deformações resultante da análise e o círculo de Mohr na forma (HIBBELER, 2010)

$$\varepsilon_a(\theta) = \frac{\varepsilon_x+\varepsilon_y}{2} + \frac{\varepsilon_x-\varepsilon_y}{2}\cos 2\theta + \frac{\gamma_{xy}}{2}\operatorname{sen} 2\theta, \tag{3.1}$$

em que θ é qualquer ângulo em relação ao eixo x do sistema de coordenadas do elemento, ε_x a deformação elementar na direção x, ε_y a deformação elementar na direção y e γ_{xy} a deformação cisalhante.

Utilizando um algoritmo implementado em linguagem TCL (*Tool Command Language* - Linguagem de Comandos de Ferramentas), em conjunto com o software HyperMesh®, possibilita a geração de um arquivo de texto contendo a posição e o ângulo de cada extensômetro candidato. Este arquivo de texto é lido diretamente pelo nCode®, o qual posiciona os extensômetros automaticamente tendo como base os seguintes campos que são gerados pelo algoritmo:

- Posição do extensômetro: o usuário escolhe entre posicionar o extensômetro no centroide de cada elemento ou nos nós da malha.
- Número de identificação: número atribuído pelo pré-processador durante a discretização do modelo para cada nó/elemento selecionado pelo usuário para compor o domínio de candidatos.
- Tipo de extensômetro: para o caso em questão considera-se a utilização apenas de extensômetros uniaxiais.
- Orientação: Este vetor orienta a direção de cada extensômetro. É determinado baseado na projeção do vetor normal à face do elemento ou do nó (calculado como a média dos vetores normais elementares adjacentes) no sistema de coordenadas globais. Neste caso, dado o vetor normal $\eta =$

$(\hat{l}, \hat{m}, \hat{n})$ do nó ou elemento no sistema de coordenadas globais, o algoritmo determina a orientação, e, do extensômetro de ângulo $0°$ baseado no menor escalar absoluto entre as componentes $(\hat{l}, \hat{m}, \hat{n})$. Considere o exemplo em que $|\hat{l}|$ é o menor escalar absoluto. Assim,

$$e = i = (1,0,0) \text{ se } |\hat{l}| < |\hat{m}|, |\hat{n}|. \tag{3.2}$$

Neste caso o vetor de orientação e é igual ao vetor unitário i na direção x do sistema de coordenadas globais, uma vez que a componente x do vetor normal à face do elemento é o menor escalar absoluto. O vetor de orientação proveniente do algoritmo gerador de domínio será sempre igual a um dos vetores unitários do sistema de coordenadas globais. Este vetor é projetado pelo software nCode® no plano cuja normal é η, e o vetor final ω de orientação para o extensômetro de ângulo $0°$ no sistema de coordenadas global será

$$\omega = \frac{\eta \times (e \times \eta)}{|\eta \times (e \times \eta)|} \tag{3.3}$$

em que $\times$ representa o produto vetorial. A Figura 8 ilustra um exemplo deste processo.

- Incremento de ângulo: O ângulo é calculado em sentido anti-horário em relação ao vetor normal η do elemento partindo do vetor de orientação ω, e o tamanho do incremento θ é definido pelo usuário.

Dependendo do componente estrutural a ser submetido ao procedimento de reconstrução de cargas, pode ser interessante excluir do domínio regiões de gradiente de tensão muito grande. Isso, porque um pequeno erro de posicionamento durante a colagem do extensômetro pode gerar uma leitura de tensão muito diferente da obtida pelo MEF. Além disso, nestas regiões, o erro do modelo de elementos finitos pode ser grande. Dessa forma, regiões próximas a soldas, raios muito pequenos e descontinuidades geométricas como cantos vivos devem ser excluídos do domínio.

No algoritmo gerador do domínio de candidatos, a identificação destas regiões de descontinuidade é feita através do ângulo entre os vetores normais de

dois elementos adjacentes, o que se convencionou chamar “arestas características” do componente.

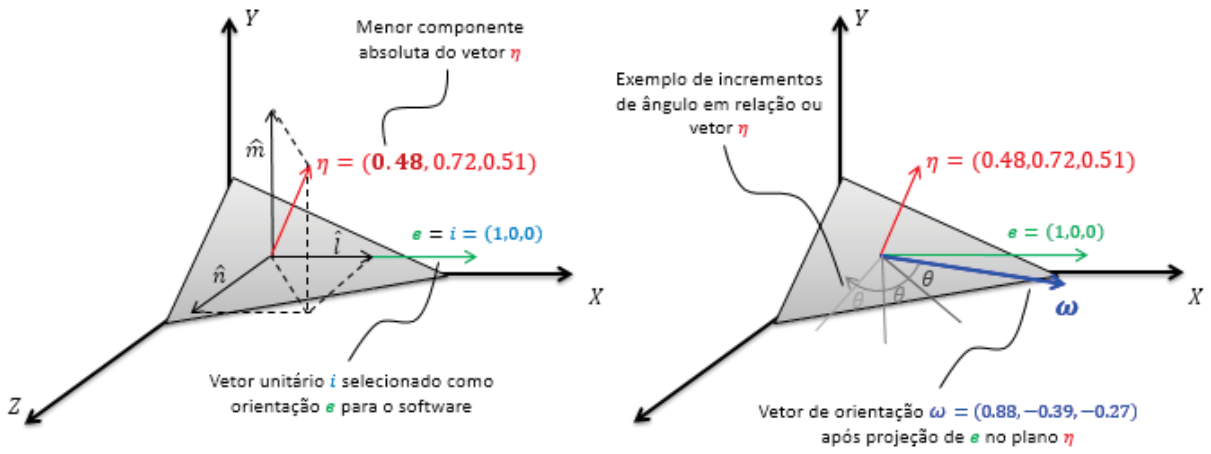


Figura 8 – Exemplo da determinação do ângulo de orientação do extensômetro de ângulo 0°.

A aresta entre dois elementos adjacentes é considerada uma aresta característica do modelo caso o ângulo entre os vetores normais for maior do que um valor especificado pelo usuário. Um exemplo da representação das arestas características em um componente pode ser visto na Figura 9, para um ângulo limite de 30º.

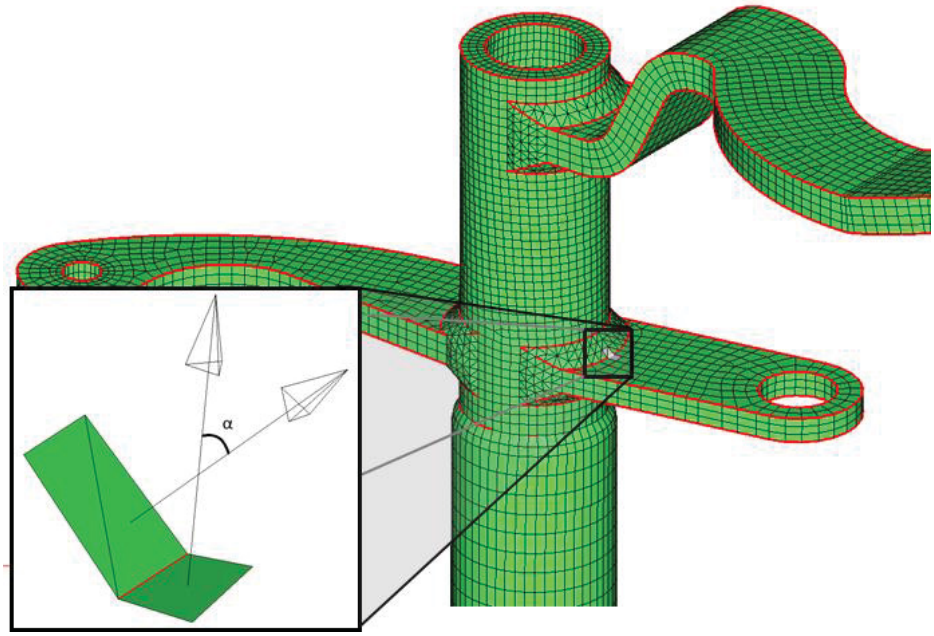


Figura 9 – Exemplo de arestas características de um componente para um ângulo limite de 30º.

3.2. DETERMINAÇÃO DA CONFIGURAÇÃO ÓTIMA DOS EXTENSÔMETROS

3.2.1. Função objetivo

O critério de projeto experimental D-ótimo foi selecionado como função objetivo para a otimização da configuração de extensômetros. Nesse caso, o problema de otimização pode ser posto como:

$$\text{Buscar } U \text{ e } \vartheta \text{ tal que, } \det(([\mathrm{A}]^{*T}[\mathrm{A}]^{*})^{-1}) = min\ (\det(([A]_N^T[A]_N)^{-1})), \qquad (3.4)$$

em que U e ϑ representam o conjunto de posições e orientações para uma dada configuração candidata N, os quais geram a *N*-ésima matriz de influência candidata $[\mathrm{A}]$, a_{ij} cada um dos seus elementos constituintes e $[\mathrm{A}]^{*}$ a matriz de influência da configuração ótima de extensômetros.

Como já exposto na seção 2.2.2, este critério é o que apresenta maior relevância para o procedimento de otimização de extensômetros para reconstrução de cargas. Wickham ***et al.*** (1994) cita como motivo para esta relevância que a raiz quadrada do determinante da matriz de dispersão $(\det([A]^T[A]))^{-1/2}$ é proporcional à variância generalizada das estimativas de força.

Segundo Aguiar ***et al***. (1995), quando um modelo de regressão é ajustado a dados experimentais, os erros experimentais são transmitidos aos coeficientes, no nosso caso, as estimativas de força. Geometricamente, os coeficientes e seus erros são representados por elipsoides cujos eixos descrevem estes erros, conforme exemplificado na Figura 10. Desta forma, quanto menores os eixos desta elipsoide, mais precisos são os coeficientes. O volume desta elipsoide é proporcional à raiz quadrada do determinante da matriz de dispersão. Desta forma, ao minimizar o determinante da matriz de dispersão, minimiza-se o volume da elipsoide.

3.2.2. Algoritmo de otimização

A metodologia adotada para a obtenção da configuração ótima dos extensômetros para a reconstrução de cargas utiliza um algoritmo genético. Esta escolha se deve principalmente à sua capacidade de obter boas aproximações para

projetos ótimos globais, permitindo sua utilização para geometrias complexas que possam conter mínimos locais e grande número de elementos no modelo de elementos finitos.

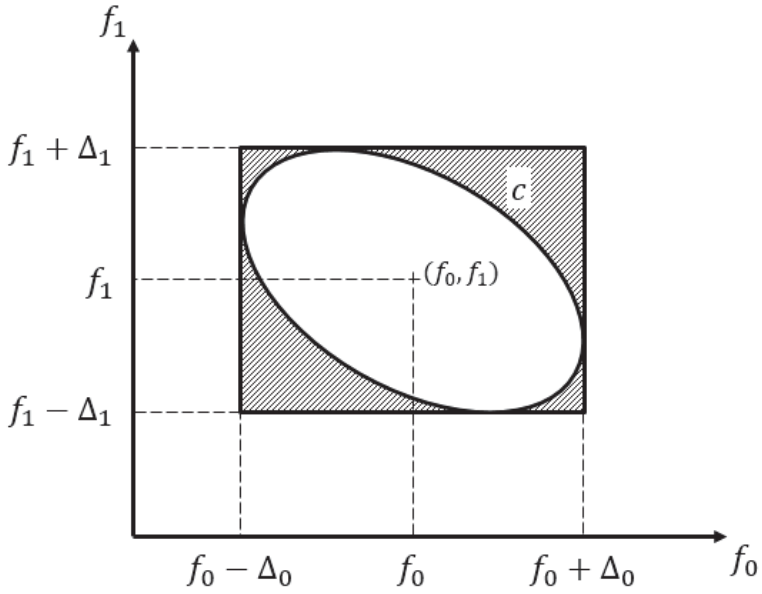


Figura 10 – Elipsoide de confiança para as estimativas de força.
Fonte: modificado de Aguiar *et al*. (1995).

O algoritmo foi desenvolvido na linguagem Python, aproveitando sua integração com o software nCode® e permitindo a simplificação da leitura do resultado de deformações do modelo de elementos finitos.

Como cada gene do domínio é representado por um número inteiro, optou-se por não codificá-los, uma vez que, conforme discutido por Xu (1999), a codificação de alguns números inteiros muito próximos pode ter uma representação binária muito diferente, comprometendo a convergência do algoritmo.

Para fazer a seleção dos pais que irão gerar a nova população através do cruzamento, utilizou-se o método de classificação linear de Baker (XU, 1999) ao invés da aptidão de cada cromossomo. Esta escolha se deu devido ao fato de que a aptidão calculada baseada no determinante da matriz de dispersão gera números muito grandes, e entre dois cromossomos de uma mesma população pode haver uma diferença de algumas ordens decimais, o que causaria uma pressão muito grande do cromossomo mais apto sobre os demais. A pressão do mais apto pode ser ajustada para obter uma boa taxa de convergência sem comprometer a diversidade das gerações.

Neste método, cada cromossomo é ordenado em ordem crescente de aptidão de 1 a N, e a expectativa de seleção para cada cromossomo é dada pela equação (XU, 1999)

$$R(j) = 2 - Max + (2Max - 2) \cdot \frac{j-1}{N-1}, \quad (3.5)$$

em que Max é o número de Baker e representa a expectativa de seleção para o cromossomo de ordem N, e j é a posição do cromossomo dentro da população.

Segundo Xu (1999), Baker recomenda que o valor de 1,1 seja utilizado para o número de Baker. Dentro do intervalo [1;2], quanto maior o valor, maior a pressão do mais apto sobre o menos apto na seleção, permitindo que o usuário ajuste de acordo com cada problema. Valores no intervalo [0;1] para o número de Baker produzirão as mesmas expectativas, entretanto, neste caso deve-se ordenar os cromossomos em ordem decrescente de aptidão. Esta última foi a forma adotada neste estudo. Na Tabela 3 a expectativa de seleção $R(j)$ é calculada para alguns valores de número de Baker em uma população de 6 cromossomos.

Tabela 3 – Expectativa de seleção do mètodo de classificação linear de baker.

		Expectativa $R(j)$					
Número de Baker		**0,7**	**0,8**	**0,9**	**1,1**	**1,2**	**1,3**
Ordem (j)	**1**	1,3	1,2	1,1	0,90	0,80	0,70
	2	1,18	1,12	1,06	0,94	1,12	0,82
	3	1,06	1,04	1,02	0,98	1,04	0,94
	4	0,94	0,96	0,98	1,02	0,96	1,06
	5	0,82	0,88	0,94	1,06	0,88	1,18
	6	0,7	0,8	0,9	1,10	0,80	1,30

Após a classificação dos cromossomos através de suas aptidões, o número de Baker é atribuído a cada cromossomo de acordo com a sua ordem na população. O método da Amostragem Universal Estocástica é utilizado para selecionar randomicamente os cromossomos que farão parte do cruzamento.

Este método de amostragem foi proposto por James Baker (XU, 1999) e é baseado no método clássico da roleta, sendo que cada fatia desta roleta terá tamanho proporcional à expectativa R(j) calculada para cada cromossomo. Diferentemente do método da roleta clássica, neste caso ela possui N marcadores

igualmente espaçados. Desta forma, a roleta é girada apenas uma vez e os cromossomos apontados pelos N marcadores são selecionados. No exemplo da Figura 11, seis marcadores são utilizados, uma vez que a população possui 6 cromossomos. Os valores de P de 1 a 6 representam a expectativa R(j) de seleção de cada cromossomo. Quando a roleta é girada, os marcadores assumem a posição mostrada em vermelho, e o cromossomo 1 é selecionado 2 vezes, os cromossomos 2, 3, 4 e 6 são selecionados 1 vez e o cromossomo 5 não é selecionado.

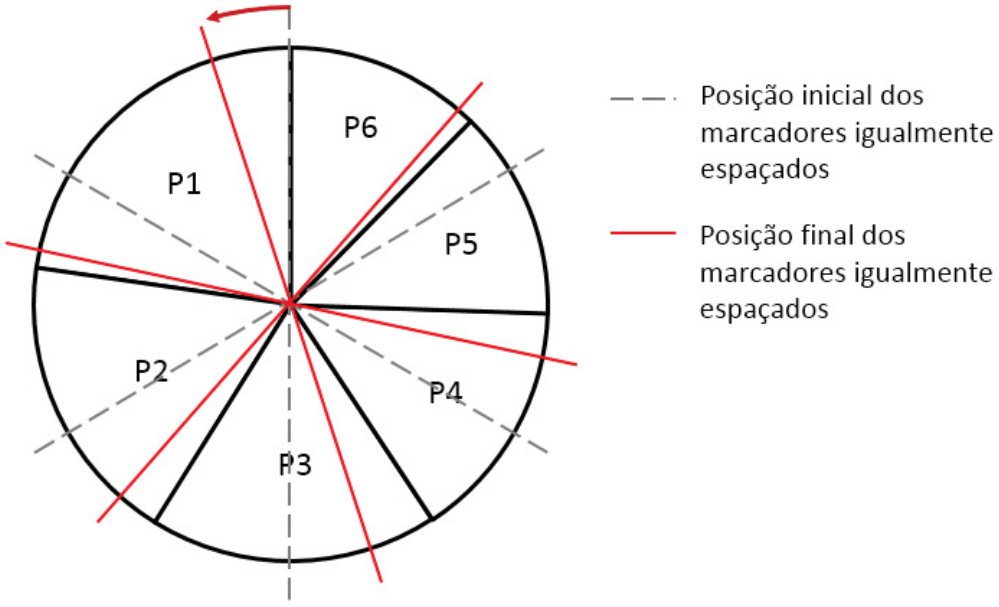


Figura 11 – Método da amostragem universal estocástica.

Com o objetivo de que o melhor cromossomo de cada geração não seja perdido durante os processos de cruzamento e mutação, o modelo de substituição da geração com elitismo foi utilizado. Desta forma, alguns dos melhores cromossomos são eleitos e serão levados à próxima geração inalterados.

Os cromossomos selecionados são pareados aleatoriamente e o cruzamento por dois pontos é realizado. Este tipo de cruzamento foi selecionado por permitir um maior número de combinações entre os pais que o cruzamento simples por um ponto. No caso específico deste estudo, trocam-se os números inteiros que representam diretamente cada extensômetro entre dois cromossomos. Um exemplo deste processo é mostrado na Tabela 4.

Uma taxa de mutação é atribuída pelo usuário e representa a probabilidade que cada gene tem de sofrer mutação. Cada gene da população é testado

individualmente e um número aleatório no intervalo [0,1] é atribuído a ele. Caso o número seja menor do que a taxa, o gene é escolhido para sofrer mutação.

Uma vez que os genes são números inteiros e não binários, os genes selecionados de cada cromossomo são substituídos por um número inteiro que representa outra posição e orientação é escolhido randomicamente do domínio.

Tabela 4 – Exemplo de cruzamento por dois pontos.

Pais	Filhos
[x1,x2, x3, x4, **x5, x6]**	[x1,x2, **y3, y4,** x5, x6]
[y1,y2, y3, y4, **y5, y6]**	[y1,y2, **x3, x4,** y5, y6]

Fonte: modificado de Xu (1999).

Os processos evolucionários aplicados à população podem gerar genes idênticos em um mesmo cromossomo, o que não é desejável no caso específico de seleção de extensômetros. Isto pode significar na prática não poder reconstruir uma certa quantidade de carregamentos, caso o número de extensômetros passe a ser menor que ela.

Para solucionar este problema, uma etapa de reparação é realizada após a mutação. Os extensômetros são ordenados para cada tipo de força aplicada no modelo, do maior para o menor valor absoluto de deformação, o que garante que estes candidatos têm uma boa resposta para o carregamento que eles devem descrever. Uma porcentagem dos extensômetros, determinada pelo usuário, com maior valor absoluto de deformação em cada carregamento é guardada na memória, e para cada gene repetido encontrado no cromossomo, este é reparado substituindo-o por outro randomicamente escolhido da lista previamente armazenada. Este procedimento tende a aumentar a aptidão média da geração, além de garantir a diversidade, impedindo a convergência para mínimos locais.

O critério de parada do algoritmo é baseado na estabilização da média das aptidões da população, e é calculado como a diferença da média dos logaritmos das aptidões do modelo entre duas gerações consecutivas de acordo com a equação:

$$C = \left| \frac{\sum_{j=1}^{N} \log(D_j)}{N} - \frac{\sum_{j=1}^{N} \log(\bar{D}_j)}{N} \right| \tag{3.5}$$

em que D_j representa o determinante de cada um dos cromossomos da atual geração, $\overline{D_j}$ o determinante dos cromossomos da geração anterior e N o número total de cromossomos na população.

O usuário define um erro admissível para a análise e quando C for menor do que o erro a análise é encerrada. Além disto, também é possível determinar um número máximo de iterações como critério de convergência secundário.

Para analisar a eficiência do algoritmo genético, o algoritmo de troca de Fedorov foi também implementado, utilizando o procedimento mostrado na seção 2.3.1.1. A implementação computacional segue a forma apresentada por Triefenbach (2008).

3.3. TESTES FÍSICOS DE VALIDAÇÃO DA METODOLOGIA

Para a validação da metodologia desenvolvida, dois componentes estruturais foram utilizados. Um componente proveniente de um trator produzido pela empresa CNH Industrial Latin America Ltda. e um suporte construído de forma similar à estrutura testada por Hunter (2012). Estes são mostrados na Figura 12.a e Figura 12.b.

Para estes componentes, foram gerados os modelos em elementos finitos utilizando os softwares HyperMesh® e RADIOSS®, aplicando os carregamentos unitários que melhor representam a combinação possível de carregamentos reais sofridos por eles durante os testes.

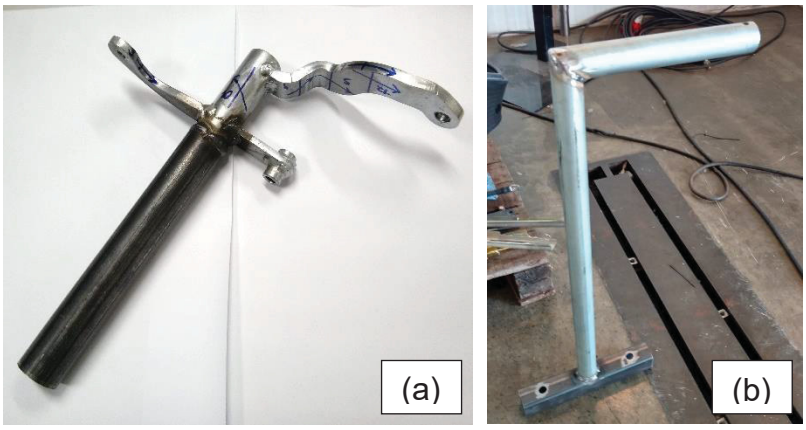


Figura 12 – Estruturas testadas: (a)-alavanca de um trator, (b)-suporte construído.

O modelo de elementos finitos para a alavanca foi gerado com elementos tridimensionais lineares e uma fina camada externa de 0,01m de espessura de elementos de casca lineares. Considerou-se que apenas os pontos A e B indicados na Figura 13 irão receber carga.

Para cada um destes pontos, é aplicado um carregamento unitário em cada um dos eixos coordenados, perfazendo um total de 6 casos de carga individuais. Estes carregamentos permitem reconstruir qualquer combinação de forças que possam ocorrer nos dois pontos. Uma porção do tubo central, que foi utilizada para fixação do componente durante os testes, teve todos os seus graus de liberdade restritos na análise.

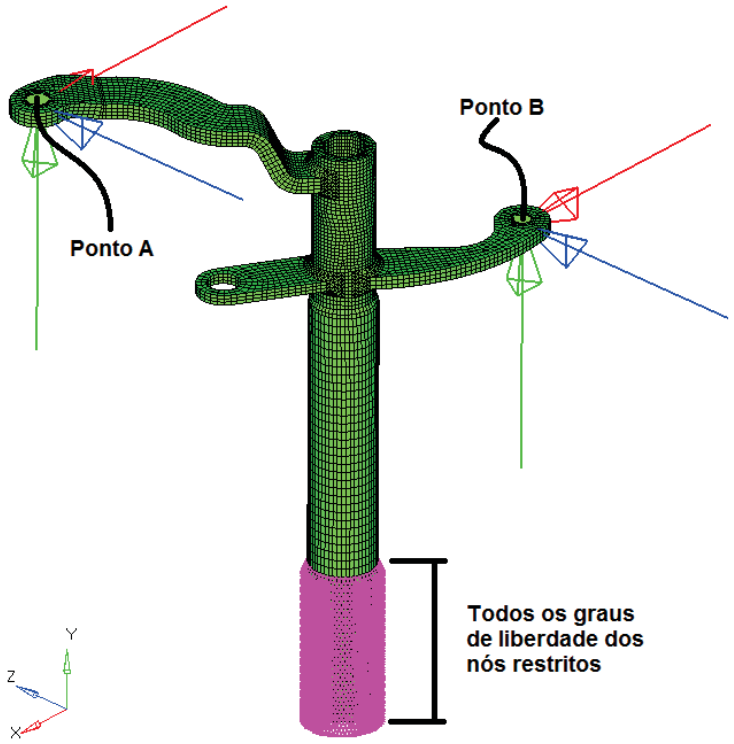


Figura 13 – Modelo em elementos finitos da alavanca.

O componente foi fixado em uma bancada de testes na mesma porção restrita no modelo de elementos finitos. Dois dispositivos que permitem a adição de massas foram pendurados nos pontos A e B da estrutura. A estrutura foi também posicionada em diferentes ângulos de inclinação e rotação de forma que a direção da força resultante pudesse ser comparada com a direção da força reconstruída. A Figura 14 mostra o componente já posicionado no dispositivo de teste e os ângulos de inclinação possíveis. A combinação de carregamentos testados pode ser vista na Tabela 5.

Suportes e pesos utilizados

Figura 14 – Dispositivo de teste para a alavanca.

Tabela 5 – Combinaçâo de carregamentos aplicados durante os testes da alavanca.

Caso de carga	Massa em A [kg]	Massa em B [kg]	Rotação ∅ [graus]	Inclinação ψ [graus]
1	11,56	0	0	0
2	11,56	0	45	0
3	11,56	0	0	-45
4	11,56	11,60	0	0
5	11,56	11,60	-37	0
6	11,56	11,60	0	-30
7	0	11,60	0	0
8	0	11,60	-45	0
9	0	11,60	0	-38

Foram aplicados 12 extensômetros uniaxiais da marca Excel, modelo PA-06-125BA-120L de 6 mm de comprimento e 120 Ω (Ohms) de resistência no componente, dentre os quais 6 posições foram determinados pelo algoritmo genético e os demais foram selecionados aleatoriamente. Desta forma, o erro da reconstrução em relação aos carregamentos reais aplicados pode ser calculado com combinações aleatórias de 6 extensômetros entre os 12, e comparado com a localização ótima.

O modelo de elementos finitos para o suporte foi gerado com elementos lineares do tipo casca considerando que somente a extremidade superior irá receber esforço. Um carregamento unitário em cada um dos eixos coordenados foi aplicado a este ponto, perfazendo 3 casos de carga individuais. Estes carregamentos permitem reconstruir qualquer direção e intensidade de força que possa ocorrer. Os dois furos do tubo inferior foram utilizados para fixar a estrutura ao chão do laboratório, e os mesmos foram restritos em todos os seus graus de liberdade na análise. O modelo gerado pode ser visto na Figura 15.

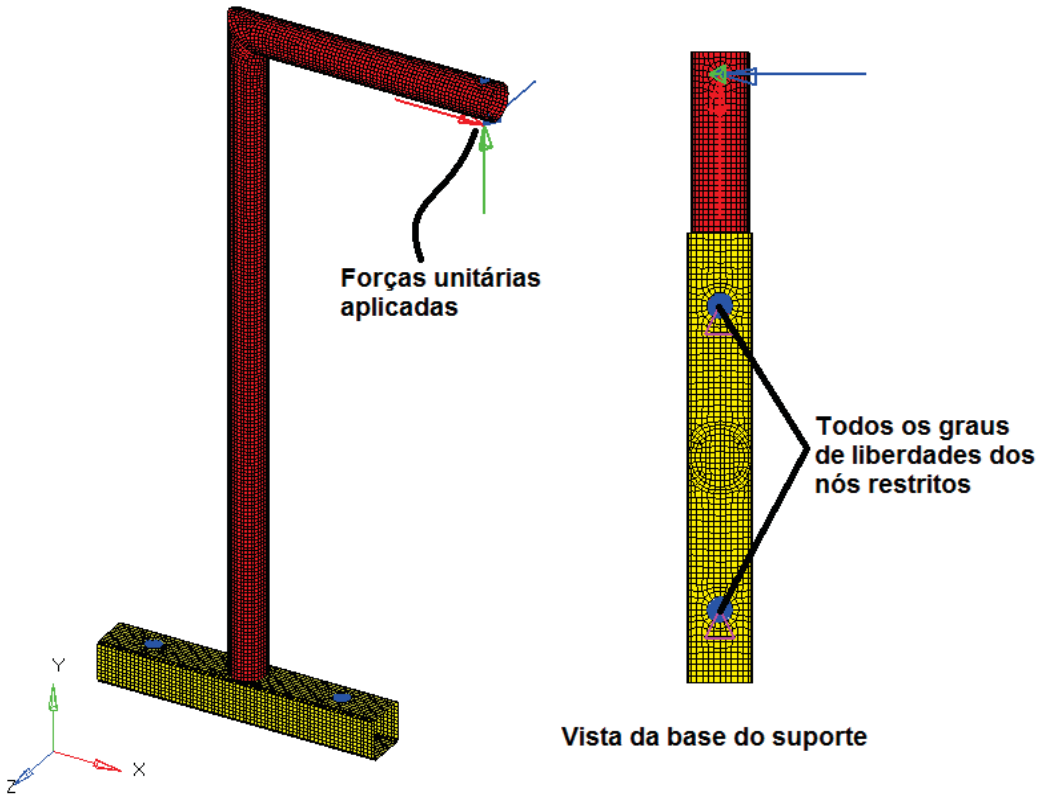


Figura 15 – Modelo em elementos finitos para o suporte.

O mesmo suporte e pesos utilizados no teste da alavanca foram utilizados para testar o suporte. O dispositivo de teste pode ser visto na Figura 16. A combinação de carregamentos testados pode ser vista na Tabela 6.

Figura 16 – Dispositivo de teste para o suporte.

Tabela 6 – Combinação de carregamentos aplicados durante os testes do suporte.

Caso de carga	Massa em A [kg]	Ângulo em relação à vertical [graus]
1	11,55	2,5
2	21,60	2,5
3	31,60	2,5
4	41,70	2,5

Foram aplicados 14 extensômetros uniaxiais da marca Excel, modelo PA-06-125BA-120L de 6 mm de comprimento e 120 Ω (Ohms) de resistência no componente, sendo 10 destes determinados pelo algoritmo de otimização e os demais posicionados aleatoriamente.

Os dois componentes foram fabricados utilizando aços normatizados pela CNH Industrial Latin America Ltda.. Desta forma, as propriedades lineares dos materiais, utilizadas nas simulações com carregamentos unitários, seguem os valores estabelecidos em norma. São eles:

- Módulo de Young: 207 GPa

- Coeficiente de Poisson: 0,29
- Densidade: 7820 kg/m^3

O sistema de aquisição de dados da marca Lynx modelo DLG4000 e seu software de aquisição AqDAnalysis® foram utilizados em todas as medições de deformação dos extensômetros. Uma imagem do equipamento pode ser vista na Figura 17.

Figura 17 – Sistema de aquisição Lynx DLG4000.

Todos os testes foram realizados nos laboratórios da CNH Industrial Latin America Ltda. e todos os itens utilizados nos testes, conforme descrito nesta seção, pertencem a ela.

3.4. RECONSTRUÇÃO DOS CARREGAMENTOS MEDIDOS

A reconstrução dos carregamentos a partir dos dados de deformação medidos fisicamente nos testes foi feita com o processador específico do software nCode®. O mesmo modelo de elementos finitos utilizado para a obtenção da configuração ótima de extensômetros nos componentes testados foi também utilizado na reconstrução dos carregamentos.

O procedimento baseia-se na inversão da matriz de influência conforme demonstrado no capítulo 2, utilizando para tal a equação

$$\{f\} = ([A]^T[A])^{-1}[A]^T\{\varepsilon\}\,. \tag{3.5}$$

O software nCode® possui uma estrutura de blocos que podem ser conectados para construir uma sequência lógica de processamento. A Figura 18 mostra como a reconstrução ocorre. O modelo de elementos finitos é introduzido no bloco 1. O bloco 2 recebe a configuração de extensômetros que foi obtida através do algoritmo de otimização, e calcula a matriz de influência para ela. No bloco 3 são inseridas as deformações medidas fisicamente em cada um dos extensômetros. O bloco 4 faz a inversão da matriz de influência proveniente do bloco 2 e multiplica pelo vetor de deformações para cada instante de tempo provenientes do bloco 3. Nos blocos 5 e 6 os vetores de força resultante podem ser visualizados e exportados.

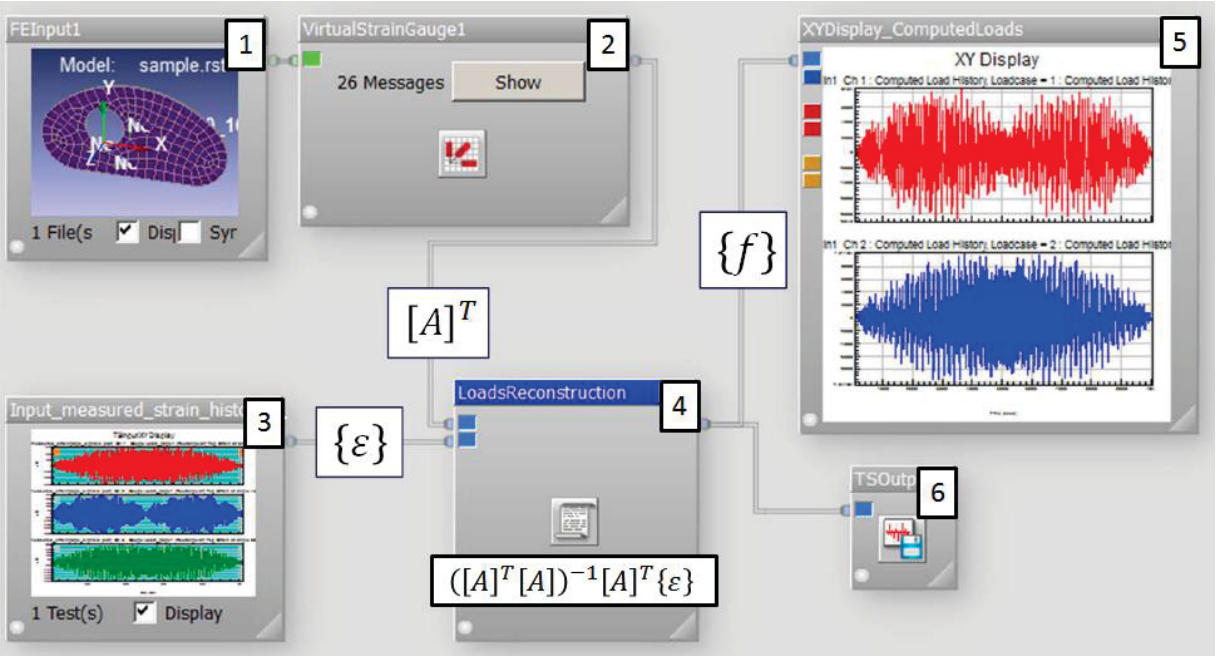


Figura 18 – Esquema da reconstrução dos carregamentos no nCode®.

4 RESULTADOS E DISCUSSÃO

4.1. ESTRUTURA COMPUTACIONAL

A estrutura computacional foi desenvolvida em duas partes, a geração do domínio de candidatos e a otimização do melhor conjunto de extensômetros para reconstrução de cargas, integradas através de um arquivo de texto. Um fluxograma exemplificando como todo o processo funciona é mostrado na Figura 19.

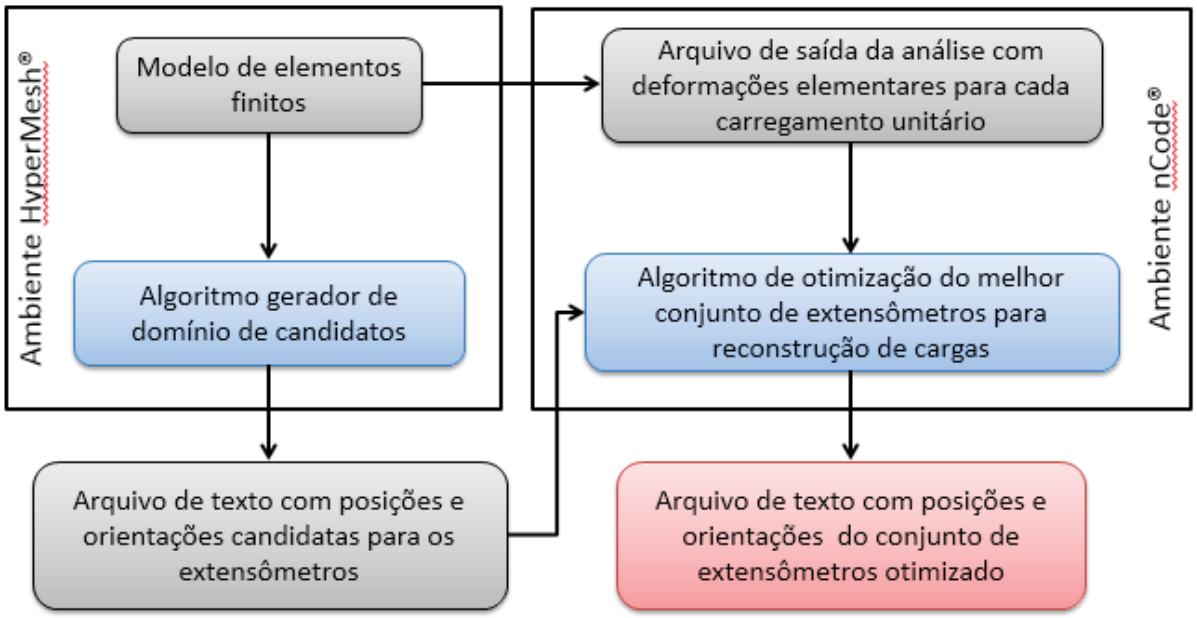


Figura 19 – Fluxograma da estrutura computacional desenvolvida.

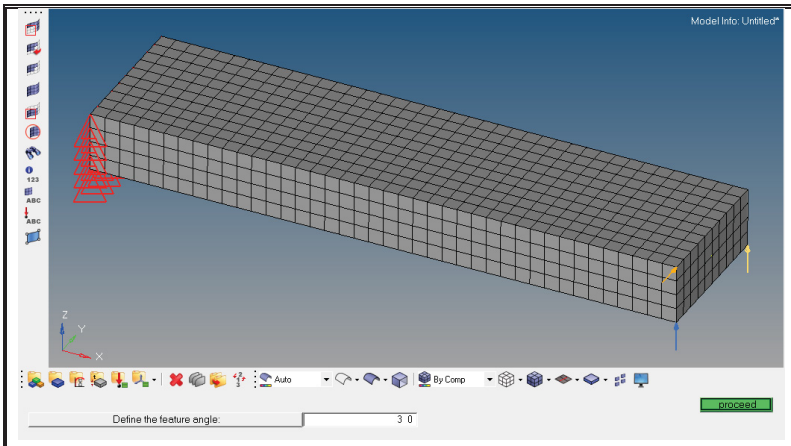
4.1.1. Gerador de Domínio de Candidatos

O algoritmo implementado na linguagem TCL aproveita comandos macro implementados no pré-processador HyperMesh®. Inicialmente, o usuário seleciona o ângulo limite para as arestas características do componente. Se o ângulo entre as normais de dois elementos adjacentes for maior que o valor preestabelecido, os dois nós que compõe esta aresta são armazenados. Na sequência o número de carreiras adjacentes à aresta que deverão ser excluídos do domínio deve ser informado. Caso os resultados de tensão em uma análise prévia com os carregamentos unitários

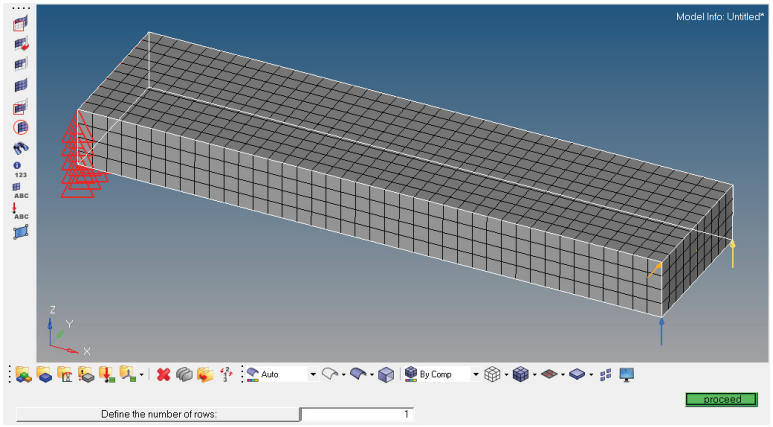
mostre um gradiente de tensão muito grande e próximo a uma aresta, mais de uma carreira de elementos adjacente a ela pode ser removido. Na Figura 20 pode-se observar um exemplo da interface com o usuário para a seleção do ângulo limite, as arestas selecionadas, o número de linhas adjacentes e os elementos selecionados para exclusão.

O próximo passo após a remoção dos elementos indesejáveis é a seleção dos elementos/nós que farão parte do domínio. Nesta etapa, o usuário pode selecionar todos os elementos/nós restantes ou apenas uma parcela que julgue de maior interesse para a reconstrução de cargas. Regiões inacessíveis fisicamente ou que possam estar expostas a contato com outros componentes podem ser excluídas neste passo. Um exemplo de seleção parcial de nós para a geração do domínio de candidatos pode ser visto na Figura 21.

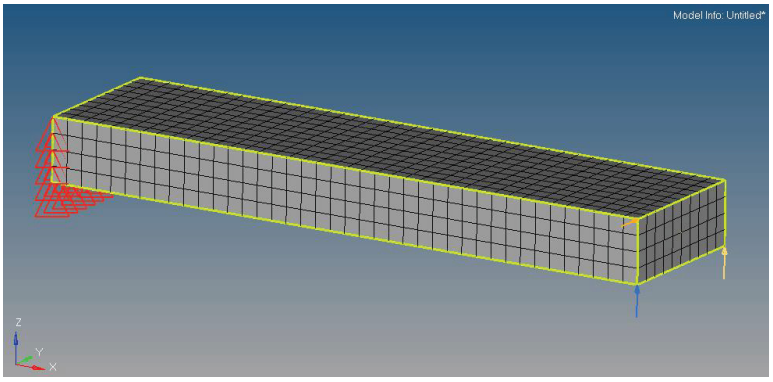
Após a seleção dos nós, um painel de seleção permite determinar o incremento de ângulo para o qual serão gerados os extensômetros virtuais candidatos. Este ângulo pode ser qualquer valor no intervalo [0º;180º[. Um arquivo de texto que será interpretado pelo software nCode® é gerado contendo para cada extensômetro virtual candidato as informações da posição (número do nó ou elemento), orientação global inicial do elemento de ângulo 0º e incremento de ângulo, conforme discutido no capítulo anterior. Um fluxograma do processo pode ser visto na Figura 22.



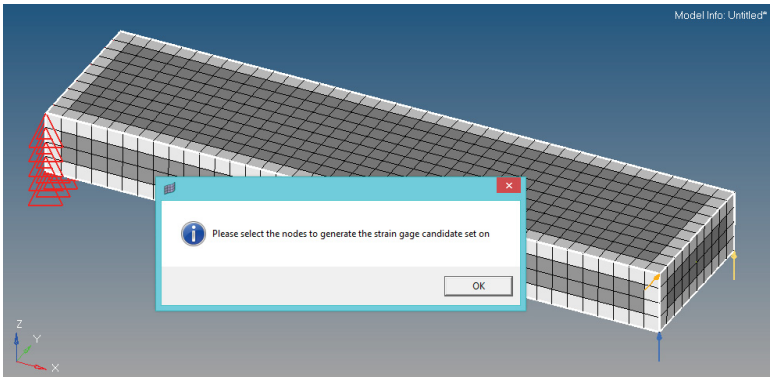
a) Painel de seleção do ângulo limite para a determinação das arestas características.



b) Painel de seleção do número de carreiras de elementos adjacentes às arestas características a serem excluídas do domínio.



c) Exemplo de arestas características em amarelo.



d) Exemplo de elementos selecionados para remoção do domínio utilizando-se apenas uma carreira adjacente às arestas características.

Figura 20 – Exemplo de remoção de elementos adjacentes a arestas características.

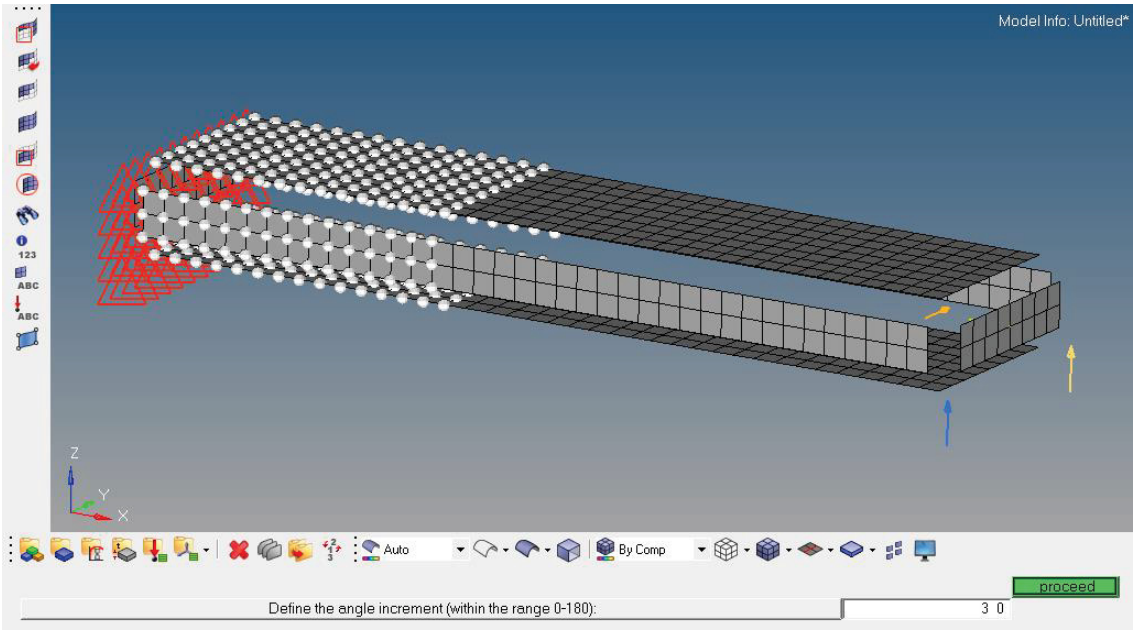


Figura 21 – Exemplo de nós selecionados para a construção do domínio de candidatos.

4.1.2. Processo de Otimização

Neste processo quatro blocos principais do nCode® foram utilizados:

- 1 – Bloco de importação do arquivo de saída da análise de elementos finitos e do arquivo de texto com os extensômetros do domínio;
- 2 – Bloco de cálculo de deformações em cada extensômetro a partir do tensor de deformações de cada elemento;
- 3 – Bloco com o algoritmo de otimização implementado em Python;
- 4 – Bloco gráfico que permite visualizar a evolução da função objetivo ao término do processo.

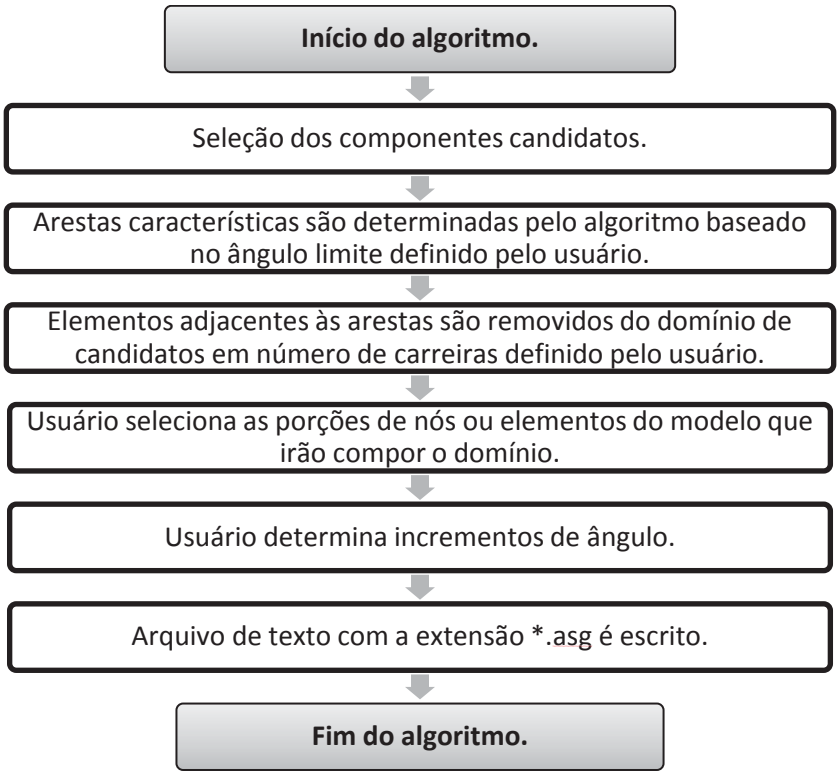


Figura 22 – Fluxograma do algorítmo gerador de domínio.

Uma imagem do fluxo de processamento do software nCode® pode ser visualizado na Figura 23. Os blocos de 1 a 4 estão dispostos na sequência da direita para a esquerda. No bloco 1, o modelo de elementos finitos é interpretado e convertido para um formato que o processador do nCode® é capaz de compreender. Além disso, o arquivo de texto, o qual foi gerado na etapa de determinação do domínio de configurações candidatas, é interpretado e cada extensômetro virtual é posicionado como pode ser visualizado na Figura 24.

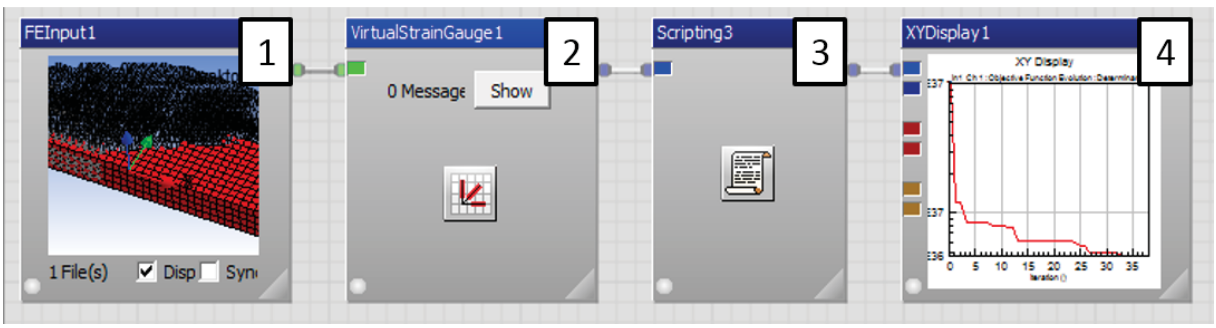


Figura 23 – Estrutura de blocos do software ncode® utilizada no processo de otimização.

No bloco 2 a deformação em cada extensômetro virtual é calculada para cada carregamento unitário disponível no arquivo de saída do modelo de elementos finitos.

No bloco 3 o algoritmo de otimização é interpretado. Uma interface que permite selecionar o método (Fedorov ou AG) e os parâmetros do algoritmo de otimização foi implementada. Um panorama dos parâmetros disponibilizados para o usuário é mostrado na Figura 25.

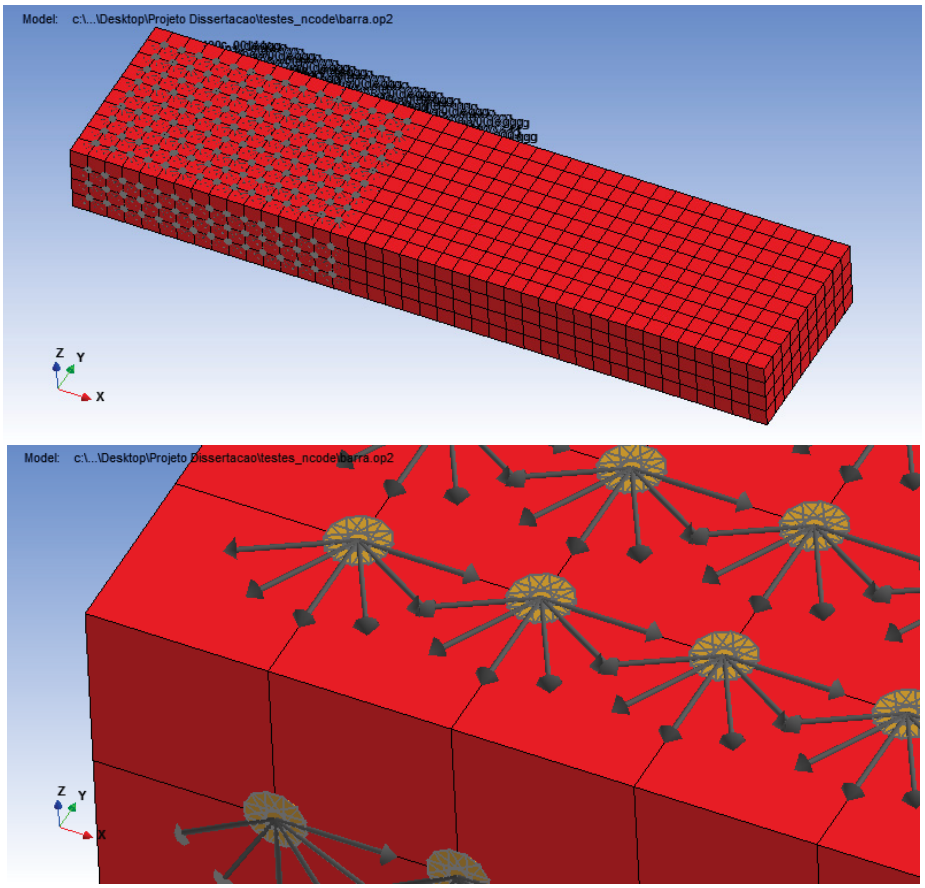


Figura 24 – Detalhe da visualizaçâo do domìnio de extensômetros conforme interpretada pelo software nCode®.

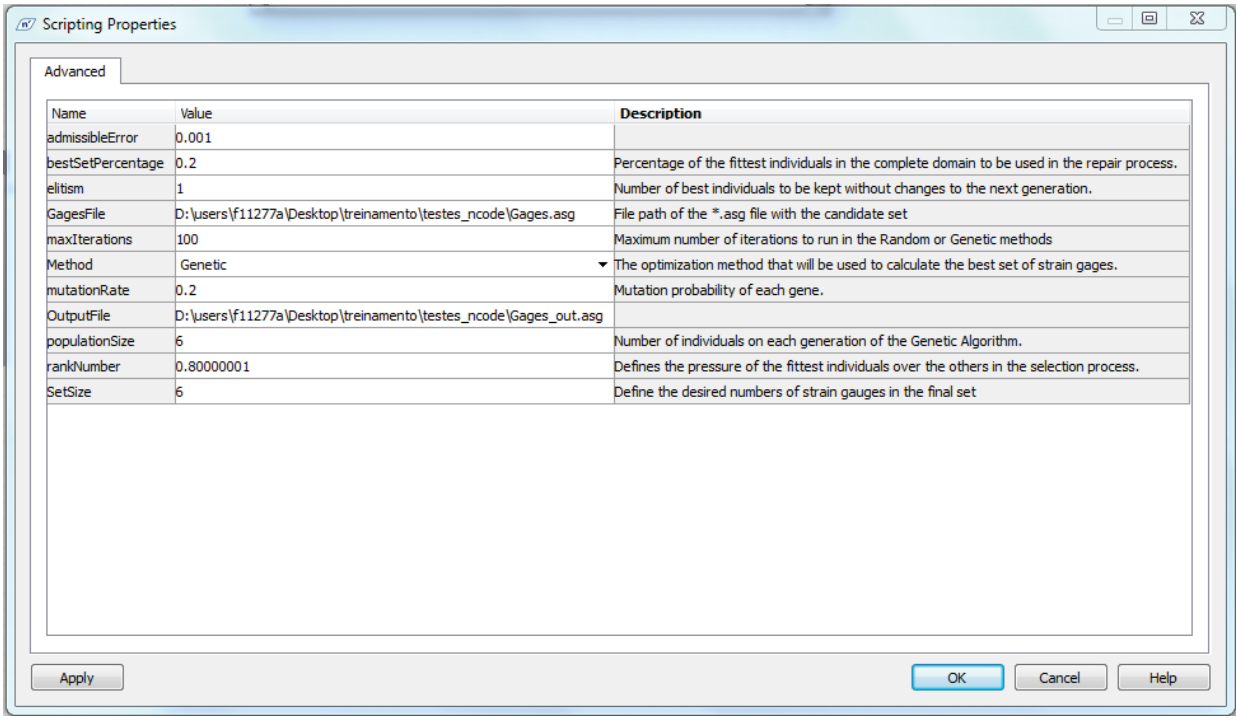


Figura 25 – Parâmetros do algorítmo de otimizaçâo disponibilizados ao usuário.

Nesta etapa do algoritmo de otimização, todos os processos descritos na seção 3.2.2 são executados. Um pseudocódigo do algoritmo implementado é mostrado na Figura 26.

Por fim, o bloco 4 permite exportar e visualizar a evolução da função objetivo do cromossomo mais apto e da média dos logaritmos das aptidões dos cromossomos da população ao longo das gerações. Um exemplo dos gráficos gerados é mostrado na Figura 27.

Início

Para cada força (f_i) do modelo:
Encontrar "bestSetPercentage"

Gerar população inicial
Para i = 1 até “populationSize”:
Para j = 1 até “SetSize” :
gene[i,j] = extensômetro aleatório $\epsilon\ \beta$

Enquanto iteração < “maxIterations” e C > “admissibleError”:

Aptidão
Para cada cromossomo[i] em “populationSize”:
Calcular aptidão $\varphi[i] = det\ ([A]^T[A])^{-1}$

Elitismo
Ordenar cromossomos em ordem decrescente de aptidão

Para i = 1 até “elitism”:
cromossomo[i] = eleitos[i]

Seleção
Para cada cromossomo[i] em “populationSize”:
Calcula expectativa R[i] de seleção para “rankNumber”

marcador = aleatório [0-1]
Para i = 1 até “populationSize”:
S[i] = S[i-1] + R[i]
Enquanto marcador <= S[i]:
selecionados = cromossomo[i]
marcador = marcador + i

Cruzamento
Parear cromossomos aleatoriamente

pos1 = posição aleatória [0 até “SetSize”-1]
pos2 = posição aleatória [Pos1 até “SetSize”]
Para i = 1 até “populationSize”:
Cromossomo1(genes pos1-pos2) = Cromossomo2(genes pos1-pos2)
Cromossomo2(genes pos1-pos2) = Cromossomo1(genes pos1-pos2)

Mutação
Para i = 1 até “populationSize”:
Para j = 1 até “SetSize” :
marcador = aleatório [0-1]
Se marcador < “mutationRate”:
gene[i,j] = extensômetro aleatório $\epsilon\ \beta$

Reparação
Para cada cromossomo[i] em “populationSize”:
Para j = 1 até “SetSize”:
Para k = 1 até “SetSize”:
Se gene[j] = gene[k]:
gene[i,j] = extensômetro ϵ "bestSetPercentage"

Escrever “OutputFile”

Fim

Figura 26 – Pseudocódigo do algorítmo de otimizaçâo.

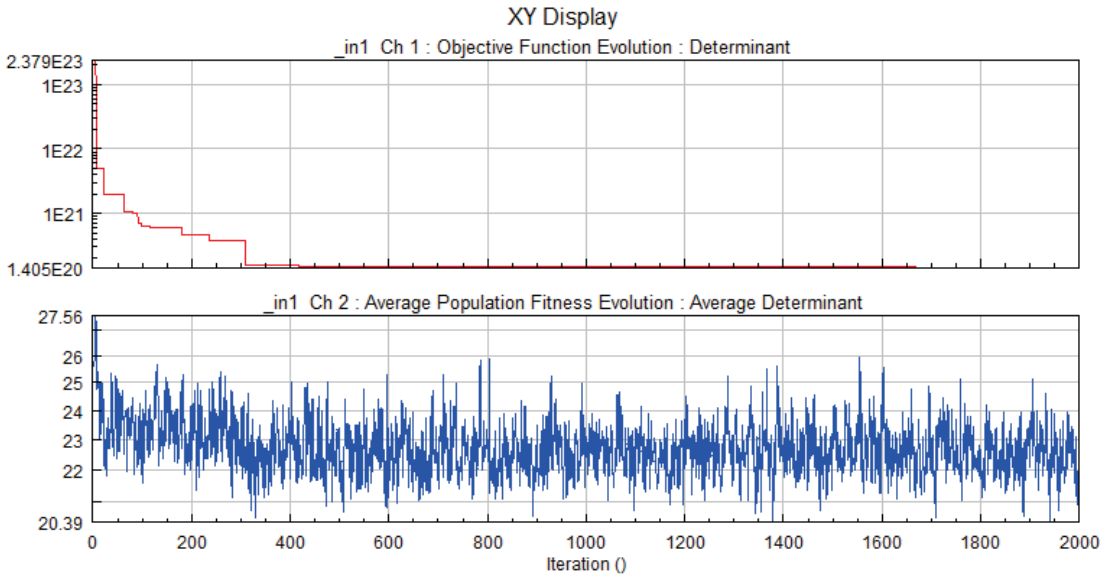


Figura 27 – Gráficos gerados ao final do processo de otimização.

4.2. ANÁLISE DA EFICIÊNCIA DO ALGORÍTMO IMPLEMENTADO

Para que fosse possível realizar uma avaliação da eficiência do algoritmo e da influência de cada uma de suas variáveis na aptidão final obtida, o exemplo da barra mostrado na Figura 6 e na Figura 7 foi utilizado. Este modelo possui 5 carregamentos aplicados no centro do furo, representando a conexão por pino na extremidade. Desta forma, são necessários pelo menos 5 extensômetros para realizar a reconstrução de forças.

Para este modelo, o domínio de configurações candidatas foi gerado tendo como referência o centroide dos elementos da membrana exterior com um incremento de ângulo de 15º. Uma imagem do domínio de candidatos gerados pode ser vista na Figura 28. Ao todo, 7128 extensômetros candidatos foram gerados.

Inicialmente, um estudo dos parâmetros do algoritmo genético foi realizado. Para este estudo, considerou-se como objetivo obter uma configuração com 5 extensômetros, o mínimo necessário para reconstruir-se as 5 entradas de força possíveis.

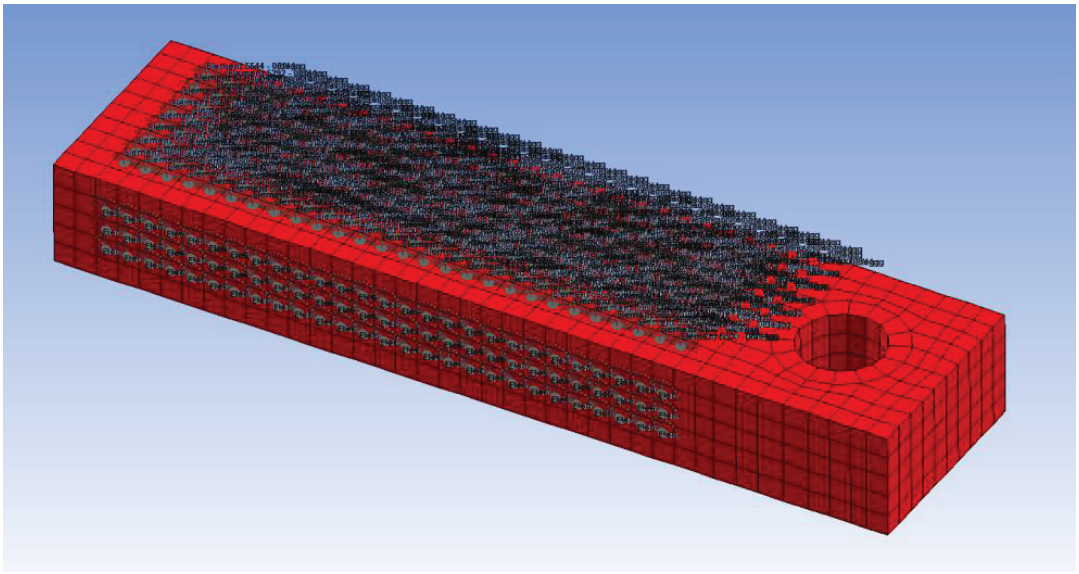


Figura 28 – Domínio de candidatos para modelo de teste.

4.2.1. Influência do tamanho da população na eficiência do algoritmo

A influência do tamanho da população foi avaliada considerando os demais parâmetros do algoritmo constantes e com valores recomendados encontrados na bibliografia (XU, 1999). Estes parâmetros são mostrados na Tabela 7.

Tabela 7 – Parâmetros do algoritmo genético para análise do tamanho da população.

Erro admissível	0,001
Porcentagem dos melhores extensômetros	0,2
Elitismo	1
Taxa de mutação	0,2
Número de Baker	0,9
Número máximo de iterações	2000

O algoritmo foi executado para tamanho de populações de 5, 10, 20, 50, 100, 500 e 1000 cromossomos (indivíduos), sendo repetido por 3 vezes para cada uma. Foram avaliados o tempo de processamento e a aptidão do cromossomo mais apto da população ao final do processamento. A aptidão foi normalizada dividindo-se o menor valor encontrado entre todas as análises pela aptidão de cada análise. Desta forma, quanto mais próximo de 1 o valor da aptidão normalizada, mais próximo do ótimo global está o cromossomo mais apto. No gráfico da Figura 29 são

apresentados os resultados para cada processamento e as curvas médias de valores de tempo e de aptidão normalizada para cada tamanho de população.

Observa-se que, apesar de a melhor aptidão ser encontrada para uma população de 20 cromossomos, os menores tempos de processamento são encontrados para populações com 50 cromossomos.

Observa-se ainda uma redução da dispersão dos resultados de aptidão para populações maiores. Para populações de 500 cromossomos encontramos uma boa relação entre a aptidão média, o tempo de processamento médio e a dispersão dos resultados.

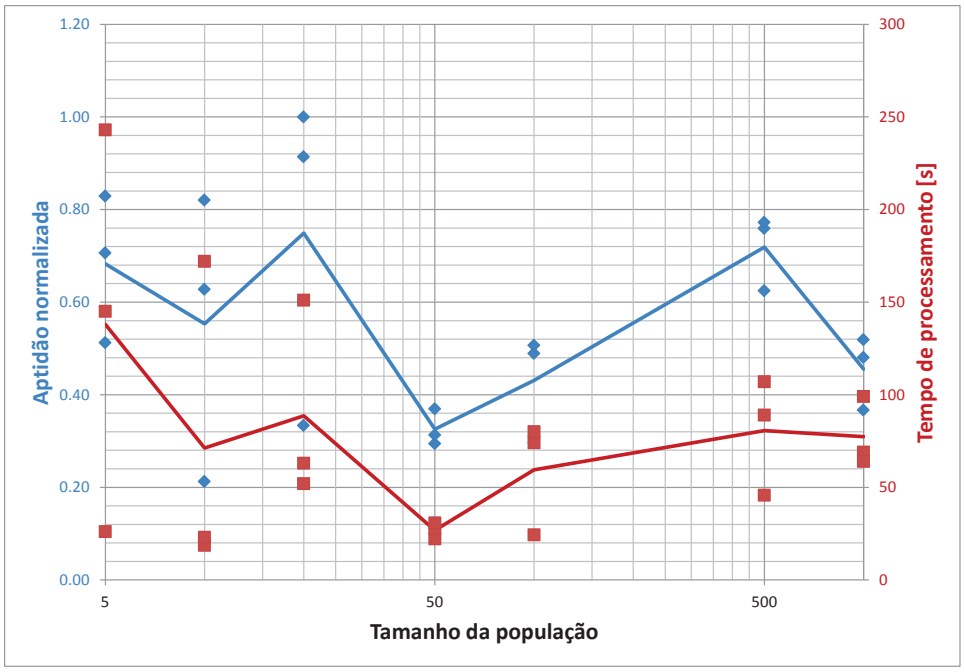


Figura 29 – Influência do tamanho da população no tempo de processamento e aptidão final.

4.2.2. Influência da porcentagem dos melhores extensômetros no processo de reparação do cruzamento

A influência da porcentagem dos extensômetros com maior valor absoluto de deformação, que são usados na etapa de reparação do algoritmo, foi avaliada considerando os demais parâmetros do algoritmo constantes e com os valores mostrados na Tabela 8.

O algoritmo foi executado para as porcentagens de 10%, 20%, 50% e 100% (o valor de 100% significa que qualquer extensômetro do domínio pode ser selecionado), sendo repetido por 4 vezes para cada uma. Optou-se por uma população de 5 cromossomos apenas, por ser essa a que apresenta maiores tempos de processamento e baixa eficiência. Desta forma, a influência da porcentagem dos melhores extensômetros na melhoria da aptidão média e na redução do tempo de convergência do algoritmo ficaria mais evidente. Foram avaliados o tempo de processamento e a aptidão do cromossomo mais apto da população ao final do processamento. No gráfico da Figura 30 são apresentados os resultados para cada processamento e as curvas médias de valores de tempo e de aptidão normalizada para cada porcentagem.

Tabela 8 – Parâmetros do algoritmo genético para análise da porcentagem dos melhores extensômetros.

Erro admissível	0,001
Tamanho da população	5
Elitismo	1
Taxa de mutação	0,2
Número de Baker	0,9
Numero máximo de iterações	2000

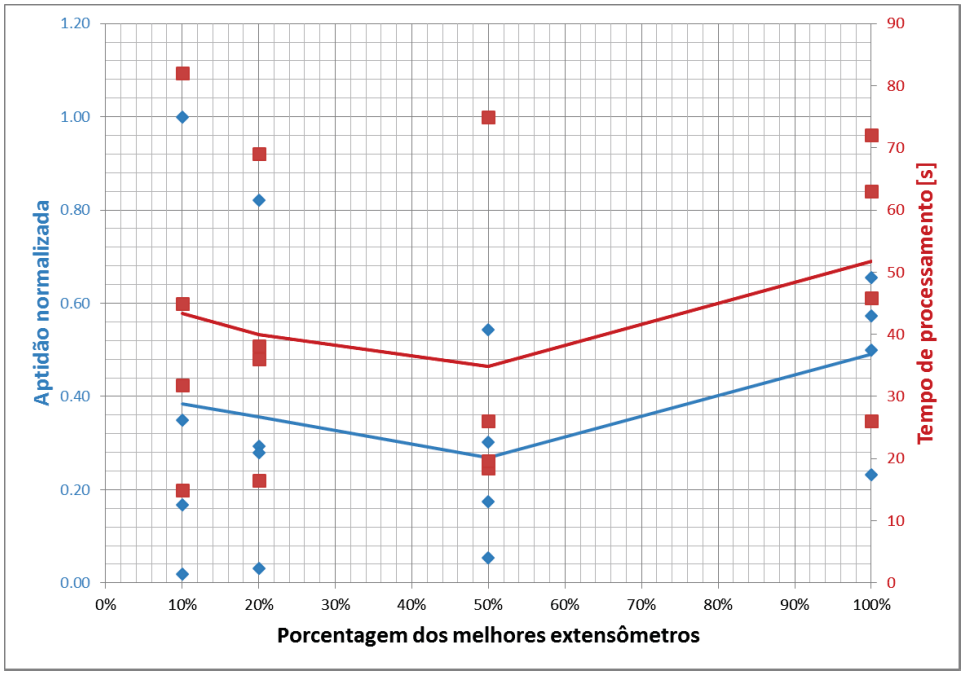


Figura 30 – Influência da porcentagem dos melhores extensômetros no tempo de processamento e aptidão final.

Como pode ser observado, não fica evidente uma influência significativa da porcentagem dos extensômetros com maior valor absoluto de deformação no tempo de processamento ou na evolução da aptidão do cromossomo mais apto, uma vez que os resultados apresentam dispersão e médias similares para todas as porcentagens.

4.2.3. Influência da taxa de mutação na eficiência do algoritmo

A influência da taxa de mutação na eficiência do algoritmo foi avaliada considerando-se os demais parâmetros do algoritmo constantes e com os valores mostrados na Tabela 9.

Tabela 9 – Parâmetros do algoritmo genético para análise da taxa de mutação.

Erro admissível	0,001
Tamanho da população	5
Elitismo	1
Porcentagem dos melhores extensômetros	0,2
Número de Baker	0,9
Numero máximo de iterações	2000

O algoritmo foi executado para as taxas de mutação 0,1, 0,2, 0,3 e 0,5, sendo repetido por 4 vezes para cada uma.

No gráfico da Figura 31 são apresentados os resultados para cada processamento e as curvas médias de valores de tempo e de aptidão normalizada para cada taxa de mutação.

Observa-se que existe um valor ótimo de aptidão para a taxa de mutação de 0.2, entretanto os valores de tempo de processamento para esta taxa de mutação é em média maior que as demais. Já para as taxas de mutação maiores que 0,2 observa-se que a aptidão média tende a reduzir. Não foram observados problemas de convergência do algoritmo para taxas de mutação até 0,5.

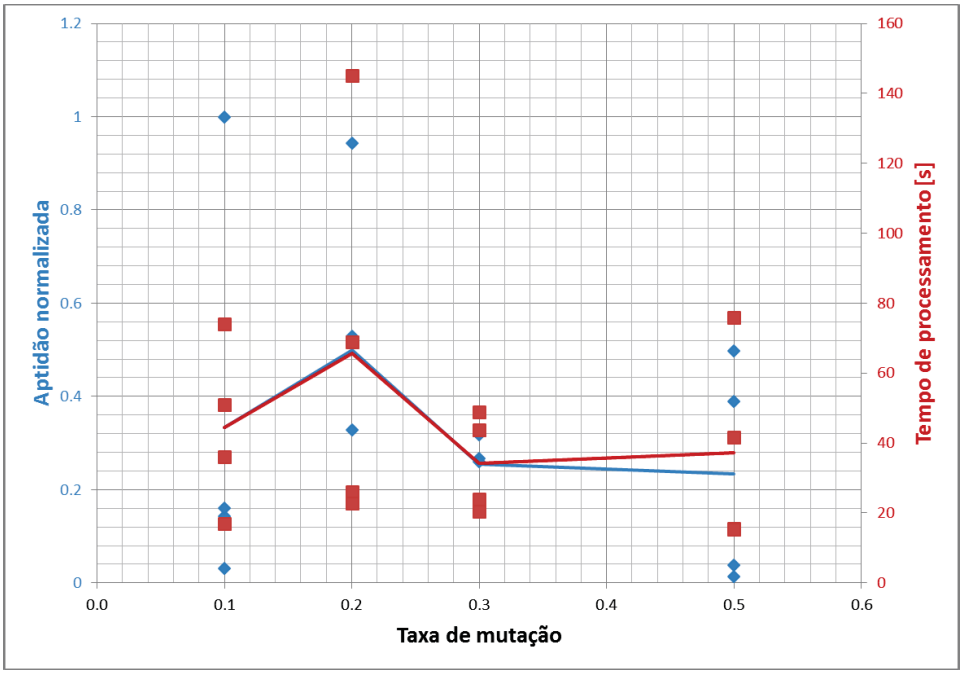


Figura 31 – Influência da taxa de mutação no tempo de processamento e aptidão final.

4.2.4. Influência do elitismo na eficiência do algoritmo

A influência do elitismo na eficiência do algoritmo foi avaliada considerando os demais parâmetros do algoritmo constantes e com os valores mostrados na Tabela 10.

O algoritmo foi executado para 0, 1, 2 e 3 cromossomos eleitos, sendo repetido por 4 vezes para cada ponto.

No gráfico da Figura 32 são apresentados os resultados para cada processamento e as curvas médias de valores de tempo e de aptidão normalizada para cada valor de elitismo.

Tabela 10 – Parâmetros do algoritmo genético para análise do elitismo.

Erro admissível	0,001
Tamanho da população	5
Taxa de mutação	0,2
Porcentagem dos melhores extensômetros	0,2
Número de Baker	0,9
Numero máximo de iterações	2000

Para todas as análises com valor de elitismo 0 o algoritmo sofreu parada prematura, desta forma, os valores de aptidão são mostrados como nulos no gráfico.

Para valores de elitismo de 1 a 3, observa-se uma tendência de crescimento da aptidão média normalizada com o aumento do número de cromossomos eleitos, sem demonstrar uma influência significativa nos tempos de processamento.

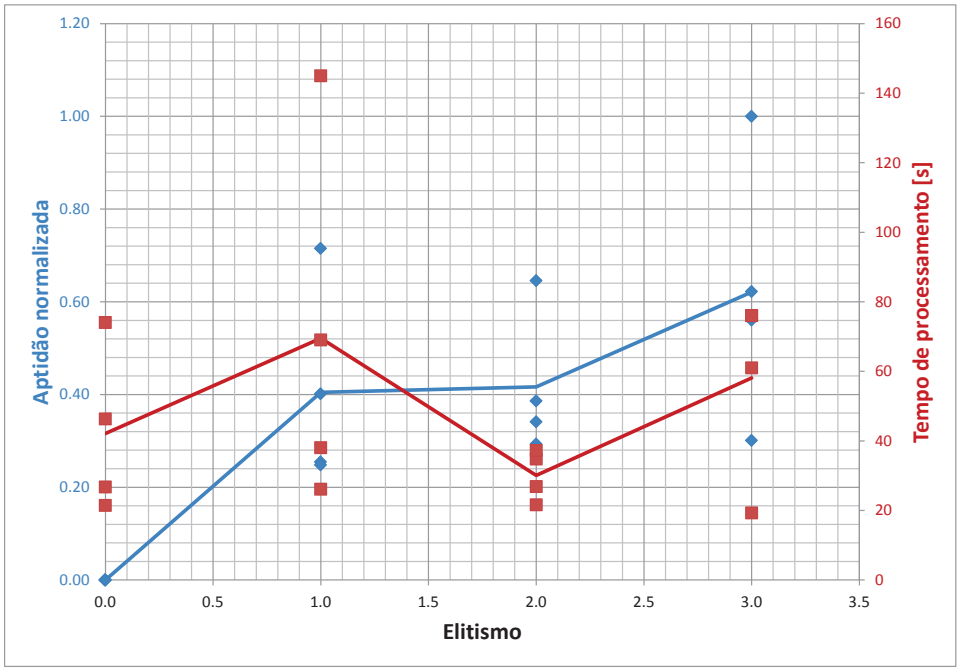


Figura 32 – Influência do elitismo no tempo de processamento e aptidão final.

4.2.5. Influência do número de Baker na eficiência do algoritmo

A influência do número de Baker na eficiência do algoritmo foi avaliada considerando-se os demais parâmetros do algoritmo constantes e com os valores mostrados na Tabela 11. O algoritmo foi executado para números de Baker de 0,3, 0,5, 0,7 e 0,9, sendo repetido por 4 vezes para cada ponto.

Tabela 11 – Parâmetros do algoritmo genético para análise do número de baker.

Erro admissível	0,001
Tamanho da população	5
Taxa de mutação	0,2
Porcentagem dos melhores extensômetros	0,2
Elitismo	1
Numero máximo de iterações	2000

No gráfico da Figura 33 são apresentados os resultados para cada processamento e para a média de valores de tempo e de aptidão normalizada para cada valor de elitismo.

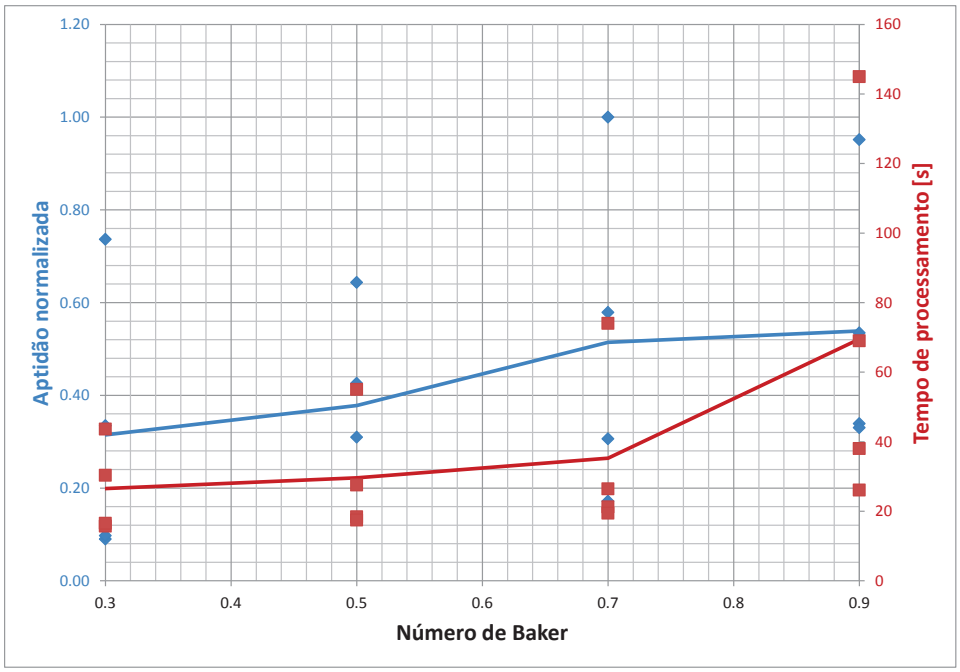


Figura 33 – Influência do número de Baker no tempo de processamento e aptidão final.

Observa-se que valores abaixo de 0,5 para o número de Baker geram uma pressão muito grande do cromossomo mais apto sobre o menos apto. Desta forma, o algoritmo converge prematuramente para um mínimo local, atingindo aptidões normalizadas menores.

Por outro lado, valores acima de 0,7 colocam muito pouca pressão do mais apto sobre o menos apto, aumentando os tempos de processamento, sem contribuir com a melhoria da aptidão média. O valor de 0,7 tem a melhor relação entre aptidões alcançadas e tempo de processamento.

4.2.6. Influência do erro admissível na eficiência do algoritmo

A influência do erro admissível na eficiência do algoritmo foi avaliada considerando-se os demais parâmetros do algoritmo constantes e com os valores

mostrados na Tabela 12. Diferentemente dos demais casos de estudo, para esta avaliação optou-se por aumentar o número máximo de iterações admissíveis, de forma a garantir que a parada do algoritmo ocorra sempre pelo critério de parada. O algoritmo foi executado para erros admissíveis de 0,1, 0,01, 0,001 e 0,0001, sendo repetido por 4 vezes para cada ponto. No gráfico da Figura 34 são apresentados os resultados para cada processamento e as curvas médias de valores de tempo e de aptidão normalizada para cada valor de erro admissível.

Tabela 12 – Parâmetros do algoritmo genético para análise do erro admissivel.

Tamanho da população	5
Taxa de mutação	0,2
Porcentagem dos melhores extensômetros	0,2
Elitismo	1
Número de Baker	0,9
Numero máximo de iterações	10000

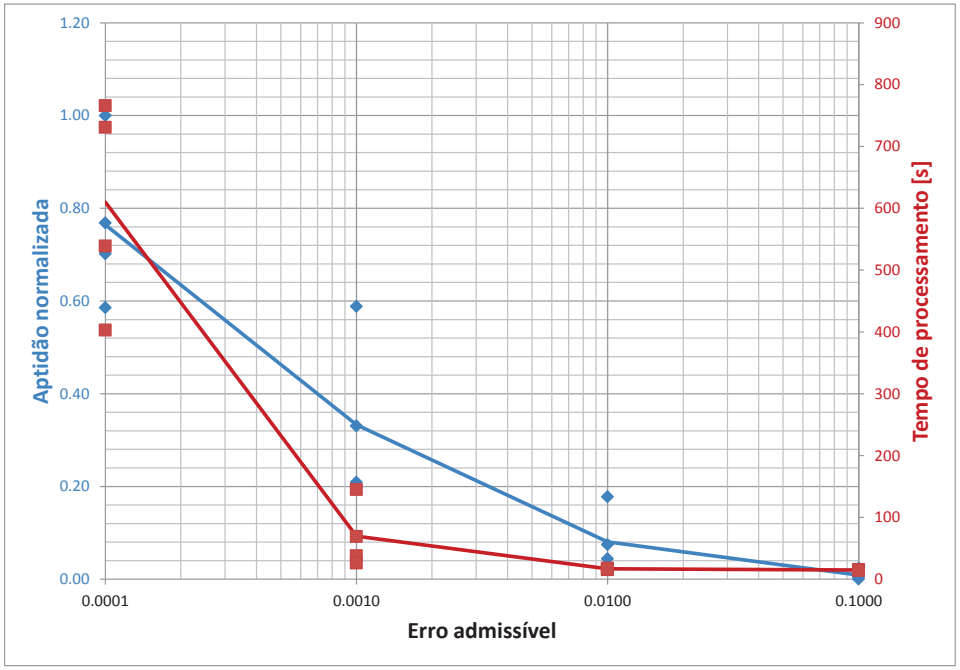


Figura 34 – Influência do erro admissivel no tempo de processamento e aptidão final.

Os valores médios da aptidão normalizada aumentam com a redução do erro admissível, conforme esperado. Entretanto, o custo computacional aumenta em aproximadamente 6 vezes para um ganho de aptidão normalizada de apenas 2 vezes. Este aumento de custo computacional pode tornar inviável a utilização de

valores de erro admissível da ordem de 0,0001 para problemas com domínios muito grandes e complexos.

4.2.7. Análise das eficiências dos algoritmos proposto e de Fedorov

Visando realizar de uma análise comparativa entre as eficiências do algoritmo proposto e o de Fedorov foram utilizados ensaios numéricos utilizando os parâmetros com os valores mostrados na Tabela 13.

Os valores escolhidos foram baseados nos resultados apresentados nas análises de eficiência de cada parâmetro. Na prática, o usuário do algoritmo irá se basear na experiência coletada em análises anteriores e nos estudos do presente trabalho. Desta forma, recomenda-se que a análise seja repetida algumas vezes variando-se os parâmetros para aumentar a probabilidade de encontrar o ótimo global.

Tabela 13 – Parâmetros do algorìtmo genético para comparativo de eficiência com o de Fedorov.

Erro admissível	0,001
Porcentagem dos melhores extensômetros	0,2
Elitismo	2
Taxa de mutação	0,2
Tamanho da população	200
Número de Baker	0,7
Numero máximo de iterações	2000

Foram executadas 10 análises com cada algoritmo. A aptidão foi normalizada pelo menor valor encontrado entre todas as análises com os dois algoritmos. Na Figura 35 são mostrados os valores de aptidão normalizada em função do tempo de processamento para todas as análises.

Observa-se que o tempo de processamento é praticamente constante para o algoritmo de Fedorov, devido ao fato de que, neste algoritmo, o número de iterações é constante e igual ao número de extensômetros que se deseja obter multiplicado pelo número de candidatos do domínio. Este tempo de processamento é até 100 vezes inferior aos tempos dispendidos para as análises com o AG.

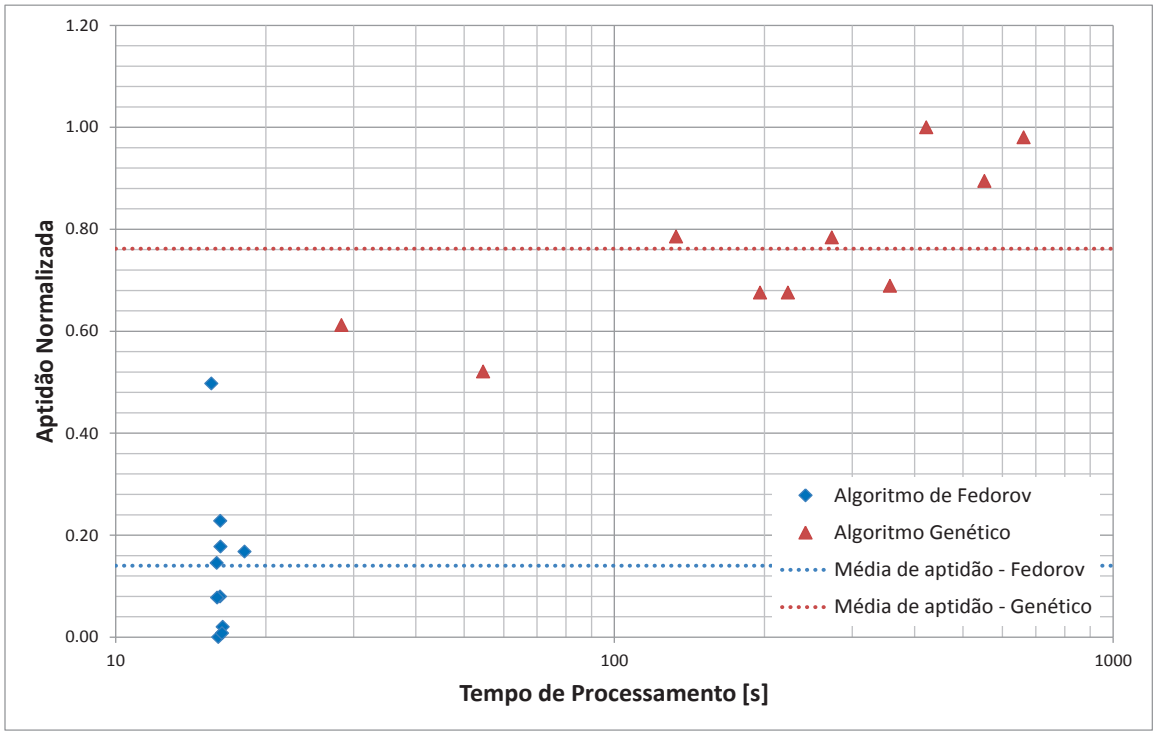


Figura 35 – Plotagem de aptidão normalizada final em função do tempo de processamento para os algoritmos de Fedorov e Genético.

Entretanto, fica também evidente que o valor médio das aptidões normalizadas encontrada com o AG é aproximadamente 13 vezes maior do que no algoritmo de Fedorov. Nas piores rodadas de análise, o resultado da aptidão do algoritmo de Fedorov chega a ser até 33000 vezes pior que a do AG. Este resultado evidencia que, frequentemente, o algoritmo de Fedorov converge para configurações de mínimos locais.

4.2.8. Análise da aptidão normalizada para diferentes quantidades de extensômetros na configuração ótima

Conforme evidenciado por Gupta (2013), o aumento no número de extensômetros utilizados para reconstruir certo número de carregamentos reduz consideravelmente o determinante da matriz de dispersão. Desta forma, foram realizados alguns testes utilizando os parâmetros mostrados na Tabela 13 e variando o número de extensômetros que se deseja obter na configuração final.

Foram realizados testes considerando 5, 10 e 15 extensômetros, sendo cada ponto repetido 5 vezes. Na Figura 36 são mostrados os valores de aptidão normalizada em função do tempo de processamento para todas as análises.

Observa-se um aumento de até 10 vezes no valor da aptidão normalizada quando se utiliza o dobro do número mínimo de extensômetros (neste caso, o mínimo são 5 extensômetros, uma vez que se deseja reconstruir 5 carregamentos), e de até 100 vezes quando se utiliza três vezes o número mínimo de extensômetros. Além disto, o tempo de processamento também é reduzido consideravelmente aumentando-se o número de extensômetros da configuração ótima. Desta forma, sempre que o usuário tiver disponibilidade de utilizar maior número de extensômetros do que o número de forças que se deseja reconstruir, é recomendável que o faça.

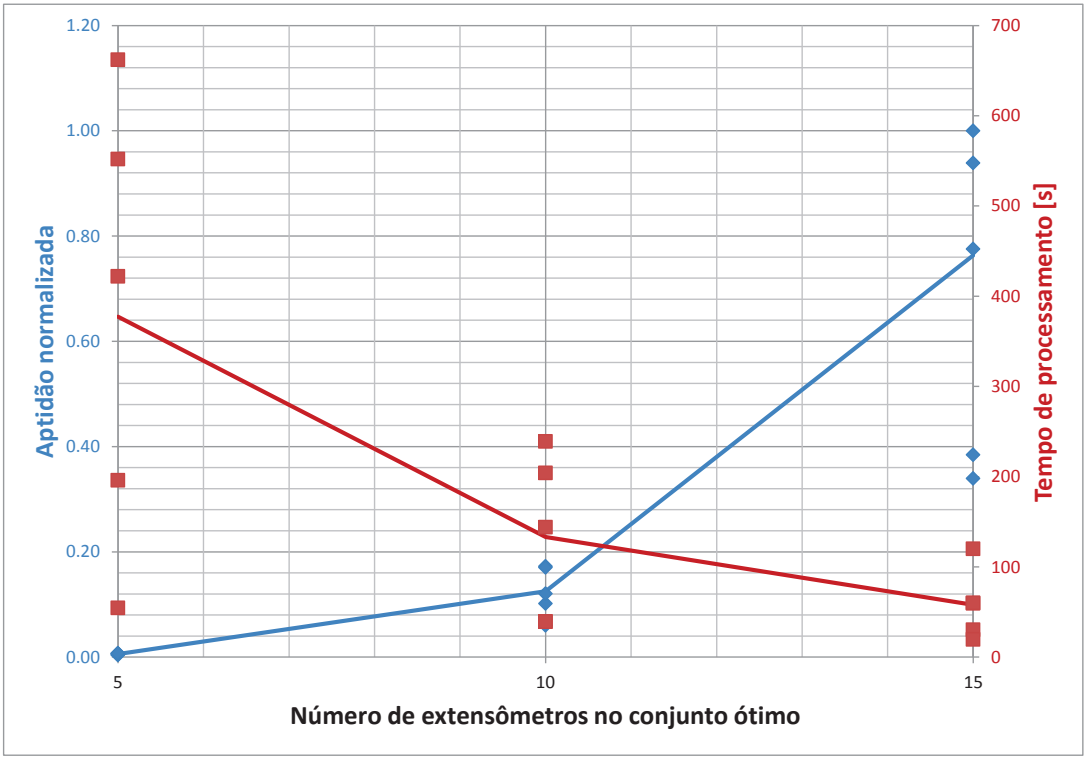


Figura 36 – Influência do número de extensômetros no conjunto ótimo no tempo de processamento e aptidão normalizada final.

4.3. TESTES FÍSICOS DE VALIDAÇÃO DA METODOLOGIA IMPLEMENTADA

Os dois componentes apresentados na seção 3.3 foram testados fisicamente e foram coletadas as deformações em cada um dos extensômetros para cada caso de carga.

Após a aquisição, os dados adquiridos foram tratados e as forças reconstruídas para cada caso de carga utilizando o software nCode®. Os resultados são apresentados nesta seção.

4.3.1. Análise dos resultados do teste para a alavanca

Inicialmente foi utilizado o modelo de elementos finitos da alavanca para a geração do domínio de configurações de extensômetros candidatas. Tendo este domínio, foi aplicado o processo de otimização utilizando o algoritmo genético. Uma série de testes consecutivos variando os parâmetros do algoritmo foi realizada, com o objetivo de obter um conjunto com 6 extensômetros (número mínimo para se reconstruir os 6 carregamentos propostos, conforme mostrado na Figura 13).

A configuração com a maior aptidão entre todos os testes foi implementada fisicamente. Além destes, outras 6 posições aleatórias foram escolhidas para permitir uma análise comparativa da eficiência do conjunto ótimo em relação a outros conjuntos aleatórios. Nas figuras Figura 37 e Figura 38 são mostrados os extensômetros da configuração ótima (extensômetros de 1 a 6), e os demais selecionados (extensômetros de 7 a 9), em sua representação virtual, ao lado de fotos dos mesmos aplicados fisicamente no componente.

Os valores médios de deformações lidas em cada extensômetro para cada um dos testes realizados podem ser vistos na Tabela 14.

A partir das deformações medidas foi feita a reconstrução dos carregamentos, e calculados os ângulos de inclinação ψ e rotação ϕ da peça baseados na direção da força resultante média entre os dois pontos de carregamento.

Para que fosse possível fazer uma avaliação da eficiência do conjunto ótimo de extensômetros em relação a outros possíveis conjuntos, 5 combinações

aleatórias de extensômetros entre os 12 instrumentados foram geradas. A Tabela 15 mostra as combinações selecionadas.

Tabela 14 – Deformações medidas com extensômetros nos testes com a alavanca.

		Casos de carga								
		1	2	3	4	5	6	7	8	9
Extensômetros	1	0,69	0,79	-1,67	-101,15	-132,76	-77,81	-103,01	-132,54	-83,18
	2	-1,13	-1,45	-2,37	-21,84	-18,49	-160,63	-24,78	-12,63	-215,25
	3	-0,27	-0,45	0,12	17,16	45,18	10,95	19,48	50,38	13,82
	4	-139,56	-154,52	91,00	-125,46	-42,56	18,28	1,48	-0,79	-0,72
	5	-7,24	-14,88	196,49	20,55	27,92	145,07	-0,69	-2,36	-0,88
	6	-116,64	-72,06	-295,12	-49,65	-32,36	-156,39	-1,68	-0,88	-0,01
	7	-4,99	6,64	-122,78	-18,30	-25,88	-92,65	-9,11	-8,85	-9,26
	8	11,13	18,31	-83,95	8,22	-5,41	-49,78	-2,18	-2,31	-2,30
	9	43,64	27,32	24,75	48,00	36,81	40,62	0,03	-0,31	-0,06
	10	-29,67	-25,88	-29,77	-19,58	-9,08	-17,26	8,79	8,29	12,78
	11	33,82	27,05	16,68	37,65	29,74	31,06	8,26	8,12	10,94
	12	6,17	-6,02	116,27	24,27	29,00	97,19	10,63	10,73	11,68

* Valores em μm/m.

Para que fosse possível fazer uma avaliação da eficiência do conjunto ótimo de extensômetros em relação a outros possíveis conjuntos, 5 combinações aleatórias de extensômetros entre os 12 instrumentados foram geradas. A Tabela 15 mostra as combinações selecionadas.

Tabela 15 – Combinações aleatórias de extensômetros da alavanca.

Combinação 1	Combinação 2	Combinação 3	Combinação 4	Combinação 5
2	1	1	2	2
3	2	2	4	3
5	4	3	7	5
6	8	6	8	9
7	9	9	9	10
12	11	12	12	11

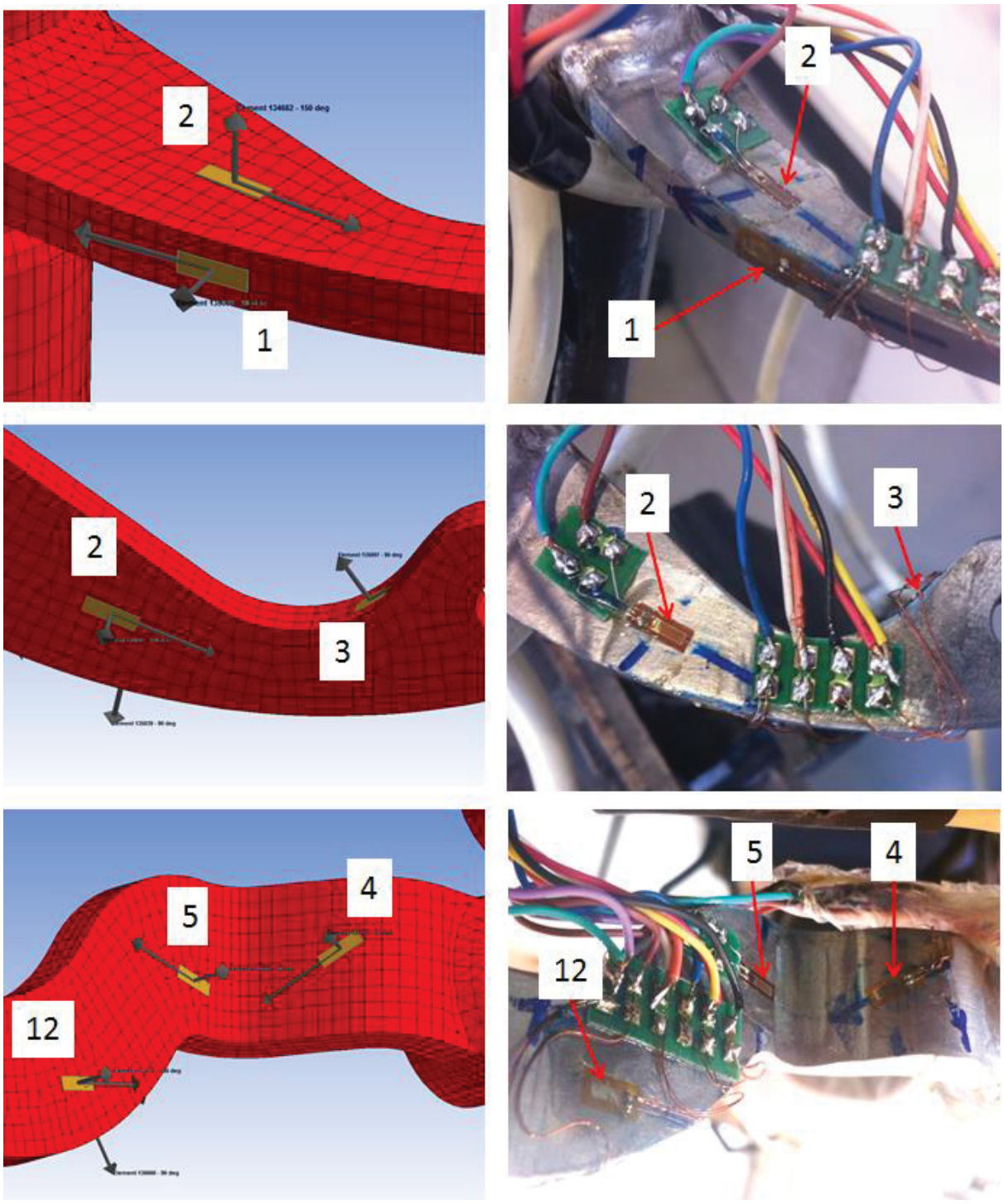


Figura 37 – Extensômetros aplicados à alavanca virtualmente e fisicamente.

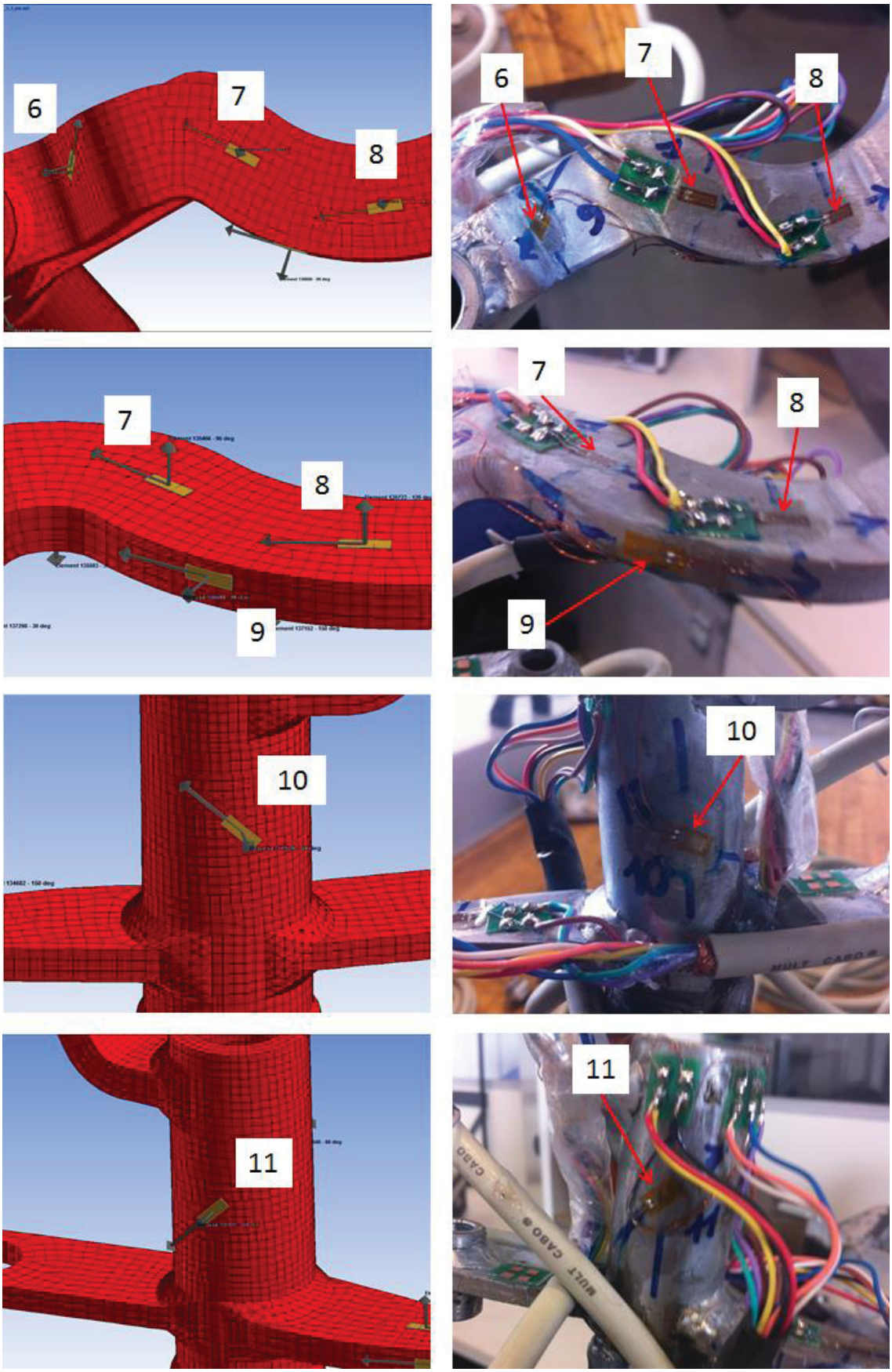


Figura 38 – Extensômetros aplicados à alavanca virtualmente e fisicamente.

Os valores de forças e ângulos reconstruídos para a configuração ótima de extensômetros podem ser vistos na Tabela 16. A partir do conhecimento das massas aplicadas aos dois braços da alavanca (ver Tabela 5), foram calculados os valores de força esperados. A diferença entre estes valores esperados de força e os valores reconstruídos foram também calculados e podem ser vistos na Tabela 17.

Tabela 16 – Força e ângulos reconstruídos para a configuração ótima de extensômetros.

Casos de carga	Valores esperados				Valores reconstruídos			
	Força em A [N]	Força em B [N]	Rotação [graus]	Inclinação [graus]	Força em A [N]	Força em B [N]	Rotação [graus]	Inclinação [graus]
1	113,4	0,0	0	0	103,2	0,9	-20	11
2	113,4	0,0	45	0	76,2	1,0	27	14
3	113,4	0,0	0	-45	128,7	2,9	-30	-38
4	113,4	113,8	0	0	131,3	148,4	11	-9
5	113,4	113,8	-37	0	76,3	155,9	-7	-3
6	113,4	113,8	0	-30	134,2	132,2	13	-27
7	0,0	113,8	0	0	6,6	147,1	9	-4
8	0,0	113,8	-45	0	7,3	149,3	-12	-2
9	0,0	113,8	0	-38	1,7	145,4	11	-33

Tabela 17 – Diferença de força e ângulos entre os valores esperados e reconstruídos para a configuração ótima de extensômetros.

Casos de carga	Diferença de força em A [N]	Diferença de força em B [N]	Diferença de rotação [graus]	Diferença de inclinação [graus]
1	10,23	0,87	19,95	10,72
2	37,24	0,97	17,94	13,75
3	15,28	2,89	30,18	7,43
4	17,86	34,62	10,96	9,26
5	37,13	42,11	30,40	2,70
6	20,80	18,41	12,80	3,04
7	6,58	33,33	9,34	3,59
8	7,27	35,47	33,16	2,08
9	1,69	31,62	10,68	4,73

Observa-se que as diferenças de forças reconstruídas em relação aos valores esperados é de até 37%. As diferenças para os ângulos de rotação chega a 82% e para os ângulos de inclinação 16%.

Alguns fatores contribuem para estes erros elevados:

- Erros associados à fabricação do componente, uma vez que ele tem pequenas dimensões e passa por vários processos de conformação e soldagem;
- Erros associados ao posicionamento dos extensômetros em relação à posição no modelo de elementos finitos;
- Erros associados às medições propriamente ditas dos extensômetros;
- A alta rigidez da estrutura, fazendo com que os valores de deformação lidos fossem muito pequenos para os carregamentos da estrutura e, consequentemente, piorando a relação sinal-ruído da leitura.
- Erros associados ao MEF no cálculo de deformações.

A diferença entre os valores esperados de força e ângulo e os valores reconstruídos para as combinações de 1 a 5 foram também calculados e podem ser vistos nas tabelas Tabela 18 a Tabela 22.

Tabela 18 – Diferença de força e ângulos entre os valores esperados e reconstruídos para a combinação 1 de extensômetros.

Casos de carga	Diferença de força em A [N]	Diferença de força em B [N]	Diferença de rotação [graus]	Diferença de inclinação [graus]
1	40,10	15307410,88	80,56	5,90
2	48,78	10221898,01	32,15	23,05
3	117,98	9899341,47	39,94	24,88
4	39,65	18500551,76	22,03	5,80
5	6,86	8563691,25	59,02	0,16
6	2,19	11574027,99	22,02	30,16
7	129,25	4463680,52	22,03	0,16
8	143,55	5541282,10	67,02	0,16
9	142,54	6701889,71	22,02	38,16

Tabela 19 – Diferença de força e ângulos entre os valores esperados e reconstruídos para a combinação 2 de extensômetros.

Casos de carga	Diferença de força em A [N]	Diferença de força em B [N]	Diferença de rotação [graus]	Diferença de inclinação [graus]
1	13,58	1936,01	0,97	2,22
2	9,89	2232,51	20,44	3,82
3	73,73	122,51	51,12	20,40
4	11,73	2697,14	58,75	2,55
5	43,82	2636,12	96,49	1,35
6	101,60	1504,59	59,50	35,17
7	3,08	2071,01	59,43	1,54
8	9,62	2251,12	104,95	1,32
9	7,97	2034,11	58,94	43,11

Tabela 20 – Diferença de força e ângulos entre os valores esperados e reconstruídos para a combinação 3 de extensômetros.

Casos de carga	Diferença de força em A [N]	Diferença de força em B [N]	Diferença de rotação [graus]	Diferença de inclinação [graus]
1	2822,69	0,49	87,10	54,42
2	366,11	1,04	123,44	25,24
3	775,24	2,81	78,70	45,71
4	17858,43	37,07	11,17	20,70
5	16995,21	44,19	30,85	1,86
6	18836,73	19,86	13,03	4,51
7	2410,96	33,63	9,37	3,46
8	2411,03	35,73	33,24	1,96
9	2856,72	31,78	10,72	4,94

Tabela 21 – Diferença de força e ângulos entre os valores esperados e reconstruídos para a combinação 4 de extensômetros.

Casos de carga	Diferença de força em A [N]	Diferença de força em B [N]	Diferença de rotação [graus]	Diferença de inclinação [graus]
1	0,07	2113042363,73	29,83	6,58
2	56,32	1142451740,60	24,79	9,58
3	79,92	829634077,90	51,97	21,11
4	10,27	3802564913,60	64,78	1,60
5	25,53	2539212806,95	101,78	1,37
6	26,32	3266618304,83	64,78	31,37
7	81,50	424089365,55	64,73	1,37
8	83,84	537495610,59	109,75	1,37
9	93,34	722971738,38	64,76	39,37

Tabela 22 – Diferença de força e ângulos entre os valores esperados e reconstruídos para a combinação 5 de extensômetros.

Casos de carga	Diferença de força em A [N]	Diferença de força em B [N]	Diferença de rotação [graus]	Diferença de inclinação [graus]
1	15,31	546,95	28,55	10,39
2	8,28	660,62	20,83	15,62
3	32,07	107,79	39,66	8,48
4	106,87	590,68	24,34	3,24
5	161,55	562,38	65,35	0,91
6	136,13	196,63	25,60	42,30
7	214,79	462,16	25,23	1,21
8	207,02	526,55	74,48	0,76
9	240,34	491,52	24,31	46,40

Como pode ser observado, a diferença entre as forças reconstruídas e os valores esperados para todas as combinações é muito maior do que a diferença encontrada com a configuração ótima.

Algumas das combinações aleatórias geradas mostram valores exorbitantes para as forças em um dos braços da alavanca. Isto ocorre porque estas combinações não são capazes de reconstruir adequadamente todas as forças que ocorrem nos dois braços da alavanca, mesmo que a configuração tenha a quantidade mínima de extensômetros necessária.

Isto prova a necessidade de se utilizar um algoritmo de otimização para determinar os pontos ótimos de instrumentação, já que a escolha de pontos de forma aleatória, ou até mesmo utilizando a experiência do experimentador, pode levar a configurações que não são capazes de reconstruir 1 ou mais carregamentos desejados.

Considerando as combinações capazes de reconstruir as forças para os dois braços da alavanca, como a combinação 5 por exemplo, a diferença de valores de força entre as reconstruídas e as esperadas é até 35 vezes maior do que a diferença encontrada na configuração ótima de extensômetros. Esta observação evidencia a redução da variância que a configuração ótima promove.

4.3.2. Análise dos resultados de teste para o suporte

Assim como para a alavanca, o modelo de elementos finitos do suporte foi utilizado para a geração de seu domínio de candidatos. Este domínio foi posteriormente otimizado com o algoritmo genético. Uma série de testes consecutivos variando os parâmetros do algoritmo foi realizado com o objetivo de obter conjuntos com 3, 5 e 7 extensômetros. Essas configurações com diferentes números de extensômetros permitem avaliar a redução da variância da reconstrução de cargas com o aumento do número de extensômetros.

Após a determinação das configurações ótimas considerando 3, 5 e 7 extensômetros, observou-se que alguns deles eram comuns aos 3 conjuntos. Desta forma, o número total de extensômetros aplicados à estrutura foi reduzido para apenas 10.

Além destas, outras 4 posições aleatórias foram escolhidas para permitir uma análise comparativa da eficiência do conjunto ótimo em relação a outros conjuntos aleatórios. Na Figura 39 são mostrados os extensômetros das configurações ótimas (extensômetros de 1 a 10), e os demais selecionados (extensômetros de 11 a 14), em sua representação virtual, ao lado de fotos dos mesmos aplicados fisicamente no componente. A Tabela 23 mostra as combinações ótimas com as numerações conforme mostrado na Figura 39.

Tabela 23 – Combinações ótimas de extensômetros do suporte.

Combinação ótima com 3 extensômetros	Combinação ótima com 5 extensômetros	Combinação ótima com 7 extensômetros
1	1	2
2	2	3
5	4	5
	5	6
	8	7
		9
		10

A partir das deformações medidas foi feita a reconstrução dos carregamentos aplicados.

Para que fosse possível fazer uma avaliação da eficiência dos conjuntos ótimos de extensômetros em relação a outros possíveis conjuntos, 6 combinações aleatórias de extensômetros entre os 14 instrumentados foram geradas, sendo 2 combinações com 3 extensômetros, duas com 5 e duas com 7. A

Tabela 24 mostra as combinações aleatórias selecionadas.

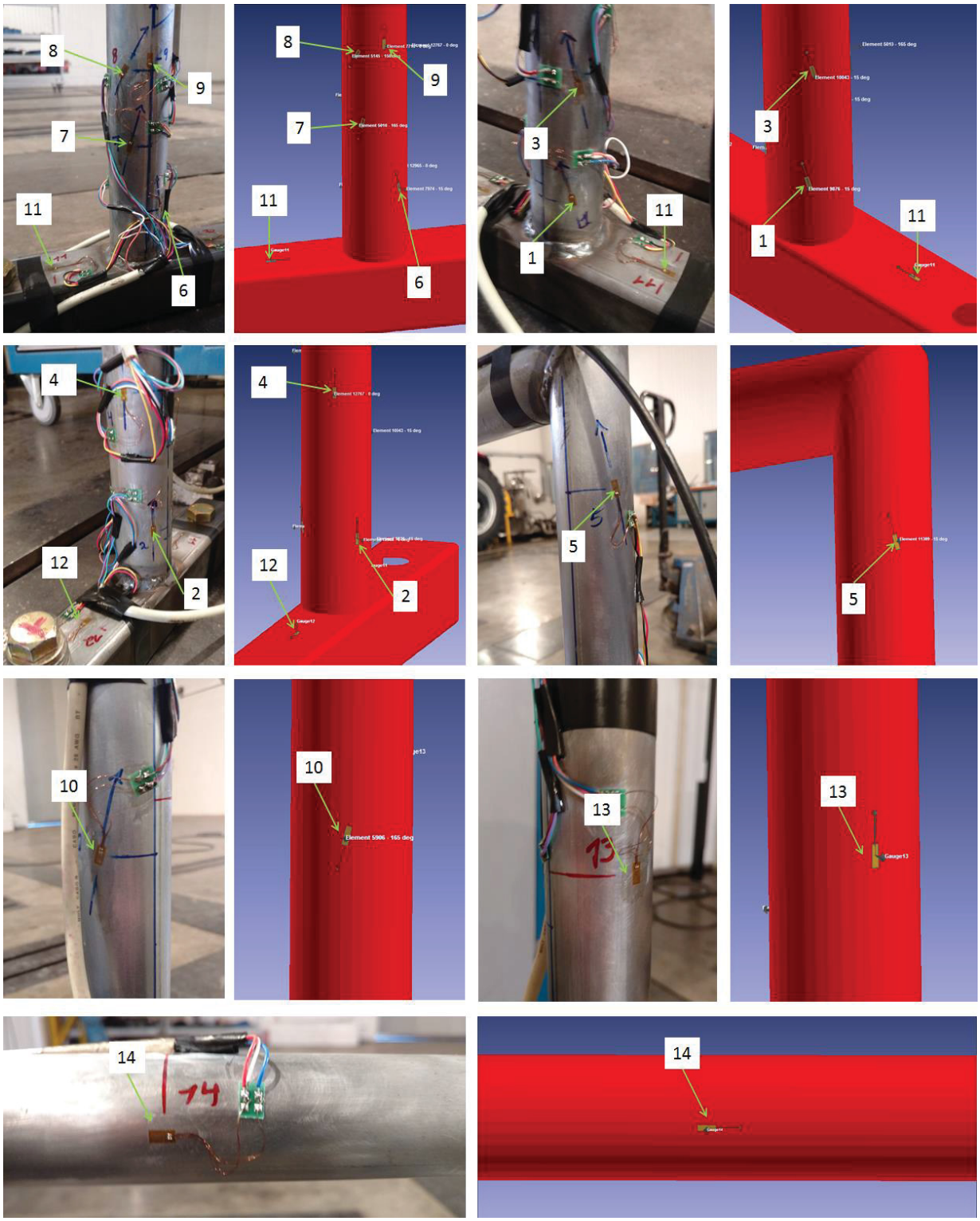


Figura 39 – Extensômetros aplicados ao suporte virtualmente e fisicamente.

Tabela 24 – Combinações aleatórias de extensômetros do suporte.

Combinação 1	Combinação 2	Combinação 3	Combinação 4	Combinação 5	Combinação 6
4	4	3	3	2	2
9	8	5	7	5	4
11	12	10	11	6	5
		13	12	8	7
		14	13	10	8
				11	13
				12	14

No gráfico da Figura 40 são plotadas as diferenças porcentuais das estimativas de força em relação aos valores esperados (conforme mostrado na Tabela 6) para cada caso de carga e para cada uma das combinações de extensômetros.

Observa-se que as configurações com 7 extensômetros possui, em geral, um erro menor comparativamente às configurações com 3 extensômetros, e em alguns casos melhor do que as configurações com 5 extensômetros. Este resultado comprova que existe uma redução na variância generalizada das estimativas de força quando se adicionam mais extensômetros.

Observa-se que a combinação 4 foi aquela que apresentou os menores valores de erro. Apesar de não ser a configuração ótima, esta combinação possui uma aptidão comparável às encontradas nas configurações ótimas.

No gráfico da Figura 41 são mostradas as diferenças porcentuais das estimativas de carga para as configurações ótimas e também para as aleatórias em função do determinante da matriz de dispersão de cada configuração. Quanto menor o valor do determinante maior a aptidão da configuração.

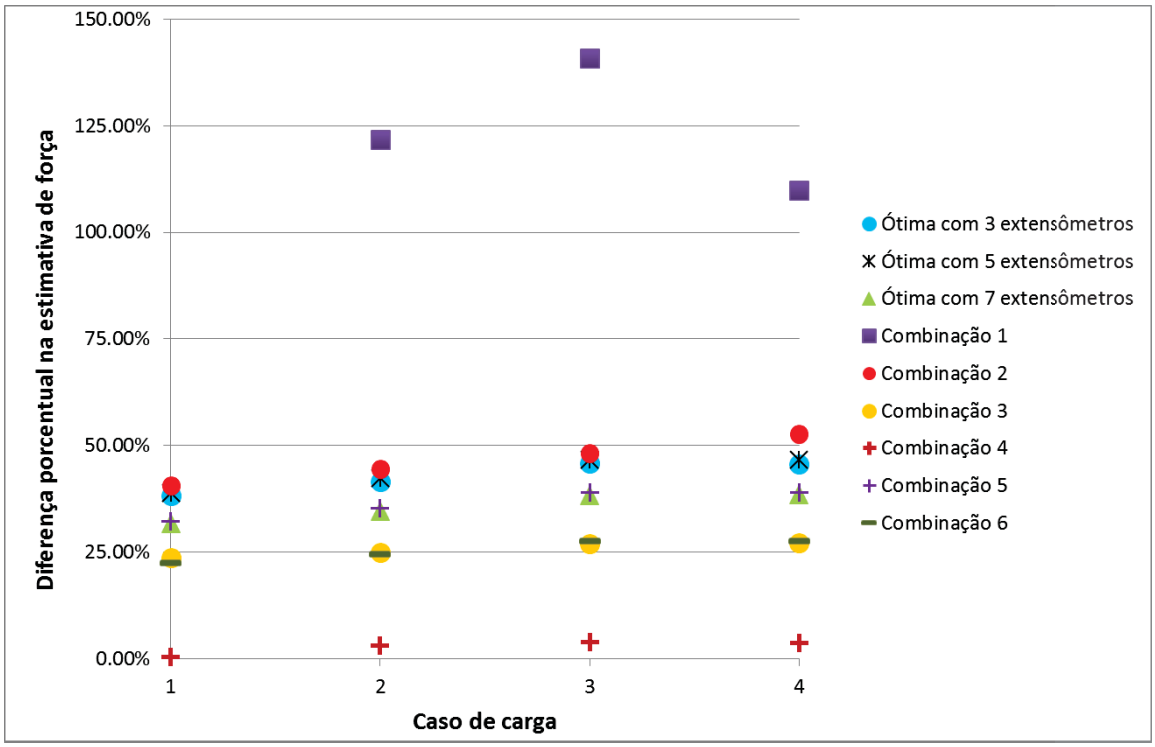


Figura 40 – Diferença porcentual na estimativa de força para cada caso de carga do suporte.

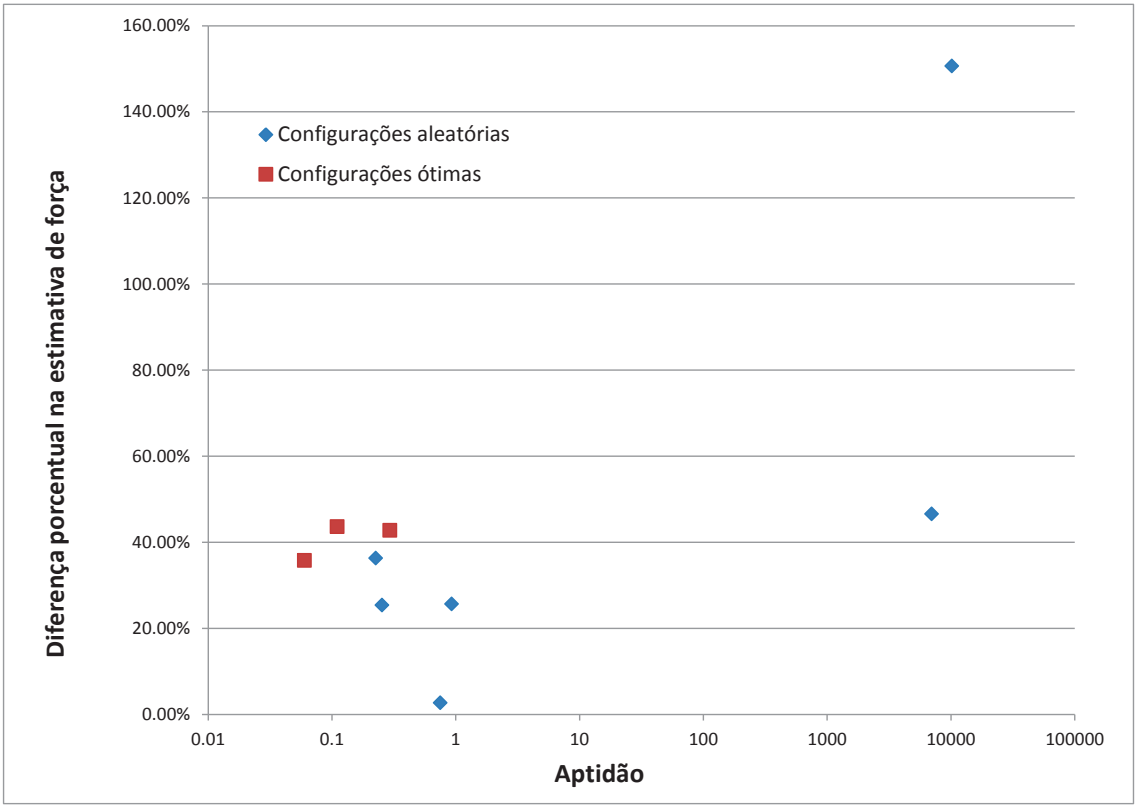


Figura 41 – Diferença porcentual na estimativa de força para cada configuração de extensômetros.

5 CONCLUSÕES E SUGESTÕES PARA TRABALHOS FUTUROS

Este trabalho buscou desenvolver, implementar e testar uma metodologia para a identificação de carregamentos mecânicos de um componente mecânico em operação, utilizando-o como seu próprio transdutor, sempre que for possível assumi-los como quase estáticos.

Para tanto, é necessário que se determine uma configuração otimizada de extensômetros a ser aplicada sobre o componente. Nesse sentido, um algoritmo de otimização genético foi implementado de forma a fornecer esta configuração ótima utilizando o conceito de projeto experimental D-ótimo. Ao final do projeto, o algoritmo foi validado através de testes físicos e virtuais.

5.1. CONCLUSÕES

Os algoritmos para a geração do domínio de candidatos e o de otimização foram implementados com sucesso, embasados nas teorias de projeto experimental ótimo. Ao término do trabalho, a interface desenvolvida permite operá-los com facilidade até mesmo por usuários inexperientes. Durante todos os testes executados também se observou uma boa confiabilidade e estabilidade dos algoritmos.

A análise dos parâmetros do algoritmo mostrou intervalos de funcionamento que permitem obter melhores respostas com relação à convergência, tempo de processamento e qualidade da configuração de extensômetros obtida. Verificou-se que, para grande parte dos parâmetros do algoritmo genético, o intervalo de valores que demonstram maior eficiência é, em geral, similar àquele recomendado na bibliografia pesquisada.

Constatou-se que o algoritmo genético implementado atinge valores médios de aptidão muito maiores do que o algoritmo de Fedorov, apesar de ter um custo computacional maior que este. Ficou evidenciada uma constância interessante nos valores da função objetivo encontrados pelo algoritmo genético, aproximando-se sempre do mínimo global da função, ao contrário do algoritmo de Fedorov que converge sempre para mínimos locais.

O número de extensômetros utilizados para reconstruir um determinado número de carregamentos mostrou ter um grande impacto na minimização da função objetivo, contribuindo ainda para a redução dos tempos de processamento. Entretanto, este impacto não ficou tão evidente no teste físico que pretendia comprová-lo.

Os testes físicos realizados mostraram que configurações de extensômetros escolhidas baseadas na experiência do experimentador ou aleatoriamente podem levar a valores de carregamentos reconstruídos muito diferentes dos reais. Isso corrobora com a comprovação de que uma configuração otimizada pelo critério D-ótimo é capaz de reduzir a variância dos mesmos.

Verificou-se que os erros de fabricação, posicionamento dos extensômetros, o nível de sinal para ruído e os erros associados ao MEF podem gerar erros consideráveis nos valores finais dos carregamentos obtidos.

Por fim, constatou-se a viabilidade de se utilizar esta metodologia como meio de obtenção das solicitações mecânicas para um componente em operação desde que os devidos cuidados na preparação do experimento e fabricação do componente sejam tomados.

5.2. SUGESTÕES PARA TRABALHOS FUTUROS

Buscando o contínuo aprimoramento e de forma a estudar melhorias possíveis para a metodologia, sugerem-se como propostas de estudos para trabalhos futuros:

I. Estudar a eficiência dos demais critérios de projeto experimental ótimo citados neste trabalho na redução da variância dos carregamentos reconstruídos;
II. Estudar uma metodologia que permita reduzir os erros de posicionamento dos extensômetros;
III. Aplicar esta metodologia de reconstrução a carregamentos dinâmicos;
IV. Realizar testes utilizando extensômetros com maior sensibilidade (maior fator de extensômetro);

V. Realizar um estudo das diferentes fontes de erro que dificultam a obtenção de valores precisos para os carregamentos reconstruídos.

REFERÊNCIAS

AGUIAR, P. F. de *et al*. D-optimal designs. **Chemometrics and Intelligent Laboratory Systems**, v. 30, p. 199-210, 1995.

ALTAIR **HyperWorks**. Version 12.0. [*S.l.*]: Altair Engineering Inc., 2014.

ANGELO, J. F. **Aplicação de Projeto Experimental Ótimo à Reação de Interesterificação de Estearina de Palma com Óleo de Linhaça**. Dissertação de Mestrado em Engenharia Química. São Paulo: Escola Politécnica da Universidade de São Paulo, 2007.

ARORA, J. S. **Introduction to Optimum Design**. 2. ed. San Diego: Elsevier, 2004.

BARKANOV, E.; **Introduction to the Finite Element Method**. Institute of Materials and Structures Faculty of Civil Engineering Riga Technical University. Riga, 2001. Disponível em: < http://icas.bf.rtu.lv/doc/Book.pdf>. Acesso em: 15 jan. 2016.

BEN-ISRAEL, A.; GREVILLE T. N. E. **Generalized Inverses. Theory and Applications.** 2. ed. New York: Springer, 2003.

BICCHI, A. A Criterion for Optimal Design of Multi-axis Force Sensors. **Robotics and Autonomous Systems,** v. 10, p. 269-286, 1992.

BROUDISCOU, A.; LEARDI, R.; PHAN-TAN-LUU, R. Genetic Algorithm as a Tool for Selection of D-Otpimal Design. **Chemometrics and Intelligent Laboratory Systems**, v. 35, p. 105-116, 1996.

COOK, R. D.; MALKUS, D. S.; PLESHA, M. E. **Conceps and Applications of Finite Element Analysis**. 3. ed. Madison: John Wiley & Sons, 1989.

DETTE, H.; PEPELYSHEV, A.; ZHIGLJAVSKY, A. Improving Updating Rules in Multiplicative Algorithms for Computing D-optimal Designs. **Computational Statistics and Data Analysis,** v. 53, p. 312-320, 2008.

FEDOROV, V. V. **Theory of Optimal Experiments.** New York: Academic Press, 1972.

GAO, W. *et al.* Efficient Computational Algorithm for Optimal Allocation in Regression Models. **Journal of Computational and Applied Mathematics,** v. 261, p. 118-126, 2014.

GUPTA, D. K.; **Inverse Methods for Load Identification Augmented by Optimal Sensor Placement.** PhD Thesis. Milwaukee: University of Wisconsin-Milwaukee, 2013.

HBM **nCode**. Version 10.0. [*S.l.*]: HBM United Kingdom Limited, 2014.

HIBBELER, R. C. **Resistência dos Materiais**. 7. ed. São Paulo: Pearson Prentice Hall, 2010.

HUNTER T. G. Experimental Correlation of an N-Dimensional Load Transducer Augmented By Finite Element Analysis. In: INTERNATIONAL CONFERENCE ON SERVICE COMPUTING, 9., 2012, Honolulu, Hawaii, USA. **2012 3DS SIMULIA Community Conference**. Disponível em: <http://www.simulia.com/SCCProceedings2012/content/>. Acesso em: 28 ago. 2014.

ILANKAMBAN, R.; PERUMALSWAMI, P.R. Back Load Calculation a Method of Measuring Component Loads Without Load Cells**.** In: MSC WORLD USERS' CONFERENCE, 1996, Monterey, California, USA. **MSC 1996 World Users' Conference Proceedings**. Disponível em: < http://web.mscsoftware.com/support/library/conf/wuc96/>. Acesso em: 28 ago. 2014.

KIEFER, J.; WOLFOWITZ, J. Optimum design in regression problems. **Mathematical Statistics**, v. 30, p. 271-294, 1959.

KUTNER, M. *et al.* **Applied Linear Statistical Models**. 5. ed. New York: McGraw-Hill, 2004.

LIAO, H. **Construction of Approximate Optimal Designs by Exchange Algorithm**. Dissertação de Mestrado em Matemática Aplicada. Taiwan: National Sun Yat-sen University Library, 2001.

MANDAL, A.; TORSNEY, B. Construction of Optimal Designs Using a Clustering Approach. **Journal of Statistical Planning and Inference**, v. 136, p.1120-1134 2006.

MANDAL, A.; WONG, W. K.; Yu, Y. Algorithmic Searches for Optimal Designs. In: BINGHAM, D. *et al*. **Handbooks of Modern Statistical Methods: Handbook of Design and Analysis of Experiments**. Ames, USA: Chapman & Hall/CRC Press, 2014.

MASROOR, S.A.; ZACHARY, L.W. Designing an All-purpose Force Transducer. **Experimental Mechanics**, v.31, p. 33-35, 1991.

MILLER, A.J.; NGUYEN, N. Algorithm AS 295: A Fedorov Exchange Algorithm for D-Optimal Design**. Journal of the Royal Statistical Society. Series C (Applied Statistics)**, v. 43, p. 669-677, 1994.

NGUYEN, N.; MILLER, A.J. A Review of Some Exchange Algorithms for Constructing Discrete D-Optimal Designs**. Computational Statistics & Data Analysis**, v. 14, p. 489-498, 1992.

O'BRIEN, T. E.; FUNK, G.M. A Gentle Introduction to Optimal Design for Regression Models. **The American Statistician**, v. 57, p.265-267, 2003.

RAMOS, H.O.C. **Um Algoritmo para Otimização Restrita com Aproximação de Derivadas.** Tese de Doutorado em Engenharia Mecânica. Rio de Janeiro: Universidade Federal do Rio de Janeiro, 2011.

REDDY, J. N. **An Introduction to the Finite Element Method**. 3. ed. Mc Graw Hill, 2006.

SMITH, K. On the Standard Deviations of Adjusted and Interpolated Values of an Observed Polynomial Function and its Constants and the Guidance They Give Towards a Proper Choice of the Distribution of Observations. **Science & Mathematics**, Biometrika, v. 12, p. 1–85, 1918.

SMUCKER, B.; CASTILLO, E.; ROSENBERGER, J.L. Exchange Algorithms for Constructing Model-Robust Experimental Designs. **Journal of Quality Technology**, v. 43, p. 28-42, 2011.

TRIEFENBACH, F. **Design of Experiments: The D-Optimal approach and its implementation as a computer algorithm**. Trabalho de Graduação em Tecnologia da Informação e Comunicação. Meschede: South Westphalia University of Applied Sciences, 2008.

WALD, A. On the efficient design of statistical investigation**. The Annals of Mathematical Statistics**, v. 14, no. 2, p. 134-140, 1943.

WICKHAM, M. *et al*. **The Design and Application of a Multi-Axis Load Transducer**. SAE Technical Paper 940250, 1994.

WONG, W.K. Comparing Robust Properties of A, D, E and G-Optimal Designs. **Computational Statistics & Data Analysis**, v. 18, p. 441-448, 1993.

XU, H. **Comparison of Genetic Operators on a General Genetic Algorithm Package**. Tese de Mestrado. Shangai: Shanghai Jiao Tong University, 1999.

YU, Y. Monotonic Convergence of a General Algorithm for Computing Optimal Designs. **The Annals of Statistics,** v. 38, p. 1593-1606, 2010.

YU, Y. D-Optimal Designs Via a Cocktail Algorithm. **Statistics and Computing**, v. 21, p. 475-481, 2011.

APÊNDICES

APÊNDICE 1 – ALGORÍTMO DE GERAÇÃO DO DOMÍNIO DE OTIMIZAÇÂO

```
# ------------------------------------------------------------------------------------#
#            Script - Virtual Strain Gages Set Generation              #
#                                                                      #
#                        Developed by DIEGO HOEPFNER                   #
#                                                                      #
#            Unauthorized copying or distribution is strictly forbidden #
# ------------------------------------------------------------------------------------#

tk_messageBox -message "Please define the angle between surfaces to avoid hard edges"
set featureangle [hm_getint "Define the feature angle:" "Define the angle between surfaces to avoid hard edges"]

*clearmark comps 1
tk_messageBox -message "Please select the components to remove the elements adjacent to the hard edges"
*createmarkpanel comps 1 "Select the components to find features"
hm_getmark comps 1

tk_messageBox -message "Please define the number of element rows adjacent to the features to be removed"
set numberofrows [hm_getint "Define the number of rows:" "Define the number of element rows adjacent to the features to be removed"]

*deletefeatures
*features comps 1 $featureangle 0 0 $featureangle 0
*createmark elems 1 "by comp name" ^feature
for {set x 0} {$x < $numberofrows} {incr x} {hm_appendmark elems 1 "advanced" "by adjacent"}
*maskentitymark elements 1 0
*clearmark elems 1
*deletefeatures

*clearmark nodes 1
#cria um painel de seleção de nós, e coloca na marca 1
tk_messageBox -message "Please select the nodes to generate the strain gage candidate set on"
*createmarkpanel nodes 1 "Select nodes to be listed:"

#extrai uma lista dos nós selecionados
set nodeslist [ hm_getmark nodes 1 ]

tk_messageBox -message "Please provide the size of the increment to generate the candidate set of strain gages around each node"
set angle_spacing [hm_getint "Define the angle increment (within the range 0-180):" "The value should be an integer"]

#cria uma lista com os incrementos de ângulo
if {[info exist angle_list]} then {
unset angle_list
}
```

```
set angle 0
while {$angle < 180} {
	lappend angle_list $angle
	set angle [expr $angle_spacing + $angle]
}

*clearmark nodes 1
foreach node_id $nodeslist {
	*createmark nodes 1 "by id only" $node_id
	*findmark nodes 1 1 1 elems 0 2
	set elems_normals [ hm_getmark elems 2 ]
	*clearmark nodes 1
	set nodenormal_x 0
	set nodenormal_y 0
	set nodenormal_z 0
	foreach elem_id $elems_normals {
		if {[lsearch [list 103 104 106 108] [hm_getentityvalue elems $elem_id config 0]] >= 0} {
			set normal_x [hm_getentityvalue element $elem_id normalx 0]
			set normal_y [hm_getentityvalue element $elem_id normaly 0]
			set normal_z [hm_getentityvalue element $elem_id normaly 0]
			set nodenormal_x [expr {double($normal_x) / double([llength $elems_normals]) + double($nodenormal_x)}]
			set nodenormal_y [expr {double($normal_y) / double([llength $elems_normals]) + double($nodenormal_y)}]
			set nodenormal_z [expr {double($normal_z) / double([llength $elems_normals]) + double($nodenormal_z)}]
		}
	}
	set testanormal [expr abs($nodenormal_x)]
	set orientation($node_id) "1,0,0"
	if {$testanormal > [expr abs($nodenormal_y)]} {
		set testanormal [expr abs($nodenormal_y)]
		set orientation($node_id) "0,1,0"
	}
	if {$testanormal > [expr abs($nodenormal_z)]} {
		set orientation($node_id) "0,0,1"
	}
	unset nodenormal_x
	unset nodenormal_y
	unset nodenormal_z
	unset elems_normals
}

#faz a leitura do diretório de trabalho
tk_messageBox -message "Please select the folder to save the output file"
set dir_name [tk_chooseDirectory]
set output_file "${dir_name}/Gages.asg"

#abre o arquivo selecionado para escrita
set fo [open $output_file "w"]
```

```
#para cada nó da nodelist
puts $fo "<StrainGauges>"
foreach node_id $nodeslist {
	#para cada passo do angulo
	foreach number $angle_list {
		#escreve no arquivo
		puts $fo "<StrainGauge ID='node $node_id - $number deg' Type='Single' Location='$node_id' AngleOffset='$number' Orientation='$orientation($node_id)' ResultsFrom='OneSurface' LocationType='Node' ShellSurface='Top'/>"
	}
}
puts $fo "</StrainGauges>"
close $fo

*unmaskall

*clearmark comps 1
*clearmark nodes 1
*clearmark nodes 2
*clearmark elems 1
*clearmark elems 2


tk_messageBox -message "Gages list saved in:\n${dir_name}/Gages.asg"
```

APÊNDICE 2 – ALGORÍTMO GENÉTICO DE OTIMIZAÇÃO DA CONFIRGURAÇÂO DE EXTENSÔMETROS PARA RECONSTRUÇÂO DE CARGAS

```
# -*- coding: cp1252 -*-
# ------------------------------------------------------------------------------------------------------------#
#  Script - Optimization algorithm to find the set of strain gauges for load reconstruction  #
#                                                                                                             #
#                                Developed by DIEGO HOEPFNER                                                  #
#                                                                                                             #
#              Unauthorized copying or distribution is strictly forbidden                                     #
# ------------------------------------------------------------------------------------------------------------#

def glyphscript(engineState):

    from numpy import dot
    from numpy import vdot
    from numpy import linalg
    from numpy import zeros
    from numpy import diag
    from numpy import matrix
    from numpy import subtract
    from numpy import dtype
    import random
#
    DEBUG = False
#
#  define the input and output pads
#
    tsin = engineState.GetInputTimeSeries(0)  # input from virtual straingage using unit loads
    mdin = tsin.GetMetaData()
    tsout = engineState.GetOutputTimeSeries(0)  # output computed force histories
    mdout = tsout.GetMetaData()
#
#  get properties from the glyph
#
    propSet = engineState.GetPropertySet()
    props = propSet.GetProperties()
    setSize = int(props['SetSize'])
    maxIterations = int(props['maxIterations'])
    method = props['Method']
    populationSize = int(props['populationSize'])
    elitism = int(props['elitism'])
    rankNumber = props['rankNumber']
    mutationRate = props['mutationRate']
    admissibleError = props['admissibleError']
    bestSetPercentage = props['bestSetPercentage']
#
#  get virtual strain data from unit load parameters
```

```
#
    numGauges = tsin.GetChannelCount()
    iChan = 0
    numLoads = tsin.GetPointCount(iChan)
    if numLoads > numGauges:
        message = 'Insufficent number of straingages to reconstruct loads, NumberOfGages >= NumberOfLoads'
        engineState.JournalError(message)
        if abort:
            return message
    if numLoads > setSize:
        message = 'Set size needs to be equal or bigger than number of loads'
        engineState.JournalError(message)
        if abort:
            return message
#
#  create strain matrix
#
    E = zeros((numLoads,numGauges),dtype='d')
    for iGauge in xrange(numGauges):
        for iLoad in xrange(numLoads):
            E[iLoad,iGauge] = tsin.GetValue(iGauge, iLoad)
#
#############################################################################
#
#
#---------------------------------------------------------------------------
#                    Preliminares
#---------------------------------------------------------------------------
#
    if method == 'Genetic':
        if setSize < 3:
            message = 'Insufficent number of straingages to optimize with Genetic Algorithm. Increase setSize(>3) or select
another method'
            engineState.JournalError(message)
            if abort:
                return message
        if (setSize*populationSize) > (numGauges):
            message = 'Insufficent number of straingages to generate the population, reduce the Set Size or increase the
number of candidates'
            engineState.JournalError(message)
            if abort:
                return message
        if (elitism) >= (populationSize):
            message = 'Elitism number should be lower than population size'
            engineState.JournalError(message)
            if abort:
                return message
        if ((rankNumber<0) or (rankNumber>1)):
            message = 'rankNumber should be within the range 0-1'
            engineState.JournalError(message)
            if abort:
```

```
                return message
        BakerRank=list(range(0, populationSize))
        S=list(range(0, populationSize))
        c=0.
        for i in xrange(populationSize):
            BakerRank[i]=(2.-rankNumber+(2.*rankNumber-2.)*((i)/(populationSize-1.)))
            c=c+BakerRank[i]
            S[i]=c
        maximum=0
        minimum=10000
        for iLoad in xrange(numLoads):
            for iGauge in xrange(numGauges):
                if abs(E[iLoad,iGauge])>maximum:
                    maximum=abs(E[iLoad,iGauge])
                elif abs(E[iLoad,iGauge])<minimum:
                    minimum=abs(E[iLoad,iGauge])
            temp=list()
            for iGauge in xrange(numGauges):
                if abs(E[iLoad,iGauge])>((1-bestSetPercentage)*maximum+minimum*bestSetPercentage):
                    temp.append(iGauge)
            exec('bestLoad_' + str(iLoad) + ' = ' + str(list(temp)))
#
#-------------------------------------------------------------------------
#                Geração inicial
#-------------------------------------------------------------------------
#
        sample = random.sample(range(0,int((numGauges))),(setSize*populationSize))
        n = 0
        population = zeros((setSize,populationSize),dtype=int)
        for j in xrange(populationSize):
            for i in xrange(setSize):
                population[i,j] = int(sample[n])
                n=n+1
        selected=zeros((setSize,elitism),dtype=int)
        Dselected=[0]*elitism
        iDet = 0
        Det=list()
        error=1
        message = str(population)
        engineState.JournalOut(message)
        while (iDet < maxIterations) or (error > admissibleError):
#
#-------------------------------------------------------------------------
#                Aptidão
#-------------------------------------------------------------------------
#
            D=[0]*populationSize
            for i in xrange(populationSize):
                A = zeros((setSize,numLoads),dtype='d')
                for iGauge in xrange(setSize):
                    for iLoad in xrange(numLoads):
```

```
                        A[iGauge,iLoad] = E[iLoad,population[iGauge,i]]
                D[i]=(1./(linalg.det(dot(A.transpose(),A))))
#
#---------------------------------------------------------------------
#                    Seleção e Elitismo
#---------------------------------------------------------------------
#
            sortpopulation = list(range(0,populationSize))
            for i in xrange(populationSize):
                j=i
                while ((j > 0) and (D[j-1] > D[j])):
                    u=D[j]
                    D[j]=D[j-1]
                    D[j-1]=u
                    v=sortpopulation[j]
                    sortpopulation[j]=sortpopulation[j-1]
                    sortpopulation[j-1]=v
                    j=j-1
            populationSorted = zeros((setSize,populationSize),dtype=int)
            for j in xrange(populationSize):
                for i in xrange(setSize):
                    populationSorted[i,j] = population[i,sortpopulation[j]]
            if iDet==0:
                for j in xrange(elitism):
                    for i in xrange(setSize):
                        selected[i,j]=populationSorted[i,j]
                    Dselected[j]=D[j]
            if elitism!=0:
                for j in xrange(elitism):
                    if abs(Dselected[j])<abs(D[j]):
                        for i in xrange(setSize):
                            populationSorted[i,j]=selected[i,j]
                        D[j]=Dselected[j]
                for j in xrange(elitism):
                    for i in xrange(setSize):
                        selected[i,j]=populationSorted[i,j]
                    Dselected[j]=D[j]
            Det.append(D[0])
            message = str(D)
            engineState.JournalOut(message)
            if iDet!=0:
                error=abs(1-Det[iDet]/Det[iDet-1])
            ptr=random.random()
            count=0
            i=0
            newpopulation = zeros((setSize,populationSize))
            while (count<populationSize):
                i=i+1
                while (ptr<S[i]):
                    for j in xrange(setSize):
                        newpopulation[j,count] = populationSorted[j,i]
```

```
                        ptr=ptr+1.
                        count=count+1
                population=newpopulation
                message = str(population)
                engineState.JournalOut(message)
#
#-----------------------------------------------------------------------
#                       Cruzamento
#-----------------------------------------------------------------------
#
                parents1=random.sample(range(0,populationSize),populationSize)
                parentsTemp=list(parents1)
                parents2=[0]*populationSize
                for i in xrange(populationSize):
                        parents2[i]=random.choice(parentsTemp)
                        pos=parents2[i]
                        while parents1[i]==parents2[i]:
                                parents2[i]=random.choice(parentsTemp)
                                pos=parents2[i]
                        parentsTemp.remove(pos)
                newpopulation = zeros((setSize,populationSize))
                for j in xrange(0,populationSize,2):
                        up = random.randrange(1,setSize-2)
                        bottom = random.randrange(up,setSize-1)
                        for i in xrange(setSize):
                                if i<up:
                                        newpopulation[i,j]=population[i,parents1[j]]
                                        newpopulation[i,(j+1)]=population[i,parents2[j]]
                                elif (i>=up and i<=bottom):
                                        newpopulation[i,j]=population[i,parents2[j]]
                                        newpopulation[i,(j+1)]=population[i,parents1[j]]
                                else:
                                        newpopulation[i,j]=population[i,parents1[j]]
                                        newpopulation[i,(j+1)]=population[i,parents2[j]]
                population=newpopulation
#
#-----------------------------------------------------------------------
#                       Mutação
#-----------------------------------------------------------------------
#
                for j in xrange(populationSize):
                        for i in xrange(setSize):
                                if (random.random()<(1-mutationRate)):
                                        population[i,j]=random.randrange(numGauges)
#
#-----------------------------------------------------------------------
#                       Reparação
#-----------------------------------------------------------------------
#
                for j in xrange(populationSize):
                        for i in xrange(setSize):
```

```
                        exec('temp' + '=' + 'bestLoad_' + str(1))
                        for k in xrange(setSize):
                            if population[i,j]==population[i,k]:
                                while (population[i,j]==population[i,k]):
                                    population[i,j]=random.choice(temp)
                iDet=iDet+1
#
#-------------------------------------------------------------------------
#
#  set output timeseries parameters
#
            tsout.SetChannelCount(1)
            tsout.SetXTitle('Iteration')
            mdout.CopyMetaDataSet(mdin, -1, -1, 'FEModel')
            tsout.SetPointCount(0,iDet)
            tsout.SetChanNumber(0,1)
            tsout.SetChanTitle(0,'Objective Function Evolution')
            tsout.SetYTitle(0,'Determinant')
            yunits = ''
            tsout.SetSampleRate(0,1)
            tsout.SetBaseTime(0,0)
#
#  output force histories
#
            for iPnt in xrange(iDet):
                tsout.PutValue(0, iPnt, Det[iPnt])
#
#
#
#
            if DEBUG:
                message = 'E - ' + str(E.shape)
                engineState.JournalOut(message)
                message = str(E)
                engineState.JournalOut(message)

                message = 'A - ' + str(A.shape)
                engineState.JournalOut(message)
                message = str(A)
                engineState.JournalOut(message)

                message = 'Determinant - ' + str(Det)
                engineState.JournalOut(message)
#
            position=[0]*setSize
            for iGauge in xrange((setSize)):
                position[iGauge] = populationSorted[iGauge,0]
            message = 'Positions - ' + str(position)
            engineState.JournalOut(message)
#
#########################################################################
```

```
#
    if method == 'Random':
#
#  create first matrix candidate
#
        sample = random.sample(range(0,int((numGauges))),setSize)
        position = sample
        A = zeros((setSize,numLoads),dtype='d')
        for iGauge in xrange(setSize):
            for iLoad in xrange(numLoads):
                A[iGauge,iLoad] = E[iLoad,sample[iGauge]]
        message = str(A)
        engineState.JournalOut(message)
    #
    #  optimize the inverse matrix determinant
    #
        iDet = 0
        Det=[(1./(linalg.det(dot(A.transpose(),A))))]
        D=Det[0]
        Minv = linalg.inv(dot(A.transpose(),A))
        Ytransp = zeros((1,numLoads),dtype='d')
        Ztransp = zeros((1,numLoads),dtype='d')
    #
        while (iDet <= maxIterations):
            contaiguais = 1
            while (contaiguais > 0):
                newVec = random.sample(range(0,int((numGauges))),1)
                contaiguais = 0
                for iGauge in xrange(setSize):
                    if position[iGauge] == newVec:
                        contaiguais = 1
            for iLoad in xrange(numLoads):
                Ytransp[0,iLoad] = E[iLoad,newVec]
            Y = Ytransp.transpose()
            MinvDotY = dot(Minv,Y)
            YtranspDotMinvDotY = dot(Ytransp,MinvDotY)
            DetAug = 1./((1./D)*(1.+YtranspDotMinvDotY))
            MinvAug = subtract(Minv,(1./(1.+YtranspDotMinvDotY))*dot(MinvDotY,MinvDotY.transpose()))
            for iLoad in xrange(numLoads):
                Ztransp[0,iLoad] = A[0,iLoad]
            Z = Ztransp.transpose()
            MinvAugDotZ = dot(MinvAug,Z)
            ZtranspDotMinvAugDotZ = dot(Ztransp,MinvAugDotZ)
            DetRed = 1./((1./DetAug)*(1.-ZtranspDotMinvAugDotZ))
            MinvRed= subtract(MinvAug,(1./(1.+ZtranspDotMinvAugDotZ))*dot(MinvAugDotZ,MinvAugDotZ.transpose()))
            if abs(DetRed) < abs(D):
                Det.append(DetRed)
                D=DetRed
                Minv = MinvRed
                for iLoad in xrange(numLoads):
                    for iGauge in xrange((setSize-1)):
```

```
                        A[iGauge,iLoad] = A[(iGauge+1),iLoad]
                    A[(setSize-1),iLoad] = Ytransp[0,iLoad]
                for iGauge in xrange((setSize-1)):
                    position[iGauge] = position[(iGauge+1)]
                position[setSize-1]=newVec
            else:
                Det.append(D)
            iDet = iDet+1
#
##            message = 'A - ' + str(A.shape)
##                engineState.JournalOut(message)
##                message = str(A)
##                engineState.JournalOut(message)
##                message = 'Determinant - ' + str(D)
##                engineState.JournalOut(message)
##                message = 'Positions - ' + str(position)
##            engineState.JournalOut(message)
#
#  set output timeseries parameters
#
    tsout.SetChannelCount(1)
    tsout.SetXTitle('Iteration')
    mdout.CopyMetaDataSet(mdin, -1, -1, 'FEModel')
    tsout.SetPointCount(0,iDet)
    tsout.SetChanNumber(0,1)
    tsout.SetChanTitle(0,'Objective Function Evolution')
    tsout.SetYTitle(0,'Determinant')
    yunits = ''
    tsout.SetSampleRate(0,1)
    tsout.SetBaseTime(0,0)
#
#  output force histories
#
    for iPnt in xrange(iDet):
        tsout.PutValue(0, iPnt, Det[iPnt])
#
#
#
#
    if DEBUG:
        message = 'E - ' + str(E.shape)
        engineState.JournalOut(message)
        message = str(E)
        engineState.JournalOut(message)

        message = 'A - ' + str(A.shape)
        engineState.JournalOut(message)
        message = str(A)
        engineState.JournalOut(message)

        message = 'Determinant - ' + str(Det)
```

```
                engineState.JournalOut(message)
        #
                for iGauge in xrange((setSize)):
                        position[iGauge] = position[iGauge]+1
                message = 'Positions - ' + str(position)
                engineState.JournalOut(message)
#
#####################################################################
#
        if method == 'Fedorov':
#
#  create first matrix candidate
#
                sample = random.sample(range(0,int((numGauges))),setSize)
                position = sample
                A = zeros((setSize,numLoads),dtype='d')
                for iGauge in xrange(setSize):
                        for iLoad in xrange(numLoads):
                                A[iGauge,iLoad] = E[iLoad,sample[iGauge]]
                message = str(A)
                engineState.JournalOut(message)

        #
        #  optimize the inverse matrix determinant
        #
                iDet = 0
                Det=[float(1./(linalg.det(dot(A.transpose(),A))))]
                D=float(Det[0])
                Minv = linalg.inv(dot(A.transpose(),A))
                Ytransp = zeros((1,numLoads),dtype='d')
                Ztransp = zeros((1,numLoads),dtype='d')
                engineState.JournalOut(message)
                message = 'Determinant - ' + str(D)
        #
                for iGauge in xrange(setSize):
                        newVec = 0
                        while (newVec < numGauges):
                                contaiguais=1
                                while (contaiguais>0):
                                        contaiguais=0
                                        for i in xrange(setSize):
                                                if position[i] == newVec:
                                                        newVec = newVec+1
                                                        contaiguais = contaiguais+1
                                if newVec < numGauges:
                                        for iLoad in xrange(numLoads):
                                                Ytransp[0,iLoad] = E[iLoad,newVec]
                                        Y = Ytransp.transpose()
                                        MinvDotY = dot(Minv,Y)
                                        YtranspDotMinvDotY = float(dot(Ytransp,MinvDotY))
                                        DetAug = float(1./((1./D)*(1.+YtranspDotMinvDotY)))
```

```
                        MinvAug = subtract(Minv,(1./(1.+YtranspDotMinvDotY))*dot(MinvDotY,MinvDotY.transpose()))
                        for iLoad in xrange(numLoads):
                            Ztransp[0,iLoad] = A[iGauge,iLoad]
                        Z = Ztransp.transpose()
                        MinvAugDotZ = dot(MinvAug,Z)
                        ZtranspDotMinvAugDotZ = float(dot(Ztransp,MinvAugDotZ))
                        DetRed = float(1./((1./DetAug)*(1.-ZtranspDotMinvAugDotZ)))
                        MinvRed=
subtract(MinvAug,(1./(1.+ZtranspDotMinvAugDotZ))*dot(MinvAugDotZ,MinvAugDotZ.transpose()))
                        if abs(DetRed) < abs(D):
                            Det.append(DetRed)
                            D=DetRed
                            Minv = MinvRed
                            for iLoad in xrange(numLoads):
                                A[iGauge,iLoad] = Ytransp[0,iLoad]
                            position[iGauge]=newVec
                        else:
                            Det.append(D)
                        iDet = iDet+1
                        newVec = newVec+1
    #
##                          message = 'newVec - ' + str(newVec)
##                          engineState.JournalOut(message)
##                          message = 'Y - ' + str(Y)
##                          engineState.JournalOut(message)
##                          message = 'DetAug - ' + str(DetAug)
##                          engineState.JournalOut(message)
##                          message = 'Z - ' + str(Z)
##                          engineState.JournalOut(message)
##                          message = 'DetRed - ' + str(DetRed)
##                          engineState.JournalOut(message)
##                          message = 'A - ' + str(A)
##                          engineState.JournalOut(message)
##                          message = 'Determinant - ' + str(D)
##                          engineState.JournalOut(message)
##                          message = 'Positions - ' + str(position)
##                          engineState.JournalOut(message)
    #
    #  set output timeseries parameters
    #
        tsout.SetChannelCount(1)
        tsout.SetXTitle('Iteration')
        mdout.CopyMetaDataSet(mdin, -1, -1, 'FEModel')
        tsout.SetPointCount(0,iDet)
        tsout.SetChanNumber(0,1)
        tsout.SetChanTitle(0,'Objective Function Evolution')
        tsout.SetYTitle(0,'Determinant')
        yunits = ''
        tsout.SetSampleRate(0,1)
        tsout.SetBaseTime(0,0)
    #
```

```
#  output force histories
#
      for iPnt in xrange(iDet):
            tsout.PutValue(0, iPnt, Det[iPnt])
#
#
#
#
      if DEBUG:
            message = 'E - ' + str(E.shape)
            engineState.JournalOut(message)
            message = str(E)
            engineState.JournalOut(message)

            message = 'A - ' + str(A.shape)
            engineState.JournalOut(message)
            message = str(A)
            engineState.JournalOut(message)

            message = 'Determinant - ' + str(Det)
            engineState.JournalOut(message)
#
      for iGauge in xrange((setSize)):
            position[iGauge] = position[iGauge]+1
      message = 'Positions - ' + str(position)
      engineState.JournalOut(message)
#
return ''
```