Daniel Paciência Ferreira

Licenciado em Ciências da Engenharia Mecânica

Desenvolvimento de uma *ToolBox* de Otimização para o PROAES – Programa de Análise Estrutural

Dissertação para a obtenção do Grau de Mestre em Engenharia Mecânica

Orientador: Prof. Doutor João Mário Burguete Botelho Cardoso
Prof. Auxiliar da Faculdade de Ciências e Tecnologias da Universidade Nova de Lisboa

Júri

Presidente: Doutor Pedro Samuel Gonçalves Coelho
Vogais: Doutor Tiago Alexandre Narciso da Silva
Doutor João Mário Burguete Botelho Cardoso

Setembro de 2019

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Desenvolvimento de uma *ToolBox* de otimização para o PROAES – Programa de Análise Estrutural

Copyright © Daniel Paciência Ferreira, Faculdade de Ciências e Tecnologia, Universidade NOVA de Lisboa.

A Faculdade de Ciências e Tecnologia e a Universidade NOVA de Lisboa têm o direito, perpétuo e sem limites geográficos, de arquivar e publicar esta dissertação através de exemplares impressos reproduzidos em papel ou de forma digital, ou por qualquer outro meio conhecido ou que venha a ser inventado, e de a divulgar através de repositórios científicos e de admitir a sua cópia e distribuição com objetivos educacionais ou de investigação, não comerciais, desde que seja dado crédito ao autor e editor.

Agradecimentos

Primeiramente gostava de agradecer ao meu orientador, o Professor Doutor João Cardoso, por me ter confiado este tema e por todo o empenho e dedicação que tornaram possível a realização desta dissertação. Gostava também de agradecer por todos os ensinamentos que me transmitiu ao longo deste percurso académico e por toda a motivação e apoio que me deu durante este trabalho.

Queria agradecer também, ao meu Professor Carlos Ribeiro que foi um grande pilar, que me transmitiu muitos conhecimentos e influenciou, positivamente, a minha vida.

Agradeço também ao Professor Krister Svanberg por ter disponibilizado o código do algoritmo MMA e aos membros do projeto YARPIZ por disponibilizarem o código dos algoritmos GA e PSO.

Agradeço a todos os meus Familiares, por todo o apoio incondicional que me deram ao longo do meu percurso académico e por acreditarem sempre nas minhas capacidades. Em especial, queria agradecer à minha Mãe, Rosa Ferreira e ao meu Pai Augusto Ferreira que contribuíram muito na minha educação e no meu percurso académico.

Por último, gostava de agradecer aos meus Amigos, Bernardo Simões, Bernardo Abrantes, João Horta e Gonçalo Arribança, por terem feito parte desta caminhada e por todos os agradáveis momentos que passámos juntos.

Resumo

A presente dissertação tem como objetivo a criação de uma *Toolbox* de otimização composta por algoritmos *Open Source* especializados na resolução de problemas de otimização. A *Toolbox* irá servir para funcionar em colaboração com o programa de elementos finitos, PROAES, para em conjunto resolverem vários tipos de problemas de otimização encontrados na engenharia.

Na realização desta dissertação são abordadas as duas grandes famílias de algoritmos de otimização: algoritmos baseados em gradientes e meta-heurísticas e são descritos os algoritmos que compõem a *Toolbox*: SQP, MMA, GA REAL, GA BINÁRIO e PSO. Neste trabalho é também explicado em detalhe o funcionamento da *Toolbox* e é abordado como deve ser realizada a construção de ficheiros que permitem a comunicação entre os algoritmos da *Toolbox* e o programa de análise, o PROAES.

Para testar o bom funcionamento da implementação, aplica-se a *Toolbox* na resolução de três problemas de otimização: dois problemas de otimização dimensional e um problema de otimização topológica. Os resultados demonstram o bom funcionamento da *Toolbox*. Demonstram também que a *Toolbox* pode ser utilizada em conjunto com outros programas de análise em problemas de otimização.

Palavras Chave:*Toolbox*,Algoritmos de otimização, SQP, MMA, GA REAL, GA BINÁRIO, PSO

Abstract

The present dissertation aims at creating an optimization toolbox composed by Open Source algorithms specialized in solving optimization problems. The Toolbox will work in collaboration with the PROAES finite element program to solve several types of optimization problems encountered in engineering.

In this dissertation, the two great families of optimization algorithms are addressed: gradient-based algorithms and metaheuristics, and the algorithms that are included in the Toolbox are described: SQP, MMA, REAL GA, BINARY GA and PSO. In this thesis it is explained in detail how the Toolbox works and also how to build files that establish the communication between the Toolbox algorithms and the analysis program, or (PROAES).

To test how well the implementation works, it is applied in the resolution of three optimization problems. Two dimensional optimization problems and one topological optimization problem. The results show the good performance of the Toolbox for solving optimization issues. Also demonstrate that the toolbox can be used together with other analysis programs on optimization problems.

Keyword: Toolbox, Optimization algorithms, SQP, MMA, REAL GA, BINARY GA, PSO

Índice

Índice XI
Índice de Figuras XIII
Índice de Tabelas XV
Siglas e Símbolos XVII
1. Introdução 1
1.1. Motivação 1
1.2. Objetivo 2
1.3. Estrutura da dissertação 2
2. Otimização Estrutural 5
2.1. Conceitos de Otimização Estrutural 5
2.1.1. Formulação matemática geral de um problema de otimização 6
2.1.2. Otimização dimensional 7
2.1.3. Otimização de forma 7
2.1.4. Otimização topológica 7
2.2. Revisão de Alguns Conceitos de Cálculo 9
2.2.1. Sensibilidades 9
2.2.2. Vetor Gradiente 10
2.2.3. Matriz Hessiana 11
2.2.4. Expansão em serie de Taylor 12
2.2.5. Condições de Optimalidade 13
2.3. Métodos de Otimização Baseados no Gradiente 14
2.3.1. Programação Quadrática Sequencial (SQP) 17
2.3.2. Método das Assintotas Moveis (MMA) 18
2.4. Métodos de otimização não baseados em gradientes 19
2.4.1. Algoritmo Genético (AG) 20
2.4.2. Otimização por Enxame de Partículas (PSO) 22
3. PROAES 25
3.1. Análise de sensibilidades pelo método contínuo 27
3.2. Otimização Estrutural 28
4. TOOLBOX de Otimização 29
4.1. O que é a TOOLBOX de Otimização? 29
4.2. Problema de Otimização – *Treliça 2 Barras* 29
4.3. Programação quadrática sequencial (SQP) 31
4.3.1. Formulação do Problema 31
4.3.2. Função objetivo 32
4.3.3. Constrangimento 33
4.3.4. Gradiente da Função objetivo 34
4.3.5. Gradiente do constrangimento 34

4.3.6. Execução da Otimização 36
4.4. Método das Assíntotas Móveis (MMA) 38
4.4.1. Formulação do Problema 39
4.4.2. Função objetivo e Gradiente 39
4.4.3. Constrangimento e Gradiente 40
4.4.4. Execução da Otimização 41
4.5. Algoritmo Genético Real 43
4.5.1. Formulação do Problema 43
4.5.2. Função objetivo 44
4.5.3. Execução da Otimização 46
4.6. Algoritmo Genético Binário 47
4.6.1. Formulação do Problema 48
4.6.2. Função objetivo 48
4.6.3. Execução da Otimização 50
4.7. Otimização por Enxame de Partículas (PSO) 50
4.7.1. Execução da Otimização 51
4.8. Resumo dos Ficheiros da *Toolbox* de Otimização 53
5. Exemplos de Verificação 55
5.1. Exemplo 1 – Treliça 11 Barras 55
5.1.1. Resultados obtidos do Algoritmo SQP 57
5.1.2. Resultados obtidos do Algoritmo MMA 59
5.1.3. Resultados obtidos do Algoritmo Genético Real 61
5.1.4. Resultados Obtidos do Algoritmo Genético Binário 63
5.1.5. Resultados Obtidos do Algoritmo PSO 65
5.1. Exemplo 2 – Treliça 38 Barras 67
5.2.1. Resultados Obtidos do Algoritmo SQP 69
5.2.2. Resultados Obtidos do Algoritmo MMA 71
5.2.3. Resultados Obtidos do Algoritmo Genético Real 73
5.2.4. Resultados obtidos do Algoritmo PSO 75
5.3. Exemplo 3 – Europa TIR 77
5.3.1. Resultados obtidos pelos Algoritmos da *Toolbox* de Otimização 80
6. Conclusões e desenvolvimentos futuros 85
Referências Bibibliográficas 87
Anexo A 91
Ficheiros relativos ao Exemplo 1 – Treliça 11 Barras 91
Anexo B 109
Ficheiros relativos ao Exemplo 2 – Treliça 38 Barras 109
Anexo C 124
Ficheiros relativos ao Exemplo 3 – Europatir 124

Índice de Figuras

Figura 2.1 - Exemplo de otimização dimensional 7
Figura 2.2 - Exemplo de otimização de forma.....7
Figura 2.3 - Exemplo de otimização topológica. 8
Figura 2.4 - Diferenças Finitas Progressivas, extraído de [8] 10
Figura 2.5 - Vetor gradiente para $f(x1, x2, x3)$ no ponto $\boldsymbol{x}*$, adaptado de [11]. 11
Figura 2.6 - Diagrama que representa as etapas iterativas de um método de otimização, extraído de [11]. 15
Figura 2.7 - Etapas de um algoritmo de otimização restrita, adaptado de [8] 16
Figura 2.8 - Exemplo retirado da YARPIZ, profundidade de um lago. 22
Figura 3.1 - Estrutura do programa PROAES, extraído de [22]. 26
Figura 3.2 - Interligação entre o programa PROAES e o OPTIMTOOL, adaptado de [3] 28
Figura 3.3 - Interligação entre o programa PROAES e o TOOLBOX. 28
Figura 4.1 - Treliça 2 Barras 30
Figura 4.2 - Diagrama do Processo de Otimização SQP 37
Figura 4.3 - Ficheiro de inicialização do problema SQP 38
Figura 4.4 - Diagrama do Processo de Otimização MMA 42
Figura 4.5 - Parte superior do ficheiro de inicialização do problema 43
Figura 4.6 - Diagrama do Processo de Otimização 47
Figura 4.7 - Ficheiro de inicialização do problema 47
Figura 4.8 - Ficheiro de inicialização do problema 50
Figura 4.9 - Diagrama do Processo de Otimização PSO 51
Figura 4.10 - Ficheiro de inicialização do problema 52
Figura 4.11 - Ficheiros que compõem a *toolbox* 53
Figura 5.1 - Treliça 11 barras do exemplo 1 [2] 56
Figura 5.2 - Treliça 11 barras otimizada pelo SQP MATLAB 58
Figura 5.3 - Treliça 11 barras otimizada pelo SQP OCTAVE 58
Figura 5.4 - Treliça 11 barras otimizada pelo MMA 60
Figura 5.5 - Gráfico de convergência do algoritmo MMA OCTAVE 60
Figura 5.6 - Treliça 11 barras otimizada pelo GA OCTAVE 62
Figura 5.7 - Gráfico de convergência do algoritmo GA OCTAVE 62
Figura 5.8 - Treliça 11 barras otimizada pelo GA OCTAVE 64
Figura 5.9 - Gráfico de convergência do algoritmo GA OCTAVE 64
Figura 5.10 - Treliça 11 barras otimizada pelo OCTAVE 66

Figura 5.11 - Gráfico de convergência do algoritmo 66
Figura 5.12 - Estrutura do 2º problema – Treliça 38 barras 67
Figura 5.13 - Topologia ótima MATLAB 70
Figura 5.14 - Topologia ótima OCTAVE 70
Figura 5.15 - Topologia ótima MMA OCTAVE 72
Figura 5.16 - Gráfico de convergência do algoritmo MMA OCTAVE 72
Figura 5.17 - Topologia ótima GA OCTAVE 74
Figura 5.18 - Gráfico de convergência do algoritmo GA REAL OCTAVE 74
Figura 5.19 - Topologia ótima PSO OCTAVE 76
Figura 5.20 - Gráfico de convergência do algoritmo PSO OCTAVE 76
Figura 5.21 - Semirreboque 77
Figura 5.22 - Elementos finitos do chassis do Camião, adaptado de [31] 78
Figura 5.23 - Detalhes da figura do chassis do semirreboque, adaptado de [31] 78
Figura 5.24 - Detalhes da figura do chassis do semirreboque, adaptado de [31] 79
Figura 5.25 - Detalhes da figura do chassis do semirreboque, adaptado de [31] 79
Figura 5.26 - Gráfico de convergência do algoritmo MMA OCTAVE. 82
Figura 5.27 - Gráfico de convergência do algoritmo PSO OCTAVE. 82
Figura 5.28 - Gráfico de convergência do algoritmo GA OCTAVE. 83

Índice de Tabelas

Tabela 2.1 - Tipos de variáveis de projeto dependendo da formulação do problema de otimização estrutural, extraído de [2]. 9
Tabela 2.2 - Algoritmo geral de otimização, adaptado de [2]. 16
Tabela 2.3 - Comparação de terminologias, adaptado de [2]. 21
Tabela 3.1 - Elementos finitos utilizados no programa PROAES. 25
Tabela 3.2 - Performances e variáveis de projeto disponíveis no programa PROAES, adaptado de [6] 27
Tabela 4.1 - Valores iniciais das variáveis. 29
Tabela 5.1 - Valores iniciais das variáveis e do Volume 55
Tabela 5.2 - Resultados da otimização com o algoritmo SQP 57
Tabela 5.3 - Resultados da otimização com o algoritmo MMA 59
Tabela 5.4 - Resultados da otimização com o algoritmo GA OCTAVE 61
Tabela 5.5 - Valores das áreas disponíveis para as variáveis do problema 63
Tabela 5.6 – Resultados da otimização com o algoritmo GA BINÁRIO 63
Tabela 5.7 - Resultados da otimização com o algoritmo PSO (OCTAVE) 65
Tabela 5.8 - Valores iniciais das variáveis e do Volume 67
Tabela 5.9 - Resultados da otimização com o algoritmo SQP 69
Tabela 5.10 - Resultados da otimização com o algoritmo MMA. 71
Tabela 5.11 - Resultados da otimização com o algoritmo GA. 73
Tabela 5.12 - Resultados da otimização com o algoritmo PSO. 75
Tabela 5.13 - Variáveis de projeto consideradas na otimização da estrutura do chassi do semirreboque, adaptado de [31] 80
Tabela 5.15 - Comparação das Variáveis de projeto otimizadas obtidas pelos algoritmos da *toolbox* de otimização e [31] 81

Siglas e Símbolos

$f(x)$	Função objetivo
$g_j(x), c(x)$	Função Constrangimento
$h_k(x)$	Constrangimento de igualdade
$x\ X_i$	Variável de projeto
x_i^l	Limite inferior da variável de projeto
x_i^u	Limite superior da variável de projeto
Ψ	Performance estrutural
$\frac{\partial \Psi}{\partial X_i}$	Derivada de Ψ em relação a X_i
SQP	Programação quadrática sequencial
MMA	Método das assintotas moveis
Δ	Pequena perturbação
x^*	Ponto ótimo da função objetivo
$\frac{\partial f(x^*)}{\partial x_1}$	Derivada parcial da função em ordem à variável
$\nabla f(x^*)$	Vetor Gradiente
H =$\nabla^2 f(x^*)$	Matriz Hessiana
$d = x - x^*$	Incremento ou perturbação na variável de projeto
v^*, u^*	Multiplicador de Lagrange
s_i, y, z	Variável artificial
k	Número da iteração
$d^{(k)}$	Direção de procura no espaço de projeto
α_k	Passo ótimo no espaço de projeto, escalar positivo
NLP	Problema de otimização não linear
IN_0	Números naturais incluindo zero
QP	Programação Quadrática
x_j^{min}	Limite inferior da variável de projeto
x_j^{max}	Limite superior da variável de projeto
$f_i^{\sim(k)}(x)$	Função convexa
$u_j^{(k)}\ l_j^k$	Assimptota móvel
GA	Algoritmo Genético
PSO	Otimização por enxame de partículas
CP	Constante de penalização
VP	Valor da penalização
SQP *	Sensibilidades calculadas pelo PROAES
SQP’	Sensibilidades calculadas por diferenças finitas
$x^{(0)}$	Ponto de partida das variáveis
Δx^0	Variação das variáveis

1. Introdução

1.1. Motivação

A grande competitividade que existe atualmente nas várias indústrias levou a que os engenheiros se empenhassem cada vez mais na busca por métodos rigorosos que auxiliassem as decisões, tendo sido desenvolvidos métodos de otimização estrutural com vista a projetar e produzir de forma económica e eficiente, económica no sentido de poupança de material e eficiente a nível estrutural Otimizar significa obter o melhor rendimento possível, ou seja, empregar técnicas de modo a selecionar as melhores alternativas para se atingir os objetivos determinados. Já na antiguidade, quando um engenheiro realizava um projeto estrutural, colocava em prática o processo de otimização, baseado sobretudo em cálculos efetuados à mão. Por exemplo, o engenheiro realizava os cálculos para saber qual a espessura que uma viga deveria ter, e depois repetia o cálculo várias vezes, experimentando diferentes perfis de viga até que encontrava uma viga que era ao mesmo tempo suficientemente leve, mas também suficientemente resistente.

Em 1960 Schmit [1][2] introduziu o conceito de otimização estrutural moderna o que provocou um grande impulso no processo de otimização. Por um lado, introduziu a ideia de combinar a análise estrutural por elementos finitos com os métodos numéricos de otimização e, por outro lado, demonstrou a viabilidade deste processo na resolução de problemas reais. No entanto, surgia agora a necessidade de encontrar formas que descrevessem os modelos que se pretendiam otimizar de uma maneira inequívoca.

O ramo da matemática que se debruça sobre a aplicação de métodos científicos a problemas de otimização, de forma a criar modelos matemáticos, apoiados em dados que permitam perceber os problemas e avaliar qual a melhor solução, é a investigação operacional. Com o passar do tempo foram-se desenvolvendo diferentes métodos de programação, para resolver distintos problemas de otimização.

Um algoritmo pode ser definido como "uma descrição, passo-a-passo, de uma metodologia que conduz à resolução de um problema ou à execução de uma tarefa [3]**.** Com o aparecimento do computador na década de 50, uma ferramenta poderosa, os métodos de programação começam a ser descritos em diferentes algoritmos especializados em otimização estrutural, que podem ser divididos em duas grandes famílias: os algoritmos de otimização baseados no gradiente e as Meta-Heurísticas, algoritmos que em grande parte se baseavam em propriedades da natureza.

Aproveitando o conceito introduzido por Schmit foram aplicados grandes esforços no desenvolvimento de programas que ligavam algoritmos de otimização a programas de elementos

finitos, pois podem ser aplicados para resolver muitos problemas de otimização estrutural encontrados na engenharia.

Com este avanço o engenheiro deixou de efetuar apenas cálculos analíticos e passou a fazer uso de programas específicos de computador que realizam o processo de otimização de forma mais rápida e rigorosa.

No departamento de Engenharia Mecânica e Industrial da Faculdade de Ciências e Tecnologias da Universidade Nova de Lisboa existe um programa, PROAES, que foi desenvolvido com o objetivo de facilitar a aprendizagem de elementos finitos, otimização estrutural e para suportar a investigação nestas áreas. O desenvolvimento do programa contou com a colaboração de vários alunos que lhe foram acrescentando funcionalidades. O programa teve início com a dissertação de Tiago Batista [4], depois Marco Correia [5] e por ultimo João Marmeleiro [6].

1.2. Objetivo

De início o programa PROAES foi desenvolvido com o objetivo de realizar otimização estrutural, contudo existindo uma *toolbox* de otimização no MATLAB não foi sentida a necessidade de desenvolver algoritmos de otimização concentrando-se os esforços no desenvolvimento do programa de análise, no cálculo das sensibilidades e no cálculo da fiabilidade estrutural e suas derivadas.

Mais recentemente sentiu-se a necessidade de utilizar um programa de domínio público, OCTAVE, verificando-se que este último não apresenta uma *toolbox* de otimização equivalente à do MATLAB. A construção da *toolbox* agrupando algoritmos *open source* (com o código fonte disponível gratuitamente), é o objetivo desta dissertação.

Numa primeira fase pretende-se agrupar um conjunto de algoritmos de otimização de forma a criar um módulo dedicado à resolução de problemas de otimização estrutural.

Numa segunda fase pretende-se estabelecer uma interface comum para os vários algoritmos de forma a permitir ao utilizador alterar o algoritmo que deseja utilizar com o mínimo de modificações nos ficheiros necessários para a solução do problema.

1.3. Estrutura da dissertação

A presente dissertação encontra-se dividida em seis capítulos. No presente capítulo (capítulo um), é feita uma introdução ao tema da dissertação em que é descrita a motivação e os objetivos que levaram à sua realização.

No capítulo dois, é realizada uma revisão da literatura sobre as principais temáticas desta dissertação. São abordados os principais conceitos da otimização estrutural, são revistos alguns

conceitos de cálculo e por último são introduzidos os métodos de otimização baseados em gradientes e não baseados em gradientes.

No capítulo três, é abordado o programa de análise estrutural, o PROAES, na sua versão 5.0, desenvolvido por João Marmeleiro [6]. É também comparada a utilização da *Toolbox* do MATLAB versus a *Toolbox* desenvolvida em OCTAVE.

No capítulo quatro é explicado em detalhe a forma como o utilizador deve interagir com cada um dos algoritmos: SQP, MMA, GA REAL, GA BINÁRIO, PSO, e é apresentado um exemplo simples de otimização.

De forma a verificar o bom funcionamento da relação entre os algoritmos e o PROAES são apresentados, no capítulo cinco, os resultados obtidos em três problemas de otimização distintos.

Por último no capítulo seis é apresentada a conclusão e desenvolvimento de trabalhos futuros, onde é feita uma retrospetiva de todo o trabalho realizado no âmbito desta dissertação.

2. Otimização Estrutural

Os primeiros grandes esforços para aplicar técnicas de programação ao projeto de sistemas estruturais representados por modelos de elementos finitos foram apresentados em 1964 por Gallagher e por Gellatly em 1966 [7] [8] [9]. Estes artigos apresentam o desenvolvimento, aplicação e avaliação de uma abordagem automatizada para o projeto estrutural de minimização de massa. É descrito ainda um programa de computador para abordar o problema.

Um segundo grande esforço para aplicar técnicas de programação matemática ao projeto de sistemas estruturais foi realizado por Karnes e Tocher em 1968 e relatado num relatório de Schmit e Hirokazu de 1976 [7], abordando o tema de 'Otimização e problemas não lineares', é descrito também um programa de computador para minimização do peso de estruturas planas dentro dos limites de espessura e tensões permitidas.

Contudo, para alguns investigadores, a aplicação de programação matemática em problemas de otimização estrutural parecia uma barreira insuperável [7], a indústria aeronáutica era o grande motor da altura, o que motivou a aplicar grandes esforços na implementação da redefinição dos procedimentos para realizar a otimização. Em 1971 [10] Dwyer desenvolveu um novo algoritmo de otimização estrutural, "Um procedimento automatizado para a otimização de estruturas aeroespaciais". O algoritmo é incorporado a um programa de elementos finitos.

Mais tarde começaram a ser estudadas analises estruturais aproximadas usando a expansão em série de Taylor [7]. O uso da aproximação da série de Taylor como primeira estimativa num processo iterativo pode levar a melhorias significativas na eficiência.

Desde estes esforços pioneiros, foi despendido um grande investimento em pesquisas destinadas a tornar a relação entre programação matemática e o método dos elementos finitos eficiente. A otimização estrutural está hoje disponível numa grande variedade de programas comerciais.

2.1. Conceitos de Otimização Estrutural

O objetivo da otimização estrutural é criar projetos de estruturas que suportem os carregamentos aplicados da forma mais eficiente possível. A primeira ideia a vir à mente pode ser criar a estrutura o mais rígida possível, ou seja, minimizar os seus deslocamentos. Outra ideia poderá ser criar a estrutura o mais leve possível, ou seja, minimizar o seu peso. Estas propriedades são contraditórias, isto é, uma estrutura mais rígida tenderá a ser mais pesada, uma estrutura mais leve tenderá a ser mais flexível. A escolha adequada da estrutura que apresenta a melhor proporção entre elas pode ser obtida através da otimização estrutural. Contudo as maximizações ou minimizações não podem ser efetuadas sem restrições. Se o problema não tiver restrições a sua solução poderá não ter viabilidade estrutural.

Num problema típico de otimização estrutural as propriedades que normalmente aparecem associadas como restrições são: os deslocamentos, as tensões, o volume ou a energia de deformação. É de notar que estas propriedades também podem estar associadas à função objetivo, isto é, podem-se minimizar os deslocamentos com constrangimento de volume ou minimizar o volume com constrangimentos de deslocamento.

2.1.1. Formulação matemática geral de um problema de otimização

De acordo com [2] a formulação de um problema de otimização ou projeto ótimo envolve três passos:

1. Identificação das **variáveis de projeto,** que podem ser dimensões ou parâmetros que definem a forma ou a topologia. As variáveis de projeto devem ser independentes umas das outras tanto quanto possível.
2. Identificação de uma função objetivo (minimizar a massa, maximizar a rigidez) ou função custo e expressá-la como função das variáveis de projeto. A **função objetivo** é o critério necessário para analisar se um determinado projeto é melhor do que outro.
3. Identificação de todos os **constrangimentos de projeto (**tensão, volume, deslocamento**)** e a sua transcrição em expressões matemáticas. Estes constrangimentos podem incluir também, limitações de recursos, resposta do sistema, critérios de cedência de material, dimensões geométricas dos membros do sistema, etc. Se um projeto satisfaz todos os constrangimentos diz-se que temos um projeto admissível.

Um problema de otimização estrutural pode ser formulado matematicamente da seguinte forma standard:

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) & & \\ Sujeito\ a \quad & g_j(\boldsymbol{x}) \leq 0 & ;\, j = 1, \dots, m & \\ & h_k(\boldsymbol{x}) = 0 & ;\, k = 1, \dots, p & \\ & x_i^l \leq x_i \leq x_i^u & ;\, i = 1, \dots, n & \end{aligned} \tag{2.1}$$

Onde $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$ é o vetor das n variáveis de projeto, $f(\boldsymbol{x})$ é a função objetivo, e as funções $g_j(\boldsymbol{x})$ são os m constrangimentos de desigualdade, $h_k(\boldsymbol{x})$ são os p constrangimentos de igualdade, x_i^l e x_i^u são os limites inferior e superior de variação das variáveis x_i, respetivamente.

O projeto ótimo de uma estrutura pode ser formulado em termos da otimização dimensional, de forma e de topologia, estes conceitos irão ser explicados a nas secções 2.1.2 2.1.4.

2.1.2. Otimização dimensional

Neste tipo de problemas as variáveis de projeto são as dimensões das secções transversais dos elementos da estrutura, podendo ser contínuas ou discretas.

• Variáveis contínuas: variam continuamente entre um valor superior e um valor inferior, como por exemplo o raio de uma barra de secção circular.

• Variáveis discretas: normalmente as variáveis só podem ter valores que façam parte de uma lista de dimensões pré-estabelecidas, como por exemplo uma lista de perfis normalizados de viga HEB.

Um exemplo de otimização dimensional é apresentado na Figura 2.1

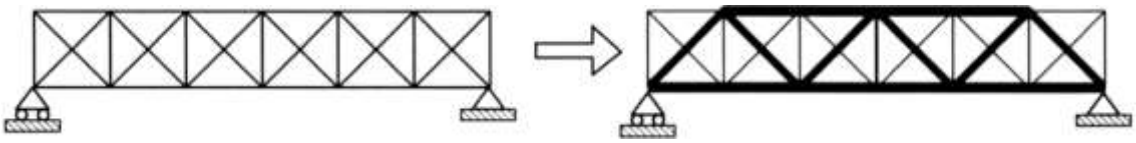

Figura 2.1 - Exemplo de otimização dimensional

2.1.3. Otimização de forma

A otimização de forma é usada quando a topologia do componente já foi definida pois irá manter-se ao longo de todo o processo de otimização. A superfície ou fronteira da peça a otimizar é então definida por um conjunto de funções matemáticas ou por um conjunto de pontos. Essas funções são definidas por alguns parâmetros e a otimização consiste em escolher a solução ótima para esses parâmetros que irão determinar a forma ideal da fronteira.

Um exemplo de otimização de forma é apresentado na Figura 2.2

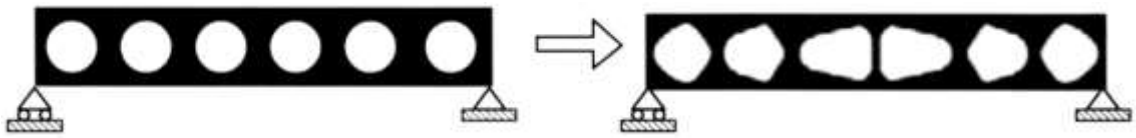

Figura 2.2 - Exemplo de otimização de forma

2.1.4. Otimização topológica

Esta é a forma mais geral de otimização estrutural, pois pode dar aos engenheiros novas ideias sobre a topologia ideal, não sendo necessário um *design* pré-estabelecido. O objetivo da

otimização topológica é descobrir onde deve ser colocado o material e onde devem ficar os vazios de material.

Várias abordagens diferentes de otimização topológica foram propostas sendo de destacar o *"Ground Sturcture Approach"*[2], específico para estruturas reticuladas e o "Soft-Kill Method*"*[2] que é uma abordagem com base na densidade.

Para problemas que envolvem estruturas reticuladas, como treliças, a otimização topológica procura otimizar o número de nós, o número de barras, bem como as condições de apoio da treliça. O tipo de variáveis a aplicar são variáveis booleanas (0 ou 1). O valor 1 representa o estado de existência e o valor 0 representa o estado de inexistência. Por exemplo se for considerado como caso de estudo uma treliça de dez barras e numa das barras ocorrer 0, significa que tal barra não deve ser considerada. Do mesmo modo, um valor 0 serve para indicar que o grau de liberdade num nó não está restringido. Este método de otimização topológica é conhecido por *Ground Structure Approach* e é usado especificamente em estruturas reticuladas.

Para problemas de elasticidade de meios contínuos pode aplicar-se um método conhecido como *Soft-Kill Method*. O método começa por considerar o domínio **Ω** da estrutura como um meio contínuo, discretizado por uma malha de elementos finitos (EF). É estabelecida uma relação entre o módulo de elasticidade de cada elemento e a sua densidade, sendo esta última a variável de projeto, isto é, existirão tantas variáveis quantos os elementos na malha. Posteriormente são realizadas sucessivas análises e a otimização topológia consiste em obter a distribuição ótima de material através da variação entre 0 e 1 do valor da densidade, até ser satisfeito um determinado critério de paragem. A configuração final resulta da inicial, no entanto apresenta-se com zonas escuras de elevada densidade, que indicam existência de estrutura, zonas claras que indicam vazio e zonas cinzentas que dão uma noção do contorno da estrutura. Um exemplo de otimização topológica é apresentado na Figura 2.3

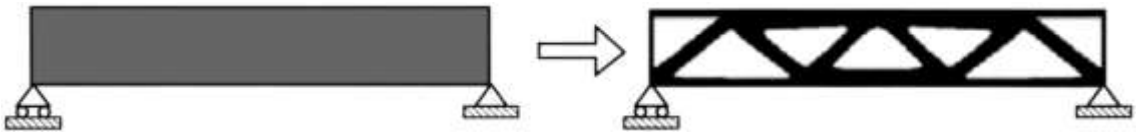

Figura 2.3 - Exemplo de otimização topológica.

De forma a resumir e a fazer uma ligação entre o tipo de otimização e as variáveis de projeto apresenta-se a Tabela 2.1.

Um problema apresenta-se especialmente complexo quando se pretende a otimização simultânea de dimensões, de forma e de topologia, em que as variáveis de projeto podem ser simultaneamente discretas, contínuas e booleanas.

Tabela 2.1 - Tipos de variáveis de projeto dependendo da formulação do problema de otimização estrutural, extraído de [2].

Tipo de otimização	Tipo da variável de projeto	Significado físico possível	Observação
Dimensional	Inteira	Seleção de secções transversais	Procura através de uma tabela de valores normalizados
	Real	Dimensões válidas num intervalo de variação contínuo	_____
Forma ou configuração	Geométrica (Real)	Coordenadas de pontos que definem a geometria	Coordenadas variam entre limites inferior e superior
Topologia	Booleana	Existência ou ausência de material	_____
	Real	Densidade do material variável	

2.2. Revisão de Alguns Conceitos de Cálculo

2.2.1. Sensibilidades

De acordo com [6] em Mecânica estrutural o cálculo da alteração que decorre no comportamento de uma estrutura quando se altera um parâmetro é denominado por análise de sensibilidades.

Esta área ganhou relevância com o desenvolvimento da otimização estrutural, a partir da década de 80, pois os algoritmos usados nessa altura requeriam o cálculo de gradientes e a análise de sensibilidades disponibilizava os métodos para o seu cálculo. Na otimização estrutural a função objetivo e os constrangimentos de um problema correspondem a medidas do desempenho ou da performance estrutural, como por exemplo: peso, volume, deslocamentos ou tensões. Os parâmetros que definem a estrutura, são designados por variáveis de projeto. A função do algoritmo de otimização é a escolha adequada desses parâmetros, ou seja, a escolha das variáveis de projetos adequadas. Se designarmos uma performance estrutural por Ψ e as variáveis de projeto por X_i, a análise de sensibilidades permite obter os gradientes da função objetivo ou dos constrangimentos em ordem às variáveis de projeto, $\frac{\partial \Psi}{\partial X_i}$. Exemplos de algoritmos que utilizam gradientes são o SQP e o MMA.

Um dos métodos mais usados para obter derivadas, é o método das diferenças finitas, que tem a vantagem de ser genérico, isto é aplicável a todo o tipo de performances e variáveis. Este método calcula uma aproximação da derivada, perturbando a variável X por adição (e/ou

subtração) de um valor ΔX e calculando novamente a performance Ψ. Nesta dissertação foi usado o método das diferenças finitas progressivas representado pela equação 2.2 e pela Figura 2.4.

$$\frac{\partial \Psi}{\partial X_i} \approx \frac{\Psi(X + \Delta X) - \Psi(X)}{\Delta X} \tag{2.2}$$

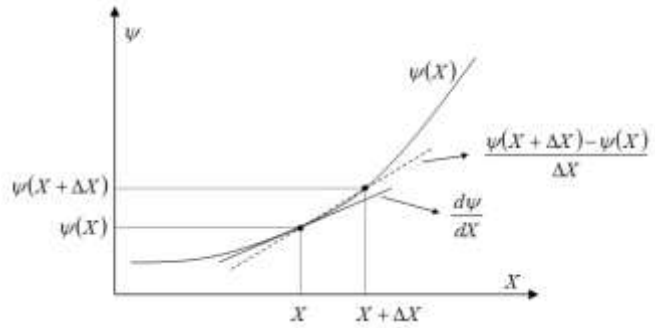

Figura 2.4 - Diferenças Finitas Progressivas, extraído de [8]

A grande vantagem deste método é a facilidade de aplicação para qualquer tipo de performance ou variável. Como inconvenientes deve mencionar-se o tempo de cálculo, pois para cada variável é necessário realizar uma análise suplementar. Por outro lado, existe a necessidade de escolher cuidadosamente o valor de ΔX, um valor elevado conduz a uma aproximação grosseira de $\frac{\partial \Psi}{\partial X_i}$, mas um valor baixo pode produzir erros de arredondamento.

Outro método que é utilizado nesta dissertação para o cálculo das sensibilidades é o **método continuo**, este método deriva a equação de equilíbrio na sua forma contínua. A grande desvantagem das sensibilidades contínuas é a sua complexidade teórica enquanto as principais vantagens são a utilização de uma formulação matemática rigorosa e uma relação entre gradientes e quantidades físicas não dependente da discretização.

2.2.2. Vetor Gradiente

Considere a função $f(\boldsymbol{x})$ com *n* variáveis $x_1, x_2, \dots, x_n$. A derivada parcial da função em relação à variável x_1 no ponto $\boldsymbol{x}^*$ é definida como $\frac{\partial f(\boldsymbol{x}^*)}{\partial x_1}$. O vetor gradiente corresponde a um vetor coluna de derivadas parciais de $f(\boldsymbol{x})$ no ponto $\boldsymbol{x}^*$ é representado por $\nabla f(\boldsymbol{x}^*)$ na equação 2.3.

$$\nabla f(\boldsymbol{x}^*) = \left[\frac{\partial f(\boldsymbol{x}^*)}{\partial x_1}\ \frac{\partial f(\boldsymbol{x}^*)}{\partial x_2} \ldots \frac{\partial f(\boldsymbol{x}^*)}{\partial x_n}\right]^T = \begin{bmatrix} \frac{\partial f(\boldsymbol{x}^*)}{\partial x_1} \\ \frac{\partial f(\boldsymbol{x}^*)}{\partial x_2} \\ \vdots \\ \frac{\partial f(\boldsymbol{x}^*)}{\partial x_n} \end{bmatrix} \tag{2.3}$$

Geometricamente, o vetor gradiente é normal ao plano tangente à superfície que representa a função f no ponto $\boldsymbol{x}^*$, como mostra a Figura 2.5 para uma função de três variáveis. Além disso aponta na direção de crescimento máximo da função [11].

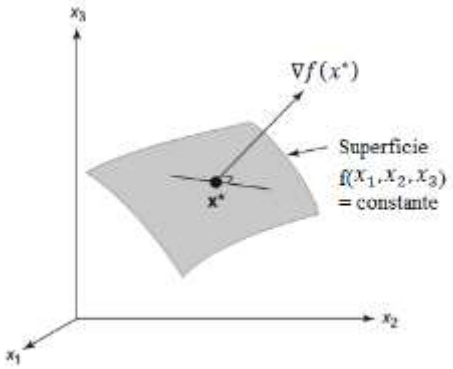

Figura 2.5 - Vetor gradiente para $f(x_1, x_2, x_3)$ no ponto $\boldsymbol{x}^*$, adaptado de [11].

2.2.3. Matriz Hessiana

Diferenciando o vetor gradiente mais uma vez, obtemos uma matriz de segundas derivadas parciais para a função $f(\boldsymbol{x})$ em ordem a cada variável x_i que é representada por $\nabla^2 f(\boldsymbol{x}^*)$ ou H apresentada na equação 2.4:

$$\mathrm{H} = \nabla^2 f(\boldsymbol{x}^*) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 x_{nj}} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_2} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \tag{2.4}$$

É importante notar que cada elemento da matriz Hessiana é uma função em si que é avaliada no ponto **x***. Além disso, $f(\boldsymbol{x})$ é assumido como duas vezes continuamente

diferenciável, as derivadas cruzadas são iguais o que implica que a matriz Hessiana é sempre uma matriz simétrica [11].

2.2.4. Expansão em serie de Taylor

Uma função pode ser aproximada por polinómios numa vizinhança de qualquer ponto em termos do seu valor e derivadas usando a expansão de Taylor. Considere uma função $f(\boldsymbol{x})$ de uma variável. A equação 2.5 apresenta a expansão de Taylor para $f(\boldsymbol{x})$ sobre o ponto **x***:

$$f(\boldsymbol{x}) = f(\boldsymbol{x}^*) + \frac{df(\boldsymbol{x}^*)}{d\boldsymbol{x}}(\boldsymbol{x} - \boldsymbol{x}^*) + \frac{1}{2}\frac{d^2 f(\boldsymbol{x}^*)}{d\boldsymbol{x}^2}(\boldsymbol{x} - \boldsymbol{x}^*)^2 + R \tag{2.5}$$

Onde *R* é o termo residual que é menor em magnitude que os termos anteriores se x for suficientemente próximo de $\boldsymbol{x}^*$.

Definindo $d = \boldsymbol{x} - \boldsymbol{x}^*$ como incremento ou perturbação na variável de projeto, a equação 2.5 transforma-se na equação 2.6:

$$f(\boldsymbol{x}^* + d) = f(\boldsymbol{x}^*) + \frac{df(\boldsymbol{x}^*)}{d\boldsymbol{x}}(d) + \frac{1}{2}\frac{d^2 f(\boldsymbol{x}^*)}{d\boldsymbol{x}^2}(d)^2 + R \tag{2.6}$$

No caso de uma função de *n* variáveis e utilizando notação a matricial a equação 2.6 transforma-se na equação 2.7:

$$f(\boldsymbol{x}^* + d) = f(\boldsymbol{x}^*) + \nabla f^T d + \frac{1}{2} d^T H d + R \tag{2.7}$$

ou

$$\Delta f = +\nabla f^T d + \tfrac{1}{2} d^T H d + R;\ \text{com}\ \Delta f = f(\boldsymbol{x}^* + d) - f(\boldsymbol{x}^*) \tag{2.8}$$

Uma variação de 1º ordem de $f(\boldsymbol{x})$ em $\boldsymbol{x}^*$ é obtida retendo apenas o primeiro termo da equação, tal como apresentado na equação 2.9:

$$\delta f = \nabla f^T \delta \boldsymbol{x};\ \delta \boldsymbol{x} = \boldsymbol{x} - \boldsymbol{x}^* \tag{2.9}$$

É uma aproximação aceitável para a variação da função original quando $\boldsymbol{x}$ está próximo de **x***[2].

2.2.5. Condições de Optimalidade

As condições de optimalidade são as condições que uma função $f(\boldsymbol{x})$ deve satisfazer no seu ponto de mínimo de acordo com [2]. O procedimento para derivar condições de optimalidade local assume que se está num ponto de mínimo **x*** e é examinada uma pequena vizinhança em sua volta, d, para estudar as propriedades da função como o declive e a curvatura através das suas derivadas. Visto que se considera uma vizinhança pequena, a função $f(\boldsymbol{x})$ pode ser aproximada por polinómios, numa vizinhança de qualquer ponto do domínio, em termos do seu valor e das suas derivadas usando a expansão em série de Taylor da equação 2.5.

Da expansão em série de Taylor de $f(\boldsymbol{x})$ decorrem as seguintes condições necessárias e suficientes (condições de optimalidade) para a existência de mínimo local num problema de otimização não constrangido:

• **Condição necessária de 1ª ordem**: Se $f(\boldsymbol{x})$ tem um mínimo local em $\boldsymbol{x}^*$, o vetor gradiente $\nabla f(\boldsymbol{x}^*)$ é igual a zero.

• **Condição necessária de 2ª ordem:** Se $f(\boldsymbol{x})$ tem um mínimo local em $\boldsymbol{x}^*$,a matriz Hessiana $\mathrm{H}(\boldsymbol{x}^*)$ é positiva semidefinida ou positiva definida no ponto $\boldsymbol{x}^*$.

• **Condição suficiente de 2ªordem:** Se a matriz $\mathrm{H}(\boldsymbol{x}^*)$ é positiva definida no ponto de estacionaridade $\boldsymbol{x}^*$, então $\boldsymbol{x}^*$ é um mínimo local da função $f(\boldsymbol{x})$.

No caso de um problema de otimização com constrangimentos de igualdade e desigualdade, as condições necessárias para a existência de mínimo local são habitualmente conhecidas como condições necessárias de Kuhn – Tucker (K–T).

• Condições necessárias de Kuhn – Tucker

Seja x* um ponto regular do conjunto dos constrangimentos e um mínimo local para $f(x)$ sujeito aos constrangimentos:

$$\begin{aligned} h_i(\boldsymbol{x}) &= 0;\ i = 1, \dots, p \\ g_i(\boldsymbol{x}) &\leq 0;\ i = 1, \dots, m \end{aligned} \tag{2.10}$$

Apresenta-se na equação 2.11 a função Lagrangeana do sistema:

$$\begin{aligned} L(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{u}, \boldsymbol{s}) &= f(x) \\ &+ \sum_{i=1}^{p} v_i h_i(\boldsymbol{x}) + \sum_{i=1}^{m} u_i (g_i(\boldsymbol{x}) + s_i^2) = f(\boldsymbol{x}) + v^T h(\boldsymbol{x}) + u^T (g(\boldsymbol{x}) + s^2) \end{aligned} \tag{2.11}$$

Então existem multiplicadores de Lagrange **v*** e **u*** tal que a Lagrangeana é estacionária em relação a x_j, v_i, u_i e s_i:

$$\underset{n+p+2m\ equações}{\begin{cases} \dfrac{\partial L}{\partial x_j} = \dfrac{\partial f}{\partial x_j} + \displaystyle\sum_{i=1}^{p} v_i^* \frac{\partial h_i}{\partial x_j} + \sum_{i=1}^{m} u_i^* \frac{\partial g_i}{\partial x_j} = 0, \quad j = 1, \dots, n \\ h_i(x^*) = 0; \quad i = 1, \dots, p \\ g_i(x^*) + s_i^2 = 0 \\ u_i^* s_i = 0; \quad i = 1, \dots, m \end{cases}} \tag{2.12}$$

Tem de verificar-se ainda que:

$$u_i^* \geq 0; \quad i = 1, \dots, m \tag{2.13}$$

2.3. Métodos de Otimização Baseados no Gradiente

Estes métodos, como o nome indica, usam gradientes das funções do problema para realizar a busca pelo ponto ótimo. Por este motivo, todas as funções do problema devem ser contínuas e pelo menos duas vezes continuamente diferenciáveis em todo o espaço de projeto admissível. Além disso, as variáveis de projeto são consideradas continuas e podem assumir qualquer valor dentro dos intervalos de variação permitidos[11].

De acordo com [2] os métodos de otimização baseados no gradiente podem ser classificados em dois grandes grupos:

- Métodos indiretos (ou analíticos) caraterizados pela aplicação de técnicas de minimização analíticas que, em geral, envolvem a resolução de um sistema de equações cujas soluções satisfazem as condições de otimalidade.
- Métodos diretos (ou numéricos) começam com uma solução que representa uma estimativa inicial do ponto ótimo e, por meio de um processo iterativo, as soluções subsequentes vão sendo melhoradas até que as condições de optimalidade sejam satisfeitas. Os métodos numéricos também são conhecidos por métodos de programação matemática.

Com o tempo, foram-se desenvolvendo algoritmos numéricos iterativos, que seguem os métodos de otimização baseados no gradiente, com o objetivo de procurar soluções ótimas para os problemas de projeto. Os algoritmos são iniciados com uma estimativa para a solução ótima que é melhorada iterativamente se não satisfizer as condições de optimalidade. Foi dado um ênfase especial aos algoritmos especializados na resolução de problemas de otimização constrangida, pois são o tipo de problemas mais abordados nesta dissertação.

De acordo com [11] muitos métodos de otimização baseados no gradiente são descritos pela seguinte ordem iterativa:

Vetor: $$x^{(k+1)} = x^{(k)} + \Delta x^{(k)}; \quad k = 0,1,2, \dots \tag{2.14}$$

Variável: $$x_i^{(k+1)} = x_i^{(k)} + \Delta x_i^{(k)}; \quad i = 1\ ate\ n\ ;\ k = 0,1,2, \dots \tag{2.15}$$

Onde *k* representa o número da iteração, *i* é número da variável, $x^{(o)}$ é o ponto de partida das variáveis e Δx^0a variação das variáveis.

O esquema iterativo descrito nas equações 2.14 e 2.15 é repetido até as condições de optimalidade serem satisfeitas e é aplicado tanto a problemas constrangidos como a problemas não constrangidos. Para problemas não constrangidos o cálculo de $\Delta x^{(k)}$ depende apenas da função objetivo e as suas derivadas. No caso de problemas constrangidos, para além da função objetivo e as suas derivadas o constrangimento também deve ser considerado bem como as suas derivadas, quando se realiza o cálculo de $\Delta x^{(k)}$.

Para problemas não constrangidos a maioria dos métodos utilizados para o cálculo de $\Delta x^{(k)}$ é decomposto em duas partes:

$d^{(k)}$- Direção de procura no espaço de projeto

α_k- Passo ótimo no espaço de projeto, escalar positivo.

O processo de passar de um ponto do projeto para o seguinte é ilustrado na Figura 2.6, onde *B* é o ponto de projeto atual $x^{(k)}$, $d^{(k)}$ é a direção de pesquisa e $\alpha_{(k)}$ é o comprimento do passo. Portanto, quando $\alpha_k d^{(k)}$ é adicionado ao ponto de projeto atual $x^{(k)}$, alcança-se um novo ponto *C* no espaço de projeto. Todo o processo é repetido a partir do ponto *C*. Existem muitos procedimentos para calcular o tamanho do passo α_k e a direção de busca $d^{(k)}$, várias combinações desses procedimentos foram usadas para desenvolver diferentes algoritmos de otimização, tais como [2]: o método do declive mais acentuado [12], o método do gradiente conjugado [13] e o método de Newton [14]

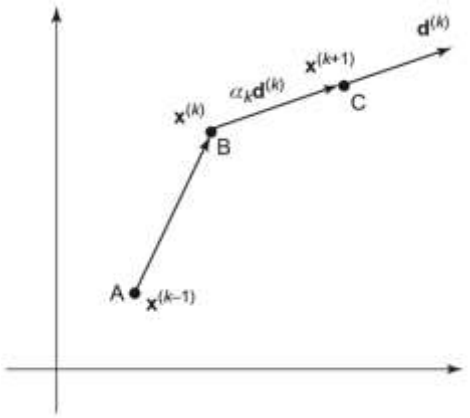

Figura 2.6 - Diagrama que representa as etapas iterativas de um método de otimização, extraído de [11].

Os algoritmos para problemas de otimização não constrangida e constrangida baseiam-se na mesma filosofia iterativa que pode ser descrita através da informação apresentada na Tabela 2.2.

Tabela 2.2 - Algoritmo geral de otimização, adaptado de [2].

1ª Etapa:	Estimar um ponto de partida razoável $x^{(0)}$ e inicializar o contador de iterações $k = 0$.
2ª Etapa:	Calcular uma direção de procura $d^{(k)}$ no espaço de projeto. Em geral, este cálculo requer o valor da função objetivo e o seu gradiente para problemas não constrangidos. Adicionalmente, para problemas constrangidos, são requeridos os valores dos constrangimentos e seus gradientes.
3ª Etapa:	Verificação de convergência do algoritmo com base na satisfação das condições de optimalidade ou na satisfação de algum outro critério de paragem. Se existe convergência termina-se o processo iterativo, caso contrário, segue-se para a 4ª Etapa.
4ª Etapa:	Cálculo do passo positivo ótimo, α_k , na direção $d^{(k)}$.
5ª Etapa:	Cálculo de um novo ponto no espaço de projeto: $x^{(k+1)} = x^{(k)} + \alpha_k d^{(k)}$ Fazer $k = k + 1$ e voltar à 2ª Etapa.

Contudo a grande diferença para problemas constrangidos é que deve levar-se em consideração o constrangimento enquanto se determina a direção de procura, bem como o passo ótimo. A Figura 2.7 apresenta uma maneira pela qual o algoritmo pode progredir no espaço de projeto.

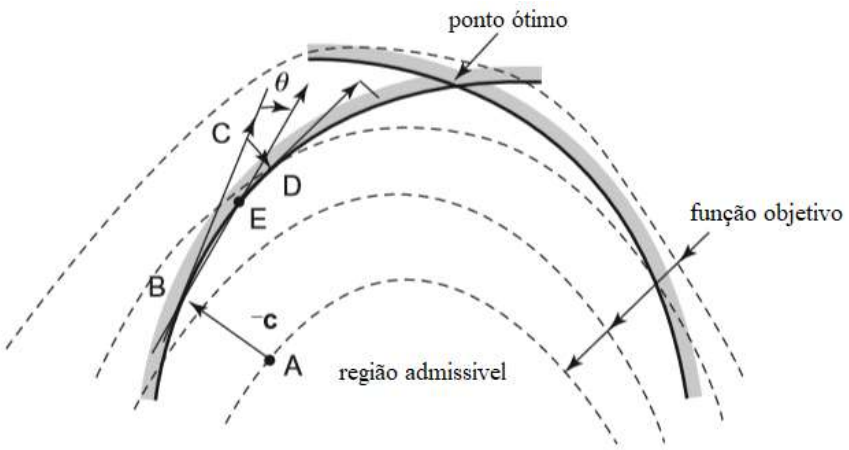

Figura 2.7 - Etapas de um algoritmo de otimização restrita, adaptado de [8]

O ponto B situa-se na fronteira da região admissível, algumas restrições estão ativas. Uma vez que o limite da restrição é encontrado no ponto B uma estratégia é deslocar-se em torno da tangente à fronteira da região admissível, tal como a direção BC. Contudo isto resulta num ponto inviável que é corrigido de forma a alcançar o ponto D. O processo repete-se até se atingir o ponto ótimo.

Algumas vantagens dos métodos baseados no gradiente são:

• A exploração da informação do gradiente pode aumentar significativamente a velocidade de convergência.

• Garantia do mínimo local ótimo.

• Clareza no critério de convergência.

2.3.1. Programação Quadrática Sequencial (SQP)

De acordo com [15] a Programação Quadrática Sequencial (SQP) ficou conhecida em 1970 como um dos métodos mais bem-sucedidos para a solução numérica de problemas de otimização não linear constrangida. O algoritmo SQP é um processo iterativo que modela um Problema de Otimização Não Linear (NLP) em redor do ponto $\boldsymbol{x}^{(k)}$, obtido para uma determinada iteração, $k \in \mathrm{IN}_0$, aproximando-o por um subproblema mais simples de Programação Quadrática (QP), e depois usa a solução do subproblema para construir uma nova iteração $\boldsymbol{x}^{(k+1)}$. A formulação de um problema recorrendo ao SQP segue a formulação clássica apresentada na equação 2.8.

A cada iteração $\boldsymbol{x}^{(k)}$ o subproblema QP que necessita de ser resolvido deve refletir as propriedades locais do NLP da iteração atual, a ideia natural passa por substituir:

• a função objetivo $f(\boldsymbol{x})$ por uma aproximação quadrática local, em concordância com a equação 2.16

$$f(\boldsymbol{x}) \approx f\left(\boldsymbol{x}^k\right) + \nabla f(\boldsymbol{x}^k)\left(\boldsymbol{x} - \boldsymbol{x}^k\right) + \frac{1}{2}\left(\boldsymbol{x} - \boldsymbol{x}^k\right)^T Hf(\boldsymbol{x}^k)(\boldsymbol{x} - \boldsymbol{x}^k) \tag{2.16}$$

• a função dos constrangimentos pelas equações 2.17 e 2.89:

$$g(\boldsymbol{x}) \approx g\left(\boldsymbol{x}^k\right) + \nabla g(\boldsymbol{x}^k)(\boldsymbol{x} - \boldsymbol{x}^k) \tag{2.17}$$
$$h(\boldsymbol{x}) \approx h\left(\boldsymbol{x}^k\right) + \nabla h(\boldsymbol{x}^k)(\boldsymbol{x} - \boldsymbol{x}^k) \tag{2.18}$$

Ao definir

$$d(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{x}^k$$

O subproblema QP fica com a forma definida na equação 2.19:

$$\begin{aligned} \min \quad & \nabla f\left(\boldsymbol{x}^k\right)^T d(\boldsymbol{x}) + \frac{1}{2} d(\boldsymbol{x})^T Hf(\boldsymbol{x}^k) d(\boldsymbol{x}) \\ Sujeito\ a \quad & h\left(\boldsymbol{x}^k\right) + \nabla h\left(\boldsymbol{x}^k\right)^T d(\boldsymbol{x}) = 0 \\ & g\left(\boldsymbol{x}^k\right) + \nabla g\left(\boldsymbol{x}^k\right)\left(\boldsymbol{x} - \boldsymbol{x}^k\right) \leq 0 \end{aligned} \tag{2.19}$$

Para considerar as não-linearidades nas restrições, mantendo a linearidade das restrições no subproblema, o método SQP usa um modelo quadrático da função Lagrangeana. Apresenta-se na equação 2.20 aproximação quadrática da serie de Taylor em *x* para a função Lagrangeana:

$$L(\boldsymbol{x}^k, \boldsymbol{u}^k, \boldsymbol{v}^k) + \nabla L(\boldsymbol{x}^k, \boldsymbol{u}^k, \boldsymbol{v}^k)^t d_x + \frac{1}{2} d_z^t HL(\boldsymbol{x}^k, \boldsymbol{u}^k, \boldsymbol{v}^k) d_z \qquad (2.20)$$

O motivo para o uso da equação 2.20 é a qualidade das aproximações produzidas que permitem gerar uma nova iteração que está mais próxima do ponto ótimo do NLP, e como resultado o algoritmo adquire uma boa convergência local. O método do algoritmo pode ser resumido nos seguintes passos:

Dadas as aproximações $(\boldsymbol{x}^0, \boldsymbol{u}^0, \boldsymbol{v}^0)$, H_0, e a função de mérito $\emptyset$, na posição $k = 0$.

1. Formar e resolver o subproblema QP para obter (d_x, d_u, d_v).
2. Escolha do passo ótimo de modo a que $\emptyset\,(\,\boldsymbol{x}^{(k)} + \alpha d_x) \leq \emptyset x^{(k)}$.
3. Definir

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \alpha d_x$$
$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \alpha d_u$$
$$\boldsymbol{v}^{k+1} = \boldsymbol{v}^k + \alpha d_v.$$

4. Parar se foi atingida a convergência
5. Calcular H_{k+1}
6. Definir $k = K + a$; voltar ao passo 1.

A utilização deste algoritmo é feita recorrendo ao Programa OCTAVE. Este algoritmo não necessitou de implementação uma vez que faz parte dos algoritmos implementados pelo programa OCTAVE.

2.3.2. Método das Assintotas Moveis (MMA)

Krister Svanberg introduz o Método das Assintotas Móveis (MMA) em 1987. De acordo com [16] o algoritmo MMA apresenta-se como um método de resolução de problemas que se encontrem em concordância com a equação 2.22.

$$\begin{array}{lll} \min\limits_{\boldsymbol{x}} & f_0(\boldsymbol{x}) + a_0 z + \sum_{i=1}^{m}(c_i y_i + \frac{1}{2} d_i y_i^2) & \\ Sujeito\ a & f_i(x) - a_i z - y_i \leq 0 & ; i = 1, \dots, m \\ & x_j^{min} \leq x_j \leq x_j^{max} & ; j = 1, \dots, n \\ & y_i \geq 0, & i = 1, \dots, m \\ & z \geq 0. & \end{array} \qquad (2.22)$$

Onde $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$ é o vetor das n variáveis de projeto, enquanto que $\boldsymbol{y} = (y_1, y_2, \dots, y_n)$ e z são as variáveis artificiais de otimização $f(\boldsymbol{x})$ é a função objetivo, x_j^{min} e x_j^{max} são os limites inferior e superior de variação das variáveis x_j. a_0 e a_i são números reais que satisfazem $a_0 > 0$ e $a_i \geq 0$. c_i e d_i são números reais que satisfazem $c_i \geq 0$, $d_i \geq 0$.

A abordagem utilizada pelo algoritmo para a resolução deste tipo de problemas pode ser descrita por este método: a cada iteração, os valores de $(\boldsymbol{x}^k, \boldsymbol{y}^k, \boldsymbol{z}^k)$ são fornecidos. Depois é gerado um subproblema aproximado, no qual as funções $f_i(x)$ são substituídas por funções convexas $f_i^{\sim(k)}(x)$. A escolha destas funções é baseada sobretudo em informação do gradiente no ponto de iteração atual, mas também em parâmetros como $u_j^{(k)}$ *e* $l_j^{(k)}$ (assimptotas moveis) atualizados em cada iteração com base nas informações dos pontos de iteração anteriores. O subproblema é resolvido e a solução será o próximo ponto do processo iterativo $(\boldsymbol{x}^{k+1}, \boldsymbol{y}^{k+1}, \boldsymbol{z}^{k+1})$. Um novo subproblema é gerado, e voltam a repetir-se os passos anteriores.

Para ser possível a utilização deste algoritmo, foi necessária a sua implementação no programa OCTAVE e a cedência por parte do Professor Krister Svanberg do algoritmo e alguns documentos com informações úteis sobre a sua utilização.

2.4. Métodos de otimização não baseados em gradientes

Nem todos os problemas de otimização são diferenciáveis. Para resolução destes últimos, é necessário a utilização de métodos alternativos aos baseados em gradientes, em que as funções dos problemas não necessitem ser diferenciáveis ou mesmo contínuas. Este objetivo tem sido atingido pelas Meta-Heurísticas, introduzidas em 1986 por Glover[2]. O termo Meta-Heurísticas serve para englobar um conjunto de técnicas conhecidas na literatura anglo-saxónica por: *Tabu Search*[17], *Simulated Annealing*[18], *Genetic Algorithms*[19], *Evolutionary Methods*[20], *Scatter Search*[21], *Neural Networks*[22], *GRASP2*[23], *Variable Neighborhood Search*[24], *Ant Colonies*[25]. Grande parte destes métodos de pesquisa são inspirados na natureza e usam apenas os valores das funções no processo de pesquisa.

De acordo com [2] uma heurística é uma técnica que procura boas soluções (quase ótimas) a custo computacional razoável sem ser capaz de garantir a admissibilidade e optimalidade das mesmas. Em alguns casos, nem sequer pode determinar quão perto do ótimo se encontra a solução

admissível. Em contraste com um método de otimização exato ou determinístico (baseado no gradiente), uma heurística é um método de pesquisa "menos ambiciosa", isto é, vai calculando o valor de uma função objetivo em pontos julgados promissores que vão sendo identificados ao longo de um processo iterativo. Quando este processo termina ter-se-á encontrado uma solução quase ótima. Eventualmente o ótimo global pode ser encontrado, mas não existem garantias disso.

Algumas vantagens dos métodos Heurísticos podem ser descritas como:

• Os métodos heurísticos não dependem das propriedades analíticas de continuidade ou de diferenciabilidade da função.

• São capazes de lidar com um maior número de problemas da vida real.

• Revestem-se de estratégias de pesquisa que permitem escapar de um ótimo local.

2.4.1. Algoritmo Genético (AG)

Holland [19] introduz o algoritmo genético (AG), uma heurística inspirada na teoria da evolução natural. Este algoritmo reflete o processo de seleção da natureza, onde os indivíduos mais aptos são selecionados e os mais fracos acabem por morrer.

A grande maioria dos processos de evolução que ocorrem na natureza são descritos através de dois acontecimentos: seleção natural e reprodução. Os indivíduos mais aptos da população sobrevivem e por este motivo tem a oportunidade de se reproduzirem, transmitindo uma combinação de genes aos seus descendentes.

De acordo com [2] os AG lidam com uma população de indivíduos. Cada um representa de forma codificada um ponto no espaço das soluções possíveis de um problema de otimização. A codificação é realizada em analogia com estruturas genéticas: cromossomas e genes. Cada indivíduo ou ponto é definido por um cromossoma cuja informação genética é única. O algoritmo evolui para a solução ótima do problema de tal modo que os indivíduos da população com melhor aptidão, ou pontos do domínio com melhores valores da função objetivo, terão uma probabilidade maior de serem selecionados para reprodução.

O método do algoritmo Genético pode ser resumido pelos seguintes passos:

1. Gen=0
2. Geração aleatória de uma população inicial de indivíduos,
3. Avaliação da aptidão ou desempenho,
4. Se o critério é verificado, fim,
5. Se o critério não é verificado, aplicam-se os operadores seleção, cruzamento e mutação,
6. Nova geração de indivíduos,
7. Gen=Gen+1; voltar ao passo 3.

O operador seleção realiza a escolha dos cromossomas com os melhores valores de aptidão. A seleção pode ser realizada de várias formas diferentes, das quais se destacam as 3 seguintes: seleção por roleta, torneiro ou aleatória. A primeira funciona como uma roleta de um casino, quanto maior for a aptidão de um indivíduo mais espaço ele ocupa na roleta dando uma maior probabilidade de ele ser escolhido [26]. Na segunda, o torneio, *n* indivíduos são selecionados aleatoriamente da população e os indivíduos selecionados competem entre si. O indivíduo com a maior aptidão ganha e será incluído como parte da população da próxima geração [27]. Por último, na seleção aleatória, como o nome indica os indivíduos são selecionados aleatoriamente não existindo qualquer influência da sua aptidão [28].

A Tabela 2.3 estabelece a correspondência entre a terminologia utilizada em algoritmos clássicos de otimização e os AG.

Tabela 2.3 - Comparação de terminologias, adaptado de [2].

Algoritmos genéticos	Otimização clássica
Indivíduo	Solução no espaço de projeto
População	Conjunto de soluções
Desempenho ou aptidão	Função objetivo
Geração	Iteração
Aplicação de operadores genéticos	Geração de novas soluções

O AG utilizado nesta dissertação divide-se em dois: REAL e BINÁRIO. O primeiro permite resolver problemas com variáveis reais e o segundo com variáveis binárias. O AG BINÁRIO utiliza uma codificação binária das variáveis caraterizada por um cromossoma contendo vários genes, cada qual podendo tomar o valor 0 ou 1. É um algoritmo apropriado para resolver problemas discretos, por exemplo selecionar perfis de uma lista de perfis disponíveis. Esses perfis constam normalmente de uma tabela, por exemplo uma tabela de perfis HEB, contendo as várias secções transversais. Cada variável do problema é codificada através de vários genes. Para se fazer uma ligação entre o gene e a respetiva variável de projeto é necessário realizar a descodificação do cromossoma. Para exemplificar refere-se a descodificação do *cromossoma* 11001 num inteiro: $1 \times 2^0 + 0 \times 2^1 + 0 \times 2^2 + 1 \times 2^3 + 1 \times 2^4 = 25$. O maior número inteiro que um cromossoma com *k* genes pode codificar é $2^k - 1$.

Assim o número de genes num problema depende do número de variáveis e do número de genes para cada variável. Isto significa, por exemplo, que num problema com *11* variáveis e com *16* opções possíveis para cada variável, são necessários $11 \times 4 = 44$ genes porque o maior número inteiro que um cromossoma com *k* genes, (bits), pode codificar é $2^k - 1$ e é necessário ter $k = 4$ para poder representar 16 opções ($2^4 - 1 = 15$, logo temos 16 opções: 0, 1, 2, … , 15).

Outros parâmetros que se apresentam comuns aos AG REAL e BINÁRIO são [2]:

- Número de gerações.
- Dimensão da população.
- Probabilidade de cruzamento entre 0.6 e 0.9.
- Probabilidade de mutação entre 0.001 e 0.01.

Para ser possível a utilização deste algoritmo, foi necessário a sua implementação no programa OCTAVE. O algoritmo foi disponibilizado pelo projeto Yarpiz [29].

2.4.2. Otimização por Enxame de Partículas (PSO)

Otimização por enxame de partículas, (PSO), é um algoritmo baseado na inteligência dos enxames. É inspirado no comportamento social dos animais como peixes, aves e abelhas. É um algoritmo simples, mas bastante poderoso. Para demostrar a maneira como o algoritmo funciona usa-se um exemplo retirado da YARPIZ [29]. Imagine que dois amigos, o investigador (A) e o investigador (B) que pretendem encontrar o ponto mais profundo de um lago, como o representado na Figura 2.8.

Os amigos pretendem cooperar para acharem a resolução deste problema. É um problema de pesquisa e também pode ser visto como um problema de otimização.

O investigador (A) e o investigador (B) têm 2 barcos e com eles ferramentas para medir a profundidade do lago em todos os locais.

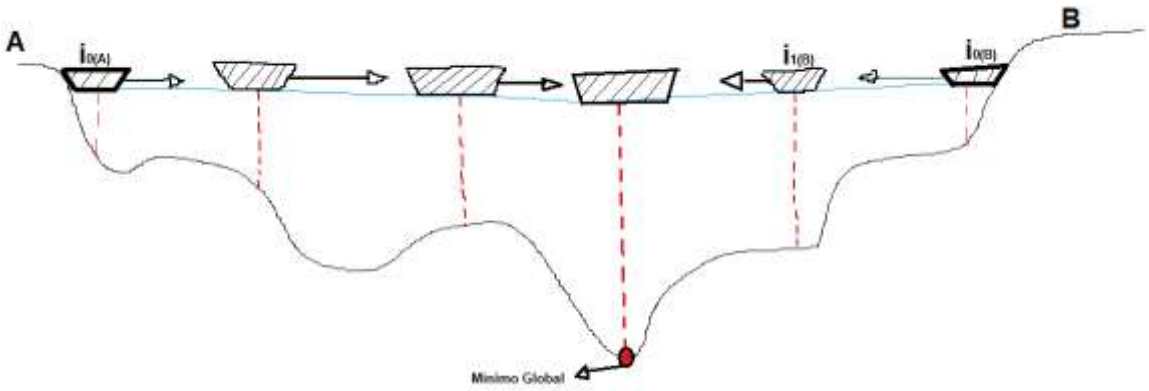

Figura 2.8 - Exemplo retirado da YARPIZ, profundidade de um lago.

No início o barco do investigador (A) está na margem esquerda i_0 (A) e o barco do investigador (B) está na margem da direita i_0 (B). Ambos medem a profundidade do lago e depois partilham a informação entre eles. A primeira medição de (A) apresenta melhores resultados. Por isso (B) aproxime-se um pouco de (A), um movimento pequeno, atualiza a sua posição e realiza outra medição. Nesta medição a profundidade de (B) aumentou o suficiente para ser a melhor.

Agora desloca-se a (A) para mais próximo de (B), um movimento pequeno, atualiza a sua posição e realiza uma medição. Mas a de (B) continua a ser a melhor. Por isso a (A) muda a sua posição e faz novas medições ate que chega ao mínimo global do problema. No entanto (B) continua a mover-se ate encontrar a (A) e chegam por fim ao mínimo global do problema.

O PSO é muito similar a este mecanismo que o investigador (A) e (B) usaram. Na estratégia desta cooperação dois princípios são usados:

• **Comunicação:** a profundidade é sempre transmitida à outra parte.

• **Aprendizagem:** a partilha da profundidade permite aprender que a localização do parceiro é melhor.

Aprender o conceito de melhor é o problema principal que um optimizador tem que resolver, por que o objetivo de um problema de otimização é encontrar o melhor.

Nesta simples historia, a cooperação do investigador (A) e (B) para achar o mínimo de uma função, foram discutidas as regras básicas por detrás da inteligência dos enxames. Os investigadores (A) e (B) não sabiam onde estava localizado o mínimo global do problema, contudo cooperaram para encontrar a resposta. A inteligência dos enxames é algo similar a isto. O PSO contém uma população de candidatos possíveis chamada enxame de partículas. Cada partícula contém uma posição no espaço de pesquisa do problema. A informação partilhada entre cada partícula vai permitir descobrir o mínimo global do problema.

Para ser possível a utilização deste algoritmo, foi necessária a sua implementação no programa OCTAVE. O algoritmo foi disponibilizado pelo projeto Yarpiz [29].

3. PROAES

O PROAES é um programa didático que foi desenvolvido na Faculdade de Ciências e Tecnologias da Universidade Nova de Lisboa. Foi especificamente desenvolvido para ser aplicado no ensino da otimização de estrutural, uma vez que está capacitado para realizar o cálculo estrutural relacionado com as performances e variáveis.

O programa PROAES é composto por um conjunto de módulos que apresentam as seguintes capacidades:

1. Realizar a análise estática de modelos estruturais pelo método dos elementos finitos. Podem ser utilizados os elementos do tipo barra ou viga indicados na Tabela 3.1.
2. Calcular derivadas de funções associadas ao desempenho estrutural (performances) em ordem a variáveis de projeto.
3. Realizar análises de fiabilidade de modelos estruturais pelo método FORM.
4. Otimizar modelos de estruturas, considerando constrangimentos associados a medidas de desempenho estrutural e/ou medidas de fiabilidade.

Tabela 3.1 - Elementos finitos utilizados no programa PROAES.

Elemento	Tipo	Esforços Internos	Matriz K_e
Barra 2D	1	N	4x4
Barra 3D	2	N	6x6
Viga 2D	3	N, V*y*, M*z*	6x6
Viga 3D	4	N, V*y*, V*z*, T, M*y*, M*z*	12x12

O programa está escrito em linguagem MATLAB, podendo também ser utilizado em ambiente OCTAVE.

O programa PROAES é constituído por um total 16 sub-rotinas e a rotina principal que, consoante os dados inseridos pelo utilizador, decide quais as sub-rotinas a usar para determinada análise. O seu funcionamento geral pode ser visualizado no fluxograma apresentado na Figura 3.1 [6]

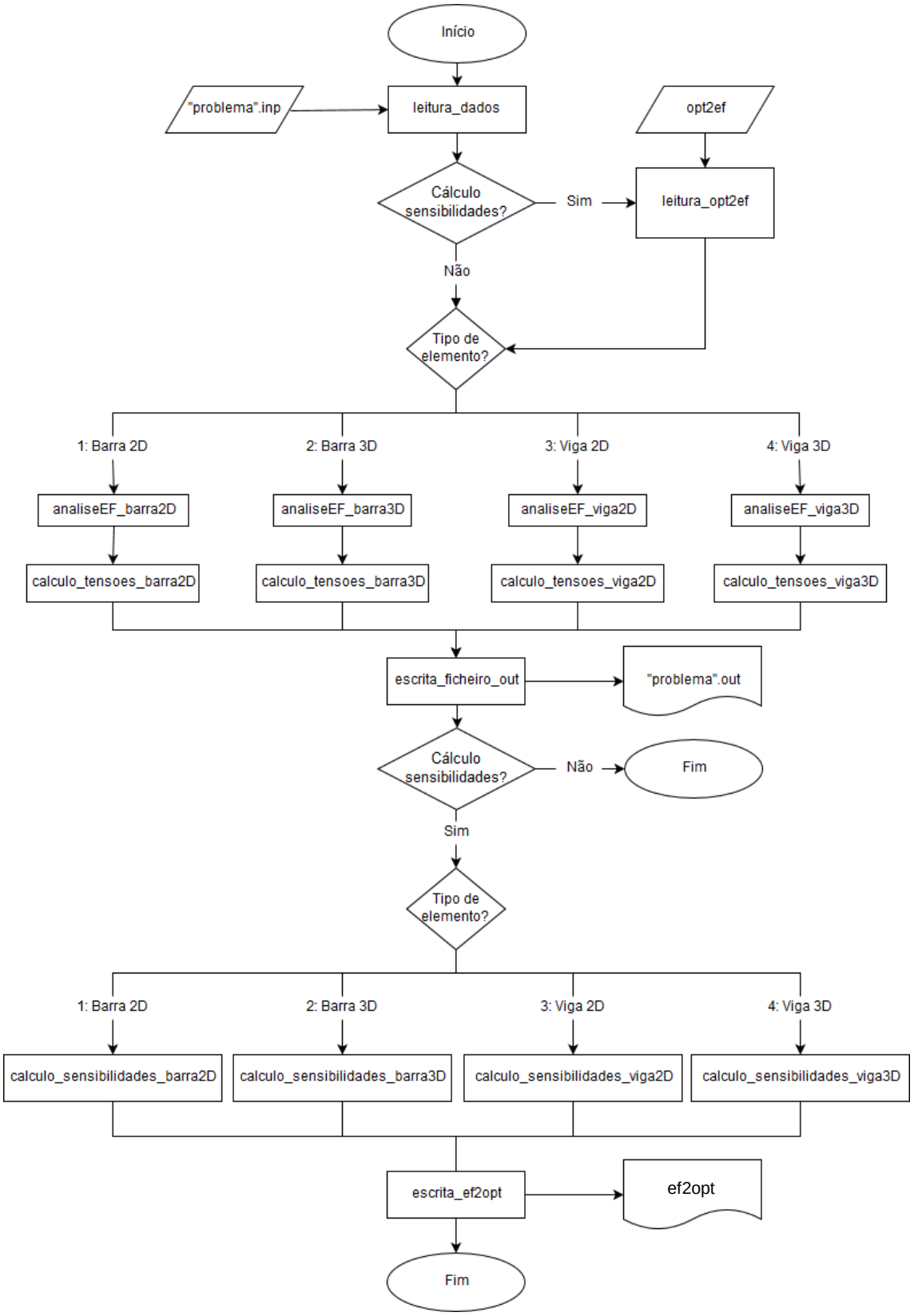

Figura 3.1 - Estrutura do programa PROAES, extraído de [22].

3.1. Análise de sensibilidades pelo método contínuo

O programa PROAES permite calcular derivadas de funções associadas ao desempenho estrutural em ordem a variáveis de projeto através do método contínuo de análise de sensibilidades. Essas derivadas podem ser usadas para calcular a fiabilidade de uma estrutura pelo método FORM ou para realizar a otimização de uma estrutura através de um algoritmo que necessite de gradientes.

A versão 5.0 apresenta a capacidade de calcular derivadas de 3 medidas do desempenho estrutural (volume, deslocamento de um nó e tensão num elemento) em relação a 5 tipos de variáveis de projeto (coordenada de um nó, módulo de elasticidade do material que constitui a estrutura, fator de carga, área da secção transversal e inércia da secção transversal). A disponibilidade de informação sobre sensibilidades para as várias performances e variáveis de projeto do programa PROAES é apresentada na Tabela 3.2

Tabela 3.2 - Performances e variáveis de projeto disponíveis no programa PROAES, adaptado de [6]

		Performances		
		Volume	Deslocamento de um nó	Tensão num elemento
Variáveis	Coordenada de um nó	Todos	Todos	Todos exceto Viga 3D
	Módulo de elasticidade	Todos	Todos	Todos exceto Viga 3D
	Fator de carga	Todos	Todos	Todos exceto Viga 3D
	Área de secção transversal	Todos	Todos	Todos exceto Viga 3D
	Inércia de secção transversal	Viga 2D e 3D	Viga 2D e 3D	Todos exceto Viga 3D

Para um melhor entendimento do funcionamento deste programa deve ser analisada a dissertação de mestrado de João Marmeleiro [6] e o manual de funcionamento do PROAES por ele criado [30].

3.2. Otimização Estrutural

Até à presente data, para realizar uma otimização recorrendo ao PROAES era necessário a utilização de um outro programa, o MATLAB, que continha os algoritmos de otimização. A forma como se processava a otimização é apresentada na Figura 3.2.

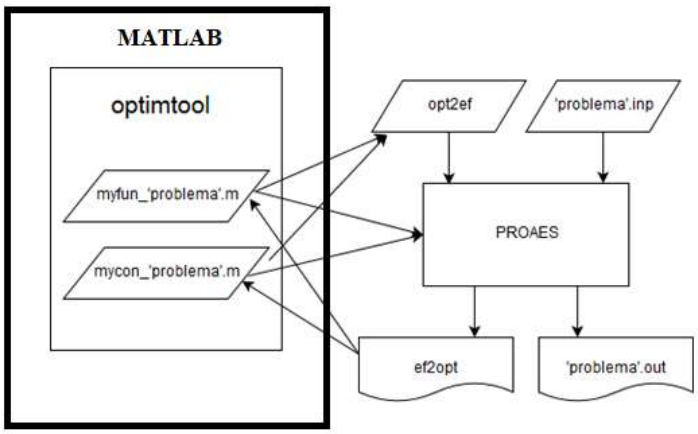

Figura 3.2 - Interligação entre o programa PROAES e o OPTIMTOOL, adaptado de [3]

O objetivo da presente dissertação é criar um módulo de otimização composto pelos algoritmos apresentados no capítulo 2. Esse módulo passa a realizar a otimização efetuada até a data pela OPTIMTOOL do MATLAB. O processo de otimização passa a funcionar de acordo com a Figura 3.3.

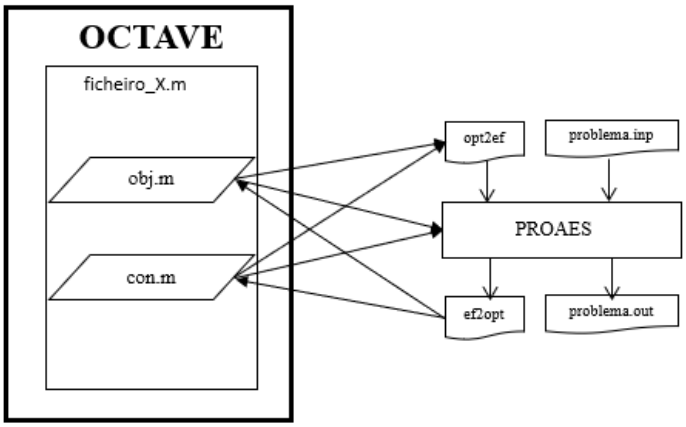

Figura 3.3 - Interligação entre o programa PROAES e o TOOLBOX.

A otimização passa a ser controlada pelo **ficheiro_X.m**. Como se explica em detalhe no Capítulo 4, em alguns casos esse ficheiro é um ficheiro de inicialização, noutros casos contém o próprio algoritmo.

4. TOOLBOX de Otimização

4.1. O que é a TOOLBOX de Otimização?

É uma ferramenta que tem disponível um conjunto de funções que permitem encontrar parâmetros que minimizam ou maximizam objetivos enquanto satisfazem restrições.

Todas as funções da TOOLBOX são arquivos M do OCTAVE, elas contêm instruções que implementam algoritmos especializados em otimização, inclusive:

• Programação Quadrática Sequencial (SQP)

• Método das assintotas moveis (MMA)

• Algoritmo Genético (GA)

• Otimização por Enxame de Partículas (PSO)

As funções podem ser utilizadas para encontrar soluções ideais em problemas contínuos ou discretos.

Para aproveitar ao máximo as capacidades de otimização destes algoritmos o utilizador deve criar ficheiros que permitam combinar o OCTAVE com outros programas.

Nas secções seguintes irá ser explicado de forma detalhada como se deve proceder para realizar otimização com cada um dos algoritmos disponíveis. Foi escolhido como exemplo um problema de otimização bastante simples. O objetivo é que o utilizador perceba quais os ficheiros que deve criar, quando está a trabalhar com determinado algoritmo.

4.2. Problema de Otimização – *Treliça 2 Barras*

A estrutura escolhida é composta por 3 nós e 2 elementos de barra (ver Figura 4.1). O objetivo é minimizar o volume total da estrutura, com constrangimento de tensão em cada barra. As variáveis do problema correspondem à área de secção transversal de cada barra.

Dados do problema:

Tabela 4.1 - Valores iniciais das variáveis.

Área inicial do 2 Elementos	$A = 0.03m^2$
Tensão Admissível	$100\ MPa$
Número de Elementos	2
Número de nós	3

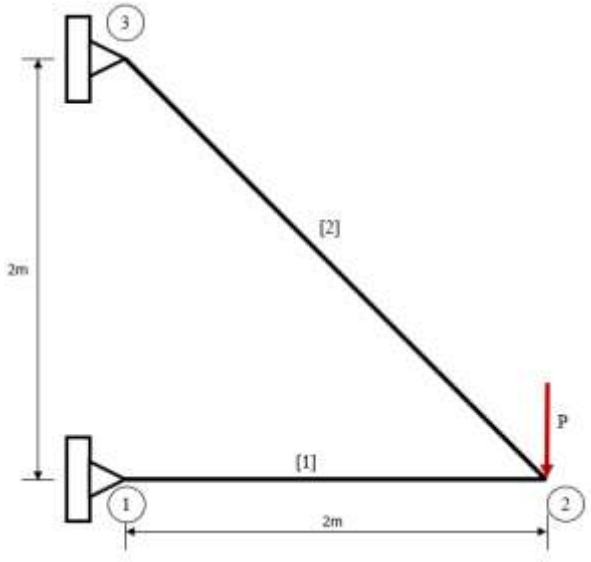

Figura 4.1 - Treliça 2 Barras

Formulação do Problema de Otimização Dimensional

• Função objetivo

- Minimizar o volume da estrutura:

•Variáveis do problema

- A área da secção transversal de cada barra:

• Constrangimentos do problema

- A tensão admissível de cada barra não deve ser ultrapassada.

$$\sigma \leq 100\ MPa \tag{4.1}$$

- A área da secção transversal de cada barra está limitada superior e inferiormente:

$$1 \times 10^{-8} < A_i < 3.0 \times 10^{-2}\ m^2 \tag{4.2}$$

O problema de otimização fica com a seguinte forma,

$$\begin{aligned} \min_{A} \quad & f(\boldsymbol{A}) = Volume \\ s.a. \quad & \sigma_j \leq 100\ MPa \qquad j = 1,2 \\ & 1 \times 10^{-8} < A_i < 3.0 \times 10^{-2}\ m^2 \qquad i = 1,2 \end{aligned} \tag{4.3}$$

4.3. Programação quadrática sequencial (SQP)

Para resolver o problema de otimização *Treliça 2 Barras* o utilizador deverá realizar os seguintes passos:

• Formular o Problema

• Definir o Problema de acordo com a sintaxe do algoritmo, o que implica:

- Definir a função objetivo, em linguagem OCTAVE, num ficheiro M
- Definir o constrangimento, em linguagem OCTAVE, num ficheiro M
- Definir o Gradiente da função objetivo, em linguagem OCTAVE, num ficheiro M (este passo é opcional)
- Definir o Gradiente do constrangimento, em linguagem OCTAVE, num ficheiro M (este paço é opcional)

• Executar a otimização

4.3.1. Formulação do Problema

Considere a formulação matemática de um problema de acordo com o SQP do OCTAVE, que é ligeiramente diferente da apresentada em (2.1)

$$\begin{array}{lll} \min\limits_{x} & f(\boldsymbol{x}) & \\ Sujeito\ a & g_j(\boldsymbol{x}) \geq 0 & ; j = 1, \dots, m \\ & h_k(x) = 0 & ; k = 1, \dots, p \\ & x_i^l \leq x_i \leq x_i^u & ; i = 1, \dots, n \end{array} \qquad (4.4)$$

Onde $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$ é o vetor das n variáveis de projeto, $f(\boldsymbol{x})$ é a função objetivo, as funções $g_j(\boldsymbol{x})$ são os m constrangimentos de desigualdade, $h_k(\boldsymbol{x})$ são os p constrangimentos de igualdade e x_i^l e x_i^u são os limites inferior e superior de variação das variáveis x_i, respetivamente. A diferença consiste no valor do sinal do constrangimento, no SQP OCTAVE passa a ser positivo.

Considere agora o problema *Treliça 2 Barras*. O utilizador deverá escrever a função objetivo e os constrangimentos recorrendo a ficheiros M do OCTAVE usando expressões que reflitam a forma matemática do algoritmo.

4.3.2. Função objetivo

O utilizador deve definir a função objetivo em linguagem OCTAVE, num ficheiro M. Um ficheiro M é um ficheiro de texto que contem comandos OCTAVE e é guardado com a extensão *.m*.

1. No editor do OCTAVE abrir um novo ficheiro;
2. Definir a função objetivo;
```
function f = myFun(x)
f = volume
```
3. Guardar o ficheiro com a extensão .m.

A função objetivo podia ser definida com a seguinte expressão: $f = x_1 \times l_1 + x_2 \times l_2$ e o OCTAVE realizava o cálculo do volume. Contudo, para demonstrar o uso do PROAES, pede-se que realize o cálculo da função objetivo. Para isso devem ser escritos ficheiros que permitam a comunicação entre o PROAES e o OCTAVE. Este tipo de colaboração entre o OCTAVE e o PROAES é bastante vantajosa, pois para estruturas mais complexas o PROAES realiza o cálculo com uma tremenda facilidade.

No parágrafo que se segue apresenta-se o pseudocódigo **myFun.m** que mostra como calcular a função objetivo com a ajuda do PROAES. Para que o utilizador perceba melhor a forma como deve construir os ficheiros a entregar ao PROAES deve consultar-se a dissertação de Marmeleiro [6].

```
Função f = myFun(x)
Escrita do ficheiro opt2ef  que posteriormente é lido pelo PROAES
%Cálculo da função objetivo recorrendo ao PROAES
resultado= PROAES('trelica_2_barras','o');
Leitura do ficheiro ef2opt (que contem o valor da função objetivo
calculado pelo PROAES).
% definição da função objetivo
f= Volume;
fim % fim do ficheiro
```

De acordo com o pseudocódigo apresentado no parágrafo anterior um conjunto de linhas de código escreve o ficheiro *opt2ef* que posteriormente é entregue ao PROAES com a informação necessária para realizar o cálculo da função objetivo. O PROAES escreve o ficheiro *ef2opt* com os valores da função objetivo. Realiza-se a leitura dos valores da função objetivo, volume. Define-se *f* igual ao valor do volume.

4.3.3. Constrangimento

A função do constrangimento deve ser formulada para que esteja na forma $g(x) \geq 0$. O constrangimento necessita de ser reformulado para $100 \times 10^6 - c_j(x) \geq 0$, onde c_j é a tensão na barra *j* para que esteja de acordo com a sintaxe do algoritmo. Assim, sempre que o valor da tensão na barra *j* for superior a 100×10^6 o valor do constrangimento fica negativo, o que indica que está a ser violado, caso contrário, sempre que o valor da tensão na barra é inferior a 100×10^6 o constrangimento é positivo o que indica que está a ser respeitado pois é superior ou igual a 0. O valor 100×10^6 é a tensão admissível para este problema.

O cálculo do constrangimento é realizado pelo PROAES, um esboço da estrutura do código para a função constrangimento, **myCon.m**, é apresentado no parágrafo seguinte. Quando se combinam constrangimentos de vários tipos, por exemplo, quando tensões e deslocamentos têm de ser simultaneamente inferiores aos valores admissíveis, é importante normalizar os constrangimentos, o que se faz dividindo-os pelos respetivos valores admissíveis. O constrangimento de tensão fica assim definido por: $\frac{100\times10^6 - c_j(x)}{100\times10^6} \geq 0$.

```
Função g = myCon(x)
%o constrangimento deve ser entregue como um vetor coluna
g= [;…;];
Escrita do ficheiro opt2ef  que posteriormente é lido pelo PROAES
% Cálculo do constrangimento recorrendo ao PROAES
resultado= PROAES('trelica_2_barras','o');
Leitura do ficheiro ef2opt que contem o valor do  constrangimento.
% Definição do valor do constrangimento
para j = 1:nº de barras
     se a tensão for  >= 0 % tração
         g(j)= (Tensão Máxima – tensão da barra j)/Tensão máxima;
     caso contrário
         g(j)= (Tensão Máxima + tensão da barra j)/Tensão máxima;
     fim
fim
Fim do ficheiro
```

De acordo com o parágrafo anterior um conjunto de linhas de código escreve o ficheiro *opt2ef* que posteriormente é entregue ao PROAES com informação necessária para realizar o cálculo da tensão. O PROAES escreve o ficheiro *ef2opt* com os valores da tensão para cada barra. Realiza-se a leitura dos valores da tensão e define-se o constrangimento para cada barra. Uma vez que é o PROAES a realizar os cálculos o importante nestes ficheiros é entregar de forma correta a informação ao algoritmo.

4.3.4. Gradiente da Função objetivo

Os gradientes são calculados usando o método das diferenças finitas , a menos que eles sejam fornecidos por uma função através de um ficheiro M. Fica a cargo do utilizador decidir se usa o método das diferenças finitas ou entrega os gradientes calculados pelo PROAES. As derivadas por diferenças finitas, normalmente consomem mais tempo e implicam um maior custo computacional e por outro lado estão sujeitas a mais erros, por isso existem vantagens em entregar os gradientes calculados pelo PROAES, que são muito mais precisos.

Um esboço da estrutura do código para o gradiente da função objetivo, **grad_myFun.m**, é apresentado no parágrafo seguinte.

```
função gradf = grad_myFun ( x )
% o gradiente deve ser entregue como um vetor coluna em que o número
% de linhas é igual ao número de variáveis
gradf = zeros(número de variáveis, 1);
Escrita do ficheiro opt2ef  que posteriormente é lido pelo PROAES
% Cálculo da função constrangimento e derivadas recorrendo ao PROAES
resultado= PROAES('trelica_2_barras','o');
Leitura dos valores das derivadas.
% definição das derivadas do volume em relação as variáveis
gradf(i)= derivada(i);
fim
fim da função
```

De acordo com o parágrafo anterior define-se o gradiente como um vetor coluna, posteriormente um conjunto de linhas de código escreve o ficheiro *opt2ef* que é entregue ao PROAES com informação necessária para realizar o cálculo da função objetivo (que é necessário calcular novamente) e das derivadas. O PROAES escreve o ficheiro *ef2opt* com os valores calculados. Realiza-se a leitura dos valores das derivadas da tensão. Definem-se as derivadas do constrangimento.

4.3.5. Gradiente do constrangimento

O gradiente do constrangimento necessita de ser reformulado para que esteja de acordo com as regras de derivação.

De acordo com a equação 4.5:

$$\frac{d}{dx}\left(-\frac{x}{a}\right) = -\frac{1}{a} \tag{4.5}$$

No ficheiro **‘myCon’.m** o valor do constrangimento para um caso em que a tensão é positiva fica definido de acordo com a equação 4.6:

$$g(i) = \frac{(Tensão\ máxima - \ tensão\ na\ barra\ i)}{Tensão\ máxima} \tag{4.6}$$

E a sua derivada fica definida de acordo com a equação 4.7:

$$g'(i) = -\frac{1}{Tensão\ máxima} \times c'(i) \tag{4.7}$$

Por este motivo o valor da derivada do constrangimento deve ser multiplicado pela equação 4.8:

$$-\frac{1}{Tensão\ máxima} \tag{4.8}$$

para obter *g’*(*i*).

Um esboço da estrutura do código para gradiente do constrangimento, **grad_myCon.m**, é apresentado no paragrafo seguinte.

```
função gradg = grad_myCon ( x )
% o valor do gradiente dos constrangimentos deve ser entregue como
% uma matriz em que o número de linhas é igual ao número de
% constrangimentos e o número de colunas é igual ao número de
% variáveis
gradg = (nº de constrangimentos, número de variáveis)
Escrita do ficheiro opt2ef  que posteriormente é lido pelo PROAES
% Cálculo da função constrangimento e derivadas recorrendo ao PROAES
resultado= PROAES('trelica_2_barras','o');
Leitura dos valores das derivadas g’.
% definição das derivadas das tensões em relação as variáveis
para j=1:número de constrangimentos
    para i=1:número de variáveis
            se tensão da barra (i) >= 0
                gradg(j,i)= -(valor da derivada)/tensão máxima;
            ou
                gradg(j,i)=  (valor da derivada)/tensão máxima;
            fim
    fim
fim
fim da função
```

De acordo com o parágrafo anterior, define-se o gradiente dos constrangimentos como uma matriz em que o número de linhas é igual ao número de constrangimentos e o número de colunas é igual ao número de variáveis. Posteriormente um conjunto de linhas de código escreve o ficheiro *opt2ef* que é entregue ao PROAES com informação necessária para realizar o cálculo da tensão e das derivadas da tensão. O PROAES escreve o ficheiro *ef2opt* com esses valores calculados. Realiza-se a leitura dos valores das derivadas da tensão. Definem-se as derivadas dos constrangimentos.

4.3.6. Execução da Otimização

Depois de todos os ficheiros do tipo M construídos a colaboração entre os OCTAVE e o PROAES devera funcionar de acordo com o diagrama da Figura 4.2. Para executar a otimização o utilizador devera reunir os seguintes ficheiros na diretoria corrente do OCTAVE:

- **inicializacao.m**
- **myFun.m**
- **myCon.m**
- **grad_myFun.m** (opcional)
- **grad_myCon.m** (opcional)
- **problema.inp**

De seguida o utilizador deve abrir o ficheiro **inicializacao.m** no editor do OCTAVE, onde irá aparecer parte da função de inicialização com parte igual a apresentada na Figura 4.3. Antes de executar a otimização devem ser definidos os seguintes parâmetros:

1- Escolher 0 ou 1 se pretende que o gradiente seja calculado por diferenças finitas ou calculado pelo PROAES respetivamente.
2- O ponto de partida das variáveis de otimização.
3- O *Lower Bound* das variáveis.
4- O *Upper Bound* das variáveis.
5- O número máximo de iterações.
6- Qual a tolerância para a convergência

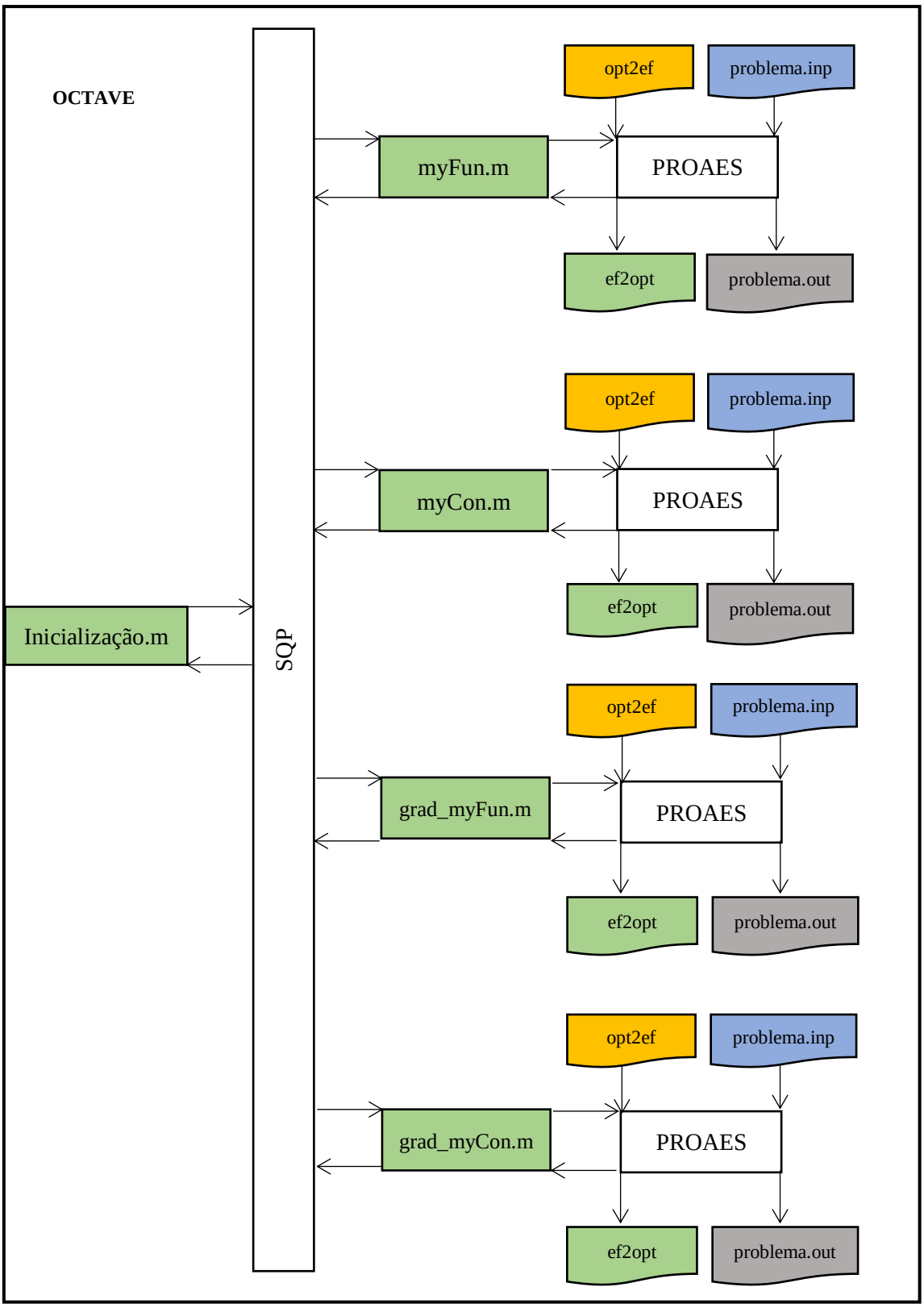

Figura 4.2 - Diagrama do Processo de Otimização SQP

```
% GRAD= 0/1, SQP calcula o gradiente/  é fornecido
GRAD= 1;
% Valores das variáveis na iteração = 0
x0 = 0.03*ones(2,1);
%
if GRAD
   ObjectiveFunction = cell(2,1);
   % Função objetivo
   ObjectiveFunction(1)= @myFun;
   % Gradiente da função objetivo
   ObjectiveFunction(2)= @grad_myFun;
   IneqConsFunction = cell(2,1);
   % Função constrangimento
   IneqConsFunction(1)= @myCon;
   % Gradiente da função constrangimento
   IneqConsFunction(2)= @grad_myCon;
else
   % Função objetivo
   ObjectiveFunction= @myFun;
   % Lembrete 1: a função do constrangimento deve devolver um vetor coluna, i.e.,
   % c = [ ; ... ; ]
   % Lembrete 2: o formato do constrangimento de desigualdade é g >= 0, i.e.,
   % constrangimento do tipo "maior ou igual" a
   IneqConsFunction= @myCon;
end
% Lower and upper bounds - Limite superior e inferior das variáveis
LB=1e-8*ones(2,1);
UB=0.03*ones(2,1);
% Número máximo de iterações; use [] para um valor padrão
maxiter=8;
% Tolerância à convergência; use [] para um valor padrão
tol=;1e-12;
```


Figura 4.3 - Ficheiro de inicialização do problema SQP

4.4. Método das Assíntotas Móveis (MMA)

Para resolver o problema de otimização *Treliça 2 Barras* o utilizador deverá realizar os seguintes passos:

• Formulação do Problema

• Definir o Problema de acordo com a sintaxe do algoritmo, o que implica:

- Definir a função objetivo e Gradiente, em linguagem OCTAVE, num único ficheiro M

- Definir o constrangimento e Gradiente, em linguagem OCTAVE, num único ficheiro M

• Executar a otimização

4.4.1. Formulação do Problema

Considere a formulação matemática de um problema de acordo com o MMA apresentada na equação 2.23.

Considere agora o Problema *Treliça 2 Barras*. O utilizador devera escrever a função objetivo e os constrangimentos recorrendo a ficheiros M do OCTAVE usando expressões que reflitam a formulação usada pelo algoritmo.

4.4.2. Função objetivo e Gradiente

O utilizador deve definir a função objetivo juntamente com o seu gradiente em linguagem OCTAVE, num ficheiro M.

De maneira semelhante ao que acontece para o SQP é o PROAES que vai realizar o cálculo da função objetivo e o cálculo do gradiente da função objetivo. Também neste caso é necessário escrever ficheiros que permitam a comunicação entre o PROAES e o OCTAVE.

No parágrafo seguinte é apresentado um Pseudocódigo **myFun.m** que mostra como calcular a função objetivo e o seu gradiente com a ajuda do PROAES. Para que o utilizador perceba melhor a forma como deve construir os ficheiros a entregar ao PROAES deve consultar a dissertação de Marmeleiro [6].

```
Função [f0val, df0dx] = myFun (x)
n = número de variáveis
% o gradiente deve ser entregue como um vetor coluna em que o número
% de linhas é igual ao número de variáveis
df0dx = zeros(n,1);
Escrita do ficheiro opt2ef  que posteriormente é lido pelo PROAES
%Cálculo da função objetivo e das sensibilidades recorrendo ao PROAES
resultado= PROAES('trelica_2_barras','o');
Leitura do ficheiro ef2opt (que contem o valor da função objetivo e
das sensibilidades calculados pelo PROAES).
% definição do valor da função objetivo
f0val= Volume;
% definição do Gradiente
para i = 1 : nº de variáveis
      df0dx(i) = sensibilidade;
fim
fim % Fim do ficheiro
```

De acordo com o pseudocódigo apresentado no parágrafo anterior, um conjunto de linhas de código escreve o ficheiro *opt2ef* que posteriormente é entregue ao PROAES com informação necessária para realizar o cálculo da função objetivo e das sensibilidades. O PROAES escreve o

ficheiro *ef2opt* com os valores da função objetivo e sensibilidades. Realiza-se a leitura dos valores. Define-se *f* igual ao valor do volume e é criado um ciclo para definir o gradiente.

4.4.3. Constrangimento e Gradiente

Para este caso também deve ser criado um ficheiro M que defina os constrangimentos e os respetivos gradientes. Cada constrangimento de desigualdade deve ser formulado para que esteja na forma $f_i(x) - a_i z - y_i \leq 0$. Para o exemplo *Treliça 2 Barras* deve então ser reformulado para $c_j(x) - 100 \times 10^6 \leq 0$ onde c_j é a tensão na barra j e assim ficar acordo com a sintaxe do algoritmo. Um Pseudocódigo para a função constrangimento, **myCon.m**, onde também é feita a normalização, é apresentado no parágrafo seguinte

.

```
Função [fval, dfdx] = myCon(x)
n =nº de variáveis;
nc= n de constrangimentos;
% o constrangimento deve ser entregue como um vetor coluna em que o
% número de linhas é igual ao número de constrangimentos
fval = zeros(nc,1);
% o gradiente deve ser entregue como uma matriz em que o número de
% linhas é igual o número de constrangimentos e o número de colunas
% igual ao número de variáveis
dfdx = zeros(nc,n);
Escrita do ficheiro opt2ef  que posteriormente é lido pelo PROAES
% Cálculo do constrangimento recorrendo ao PROAES
resultado= PROAES('trelica_2_barras','o');
Leitura do ficheiro ef2opt que contem os valores calculados.
% Definição do valor do constrangimento
para i= 1:número de variáveis
     se tensão da barra (i) >= 0
         fval(i)=(tensão da barra i–Tensão máxima)/Tensão máxima;
     ou
         fval(i)=(-tensão da barra i–Tensão máxima)/Tensão máxima;
     fim
fim
% Definição do gradiente
para j= 1:número de constrangimentos
    para i= 1:número de variáveis
            se tensão da barra(i) >= 0
                dfdx(j,i)=  (valor da derivada)/Tensão máxima;
            ou
                dfdx(j,i)= -(valor da derivada)/Tensão máxima;
            se
    fim
fim
Fim do ficheiro
```

Sempre que o valor da tensão na barra *j* for inferior a 100×10^6 o valor do constrangimento fica negativo, o que indica que está a ser respeitado, sempre que o valor da tensão na barra é superior a 100×10^6 o constrangimento é positivo o que indica que está a ser violado. O valor 100×10^6 é a tensão admissível para este problema.

De acordo com Pseudocódigo **myCon.m** um conjunto de linhas de código escreve o ficheiro *opt2ef* que posteriormente é entregue ao PROAES com informação necessária para realizar o cálculo da tensão e das sensibilidades. O PROAES escreve o ficheiro *ef2opt* com os valores calculados. Realiza-se a leitura dos valores da tensão e das sensibilidades. Define-se um ciclo para construir o vetor do constrangimento *fval*, e por fim define-se um ciclo para definir a matriz do gradiente *dfdx* respeitando as regras das derivadas referidas na equação 4.5.

4.4.4. Execução da Otimização

Depois de todos os ficheiros do tipo M construídos, a colaboração entre os OCTAVE e o PROAES devera funcionar de acordo com o diagrama da Figura 4.4. Para executar a otimização, o utilizador devera reunir os seguintes ficheiros na diretoria corrente do OCTAVE:

- **inicializacao.m**
- **myFun.m**
- my**Con.m**
- **problema.inp**

De seguida o utilizador deve abrir o ficheiro **inicializacao.m** no editor do OCTAVE, onde ira aparecer uma função igual à apresentada na Figura 4.5.

Antes de executar a otimização devem ser definidos os seguintes parâmetros do problema:

1- O número de variáveis .
2- O número de constrangimentos.
3- O número máximo de iterações.
4- Escolher o ponto de partida das variáveis de otimização.
5- O *Lower Bound* das variáveis.
6- O *Upper Bound* das variáveis.
7- A tolerância para a convergência

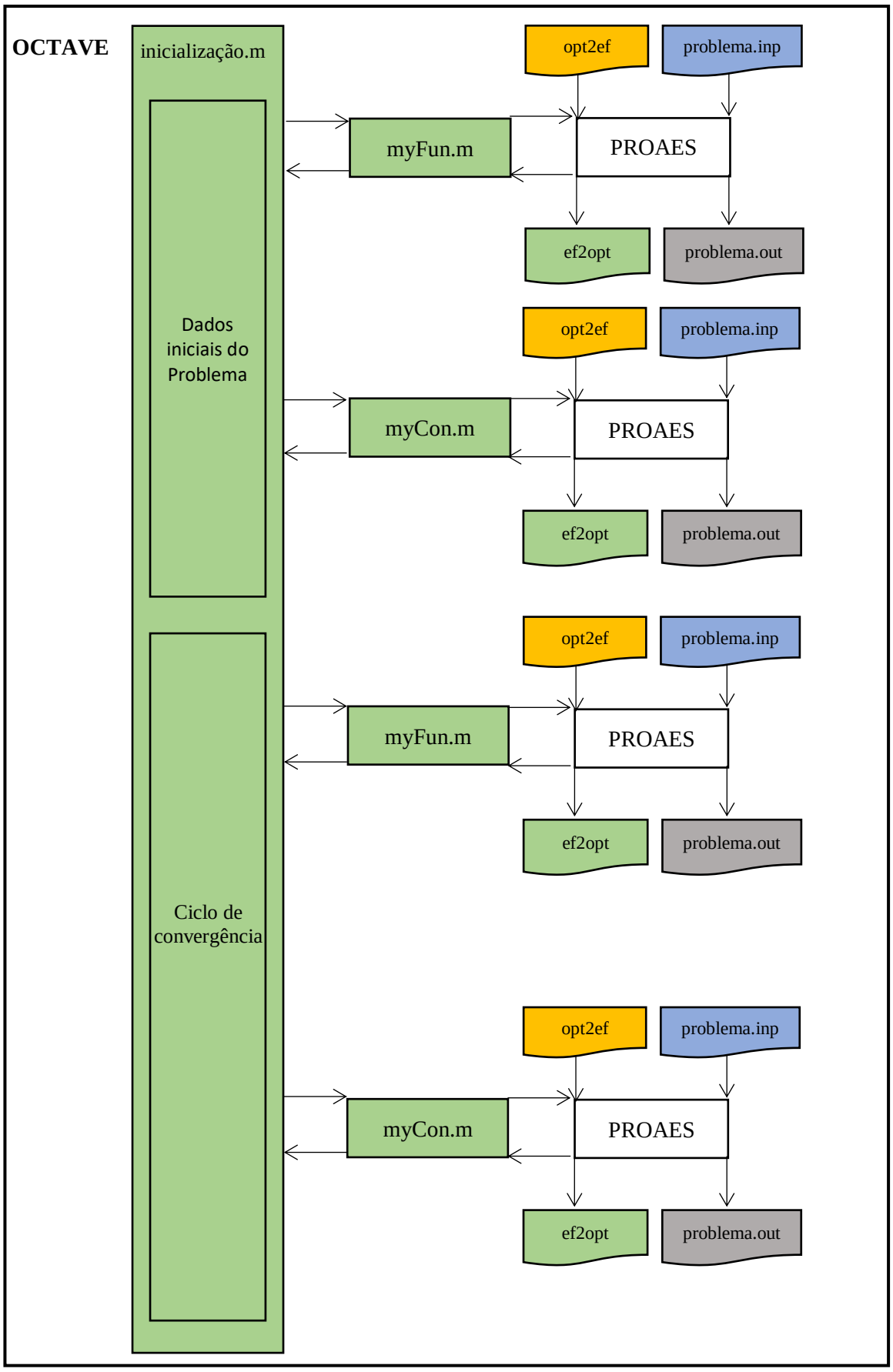

Figura 4.4 - Diagrama do Processo de Otimização MMA

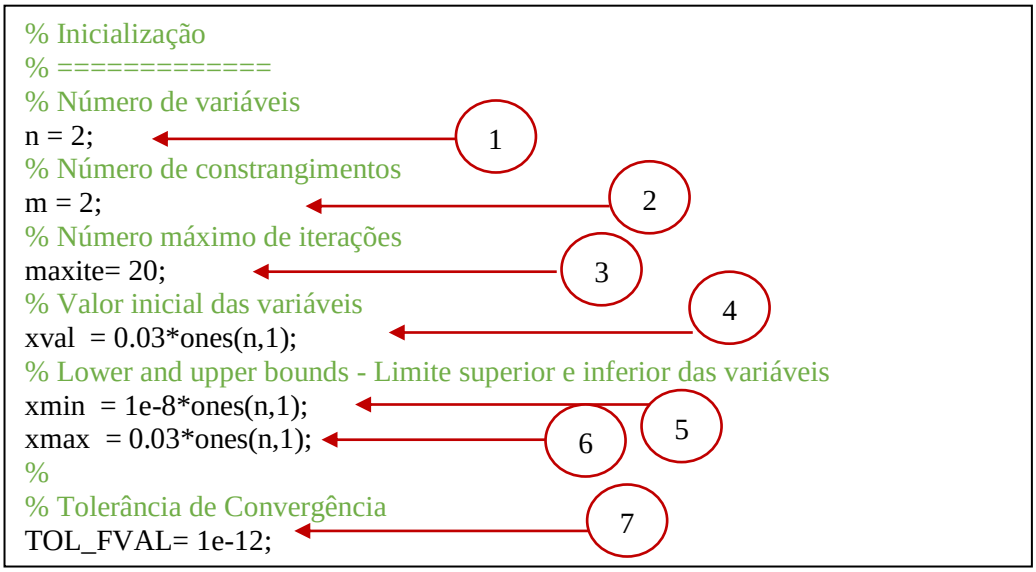

Figura 4.5 - Parte superior do ficheiro de inicialização do problema

4.5. Algoritmo Genético Real

Para resolver o problema de otimização *Treliça 2 Barras* com o algoritmo genético de genes reais, o utilizador deverá realizar os seguintes passos:

• Formular do Problema

• Escrever um ficheiro M em linguagem OCTAVE, definindo a função objetivo e a penalização, que é função do constrangimento

• Executar a otimização

4.5.1. Formulação do Problema

A formulação matemática do problema que vai ser resolvido recorrendo ao AG REAL necessita de ser substancialmente alterada, pois não segue a equação 2.1 mas sim a formulação de um problema de otimização sem constrangimentos.

$$F(x) = \begin{cases} f(x) & se\ a\ solução\ é\ admissivel \\ f(x) + CP \times VP & se\ a\ solução\ é\ inadmissivel \end{cases} \qquad (4.10)$$
$$Sujeito\ a\ , \quad x_j^{min} \leq x_j \leq x_j^{max} \quad ;i = 1, \dots, n$$

Onde $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ é o vetor das n variáveis de projeto, $f(\boldsymbol{x})$ é a função objetivo, CP é a constante de penalização do problema, VP é a violação verificada no constrangimento e x_j^{min} e x_j^{max} são os limites inferior e superior de variação das variáveis x_j.

Considere agora o Problema *Treliça 2 Barras*. O utilizador deverá escrever a função objetivo, recorrendo a um ficheiro M do OCTAVE e usando expressões que reflitam a forma matemática do algoritmo.

4.5.2. Função objetivo

O utilizador deve definir a função objetivo em linguagem OCTAVE, num ficheiro M. Deve no mesmo ficheiro calcular o constrangimento, pois é necessário para calcular a penalização da função objetivo.

Um esboço para a construção do ficheiro **myFun,m** é apresentado no pseudocódigo em completo na página seguinte.

O espaço que se segue foi deixado propositadamente em branco.

```
Função f = myFun(x)
n = nº de variáveis;
nc = nº de constrangimentos;
% definição do valor máximo de tensão
sxmax = 100e6;
% Constante de penalização que deve ser da mesma ordem de grandeza que
% a função objetivo
C=0.2
Escrita do ficheiro opt2ef  que posteriormente é lido pelo PROAES
% Cálculo da função objetivo e do constrangimento recorrendo ao PROAES
resultado= PROAES('trelica_2_barras','o');
Leitura do ficheiro ef2opt que contem o valor do constrangimento.
% Definição do valor do constrangimento
para i = 1:número de variáveis
     sx(i)=valor da tensão;
fim
Leitura do ficheiro ef2opt que contem o valor da função objetivo
% Definição da função objetivo
Vol = volume;
% Cálculo do valor de penalização
VP=0;
para i = 1 até nº de variáveis
      se sx(i)>sxmax
            VP = VP + (sx(i)-sxmax)/sxmax;
      Fim
Fim
% Função objetivo penalizada
f = Vol + CP * VP;
% Print da função objetivo
tensao= max(sx);
se tensao > sxmax
   % Faz o print de solução inadmissível
   fprintf('obj= %10.4e, v= %8.6f, smax= %10.4e, %9.7f, %9.7f, ...
   VIOLADO\n',f,Vol,tensao,x(1),x(2);
ou
   % Faz o print de solução admissível
   fprintf('obj= %10.4e, v= %8.6f, smax= %10.4e, %9.7f, %9.7f, \n',...
   f,Vol,tensao,x(1),x(2);
fim
Fim do ficheiro
```

De acordo como o parágrafo anterior um conjunto de linhas de código escreve o ficheiro *opt2ef* que posteriormente é entregue ao PROAES com informação necessária para realizar o cálculo do volume e das tensões. O PROAES escreve o ficheiro *ef2opt* com os valores calculados. Realiza-se a leitura dos valores do volume e da tensão. Define-se o valor de penalização a aplicar à função objetivo através de um ciclo e aplica-se a penalização. Calcula-se o valor final da função objetivo.

Por último são impressos os valores da função objetivo, do volume, o deslocamento e as variáveis para que o utilizador possa analisar e ver quais os valores que a otimização esta a tomar. Se por exemplo os valores da otimização forem todos inadmissíveis, violados, deve alterar-se o valor da constante de penalização. Uma maneira é começar por valores baixos da constante de penalização e ir aumentado, à medida que se vão correndo otimizações, ate os valores se tornarem admissíveis. É um processo moroso e pode levar algum tempo até se acertar com a constante.

4.5.3. Execução da Otimização

Depois de todos os ficheiros do tipo M construídos a colaboração entre os OCTAVE e o PROAES devera funcionar de acordo com o diagrama da Figura 4.6. Para executar a otimização o utilizador devera reunir os seguintes ficheiros na diretoria corrente do OCTAVE:

- **inicializacao.m**
- **myFun.m**
- **problema.inp**

De seguida o utilizador deve abrir o ficheiro **inicializacao.m** no editor do OCTAVE, onde ira aparecer uma função igual a apresentada na Figura 4.7.

Antes de executar a otimização devem ser definidos os seguintes parâmetros:

1- O número de variáveis .
2- O *Lower Bound* das variáveis.
3- O *Upper Bound* das variáveis.
4- O número máximo de gerações
5- A dimensão da população
6- Qual o tipo de seleção

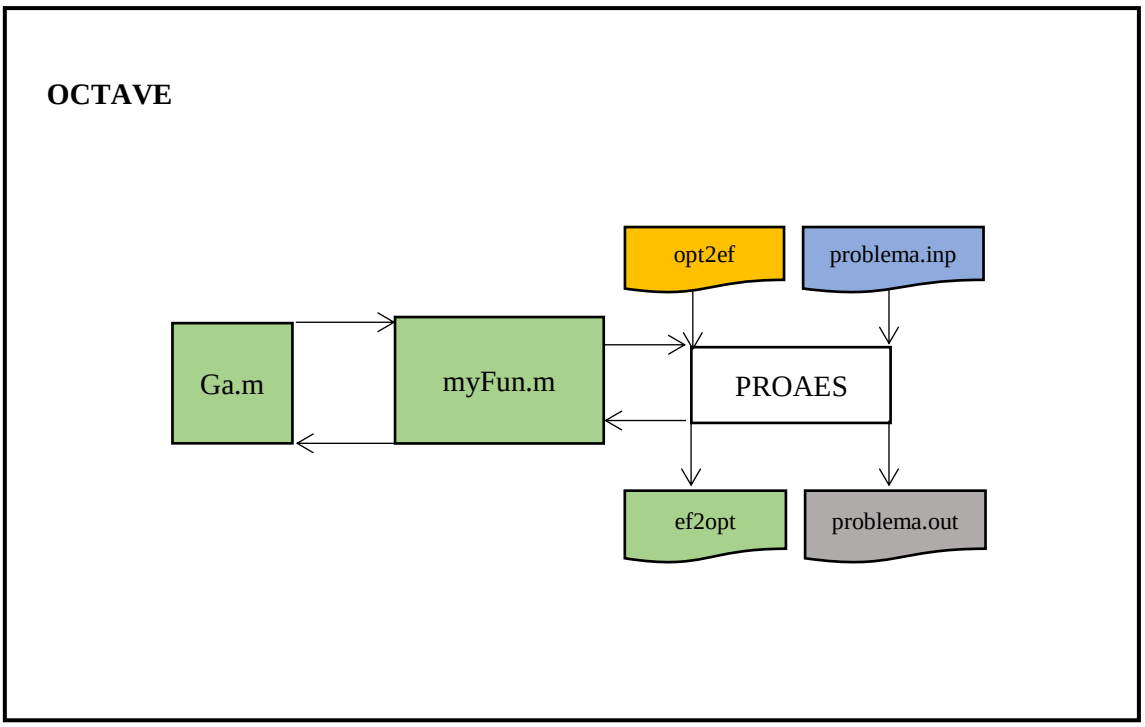

Figura 4.6 - Diagrama do Processo de Otimização

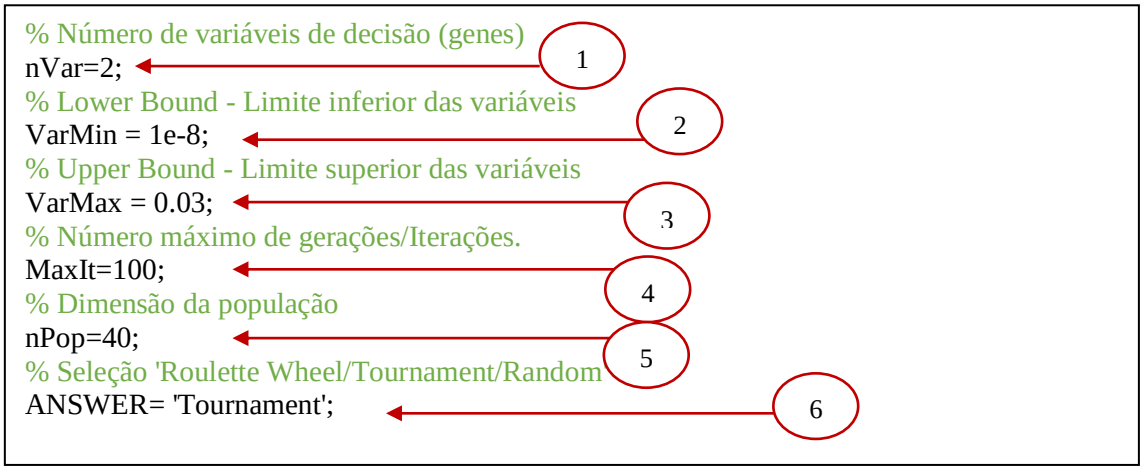

Figura 4.7 - Ficheiro de inicialização do problema

Concluídos os ficheiros, devem ser guardados. Para correr a otimização o ficheiro **Ga.m** deve ser executado.

4.6. Algoritmo Genético Binário

Para resolver o problema de otimização *Treliça 2 Barras* com o algoritmo genético de genes binários o utilizador deverá realizar os seguintes passos:

• Formulação do Problema

• Escrever um ficheiro M em linguagem OCTAVE, definindo a função objetivo e a penalização, que é função do constrangimento

• Executar a otimização

4.6.1. Formulação do Problema

A formulação matemática de um problema recorrendo ao AG BINÁRIO é igual a do AG REAL. A única diferença está ao nível das variáveis.

4.6.2. Função objetivo

A função objetivo do AG REAL permanece igual para o AG BINÁRIO, apenas é necessário colocar informação em forma de tabela sobre as variáveis e realizar a sua descodificação.

Um esboço para a construção do ficheiro **myFun,m** é apresentado no pseudocódigo em completo na página seguinte.

O espaço que se segue foi deixado propositadamente em branco,

```
Função f = myFun(x)
n = nº de variáveis;
% Tabela com informação das variáveis
Tabela= [ 1.06e-2;...     %HEB240
1.18e-2; ...              %HEB160
1.31e-2; ...              %HEB200
1.49e-2; ...              %HEB220
1.61e-2; ...              %HEB240
1.71e-2; ...              %HEB260
1.81e-2; ...              %HEB280
1.98e-2; ...              %HEB300
2.18e-2; ...              %HEB320
2.39e-2; ...              %HEB340
2.54e-2; ...              %HEB360
2.70e-2; ...              %HEB400
2.86e-2; ...              %HEB450
3.06e-2; ...              %HEB500
3.21e-2; ...              %HEB650
3.34e-2]; ...             %HEB550
% definição do valor máximo de tensão
sxmax = 100e6;
% Constante de penalização que deve ser da mesma ordem de grandeza que
% a função objetivo
C=0.2
Escrita do ficheiro opt2ef  que posteriormente é lido pelo PROAES
ii=zeros(n,1);
para i=1:n
% Descodifica cada segmento do cromossoma binário em um nº inteiro
    indice=1+x((i-1)*4+1)*8+x((i-1)*4+2)*4+x((i-1)*4+3)*2+x(…
(i-1)*4+4);
% Guarda num vetor o inteiro que representa o nº de posição do perfil
% na tabela
    ii(i)=indice;
    fprintf(fout,'%d, %18.12e\n', i, Tabela(indice));
end
% Cálculo da função objetivo e do constrangimento recorrendo ao PROAES

(…)

% O restante do código é igual ao AG REAL
Fim do ficheiro
```

De acordo com o paragrafo anterior. A grande diferença entre a função **myFun.m** de AG REAL e do AG BINÁRIO reside no início do código em que as variáveis são entregues através de uma tabela e a escolha delas é feita pela descodificação das variáveis binarias.

4.6.3. Execução da Otimização

Depois de todos os ficheiros do tipo M construídos a colaboração entre os OCTAVE e o PROAES devera funcionar de acordo com o diagrama da Figura 4.6. Para executar a otimização o utilizador devera reunir os seguintes ficheiros na diretoria corrente do OCTAVE:

- **inicializacao.m**
- **myFun.m**
- **problema.inp**

De seguida o utilizador deve abrir o ficheiro **inicializacao.m** no editor do OCTAVE, onde irá aparecer uma função igual a apresentada na Figura 4.8.

Antes de executar a otimização devem ser definidos os seguintes parâmetros:

1- O número de variáveis.
2- O número de variáveis de projeto codificadas.
3- O número de variáveis de decisão.
4- O número máximo de iterações.
5- O Tamanho da população.
6- Qual o tipo de seleção.

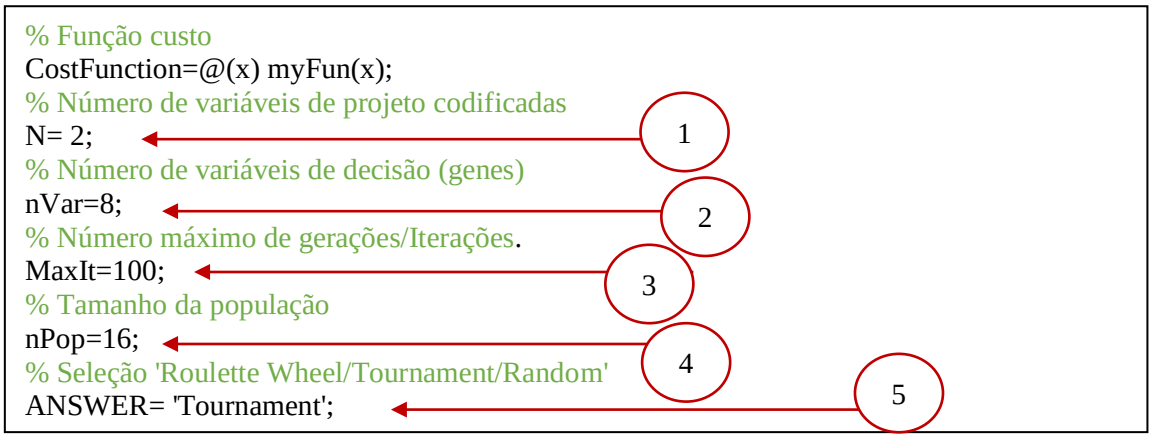

Figura 4.8 - Ficheiro de inicialização do problema

Concluídos, os ficheiros devem ser guardados. Para correr a otimização o ficheiro **ga_real_coded.m** deve ser executado.

4.7. Otimização por Enxame de Partículas (PSO)

Para resolver o problema de otimização *Treliça 2 Barras* com este algoritmo o utilizador deverá realizar os seguintes passos:

- Formulação do Problema

• Escrever um ficheiro M em linguagem OCTAVE, definindo a função objetivo e a penalização, que é função do constrangimento.

• Executar a otimização

A Formulação do Problema e a Função objetivo seguem as mesmas instruções que as do AG REAL, pelo que o utilizador deve consultar o subcapítulo 4.5.1 e 4.5.2.

4.7.1. Execução da Otimização

Depois de todos os ficheiros do tipo M construídos a colaboração entre os programas OCTAVE o PROAES deverá funcionar de acordo com o diagrama da Figura 4.9. Para executar a otimização o utilizador deverá reunir os seguintes ficheiros na diretoria corrente do OCTAVE:

- **inicializacao.m**
- **myFun.m**
- **problema.inp**

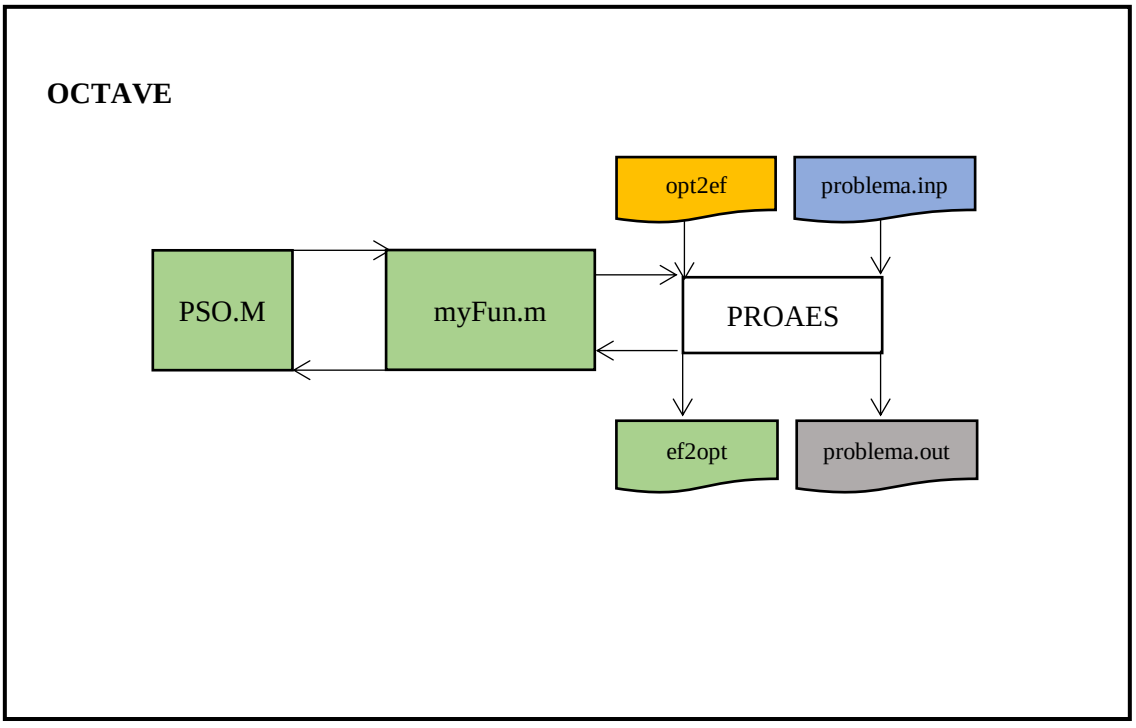

Figura 4.9 - Diagrama do Processo de Otimização PSO

De seguida o utilizador deve abrir o ficheiro **PSO.m** no editor do OCTAVE, onde ira aparecer uma função igual a apresentada na Figura 4.10.

Antes de executar a otimização devem ser definidos os seguintes parâmetros:

1- O número de variáveis .
2- O *Lower Bound* das variáveis.
3- O *Upper Bound* das variáveis.
4- O número máximo de iterações
5- O número da população

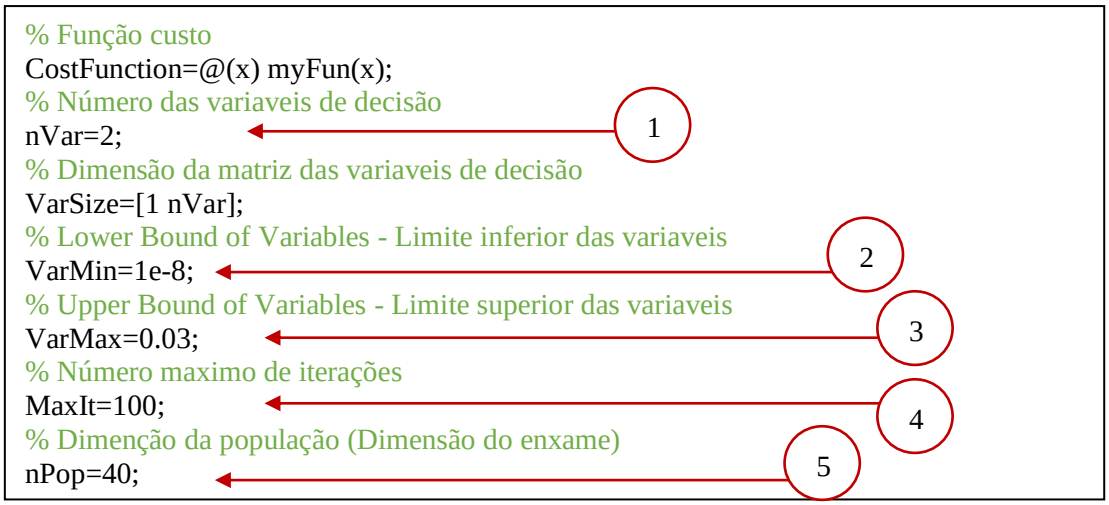

Figura 4.10 - Ficheiro de inicialização do problema

Concluídos, os ficheiros devem ser guardados. Para executar a otimização o ficheiro **pso.m** deve ser executado.

4.8. Resumo dos Ficheiros da *Toolbox* de Otimização

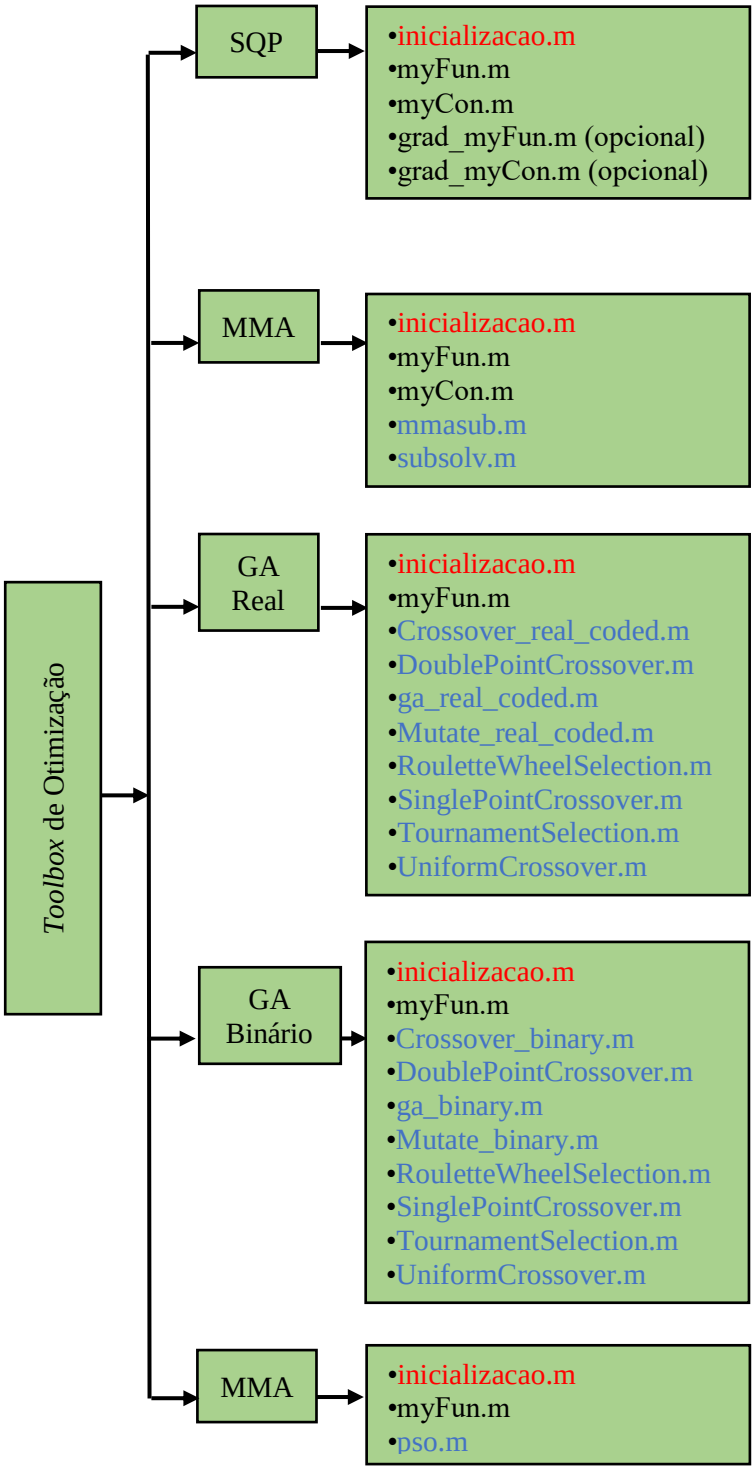

Figura 4.11 - Ficheiros que compõem a *toolbox*

De acordo com o diagrama da figura se o utilizador quiser resolver um problema de otimização recorrendo a um dos algoritmos da *toolbox* o que deve fazer? A vermelho estão representados os ficheiros que servem de interface entre o algoritmo e o utilizador, nestes devem ser colocados os parâmetros do problema. A preto estão representados os ficheiros que permitem a comunicação entre o algoritmo e o PROAES, estes ficheiros devem ser contruídos pelo utilizador. A azul estão representadas as funções que contêm o código de cada algoritmo, não existindo necessidade de qualquer ação por parte do utilizador sobre estes.

5. Exemplos de Verificação

Com o objetivo de avaliar os resultados obtidos pelos algoritmos adicionados à *toolbox* foram efetuadas comparações com os valores obtidos utilizando a OPTIMTOOL do MATLAB para o exemplo 1 e 2. Os resultados do exemplo 3 são comparados com os publicados em [31].

Foram escolhidos 3 exemplos:

1. O exemplo 1 foi retirado de [2] .
2. O exemplo 2 é apresentado na dissertação do PROAES por Marmeleiro [6].
3. O exemplo 3 é apresentado no artigo [31].

Os exemplos foram executados recorrendo a um computador com um processador i7 8850U 1.80GHz 1.99GHz.

5.1. Exemplo 1 – Treliça 11 Barras

A estrutura do problema em estudo está ilustrada na Figura 5.1.

Dados do Problema:

Tabela 5.1 - Valores iniciais das variáveis e do Volume

Área inicial do 11 Elementos de barra	$A = 0.0334\ m^2$
Volume inicial da Estrutura	$\mathrm{V} = 0.5407167\ m^3$
Densidade	$\rho = 2710\ kg/m^3$
Modulo de Young	$E = 70\ GPa$
Tipo de perfil	$HEB\ 800$
Carregamento Aplicado	$P = 513846\ N$
Tensão Admissível	$100\ MPa$
Número de Elementos	11
Número de nós	7

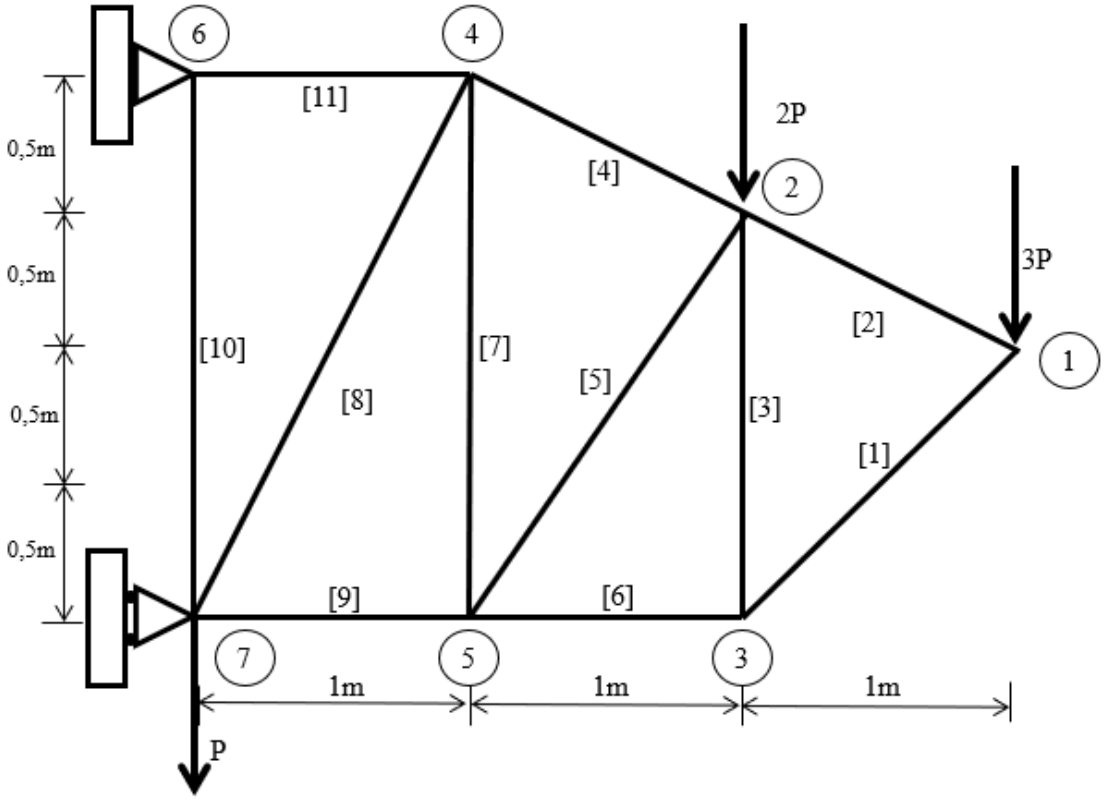

Figura 5.1 - Treliça 11 barras do exemplo 1 [2]

Formulação do Problema de Otimização Dimensional

• Função objetivo

- Minimizar o volume da estrutura.

•Variáveis do problema

- A área da secção transversal de cada barra.

• Constrangimentos do problema

- A tensão admissível de cada barra não deve ser ultrapassada.

$$\sigma \leq 100\ MPa \tag{5.1}$$

- A área da secção transversal de cada barra está limitada superior e inferiormente.

$$1 \times 10^{-8} < A_i < 3.34 \times 10^{-4}\ m^2 \tag{5.2}$$

O problema de otimização fica com a seguinte forma,

$$\begin{aligned} \min_{\mathbf{x}} \quad & Volume \\ s.a. \quad & \sigma_j \leq 100\ MPa \qquad j = 1, \ldots, 11 \\ & 1 \times 10^{-8} < x_i < 3.34 \times 10^{-4}\ m^2 \qquad i = 1, \ldots, 11 \end{aligned} \tag{5.3}$$

5.1.1. Resultados obtidos do Algoritmo SQP

O valor da função objetivo e as variáveis são apresentados na Tabela 5.2, sendo comparados com os resultados provenientes do Programa MATLAB.

SQP* - Sensibilidades Analíticas.

SQP' – Sensibilidades por diferenças finitas.

Tabela 5.2 - Resultados da otimização com o algoritmo SQP

ELEMENTOS	SQP (MATLAB) $[m^2]$	SQP* (OCTAVE) $[m^2]$	SQP' (OCTAVE) $[m^2]$	ERRO SQP(MATLAB) SQP*(OCTAVE) (%)	ERRO SQP(MATLAB) SQP'(OCTAVE) (%)
1	0.0145337	0.0145337	0.0145337	0	0
2	0.0114899	0.0114899	0.0114899	0	0
3	0.0102769	0.0102769	0.0102769	0	0
4	0.0229798	0.0229798	0.0229798	0	0
5	0.0185269	0.0185269	0.0185269	0	0
6	0.0102769	0.0102769	0.0102769	0	0
7	0.0154153	0.0154153	0.0154153	0	0
8	0.0287248	0.0287248	0.0287248	0	0
9	0.0205538	0.0205538	0.0205538	0	0
10	0.0308307	0.0308307	0.0308307	0	0
11	0.0334000	0.0334000	0.0334000	0	0
Valor da Função Objetivo m^3	0.328862	0.328862	0.328862	0	0
Nº de Iterações	7	7	6	-	-
Tempo Computacional	-	3.302 s	7.465 s	-	-

A otimização utilizando o gradiente calculado pelo PROAES é mais rápida do que utilizando o método das diferenças finitas disponibilizado pelo algoritmo SQP. Os graficos das áreas otimizadas obtidos recorrendo ao SQP(OCTAVE) são iguais às obtidas pelo SQP(MATLAB), como tal a estrutura otimizada que se obeteve foi a mesma (ver Figura 5.2 e Figura 5.3).Conclui-se assim que o SQP do OCTAVE permite obter resultados admissíveis sendo o valor do erro igual a 0% quando comparado com o SQP MATLAB.

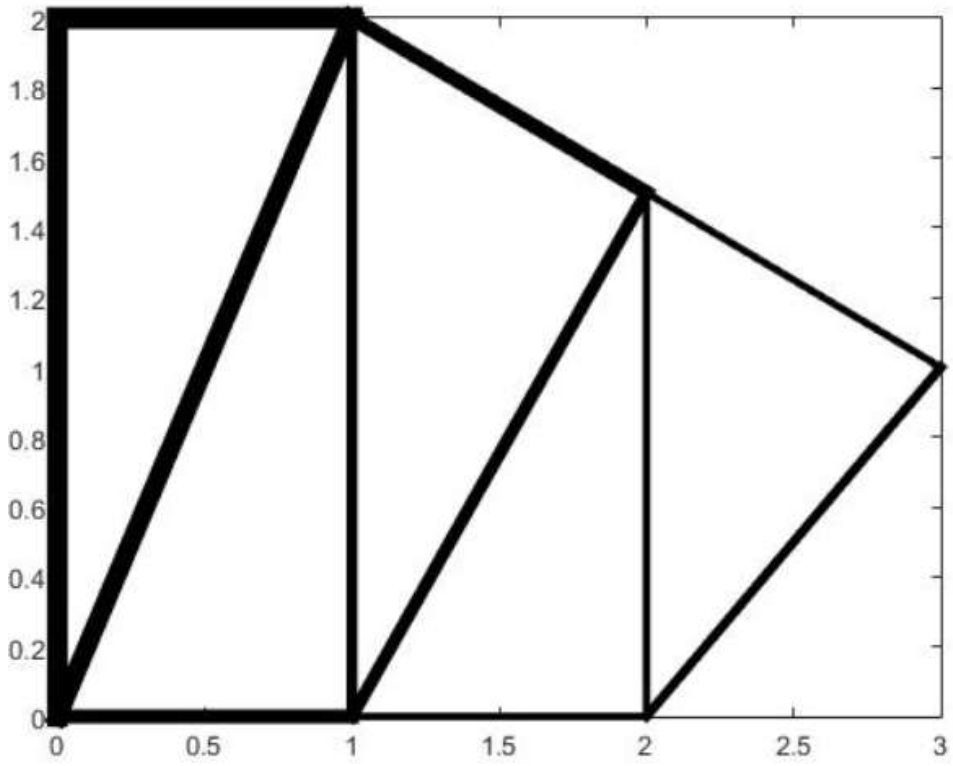

Figura 5.2 - Treliça 11 barras otimizada pelo SQP MATLAB

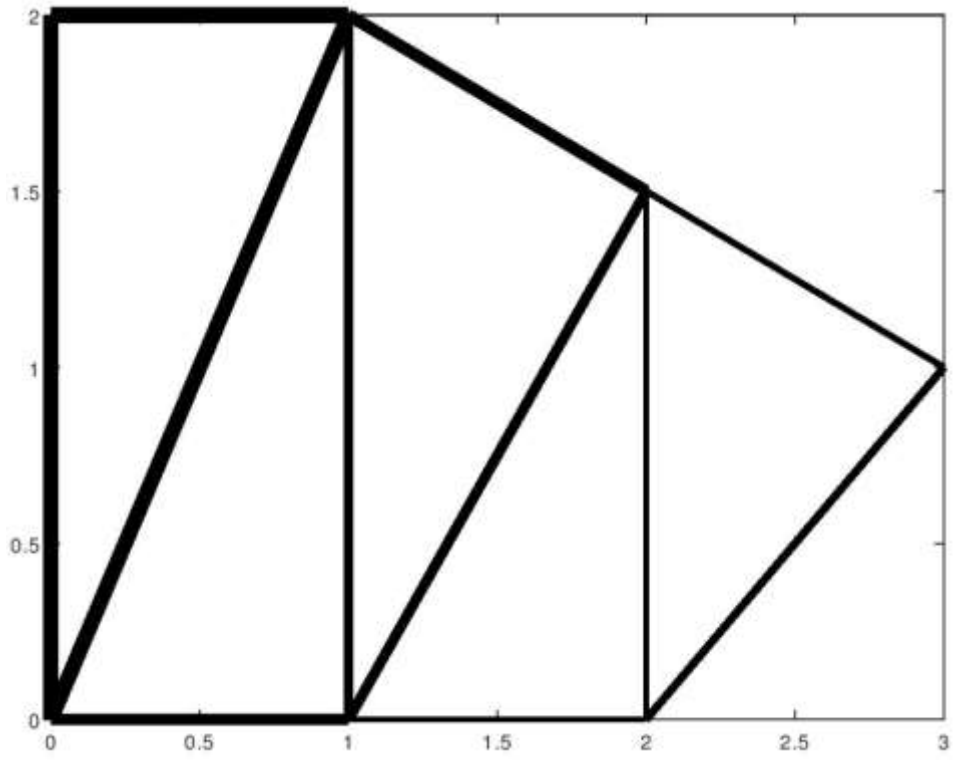

Figura 5.3 - Treliça 11 barras otimizada pelo SQP OCTAVE

5.1.2. Resultados obtidos do Algoritmo MMA

Os valores da função objetivo e das variáveis otimizadas são apresentados na Tabela 5.3 e são comparados com os resultados provenientes da otimização com o Programa MATLAB.

Tabela 5.3 - Resultados da otimização com o algoritmo MMA

ELEMENTOS	SQP(MATLAB) [m^2]	MMA(OCTAVE) [m^2]	ERRO (%)
1	0.0145337	0.0145337	0
2	0.0114899	0.0114899	0
3	0.0102769	0.0102769	0
4	0.0229798	0.0229798	0
5	0.0185269	0.0185269	0
6	0.0102769	0.0102769	0
7	0.0154153	0.0154153	0
8	0.0287248	0.0287248	0
9	0.0205538	0.0205538	0
10	0.0308307	0.0308307	0
11	0.0334000	0.0334000	0
Valor da Função Objetivo	0.328862 m^3	0.328862 m^3	0
Nº de Iterações	7	6	-
Tempo Computacional	-	1.249 s	-

Os valores obtidos recorrendo ao algoritmo de otimização MMA(OCTAVE) têm um erro igua a 0% quando comparados com os obtidos pelo SQP(MATLAB). O que indica que o algoritmo atinge resultados admissíveis. A estrutura com as barras otimizadas é apresentada na Figura 5.4.

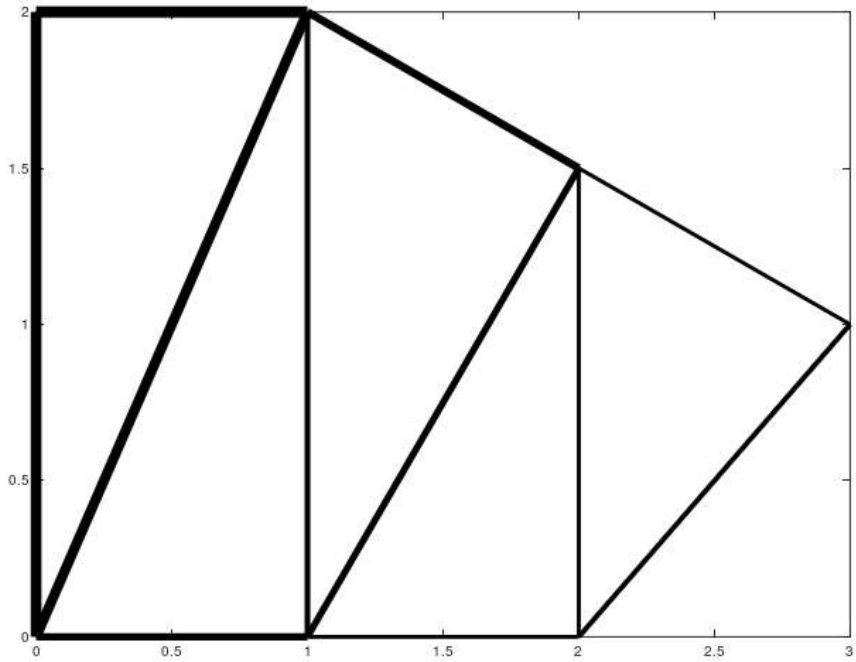

Figura 5.4 - Treliça 11 barras otimizada pelo MMA

Na Figura 5.5 é apresentado o grafico de convergência do algoritmo MMA. Foi dado ao algoritmo um limite de 20 iterações mas acabou por convergir na sexta iteração.

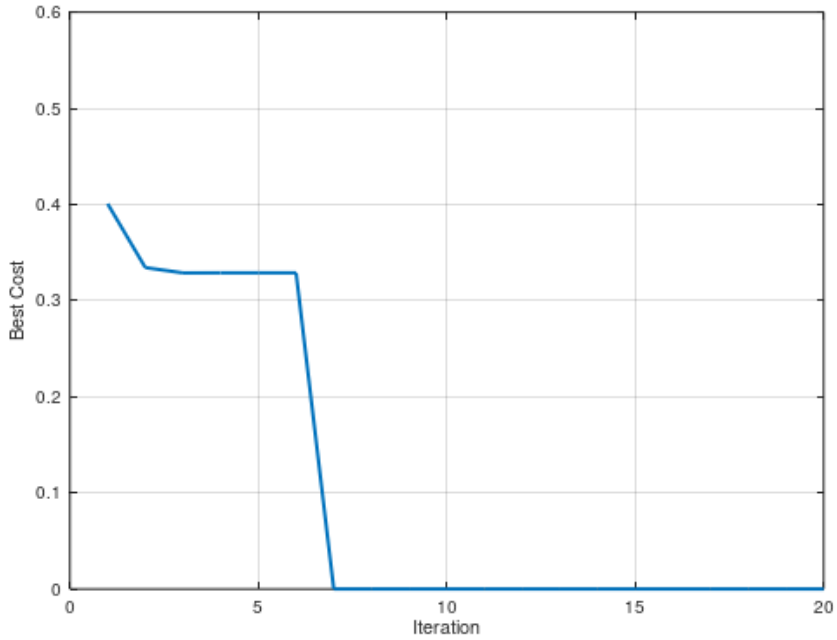

Figura 5.5 - Gráfico de convergência do algoritmo MMA OCTAVE

5.1.3. Resultados obtidos do Algoritmo Genético Real

O valor da função objetivo e as variáveis otimizadas pelo Algoritmo são apresentados na Tabela 5.4, e são comparados com os resultados provenientes da otimização com o Programa MATLAB.

Tabela 5.4 - Resultados da otimização com o algoritmo GA OCTAVE

ELEMENTO	SQP(MATLAB) [m^2]	GA (OCTAVE) [m^2]	ERRO (%)
1	0.0145337	0.0145490	0,105272573
2	0.0114899	0.0115430	0,462145014
3	0.0102769	0.0104420	1,606515583
4	0.0229798	0.0231410	0,701485653
5	0.0185269	0.0188530	1,760143359
6	0.0102769	0.0107190	4,301880917
7	0.0154153	0.0157070	1,892275856
8	0.0287248	0.0288640	0,484598674
9	0.0205538	0.0208940	1,655168387
10	0.0308307	0.0310690	0,772930877
11	0.0334000	0.0334000	0
Valor da Função Objetivo	0.328862 m^3	0.332111 m^3	0,987952393
Nº de Iterações	7	80	-
Tempo Computacional	-	81.889 s	-

Os valores obtidos recorrendo ao algoritmo de otimização GA(OCTAVE) são bastante semelhantes em relação aos obtidos pelo SQP(MATLAB), embora apresentem algum valor de erro esse valor nunca é superior a 4,4% o que permite concluir que o algoritmo atinge resultados admissíveis.

O número de iterações para os algoritmos não baseados no gradiente tende a ser superior ao dos algoritmos baseados no gradiente, pois não possuem um critério de convergência claro, e na maioria das vezes para se convergir para um valor é necessário um grande número de iterações. Consequentemente quanto mais elevado o número de iterações maior será o tempo computacional.

A estrutura com as barras otimizadas é apresentada na Figura 5.6

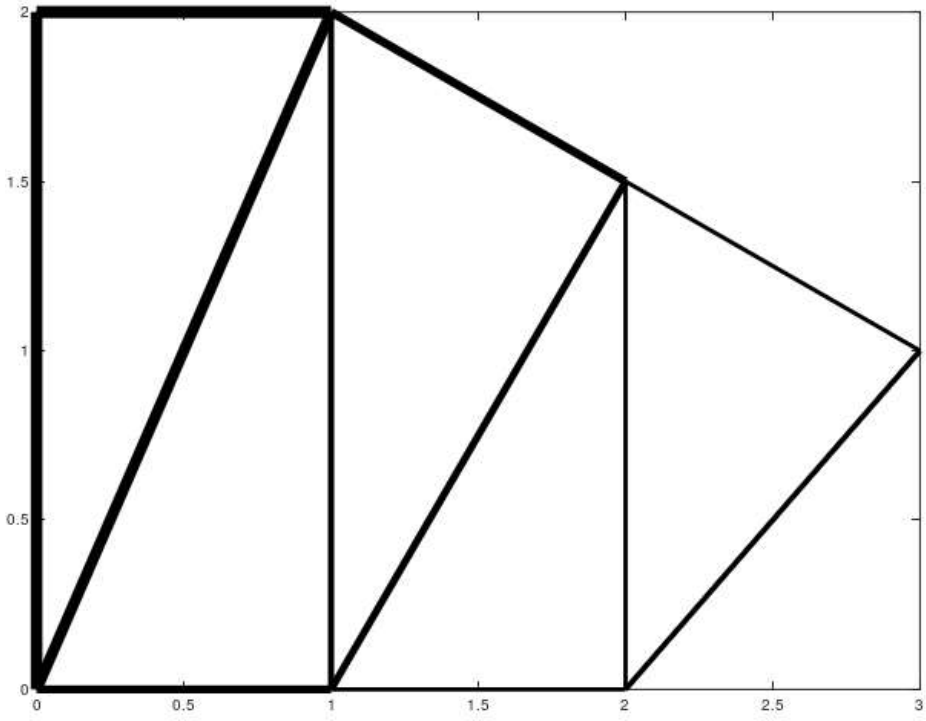

Figura 5.6 - Treliça 11 barras otimizada pelo GA OCTAVE

Na Figura 5.7 é apresentada a convergencia para o AG REAL.

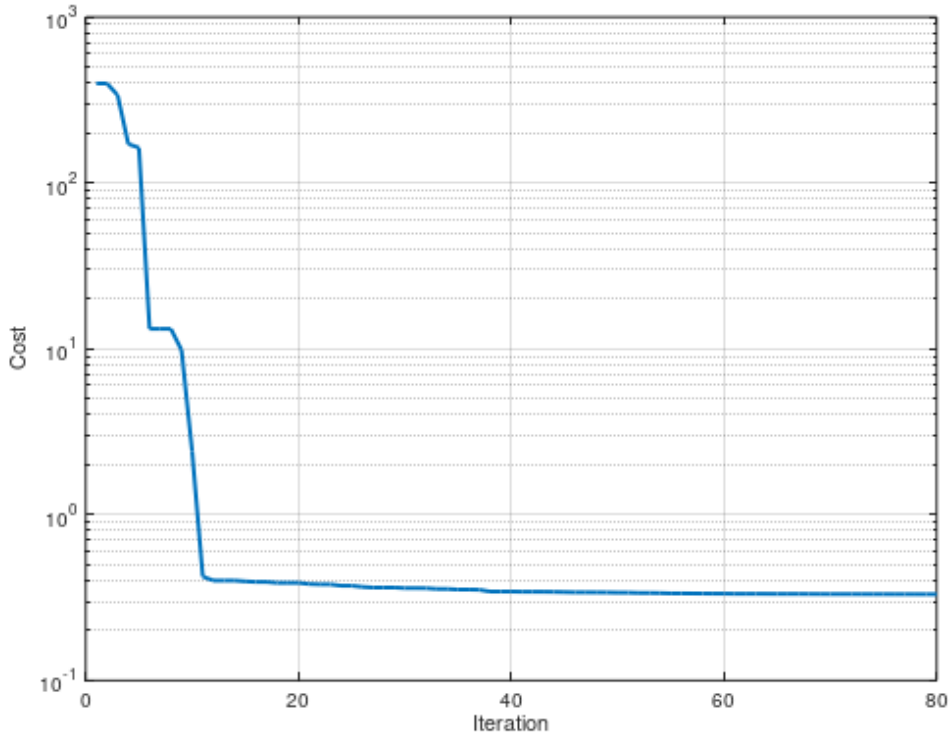

Figura 5.7 - Gráfico de convergência do algoritmo GA OCTAVE

5.1.4. Resultados Obtidos do Algoritmo Genético Binário

As variáveis que são apresentadas na Tabela 5.5 pertenciam a um conjunto de perfis HEB e foram escolhidos de forma a que os valores disponíveis pertençam ao intervalo compreendido entre o *lower* e o *upper bound* do problema. O valor da função objetivo e das variáveis otimizadas são apresentados na Tabela 5.6

Tabela 5.5 - Valores das áreas disponíveis para as variáveis do problema

DESIGNAÇÃO	ÁREA $[m^2]$
1-HEB 220	0.0091
2-HEB 240	0.0106
3-HEB 260	0.0118
4-HEB 280	0.0131
5-HEB 300	0.0149
6-HEB 320	0.0161
7-HEB 340	0.0171
8-HEB 360	0.0181
9-HEB 400	0.0198
10-HEB 450	0.0218
11-HEB 500	0.0239
12-HEB 550	0.0254
13-HEB 600	0.0270
14-HEB 650	0.0286
15-HEB 700	0.0306
16-HEB 800	0.0334

Tabela 5.6 – Resultados da otimização com o algoritmo GA BINÁRIO

ELEMENTO	SQP(MATLAB) [m^2]	GA (OCTAVE) $[m^2]$	ERRO (%)
1	0.0145337	HEB 280 - 0.0131	-9,8646594
2	0.0114899	HEB 260 - 0.0118	2,6988921
3	0.0102769	HEB 220 - 0.0091	-11,4518970
4	0.0229798	HEB 450 - 0.0218	-5,1340743
5	0.0185269	HEB 360 - 0.0181	-2,3042171
6	0.0102769	HEB 220 - 0.0091	-11,4518970
7	0.0154153	HEB 300 - 0.0149	-3,3427828
8	0.0287248	HEB 700 - 0.0306	6,5281569
9	0.0205538	HEB 400 - 0.0198	-3,6674484
10	0.0308307	HEB 700 - 0.0306	-0,7482801
11	0.0334000	HEB 800 - 0.0334	0,0000000
Valor da Função Objetivo	0.328862 m^3	0.332111 m^3	0,9879524
Nº de Iterações	7	80	-
Tempo Computacional	-	81.889 s	-

Embora se estejam a comparar resultados de dois problemas, um com variáveis contínuas e outro com variáveis discretas, os valores atingidos pelo algoritmo são admissíveis. Os valores de erro que apresentam estão de acordo com o esperado. A Figura 5.8 e Figura 5.9 apresentam respetivamente o gráfico com as barras otimizadas e o gráfico de convergência do algoritmo.

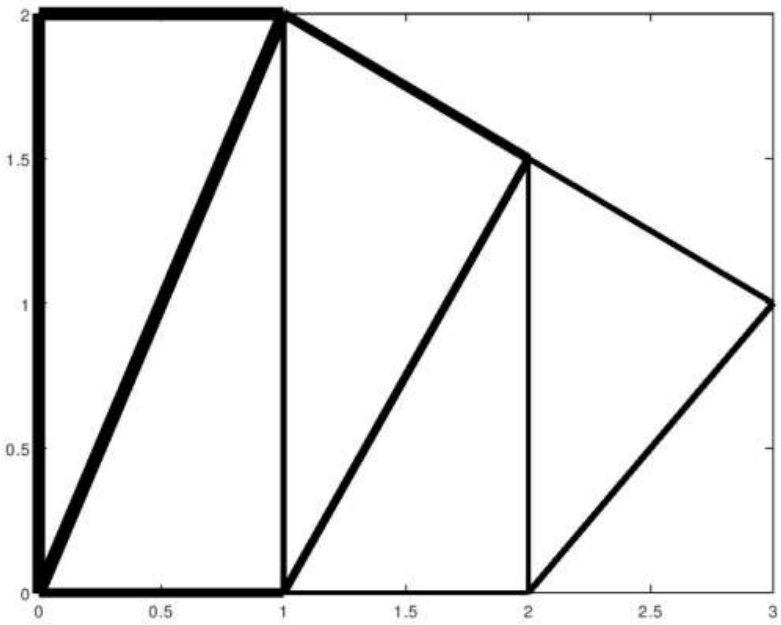

Figura 5.8 - Treliça 11 barras otimizada pelo GA OCTAVE

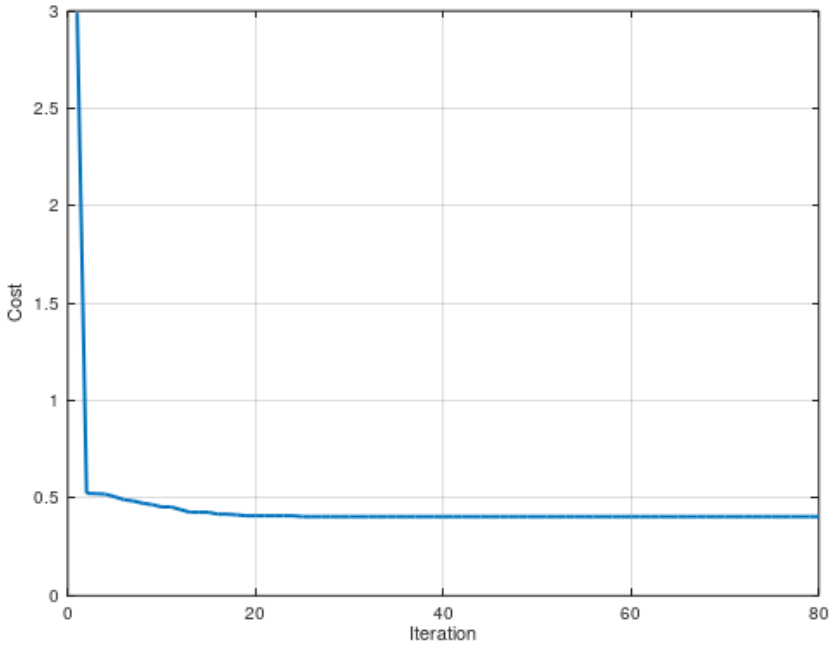

Figura 5.9 - Gráfico de convergência do algoritmo GA OCTAVE

5.1.5. Resultados Obtidos do Algoritmo PSO

O valor da função objetivo e as variáveis otimizadas pelo algoritmo são apresentados na Tabela 5.7. Os valores são comparados com os resultados provenientes do SQP, do programa MATLAB.

Tabela 5.7 - Resultados da otimização com o algoritmo PSO (OCTAVE)

ELEMENTO	SQP(MATLAB) [m^2]	PSO(OCTAVE) [m^2]	ERRO (%)
1	0.0145337	0.0145338	0,0006881
2	0.0114899	0.0114899	0,0000000
3	0.0102769	0.0102769	0,0000000
4	0.0229798	0.0229798	0,0000000
5	0.0185269	0.0185269	0,0000000
6	0.0102769	0.0102773	0,0038922
7	0.0154153	0.0154154	0,0006487
8	0.0287248	0.0287254	0,0020888
9	0.0205538	0.0205538	0,0000000
10	0.0308307	0.0308307	0,0000000
11	0.0334000	0.0333999	-0,0002994
Valor da Função Objetivo	0.328862 m^3	0.328863 m^3	0,0003041
Nº de Iterações	7	380	-
Tempo Computacional	-	276.691 s	-

É possivel observar que os valores obtidos recorrendo ao algoritmo de otimização PSO(OCTAVE) são bastante semelhantes em relação aos valores obtidos pelo SQP(MATLAB). O que leva a que o valor do erro tanto para as areas dos elementos como o valor da função objetivo seja proximo de zero. A estrutura com as barras otimizadas é apresentada na Figura 5.10. e na figura 5.11 a convergência do algoritmo.

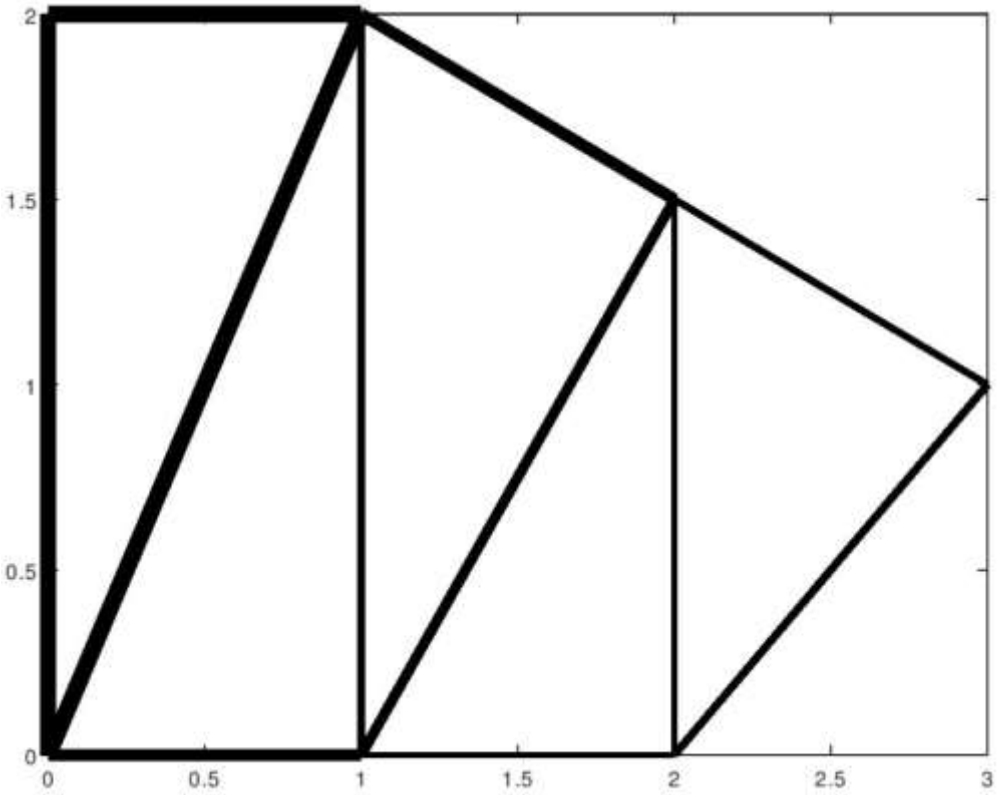

Figura 5.10 - Treliça 11 barras otimizada pelo OCTAVE

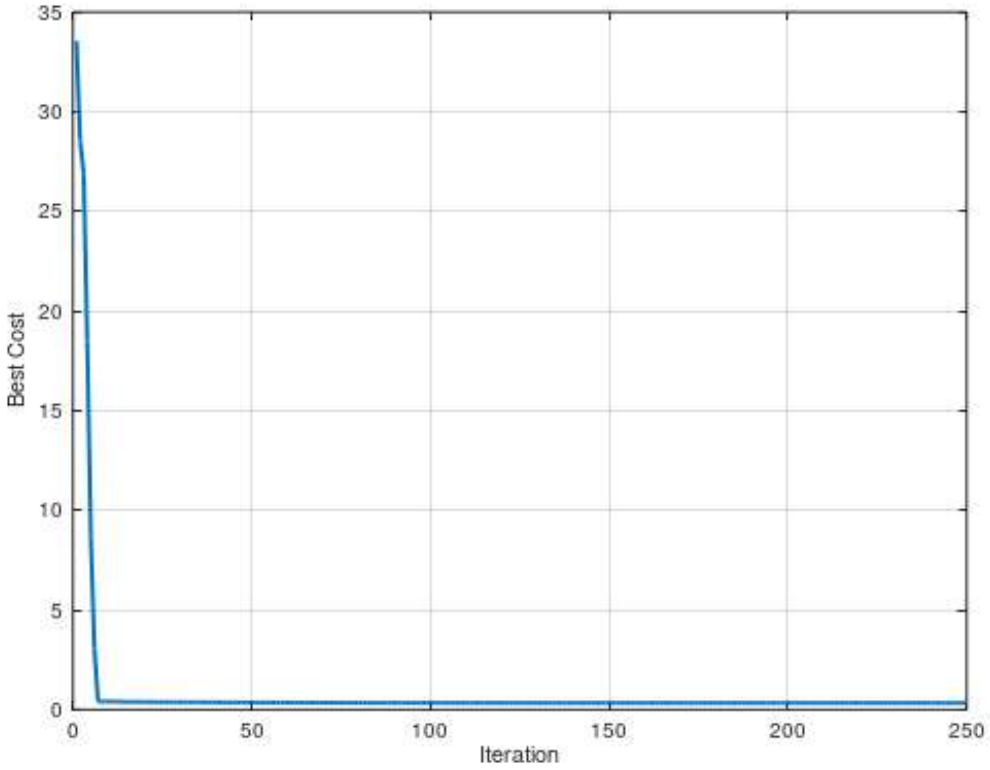

Figura 5.11 - Gráfico de convergência do algoritmo

5.1. Exemplo 2 – Treliça 38 Barras

A estrutura em estudo do exemplo 2 encontra-se presente no Manual do Programa PROAES 5.0 e encontra-se representada na Figura 5.12. O objetivo deste problema era a minimização do deslocamento do nó 14 com constrangimento de volume, utilizando o Método de Programação Quadrática Sequencial (SQP) disponibilizado pelo programa MATLAB. Os resultados obtidos usando o programa MATLAB são utilizados para comparar com os resultados obtidos pelos algoritmos:

Dados do Problema:

Tabela 5.8 - Valores iniciais das variáveis e do Volume

Área inicial do 38 Elementos, A [m^2]	0.01
Volume inicial da Estrutura, V [m^3]	0.2231
Módulo de Young, E [GPa]	210
Carregamento Aplicado, P [N]	1000
Tensão Admissível, σ_{adm} [MPa]	100
Número de Elementos	38
Número de nós	15

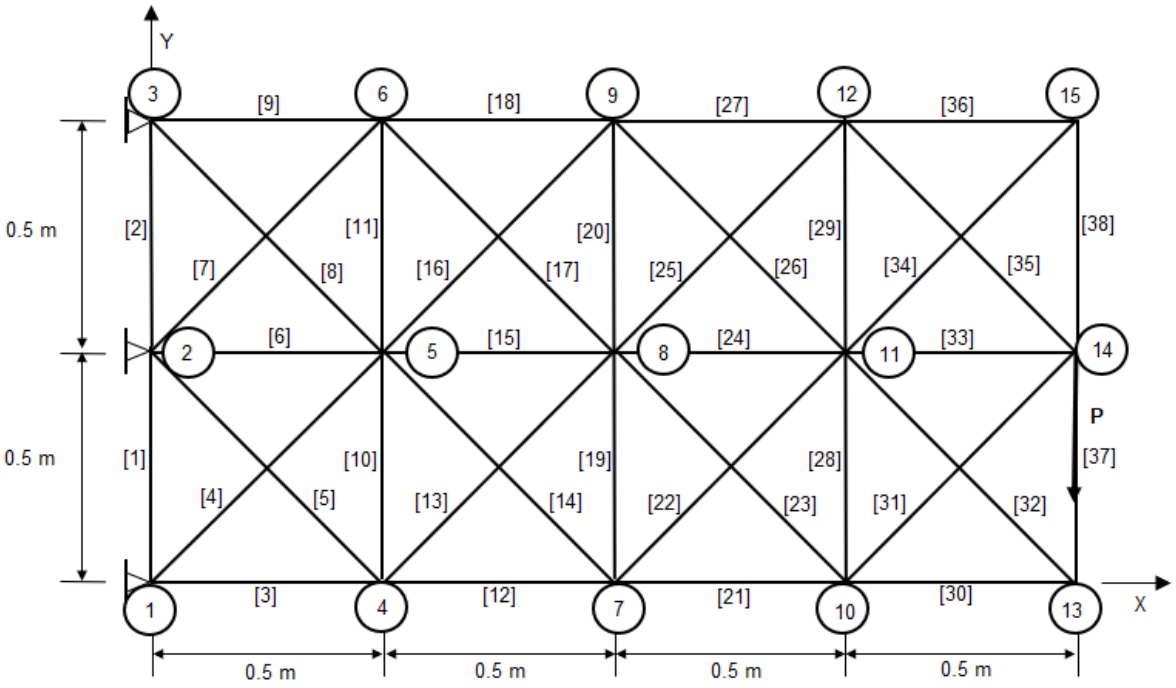

Figura 5.12 - Estrutura do 2º problema – Treliça 38 barras

Formulação do Problema de Otimização Dimensional

• Função objetivo

- Minimizar o deslocamento do nó 14.

•Variáveis do problema

- A área da secção transversal de cada barra.

• Constrangimentos do problema

- O volume da configuração final não pode ser superior a 10% do volume inicial.

$$V_f \leq 0.10 \times 0.223137084 \ m\text{^}3 \tag{5.4}$$

- A área da secção transversal de cada barra está limitada superior e inferiormente.

$$1 \times 10^{-8} \ m\text{^}2 < x_i < 1 \times 10^{-2} \ m^2 \tag{5.5}$$

O problema de otimização fica com a seguinte forma,

$$\begin{aligned} \min_{x} \quad & \delta_{14} \\ s.a. \quad & V_f \leq (0.10 \times 0.223137084\ ) \ m^3 \\ & 1 \times 10^{-8} \ m^2 < x_i < 1 \times 10^{-2} \ m^2 \qquad ;i = 1, \dots, 38 \end{aligned} \tag{5.6}$$

5.2.1. Resultados Obtidos do Algoritmo SQP

O valor da função objetivo e as variáveis otimizadas pelo Algoritmo são apresentados na Tabela 5.9 são comparados com os resultados provenientes do Programa MATLAB

Tabela 5.9 - Resultados da otimização com o algoritmo SQP

ELEMENTO	SQP (MATLAB) $[m^2]$	SQP* (OCTAVE) $[m^2]$	SQP' (OCTAVE) $[m^2]$	ERRO SQP(MATLAB) SQP*(OCTAVE) (%)	ERRO SQP(MATLAB) SQP*(OCTAVE) (%)
1	0,0000000	0,0000000	0,0000000	-	-
2	0,0000000	0,0000000	0,0000000	-	-
3	0,0041833	0,0043048	0,0043049	2,9044056	2,9067961
4	0,0019704	0,0018011	0,0018010	-8,5921640	-8,5972391
5	0,0000000	0,0001712	0,0001713	-	-
6	0,0000000	0,0000000	0,0000000	-	-
7	0,0000038	0,0001712	0,0001713	4405,2631579	4407,8947368
8	0,0019704	0,0018011	0,0018010	-8,5921640	-8,5972391
9	0,0041864	0,0043048	0,0043049	2,8282056	2,8305943
10	0,0000082	0,0006368	0,0006368	7665,8536585	7665,8536585
11	0,0000074	0,0006368	0,0006368	8505,4054054	8505,4054054
12	0,0041758	0,0034260	0,0034259	-17,9558408	-17,9582355
13	0,0000116	0,0010718	0,0010719	9139,6551724	9140,5172414
14	0,0019592	0,0009005	0,0009003	-54,0373622	-54,0475704
15	0,0000000	0,0000000	0,0000000	-	-
16	0,0019591	0,0009005	0,0009003	-54,0350161	-54,0452248
17	0,0000143	0,0010718	0,0010719	7395,1048951	7395,8041958
18	0,0041737	0,0034260	0,0034259	-17,9145602	-17,9169562
19	0,0000000	0,0000000	0,0000000	-	-
20	0,0000000	0,0000000	0,0000000	-	-
21	0,0014048	0,0021524	0,0021525	53,2175399	53,2246583
22	0,0019597	0,0009005	0,0009003	-54,0490891	-54,0592948
23	0,0000143	0,0010718	0,0010719	7395,1048951	7395,8041958
24	0,0000000	0,0000000	0,0000000	-	-
25	0,0000116	0,0010718	0,0010719	9139,6551724	9140,5172414
26	0,0019585	0,0009005	0,0009003	-54,0209344	-54,0311463
27	0,0014028	0,0021524	0,0021525	53,4359852	53,4431138
28	0,0013838	0,0006367	0,0006366	-53,9890158	-53,9962422
29	0,0013867	0,0006367	0,0006366	-54,0852383	-54,0924497
30	0,0000000	0,0000000	0,0000000	-	-
31	0,0019722	0,0019723	0,0019723	0,0050705	0,0050705
32	0,0000000	0,0000000	0,0000000	-	-
33	0,0000000	0,0000000	0,0000000	-	-
34	0,0000000	0,0000000	0,0000000	-	-
35	0,0019722	0,0019723	0,0019723	0,0050705	0,0050705
36	0,0000000	0,0000000	0,0000000	-	-
37	0,0000000	0,0000000	0,0000000	-	-
38	0,0000000	0,0000000	0,0000000	-	-
Valor da Função Objetivo	$1.365809 \times 10^{-5}\ m$	$1.365809 \times 10^{-5}\ m$	$1.365809 \times 10^{-5}\ m$	0	0
Nº de Iterações	174	28	28	-	-
Tempo Computacional	-	15.710 s	171.979 s	-	-

Os valores das áreas dos elementos apresentam dissemelhanças, o SQP(OCTAVE) tomou uma decisão diferente na escolha das variáveis quando comparado com o SQP(MATLAB). Essa diferença é mais suscetível de observar quando se analisam os valores do erro para as áreas e os gráficos de topologia ótima apresentados nas Figura 5.13 e Figura 5.14. Contudo a percentagem de erro para a função objetivo é zero, o que indica que para a mesma função existem vários mínimos locais.

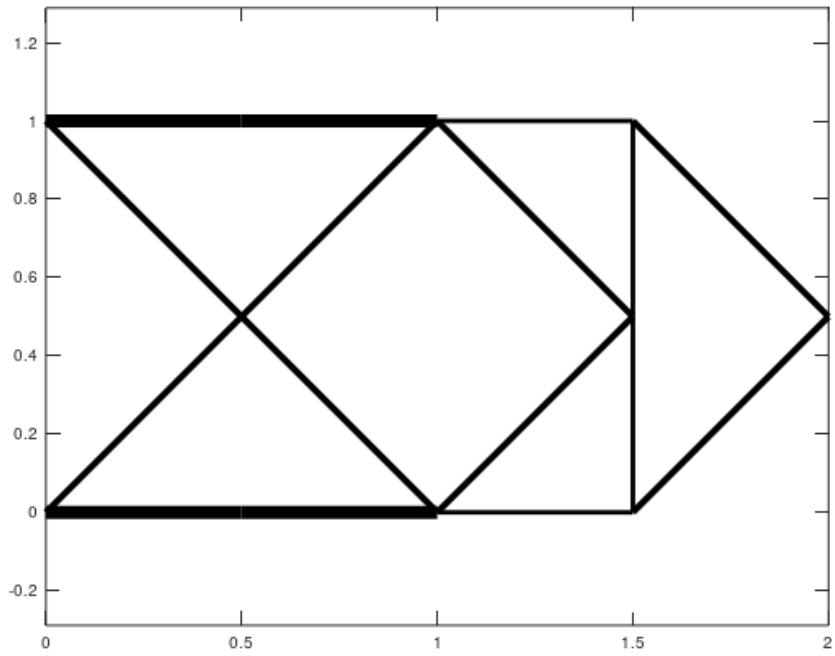

Figura 5.13 - Topologia ótima MATLAB

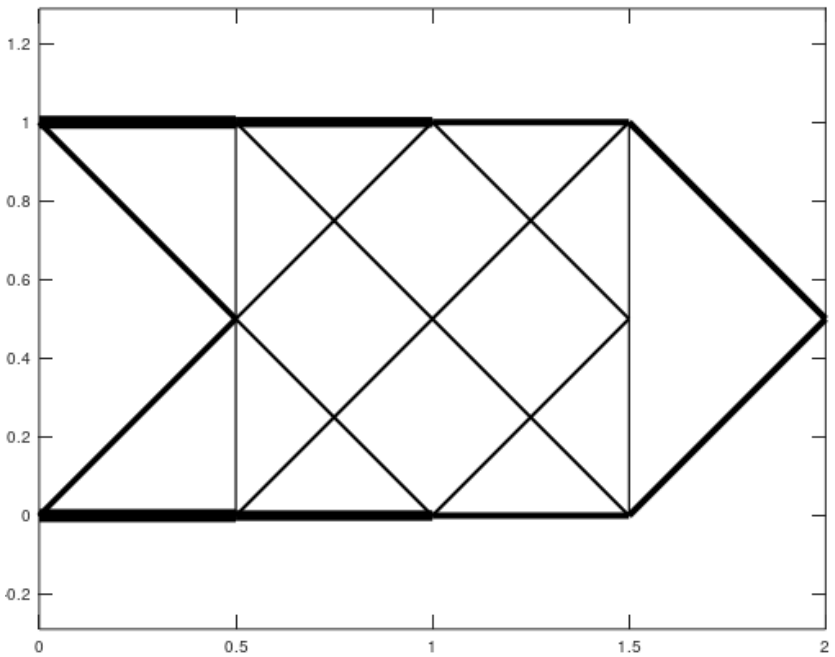

Figura 5.14 - Topologia ótima OCTAVE

5.2.2. Resultados Obtidos do Algoritmo MMA

O valor da função objetivo e as variáveis otimizadas pelo algoritmo são apresentados na Tabela 5.10.

Tabela 5.10 - Resultados da otimização com o algoritmo MMA.

ELEMENTO	SQP (MATLAB) $[m^2]$	MMA (OCTAVE) $[m^2]$	ERRO (%)
1	0,0000000	0,0000179	-
2	0,0000000	0,0000179	-
3	0,0041833	0,0044980	2,9067961
4	0,0019704	0,0014733	-8,5972391
5	0,0000000	0,0004866	-
6	0,0000000	0,0000179	-
7	0,0000038	0,0004866	4407,8947368
8	0,0019704	0,0014733	-8,5972391
9	0,0041864	0,0044980	2,8305943
10	0,0000082	0,0005103	7665,8536585
11	0,0000074	0,0005103	8505,4054054
12	0,0041758	0,0032987	-17,9582355
13	0,0000116	0,0012066	54,3103448
14	0,0019592	0,0007525	-99,0863618
15	0,0000000	0,0000179	-
16	0,0019591	0,0007525	-24,7971007
17	0,0000143	0,0012066	3302,7972028
18	0,0041737	0,0032987	-99,5711239
19	0,0000000	0,0000177	-
20	0,0000000	0,0000177	-
21	0,0014048	0,0022400	220,1879271
22	0,0019597	0,0007524	-73,9603000
23	0,0000143	0,0012137	3468,5314685
24	0,0000000	0,0000179	-
25	0,0000116	0,0012137	10301,7241379
26	0,0019585	0,0007524	-61,5777381
27	0,0014028	0,0022400	-98,7239806
28	0,0013838	0,0005107	-45,6207544
29	0,0013867	0,0005107	-12,9876686
30	0,0000000	0,0000325	-
31	0,0019722	0,0019273	-99,1025251
32	0,0000000	0,0000380	-
33	0,0000000	0,0000179	-
34	0,0000000	0,0000380	-
35	0,0019722	0,0019273	-38,4595883
36	0,0000000	0,0000325	-
37	0,0000000	0,0000325	-
38	0,0000000	0,0000325	-
Valor da Função Objetivo	$1.365809 \times 10^{-5}\ m$	$1.374500 \times 10^{-5}\ m$	0,636326
Nº de Iterações	174	243	-
Tempo Computacional	-	199.506 s	-

Os valores das áreas dos elementos apresentam diferenças, e quando comparado, o erro de alguns elementos é bastante elevado, o que indica que o SQP(MATLAB) tomou uma decisão

diferente na escolha das variáveis quando comparado com o MMA(OCTAVE), essa diferença é a causa para os valores de erro serem tão altos. Essa disparidade é mais suscetível de observar quando se analisam os gráficos de topologia ótima apresentados nas Figura 5.15 e Figura 5.13. Contudo o valor do erro para a função objetivo é inferior a 0,7%.

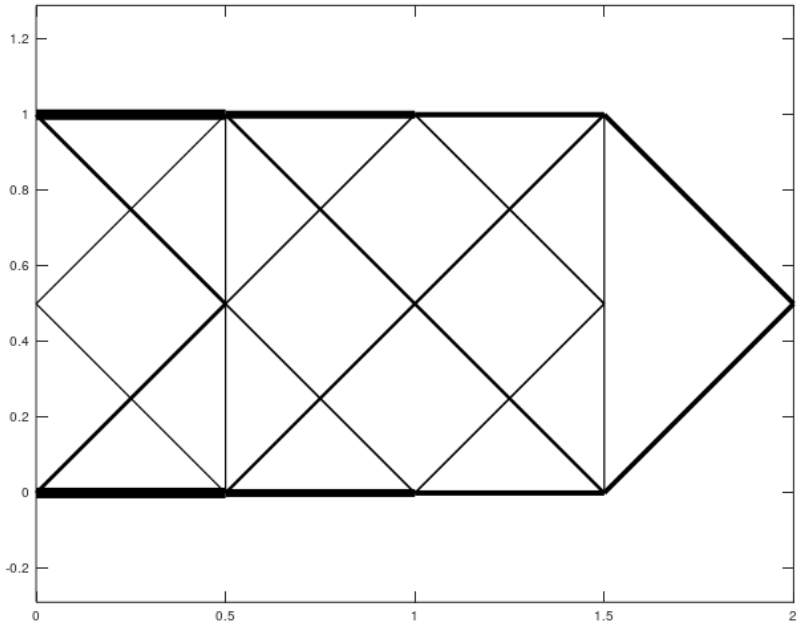

Figura 5.15 - Topologia ótima MMA OCTAVE

De acordo com o Figura 5.16. o algoritmo converge rapidamente para o resultado

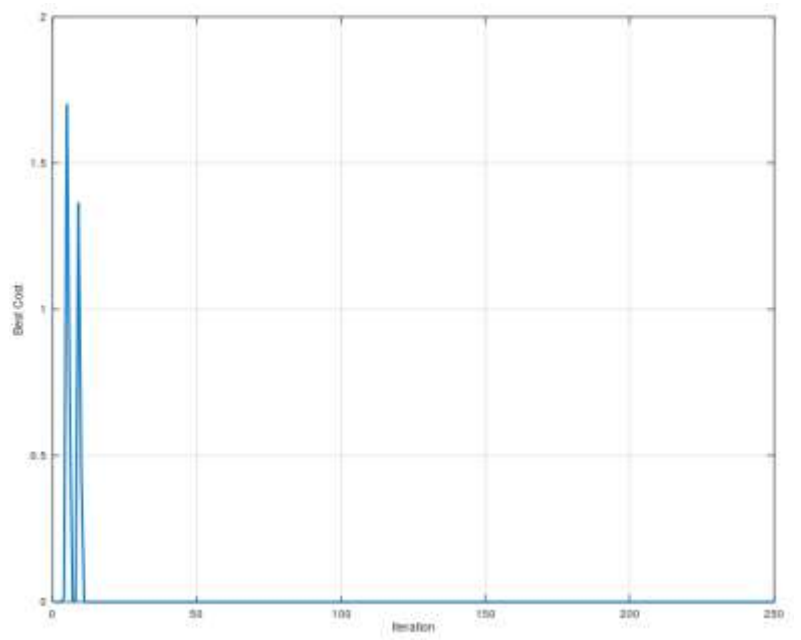

Figura 5.16 - Gráfico de convergência do algoritmo MMA OCTAVE

5.2.3. Resultados Obtidos do Algoritmo Genético Real

O valor da função objetivo e as variáveis otimizadas pelo algoritmo são apresentados na Tabela 5.11

Tabela 5.11 - Resultados da otimização com o algoritmo GA.

Barras	SQP (MATLAB) $[m^2]$	GA (OCTAVE) $[m^2]$	ERRO (%)
1	0,0000000	0,0000000	-
2	0,0000000	0,0000000	-
3	0,0041833	0,0042290	1,0924390
4	0,0019704	0,0016290	-17,3264312
5	0,0000000	0,0000080	-
6	0,0000000	0,0000000	-
7	0,0000038	0,0008480	22215,7894737
8	0,0019704	0,0015520	-21,2342672
9	0,0041864	0,0048310	15,3974775
10	0,0000082	0,0013710	16619,5121951
11	0,0000074	0,0004450	5913,5135135
12	0,0041758	0,0027530	-34,0725131
13	0,0000116	0,0019350	16581,0344828
14	0,0019592	0,0002520	-87,1376072
15	0,0000000	0,0000000	-
16	0,0019591	0,0002320	-88,1578276
17	0,0000143	0,0015570	10788,1118881
18	0,0041737	0,0030950	-25,8451733
19	0,0000000	0,0000000	-
20	0,0000000	0,0000000	-
21	0,0014048	0,0024280	72,8359909
22	0,0019597	0,0002440	-87,5491147
23	0,0000143	0,0014980	10375,5244755
24	0,0000000	0,0000000	-
25	0,0000116	0,0018730	16046,5517241
26	0,0019585	0,0002590	-86,7755936
27	0,0014028	0,0027980	99,4582264
28	0,0013838	0,0003190	-76,9475358
29	0,0013867	0,0000140	-98,9904089
30	0,0000000	0,0000000	-
31	0,0019722	0,0019580	-0,7200081
32	0,0000000	0,0000000	-
33	0,0000000	0,0000000	-
34	0,0000000	0,0000000	-
35	0,0019722	0,0019540	-0,9228273
36	0,0000000	0,0000000	-
37	0,0000000	0,0000000	-
38	0,0000000	0,0000000	-
Valor da Função Objetivo	$1.365809 \times 10^{-5}\ m$	$1.366428 \times 10^{-5}\ m$	0,045321
Nº de Iterações	174	400	
Tempo Computacional	-	1665s	

Os elevados valores de erro entre as áreas do SQP(MATLAB) e do GA(OCTAVE) indicam que o GA tomou uma decisão diferente na escolha das variáveis. Essa diferença é mais suscetível de observar quando se analisam os gráficos de topologia ótima apresentados nas Figura 5.17e Figura 5.13. Contudo o valor do erro para a função objetivo é inferior a 0,05%.

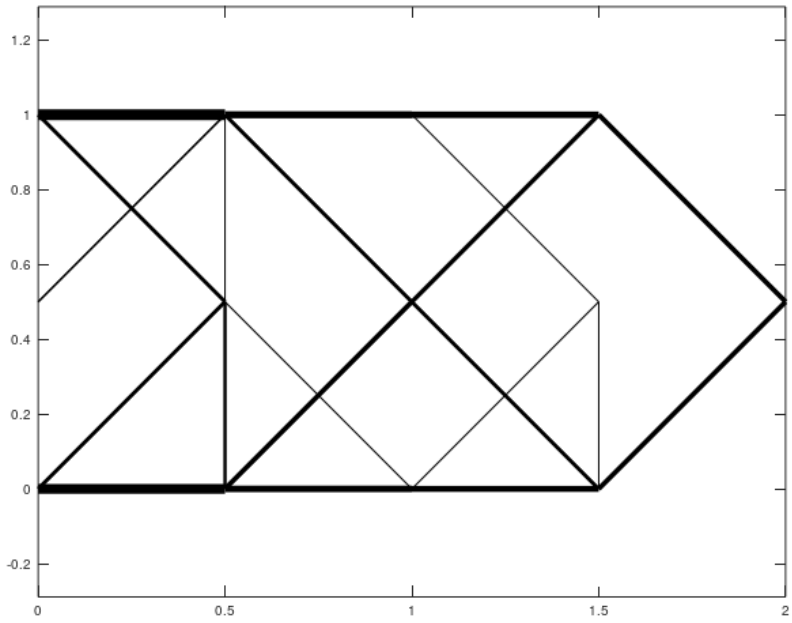

Figura 5.17 - Topologia ótima GA OCTAVE

Na Figura 5.18 é apresentado o grafico de convergencia obtido pelo OCTAVE quando utilizado o GA REAL.

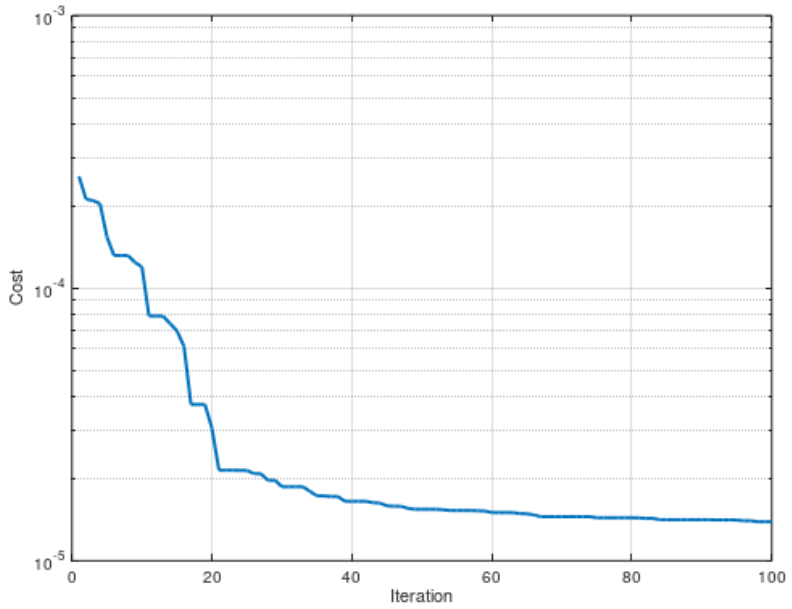

Figura 5.18 - Gráfico de convergência do algoritmo GA REAL OCTAVE

5.2.4. Resultados obtidos do Algoritmo PSO

O valor da função objetivo e as variáveis otimizadas pelo Algoritmo são apresentados na Tabela 5.12.

Tabela 5.12 - Resultados da otimização com o algoritmo PSO.

Barras	SQP (MATLAB) $[m^2]$	PSO (OCTAVE) $[m^2]$	ERRO (%)
1	0,0000000	0,0000000	-
2	0,0000000	0,0000001	-
3	0,0041833	0,0042153	0,7649463
4	0,0019704	0,0014375	-27,0452700
5	0,0000000	0,0000249	-
6	0,0000000	0,0000000	-
7	0,0000038	0,0010664	27963,1578947
8	0,0019704	0,0014354	-27,1518473
9	0,0041864	0,0049082	17,2415440
10	0,0000082	0,0004247	5079,2682927
11	0,0000074	0,0001833	2377,0270270
12	0,0041758	0,0037560	-10,0531635
13	0,0000116	0,0006147	5199,1379310
14	0,0019592	0,0009892	-49,5100041
15	0,0000000	0,0000001	-
16	0,0019591	0,0009997	-48,9714665
17	0,0000143	0,0013301	9201,3986014
18	0,0041737	0,0032473	-22,1961329
19	0,0000000	0,0000000	-
20	0,0000000	0,0000000	-
21	0,0014048	0,0023337	66,1232916
22	0,0019597	0,0009913	-49,4157269
23	0,0000143	0,0013283	9188,8111888
24	0,0000000	0,0000001	-
25	0,0000116	0,0006254	5291,3793103
26	0,0019585	0,0010035	-48,7618075
27	0,0014028	0,0018298	30,4391218
28	0,0013838	0,0004449	-67,8494002
29	0,0013867	0,0009431	-31,9896156
30	0,0000000	0,0000000	-
31	0,0019722	0,0019832	0,5577528
32	0,0000000	0,0000001	-
33	0,0000000	0,0000003	-
34	0,0000000	0,0000002	-
35	0,0019722	0,0019672	-0,2535240
36	0,0000000	0,0000001	-
37	0,0000000	0,0000000	-
38	0,0000000	0,0000001	-
Valor da Função Objetivo	$1.365809 \times 10^{-5}\ m$	$1.365875 \times 10^{-5}\ m$	0,0000483
Nº de Iterações	174	500	
Tempo Computacional	-	1310s	

Igualmente para o algoritmo PSO existem valores de erro elevados quando comparados com o SQP(MATLAB. Essa diferença é mais suscetível de observar quando se analisam os gráficos de topologia ótima apresentados nas Figura 5.19 e. Figura 5.13. Contudo também para este caso o valor do erro para a função objetivo é muito baixo aproximando-se muito de zero.

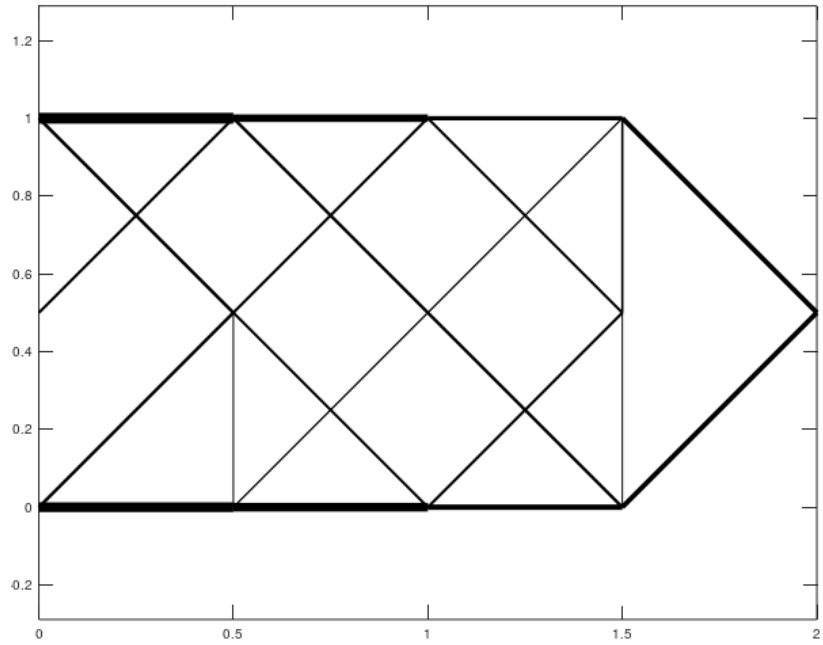

Figura 5.19 - Topologia ótima PSO OCTAVE

O algoritmo converge rapidamente para o resultado.

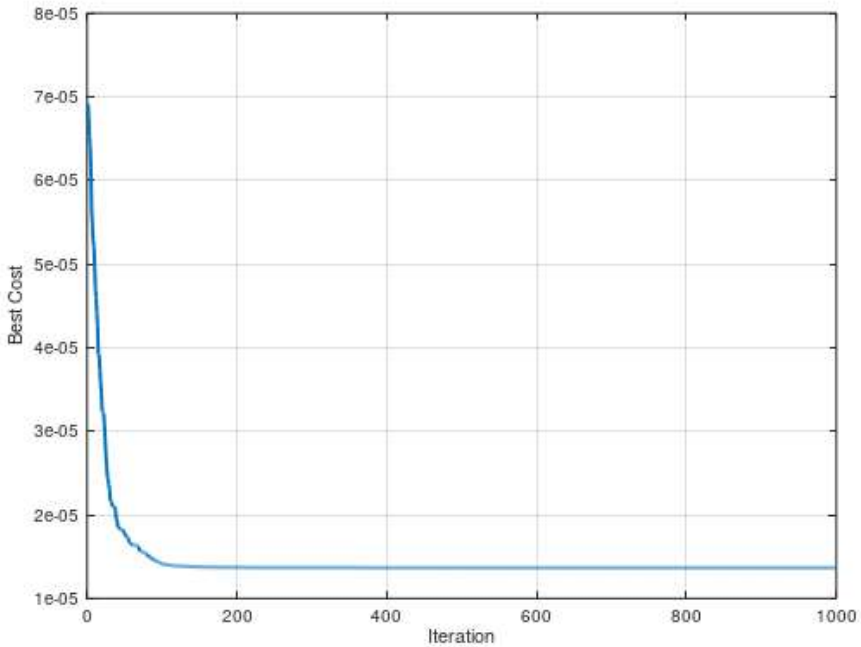

Figura 5.20 - Gráfico de convergência do algoritmo PSO OCTAVE

5.3. Exemplo 3 – Europa TIR

A estrutura em estudo do problema 3, ver Figura 5.21, é retirada de [31], um artigo desenvolvido no DEMI sobre Pesquisa Paralela na otimização Estrutural. Um dos objetivos deste trabalho era realizar otimização estrutural do chassis do semirreboque de um Camião TIR, onde o eixo e as rodas estão ligados a uma plataforma de carga. Esta plataforma é construída com vigas e placas e projetada para suportar as diversas cargas operacionais. O cenário de criação considerado corresponde à torção que ocorre na plataforma, quando se realiza uma curva.

Figura 5.21 - Semirreboque

Os pontos de ligação da suspensão traseira são fixos e aplica-se um binário de 5000 Nm ao pino mestre, isto é, ao pino que acopla o semirreboque ao trator.

Para realizar os cálculos da função objetivo do Problema foi usado o Programa ANSYS uma vez que o tipo de elementos não é suportado pelo programa PROAES. Os elementos finitos da estrutura são apresentados na Figura 5.22 a sua modelação é feita com elementos de casca e é composta por 12747 nós e 11864 elementos.

Dados do Problema:

São apresentados na página seguinte.

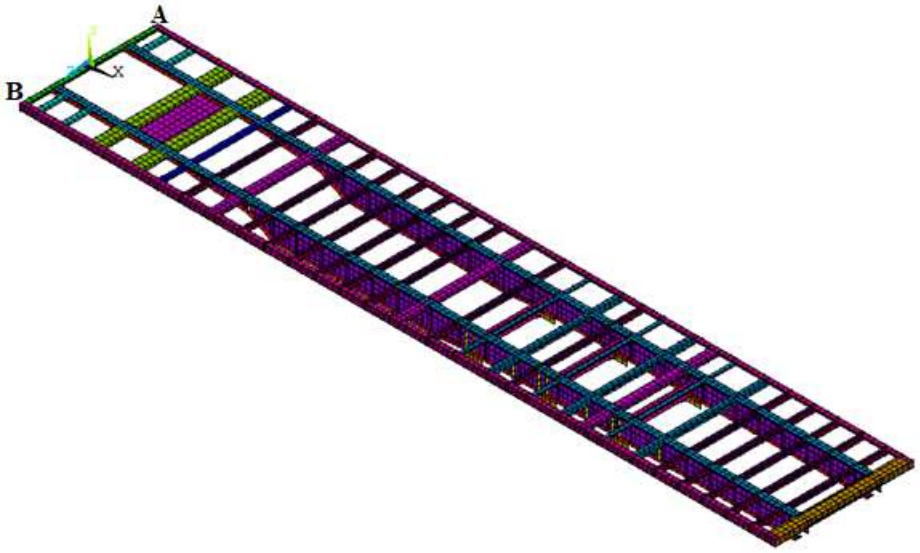

Figura 5.22 - Elementos finitos do chassis do Camião, adaptado de [31]

Um conjunto de 12 variáveis de projeto são apresentados nas Figura 5.23, Figura 5.24 e Figura 5.25. A Tabela 5.13 mostra as variáveis e os intervalos de variação admissíveis.

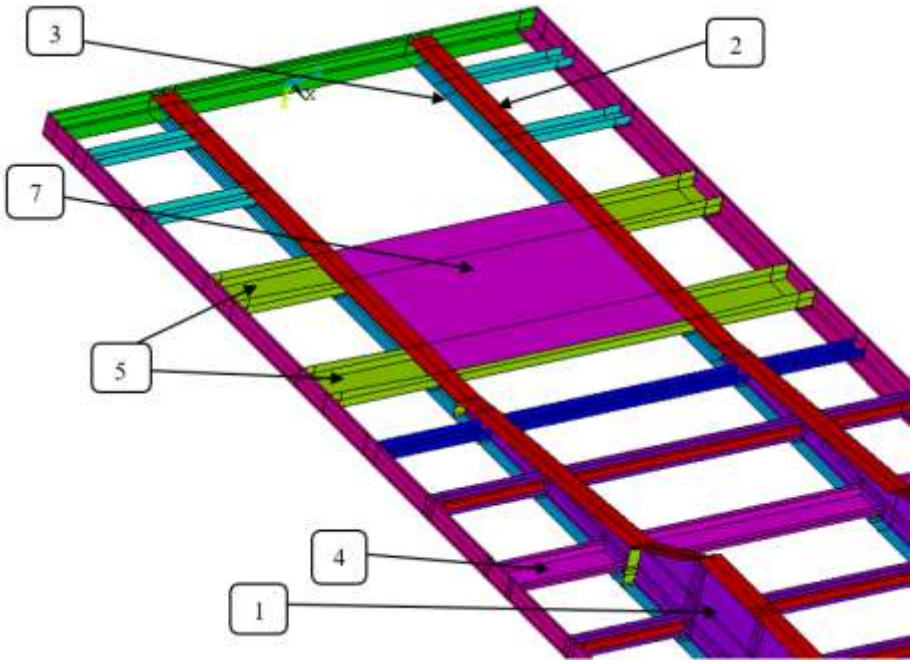

Figura 5.23 - Detalhes da figura do chassis do semirreboque, adaptado de [31]

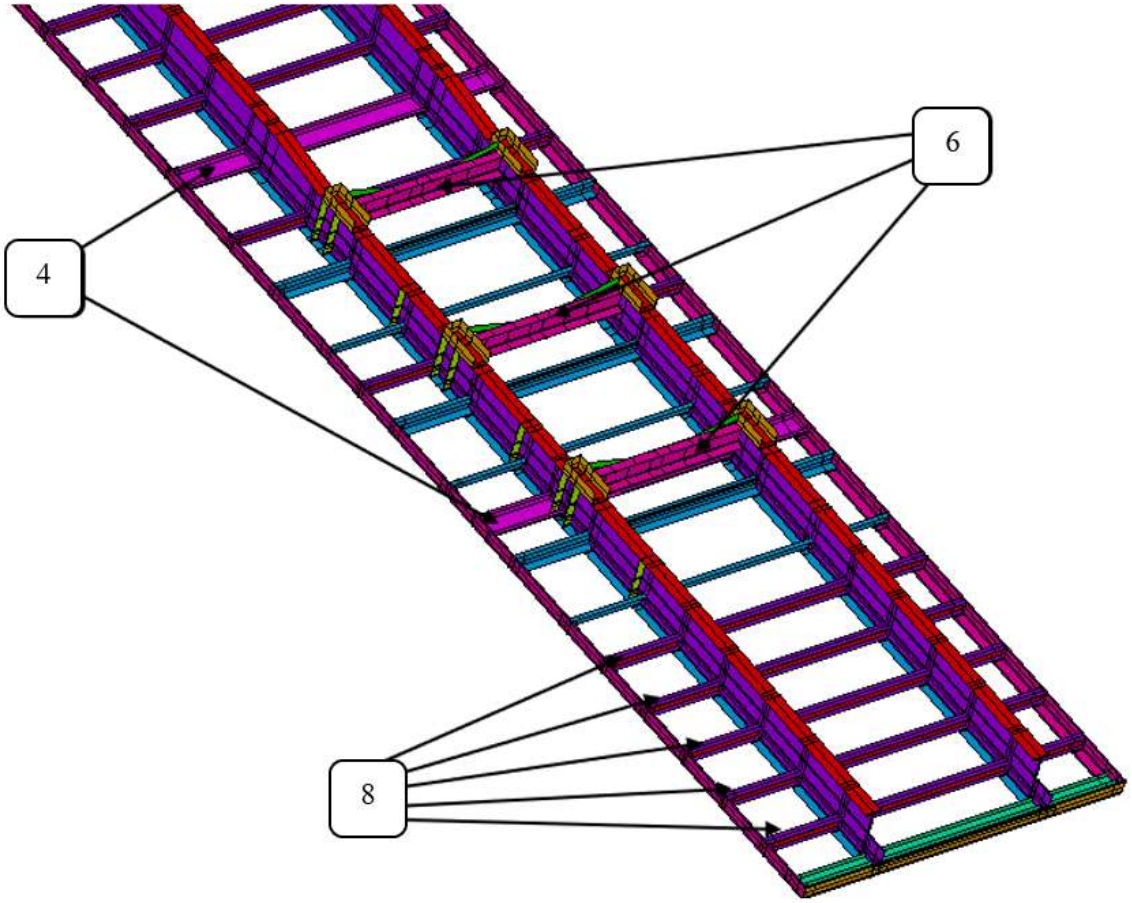

Figura 5.24 - Detalhes da figura do chassis do semirreboque, adaptado de [31]

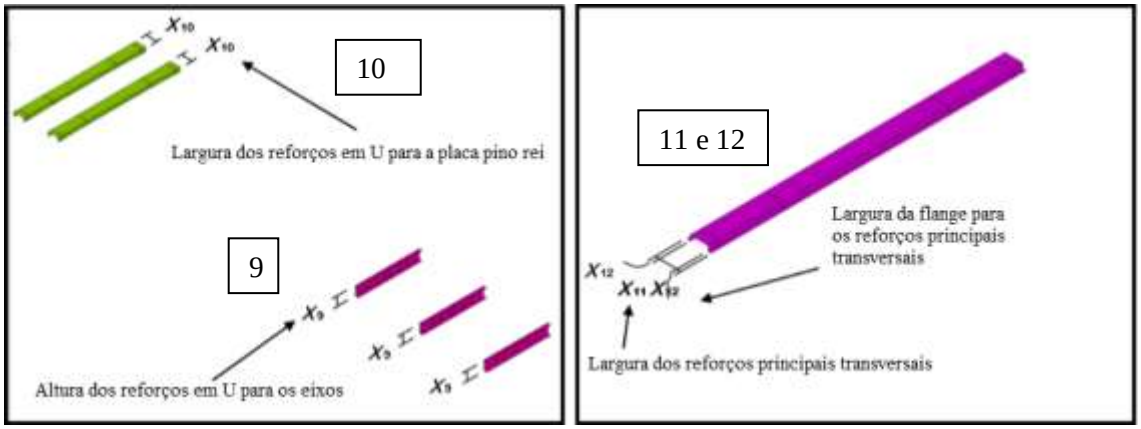

Figura 5.25 - Detalhes da figura do chassis do semirreboque, adaptado de [31]

Tabela 5.13 - Variáveis de projeto consideradas na otimização da estrutura do chassi do semirreboque, adaptado de [31]

Número da Variável	Descrição	Faixa da Variável (mm)
1	Alma central e Traseira da viga Principal	$5 \leq x_1 \leq 10$
2	Banzo inferior da viga principal	$12 \leq x_2 \leq 15$
3	Banzo superior da viga principal	$12 \leq x_3 \leq 15$
4	Reforços Transversais Principais	$3 \leq x_4 \leq 4.5$
5	Reforços de Perfil U da Placa Pino Rei	$4 \leq x_5 \leq 8$
6	Reforços de Perfil U dos Eixos	$2 \leq x_6 \leq 6$
7	Placa do pino rei	$5 \leq x_7 \leq 10$
8	Perfis IPN80	$0.001 \leq x_8 \leq 1$
9	Altura para os reforços em U para os eixos	$180 \leq x_9 \leq 240$
10	Largura dos reforços em U para o pino rei	$200 \leq x_{10} \leq 350$
11	Largura dos reforços principais transversais	$142 \leq x_{11} \leq 186$
12	Largura da flange para os reforços transversais principais	$42 \leq x_{12} \leq 44$

A função objetivo do problema é a minimização da massa. As variáveis do problema são apresentadas na Tabela 5.13. Para além do constrangimento imposto às variáveis é imposto um constrangimento no ângulo de rotação da secção frontal, que não pode exceder os 4.15° este constrangimento é imposto ao impedir um valor máximo de 180 mm ao somatório do deslocamentos absolutos dos pontos A e B (ver Figura 5.22).

Contudo existe uma diferença entre as variáveis usadas por [31] e as usadas nos algoritmos da *toolbox.* As usadas por [31] são variáveis discretas e as usadas nos algoritmos da *toolbox* são continuas. Embora exista esta diferença tomou-se a decisão de mesmo assim usar-se este problema como comparação.

5.3.1. Resultados obtidos pelos Algoritmos da *Toolbox* de Otimização

O valor da função objetivo e as variáveis otimizadas pelo Algoritmo são apresentados na Tabela 5.14. Os valores são comparados com resultados apresentados no artigo [31].

O espaço que se segue foi propositadamente deixado em branco.

Tabela 5.14 - Comparação das Variáveis de projeto otimizadas obtidas pelos algoritmos da *toolbox* de otimização e [31]

Número da Variável	Descrição	SQP **(OCTAVE)**	MMA **(OCTAVE)**	PSO **(OCTAVE)**	GA **(OCTAVE)**	PSD-MADS	P-MADS	PSO	GA
1	Alma central e Traseira da viga Principal	5,00	5,00	5,00	5,00	5,00	5,00	5,00	5,00
2	Banzo inferior da viga principal	12,00	12,00	14,21	15,00	13,75	13,76	12,19	15,00
3	Banzo superior da viga principal	12,00	12,00	12,00	12,00	12,00	12,00	12,00	12,00
4	Reforços Transversais Principais	3,00	3,00	3,00	3,00	3,00	3,00	3,73	3,00
5	Reforços de Perfil U da Placa Pino Rei	4,00	4,00	4,00	4,72	4,00	4,00	6,48	4,00
6	Reforços de Perfil U dos Eixos	2,00	2,00	2,00	2,00	2,00	2,00	2,00	2,00
7	Placa do pino rei	5,00	5,00	5,00	5,00	5,00	5,00	5,76	5,00
8	Perfis IPN80	0,00	0,02	0,00	0,00	0,00	0,00	0,00	-
9	Altura para os reforços em U dos eixos	236,03	240,00	232,66	180,00	180,00	180,00	180,00	183,75
10	Largura dos reforços em U para o pino rei	345,79	350,00	306,30	344,22	350,00	350,00	350,00	275,00
11	Largura dos reforços principais transversais	176,38	186,00	175,03	171,96	186,00	142,00	151,31	169,50
12	Largura da flange para os reforços transversais principais	30,22	44,00	28,85	24,00	24,00	24,00	28,10	32,75
Valor da função objetivo (N)		14973,00	15035,94	15410,52	15634,20	15258,47	15247,09	15512,65	15503,43
Tempo computacional		2093s	40942s	53225s	40942s	26877s	40296s	42665s	42840s

A convergência do algoritmo MMA é apresentada na Figura 5.26

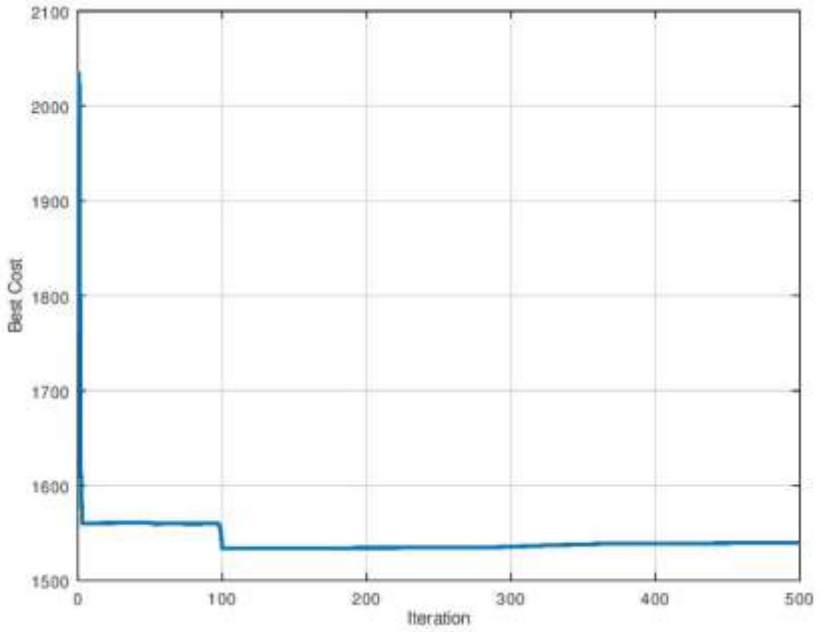

Figura 5.26 - Gráfico de convergência do algoritmo MMA OCTAVE.

A convergência do algoritmo PSO é apresentada na Figura 5.27.

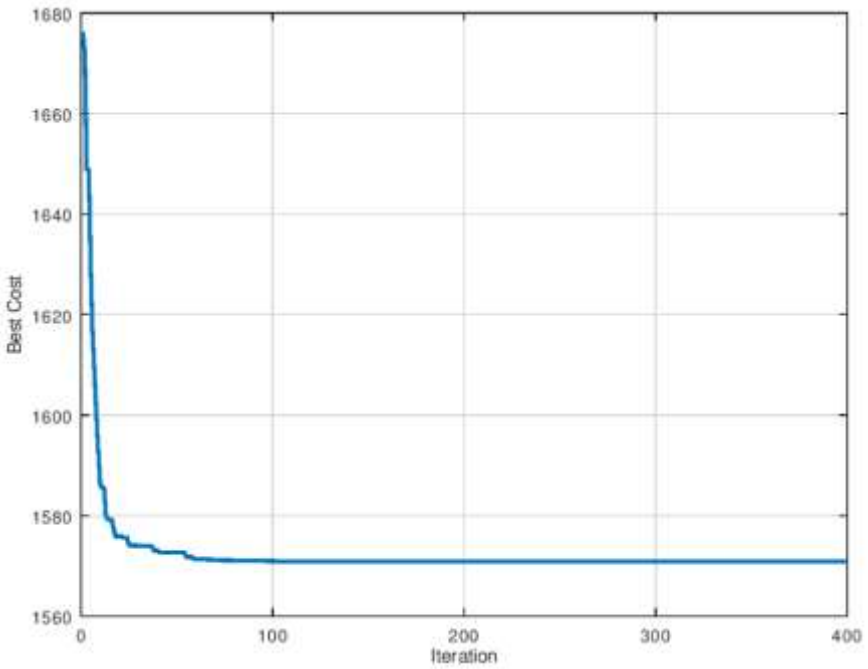

Figura 5.27 - Gráfico de convergência do algoritmo PSO OCTAVE.

A convergência do algoritmo GA REAL é apresentada na figura 5.28.

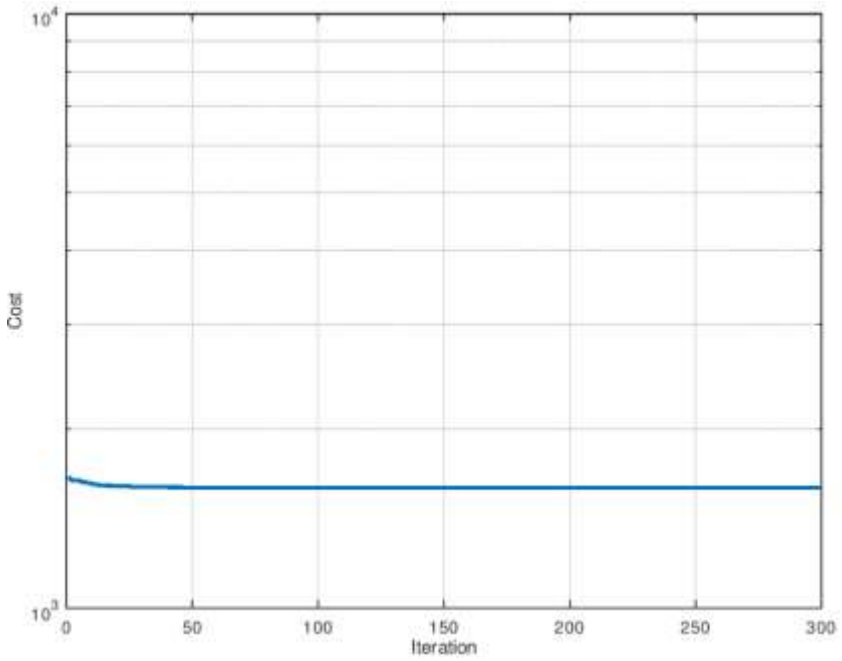

Figura 5.28 - Gráfico de convergência do algoritmo GA OCTAVE.

Pela comparação dos resultados obtidos com os diferentes algoritmos na otimização destes 3 últimos problemas, é possível de concluir que os algoritmos implementados chegam a resultados admissíveis.

6. Conclusões e desenvolvimentos futuros

O principal objetivo desta dissertação é a construção de uma *toolbox* de otimização para o PROAES, que agrupe um conjunto de algoritmos *Open Source* especializados na resolução de problemas de otimização estrutural.

Para agrupar um conjunto de algoritmos efetuou-se uma pesquisa, de onde se reuniram os seguintes 5 algoritmos: SQP, MMA, GA REAL, GA BINÁRIO e o PSO. O algoritmo SQP e o algoritmo MMA são algoritmos baseados em gradientes, especializados na resolução de problemas de otimização não linear com restrições. O algoritmo GA REAL e o PSO são Meta-Heurísticas, uma alternativa aos métodos baseados em gradientes, especializados na resolução de problemas cujas funções não necessitem de ser diferenciáveis ou mesmo contínuas. E por último o algoritmo GA BINÁRIO é também uma Meta-Heurística especializada na resolução de problemas cujas funções não necessitam de ser diferenciáveis ou mesmo contínuas, mas para o caso em que as variáveis são discretas, aplicado a problemas em que os valores discretos que as variáveis podem assumir são entregues sob a forma de tabelas.

Um dos objetivos da dissertação era conseguir criar uma interface comum para todos algoritmos, ou pelo menos para todos os algoritmos da mesma categoria (Baseados em gradientes/ Meta-heurísticas). Este objetivo foi parcialmente conseguido, isto é, para os algoritmos não baseados em gradientes a interface de funcionamento é muito semelhante (inclusive é possível trocar o GA REAL e o PSO sem fazer alterações no ficheiro **'myfun.m'**). Em todas as Meta-heurísticas a otimização é iniciada no próprio algoritmo. Já para os algoritmos baseados em gradientes isso não acontece em todos, por um lado a interface de funcionalmente é o mais semelhante possível, mas por outro lado não é possível recolher informação acerca de cada iteração no algoritmo SQP. Isto acontece porque é o algoritmo que controla a execução, ele decide quando chamar as funções internamente, contrariamente ao MMA que realiza a otimização através de um ciclo em que quem decide quando as funções são chamadas é o utilizador, que inclui no seu código M o ciclo iterativo que permite obter informação que posteriormente é usada para a construção de gráficos.

De forma a avaliar os resultados produzidos pelos algoritmos, estes foram aplicados na resolução de 3 problemas de otimização:

- O primeiro exemplo é um exemplo de otimização dimensional e foi resolvido recorrendo ao programa PROAES. Neste exemplo todos os algoritmos chegaram a bons resultados. O que indica que os algoritmos estão aptos para a resolução deste tipo de problemas.
- O segundo exemplo é um exemplo de otimização topológica e foi resolvido recorrendo também ao programa PROAES. Neste problema também todos os algoritmos chegam a solução admissível do problema. Contudo chegam a mínimos locais da função, pois todos eles distribuíram a área

das barras de forma diferente, resultando por isso numa solução topológica diferente para todos eles.

- O terceiro problema serviu para demostrar a versatilidade do Programa OCTAVE, e da *toolbox* de otimização na resolução de problemas de engenharia com recurso a algoritmos de otimização quando em colaboração com outros programas de análise.

Pela análise de todos os resultados obtidos nestes três exemplos de otimização pode ser dito que globalmente os objetivos principais desta dissertação foram atingidos. Os algoritmos implementados na *toolbox* estão a funcionar e são capazes de resolver os problemas de otimização propostos.

Como desenvolvimento futuro desta dissertação seria interessante a criação de uma interface gráfica de forma a acolher os algoritmos recolhidos e utilizados na *toolbox*. Seria também interessante que fossem adicionamos mais algoritmos de otimização.

Referências Bibibliográficas

[1] L. A. Schmit, “Structural Design by Systematic Synthesis,” *Proceedings, 2nd Conf. Electron. Comput. ASCE*, pp. 105–122, 1960.

[2] P. G. Coelho, “Apontamentos da disciplina de Tópicos Avançados em Mecânica Estrutural, Faculdade de Ciências e Tecnologias da Universidade Nova de Lisboa,” 2018.

[3] A. Gomes, “Programação Algoritmos e Programação Estruturada.” Universidade Beira Interior - Dep. Informática, Covilhã, 2015.

[4] T. Baptista, “Análise de Fiabilidade de Estruturas,” *Tese Mestrado. Fac. Ciências e Tecnol. Univ. Nov. Lisboa*, 2012.

[5] M. Correia, “Optimização Robusta de Estruturas,” *Tese Mestrado. Fac. Ciências e Tecnol. da Univ. Nov. Lisboa*, 2014.

[6] J. Marmeleiro, “PROAES – Programa de análise de estruturas para aplicação em otimização e análise de fiabilidade,” *Tese Mestrado. Fac. Ciências e Tecnol. da Univ. Nov. Lisboa*, 2017.

[7] M. A, Lucien. Schmit, Jr. Hirokazu, “Approximation Concepts For Efficinet Structural Sysnthesis,” *Nasa Rep. CR - 2552, NASA*, pp. 4–8, 1976.

[8] R. A. Gellatly and R. H. Gallagher, “A Procedure for Automated Minimum Weight Structural Design: Part I : Theoretical Basis,” *Aeronaut. Q.*, pp. 216–230, 1965.

[9] R. A. Gellatly and R. H. Gallagher, “A Procedure for Automated Minimum Weight Structural Design:Part II : Applications,” *Aeronaut. Quaterly*, no. March, pp. 332–342, 1965.

[10] W. J. Dwyer, R. K. Emerton, P. L. Sabatelu, and G. A. Corporation, “An Automated Procedure For The Optimization Of Practical Aerospace Structures,” *Rep. AFFDL-TR-70-118*, vol. ll, 1971.

[11] J. S. Arora, “Introduction to Optimum Design,” *Elsevier 225 Wyman Street, Waltham, MA 02451, USA*, 2012.

[12] L. M. G. Drummond and B. F. Svaiter, “A steepest descent method for vector

optimization," *J. Comput. Appl. Math.*, vol. 175, pp. 395–414, 2005.

[13] TROND STEIHAUG, "The Conjugate Gradient Method and Trust Regions in Large Scale Optimization," no. 3, pp. 626–637, 1983.

[14] R. M. Freund, "Newton's Method for Unconstrained Optimization," *Massachusetts Inst. Technol.*, pp. 1–12, 2004.

[15] P. T. Boggs and J. W. Tolle, "Sequential Quadratic Programming," *National Institute of Standards and Technology*, pp. 1–4, 1996.

[16] K. Svanberg, "Some modelling aspects for the Matlab implementation of MMA," *Optimization and Systems Theory Department of Mathematics. KTH, SE–10044 Stock.*, pp. 1–5, 2004.

[17] F. Glover, "Tabu Search -Part I and Part II," *J. Comput.*, vol. 1 and 2, December 2018, pp. 190–206, 4-32, 1989.

[18] H. Szu and R. Hartley, "Fast simulated annealing," *Phys. Lett. A*, vol. 122, no. 3–4, pp. 157–162, 1987.

[19] J. H. Holland, "Genetic Algotrithms," *Scientific American*, vol. 267, no. 1, pp. 66–73, 1992.

[20] E. Zitzler, "Evolutionary Algorithms for Multiobjective Optimization," Swiss Federal Institute ofTechnology Zurich, 1999.

[21] F. Glover, M. Laguna, and R. Marti, "Fundamentals of scatter search and path relinking" *Univ. Color.*, vol. 29, no. 3, p. 653, 2000.

[22] B. Peterson, C. Soderberg, "A new method for mapping optimization problems onto neural networks," *Int. J. Neural Syst.*, vol. 1, p. 20, 1989.

[23] R. M, C. Hirsch, P.Meneses, M. Pardalos, "Global optimization by continuous grasp," *Springer - Verlag*, p. 12, 2006.

[24] H. N, P. Mladenovic, "Variable Neighborhood Search," *Elsevier Sci.*, vol. 24, no. 1, p. 1097, 1997.

[25] M. Dorigo, M. Birattari, and T. St, "Ant Colony Optimization," *Univ. Libr. Bruxelles*, no. November, 2006.

[26] A. Lipowski and D. Lipowska, “Roulette-wheel selection via stochastic acceptance,” *Phys. A Stat. Mech. its Appl.*, vol. 391, no. 6, pp. 2193–2196, 2012.

[27] B. Miller and D. G. Systems, “Genetic Algorithms, Tournament Selection, and the Effects of Noise,” *Department of Computer Science University of Illinois at Urbana-Cllampaign , USA*, vol. 9, pp. 193–212, 1995.

[28] N. M. Razali and J. Geraghty, “Genetic algorithm performance with different selection strategiesin solving TSP,” *Proc. World Congr. Eng. 2011, WCE 2011*, vol. 2, pp. 1134–1139, 2011.

[29] “Yarpiz - Academic Source Codes and Tutorials.” Acedido em:7 de Abril de 2019, em https://yarpiz.com/.

[30] J. Marmeleiro, “Manual de Utilização Programa PROAES Versão 5.0,” *Fac. Ciências e Tecnol. Univ. Nov. Lisboa*, pp. 4–5, 2017.

[31] J. B. Cardoso, P. G. Coelho, “Parallel Direct Search in Structural Optimization,” *Fac. Ciências e Tecnol. Univ. Nov. Lisboa*, vol. 1960, pp. 1–15, 2011.

Anexo A

Ficheiros relativos ao Exemplo 1 – Treliça 11 Barras

A.1 Função objetivo - SQP

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Calcula a função objetivo do Problema 1 - Trelica 11 barras
%
%   Problema de minimização do Volume com constrangimentos de tensão
%
%   Dissertação de Mestrado de Daniel Paciência Ferreira
%   16/05/2019
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function f = z_obj_trelica_11_barras(x)
%
%x = 0.0334*ones(11,1)
[n,m]= size(x);
%
% Escreve o ficheiro opt2ef
fout= fopen('opt2ef','w');
fprintf(fout,'1\n');         % Calcular a performance volume
fprintf(fout,'%d\n',n);      % Definir o nº de variaveis
for i=1:n
    fprintf(fout,'%d,%18.12e\n', i, x(i)); % Identifico o valor da variavel
end
fprintf(fout,'1\n');         % Só é necessária uma performance
fprintf(fout,'12\n');        % Identificar o ID da performance volume
fclose(fout);
%
% Chama o PROAES
resultado= PROAES('trelica_11_barras','o');
%
% Lê o ficheiro ef2opt
fid= fopen('ef2opt','r');
tline = fgets(fid);     % Lê o tipo de análise
tline = fgets(fid);     % Lê o número de performances
tline = fgets(fid);     %
[data,count]= sscanf(tline,'%d, %f',inf);
if count == 2
    %
    % Lê o valor do volume
    f= data(2);
else
    fprintf('Erro na leitura de ef2opt\n');
end
fclose(fid);
%
end % of function
```

A.2 Função constrangimento – SQP

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Calcula os constrangimentos do Problema 1 - Trelica 11 barras
```

```
%
%   Problema de minimização do Volume com constrangimentos de tensão
%
%   Dissertacao de Mestrado de Daniel Paciência Ferreira
%   16/05/2019
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function c = z_con_trelica_11_barras( x )
%
%x = 0.0334*ones(11,1) ;
[n,m]= size(x);
%
% número de constrangimento de tensão (existe 1  por barra)
nc= 11;
c= zeros(nc,1);
SMAX= 100e6;
%
% Escreve o ficheiro opt2ef
fout= fopen('opt2ef','w');
fprintf(fout,'1\n');         % Calcular as performances tensão
fprintf(fout,'%d\n',n);      % Definir o nº de variáveis
for i=1:n
    fprintf(fout,'%d,%18.12e\n', i, x(i)); % Identifico o valor da variável
end
fprintf(fout,'11\n');        % São necessárias 11 performances
for i=1:nc
    fprintf(fout,'%d\n',i);  % escreve as 11 performances de tensão
end
fclose(fout);
%
% Chama o PROAES
resultado= PROAES('trelica_11_barras','o');
%
% Lê o ficheiro ef2opt
sx= zeros(n,1);         % Cria o vetor sx para guardar as tensões
fid= fopen('ef2opt','r');
tline = fgets(fid);     % Lê o tipo de análise
tline = fgets(fid);     % Lê o número de performances
for i=1:nc
    tline = fgets(fid);
    [data,count]= sscanf(tline,'%d, %f',inf);
    if count == 2
        %
        % Lê o valor da tensão que neste problema é o constrangimento
        sx(i)= data(2);
    else
        fprintf('Erro na leitura de ef2opt\n');
    end
end
fclose(fid);
%
%  Calcula o valor dos constrangimentos
%
for i= 1:nc
     if sx(i) >= 0
         c(i)= ( SMAX - sx(i) ) / SMAX;
     else
         c(i)= ( SMAX + sx(i) ) / SMAX;
     end
end
% end function
```

A.3 Gradiente da função objetivo – SQP

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Calcula o gradiente da função objetivo do Problema 1 - Treliça 11 barras
```

```
%
%   Problema de minimização do Volume com constrangimentos de tensão
%
%   Dissertação de Mestrado de Daniel Paciência Ferreira
%   16/05/2019
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function gradf = z_grad_obj_trelica_11_barras(x)
%
%x = 0.0334*ones(11,1);
[n,m]= size(x);
gradf= zeros(n,1);
%
% Escreve o ficheiro opt2ef
fout= fopen('opt2ef','w');
fprintf(fout,'3\n');         % Calcular a performance volume e as derivadas
fprintf(fout,'%d\n',n);      % Definir o nº de variáveis
for i=1:n
    fprintf(fout,'%d,%18.12e\n', i, x(i)); % Identifico o valor da variável
end
fprintf(fout,'1\n');         % Só é necessária uma performance
fprintf(fout,'12\n');        % Identificar o ID da performance volume
fclose(fout);
%
% Chama o PROAES
resultado= PROAES('trelica_11_barras','o');
%
% Lê o ficheiro ef2opt
fid= fopen('ef2opt','r');
tline = fgets(fid);     % Lê o tipo de análise
tline = fgets(fid);     % Lê o número de performances
tline = fgets(fid);     %
[data,count]= sscanf(tline,'%d, %f',inf);
if count == 2
    %
    % Lê o valor do volume
    f= data(2);
else
    fprintf('Erro na leitura de ef2opt\n');
end
%
% Lê as derivadas do volume em relação a variável área
tline = fgets(fid);
[data,count]= sscanf(tline,'%d',inf);
if count == 1
    NV= data(1);
end
if n ~= NV
    fprintf('****** Erro na leitura de ef2opt (número de variáveis != %d)\n',NV);
end
for i= 1:n
    tline = fgets(fid);
    [data,count]= sscanf(tline,'%d, %d, %f',inf);
    if count == 3
        gradf(i)= abs(data(3));
    end
end
fclose(fid);
%
end % of function
```

A.4 Gradiente da função constrangimento – SQP

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Calcula o gradiente dos constrangimentos do Problema 1 - Trelica 11 barras
```

```
%
%   Problema de minimização do Volume com constrangimentos de tensão
%
%   Dissertacao de Mestrado de Daniel Paciência Ferreira
%   16/05/2019
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function gradc = z_grad_con_trelica_11_barras( x )
%
%x = 0.0334*ones(11,1) ;
[n,m]= size(x);
%
% número de constrangimento de tensao (existe 1  por barra)
nc= 11;
gradc= zeros(nc,n);
SMAX= 100e6;
%
% Escreve o ficheiro opt2ef
fout= fopen('opt2ef','w');
fprintf(fout,'3\n');         % Calcular as performances tensao e os gradientes
fprintf(fout,'%d\n',n);      % Definir o nº de variaveis
for i=1:n
    fprintf(fout,'%d,%18.12e\n', i, x(i)); % Identifico o valor da variavel
end
fprintf(fout,'11\n');        % São necessárias 11 performances
for i=1:nc
    fprintf(fout,'%d\n',i);  % escreve as 11 performances de tensão
end
fclose(fout);
%
% Chama o PROAES
resultado= PROAES('trelica_11_barras','o');
%
% Lê o ficheiro ef2opt
sx= zeros(n,1);         % Cria o vector sx para guardar as tensoes
fid= fopen('ef2opt','r');
tline = fgets(fid);     % Lê o tipo de análise
tline = fgets(fid);     % Lê o número de performances
for i=1:nc
    tline = fgets(fid);
    [data,count]= sscanf(tline,'%d, %f',inf);
    if count == 2
        %
        % Lê o valor da tensão que neste problema é o constrangimento
        sx(i)= data(2);
    else
        fprintf('Erro na leitura de ef2opt\n');
    end
end
%
% Lê as derivadas das tensões em relação às variaveis área
tline = fgets(fid);
[data,count]= sscanf(tline,'%d',inf);
if count == 1
    NC= data(1);
end
if nc ~= NC
    fprintf('****** Erro na leitura de ef2opt (numero de performances != %d)\n',NC);
end
for i= 1:nc
    for j= 1:n
        tline = fgets(fid);
        [data,count]= sscanf(tline,'%d, %d, %f',inf);
        if count == 3
            if sx(i) >= 0
                gradc(i,j)= -data(3)/SMAX;
```

```
            else
                gradc(i,j)= data(3)/SMAX;
            end
        end
    end
end
fclose(fid);
%
end % function
```

A.5 Ficheiro de inicialização – SQP

```
%%%
```

```
[x,obj,info,iter,nf,lambda]=sqp(x0,ObjectiveFunction,[],IneqConsFunction,LB,UB,maxite
r,tol)
%
%
tf=time;
t=tf-ti;
fprintf('t= %.3f s\n',t);
% Fim do ficheiro
```

A.6 Função objetivo e Grandiente – MMA

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Calcula a função objetivo e suas derivadas
%   para o Problema 1 - Trelica 11 barras (MMA)
%
%   Problema de minimização do Volume com constrangimentos de tensão
%
%   Dissertacao de Mestrado de Daniel Paciência Ferreira
%   16/05/2019
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [f0val, df0dx] = myFun(x)
%
[n,m]= size(x);
df0dx= zeros(n,1);
%
% Escreve o ficheiro opt2ef
fout= fopen('opt2ef','w');
fprintf(fout,'3\n');         % Calcular a performance volume e as derivadas
fprintf(fout,'%d\n',n);      % Definir o nº de variaveis
for i=1:n
    fprintf(fout,'%d,%18.12e\n', i, x(i)); % Identifico o valor da variavel
end
fprintf(fout,'1\n');         % Só é necessária uma performance
fprintf(fout,'12\n');        % Identificar o ID da performance volume
fclose(fout);
%
% Chama o PROAES
resultado= PROAES('ztrelica','o');
%
% Lê o ficheiro ef2opt
fid= fopen('ef2opt','r');
tline = fgets(fid);     % Lê o tipo de análise
tline = fgets(fid);     % Lê o número de performances
tline = fgets(fid);     %
[data,count]= sscanf(tline,'%d, %f',inf);
if count == 2
    %
    % Lê o valor do volume
    f0val= data(2);
else
    fprintf('Erro na leitura de ef2opt\n');
end
%
% Lê as derivadas do volume
tline = fgets(fid);
[data,count]= sscanf(tline,'%d',inf);
if count == 1
    NV= data(1);
end
if n ~= NV
    fprintf('****** Erro na leitura de ef2opt (numero de variáveis != %d)\n',NV);
end
for i= 1:NV
```

```
    tline = fgets(fid);
    [data,count]= sscanf(tline,'%d, %d, %f',inf);
    if count == 3
        df0dx(i)= data(3);
    end
end
fclose(fid);
%
end % function
```

A.7 Função constrangimento e Gradiente – MMA

```
%%%
```

```
%
%  Calcula o valor dos constrangimentos
%
for i= 1:nc
     if sx(i) >= 0
         fval(i)= ( sx(i) - SMAX ) / SMAX;
     else
         fval(i)= ( -sx(i) - SMAX ) / SMAX;
     end
end
%
% Lê as derivadas das tensões em relação às variaveis área
tline = fgets(fid);
[data,count]= sscanf(tline,'%d',inf);
if count == 1
    NC= data(1);
end
if nc ~= NC
    fprintf('****** Erro na leitura de ef2opt (numero de performances != %d)\n',NC);
end
for i= 1:nc
    for j= 1:n
        tline = fgets(fid);
        [data,count]= sscanf(tline,'%d, %d, %f',inf);
        if count == 3
            if sx(i) >= 0
                dfdx(i,j)= data(3)/SMAX;
            else
                dfdx(i,j)= -data(3)/SMAX;
            end
        end
    end
end
fclose(fid);
%
end % function
```

A.8 Ficheiro de inicilialização – MMA

```
%%%
```

```
% Numero maximo de iteracoes
maxite= 20;
% Valor inicial das variaveis
xval  = 0.0334*ones(n,1);
% Lower and upper bounds on design variables
xmin  = 1e-8*ones(n,1);
xmax  = 0.0334*ones(n,1);
%
% Tolerancia de Convergencia
TOL_FVAL= 1e-12;
%
% Parametros internos do MMA
% ==========================
low   = xmin;
upp   = xmax;
df0dx= zeros(n,1);
df0dx2= 0*df0dx;
fval  = zeros(m,1);
dfdx= zeros(m,n);
dfdx2= 0*dfdx;
xold1 = xval;
xold2 = xval;
c = 1000*ones(m,1);
d = zeros(m,1);
a0 = 1;
a = zeros(m,1);
%
% Executa o PROAES para obter os valores iniciais da função
% objectivo, dos constrangimentos e das suas derivadas
% ==========================================================
iter = 0;
%
% Cálculo da funcao objectivo f0val
% =================================
[f0val, df0dx] = myFun(xval);
%
% Cálculo dos constrangimentos fval
% =================================
[fval, dfdx] = myCon(xval);
%
% Escreve os resultados da inicializacao
% ======================================
outvector = [iter f0val fval' xval']';
%
% Optimizacao
% ===========
BestCost=zeros(maxite,1); %%%%%%
itte = 0;
delta_f0val= f0val;
%
while (itte < maxite) && (delta_f0val > TOL_FVAL)
    iter = iter+1;
    itte = itte+1;
    %
    [xmma,ymma,zmma,lam,xsi,eta,mu,zet,s,low,upp] = ...
                    mmasub(m,n,iter,xval,xmin,xmax,xold1,xold2, ...
                           f0val,df0dx,df0dx2,fval,dfdx,dfdx2,low,upp,a0,a,c,d);
    %
    xold2 = xold1;
    xold1 = xval;
    xval = xmma;
    %
    % Executa o PROAES para obter os valores da função
    % objectivo, dos constrangimentos e das suas derivadas
    % para cada iteracao
    %
```

```
    % Cálculo da funcao objectivo f0val
    % ================================
    f0val_old= f0val;
    [f0val, df0dx] = myFun(xval);
    %
    % Verificação da Convergencia
    % ===========================
    delta_f0val= abs(f0val-f0val_old);
    %
    % Cálculo dos constrangimentos fval
    % ================================
    %
    [fval, dfdx] = myCon(xval);
    %
    outvector = [iter f0val fval' xval']' %%%%%% %
    %disp(['Iteration ' num2str(iter) ': Best Cost = ' num2str(f0val)]); %%%%%%
    BestCost(itte)=f0val; %%%%%%
end
%
tf=time;
t=tf-ti;
fprintf('t= %.3f s\n',t);
% end of file

%% Results %%%%%%
figure; %%%%%%
plot(BestCost,'LineWidth',2); %%%%%%
xlabel('Iteração'); %%%%%%
ylabel('Função Objetivo'); %%%%%%
grid on; %%%%%%
```

A.9 Função objetivo – GA REAL

```
%%%%%
```

```
resultado= PROAES('trelica_11_barras','o');
%
% Lê o ficheiro ef2opt
sx= zeros(n,1);
fid= fopen('ef2opt','r');
tline = fgets(fid);     % Lê o tipo de análise
tline = fgets(fid);     % Lê o número de performances
for i=1:n
    tline = fgets(fid);
    [data,count]= sscanf(tline,'%d, %f',inf);
    if count == 2
        %
        % Lê o valor da tensão
        sx(i)= abs(data(2));
    else
          fprintf('Erro na leitura de ef2opt\n');
    end
end
%
tline = fgets(fid);
if count == 2
    %
    % Lê o valor do volume
    [data,count]= sscanf(tline,'%d, %f',inf);
    volume= data(2);
    else
        fprintf('Erro na leitura de ef2opt\n');
end
fclose(fid);
%
%constrangimento de tensao (existe 1  por barra)
sxmax=100e6;
%
% Calcula a função de mérito a partir do volume e dos constrangimentos
g= 0;
for i=1:11
    if sx(i) > sxmax
        g = g + (sx(i) - sxmax)/sxmax;
    end
end
% Funcao objectivo penalizada
%
% Esta constante de penalidade em baixo tem de ser 'afinada' para o problema
Constante= 20;
%
y= volume + Constante * g ;
%
% Faz o print de função objectivo penalizada, volume, e tensao
% valores das areas que se encontram entre o lower e upper bounds.
tensao=max(sx);
if tensao > sxmax
    % Faz o print de solução inadmissível
    fprintf(' obj= %10.4e, v= %8.6f, smax= %10.4e, %9.7f, %9.7f, %9.7f, %9.7f, %9.7f,
%9.7f, %9.7f, %9.7f, %9.7f, %9.7f, %9.7f, VIOLADO\n',...
    y,volume,tensao,x(1),x(2),x(3),x(4),x(5),x(6),x(7),x(8),x(9),x(10),x(11));
else
    % Faz o print de solução admissível
    fprintf(' obj= %10.4e, v= %8.6f, smax= %10.4e, %9.7f, %9.7f, %9.7f, %9.7f, %9.7f,
%9.7f, %9.7f, %9.7f, %9.7f, %9.7f, %9.7f\n',...
    y,volume,tensao,x(1),x(2),x(3),x(4),x(5),x(6),x(7),x(8),x(9),x(10),x(11));
end
%
```

A.10 Ficheiro de inicialização – GA REAL

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Ficheiro de inicialização do problema - trelica 11 barras
%
%   Executa a optimizacao da trelica 11 barras com o Algoritmo Genético Real
%
%   Trabalho de TAME, 2018-2019
%
%   Necessita do ficheiro : myFun_real
%   Antes de executar verificar o seguinte através da janela de comandos:
%   >> pkg list (se faltar a toolbox statistics realizar a instalação em baixo)
%   >> pkg install -forge statistics (instala)
%   >> pkg load statistics (carrega a toolbox)
%
%   Dissertacao de Mestrado de Daniel Paciência Ferreira
%   20/05/2019
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Função Custo
CostFunction=@(x) myFun_real(x);
% Numero de variaveis de decião (genes)
nVar=11;
% Lower Bound - Limite inferior das variaveis
VarMin = 1e-6;
% Upper Bound - Limite superior das variaveis
VarMax = 0.0334;
% Numero máximo de gerações/Iterações.
MaxIt=80;
% Tamanho da população
nPop=40;
% Seleção 'Roulette Wheel/Tournament/Random'
ANSWER= 'Tournament';
% Percentagem de cruzamento
pc=0.7;
% Fator extra de variação para o cruzamento
gamma=0.4;
% Percentagem de mutação
pm=0.3;
% Taxa de mutação
mu=0.1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

A.11 Função objetivo - GA BINÁRIO

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Calcula a funcao objetivo e suas derivadas
%   para o Problema 1 - Trelica 11 barras (Algoritmo Genetico)
%
%   Problema de minimização do Volume com constrangimentos de tensão
%
%   Dissertacao de Mestrado de Daniel Paciência Ferreira
%   20/05/2019
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [y]=myFun_binary(x)
%
% Tabela com as áreas A dos perfis HEB retirado do
% formulário da disciplina
Tabela= [ 1.06e-2;...     %HEB240
1.18e-2; ...              %HEB160
1.31e-2; ...              %HEB200
1.49e-2; ...              %HEB220
1.61e-2; ...              %HEB240
1.71e-2; ...              %HEB260
```

```
1.81e-2; ...              %HEB280
1.98e-2; ...              %HEB300
2.18e-2; ...              %HEB320
2.39e-2; ...              %HEB340
2.54e-2; ...              %HEB360
2.70e-2; ...              %HEB400
2.86e-2; ...              %HEB450
3.06e-2; ...              %HEB500
3.21e-2; ...              %HEB650
3.34e-2]; ...             %HEB550

%
% Cálculo da funcao objectivo
n= 11; % número de barras
y= 0;
%
% Escreve o ficheiro opt2ef
fout= fopen('opt2ef','w');
fprintf(fout,'1\n');    % Calcular a performance volume e tensão
fprintf(fout,'%d\n',n); % Definir o nº de variaveis
%
ii=zeros(n,1);
for i=1:n % Descodifica cada segmento do cromossoma binário em um nº inteiro
    indice=1+x((i-1)*4+1)*8+x((i-1)*4+2)*4+x((i-1)*4+3)*2+x((i-1)*4+4);
    ii(i)=indice; % Guarda num vector o inteiro que representa
                  % o nº de posição do perfil na tabela
    fprintf(fout,'%d, %18.12e\n', i, Tabela(indice));
end
fprintf(fout,'12\n');       % 11 performances de tensao e uma de volume
for i=1:n
    fprintf(fout,'%d\n',i); % Identificar o ID de cada performance tensao
end
fprintf(fout,'%d\n',n+1);   % Identificar o ID da performance volume
fclose(fout);
%
% Chama o PROAES
resultado= PROAES('trelica_11_barras','o');
%
% Lê o ficheiro ef2opt
sx= zeros(n,1);
fid= fopen('ef2opt','r');
tline = fgets(fid);     % Lê o tipo de análise
tline = fgets(fid);     % Lê o número de performances
for i=1:n
    tline = fgets(fid);
    [data,count]= sscanf(tline,'%d, %f',inf);
    if count == 2
        %
        % Lê o valor da tensão
        sx(i)= abs(data(2));
    else
          fprintf('Erro na leitura de ef2opt\n');
    end
end
%
tline = fgets(fid);
[data,count]= sscanf(tline,'%d, %f',inf);
if count == 2
    %
    % Lê o valor do volume
    volume= data(2);
    else
        fprintf('Erro na leitura de ef2opt\n');
end
fclose(fid);
%
```

```
%constrangimento de tensao (existe 1  por barra)
sxmax=100e6; %
%
% Calculo a função de mérito a partir do volume e dos constrangimentos
g= 0;
for i=1:11
    if sx(i) > sxmax
        g = g + (sx(i) - sxmax)/sxmax;
    end
end
% Funcao objectivo penalizada
%
% Esta constante de penalidade em baixo tem de ser 'afinada' para o problema
Constante= 20;
%
y= volume + Constante * g ;
%
% Faz o print de função objectivo penalizada, volume, tensao e
% valores dos índices de posição dos perfis na tabela
tensao=max(sx);
if tensao > sxmax
    % Faz o print de solução inadmissível
    fprintf(' obj= %e, v= %f, smax= %e, %2d %2d %2d %2d %2d %2d %2d %2d %2d %2d %2d
VIOLADO\n',...

y,volume,tensao,ii(1),ii(2),ii(3),ii(4),ii(5),ii(6),ii(7),ii(8),ii(9),ii(10),ii(11));
else
    % Faz o print de solução admissível
    fprintf(' obj= %e, v= %f, smax= %e, %2d %2d %2d %2d %2d %2d %2d %2d %2d %2d
%2d\n',...

y,volume,tensao,ii(1),ii(2),ii(3),ii(4),ii(5),ii(6),ii(7),ii(8),ii(9),ii(10),ii(11));
end
%
```

A.12 Ficheiro de inicialização - GA BINÁRIO

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Ficheiro de inicialização do problema da trelica 11 barras
%
%   Executa a optimizacao da trelica 11 barras com o Algoritmo Genético Binario
%
%   Trabalho de TAME, 2018-2019
%
%   Necessita do ficheiro : myFun_binary
%   Antes de executar verificar o seguinte através da janela de comandos:
%   >> pkg list (se faltar a toolbox statistics realizar a instalação em baixo)
%   >> pkg install -forge statistics (instala)
%   >> pkg load statistics (carrega a toolbox)
%
%   Dissertacao de Mestrado de Daniel Paciência Ferreira
%   20/05/2019
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Função custo
CostFunction=@(x) myFun_binary(x);
% Número de variaveis de projeto codificadas
N= 11;
% Número de variaveis de decisão (genes)
nVar=44;
% Número máximo de gerações/Iterações.
MaxIt=30;
% Tamanho da população
nPop=88;
```

```
% Seleção 'Roulette Wheel/Tournament/Random'
ANSWER= 'Tournament';
% Percentagem de cruzamento
pc=0.8;
% Percentagem de Mutação
pm=0.05;
% Taxa de Mutação
mu=0.02;
%%%
```

```
    % Lê o valor do volume
    [data,count]= sscanf(tline,'%d, %f',inf);
    volume= data(2);
    else
       fprintf('Erro na leitura de ef2opt\n');
end
fclose(fid);
%
%constrangimento de tensao (existe 1  por barra)
sxmax=100e6;
%
% Calcula a função de mérito a partir do volume e dos constrangimentos
g= 0;
for i=1:11
    if sx(i) > sxmax
        g = g + (sx(i) - sxmax)/sxmax;
    end
end
% Funcao objectivo penalizada
%
% Esta constante de penalidade em baixo tem de ser 'afinada' para o problema
Constante= 20;
%
f= volume + Constante * g ;
%
% Faz o print de função objectivo penalizada, volume, e tensao
% valores das areas que se encontram entre o lower e upper bounds.

% DESCOMENTAR CASO EU QUERIA VER O VALOR QUE AS VARIAVEIS ESTAO A TOMAR

tensao=max(sx);
%if tensao > sxmax
%    % Faz o print de solução inadmissível
%    fprintf(' obj= %10.4e, v= %8.6f, smax= %10.4e, %9.7f, %9.7f, %9.7f, %9.7f,
%9.7f, %9.7f, %9.7f, %9.7f, %9.7f, %9.7f, %9.7f, VIOLADO\n',...
%    f,volume,tensao,x(1),x(2),x(3),x(4),x(5),x(6),x(7),x(8),x(9),x(10),x(11));
%else
%    % Faz o print de solução admissível
%    fprintf(' obj= %10.4e, v= %8.6f, smax= %10.4e, %9.7f, %9.7f, %9.7f, %9.7f,
%9.7f, %9.7f, %9.7f, %9.7f, %9.7f, %9.7f, %9.7f\n',...
%    f,volume,tensao,x(1),x(2),x(3),x(4),x(5),x(6),x(7),x(8),x(9),x(10),x(11));
%end
%
end
%
```

A.14 Ficheiro de inicialização - PSO

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Ficheiro de inicialização do problema da trelica 11 barras
%
%   Executa a optimizacao da trelica 11 barras com o Algoritmo PSO
%
%   Trabalho de TAME, 2018-2019
%
%   Necessita do ficheiro : myFun
%   Antes de executar verificar o seguinte através da janela de comandos:
%   >> pkg list (se faltar a toolbox statistics realizar a instalação em baixo)
%   >> pkg install -forge statistics (instala)
%   >> pkg load statistics (carrega a toolbox)
%
%   Dissertacao de Mestrado de Daniel Paciência Ferreira
%   20/05/2019
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Função custo
CostFunction=@(x) myFun(x);
% Numero das variaveis de decisão
nVar=11;
% Dimensão da matriz das variaveis de decisão
VarSize=[1 nVar];
% Lower Bound of Variables
VarMin=0.0001;
% Upper Bound of Variables
VarMax=0.0334;
% Numero maximo de iterações
MaxIt=250;
% Dimenção da população (Dimensão do enxame)
nPop=30;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Anexo B

Ficheiros relativos ao Exemplo 2 – Treliça 38 Barras

B.1 Função objetivo – SQP

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%     Calcula a funcao objetivo do Problema 2 - Trelica 38 barras
%
%     Problema de minimização do Deslocamento no nó 14
%     com constrangimentos de Volume
%
%     Dissertacao de Mestrado de Daniel Paciência Ferreira
%     16/05/2019
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [D14] = z_obj_trelica_38_barras(x)
%x = 0.01*ones(38,1);
[n,m]= size(x);
%
% Escreve o ficheiro opt2ef
fout= fopen('opt2ef','w');
fprintf(fout,'1\n');       % Calcular a performance Deslocamento
fprintf(fout,'%d\n',n);    % Definir o nº de variaveis
for i=1:n
    fprintf(fout,'%d,%18.12e\n', i, x(i)); % Identifico o valor da variavel
end
fprintf(fout,'1\n');  % Sé é necessaria uma performance
fprintf(fout,'2\n');  % Identificar o ID da performance Deslocamento
fclose(fout);
%
% Chama o PROAES
resultado= PROAES('trelica38','o');
%
% Lê o ficheiro ef2opt
fid= fopen('ef2opt','r');
tline = fgets(fid);     % Lê o tipo de análise
tline = fgets(fid);     % Lê o número de performances
tline = fgets(fid);     %
[data,count]= sscanf(tline,'%d, %f',inf);
if count == 2
    %
    % Lê o valor do deslocamento no nó 14
    D14= abs(data(2));
else
    fprintf('Erro na leitura de ef2opt\n');
end
fclose(fid);
end % of function
```

B.2 Função constrangimento – SQP

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Calcula os constrangimentos do Problema 2 - Trelica 38 barras
%
%   Problema de minimização do deslocamento com constrangimentos de volume
%
%   Dissertacao de Mestrado de Daniel Paciência Ferreira
%   16/05/2019
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function [c] = z_con_trelica_38_barras( x )
n= 38;                                % numero de barras
VOL_INICIAL= 2.231370849898e-1 ;    % Volume inicial
FRA_VOL= 0.10;                        % Percentagem do volume
VOL_MAXIMO = VOL_INICIAL * FRA_VOL; % Valor maximo do volume
%x = 0.01*ones(38,1);                % Valor inicial das variaveis
%
%
% Escreve o ficheiro opt2ef
fout= fopen('opt2ef','w');
fprintf(fout,'1\n');    % Calcular as performances Volume
fprintf(fout,'%d\n',n); % Definir o nº de variaveis
%
for i=1:n
    fprintf(fout,'%d, %18.12e\n', i, x(i)); % Identifico o valor da variavel
end
fprintf(fout,'1\n'); % 1 performance de volume o que é o constrangimento
fprintf(fout,'1\n'); % Identificar o ID da performance Volume
fclose(fout);
%
%
% Chama o PROAES
resultado = PROAES('trelica38','o');
%
% Lê o ficheiro ef2opt
fid= fopen('ef2opt','r');
tline = fgets(fid);     % Lê o tipo de análise
tline = fgets(fid);     % Lê o número de performances
tline = fgets(fid);
[data,count]= sscanf(tline,'%d, %f',inf);
if count == 2
        %
        % Lê o valor do volume
   volume= data(2);
else
    fprintf('Erro na leitura de ef2opt\n');
end
fclose(fid);
%
%  Calcula o valor dos constrangimentos
%
c = [(VOL_MAXIMO - volume)];
end % end function
```

B.3 Gradiente da função objetivo – SQP

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%     Calcula o gradiente da funcao objetivo do Problema 2 - Trelica 38 barras
%
%     Problema de minimização do Deslocamento no nó 14
%     com constrangimentos de Volume
%
%     Dissertacao de Mestrado de Daniel Paciência Ferreira
%     16/05/2019
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [gradf] = z_grad_obj_trelica_38_barras(x)
%x = 0.01*ones(38,1);
[n,m]= size(x);
gradf= zeros(n,1);
%
% Escreve o ficheiro opt2ef
fout= fopen('opt2ef','w');
fprintf(fout,'3\n');   % Calcular a performance Deslocamento e as suas derivadas
```

```
fprintf(fout,'%d\n',n);% Definir o nº de variaveis
for i=1:n
    fprintf(fout,'%d,%18.12e\n', i, x(i)); % Identifico o valor da variavel
end
fprintf(fout,'1\n');  % Sé é necessaria uma performance
fprintf(fout,'2\n');  % Identificar o ID da performance Deslocamento
fclose(fout);
%
% Chama o PROAES
resultado= PROAES('trelica38','o');
%
% Lê o ficheiro ef2opt
fid = fopen('ef2opt','r');
tline = fgets(fid);     % Lê o tipo de análise
tline = fgets(fid);     % Lê o número de performances
tline = fgets(fid);     %
[data,count]= sscanf(tline,'%d, %f',inf);
if count == 2
    %
    % Lê o valor do deslocamento no nó 14
    D14= abs(data(2));
else
    fprintf('Erro na leitura de ef2opt\n');
end
%
% Lê as derivadas do deslocamento em relação a variavel área
tline = fgets(fid);
[data,count]= sscanf(tline,'%d',inf);
if count == 1
    NV= data(1);
end
if n ~= NV
    fprintf('****** Erro na leitura de ef2opt (numero de variáveis != %d)\n',NV);
end
for i= 1:n
    tline = fgets(fid);
    [data,count]= sscanf(tline,'%d, %d, %f',inf);
    if count == 3
        gradf(i)= -(data(3));
    end
end
fclose(fid);
%
end % of function
```

B.4 Gradiente da função constrangimento – SQP

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Calcula o gradiente dos constrangimentos do Problema 2 - Trelica 38 barras
%
%   Problema de minimização do Volume com constrangimentos de tensão
%
%   Dissertacao de Mestrado de Daniel Paciência Ferreira
%   16/05/2019
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [gradc] = z_grad_con_trelica_38_barras( x )
n= 38;                              % numero de barras
x = 0.01*ones(38,1);                % Valor inicial das variaveis
% número de constrangimento
nc=1;
gradc= zeros(nc,n);
VOL_INICIAL= 2.231370849898e-1 ;    % Volume inicial
FRA_VOL= 0.10;                      % Percentagem do volume
```

```
VOL_MAXIMO = VOL_INICIAL * FRA_VOL; % Valor maximo do volume
%
%
% Escreve o ficheiro opt2ef
fout= fopen('opt2ef','w');
fprintf(fout,'3\n');    % Calcular as performances volume e as derivadas
fprintf(fout,'%d\n',n); % Definir o nº de variaveis
%
for i=1:n
    fprintf(fout,'%d, %18.12e\n', i, x(i)); % Identifico o valor da variavel
end
fprintf(fout,'1\n'); % 1 performance de volume o que é o constrangimento
fprintf(fout,'1\n'); % Identificar o ID da performance Volume
fclose(fout);
%
%
% Chama o PROAES
resultado = PROAES('trelica38','o');
%
% Lê o ficheiro ef2opt
fid= fopen('ef2opt','r');
tline = fgets(fid);     % Lê o tipo de análise
tline = fgets(fid);     % Lê o número de performances
tline = fgets(fid);
[data,count]= sscanf(tline,'%d, %f',inf);
if count == 2


   volume= data(2); % Lê o valor do volume
else
    fprintf('Erro na leitura de ef2opt\n');
end
%fclose(fid);
%
%  Calcula o valor dos constrangimentos
%
c = [(VOL_MAXIMO - volume)];
%
% Lê as derivadas do volume em relação a variavel área
tline = fgets(fid);
[data,count]= sscanf(tline,'%d',inf);
if count == 1
    NV= data(1);
end
if n ~= NV
    fprintf('****** Erro na leitura de ef2opt (numero de variáveis != %d)\n',NC);
end
for i= 1:n
    tline = fgets(fid);
    [data,count]= sscanf(tline,'%d, %d, %f',inf);
    if count == 3
        gradc(i)= (-data(3));
    end
end
fclose(fid);
%
end
```

B.5 Ficheiro de inicialização – SQP

```
%%%
```

```
%
%   Executa a optimizacao da trelica 38 barras com o SQP
%
%   Necessita dos 4 ficheiros:  z_obj_trelica_38_barras
%                               grad_obj_trelica_38_barras
%                               z_con_trelica_38_barras
%                               grad_con_trelica_38_barras
%
%   Dissertacao de Mestrado de Daniel Paciência Ferreira
%   16/05/2019
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc
clear all
ti=time;
%
% GRAD= 0/1, SQP calcula o gradiente/  é fornecido
GRAD= 1;
%
display('38 bar truss optimization');
% Valores das variaveis na iteração = 0
x0 = 0.01*ones(38,1);
%
if GRAD
    ObjectiveFunction = cell(2,1);
    % Função objetivo
    ObjectiveFunction(1)= @z_obj_trelica_38_barras;
    % Gradiente da função objetivo
    ObjectiveFunction(2)= @z_grad_obj_trelica_38_barras;
    %
    IneqConsFunction = cell(2,1);
    % Função constrangimento
    IneqConsFunction(1)= @z_con_trelica_38_barras;
    % Gradiente da função constrangimento
    IneqConsFunction(2)= @z_grad_con_trelica_38_barras;
else
    % Função objetivo
    ObjectiveFunction= @z_obj_trelica_38_barras;
    % Lembrete 1: a função do constrangimento deve devolver um vetor coluna, i.e.,
    % c = [ ; ... ; ]
    % Lembrete 2: o formato do constrangimento de desigualdade é g >= 0, i.e.,
    % constrangimento do tipo "maior ou igual" a
    IneqConsFunction= @z_con_trelica_38_barras;
end
% Lower and upper bounds - Limite superior e inferior das variaveis
LB=0.00000001*ones(38,1);
UB=0.01*ones(38,1);
% Numero máximo de iterações; use [] para um valor padrão
maxiter=100;
% Tolerância à convergência; use [] para um valor padrão
tol=1e-12;
% lança 'sqp' do Octave:
[x,obj,info,iter,nf,lambda]=sqp(x0,ObjectiveFunction,[],IneqConsFunction,LB,UB,maxite
r,tol)
%
tf=time;
t=tf-ti;
fprintf('t= %.3f s\n',t);
% Fim do ficheiro
```

B.6 Função objetivo e Grandiente – MMA

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Calcula a funcao objetivo e suas derivadas
%   para o Problema 2 - Trelica 38 barras (MMA)
%
```

```
%   Problema de minimização do deslocamento com constrangimentos de volume
%
%   Dissertacao de Mestrado de Daniel Paciência Ferreira
%   16/05/2019
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [f0val, df0dx] = myFun(x)
%
[n,m]= size(x);
df0dx= zeros(n,1);
%
% Escreve o ficheiro opt2ef
fout= fopen('opt2ef','w');
fprintf(fout,'3\n');    % Calcular a performance deslocamento e as derivadas
fprintf(fout,'%d\n',n); % Definir o nº de variaveis
for i=1:n
    fprintf(fout,'%d, %18.12e\n', i, x(i));  % Identifico o valor da variavel
end
fprintf(fout,'1\n'); % Só é necessária uma performance
fprintf(fout,'2\n'); % Identificar o ID da performance deslocamento
fclose(fout);
%
% Chama o PROAES
resultado= PROAES('trelica38','o');
%
% Lê o ficheiro ef2opt
fid= fopen('ef2opt','r');
tline = fgets(fid);     % Lê o tipo de análise
tline = fgets(fid);     % Lê o número de performances
tline = fgets(fid);     %
[data,count]= sscanf(tline,'%d, %f',inf);
if count == 2
    %
    % Lê o valor do deslocamento
    f0val= -data(2);
else
    fprintf('Erro na leitura de ef2opt\n');
end
%
% Lê as derivadas do deslocamento em relação à variavel área
tline = fgets(fid);
[data,count]= sscanf(tline,'%d',inf);
if count == 1
    NV= data(1);
end
if n ~= NV
    fprintf('****** Erro na leitura de ef2opt (numero de variáveis != %d)\n',NV);
end
for i= 1:NV
    tline = fgets(fid);
    [data,count]= sscanf(tline,'%d, %d, %f',inf);
    if count == 3
        df0dx(i)= -data(3);
    end
end
fclose(fid);
%
end % of function
```

B.7 Função constrangimento e Gradiente – MMA

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Calcula a funcao objetivo e suas derivadas
%   para o Problema 2 - Trelica 38 barras (MMA)
%
```

```
%   Problema de minimização do deslocamento com constrangimentos de volume
%
%   Dissertacao de Mestrado de Daniel Paciência Ferreira
%   16/05/2019
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [fval, dfdx] = myCon(x)
%
% Volume inicial
VOL_INICIAL= 2.231370849898e-1 ;
% Percentagem do volume
FRA_VOL= 0.10;
% Valor maximo do volume
VOL_MAXIMO = VOL_INICIAL * FRA_VOL;
%
[n,m]= size(x);
dfdx= zeros(n,1);
%
% Escreve o ficheiro opt2ef
fout= fopen('opt2ef','w');
fprintf(fout,'3\n');    % Calcular a performance deslocamento e as derivadas
fprintf(fout,'%d\n',n); % Definir o nº de variaveis
for i=1:n
    fprintf(fout,'%d, %18.12e\n', i, x(i));  % Identifico o valor da variavel
end
fprintf(fout,'1\n'); % Só é necessária uma performance
fprintf(fout,'1\n'); % Identificar o ID da performance deslocamento
fclose(fout);
%
% Chama o PROAES
resultado= PROAES('trelica38','o');
%
% Lê o ficheiro ef2opt
fid= fopen('ef2opt','r');
tline = fgets(fid);     % Lê o tipo de análise
tline = fgets(fid);     % Lê o número de performances
tline = fgets(fid);     %
[data,count]= sscanf(tline,'%d, %f',inf);
if count == 2
    %
    % Lê o valor do volume
    fval = (data (2)-VOL_MAXIMO);
else
    fprintf('Erro na leitura de ef2opt\n');
end
%
% Lê as derivadas do deslocamento em relação à variavel área
tline = fgets(fid);
[data,count]= sscanf(tline,'%d',inf);
if count == 1
    NV= data(1);
end
if n ~= NV
    fprintf('****** Erro na leitura de ef2opt (numero de variáveis != %d)\n',NV);
end
for i= 1:NV
    tline = fgets(fid);
    [data,count]= sscanf(tline,'%d, %d, %f',inf);
    if count == 3
         dfdx(i)= data(3);
    end
end
fclose(fid);
%
end % of function
```

B.8 Ficheiro de inicialização – MMA

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Ficheiro de inicialização do problema da trelica 38 barras
%
%   Executa a optimizacao da trelica 38 barras com o MMA
%
%
%   Necessita dos 2 ficheiros: myFun
%                              myCon
%
%   Dissertacao de Mestrado de Daniel Paciência Ferreira
%   16/05/2019
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc
clear all
warning ("off");
ti=time;
%
% Inicializacao
% =============
% Numero de variaveis
n = 38;
% Numero de constrangimentos
m = 1;
% Numero maximo de iteracoes
maxite= 400;
% Valor inicial das variaveis
xval  = 0.01*ones(n,1);
% Lower and upper bounds on design variables
xmin  = 1e-8*ones(n,1);
xmax  = 0.01*ones(n,1);
% Volume inicial
VOL_INICIAL= 2.231370849898e-1 ;
% Percentagem do volume
FRA_VOL= 0.10;
% Valor maximo do volume
VOL_MAXIMO = VOL_INICIAL * FRA_VOL;
%x = 0.01*ones(38,1);
%
% Parametros internos do MMA
% ==========================
low   = xmin;
upp   = xmax;
df0dx= zeros(n,1);
df0dx2= 0*df0dx;
fval  = zeros(m,1);
dfdx= zeros(m,n);
dfdx2= 0*dfdx;
xold1 = xval;
xold2 = xval;
c = 1000*ones(m,1);
d = zeros(m,1);
a0 = 1;
a = zeros(m,1);
%
% Executa o PROAES para obter os valores iniciais da função
% objectivo, dos constrangimentos e das suas derivadas
% ====================================================
iter = 0;
%
% Cálculo da funcao objectivo f0val
% =================================
```

```
[f0val, df0dx] = myFun(xval);
%
% Cálculo dos constrangimentos fval
% ================================
[fval, dfdx] = myCon(xval);
%
% Escreve os resultados da inicializacao
% =====================================
outvector = [iter f0val fval' xval']';
%
% Optimizacao
% ===========
BestCost=zeros(maxite,1); %%%%%%
itte = 0;
delta_f0val= f0val;
TOL_FVAL= 1e-12;
%
while (itte < maxite) && (delta_f0val > TOL_FVAL)
    iter = iter+1;
    itte = itte+1;
    %
    [xmma,ymma,zmma,lam,xsi,eta,mu,zet,s,low,upp] = ...
                    mmasub(m,n,iter,xval,xmin,xmax,xold1,xold2, ...
                           f0val,df0dx,df0dx2,fval,dfdx,dfdx2,low,upp,a0,a,c,d);
    %
    xold2 = xold1;
    xold1 = xval;
    xval = xmma;
    %
    % Executa o PROAES para obter os valores da função
    % objectivo, dos constrangimentos e das suas derivadas
    % para cada iteracao
    %
    % Cálculo da funcao objectivo f0val
    % ================================
    f0val_old= f0val;
    [f0val, df0dx] = myFun(xval);
    %
    % Verificação da Convergencia
    % ===========================
    delta_f0val= abs(f0val-f0val_old);
    %
    % Cálculo dos constrangimentos fval
    % ================================
    %
    [fval, dfdx] = myCon(xval);
    %
    outvector = [iter f0val fval' xval']' %%%%%% % comentar caso retirar o grafico
    %disp(['Iteration ' num2str(iter) ': Best Cost = ' num2str(f0val)]); %%%%%%
    BestCost(itte)=f0val; %%%%%%
end
%
tf=time;
t=tf-ti;
fprintf('t= %.3f s\n',t);
%
% end of file

%% Results %%%%%%
figure; %%%%%%
plot(BestCost,'LineWidth',2); %%%%%%
xlabel('Iteration'); %%%%%%
ylabel('Best Cost'); %%%%%%
grid on; %%%%%%
```

B.9 Função objetivo – GA REAL

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Calcula a funcao objetivo e suas derivadas
%   para o Problema 2 - Trelica 38 barras (Algoritmo Genetico)
%
%   Problema de minimização do Deslocamento com constrangimentos de Volume
%
%   Dissertacao de Mestrado de Daniel Paciência Ferreira
%   29/05/2019
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [y] = myFun_real(x)
%
% Volume inicial
VOL_INICIAL= 2.231370849898e-1 ;
% Percentagem do volume
FRA_VOL= 0.10;
% Valor maximo do volume
VOL_MAXIMO = VOL_INICIAL * FRA_VOL;
%
% Cálculo da funcao objetivo
p= 2;   % numero de performances
n = 38; % numero de barras
%
% Escreve o ficheiro opt2ef
fout= fopen('opt2ef','w');
fprintf(fout,'1\n');    % Calcular a performance deslocamento e tensao
fprintf(fout,'%d\n',n); % Definir o nº de variaveis
for i=1:n
    fprintf(fout,'%d, %18.12e\n', i, x(i));
end
fprintf(fout,'2\n'); % 1 performances de tensao e 1 de deslocamento
for i= 1:p
    fprintf(fout,'%d\n',i); % Identificar o ID das performances
end
fclose(fout);
%
% Chama o PROAES
resultado = PROAES('trelica38','o');
%
% Lê o ficheiro ef2opt
fid= fopen('ef2opt','r');
tline = fgets(fid);     % Lê o tipo de análise
tline = fgets(fid);     % Lê o número de performances
tline = fgets(fid);
[data,count]= sscanf(tline,'%d, %f',inf);
if count == 2
    %
    % Lê o valor do volume
    volume= data(2);
else
    fprintf('Erro na leitura de ef2opt\n');
end
tline = fgets(fid);
[data,count]= sscanf(tline,'%d, %f',inf);
if count == 2
    %
    % Lê o valor do deslocamento no nó 14
    D14= -data(2);
else
    fprintf('Erro na leitura de ef2opt\n');
end
fclose(fid);
%
```

```
% Calcula a função de mérito a partir do deslocamento e do constrangimento de volume
g= 0;
if volume > VOL_MAXIMO
   g = (volume - VOL_MAXIMO)/(VOL_MAXIMO);
end
% Funcao objectivo penalizada
%
% Esta constante de penalidade em baixo tem de ser 'afinada' para o problema
Constante=0.00003; %=0.00009 o valor da constante deve ser da mesma ordem de grandeza
%que o deslocamento
%
y= D14 + Constante * g;
%
%
% Faz o print de função objectivo penalizada, deslocamento, e volume
% valores das areas que se encontram entre o lower e upper bounds.
if volume > VOL_MAXIMO
    % Faz o print de solução inadmissível
    fprintf(' obj= %10.4e, d= %10.4e, v= %10.4e,VIOLADO\n',...
    y,D14,volume);
else
    % Faz o print de solução admissível
    fprintf(' obj= %10.4e, d= %10.4e, v= %10.4e,\n',...
    y,D14,volume);
end
end
%
fclose all
```

B.10 Ficheiro de inicialização – GA REAL

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Ficheiro de inicialização do problema da trelica 38 barras
%
%   Executa a optimizacao da trelica 38 barras com o Algoritmo Genético Real
%
%   Trabalho de TAME, 2018-2019
%
%   Necessita do ficheiro : myFun_real
%   Antes de executar verificar o seguinte através da janela de comandos:
%   >> pkg list (se faltar a toolbox statistics realizar a instalação em baixo)
%   >> pkg install -forge statistics (instala)
%   >> pkg load statistics (carrega a toolbox)
%
%   Dissertacao de Mestrado de Daniel Paciência Ferreira
%   29/05/2019
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Função custo
CostFunction=@(x)myFun_real(x);
% Numero de variaveis de decisão
nVar=38;
% Lower Bound 1e-8 - Limite inferior das variaveis
VarMin = 0.00000001;
% Upper Bound 1e-2 - Limite superior das variaveis
VarMax = 0.01;
% Número máximo de gerações /Iterações
MaxIt=100;
% Dimensão da população
nPop=76;
% Seleção 'Roulette Wheel/Tournament/Random'
ANSWER= 'Tournament';
% Percentagem de cruzamento
pc=0.7;
```

```
% Fator extra de variação para o cruzamento
gamma=0.4;
% Percentagem de mutação
pm=0.3;
% Taxa de mutação
mu=0.1;
%%%
```

```
    if count == 2
    %
    % Lê o valor do deslocamento no nó 14
      deslocamento = abs(data(2));
    else
       fprintf('Erro na leitura de ef2opt\n');
    end
    fclose(fid);
%
% Constrangimento de Volume
%
VOL_INICIAL= 2.231370849898e-1 ;    % pode ser retirado de ef2opt
FRA_VOL= 0.10;                      % Percentagem do volume
VOL_MAXIMO = VOL_INICIAL * FRA_VOL; % VALOR DO VOLUME MAXIMO
%
% Calcula a função de mérito a partir do deslocamento e dos constrangimentos
%
g= 0;
if volume > VOL_MAXIMO
   g = (volume - VOL_MAXIMO)/VOL_MAXIMO; %
end
%
% Funcao objectivo penalizada
%
%
% Esta constante de penalidade em baixo tem de ser 'afinada' para o problema
Constante=0.00002; % =0.00009 o valor da constante deve ser da mesma ordem de
                   % grandeza que o deslocamento
%
D14 = deslocamento + Constante * g;
%
%
endfunction
```

B.12 Ficheiro de inicialização – PSO

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Ficheiro de inicialização do problema da trelica 38 barras
%
%   Executa a optimizacao da trelica 38 barras com o Algoritmo PSO
%
%   Trabalho de TAME, 2018-2019
%
%   Necessita do ficheiro : myFun
%   Antes de executar verificar o seguinte através da janela de comandos:
%   >> pkg list (se faltar a toolbox statistics realizar a instalação em baixo)
%   >> pkg install -forge statistics (instala)
%   >> pkg load statistics (carrega a toolbox)
%
%   Dissertacao de Mestrado de Daniel Paciência Ferreira
%   29/05/2019
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Função Custo
CostFunction=@(x) myFun(x);
% Número de variaveis de decisão
nVar=38;
% Dimensão da matriz das variaveis de decisão
VarSize=[1 nVar];
% Lower Bound - limite inferior das variaveis
VarMin=0.00000001;
% Upper Bound - Limite superior das variaveis
VarMax=0.01;
% Numero máximo de iterações
MaxIt=500;
```

```
% Dimensão da população (Dimensão do enxame)
nPop=100;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Anexo C

Ficheiros relativos ao Exemplo 3 – Europatir

C.1 Função objetivo – SQP

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%     Calcula a funcao objetivo do Problema 3 - Europa Tir
%
%     Problema de minimização da massa
%     com constrangimentos de rotação
%
%     Dissertacao de Mestrado de Daniel Paciência Ferreira
%     09/07/2019
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [m] = myFun(x)
%
fid= fopen('variaveis.txt','w');
fprintf(fid,'R, 2, %20.15e\n',x(1));
fprintf(fid,'R, 3, %20.15e\n',x(2));
fprintf(fid,'R, 4, %20.15e\n',x(3));
fprintf(fid,'R, 8, %20.15e\n',x(4));
fprintf(fid,'R, 9, %20.15e\n',x(5));
fprintf(fid,'R, 16, %20.15e\n',x(6));
fprintf(fid,'R, 19, %20.15e\n',x(7));
fprintf(fid,'R, 13, %20.15e\n',x(8)*6);
fprintf(fid,'R, 14, %20.15e\n',x(8)*4);
fprintf(fid,'X1=%20.15e\n',x(9));
fprintf(fid,'X2=%20.15e\n',x(10));
fprintf(fid,'X3=%20.15e\n',x(11));
fprintf(fid,'X4=%20.15e\n',x(12));
fclose(fid);
%
% Executa o Ansys
system('"C:\Program Files\ANSYS Student\ANSYS Inc\v182\ansys\bin\winx64\ANSYS...
182.exe" -b -p aa_t_i -i optimizacao.txt -o optimizacao.out');
%
%
% Lê o valor da massa
%
fid= fopen('resultados.txt','r');
tline = fgets(fid); % Lê o titulo massa (Não interessa)
tline = fgets(fid); % Lê o valor da massa
[data]= sscanf(tline,'%f',inf);
m = data(1);
%fprintf('massa= %25.15f\n',m);
fclose(fid);
end % of function
```

C.2 Função constrangimento – SQP

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Calcula os constrangimentos do Problema 3 - Europa Tir
%
%   Problema de minimização da Massa com constrangimentos de Rotação
%
%   Dissertacao de Mestrado de Daniel Paciência Ferreira
%   09/07/2019
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function c = myCon(x)
%
nd = 2;           % numero de deslocamentos
d= zeros(nd,1);   % vetor que guarda o valor dos deslocamentos
% escreve o ficheiro (variaveis.txt)
%
fid= fopen('variaveis.txt','w');
fprintf(fid,'R, 2, %20.15e\n',x(1));
fprintf(fid,'R, 3, %20.15e\n',x(2));
fprintf(fid,'R, 4, %20.15e\n',x(3));
fprintf(fid,'R, 8, %20.15e\n',x(4));
fprintf(fid,'R, 9, %20.15e\n',x(5));
fprintf(fid,'R, 16, %20.15e\n',x(6));
fprintf(fid,'R, 19, %20.15e\n',x(7));
fprintf(fid,'R, 13, %20.15e\n',x(8)*6);
fprintf(fid,'R, 14, %20.15e\n',x(8)*4);
fprintf(fid,'X1=%20.15e\n',x(9));
fprintf(fid,'X2=%20.15e\n',x(10));
fprintf(fid,'X3=%20.15e\n',x(11));
fprintf(fid,'X4=%20.15e\n',x(12));
fclose(fid);
%
% Executa o Ansys
system('"C:\Program Files\ANSYS Student\ANSYS Inc\v182\ansys\bin\winx64\ANSYS...
182.exe" -b -p aa_t_i -i optimizacao.txt -o optimizacao.out');
%
% Lê o valor dos deslocamentos
%
fid= fopen('resultados.txt','r');
tline = fgets(fid);     % Lê o titulo massa (lixo)
tline = fgets(fid);     % Lê o valor da massa (Não tem interesse para o deslocamento)
tline = fgets(fid);     % Lê o titulo deslocamentos
for i=1:nd
    tline = fgets(fid);
    [data]= sscanf(tline,'%f',inf);
        %
        % Lê o valor dos deslocamentos
        d(i)= data(1);
end
fclose(fid);
%
% deslocamento maximo do chassis
dtmax = 180;
%
% calculo dos constrangimentos
%
c= 1 + ( d(2) - d(1) ) / dtmax ;
end
% end function
```

C.3 Gradiente da função objetivo – SQP

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%     Calcula o gradiente da funcao objetivo do Problema 3 - Europa Tir
%
%     Problema de minimização da massa
%     com constrangimentos de rotação
%
%     Dissertacao de Mestrado de Daniel Paciência Ferreira
%     09/07/2019
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [dfdx] = grad_myFun(x)
% numero de variaveis
n=12;
```

```
dfdx=zeros(n,1);
%
% Escreve o ficheiro variaveis.txt
%
fid= fopen('variaveis.txt','w');
fprintf(fid,'R, 2, %20.15e\n',x(1));
fprintf(fid,'R, 3, %20.15e\n',x(2));
fprintf(fid,'R, 4, %20.15e\n',x(3));
fprintf(fid,'R, 8, %20.15e\n',x(4));
fprintf(fid,'R, 9, %20.15e\n',x(5));
fprintf(fid,'R, 16, %20.15e\n',x(6));
fprintf(fid,'R, 19, %20.15e\n',x(7));
fprintf(fid,'R, 13, %20.15e\n',x(8)*6);
fprintf(fid,'R, 14, %20.15e\n',x(8)*4);
fprintf(fid,'X1=%20.15e\n',x(9));
fprintf(fid,'X2=%20.15e\n',x(10));
fprintf(fid,'X3=%20.15e\n',x(11));
fprintf(fid,'X4=%20.15e\n',x(12));
fclose(fid);
%
% Executa o Ansys
system('"C:\Program Files\ANSYS Student\ANSYS Inc\v182\ansys\bin\winx64\ANSYS...
182.exe" -b -p aa_t_i -i optimizacao.txt -o optimizacao.out');
%
%
% Lê o valor da massa
%
fid= fopen('resultados.txt','r');
tline = fgets(fid); % Lê o titulo massa (Não interessa)
tline = fgets(fid); % Lê o valor da massa
[data]= sscanf(tline,'%f',inf);
massa_inicial = data(1);
fclose(fid);
%
% Realiza o calculo das derivadas por diferenças finitas
%=======================================================
% Perturba as variavel
%
DELTA= 0.000000001;
for ivar=1:n
          x_delta= x;
          x_delta(ivar)= x_delta(ivar)*(1+DELTA);
          %
          % Escreve o ficheiro com as variaveis perturbadas
          %
          fid= fopen('variaveis.txt','w');
          fprintf(fid,'R, 2, %20.15e\n',x_delta(1));
          fprintf(fid,'R, 3, %20.15e\n',x_delta(2));
          fprintf(fid,'R, 4, %20.15e\n',x_delta(3));
          fprintf(fid,'R, 8, %20.15e\n',x_delta(4));
          fprintf(fid,'R, 9, %20.15e\n',x_delta(5));
          fprintf(fid,'R, 16, %20.15e\n',x_delta(6));
          fprintf(fid,'R, 19, %20.15e\n',x_delta(7));
          fprintf(fid,'R, 13, %20.15e\n',x_delta(8)*6);
          fprintf(fid,'R, 14, %20.15e\n',x_delta(8)*4);
          fprintf(fid,'X1=%20.15e\n',x_delta(9));
          fprintf(fid,'X2=%20.15e\n',x_delta(10));
          fprintf(fid,'X3=%20.15e\n',x_delta(11));
          fprintf(fid,'X4=%20.15e\n',x_delta(12));
          fclose(fid);
          % Executa o Ansys
          system('"C:\Program Files\ANSYS Student\ANSYS...
          Inc\v182\ansys\bin\winx64\ANSYS182.exe" -b -p aa_t_i -i optimizacao.txt...
          -o optimizacao.out');
          %
          % Lê o valor da massa perturbado
```

```
          %
          fid= fopen('resultados.txt','r');
          tline = fgets(fid); % Lê o titulo massa (Não interessa)
          tline = fgets(fid); % Lê o valor da massa
          [data]= sscanf(tline,'%f',inf);
          m_delta = data(1);
          fclose(fid);
          %
          % Calcula  dfdx
          dfdx(ivar)= (m_delta-massa_inicial)/(x(ivar)*DELTA);
end
%fprintf('dfdx= %25.15f\n',dfdx);
%
end % of function
```

C.4 Gradiente da função constrangimento – SQP

```
%%%
```

```
fid= fopen('resultados.txt','r');
tline = fgets(fid);     % Lê o titulo massa (lixo)
tline = fgets(fid);     % Lê o valor da massa (Não tem interesse para o deslocamento)
tline = fgets(fid);     % Lê o titulo deslocamentos
for i=1:nd
    tline = fgets(fid);
    [data]= sscanf(tline,'%f',inf);
        %
        % Lê o valor do deslocamento
        d(i)= data(1);
end
fclose(fid);
% deslocamento maximo do chassis
%
dtmax = 180;
%
% Calculo o valor dos constrangimentos
%
% deslocamento maximo do chassis
c = 1 + ( d(2) - d(1) ) / dtmax ;
%
% Perturba as variavel
%
d_delta = zeros(nd,1);   % vetor que guarda o valor dos deslocamentos
DELTA = 0.000000001;
for ivar = 1:n
          x_delta = x;
          x_delta(ivar)= x_delta(ivar)*(1+DELTA);
          %
          % Escreve o ficheiro com as variaveis perturbadas
          %
          fid= fopen('variaveis.txt','w');
          fprintf(fid,'R, 2, %20.15e\n',x_delta(1));
          fprintf(fid,'R, 3, %20.15e\n',x_delta(2));
          fprintf(fid,'R, 4, %20.15e\n',x_delta(3));
          fprintf(fid,'R, 8, %20.15e\n',x_delta(4));
          fprintf(fid,'R, 9, %20.15e\n',x_delta(5));
          fprintf(fid,'R, 16, %20.15e\n',x_delta(6));
          fprintf(fid,'R, 19, %20.15e\n',x_delta(7));
          fprintf(fid,'R, 13, %20.15e\n',x_delta(8)*6);
          fprintf(fid,'R, 14, %20.15e\n',x_delta(8)*4);
          fprintf(fid,'X1=%20.15e\n',x_delta(9));
          fprintf(fid,'X2=%20.15e\n',x_delta(10));
          fprintf(fid,'X3=%20.15e\n',x_delta(11));
          fprintf(fid,'X4=%20.15e\n',x_delta(12));
          fclose(fid);
          %
          % Executa o Ansys
          system('"C:\Program Files\ANSYS Student\ANSYS...
          Inc\v182\ansys\bin\winx64\ANSYS182.exe" -b -p aa_t_i -i optimizacao.txt...
          -o optimizacao.out');
          %
          % Lê o valor dos deslocamentos
          %
          fid= fopen('resultados.txt','r');
          tline = fgets(fid);     % Lê o titulo massa (lixo)
          tline = fgets(fid);     % Lê o valor da massa (Não tem interesse para o
          %                       % deslocamento)
          tline = fgets(fid);     % Lê o titulo deslocamentos
          for i=1:nd
              tline = fgets(fid);
              [data]= sscanf(tline,'%f',inf);
              %
              % Lê o valor do deslocamento
              d_delta(i) = data(1);
          end
```

```
          fclose(fid);
          %
          %
          % Define o constrangimento
          %
          c_delta = 1 + ( d_delta(2) - d_delta(1) ) / dtmax ;
          %
          % diferenças finitas
          %
          dfcdx(ivar)= (c_delta-c)/(x(ivar)*DELTA);
end
%
%fprintf('dfcdx= %25.15f\n',dfcdx);
end
% end function
```

C.5 Ficheiro de inicialização – SQP

```
%%%
```

```
end
% Lower and upper bounds - Limite superior e inferior das variaveis
LB=[5;12;12;3;4;2;5;0.001;180;200;142;24];
UB=[10;15;15;4.5;8;6;10;1;240;350;186;44];
% Numero maximo de iterações; use [] para um valor padrão
maxiter=80;
% Tolerância à convergência; use [] para um valor padrão
tol=1e-12;
% lança 'sqp' do Octave:
[x,obj,info,iter,nf,lambda]=sqp(x0,ObjectiveFunction,[],IneqConsFunction,LB, ...
UB,maxiter,tol)
%
tf=time;
t=tf-ti;
fprintf('t= %.3f s\n',t);
% Fim do ficheiro
```

C.6 Função objetivo e Grandiente – MMA

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%     Calcula a funcao objetivo e suas derivadas do Problema 3 - Europa Tir
%
%     Problema de minimização da massa
%     com constrangimentos de rotação
%
%     Dissertacao de Mestrado de Daniel Paciência Ferreira
%     09/07/2019
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [f0val, df0dx] = myFun(x)
%
[n,m]= size(x);
df0dx= zeros(n,1);
%
% Escreve o ficheiro variaveis.txt
%
fid= fopen('variaveis.txt','w');
fprintf(fid,'R, 2, %20.15e\n',x(1));
fprintf(fid,'R, 3, %20.15e\n',x(2));
fprintf(fid,'R, 4, %20.15e\n',x(3));
fprintf(fid,'R, 8, %20.15e\n',x(4));
fprintf(fid,'R, 9, %20.15e\n',x(5));
fprintf(fid,'R, 16, %20.15e\n',x(6));
fprintf(fid,'R, 19, %20.15e\n',x(7));
fprintf(fid,'R, 13, %20.15e\n',x(8)*6);
fprintf(fid,'R, 14, %20.15e\n',x(8)*4);
fprintf(fid,'X1=%20.15e\n',x(9));
fprintf(fid,'X2=%20.15e\n',x(10));
fprintf(fid,'X3=%20.15e\n',x(11));
fprintf(fid,'X4=%20.15e\n',x(12));
fclose(fid);
%
% Executa o Ansys
system('"C:\Program Files\ANSYS Student\ANSYS Inc\v182\ansys\bin\winx64\ANSYS...
182.exe" -b -p aa_t_i -i optimizacao.txt -o optimizacao.out');
%
%
% Lê o valor da massa
%
fid= fopen('resultados.txt','r');
tline = fgets(fid); % Lê o titulo massa (Não interessa)
tline = fgets(fid); % Lê o valor da massa
[data]= sscanf(tline,'%f',inf);
f0val = data(1);
```

```
fclose(fid);
%
% Realiza o calculo das derivadas por diferenças finitas
%=====================================================
% Perturba as variavel
%
DELTA= 0.001;
for ivar=1:n
          x_delta= x;
          x_delta(ivar)= x_delta(ivar)*(1+DELTA);
          %
          % Escreve o ficheiro com as variaveis perturbadas
          %
          fid= fopen('variaveis.txt','w');
          fprintf(fid,'R, 2, %20.15e\n',x_delta(1));
          fprintf(fid,'R, 3, %20.15e\n',x_delta(2));
          fprintf(fid,'R, 4, %20.15e\n',x_delta(3));
          fprintf(fid,'R, 8, %20.15e\n',x_delta(4));
          fprintf(fid,'R, 9, %20.15e\n',x_delta(5));
          fprintf(fid,'R, 16, %20.15e\n',x_delta(6));
          fprintf(fid,'R, 19, %20.15e\n',x_delta(7));
          fprintf(fid,'R, 13, %20.15e\n',x_delta(8)*6);
          fprintf(fid,'R, 14, %20.15e\n',x_delta(8)*4);
          fprintf(fid,'X1=%20.15e\n',x_delta(9));
          fprintf(fid,'X2=%20.15e\n',x_delta(10));
          fprintf(fid,'X3=%20.15e\n',x_delta(11));
          fprintf(fid,'X4=%20.15e\n',x_delta(12));
          fclose(fid);
          % Executa o Ansys
          system('"C:\Program Files\ANSYS Student\ANSYS...
          Inc\v182\ansys\bin\winx64\ANSYS182.exe" -b -p aa_t_i -i optimizacao.txt...
          -o optimizacao.out');
          %
          % Lê o valor da massa perturbado
          %
          fid= fopen('resultados.txt','r');
          tline = fgets(fid); % Lê o titulo massa (Não interessa)
          tline = fgets(fid); % Lê o valor da massa
          [data]= sscanf(tline,'%f',inf);
          m_delta = data(1);
          fclose(fid);
          %
          % Calcula  dfdx
          df0dx(ivar)= (m_delta-f0val)/(x(ivar)*DELTA);
end
%
end % of function
```

C.7 Função constrangimento e Gradiente – MMA

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Calcula os constrangimentos e suas derivadas do Problema 3 - Europa Tir
%
%   Problema de minimização da Massa com constrangimentos de Rotação
%
%   Dissertacao de Mestrado de Daniel Paciência Ferreira
%   09/07/2019
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [fval, dfdx] = myCon(x)
%
% numero de variaveis
%
n = 12;
```

```
dfdx=zeros(n,1);
%
% numero de deslocamentos
%
nd = 2;
%
% vetor que guarda o valor dos deslocamentos
%
d= zeros(nd,1);
% escreve o ficheiro variaveis.txt
##%
fid= fopen('variaveis.txt','w');
fprintf(fid,'R, 2, %20.15e\n',x(1));
fprintf(fid,'R, 3, %20.15e\n',x(2));
fprintf(fid,'R, 4, %20.15e\n',x(3));
fprintf(fid,'R, 8, %20.15e\n',x(4));
fprintf(fid,'R, 9, %20.15e\n',x(5));
fprintf(fid,'R, 16, %20.15e\n',x(6));
fprintf(fid,'R, 19, %20.15e\n',x(7));
fprintf(fid,'R, 13, %20.15e\n',x(8)*6);
fprintf(fid,'R, 14, %20.15e\n',x(8)*4);
fprintf(fid,'X1=%20.15e\n',x(9));
fprintf(fid,'X2=%20.15e\n',x(10));
fprintf(fid,'X3=%20.15e\n',x(11));
fprintf(fid,'X4=%20.15e\n',x(12));
fclose(fid);
##%
% Executa o Ansys
system('"C:\Program Files\ANSYS Student\ANSYS Inc\v182\ansys\bin\winx64\ANSYS...
182.exe" -b -p aa_t_i -i optimizacao.txt -o optimizacao.out');
##%
% Lê o valor dos deslocamentos
%
fid= fopen('resultados.txt','r');
tline = fgets(fid);     % Lê o titulo massa (lixo)
tline = fgets(fid);     % Lê o valor da massa (Não tem interesse para o deslocamento)
tline = fgets(fid);     % Lê o titulo deslocamentos
for i=1:nd
    tline = fgets(fid);
    [data]= sscanf(tline,'%f',inf);
        %
        % Lê o valor do deslocamento
        d(i)= data(1);
end
fclose(fid);
% deslocamento maximo do chassis
%
dtmax = 180;
%
% Calculo o valor dos constrangimentos
%
% deslocamento maximo do chassis
fval =(( d(1) - d(2) ) / dtmax)-1;
%
% Perturba as variavel
%
d_delta = zeros(nd,1);   % vetor que guarda o valor dos deslocamentos
DELTA = 0.001;
for ivar = 1:n
          x_delta = x;
          x_delta(ivar)= x_delta(ivar)*(1+DELTA);
          %
          % Escreve o ficheiro com as variaveis perturbadas
          %
          fid= fopen('variaveis.txt','w');
          fprintf(fid,'R, 2, %20.15e\n',x_delta(1));
```

```
            fprintf(fid,'R, 3, %20.15e\n',x_delta(2));
            fprintf(fid,'R, 4, %20.15e\n',x_delta(3));
            fprintf(fid,'R, 8, %20.15e\n',x_delta(4));
            fprintf(fid,'R, 9, %20.15e\n',x_delta(5));
            fprintf(fid,'R, 16, %20.15e\n',x_delta(6));
            fprintf(fid,'R, 19, %20.15e\n',x_delta(7));
            fprintf(fid,'R, 13, %20.15e\n',x_delta(8)*6);
            fprintf(fid,'R, 14, %20.15e\n',x_delta(8)*4);
            fprintf(fid,'X1=%20.15e\n',x_delta(9));
            fprintf(fid,'X2=%20.15e\n',x_delta(10));
            fprintf(fid,'X3=%20.15e\n',x_delta(11));
            fprintf(fid,'X4=%20.15e\n',x_delta(12));
            fclose(fid);
            %
            % Executa o Ansys
            system('"C:\Program Files\ANSYS Student\ANSYS...
            Inc\v182\ansys\bin\winx64\ANSYS182.exe" -b -p aa_t_i -i optimizacao.txt...
            -o optimizacao.out');
            %
            % Lê o valor dos deslocamentos
            %
            fid= fopen('resultados.txt','r');
            tline = fgets(fid);     % Lê o titulo massa (lixo)
            tline = fgets(fid);     % Lê o valor da massa (Não tem interesse para o
            %                       % deslocamento)
            tline = fgets(fid);     % Lê o titulo deslocamentos
            for i=1:nd
                tline = fgets(fid);
                [data]= sscanf(tline,'%f',inf);
                %
                % Lê o valor do deslocamento
                d_delta(i) = data(1);
            end
            fclose(fid);
            %
            %
            % Define o constrangimento
            %
            c_delta = 1 + ( d_delta(2) - d_delta(1) ) / dtmax ;
            %
            % diferenças finitas
            %
            dfdx(ivar)= (c_delta-fval)/(x(ivar)*DELTA);
end
end % of function
```

C.8 Ficheiro de inicialização – MMA

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Ficheiro de inicialização do problema 3 - Europa Tir
%   Executa a optimizacao do chassis de um camião com o MMA
%
%   Necessita dos 4 ficheiros:  myFun.m
%                               myCon.m
%                               variavies.txt
%                               otimização.txt
%
%   Dissertacao de Mestrado de Daniel Paciência Ferreira
%   09/07/2019
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc
clear all
warning ("off");
```

```
display('                                          truck chassis optimization');
ti=time;
%
% Inicializacao
% =============
% Numero de variaveis
n = 12;
% Numero de constrangimentos
m = 1;
% Numero maximo de iteracoes
maxite= 250;
% Valor inicial das variaveis
xval  = [10;15;15;4.5;8;6;10;1;240;350;186;44];
% Lower and upper bounds das variaveis de projeto
xmin  = [5;12;12;3;4;2;5;0.001;180;200;142;24];
xmax  = [10;15;15;4.5;8;6;10;1;240;350;186;44];
% Valor maximo do volume
dtmax=180;
%
% Parametros internos do MMA
% ==========================
low   = xmin;
upp   = xmax;
df0dx= zeros(n,1);
df0dx2= 0*df0dx;
fval  = zeros(m,1);
dfdx= zeros(m,n);
dfdx2= 0*dfdx;
xold1 = xval;
xold2 = xval;
c = 1000*ones(m,1);
d = zeros(m,1);
a0 = 1;
a = zeros(m,1);
%
% Executa o PROAES para obter os valores iniciais da função
% objectivo, dos constrangimentos e das suas derivadas
% ========================================================
iter = 0;
%
% Cálculo da funcao objectivo f0val
% =================================
[f0val, df0dx] = myFun(xval);
%
% Cálculo dos constrangimentos fval
% =================================
[fval, dfdx] = myCon(xval);
%
% Escreve os resultados da inicializacao
% ======================================
outvector = [iter f0val fval' xval']';
%
% Optimizacao
% ===========
BestCost=zeros(maxite,1); %%%%%%
itte = 0;
delta_f0val= f0val;
TOL_FVAL= 1e-12;
%
while (itte < maxite) && (delta_f0val > TOL_FVAL)
    iter = iter+1;
    itte = itte+1;
    %
    [xmma,ymma,zmma,lam,xsi,eta,mu,zet,s,low,upp] = ...
                    mmasub(m,n,iter,xval,xmin,xmax,xold1,xold2, ...
                           f0val,df0dx,df0dx2,fval,dfdx,dfdx2,low,upp,a0,a,c,d);
```

```
    %
    xold2 = xold1;
    xold1 = xval;
    xval = xmma;
    %
    % Executa o PROAES para obter os valores da função
    % objectivo, dos constrangimentos e das suas derivadas
    % para cada iteracao
    %
    % Cálculo da funcao objectivo f0val
    % ================================
    f0val_old= f0val;
    [f0val, df0dx] = myFun(xval);
    %
    % Verificação da Convergencia
    % ==========================
    delta_f0val= abs(f0val-f0val_old);
    %
    % Cálculo dos constrangimentos fval
    % ================================
    %
    [fval, dfdx] = myCon(xval);
    %
    outvector = [iter f0val fval' xval']' %%%%%%
    %disp(['Iteration ' num2str(iter) ': Best Cost = ' num2str(f0val)]);%%%%%%
    BestCost(itte)=f0val; %%%%%%
end
%
tf=time;
t=tf-ti;
fprintf('t= %.3f s\n',t);
%
% end of file

%% Results %%%%%%
figure; %%%%%%
plot(BestCost,'LineWidth',2);
xlabel('Iteration'); %%%%%%
ylabel('Best Cost'); %%%%%%
grid on; %%%%%%
```

C.9 Função objetivo – GA REAL

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Calcula a funcao objetivo e suas derivadas
%   para o Problema 1 - Trelica 11 barras (Algoritmo Genetico)
%
%   Problema de minimização do Volume com constrangimentos de tensão
%
%   Dissertacao de Mestrado de Daniel Paciência Ferreira
%   20/05/2019
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [m]= myFun(x)
nd= 2; % numero de deslocamentos
d= zeros(nd,1);
% Escreve o ficheiro variaveis.txt com tratamento das variaveis
%
fid= fopen('variaveis.txt','w');
fprintf(fid,'R, 2, %20.15e\n',x(1)*5+5);
fprintf(fid,'R, 3, %20.15e\n',x(2)*3+12);
fprintf(fid,'R, 4, %20.15e\n',x(3)*3+12);
fprintf(fid,'R, 8, %20.15e\n',x(4)*1.5+3);
fprintf(fid,'R, 9, %20.15e\n',x(5)*4+4);
```

```
fprintf(fid,'R, 16, %20.15e\n',x(6)*4+2);
fprintf(fid,'R, 19, %20.15e\n',x(7)*5+5);
fprintf(fid,'R, 13, %20.15e\n',x(8)*6*0.999+0.001);
fprintf(fid,'R, 14, %20.15e\n',x(8)*4*0.999+0.001);
fprintf(fid,'X1=%20.15e\n',x(9)*60+180);
fprintf(fid,'X2=%20.15e\n',x(10)*150+200);
fprintf(fid,'X3=%20.15e\n',x(11)*44+142);
fprintf(fid,'X4=%20.15e\n',x(12)*22+24);
fclose(fid);
%
% Executa o Ansys
system('"C:\Program Files\ANSYS Student\ANSYS Inc\v182\ansys\bin\winx64\ANSYS...
182.exe" -b -p aa_t_i -i optimizacao.txt -o optimizacao.out');
%
%
% Lê o valor da massa
%
fid= fopen('resultados.txt','r');
tline = fgets(fid); % Lê o titulo massa (Não interessa)
tline = fgets(fid); % Lê o valor da massa
[data]= sscanf(tline,'%f',inf);
massa = data(1);
tline = fgets(fid); % Lê o titulo deslocamentos
for i=1:nd
    tline = fgets(fid);
    [data]= sscanf(tline,'%f',inf);
        %
        % Lê o valor do deslocamento
        d(i)= data(1);
end
fclose(fid);
%
% Constrangimento de Rotação
dt= ( d(1) - d(2));
%
% Deslocamento  maximo
dtmax=180;
%
% Calcula a função de mérito a partir do deslocamento e dos constrangimentos
%
g= 0;
if dt > dtmax
   g = (dt - dtmax)/dtmax; %
end
% Funcao objectivo penalizada
%
%
% Esta constante de penalidade em baixo tem de ser 'afinada' para o problema
Constante=200;    %  o valor da constante deve ser da mesma ordem de
                  %  grandeza que o deslocamento
%
m = massa + Constante * g;
%
%% Faz o print de função objectivo penalizada, volume, e tensao
%% valores das areas que se encontram entre o lower e upper bounds.
%tensao=max(sx);
%if tensao > sxmax
%    % Faz o print de solução inadmissível
%    fprintf(' obj= %10.4e, v= %8.6f, smax= %10.4e, %9.7f, %9.7f, %9.7f, %9.7f,
%9.7f, %9.7f, %9.7f, %9.7f, %9.7f, %9.7f, %9.7f, VIOLADO\n',...
%    y,volume,tensao,x(1),x(2),x(3),x(4),x(5),x(6),x(7),x(8),x(9),x(10),x(11));
%else
%    % Faz o print de solução admissível
%    fprintf(' obj= %10.4e, v= %8.6f, smax= %10.4e, %9.7f, %9.7f, %9.7f, %9.7f,
%9.7f, %9.7f, %9.7f, %9.7f, %9.7f, %9.7f, %9.7f\n',...
%    y,volume,tensao,x(1),x(2),x(3),x(4),x(5),x(6),x(7),x(8),x(9),x(10),x(11));
```

```
end
%
```

C.10 Ficheiro de inicialização – GA REAL

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Ficheiro de inicialização do problema da trelica 11 barras
%
%   Executa a optimizacao da trelica 11 barras com o Algoritmo Genético Real
%
%   Trabalho de TAME, 2018-2019
%
%   Necessita do ficheiro : myFun
%   Antes de executar verificar o seguinte através da janela de comandos:
%   >> pkg list (se faltar a toolbox statistics realizar a instalação em baixo)
%   >> pkg install -forge statistics (instala)
%   >> pkg load statistics (carrega a toolbox)
%
%   Dissertacao de Mestrado de Daniel Paciência Ferreira
%   20/05/2019
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Função custo
CostFunction=@(x) myFun(x);
% Numero de variaveis de decisão (genes)
nVar=12;
% Lower Bound - Limite inferior das variaveis
VarMin = 0;
% Upper Bound - Limite superior das variaves
VarMax = 1;
% Número máximo de gerações/Iterações.
MaxIt=80;
% Tamanho da população
nPop=40;
% Seleção 'Roulette Wheel/Tournament/Random'
ANSWER= 'Tournament';
% Percentagem de cruzamento
pc=0.7;
% Fator extra de variação para o cruzamento
gamma=0.4;
% Percentagem de mutação
pm=0.3;
% Taxa de mutação
mu=0.1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.11 Função objetivo – PSO

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%     Calcula a função objetivo, as suas derivadas e controla os
%     constrangimentos  do Problema 3 - Europa Tir
%
%     Problema de minimização da massa
%     com constrangimentos de rotação
%
%     Dissertacao de Mestrado de Daniel Paciência Ferreira
%     09/07/2019
```

```
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function m = myFun(x)
nd= 2; % numero de deslocamentos
d= zeros(nd,1);
% Escreve o ficheiro variaveis.txt com tratamento das variaveis
%
fid= fopen('variaveis.txt','w');
fprintf(fid,'R, 2, %20.15e\n',x(1)*5+5);
fprintf(fid,'R, 3, %20.15e\n',x(2)*3+12);
fprintf(fid,'R, 4, %20.15e\n',x(3)*3+12);
fprintf(fid,'R, 8, %20.15e\n',x(4)*1.5+3);
fprintf(fid,'R, 9, %20.15e\n',x(5)*4+4);
fprintf(fid,'R, 16, %20.15e\n',x(6)*4+2);
fprintf(fid,'R, 19, %20.15e\n',x(7)*5+5);
fprintf(fid,'R, 13, %20.15e\n',x(8)*6*0.999+0.001);
fprintf(fid,'R, 14, %20.15e\n',x(8)*4*0.999+0.001);
fprintf(fid,'X1=%20.15e\n',x(9)*60+180);
fprintf(fid,'X2=%20.15e\n',x(10)*150+200);
fprintf(fid,'X3=%20.15e\n',x(11)*44+142);
fprintf(fid,'X4=%20.15e\n',x(12)*22+24);
fclose(fid);
%
% Executa o Ansys
system('"C:\Program Files\ANSYS Student\ANSYS Inc\v182\ansys\bin\winx64\ANSYS...
182.exe" -b -p aa_t_i -i optimizacao.txt -o optimizacao.out');
%
%
% Lê o valor da massa
%
fid= fopen('resultados.txt','r');
tline = fgets(fid); % Lê o titulo massa (Não interessa)
tline = fgets(fid); % Lê o valor da massa
[data]= sscanf(tline,'%f',inf);
massa = data(1);
tline = fgets(fid); % Lê o titulo deslocamentos
for i=1:nd
    tline = fgets(fid);
    [data]= sscanf(tline,'%f',inf);
        %
        % Lê o valor do deslocamento
        d(i)= data(1);
end
fclose(fid);
%
% Constrangimento de Rotação
dt= ( d(1) - d(2));
%
% Deslocamento  maximo
dtmax=180;
%
% Calcula a função de mérito a partir do deslocamento e dos constrangimentos
%
g= 0;
if dt > dtmax
   g = (dt - dtmax)/dtmax; %
end
% Funcao objectivo penalizada
%
%
% Esta constante de penalidade em baixo tem de ser 'afinada' para o problema
Constante=2000;    %  o valor da constante deve ser da mesma ordem de
                  %  grandeza que o deslocamento
%
m = massa + Constante * g;
endfunction
```

C.12 Ficheiro de inicialização – PSO

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Ficheiro de inicialização do problema da treliça 11 barras
%
%   Executa a otimização da treliça 11 barras com o PSO
%
%   Trabalho de TAME, 2018-2019
%
%   Necessita do ficheiro : myFun
%   Antes de executar verificar o seguinte através da janela de comandos:
%   >> pkg list (se faltar a toolbox statistics realizar a instalação em baixo)
%   >> pkg install -forge statistics (instala)
%   >> pkg load statistics (carrega a toolbox)
%
%   Dissertação de Mestrado de Daniel Paciência Ferreira
%   20/05/2019
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Função custo
CostFunction=@(x) myFun(x);
% Números de variáveis de decisão
nVar=12;
% Dimensão da matriz das variaveis de decisão
VarSize=[1 nVar];
% Lower Bound of Variables
VarMin=0;
% Upper Bound of Variables
VarMax=1;
% Número máximo de iterações
MaxIt=500;      % Maximum Number of Iterations
% Dimensão da população (Dimensão do enxame)
nPop=48;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```