Universidade Federal da Paraíba
Centro de Informática
Programa de Pós-graduação em Modelagem Matemática e Computacional

TÉCNICA DE PROGRAMAÇÃO SEMIDEFINIDA POR ARCO VIÁVEL E APLICAÇÃO À MAXIMIZAÇÃO DA FREQUÊNCIA NATURAL DE ESTRUTURAS MECÂNICAS

Ligianne Nascimento Barros

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Modelagem Matemática e Computacional, UFPB, da Universidade Federal da Paraíba, como parte dos requisitos necessários à obtenção do título de Mestre em Modelagem Matemática e Computacional.

Orientador: José Miguel Aroztegui Massera

João Pessoa
Julho de 2017

Catalogação na publicação
Seção de Catalogação e Classificação

B277t Barros, Ligianne Nascimento.
Técnica de programação semidefinida por arco viável e aplicação à maximização da frequência natural de estruturas mecânicas / Ligianne Nascimento Barros. - João Pessoa, 2017.
72 f. : il.

Orientador: Prof. José Miguel Aroztegui Massera.
Dissertação (Mestrado) – UFPB/CI

1. Informática. 2. Programação semidefinida. 3. Estruturas mecânicas. 4. Otimização estrutural. I. Título.

UFPB/BC CDU – 004 (043)

TÉCNICA DE PROGRAMAÇÃO SEMIDEFINIDA POR ARCO VIÁVEL E APLICAÇÃO À MAXIMIZAÇÃO DA FREQUÊNCIA NATURAL DE ESTRUTURAS MECÂNICAS

Ligianne Nascimento Barros

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM MATEMÁTICA E COMPUTACIONAL (PPGMMC) DO CENTRO DE INFORMÁTICA DA UNIVERSIDADE FEDERAL DA PARAÍBA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM MODELAGEM MATEMÁTICA E COMPUTACIONAL.

Examinada por:

Prof. José Miguel Aroztegui Massera, D.Sc.

Prof. Roberto Quirino do Nascimento, D.Sc.

Prof. Sérgio de Carvalho Bezerra, D.Sc.

JOÃO PESSOA, PB – BRASIL
JULHO DE 2017

Aos meus familiares, pessoas que muito amo.

Agradecimentos

Primeiramente, agradeço a Deus que me deu força e trouxe luz em todos os momentos que precisei.

Agradeço a minha mãe, Fátima Dias, que em todos os momentos da minha vida sempre esteve incondicionalmente ao meu lado. Ao meu filho, Rhian, pela compreensão das minhas ausências durante os estudos. Aos meus irmãos, Lorena Simone Nascimento, Bruno Costa, Breno Costa, pela paciência e apoio demonstrados, torcendo sempre por minhas conquistas. A minha querida avó Olindina Dias *(in memoriam)*.

Ao querido Jair pelo apoio e incentivo.

Em especial, agradeço ao orientador Jóse Miguel Aroztegui, pelas orientações recebida e por toda atenção e paciência durante este processo.

Agradeço a todos os amigos do PPGMMC que de alguma forma estiveram ao meu lado durante esta caminhada, colaborando com palavras de apoio e incentivo.

Aos professores do programa, agradeço pelo aprendizado e experiências compartilhadas.

Aos professores Sérgio de Carvalho e Roberto Quirino por terem aceitado, prontamente, a participar da banca examinadora na certeza de colaborar com o estudo.

A todos que, de forma direta ou indireta, me ajudaram e contribuíram com o estudo, meus sinceros agradecimentos.

A CAPES pelo apoio financeiro.

Resumo da Dissertação apresentada ao PPGMMC/CI/UFPB como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

TÉCNICA DE PROGRAMAÇÃO SEMIDEFINIDA POR ARCO VIÁVEL E APLICAÇÃO À MAXIMIZAÇÃO DA FREQUÊNCIA NATURAL DE ESTRUTURAS MECÂNICAS

Ligianne Nascimento Barros

Julho/2017

Orientador: José Miguel Aroztegui Massera

Programa: Modelagem Matemática e Computacional

A partir das técnicas de programação semidefinida tornou-se possível otimizar de maneira eficiente problemas de engenharia estrutural. Neste trabalho, apresenta-se o problema programação semidefinida, suas aplicações e condições de otimalidade. Também é proposta uma nova técnica de pontos interiores com arcos viáveis para o problema não convexos de programação semidefinida. Apresenta-se o problema de maximização da frequência natural de estruturas. Este problema é reescrito como um problema de programação semidefinida não convexo. São apresentados alguns testes numéricos que comprovam a eficácia da nova técnica quando aplicados ao problema de maximização da frequência natural de estruturas.

Resumo da Dissertação apresentada ao PPGMMC/CI/UFPB como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

TÉCNICA DE PROGRAMAÇÃO SEMIDEFINIDA POR ARCO VIÁVEL E APLICAÇÃO À MAXIMIZAÇÃO DA FREQUÊNCIA NATURAL DE ESTRUTURAS MECÂNICAS

Ligianne Nascimento Barros

Julho/2017

Orientador: José Miguel Aroztegui Massera

Programa: Modelagem Matemática e Computacional

The semidefinite programming techniques allow to optimize structural engineering problems efficiently. In this work, we present the problem semidefinite programming, its applications and optimality conditions. In addition, a new method of interior points by feasible arcs for the non-convex problem of semidefinite programming is proposed. We present the problem of maximizing the natural frequency of structures. This problem is rewritten as a non-convex semidefinite programming problem. We present some numerical tests that prove the effectiveness of the new technique when applied to the problem of maximizing the natural frequency of structures.

Sumário

Lista de Figuras **x**

Lista de Tabelas **xi**

1 Introdução **1**

1.1 Introdução da pesquisa 1

1.2 Motivação 2

1.3 Objetivos 2

1.4 Resumo dos capítulos 2

2 Problema de otimização **4**

2.1 Notações e definições preliminares 4

2.2 Problema de Programação não linear e semidefinida 7

2.2.1 Restrições LMI/BMI 9

2.3 Condições de otimalidade 10

3 Problema de otimização estrutural **14**

3.1 Método de elementos finitos 15

3.1.1 Matriz de rigidez 19

3.1.2 Matriz de massa 21

3.1.3 Complacência 21

3.1.4 Frequência natural e fundamental de uma estrutura 22

3.1.5 Definição e propriedades da função $\lambda_{min}(x)$ 23

3.2 Formulação clássica dos problemas de mínimo volume e de máxima frequência 24

3.2.1 Dificuldades da formulação clássica 25

3.3 Formulação SDP 26

3.4 Relação entre os modelos SDP 27

4 Algoritmo de pontos interiores por arcos viáveis para programação semidefinida **29**

4.1 Notações e definições preliminares 30

4.2 FAIPA 30
4.2.1 Descrição do algoritmo 30
4.2.2 Cálculo da direção de descida e viável 31
4.2.3 Cálculo do arco 34
4.2.4 Busca linear 34
4.2.5 Atualizações do FAIPA 35
4.2.6 Algoritmo FAIPA 36
4.3 FAIPA-SDP 37
4.3.1 Descrição do algoritmo 37
4.3.2 Cálculo da direção de descida 38
4.3.3 Cálculo da direção de descida viável 39
4.3.4 Cálculo do arco 40
4.3.5 Busca linear 41
4.3.6 Atualizações do FAIPA-SDP 41
4.3.7 Algoritmo FAIPA-SDP 41

5 Testes numéricos **44**
5.1 Descrição dos problemas em termos do problema de programação semidefinida 44
5.1.1 Dados 46
5.1.2 Construção dos problemas de otimização 47
5.1.3 Geometria e condições de contorno 47
5.2 Execução dos algoritmos 48
5.2.1 Ponto viável 48
5.2.2 Resultados 48

6 Considerações finais **57**
6.1 Conclusões 57
6.2 Sugestões para trabalhos futuros 58

Referências Bibliográficas **59**

Lista de Figuras

3.1 Treliça bidimensional. 15
3.2 Geometria de uma barra. 16
3.3 Comportamento da barra quando submetida a uma força axial 16
3.4 Malha de elementos finitos. 16
3.5 Elemento (barra) bidimensional. 17
3.6 Vetor de força nodais. 20

4.1 Direção viável. 32
4.2 Direção de busca do FAIPA. 35
4.3 Direção de busca do FDIPA. 35
4.4 Busca linear de Armijo. 36

5.1 Geometria das treliças dos exemplos 1, 2, 3 e 4. As barras estão respresentadas por linhas tracejadas. 48
5.2 Solução ótima do exemplo 1. 50
5.3 Solução ótima do exemplo 2. 52
5.4 Solução ótima do exemplo 3. 53
5.5 Solução ótima do exemplo 4. 55

Lista de Tabelas

5.1 Convergência do FDIPA-SDP e FAIPA-SDP, menor autovalor de L0. 49
5.2 Número de iterações do FDIPA-SDP e FAIPA-SDP. 49
5.3 Exemplo 1: iterações da otimização. 51
5.4 Exemplo 2: Iterações da otimização. 52
5.5 Exemplo 3: Iterações da otimização. 54
5.6 Exemplo 4: Iterações da otimização. 56

Capítulo 1

Introdução

1.1 Introdução da pesquisa

A programação semidefinida representa uma importante teoria para resolver problemas de otimização em várias áreas do conhecimento, como por exemplo na Otimização Combinatória [1], Teoria de Controle [2] e Otimização Estrutural [3].

Originalmente, problemas de programação semidefinida são problemas convexos, onde se procura minimizar uma função linear com restrições nos autovalores de funções matriciais [3].

No final dos anos 80, com o desenvolvimento de métodos de pontos interiores, tornou-se possível a elaboração de técnicas computacionais para se trabalhar com programação linear e não linear [4], [5]. Posteriormente, algumas dessas técnicas de pontos interiores foram generalizados para programação semidefinida [3]. Esse fato levantou o interesse dos estudiosos da época e, desde então, foram propostos novos algoritmos que buscam determinar uma solução ótima para problemas de otimização a partir de: Direções Viáveis ou Arcos Viáveis. Algumas implementações para os algoritmos propostos já estão disponíveis, e seu uso encontra-se em processo de aperfeiçoamento [6] e [7].

Através da criação de famílias de algoritmos para programação semidefinida tornou-se possível a busca de soluções ótimas para problemas de otimização. Tais algoritmos são empregados com eficiência a problemas de grande porte, especialmente na otimização combinatória [1].

Entre as muitas razões para o estudo da programação semidefinida, podemos destacar que: suas restrições semidefinidas positivas aparecem de imediato em várias aplicações importantes; vários problemas de otimização convexa como por exemplo, programação linear e a programação quadrática pode ser apresentados como um problema de programação semidefinida. Os fatos citados anteriormente indicam que a programação semidefinida possui características que podem facilitar a busca

de soluções para os problemas de otimização. Dessa maneira, com a disponibilidade de software, pode-se resolver uma quantidade maior de problemas de otimização através do uso da programação semidefinida, o que representa, juntamente com a programação linear e quadrática, uma ferramenta básica para modelagem em otimização [5].

1.2 Motivação

A programação semidefinida se apresenta como uma área de estudo na qual se consegue integrar vários problemas padrões. Tem importância prática pois, encontra muitas aplicações em problemas de engenharia e análise combinatória.

A partir das ideias das técnicas de pontos interiores se pôde construir algoritmos inovadores para resolver problemas de otimização, como por exemplo, *Feasible Direction Interior Point Algorithm (FDIPA)* [8] e *Feasible Arc Interior Point Algorithm (FAIPA)* [5]. No que diz respeito a programação semidefinida, foi criado o *Feasible Direction Interior Point Algorithm for Semidefinite Programming (FDIPA-SDP)* [7] baseado nas ideias do *FDIPA*.

Este trabalho tem por interesse apresentar um algoritmo eficiente para resolver problemas de programação semidefinida não convexo com restrições fortemente não lineares. Para tanto, utilizamos o método de pontos interiores por arcos viáveis. Nossa análise se dará a partir do algoritmo *Feasible Arc Interior Point Algorithm (FAIPA)* proposto por Herkovits [5].

1.3 Objetivos

Os objetivos da nossa pesquisa :

- Dissertar sobre conceitos fundamentais da programação semidefinida e aplicações relacionadas a maximização da frequência natural de estruturas.
- Dissertar sobre algoritmos de pontos interiores por direções e/ou arcos viáveis desenvolvidos por Herskovits [5].
- Propor uma nova técnica de pontos interiores por arcos viáveis, empregando-a na resolução de problemas de programação semidefinida não convexa.

1.4 Resumo dos capítulos

Os próximos capítulos da dissertação estão organizados da seguinte forma:

No Capítulo 2, apresenta-se algumas notações e definições fundamentais para a formulação dos problemas gerais de programação não linear e de programação semidefinida, e também, são expostas as condições de otimalidade para cada um destes problemas.

O Capítulo 3 tem por objetivo apresentar a formulação tradicional e a formulação para programação semidefinida de um problema de otimização estrutural. Mostra-se também uma introdução ao método dos elementos finitos para a análise estrutural de Treliças.

No Capítulo 4 apresenta-se o algoritmo *Feasible Arc Interior Point Algorithm* (FAIPA) para resolver problemas de otimização não linear. É também proposta a nova técnica *Feasible Arc Interior Point Algorithm of Semidefinite Programming* (FAIPA-SDP), que é um algoritmo de pontos interiores por arcos viáveis, utilizado para resolver problemas de programação semidefinida. Este algoritmo é obtido a partir do algoritmo FDIPA-SDP, desenvolvido em [9] e baseando-se nas mesmas ideias do FDIPA.

No capítulo 5, a partir da implementação do algoritmo FAIPA-SDP, apresentam-se os testes numéricos que mostram a utilização deste algoritmo na resolução dos problemas descritos no capítulo 3.

O capítulo 6 dedica-se às conclusões e sugestões para trabalhos futuros.

Capítulo 2

Problema de otimização

Em um problema de otimização se procura determinar os valores extremos, isto é, o valor máximo ou mínimo que uma função pode assumir em um dado conjunto. Na formulação de um problema de otimização com restrições precisamos definir, em termos matemáticos, qual é o vetor de variáveis de projeto, qual é a função objetivo (função que se pretende minimizar ou maximizar) e quais são as funções de restrição do problema (limitações que devem ser verificada).

Neste capítulo, em primeiro lugar, serão apresentadas as notações adequadas e definições necessárias que são utilizadas nas formulações dos problemas de programação não linear e de programação semidefinida. Estes conceitos são úteis para a definição do problema de maximização da frequência natural, apresentado no próximo capítulo, assim como para a descrição das técnicas de otimização que serão apresentadas no capítulo 4.

Posteriormente, são tecidos comentários acerca dos tipos de restrições semidefinidas, isto é, as restrições de desigualdades de matriz linear e as restrições de desigualdades de matriz bilinear. Por fim, mostram-se as condições de otimalidade de *Karush-Kuhn-Tucker* para estes dois problemas.

2.1 Notações e definições preliminares

Nesta seção se apresentam definições e notações que serão utilizadas em diversas partes deste trabalho. Estes objetos são apresentados na literatura [4], [10], [11] e [2].

Para todo $n \in \mathbb{N}$, define-se $\overline{n} = \frac{1}{2}n(n+1)$.

O espaço de matrizes de tamanho $k \times n$ com coeficientes reais denota-se por $\mathbb{R}^{k\times n}$.

Seja $M \in \mathbb{R}^{k\times n}$. Denota-se por $sym(M)$ a parte simétrica de M e por $skw(M)$

sua parte antissimétrica, isto é:

$$sym(M) = \frac{M + M^\top}{2}, \quad skw(M) = \frac{M - M^\top}{2}.$$

Dada uma matriz $A \in \mathbb{R}^{k\times n}$, denotamos o núcleo de A por $Ker(A)$, sua entrada (i, j) representamos por a_{ij} e sua transposta por $A^\top$.

Chamaremos de I_n a matriz identidade de ordem $n \times n$.

Definição 1. *O espaço das matrizes simétricas com coeficientes reais e de tamanho $m \times m$ denota-se por $\mathbb{S}^m$. Os conjuntos das matrizes simétricas semidefinidas positivas e definidas positivas de ordem $k \times k$ serão representado por $\mathbb{S}^k_+$ e $\mathbb{S}^k_{++}$, respectivamente. Os conjuntos de matrizes simétricas semidefinidas negativas e definidas negativas podem ser obtidos de maneira similar.*

Definição 2. *A aplicação que organiza em um vetor os componentes de uma matriz $A \in \mathbb{S}^k$ é definida como*

$$svec : \mathbb{S}^k \to \mathbb{R}^{\bar{k}}$$

onde

$$svec(A) = \begin{bmatrix} a_{11} & \sqrt{2}a_{12} & a_{22} & \sqrt{2}a_{13} & \sqrt{2}a_{23} & a_{33} & \cdots & a_{kk} \end{bmatrix}^\top \in \mathbb{R}^{\bar{k}}. \tag{2.1}$$

Denota-se por $smat : \mathbb{R}^{\bar{k}} \to \mathbb{S}^k$ a inversa de svec.

Exemplo 1: Seja A uma matriz simétrica com $k = 3$, $\bar{k} = 6$.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{12} & a_{22} & a_{23} \\ a_{13} & a_{23} & a_{33} \end{bmatrix}.$$

Assim,

$$svec(A) = [a_{11} \quad \sqrt{2}a_{12} \quad \sqrt{2}a_{13} \quad a_{22} \quad \sqrt{2}a_{23} \quad a_{33}].$$

Exemplo 2: Dado o vetor $v = [v_1 \quad v_2 \quad v_3]$.

$$smat(v) = \begin{bmatrix} v_1 & \frac{1}{\sqrt{2}}v_2 \\ \frac{1}{\sqrt{2}}v_2 & v_3 \end{bmatrix}.$$

Definição 3. *Define-se o produto interno entre duas matrizes $A, B \in \mathbb{S}^m$ como:*

$$\langle A, B\rangle = tr\,(AB) = svec(A)^\top svec(B). \tag{2.2}$$

Definição 4. *No conjunto de matrizes simétricas $A \preccurlyeq B$ significa que $A - B$ é semidefinida negativa. Se $A \succcurlyeq B$ significa que $A - B$ é semidefinida positiva. De maneira equivalente, $A \prec B$ significa que $A-B$ é definida negativa e $A \succ B$ significa $A - B$ é definida positiva.*

Definição 5. *Sejam $\zeta : \mathbb{R}^n \to \mathbb{R}^n$, $V \subset \mathbb{R}^n$ e $L \geqslant 0$ uma constante. Dizemos que ζ é Lipschitz em V, com constante L, se*

$$\|\zeta(x) - \zeta(y)\| \leqslant L \|x - y\| \tag{2.3}$$

para $x, y \in V$.

Definição 6. *Sejam as matrizes $A, B \in \mathbb{R}^{n\times n}$ simétricas com $n \in \mathbb{N}$. O produto simétrico de Kronecker entre essas duas matrizes é denotado por $A \circledast B$ e verifica, para qualquer matriz $C \in \mathbb{S}^n$, a seguinte igualdade:*

$$(A \circledast B)\, svec\,(C) = svec\,(sym\,(BCA)) \tag{2.4}$$

Proposição 2.1.1. *Para A, B, C e $D \in \mathbb{S}^m$ e uma matriz não singular P, o produto simétrico de Kromecker e svec satifazem as seguintes propriedades:*

1. $(A \circledast B)svec(C) = (B \circledast C)svec(A)$
2. $A \circledast B = B \circledast A$
3. $(A \circledast B)(C \circledast D) = \frac{1}{2}(AC \circledast BD + AD \circledast BC)$
4. $(P \circledast P)^{-1} = (P)^{-1} \circledast (P)^{-1}$
5. *Se* $A, B \succeq 0$, *então* $(A \circledast B) \succeq 0$
6. *Se* $A \succeq 0$ *e* $B \preceq 0$, *então* $(A \circledast B) \preceq 0$

Definição 7. *Seja $v \in \mathbb{R}^n$. Define-se $diag(v)$ como a matriz diagonal cujos componentes na diagonal são v_i, $i = 1, ..., n$.*

Definição 8. *Seja C um subconjunto de $\mathbb{R}^n$. Dizemos que C é um conjunto convexo se, para qualquer par de pontos $x, y \in C$, e para qualquer $\lambda \in [0, 1]$, o ponto $\lambda x + (1 - \lambda)y$ pertence também a C.*

Definição 9. *Uma função $f : C \to \mathbb{R}$ definida em um conjunto convexo C é convexa se, e somente se, para todo $x, y \in C$ e para $\lambda \in [0, 1]$, se verifica que:*

$$f(\lambda x + (1 - \lambda)y) \leqslant \lambda f(x) + (1 - \lambda)f(y). \tag{2.5}$$

Definição 10. *Uma função f é quasiconvexa se, e somente se, para qualquer x e y, e para qualquer $\lambda \in [0,1]$, a seguinte desigualdade é satisfeita:*

$$f(\lambda x + (1-\lambda)y) \leqslant max\left\{f(x), f(y)\right\}. \tag{2.6}$$

Definição 11. *Uma função $f : X \to \mathbb{R}$ é semi-contínua superiormente no ponto $a \in X$ quando, para cada $c > f(a)$, existe $\delta > 0$ tal que $x \in X, \|x - a\| < \delta$ implica $f(x) < c$.*

2.2 Problema de Programação não linear e semidefinida

Sejam dadas $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^m$ e $G : \mathbb{R}^n \to \mathbb{S}^k$, funções suaves. Usa-se a seguinte notação:

$$g(x) = \begin{bmatrix} g_1(x) \\ \vdots \\ g_m(x) \end{bmatrix}, \quad G(x) = \begin{bmatrix} G_{11}(x) & \cdots & G_{1k}(x) \\ \vdots & \ddots & \vdots \\ G_{1k}(x) & \cdots & G_{kk}(x) \end{bmatrix}. \tag{2.7}$$

O problema de programação não linear (*Nonlinear Programming Problem, NLP*), empregado neste trabalho, se define como:

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} f(x) \\ &sujeito\ a \quad g(x) \leqslant 0 \end{aligned} \tag{2.8}$$

onde f ou g_i são não lineares. A função f se denomina função objetivo e a função g de restrição de desigualdade. A desigualdade $g(x) \leqslant 0$ significa $g_i(x) \leqslant 0$ para $i = 1, \ldots, m$.

O vetor $x = [x_1, ..., x_n]^\top$ denomina-se vetor de variáveis de projeto.

Similarmente, o problema de programação semidefinida (*Semidefinite Programming Problem, SDP*) se define como:

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} f(x) \\ &sujeito\ a \quad G(x) \preccurlyeq 0. \end{aligned} \tag{2.9}$$

A restrição $G(x) \preccurlyeq 0$ denomina-se de restrição semidefinida negativa. A matriz $G(x)$ é simétrica de tamanho $k \times k$.

O problema de programação semidefinida (2.9) se reduz ao problema de programação não linear (2.8) quando $G(x) = diag(g(x))$. O significado de ambos problemas é encontrar $x \in \mathbb{R}^n$ que minimiza $f(x)$ com x pertencendo a uma região do $\mathbb{R}^n$ denominada conjunto de pontos viáveis ou região viável como definida a seguir.

Definição 12. *O conjunto de pontos viáveis define-se por:*

$$\Omega = \{x \in \mathbb{R}^n : G(x) \preccurlyeq 0\}. \tag{2.10}$$

Para o caso do problema (2.8) a região viável se reduz a:

$$\Omega = \{x \in \mathbb{R}^n : g(x) \leqslant 0\}. \tag{2.11}$$

Definição 13. *Um ponto x é dito ponto interior do conjunto $\mathbb{X}$, se existe $\epsilon > 0$ tal que $B(x, \epsilon) \subset \mathbb{X}$, para todo $x \in \mathbb{X}$. $B(x, \epsilon) \subset \mathbb{X}$ denota uma bola aberta de centro em x e raio ϵ.*

O conjunto formado por todos os pontos interiores de $\mathbb{X}$ denominamos interior de $\mathbb{X}$ e se denota por $int(\mathbb{X})$.

Na programação não linear, o interior do conjunto Ω é caracterizado por $g(x) < 0$. Para programação semidefinida, o interior de Ω se caracteriza como sendo $G(x) \prec 0$.

Neste trabalho usa-se a seguinte notação para os gradientes das seguintes funções $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^m$ e $G : \mathbb{R}^n \to \mathbb{S}^k$:

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(x) \\ \vdots \\ \frac{\partial f}{\partial x_i}(x) \\ \vdots \\ \frac{\partial f}{\partial x_n}(x) \end{bmatrix} \in \mathbb{R}^n, \tag{2.12}$$

$$\nabla g(x) = \begin{bmatrix} \nabla g_1(x) & \cdots & \nabla g_i(x) & \cdots & \nabla g_m(x) \end{bmatrix} \in \mathbb{R}^{n \times m}, \ e \tag{2.13}$$

$$\nabla G(x) = \nabla svec\,(G(x)) = \begin{bmatrix} svec\left(\frac{\partial G}{\partial x_1}(x)\right)^\top \\ \vdots \\ svec\left(\frac{\partial G}{\partial x_i}(x)\right)^\top \\ \vdots \\ svec\left(\frac{\partial G}{\partial x_n}(x)\right)^\top \end{bmatrix} \in \mathbb{R}^{n \times \bar{k}} \tag{2.14}$$

onde $\frac{\partial G}{\partial x_i}(x)$ é a derivada parcial de G, em relação a x_i, $i = 1, ..., n$.

A derivada direcional de G na direção d, no ponto x escreve-se:

$$\frac{\partial G}{\partial d}(x) = \sum_{i=1}^{n} d_i \frac{\partial G}{\partial x_i}(x) = smat\left(\nabla G(x)^\top d\right). \tag{2.15}$$

A expansão de Taylor de uma função matricial G pode ser escrita da seguinte

forma:

$$\begin{array}{rcccccl} G(x+d) & = & G(x) & + & \frac{\partial G}{\partial d}(x) & + & R(x,d) \\ & = & G(x) & + & smat\left(\nabla G(x)^{\top} d\right) & + & R(x,d) \end{array} \tag{2.16}$$

onde $R(x,d)$ contém informação das derivadas parciais de segunda ordem de G.

Definição 14. *Um problema de otimização convexo é um problema de otimização onde a função objetivo f é convexa e a região viável Ω é um conjunto convexo.*

2.2.1 Restrições LMI/BMI

O problema (2.9) chama-se problema de programação semidefinida (*Semidefinite Programming Problem, SDP*) quando a função objetivo f é uma função linear e a função G é uma combinação afim de matrizes simétricas. Neste caso, a função matricial G assume a seguinte forma:

$$G(x) = A_0 + \sum_{i=1}^{n} x_i A_i \tag{2.17}$$

onde $A_i \in \mathbb{S}^k$, $i = 0, ..., n$.

A restrição semidefinida

$$A_0 + \sum_{i=1}^{n} x_i A_i \preccurlyeq 0 \tag{2.18}$$

denomina-se desigualdade matricial linear (*Linear Matrix Inequality, LMI*).

Diversos problemas de otimização podem ser reescritos como problemas de programação semidefinida, dentre eles podemos citar os problemas de: programação quadrática, máximo autovalor, minimização da norma matricial, aproximação logarítmica de Chebyshev, otimização de estruturas e minimização do traço fatorial [3].

Neste trabalho se consideram restrições LMI contudo, se dará ênfase em retrições onde a relação entre G e x é bilinear. Neste caso a matriz de restrição $G(x)$ assume o seguinte formato:

$$G(x) = A_0 + \sum_{i=1}^{n} x_i A_i + \sum_{p=1}^{n} \sum_{q=1}^{n} x_p x_q A_{pq} \tag{2.19}$$

sendo a matriz A_{pq} simétrica da mesma dimensão de A_i.

A restrição semidefinida

$$A_0 + \sum_{i=1}^{n} x_i A_i + \sum_{p=1}^{n} \sum_{q=1}^{n} x_p x_q A_{pq} \preccurlyeq 0 \tag{2.20}$$

chama-se desigualdade matricial bilinear (*Bilinear Matrix Inequality-BMI*).

O projeto mecânico de maximização da frequência natural é um exemplo de problema de otimização onde acontece a restrição BMI.

O presente trabalho propõe resolver problemas onde a restrição $G(x) \preccurlyeq 0$ não se limita as formas LMI ou BMI. A única hipótese que G deve verificar é que seja suave.

Tendo em conta (2.16), conclui-se que em problemas com restrições LMI, $R(x,d) = 0$. Contudo, para problemas com restrições BMI, $R(x,d)$ é uma função independente de x (constante em relação a x).

Pode-se provar que um problema de programação semidefinida com restrições LMI é um problema de otimização convexo. Contudo, um problema de otimização com restrições BMI não é convexo [2], [3], [12].

2.3 Condições de otimalidade

Nesta seção se apresentam as condições necessárias de otimalidade de primeira ordem. Estas condições são denominadas condições de *Karush-Kuhn-Tucker, KKT*. A importância destas condições é muito grande pois caracterizam o mínimo relativo do problema de otimização.

As condições de KKT para o problema (2.9) são discutidas por Shapiro [11] e Maurer [13]. As mesmas foram empregadas em [14] e [9] como base para o desenvolvimento de algoritmos gerais que resolvem o problema (2.9).

As condições de KKT para o problema (2.8) são amplamente apresentados na literatura [15], [4] e também constituem a base para o desenvolvimento das técnicas de pontos interiores apresentados neste trabalho.

Definição 15. *O ponto $x^* \in \Omega$ é um mínimo relativo para o problema de otimização (2.9) quando existe um $\varepsilon > 0$ tal que $f(x^*) \leqslant f(x)$ para todo $x \in \Omega \cap B(x^*, \varepsilon)$.*

A seguinte função permite escrever as condições de otimalidade de forma compacta.

Definição 16. *Define-se a função Lagrangeana L para o problema (2.9) como $L : \mathbb{R}^n \times \mathbb{S}^k \to \mathbb{R}$ onde*

$$L(x,\Lambda) = f(x) + \langle G(x), \Lambda \rangle = f(x) + svec(G(x))^\top svec(\Lambda). \tag{2.21}$$

Desta forma, o gradiente da função Lagrangeana (2.23) com respeito a x é:

$$\nabla_x L(x,\Lambda) = \nabla f(x) + \nabla G(x)\lambda. \tag{2.22}$$

onde $\lambda = svec(\Lambda)$ e $\nabla G(x)$ foi definido em (2.14).

Definição 17. *Define-se a função Lagrangeana L para o problema (2.8) como* $L : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ *onde*

$$L(x, \lambda) = f(x) + g(x)^\top \lambda. \tag{2.23}$$

Considerando (2.12) e (2.13), o gradiente da função Lagrangeana (2.23) respeito de x é:

$$\nabla_x L(x, \Lambda) = \nabla f(x) + \nabla g(x)\lambda. \tag{2.24}$$

Definição 18. *Define-se a matriz Hessiana da função Lagrengeana (2.23) como sendo:*

$$H(x, \lambda) = \nabla_{xx} L = \nabla^2 f(x) + \sum_{i=1}^{m} \nabla^2 g_i(x)\lambda_i. \tag{2.25}$$

Para enunciar o teorema das condições de otimalidade do problema (2.9) é necessário definir a condição de Mangasarian-Fromovitz:

Definição 19. *A condição de Mangasarian-Fromovitz se verifica no ponto x se existe d tal que:*

$$G(x) + \frac{\partial G}{\partial d}(x) \prec 0. \tag{2.26}$$

Sob a condição de regularidade apresentada, são válidas as seguintes condições para o problema de programação semidefinida (2.9).

Teorema 2.3.1. *Se f e G são funções da classe C^1, x^* é um mínimo relativo do problema (2.9) e se x^* verifica a condição de Mangasarian-Fromovitz, então existe uma matriz de multiplicadores de Lagrange $\Lambda^* \in \mathbb{S}^k$ tais que:*

$$\begin{aligned} \nabla f(x^*) + \nabla G(x^*)\lambda^* &= 0 \\ \Lambda^* G(x^*) &= 0 \\ G(x^*) &\preccurlyeq 0 \\ \Lambda^* &\succcurlyeq 0. \end{aligned} \tag{2.27}$$

onde $\lambda^* = svec(\Lambda^*)$.

As condições (2.27) são denominadas condições de *Karush-Kuhn-Tucker, KKT*. A igualdade $\Lambda^* G(x^*) = 0$ é chamada de condição de complementaridade. A proposição a seguir mostra outras formas equivalentes de representar a condição de complementaridade.

Proposição 2.3.1. *Se $A \preccurlyeq 0$ e $B \succcurlyeq 0$, então são equivalentes as seguintes igualdades*

$$\begin{aligned} AB &= 0 \\ tr(AB) &= 0 \\ \lambda_i(A)\lambda_i(B) &= 0, i = 1, \cdots, m \end{aligned} \tag{2.28}$$

sendo $\lambda_i(A)$ o i-ésimo autovalor da matriz A com $\lambda_i(A) \leqslant \lambda_{i+1}(A)$.

Como nas condições (2.27) $G(x^*) \preccurlyeq 0$ e $\Lambda^* \succcurlyeq 0$ a equação $\Lambda^* G(x^*) = 0$ equivale a uma complementaridade nos autovalores de Λ^* e $G(x^*)$.

A unicidade da matriz Λ^* em (2.27) é garantida somente sob a hipótese de complementaridade estrita e regularidade. Estes conceitos são definidos a seguir.

Definição 20. *As matrizes Λ^* e $G(x^*)$ verificam a condição de complementaridade estrita quando verificam as condições de KKT (2.27) e*

$$posto(\Lambda^*) + posto(G(x^*)) = k \tag{2.29}$$

onde $posto(A)$ é a dimensão do espaço vetorial gerado pelas colunas de A.

Definição 21. *Seja $p = dim(ker(G(x)))$ a dimensão do núcleo de $G(x)$, e, seja $\{b_1, \ldots, b_p\}$ uma base ortonormal do núcleo de $G(x)$. O ponto x^* é um ponto regular da restrição $G(x) \preccurlyeq 0$ se o conjunto de vetores:*

$$\left\{ \begin{bmatrix} b_i^\top \frac{\partial G}{\partial x_1}(x) b_j \\ \vdots \\ b_i^\top \frac{\partial G}{\partial x_n}(x) b_j \end{bmatrix} : i \leqslant j, \quad i, j = 1, \cdots p \right\} \tag{2.30}$$

é linearmente independente.

Quando as hipótese do teorema (2.3.1), as condições de complementaridade estrita (2.29) e as condições de regularidade (2.30) são verdadeiras, é possível provar que a matriz (multiplicador de Lagrange) Λ^* é única, [11]. Neste trabalho não se fará uso das condições de complementaridade estrita. Contudo, a condição de regularidade (2.30) é uma hipótese empregada na justificação da técnica *Feasible Direction Interior Point Algorithm (FDIPA-SDP)*, que será apresentado no capítulo 4.

A seguir se apresentam definições utilizadas para escrever o teorema das condições de otimalidade (KKT) do problema de programação não linear (2.8).

Definição 22. *O conjunto $I(x) = \{j : g_j(x) = 0,\ j = 1, ..., m\}$ é denominado conjunto de restrições ativas em x.*

Definição 23. *O ponto x^* é um ponto regular da restrição $g(x) \leqslant 0$ se o conjunto de vetores:*

$$\{\nabla g_i(x) : i \in I(x)\} \tag{2.31}$$

é linearmente independente.

O teorema das condições de KKT para o problema (2.8) pode ser formulado da seguinte forma:

Teorema 2.3.2. *Se x^* é um mínimo relativo do problema (2.8) e x^* é um ponto regular da restrição $g(x) \leqslant 0$, então existe $\lambda^* \in \mathbb{R}^m$ tal que:*

$$\begin{aligned} \nabla f(x^*) + \nabla g(x^*)\lambda^* &= 0 \\ G(x^*)\lambda^* &= 0 \\ g(x^*) &\leqslant 0 \\ \lambda^* &\geqslant 0 \end{aligned} \tag{2.32}$$

onde $G(x) = diag(g(x))$.

Definição 24. *Um ponto x^* é um ponto estacionário do problema (2.9) se existe $\Lambda^* \in \mathbb{S}^k$ tal que verifica as três primeiras condições de (2.27), isto é:*

$$\begin{aligned} \nabla L_x(x^*, \Lambda^*) &= 0 \\ \Lambda^* G(x^*) &= 0 \\ G(x^*) &\preccurlyeq 0. \end{aligned} \tag{2.33}$$

Definição 25. *O ponto x^* é um ponto de KKT do problema (2.9) quando x^* é um ponto estacionário associado a matriz de multiplicadores de Lagrange $\Lambda^* \in \mathbb{S}^k$ tal que $\Lambda^* \succcurlyeq 0$.*

Definições análogas são empregadas para o problema (2.8).

Capítulo 3

Problema de otimização estrutural

A engenharia de estruturas, nas últimas décadas, vem levando em consideração vários problemas nos quais são consideradas as vibrações em estruturas quando submetidas a ação de carregamentos dinâmicos. Os impactos sofridos pelas estruturas quando submetidas a ações de cargas que variam ao longo do tempo podem ser determinados através de análise das propriedades que tratam dos modos naturais de vibração, dos fatores de amortecimento e das frequências naturais destas estruturas.

A otimização estrutural tem por objetivo buscar uma configuração para uma estrutura onde se obtenha um desempenho ótimo segundo alguns parâmetros pré-definidos. Ao formular um problema de otimização estrutural é indispensável determinar quais são as: variáveis de projeto, função objetivo e função de restrição.

Concretamente, nesta seção, aborda-se o problema de maximização da menor frequência natural de uma estrutura com restrições de volume e energia de deformação. Em [12] se apresenta resultados teóricos que vinculam o problema em questão com outros problemas.

A formulação clássica do problema de maximização da menor frequência natural possui dificuldades técnicas (como a não diferenciabilidade e a não continuidade). Em [16] é apresentado uma formulação de programação semidefinida que supera todas essas dificuldades, contudo as restrições deste problema de programação semidefinida são bilineares, portanto um problema não convexo. A solução numérica deste problema, em geral, é de difícil obtenção, uma vez que, há poucos algoritmos que resolvem o mesmo. Em [16] é apresentado um algoritmo para obter suas soluções através do código PENBMI.

Será definido na seção (3.2) o problema de otimização estrutural comentado anteriormente.

3.1 Método de elementos finitos

O Método dos Elementos Finitos (MEF) é um procedimento para resolver problemas de valor de contorno (sistema de equações diferenciais em derivadas parciais definido em um domínio e condições de contorno) [17]. No contexto deste trabalho, modela-se o problema de valor de contorno de elastoestática por meio de um sistema de barras interconectadas. O domínio é subdividido em sub-domínios denominados elementos (barras). O domínio do problema original é aproximado (discretizado) por um conjunto finito de sub-domínios (malha de elementos finitos). Para cada elemento é definido um conjunto de funções de interpolação (denominadas funções de forma). Quando se empregam funções definidas como combinação linear das funções de interpolação, o sistema de equações diferenciais original é transformado em um problema numérico (um sistema de equações lineares).

No presente trabalho, o domínio é representado por uma estrutura denomidada **treliça**. Uma treliça é conjunto de barras interconectadas nas extremidades. Cada barra representa um elemento finito. Todo elemento possui dois nós (extremidades). Um exemplo de tal domínio é o apresentado na figura 3.1. Quando algum nó é submetido a uma força, o conjunto de barras é deformado e os nós são deslocados. O sistema de equações diferenciais em derivadas parciais que governa a deformação das barras e portanto, o deslocamento dos nós, é um problema de elastoestática.

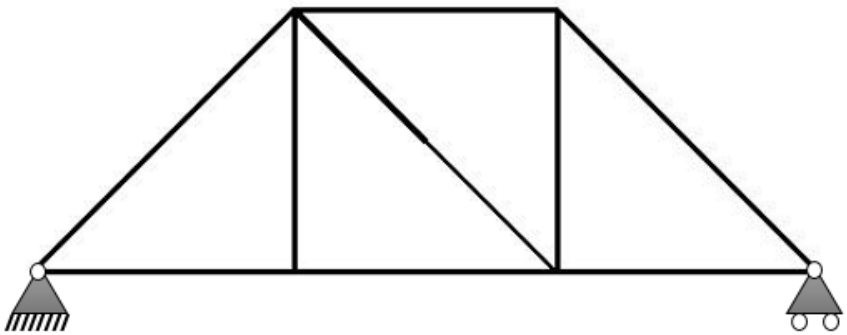

Figura 3.1: Treliça bidimensional.

No modelo de treliças as barras são submetidas a forças de tração ou compressão. Cada uma das barra da estrutura (3.1) será modelada com a seguinte geometria apresentada na figura (3.2), onde A é a área da seção transversal da barra e L o seu comprimento.

Uma barra de comprimento L, quando submetida a uma força axial f produz um deslocamento u relacionado a f definido pela relação constitutiva linear elástica:

$$f = \frac{EA}{L}u \tag{3.1}$$

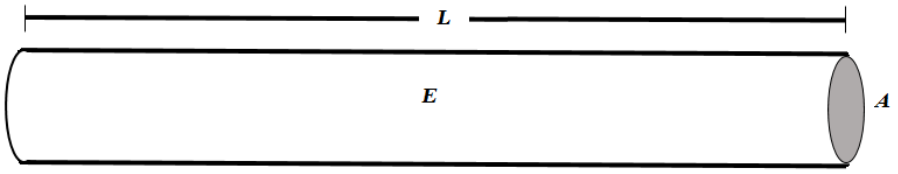

Figura 3.2: Geometria de uma barra.

onde E é o módulo de elasticidade do material da barra (módulo de Young).

A figura (3.3) ilustra o deslocamento u do nó (extremo) direito da barra, quando submetida a uma força axial de tração f.

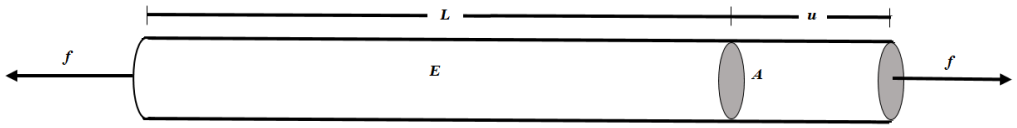

Figura 3.3: Comportamento da barra quando submetida a uma força axial

Em uma treliça, a discretização do domínio é imediata: uma barra representa um elemento, cada elemento pode se conectar a outro elemento na extremidade (no nó). Ao conjunto de elementos e nós, denomina-se de **malha de elementos finitos**, como mostra a figura (3.4).

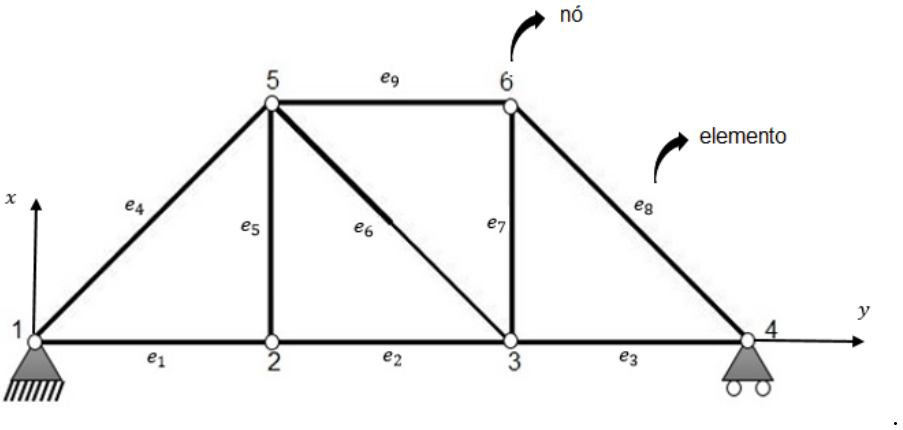

Figura 3.4: Malha de elementos finitos.

Denota-se por n_{elem} como o número de elementos da treliça. No exemplo da figura (3.4), $n_{elem} = 9$.

Seja fixo um sistema de coordenadas cartesiano $Oxyz$, denominado sistema global de coordenadas. Todo nó da malha de elementos finitos possui uma posição, definida por três números reais que representam as coordenadas do nó. Dependendo do modelo de treliças, se for $2D$ ou $3D$, cada nó poderá ter, no máximo, 2 ou 3 graus de liberdade por nó. Um grau de liberdade representa um possível movimento de translação. Um nó poderá se deslocar na direção x, na direção y e até na direção z.

Na figura (3.4), ilustra-se o caso de uma treliça $2D$, portanto, todo nó poderá, no máximo ter dois graus de liberdade (não haverá deslocamento na direção z). No nó 1 define-se uma condição de contorno: o nó 1 não tem nenhum grau de liberdade (o nó 1 está fixo). No caso do nó 4, define-se outra condição de contorno: o nó 4 tem apenas um grau de liberdade (o nó 4 pode se mover na direção x mas não na direção y).

Define-se **eixo** de uma barra à reta que passa por seus nós. Toda barra possui um sistema de coordenadas local (Os) onde a coordenada local s é paralela ao eixo da barra.

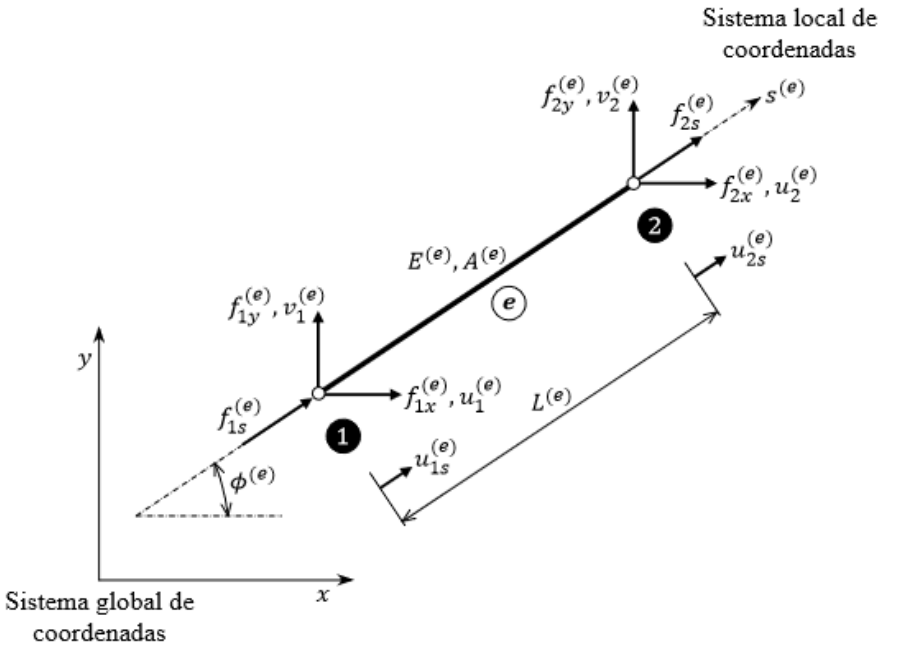

Figura 3.5: Elemento (barra) bidimensional.

A figura (3.5) mostra um elemento (barra) bidimensional de uma estrutura de treliças. Os nós estão representados por círculos pretos e númerados com 1 ou 2. O elemento é representado pela letra e e possui dois nós: o nó inicial 1 (a origem no sistema local de coordenadas) e o nó final 2. Cada nó apresenta dois graus de liberdade no sistema global de coordenadas. O ângulo ϕ é o ângulo que forma o eixo do elemento com a direção x do sistema de coordenadas globais.

No sistema local de coordenadas são definidos o vetor deslocamento e o vetor de força, respectivamente por:

$$\begin{aligned} \bar{u}^{(e)} &= \begin{bmatrix} u_{1s}^{(e)} & u_{2s}^{(e)} \end{bmatrix}^\top \\ \bar{f}^{(e)} &= \begin{bmatrix} f_{1s}^{(e)} & f_{2s}^{(e)} \end{bmatrix}^\top . \end{aligned} \tag{3.2}$$

No sistema global de coordenadas são definidos o vetor deslocamento e o vetor

de força, respectivamente por:

$$\begin{aligned} u^{(e)} &= \left[u_1^{(e)}\ v_1^{(e)}\ u_2^{(e)}\ v_2^{(e)}\right]^\top \\ f^{(e)} &= \left[f_{1x}^{(e)}\ f_{1y}^{(e)}\ f_{2x}^{(e)}\ f_{2y}^{(e)}\right]^\top. \end{aligned} \tag{3.3}$$

A partir dos nós do elemento, é possível determinar o comprimento da barra da segunte maneira:

$$L^{(e)} = \sqrt{\left(x_2^{(e)} - x_1^{(e)}\right)^2 + \left(y_2^{(e)} - y_1^{(e)}\right)^2} \tag{3.4}$$

onde $\left(x_1^{(e)}, y_1^{(e)}\right)$ e $\left(x_2^{(e)}, y_2^{(e)}\right)$ são as coordenadas dos nós 1 e 2 do elemento e.

Define-se $l^{(e)}$ e $m^{(e)}$ como sendo os co-senos diretores do ângulo $\phi^{(e)}$:

$$\begin{aligned} l^{(e)} &= \cos\phi^{(e)} = \tfrac{x_2^{(e)} - x_1^{(e)}}{L^{(e)}} \\ m^{(e)} &= \sin\phi^{(e)} = \tfrac{y_2^{(e)} - y_1^{(e)}}{L^{(e)}}. \end{aligned} \tag{3.5}$$

A partir dos valores encontrados para $l^{(e)}$ e $m^{(e)}$, pode-se definir a matriz de transformação T como sendo:

$$T^{(e)} = \begin{bmatrix} l^{(e)} & m^{(e)} & 0 & 0 \\ 0 & 0 & l^{(e)} & m^{(e)} \end{bmatrix}. \tag{3.6}$$

Os componentes globais definidos em (3.3) e locais definidos em (3.2) podem ser vinculados do seguinte modo:

$$\begin{aligned} \bar{u}^{(e)} &= T^{(e)} u^{(e)} \\ \left(T^{(e)}\right)^\top \bar{f}^{(e)} &= f^{(e)}. \end{aligned} \tag{3.7}$$

No sistema local de coordenadas o equilíbrio elastoestático (para o elemento) pode escrever-se:

$$\bar{K}^{(e)} \bar{u}^{(e)} = \bar{f}^{(e)}$$

onde $\bar{K}^{(e)}$ é a matriz de rigidez do elemento de barra no sistema local de coordenadas definida por:

$$\bar{K}^{(e)} = \tfrac{A^{(e)} E^{(e)}}{L^{(e)}} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}. \tag{3.8}$$

Considerando (3.7) e (3.8), o equilibrio elastoestático (para o elemento), no sistema global de coordenadas pode escrever-se como:

$$K^{(e)} u^{(e)} = f^{(e)} \tag{3.9}$$

onde $K^{(e)}$ é a matriz de rigidez do elemento de barra no sistema global de coor-

denadas definida por:

$$K^{(e)} = T^{(e)} \bar{K}^{(e)} T^{(e)}. \tag{3.10}$$

Substituindo $T^{(e)}$ definida em (3.6) e $\bar{K}^{(e)}$ definida em (3.8) tem-se:

$$K^{(e)} = \frac{A^{(e)} E^{(e)}}{L^{(e)}} \begin{bmatrix} l^{2(e)} & l^{(e)} m^{(e)} & -l^{2(e)} & -l^{(e)} m^{(e)} \\ l^{(e)} m^{(e)} & m^{2(e)} & -l^{(e)} m^{(e)} & -m^{2(e)} \\ -l^{2(e)} & -l^{(e)} m^{(e)} & l^{2(e)} & l^{(e)} m^{(e)} \\ -l^{(e)} m^{(e)} & -m^{2(e)} & l^{(e)} m^{(e)} & m^{2(e)} \end{bmatrix}. \tag{3.11}$$

A matriz de rigidez $K^{(e)}$ se caracteriza por ser simétrica, semidefinida positiva e dependente dos parâmetros L, E, A e ϕ.

3.1.1 Matriz de rigidez

Como (3.9) é verificado para todo elemento da treliça, é possível, por meio de um processo de montagem (*assembly*), construir a equação de equilíbrio global da estrutura:

$$Ku = f \tag{3.12}$$

onde $K \in \mathbb{S}^{n_{dof}}$ é denominada matriz de rigidez da estrutura, $u \in \mathbb{R}^{n_{dof}}$ é chamado de vetor de deslocamentos nodais e $f \in \mathbb{R}^{n_{dof}}$ de vetor de forças nodais. O número n_{dof} é chamado de número de graus de liberdade da estrutura (*number of degree of freedom, ndof*) e corresponde a soma dos graus de liberdade de cada nó da estrutura.

No exemplo da figura (3.6), $n_{dof} = 9 = 2$(nó 2)+2(nó 3)+1(nó 4)+2(nó 5)+2(nó 6). A ordem de como os graus de liberdade são organizados é arbitrária. O vetor de deslocamentos u e o vetor de forças nodais f pode ter o seguinte aspecto:

$$\begin{aligned} u &= [u_{12} \quad u_{22} \quad u_{13} \quad u_{23} \quad u_{14} \quad u_{15} \quad u_{25} \quad u_{16} \quad u_{26}]^\top \\ f &= [0 \quad f_{22} \quad 0 \quad f_{23} \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^\top \end{aligned}$$

onde u_{ij} e f_{ij} representa o componente do deslocamento e o componente da força aplicada no nó j na direção i ($i = 1$ significa direção x e $i = 2$ significa direção y), respectivamente. Os componentes f_{22} e f_{23} são neste caso números negativos.

Sempre que as condições de contorno impossibilitem que a estrutura tenha um movimento de corpo rígido, a matriz de rigidez da estrutura sempre resultará em uma matriz definida positiva [17].

O vetor de forças nodais f corresponde a forças aplicadas em cada nó. Por estar do lado direito da equação (3.12), f é denominado de lado direito (*Right Hand Side, RHS*). Para cada escolha do vetor de forças, tem-se um deslocamento u. O deslocamento u é a solução única do sistema de equações (3.12).

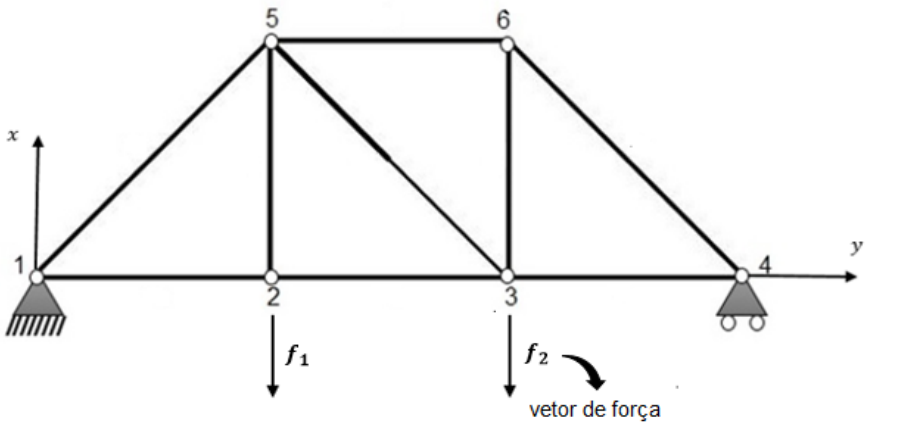

Figura 3.6: Vetor de força nodais.

A matriz de rigidez K é uma função matricial que depende dos seguinte parâmetros do elemento (e):

- área: $A^{(e)}$
- comprimento: $L^{(e)}$
- módulo de elasticidade: $E^{(e)}$

Neste trabalho, considera-se que todas as barras possuem o mesmo módulo de elasticidade, isto é, todas as barras estão constituidas do mesmo material elástico (aço, por exemplo). O comprimento de cada barra é definido por meio das coordenadas dos nós, e também se considerará constante.

O que se considerará variável neste trabalho é a área da seção transversal de cada elemento, $A^{(e)}$.

Desse modo, o **volume do elemento,** x_e define-se por:

$$x_e = A^{(e)} L^{(e)}. \tag{3.13}$$

Desta forma é possível definir o vetor de volumes da treliça $x \in \mathbb{R}^{n_{elem}}$como sendo:

$$x = [x_1 \quad x_2 \quad \dots \quad x_e \quad \dots x_{n_{elem}}]^\top. \tag{3.14}$$

Assim, a matriz de rigidez K será considerada uma função do vetor x. A motagem (*assembly*) de K pode escrever-se como a soma de matrizes de rigidez elementares K_e:

$$K(x) = \sum_{e=1}^{n_{elem}} x_e K_e. \tag{3.15}$$

Com esta notação:

- $x_e = 0$ significa que o elemento (e) tem volume nulo (tem o mesmo efeito que retirar a barra (e) da estrutura).
- $x \geqslant 0$ significa que $x_e \geqslant 0$ qualquer que seja o elemento (e).

3.1.2 Matriz de massa

A equação de movimento dos nós da estrutura é governada pela formulação do problema de elastodinâmica [17]. Este modelo é resultado da formulação das leis físicas da dinâmica (leis de Newton) sobre um corpo deformável (desconsiderando forças de atrito). O Método dos Elementos Finitos também pode ser aplicado a este modelo. Neste caso, o problema de elastodinâmica resulta no seguinte modelo numérico de elementos finitos:

$$M(x)\ddot{u} + K(x)u = f \tag{3.16}$$

onde $M(x) \in \mathbb{S}^{n_{dof}}$ é denominada matriz de massa da estrutura e $\ddot{u} \in \mathbb{R}^{n_{dof}}$ é chamado de vetor de acelerações nodais. A matriz $K(x)$ e os vetores $u \in \mathbb{R}^{n_{dof}}$ e $f \in \mathbb{R}^{n_{dof}}$ tem os mesmos significados que na equação (3.12).

De forma similar que para a matriz de rigidez, M é resultado da soma de matrizes de massa elementares M_e:

$$M(x) = \sum_{e=1}^{n_{elem}} x_e M_e \tag{3.17}$$

onde x_e é o volume do elemento (e) definido em (3.13).

A matriz de massa do elemento M_e depende, adicionalmente, da **densidade do material do elemento,** ρ_e^{mat}. O produto $\rho_e^{mat} x_e$ representa a **massa do elemento,** m_e:

$$m_e = \rho_e^{mat} x_e. \tag{3.18}$$

3.1.3 Complacência

A complacência (*compliance*) de uma estrutura se define como uma medida da energia de deformação. Esta medida é relativa a força f aplicada na estrutura.

Seja dada uma treliça com determinada distribuição de volumes $x \geqslant 0$. Seja f um vetor de foças aplicadas na treliça. O deslocamento u do equilíbrio é a solução do sistema linear:

$$K(x)u = f. \tag{3.19}$$

A complacência da estrutura, relativamente a f e x, se define como:

$$f^\top u. \tag{3.20}$$

A complacência é um número não negativo pois, $f^\top u = u^\top K(x)u$ e $K(x) \succcurlyeq 0$, qualquer que seja $x \geqslant 0$.

Fixado f e $\bar{\gamma} > 0$, o conjunto de restrições:

$$\begin{aligned} f^\top u &\geqslant \bar{\gamma} \\ K(x)u &= f \\ x &\geqslant 0 \end{aligned} \tag{3.21}$$

representa um subconjunto de treliças com distribuição de volume x no qual a estrutura, no equilibrio, não pode ter uma complacência maior que $\bar{\gamma}$. De outra forma, o conjunto de restrições (3.21) é o conjunto de treliças com distribuição de volume x que podem se deformar até o nível $\bar{\gamma}$, devido a força f. Quanto maior é $\bar{\gamma}$, mais "complacente"a estrutura pode ser (poderá se deformar mais).

3.1.4 Frequência natural e fundamental de uma estrutura

Para entender as vibrações de uma treliça considera-se a equação do movimento (3.16) com $f = 0$:

$$M(x)\ddot{u} + K(x)u = 0. \tag{3.22}$$

A equação (3.22) pode significar, por exemplo, o movimento dos nós da treliça, após uma "pancada", ou seja uma força f que se aplica por um instante de tempo e cessa imediatamente. Neste movimento não se consideram forças de atrito.

Seja $j = \sqrt{-1}$. Assumindo a solução da forma:

$$u = we^{j\omega t} \tag{3.23}$$

e substituindo-a em (3.22) tem-se:

$$-\omega^2 M(x)we^{j\omega t} + K(x)we^{j\omega t} = \left(-\omega^2 M(x)w + K(x)w\right)e^{j\omega t} = 0$$

portanto,

$$-\omega^2 M(x)w + K(x)w = 0$$

ou também:

$$K(x)w = \lambda M(x)w \tag{3.24}$$

onde $\lambda = \omega^2$.

A equação (3.24) define o problema de autovalor-autovetor generalizado [18] em que: dado x, pretende-se encontrar w e λ de modo que (3.24) seja verificada. O par (λ, w) denomina-se autovalor e autovetor do par de matrizes $(K(x), M(x))$ (*matrix pencil*). Nesta seção mostra-se que a relação funcional entre x e o par $(\lambda(x), w(x))$

satisfazendo (3.24) pode ser não suave e até discontínua [12].

Os autovalores λ e autovetores w são chamados de quadrado das **frequências naturais** e de **modos (ou formas) de vibração** da estrutura, repectivamente. Em [12] mostra-se que se $x > 0$, então $K(x)$ e $M(x)$ são definidas positivas (matrizes invertíveis). Contudo, se $x \geqslant 0$, as matrizes $K(x)$ e $M(x)$ podem ser singulares, podendo-se afirmar, apenas que são semidefinidas positivas. Quando $K(x) \succcurlyeq 0$ e $M(x) \succ 0$, o problema de autovalor-autovetor generalizado (3.24) admite n_{dof} autovalores reais:

$$\lambda_{min}(x) = \lambda_1 \leqslant \ldots \leqslant \lambda_{n_{dof}}$$

associados a n_{dof} autovetores (com componentes reais):

$$w_1, \ldots, w_{n_{ndof}}$$

de modo que $w_i^\top M(x) w_j = 0$ se $i \neq j$. A raíz quadrada do menor autovalor λ_{min} se denomina **frequência fundamental da estrutura**.

Quando uma estrutura é submetida a forças dinâmicas (por exemplo: vento, vibração de um motor) é desejável que tal estrutura seja projetada de modo que sua frequência fundamental esteja acima da frequência apresentada pelas tais forças dinâmicas.

Por exemplo, se o motor de um carro vibra a uma frequência λ_f e se quer projetar uma treliça (com distribuição de volume x) que sustente o motor, a seguinte restrição é muito desejável:

$$\lambda_{min}(x) \geqslant \bar{\lambda}$$

onde a constante $\bar{\lambda}$ é escolhida de modo que $\bar{\lambda} > \lambda_f$. Este tipo de restrições previne que a estrutura falhe pois, se a frequência fundamental da estrutura for próxima a λ_f, a estrutura pode entrar em ressonância: os deslocamentos podem aumentar até que a estrutura quebre.

3.1.5 Definição e propriedades da função $\lambda_{min}(x)$

Seja $X := \{x \in \mathbb{R}^n : x \geqslant 0, x \neq 0\}$. O artigo [12] realiza um estudo matemático da seguinte função $\lambda_{min}(x)$ com $x \in X$.

$$\lambda_{min}(x) = min\left\{\lambda : \exists w \in \mathbb{R}^{n_{dof}} : K(x)w = \lambda M(x)w, w \notin Ker(M(x))\right\}. \quad (3.25)$$

Mostra-se que $\lambda_{min} : X \to \mathbb{R} \cup \{+\infty\}$ tem as seguintes propriedades:

Proposição 3.1.1. *1. λ_{min} é finito e não negativo em X;*

2. Para todo $x \in X$, $\lambda_{min}(x) = sup\{\lambda : K(x) - \lambda M(x) \succcurlyeq 0\}$;

3. λ_{min} *é semi-contínua superiormente em* X;

4. λ_{min} *é contínua em* $\{x \in \mathbb{R}^n : x \geqslant \varepsilon > 0\}$;

5. $-\lambda_{min}$ *é quasiconvexa em* X.

3.2 Formulação clássica dos problemas de mínimo volume e de máxima frequência

Nesta seção se apresentam dois problemas de otimização de estruturas:

1. Problema da minimização do volume com restrições de complacência e de frequência natural.

2. Problema de máximização da frequência fundamental com restrições de complacência e volume.

Os problemas 1 e 2 são denominados de problemas clássicos pois possuem restrições inicialmente abordadas na literatura da programação não linear (restrições de igualdade e de desigualdade). Estes problemas apresentam grande destaque na engenharia mecânica uma vez que possuem importantes aplicações práticas.

Seja dada uma treliça com n_{elem} barras, com um total de $p = n_{dof}$ graus de liberdade. Considera-se fixo, para todo elemento (e), o módulo de elasticidade $E^{(e)}$ e a densidade ρ_e^{mat}. Para cada elemento (e) da treliça, define-se uma variável de projeto x_e que significa o volume do elemento (3.13). O vetor $x \in \mathbb{R}^{n_{elem}}$ corresponde a uma distribuição de volumes para todas as barras da treliça, como estabelecido em (3.14). Seja $f \in \mathbb{R}^p$, com $f \neq 0$, um vetor de carga nodais.

Pode-se definir o problema da minimização do volume com restrições de complacência e de frequência natural como sendo:

$$\begin{aligned} \max_{x \in \mathbb{R}^{n_{elem}}, u \in \mathbb{R}^{n_{dof}}} & \sum_{i=1}^{n_{elem}} x_i \\ sujeito\ a \quad K(x)u &= f \\ f^\top u &\leqslant \bar{\gamma} \\ \lambda_{min}(x) &\geqslant \bar{\lambda} \\ x &\geqslant 0 \end{aligned} \tag{3.26}$$

onde $\bar{\gamma}$ e $\bar{\lambda}$ são constantes positivas.

O problema (3.26) significa encontrar uma treliça com distribuição de volumes x, com o menor volume possível, dentre todas as treliças que verificam determinado nível de deformação $(\bar{\gamma})$ e tenham sua frequência fundamental maior ou igual a determinada frequência mínima admissível $(\bar{\lambda})$. Quando se minimiza o volume de

uma treliça, se minimiza a massa empregada em sua construção. Isto tem significado econômico imediato pois o custo da estrutura esta diretamente relacionado com a quantidade de material que se empregará. Assim, a solução do problema (3.26) é também a estrutura mais econômica, dentre todas as estruturas que verificam as restrições de complacência e frequência.

O problema da máximização da frequência fundamental com restrições de complacência e volume se define da seguinte forma:

$$\begin{array}{rrcl} \max\limits_{x\in\mathbb{R}^{n_{elem}},u\in\mathbb{R}^{n_{dof}}} & \lambda_{min}(x) & & \\ sujeito\ a & K(x)u & = & f \\ & f^\top u & \leqslant & \bar{\gamma} \\ & \sum_{i=1}^{n_{elem}} x_i & \leqslant & \bar{V} \\ & x & \geqslant & 0 \end{array} \tag{3.27}$$

onde $\bar{V}$ é uma constante positiva.

O problema (3.26) significa encontrar uma treliça com distribuição de volumes x, dentre todas as treliças que verificam determinado nível de deformação ($\bar{\gamma}$) e tenham seu volume limitado a uma quantidade máxima ($\bar{V}$). Este problema é um dos desafios mais importantes da engenharia mecânica: encontrar a estrutura x com a maior frequência possível limitando-se apenas a quantidade de material ($\bar{V}$) e deformação ($\bar{\gamma}$).

Tanto o problema (3.26) como o problema (3.27) possuem, como variáveis de projeto, o deslocamento $u \in \mathbb{R}^{n_{dof}}$ e os volumes das barras $x \in \mathbb{R}^{n_{elem}}$. Isto representa um total de $n_{dof} + n_{elem}$ número de variáveis.

Os problemas de otimização apresentados são exemplos de **problemas de topologia**. A restrição $x \geqslant 0$ significa que alguma barra com volume x_i pode se anular. Isto significa que a barra i pode desaparecer durante o processo de otimização. Na seção seguinte se indicam algumas dificuldades teóricas destes problemas.

3.2.1 Dificuldades da formulação clássica

Os problemas (3.26) e (3.27) possuem sérias dificuldades técnicas.

Para ambos problemas o número de variáveis de projeto pode ser muito grande, pois depende, não somente do número de barras da treliça (n_{elem}), como também do número de graus de liberdade (n_{dof}).

A função λ_{min} só será uma função diferenciável quando a multiplicidade do menor autovalor de (3.24) for um.

Para tratar a não diferenciabilidade de λ_{min} é preciso utilizar métodos de otimização não suaveis, que utilizam informações do gradiente generalizado, os quais levam em consideração a não suavidade. Para problemas numericamente tratáveis

existe o cálculo de Clarke [19], que trabalha com funções que são localmente Lipschitz contínuo. Para esse tipo de cálculo já existem alguns algoritmos e códigos que calcula um ótimo local de uma função localmente Lipschitz contínuo. Dessa forma, se λ_{min} for uma função localmente Lipschitz contínuo, pode-se usar o cálculo de Clarke e procedimentos numéricos conhecidos para resolver o problema (3.27).

Outro fato que mostra a dificuldade de se trabalhar com a função λ_{min} é a possibilidade de que seja descontinua em alguns pontos na fornteira de X.

A computação de $\lambda_{min}(x)$ apresenta-se como um desafio, pois, apesar da existência de procedimentos numéricos bem estabelecidos, ainda requer um tempo computacional razoável para que o resultado tenha uma precisão apropriada.

Para superar essas dificuldades na seção seguinte apresenta-se uma forma de resolver os problemas em questão através de modelos de programação semidefinida equivalentes.

3.3 Formulação SDP

A variável de projeto $u \in \mathbb{R}^{n_{dof}}$ pode ser eliminada de ambos problemas (3.26) e (3.27), tendo em vista a seguinte proposição (extraída de [12]):

Proposição 3.3.1. *Seja $A \succcurlyeq 0$ e $c \in \mathbb{R}$. Portanto,*

$$\begin{array}{rcl} Au & = & f \\ f^\top u & \leqslant & c \end{array} \iff \begin{pmatrix} c & -f^\top \\ -f & A \end{pmatrix} \succcurlyeq 0. \tag{3.28}$$

Substituindo A por $K(x)$ e c por $\bar{\gamma}$ em (3.28), tem-se:

$$\begin{array}{rcl} K(x)u & = & f \\ f^\top u & \leqslant & \bar{\gamma} \end{array} \iff \begin{pmatrix} \bar{\gamma} & -f^\top \\ -f & K(x) \end{pmatrix} \succcurlyeq 0. \tag{3.29}$$

Desta forma a restrição de igualdade $K(x)u = f$ e a de desigualdade $f^\top u \leqslant \bar{\gamma}$ são substituíveis por uma restrição semidefinida onde não aparece a variável de projeto $u \in \mathbb{R}^{n_{dof}}$.

Para eliminar o uso da função $\lambda_{min}(x)$, emprega-se a proposição (3.1.1). Em particular, para todo $x \in X$:

$$\lambda_{min}(x) = sup\left\{\lambda : K(x) - \lambda M(x) \succcurlyeq 0\right\}.$$

A restrição $\lambda_{min}(x) \geqslant \bar{\lambda}$ em (3.26) equivale a restrição semidefinida:

$$K(x) - \bar{\lambda} M(x) \succcurlyeq 0.$$

Portanto, (3.26) é equivalente a:

$$\begin{array}{rcl} \max\limits_{x\in\mathbb{R}^{n_{elem}}} \sum_{i=1}^{n_{elem}} x_i & & \\ sujeito\ a \begin{pmatrix} \bar{\gamma} & -f^\top \\ -f & K(x) \end{pmatrix} & \succcurlyeq & 0 \\ K(x) - \bar{\lambda}M(x) & \succcurlyeq & 0 \\ x & \geqslant & 0. \end{array} \tag{3.30}$$

O problema (3.30) é um problema de programação semidefinida onde possui apenas restrições LMI. Este problema é convexo. Portanto, grande número de algoritmos podem resolver o mesmo eficientemente [20], [21].

Por outro lado, para eliminar λ_{min} da função objetivo de (3.27) basta seguir o seguinte raciocínio: maximizar $\lambda_{min}(x)$ equivale a maximizar λ onde $\lambda \leqslant \lambda_{min}(x)$. Isto equivale a minimizar $-\lambda$ onde $\lambda \leqslant \lambda_{min}(x)$. Portanto, $K(x) - \lambda M(x) \succcurlyeq 0$.

Desta forma, o problema (3.27) equivale a:

$$\begin{array}{rcl} \min\limits_{x\in\mathbb{R}^{n_{elem}},\lambda\in\mathbb{R}} -\lambda & & \\ sujeito\ a \begin{pmatrix} \bar{\gamma} & -f^\top \\ -f & K(x) \end{pmatrix} & \succcurlyeq & 0 \\ \sum_{i=1}^{n} x_i & \leq & \bar{V} \\ K(x) - \lambda M(x) & \succcurlyeq & 0 \\ x & \geqslant & 0 \end{array} \tag{3.31}$$

onde agora as variáveis de projeto são x e λ.

O problema (3.31) é um problema de programação semidefinida com uma restrição BMI:

$$K(x) - \lambda M(x) \succcurlyeq 0.$$

Portanto, este problema é não convexo, e não pode ser resolvido com técnicas numéricas para problemas convexos. Este problema será resolvido com a técnica proposta nesta dissertação (FAIPA-SDP).

3.4 Relação entre os modelos SDP

O motivo de apresentar os problemas (3.26) e (3.27) é por que estão vinculados entre si como indica a seguinte proposição [12]:

Teorema 3.4.1. *Considere o problema (3.30) com as constantes $\bar{\gamma}$ e $\bar{\lambda}$. Seja x^* a solução ótima do problema (3.30) e $V^* = \sum_{i=1}^{n_{elem}} x_i^*$ (volume ótimo da treliça). Considere o problema (3.31) com as constantes $\bar{\gamma}$ e $\bar{V} = V^*$. Então o problema (3.31) terá solução ótima (x^*, λ^*) onde $\lambda^* = \bar{\lambda}$.*

O teorema (3.4.1) é empregado para a construção dos testes numéricos apresentados neste trabalho.

Capítulo 4

Algoritmo de pontos interiores por arcos viáveis para programação semidefinida

Este capítulo é dedicado a apresentação de um conjunto de algoritmos que engloba duas classes de problemas da programação matemática: o problema não linear com restrições de igualdade e desigualdade vetorial *(non-linear programming problem)* e o problema não linear com restrições de desigualdade matriciais *(semidefinite programming problem)*.

Os algoritmos que iremos apresentar tem como foco um algoritmo proposto por Herskovits [6], denominado FAIPA *(Feasible Arc Interior Point Algorithm)*, que é um algoritmo que utiliza o método de pontos interiores e resolve problemas de programação não linear com restrições de igualdade e desigualdade vetorial.

O FAIPA realiza iterações de pontos fixos para resolver as condições de otimalidade de primeira ordem de *Karush-Kuhn-Tucker (KKT)*, onde a cada iteração se obtem uma direção de descida e viável que será definida através da resolução dois sistemas lineares. Um terceiro sistema linear deve ser resolvido para a construção de um arco viável. Posteriormente é feita uma busca linear ao longo do arco para obter o novo ponto. A sequência gerada pelo algoritmo converge a pontos de KKT do problema (2.8).

O FAIPA apresenta importantes propriedades teóricas, e vem sendo aprimorado desde a década de 80, onde este algoritmo tem sido empregado a uma ampla gama de aplicações de engenharia.

Quando no FAIPA são resolvidos apenas os dois primeiros sistemas lineares, e a busca linear é realizada em uma direção, o algoritmo em questão passa a ser chamado de FDIPA.

A partir do FDIPA, foi desenvolvido em [9] o algoritmo de pontos interiores

por direções viáveis para programação semidefinida (FDIPA-SDP). Este algoritmo é uma generalização do FDIPA para resolver problemas (convexos ou não convexo) de programação semidefinida. Foi provado em [9] a convergência global do FDIPA-SDP para pontos estacionários do problema (convexo ou não convexo) de programação semidefinida. O FDIPA-SDP vem sendo utilizado em algumas aplicações de otimização estrutural [9], [7], [22].

Neste capítulo se apresenta um novo procedimento para resolver problemas (não necessáriamente convexos) de programação semidefinida. Este procedimento se denomina algoritmo de pontos interiores por arcos viáveis para programação semidefinida (*Feasible Arc Interior Point Algorithm for Semidefinite Programming, FAIPA-SDP*). Este algoritmo pretende extender as ideias do FAIPA para programação semidefinida.

O capítulo está organizado da seguinte forma. Se apresentam as notações e definições necessárias para a descrição do FAIPA e do FAIPA-SDP. Posteriormente, descreve-se o algoritmo FAIPA e depois se apresenta a nova técnica FAIPA-SDP.

4.1 Notações e definições preliminares

Definição 26. *Seja $f : \mathbb{R}^n \to \mathbb{R}$. Um vetor $d \in \mathbb{R}^n$ é uma direção de descida para a função f no ponto $x \in \mathbb{R}^n$ se existir $\delta > 0$ tal que $f(x + td) < f(x)$ para todo $t \in (0, \delta)$.*

Definição 27. *Um vetor $d \in \mathbb{R}^n$ é uma direção viável para o conjunto $\Omega \subset \mathbb{R}^n$, no ponto $x \in \Omega$, se existir um $\theta > 0$ tal que $x + td \in \Omega$ para todo $t \in [0, \theta]$.*

4.2 FAIPA

A presente seção será dedicada ao estudo do algoritmo de pontos interiores por arcos viáveis (FAIPA) proposto por Herskovits [8]. Este algoritmo resolve problemas de otimização com restrições de igualdade e desigualdade vetoriais. Para fins do presente estudo, somente se consideram restrições de desigualdade vetoriais $g(x) \leqslant 0$, $g(x) \in \mathbb{R}^m$, como definido no problema (2.8).

4.2.1 Descrição do algoritmo

O FAIPA resolve problemas gerais de programação não linear com restrições (2.8), sendo esses problemas possivelmente não convexos. Dado um ponto inicial x^0 dentro da região viável Ω do problema de otimização, o FAIPA gera uma sequência x^k de pontos interiores a esse conjunto viável. Para isto, o FAIPA realiza cálculos para obter uma direção d e um arco $A(t)$, ambos de descida para a função objetivo f e

viável para Ω. Quando no FAIPA é calculado apenas a direção viável d, este algoritmo passa a se denominar FDIPA *(Feasible Direction Interior Point Algorithm)*. Depois do cálculo da direção viável (e arco viável) é feita uma busca linear (ao longo de $x^k + td$ ou ao longo do arco $A(t)$) para calcular um passo $t > 0$ de modo que o novo ponto, x^{k+1}, seja interior a região viável. Se o problema de otimização apresentar apenas restrições de desigualdade a função objetivo se reduz a cada iteração do algoritmo, isto é $f(x^{k+1}) < f(x^k)$.

Foi provado em [23] que o FDIPA possui convergência global a pontos de KKT. Isto significa que a sequência gerada pelo FDIPA converge a um ponto de KKT, seja qual for o ponto inicial x^0 dentro da região viável do problema de otimização. Já no FAIPA, foi provado em [24], que comprimento do passo na busca linear é unitário em iterações próximas da solução. Desta forma, sob algumas hipóteses o FAIPA verifica a convergência superlinear em dois passos, isto é:

$$\lim_{k\to\infty} \frac{\left\|x^{k+2} - x^*\right\|}{\|x^k - x^*\|} = 0 \tag{4.1}$$

onde x^* é um ponto de KKT.

O FAIPA é um método iterativo. Ele realiza iterações nas variáveis primais x (variáveis de projeto) e duais λ (multiplicadores de Lagrange) com o objetivo de resolver as condições de otimalidade de KKT. Partindo de uma estimativa inicial $x_0 \in int(\Omega)$, o FAIPA gera uma sequência de pontos $(x^k, \lambda^k)_{k\in\mathbb{N}}$, $g(x^k) < 0$ tal que $f(x^{k+1}) < f(x^k)$ de modo que x^k seja convergente a um ponto de KKT. A partir de x^k se calcula uma direção d^k, viável para o problema (2.8) e de descida para f no ponto x^k.

4.2.2 Cálculo da direção de descida e viável

Define-se a função $\Phi : \mathbb{R}^n \times \mathbb{R}^k \to \mathbb{R}^n \times \mathbb{R}^k$ composta pelas igualdades de (2.27) da seguinte forma:

$$\Phi(x, \lambda) = \begin{bmatrix} \nabla L(x, \lambda) \\ G(x)\lambda \end{bmatrix} \tag{4.2}$$

onde $G(x) = diag(g(x))$.

A solução do sistema não-linear $\Phi(x, \lambda) = 0$, é obtida através de uma iteração de Newton como indicado a seguir:

$$\begin{bmatrix} B^k & \nabla g(x^k) \\ \Lambda^k \nabla g^\top(x^k) & G(x^k) \end{bmatrix} \begin{bmatrix} x_0^{k+1} - x^k \\ \lambda_0^{k+1} - \lambda^k \end{bmatrix} = - \begin{bmatrix} \nabla f(x^k) + \nabla g(x^k)\lambda^k \\ G(x^k)\lambda^k \end{bmatrix} \tag{4.3}$$

sendo (x^k, λ^k) a estimativa inicial, $(x_0^{k+1}, \lambda_0^{k+1})$ uma nova estimativa e $\Lambda = diag(\lambda)$.

Para o cálculo do FAIPA, B^k é uma matriz simétrica positiva definida podendo ser: a Matriz Identidade, a Hessiana da função Lagrangeana (18), ou a aproximação da Hessiana obtida através de um método quase-Newton.

Denominando $d_0^k = x_0^{k+1} - x^k$, o sistema (4.3) pode ser reescrito da seguinte forma:

$$\begin{array}{rcl} B^k d_0^k + \nabla g(x^k)\lambda_0^{k+1} & = & -\nabla f(x^k) \\ \Lambda^k \nabla g(x^k)^\top d_0^k + G(x^k)\lambda_0^{k+1} & = & 0 \end{array} \tag{4.4}$$

onde d_0 é uma direção de descida para f, no entanto pode não ser viável. A não viabilidade de d_0 ocorrerá sempre que alguma restrição $g_i(x^k)$ for próxima de zero. Dessa maneira, a direção d_0 torna-se tangente ao conjunto viável Ω.

Esse fato pode ser observado reescrevendo a segunda equação de (4.4)

$$\lambda_i^k \nabla g_i(x^k)^\top d_0^k + g_i(x^k)\lambda_{0i}^{k+1} = 0, \quad i = 1, \cdots, m \tag{4.5}$$

que resulta em $\nabla g_i(x^k)^\top d_0^k = 0$ para algum i tal que $g_i(x^k) = 0$.

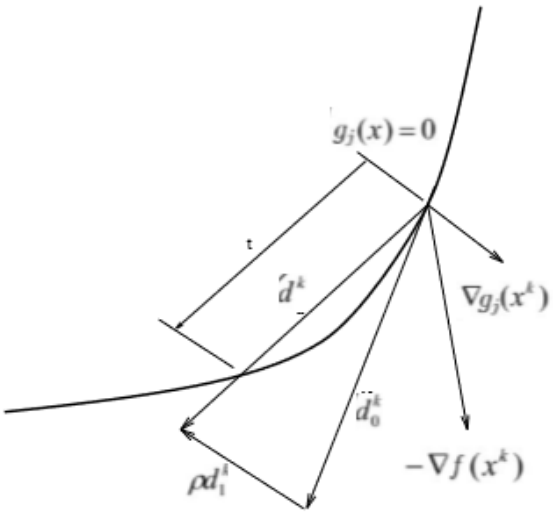

Figura 4.1: Direção viável.

Buscando obter a viabilidade para uma direção d_0^k, adicionamos um vetor negativo no lado direito de segunda equação de (4.4). Com isso, define-se um novo sistema linear nas variáveis d^k e $\bar{\lambda}^k$:

$$\begin{array}{rcl} B^k d^k + \nabla g(x^k)\bar{\lambda}^{k+1} & = & -\nabla f(x^k) \\ \Lambda^k \nabla g(x^k)^\top d^k + G(x^k)\bar{\lambda}^{k+1} & = & -\rho^k \lambda^k \end{array} \tag{4.6}$$

onde ρ^k é um número positivo e $\bar{\lambda}$ uma nova estimativa para λ.

A segunda equação de (4.6) pode ser expressa igualmente por:

$$\lambda_i^k \nabla g_i(x^k)^\top d^k + g_i(x^k)\bar{\lambda}_i^{k+1} = -\rho^k \lambda_i^k, \quad i = 1, \cdots, m. \tag{4.7}$$

Observa-se que, d^k é uma direção viável, pois $\nabla g_i(x^k)^\top d^k = -\rho^k < 0$ para todo $i \in I(x^k)$.

A inclusão de um termo negativo no lado direito da segunda equação de (4.6) acarreta em um desvio na direção d_0 para o interior da região viável, sendo esse desvio propocional ao parâmetro ρ.

Como d_0 é uma direção de descida para f, podemos limitar superiormente o parâmetro ρ fazendo com que a direção d seja também uma direção de descida [4]. Sendo $d_0^\top \nabla f(x) < 0$, podemos impôr a seguinte condição:

$$\left(d^k\right)^\top \nabla f(x^k) \leqslant \xi (d_0^k)^\top \nabla f(x^k), \quad \xi \in (0,1) \tag{4.8}$$

que resulta em $\left(d^k\right)^\top \nabla f(x^k) < 0$. Logo, d^k é uma direção viável e de descida para f em x^k.

Apesar da direção d^k ser de descida, a função f ainda apresenta um decrescimento maior ao longo da direção d_0^k.

O cálculo da direção d^k pode ser obtido através da combinação de dois sistemas lineares com a mesma matriz. Para isto, considera-se o sistema auxiliar nas variáveis d_1^k e λ_1^k dado a seguir:

$$\begin{aligned} B^k d_1^k + \nabla g(x^k)\lambda_1^{k+1} &= 0 \\ \Lambda^k \nabla g(x^k)^\top d_1^k + G(x^k)\lambda_1^{k+1} &= -\lambda^k. \end{aligned} \tag{4.9}$$

Para obter a direção viável e de descida d^k, calcula-se (d_0^k, λ_0^{k+1}) resolvendo o sistema (4.4) e depois calcula-se (d_1^k, λ_1^{k+1}) resolvendo o sitema (4.9). A direção d^k é obtida como sendo:

$$d^k = d_0^k + \rho^k d_1^k. \tag{4.10}$$

e para calcular $\bar{\lambda}^{k+1}$:

$$\bar{\lambda}^{k+1} = \lambda_0^{k+1} + \rho^k \lambda_1^{k+1}.$$

Para calcular ρ^k se substitui d^k definido em (4.10), na condição (4.8). Dessa forma:

$$\rho^k \leqslant (\xi - 1)\, \frac{\left(d_0^k\right)^\top \nabla f(x^k)}{\left(d_1^k\right)^\top \nabla f(x^k)}$$

caso $d_1^\top \nabla f(x) > 0$. Se $d_1^\top \nabla f(x) \leqslant 0$, toma-se

$$\rho^k \leqslant \varphi \left\| d_0 \right\|^2,$$

para algum parâmetro $\varphi > 0$.

4.2.3 Cálculo do arco

Alguns problemas de otimização apresentam restrições de desigualdade altamente não lineares. Para esse tipo de problemas os algoritmos de pontos interiores demonstram uma baixa velocidade de convergência em consequência do comprimento do passo ser inferior ao unitário.

Para resolver este problema, é realizada uma busca linear ao longo de um arco no interior da região viável, com finalidade de definir um arco tangente à direção d^k obtida em (4.10).

Para determinar o arco, leva-se em conta a curvatura das restrições de desigualdade. Essa medida determina o fator de viabilidade, $\tilde{\omega}$, que pode ser estabelecido como uma aproximação de derivadas segundas das restrições.

$$\tilde{\omega} = g\left(x + d\right) - g\left(x\right) - \nabla g^{\top}\left(x\right) d. \tag{4.11}$$

A partir disto, uma nova direção $\tilde{d}$ é obtida incluindo o fator de viabilidade $\tilde{\omega}$ no termo independente do sistema linear (4.9)

$$\begin{array}{rcl} B^k \tilde{d}^k + \nabla g(x^k)\tilde{\lambda}^{k+1} & = & 0 \\ \Lambda^k \nabla g(x^k)^{\top} \tilde{d}^k + G(x^k)\tilde{\lambda}^{k+1} & = & -\Lambda^k \tilde{\omega}^k. \end{array} \tag{4.12}$$

Dessa forma, define-se o arco tangente às direções d e $\tilde{d}$ dado por:

$$A(t) = x^k + td^k + t^2\tilde{d}^k \tag{4.13}$$

onde $t \in \mathbb{R}$.

O novo ponto da sequência, x^{k+1}, será determinado por um $t > 0$ que verifique as condições de uma busca linear no arco. Isso significa que a partir do ponto x^k pode-se caminhar ao longo do arco até um novo ponto x^{k+1} dentro de Ω. A figura (4.2) apresenta o arco viável quando temos duas variáveis e uma restrição ativa.

O algoritmo FDIPA é definido como sendo o FAIPA onde se considera $\tilde{d} = 0$. Neste caso a busca linear é realizada através de uma reta ao longo da direção d, como se mostra na figura (4.3).

Essa busca linear pode ser feita através do critério de Armijo [15].

4.2.4 Busca linear

Seja x^k um ponto interior a Ω e d uma direção de descida para f em x^k. O critério de Armijo para a busca linear define o passo t como sendo o primeiro da sequência

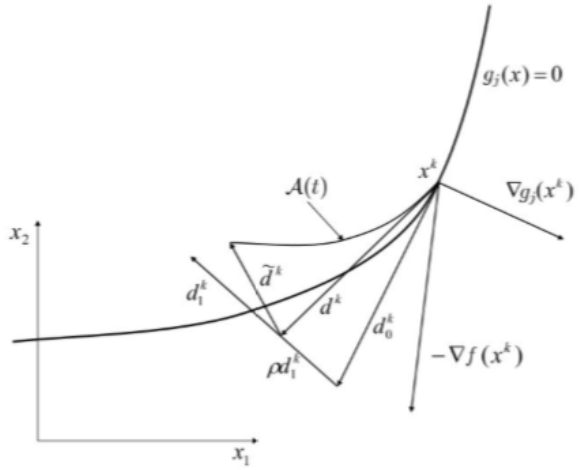

Figura 4.2: Direção de busca do FAIPA.

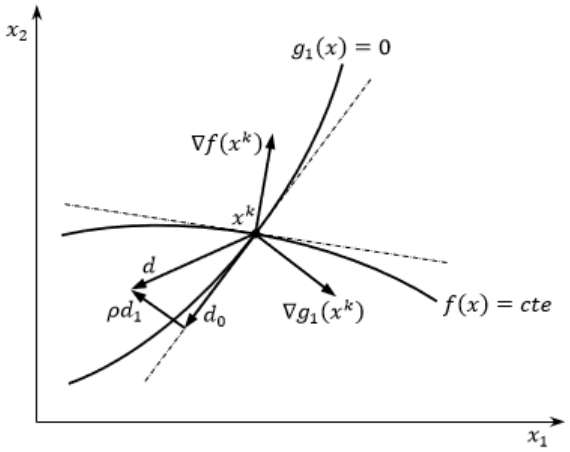

Figura 4.3: Direção de busca do FDIPA.

$\{1, \nu, \nu^2, \nu^3, \ldots\}$ tal que as seguintes condições sejam verificadas:

$$\begin{array}{rcl} f(A(t)) & \leqslant & f(x^k) + t\eta {d^k}^\top \nabla f(x^k) \\ g(A(t)) & < & 0 \end{array} \tag{4.14}$$

onde $\nu \in (0,1)$, $\eta \in (0,1)$ e A está definido em (4.13).

4.2.5 Atualizações do FAIPA

Ao final de cada iteração, o algoritmo FAIPA realiza uma atualização dos dados que serão empregados na próxima iteração.

- Atualiza-se o ponto: $x^{k+1} = x^k + td^k + t^2\tilde{d}^k$;
- Atualiza-se a matriz B^k por uma nova matriz simétrica definida positiva;
- Atualiza-se o vetor $\lambda^{k+1} > 0$.

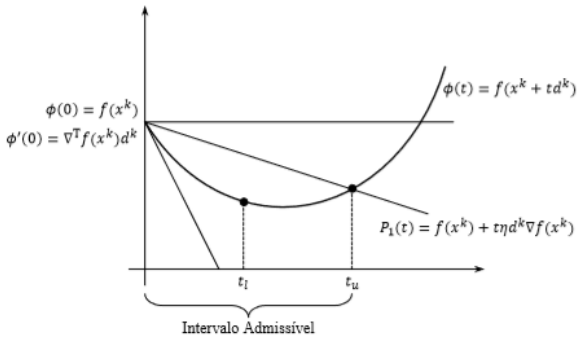

Figura 4.4: Busca linear de Armijo.

Na seção seguinte coloca-se o passo a passo do FAIPA.

4.2.6 Algoritmo FAIPA

Parâmetro. $\xi, \eta, \nu \in (0,1)$ e $\varphi > 0$.

Dados. $x \in int\,(\Omega)$, $\lambda \in \mathbb{R}^m$, $\lambda > 0$, $B \in \mathbb{S}^n_{++}$.

Passo 1. Cálculo da direção viável de descida d.

1.1 Resolver o sistema linear (4.4) para obter $d_0 \in \mathbb{R}^n$e $\lambda_0 \in \mathbb{R}^m$.

1.2 Resolver o sistema linear (4.9) para obter $d_1 \in \mathbb{R}^n$e $\lambda_1 \in \mathbb{R}^m$.

1.3 Se $d_1^\top \nabla f(x) > 0$ faça

$$\rho = min\left\{\varphi \left\|d_0\right\|^2, (\xi - 1) d_0^\top \nabla f(x) / d_1^\top \nabla f(x)\right\} \tag{4.15}$$

caso contrário define-se

$$\rho = \varphi \left\|d_0\right\|^2. \tag{4.16}$$

1.4 Calcule d como sendo

$$d = d_0 + \rho d_1. \tag{4.17}$$

Passo 2. Busca em arco.

2.1 Calcule

$$\tilde{\omega} = g\left(x^k + d^k\right) - g\left(x^k\right) - \nabla g\left(x^k\right)^\top d^k. \tag{4.18}$$

2.2 Resolva o sistema linear (4.12) para obter $\tilde{d} \in \mathbb{R}^n$ e $\tilde{\lambda} \in \mathbb{R}^m$.

2.3 Encontre t, o primeiro na sequência $\{1, \nu, \nu^2, \nu^3, \ldots\}$ que verifique

$$\begin{aligned} f(x^k + td^k + t^2\tilde{d}^k) &\leqslant f(x^k) + t\eta {d^k}^\top \nabla f(x^k) \\ g_i(x^k + td^k + t^2\tilde{d}^k) &< 0, \quad i = 1, \cdots, m. \end{aligned} \tag{4.19}$$

Passo 3. Atualizações.

3.1 Definir um novo ponto como sendo $x^{k+1} = x^k + td^k + t^2\tilde{d}^k$ e novos valores para $B \in \mathbb{S}^n_{++}$, $\lambda \in \mathbb{R}^m$, $\lambda > 0$.

3.2 Voltar ao Passo 1.

4.3 FAIPA-SDP

Nesta seção apresenta-se um novo algoritmo para programação semidefinida denominado *Feasible Arc Interior Point Algorithm for semidefinite programming, FAIPA-SDP*. A presente técnica resolve o problema de otimização (2.9).

Assume-se que a função de restrição está definida como $G : \mathbb{R}^n \to \mathbb{S}^m$, isto é $G(x) \in \mathbb{S}^m$.

Este algoritmo é uma extensão do FAIPA para atender restrições semidefinidas. Ele é considerado um algoritmo de pontos interiores, pois todo ponto gerado por ele pertence ao interior da região viável do problema. Igualmente como no FAIPA, é feita uma busca linear em um arco para obter o novo ponto interior a região viável.

4.3.1 Descrição do algoritmo

Este algoritmo representa uma técnica geral para solucionar o problema de programação semidefinida (2.9), podendo este problema ser não convexo.

O procedimento é idêntico ao do FAIPA:

1. Calcular uma direção de descida e viável $d = d_0 + \rho d_1$.
2. Calcular uma direção de restauração $\tilde{d}$.

3. Fazer uma busca linear em arco $A(t) = x + td + t^2\tilde{d}$ para encontar o novo ponto.

Os sistemas lineares que devem ser resolvidos no FAIPA-SDP são diferentes aos do FAIPA, pois as condições de KKT para o problema de programação não linear são diferentes aos do problema de programação semidefinida.

4.3.2 Cálculo da direção de descida

Os sistemas lineares do FAIPA são iterações tipo Newton para encontrar as raizes da função (4.2). Analogamente, os sistemas lineares do FAIPA-SDP são iterações tipo Newton para encontrar as raízes da seguinte função $\psi : \mathbb{R}^n \times \mathbb{R}^{\bar{m}} \to \mathbb{R}^n \times \mathbb{R}^{\bar{m}}$:

$$\psi(x,\lambda) = \begin{bmatrix} \psi_l(x,\lambda) \\ \psi_c(x,\lambda) \end{bmatrix} = \begin{bmatrix} \nabla f(x) + \nabla G(x)\lambda \\ svec(sym(\Lambda G(x))) \end{bmatrix} \tag{4.20}$$

onde $\lambda = svec(\Lambda)$ e $\Lambda \in \mathbb{S}^m$.

Usa-se a parte simétrica do produto $\Lambda G(x)$ para que o domínio e o contradomínio de ψ sejam iguais. Por outro lado, $sym(\Lambda G(x)) = 0$ é equivalente a $\Lambda G(x) = 0$ em virtude das equivalências (2.28). Portanto, encontrar raizes de ψ significa encontrar x e Λ tais que $G(x)\Lambda = 0$.

As raizes da função $\psi(x,\lambda)$ são calculadas através do método de Newton. Para escrever o sistema linear do método de Newton é necessário calcular a matriz Jacobiana de ψ. O cálculo da Matriz Jacobiana será dado através da derivada da função ψ com respeito a x e Λ.

A definição do produto simétrico de Kronecker (6), facilita o cálculo da matriz Jacobiana de ψ, pois:

$$\begin{aligned} \psi_c(x,\lambda) &= svec(sym(\Lambda G(x) I)) = [\Lambda \circledast I] svec(G(x)) \\ \psi_c(x,\lambda) &= svec(sym(I \Lambda G(x))) = [I \circledast G(x)] svec(\Lambda). \end{aligned} \tag{4.21}$$

Da primeira equação de (4.21) e por (2.14) tem-se:

$$\nabla_x \psi_c(x,\lambda) = (\Lambda \circledast I)\nabla G(x)^\top.$$

Considerando-se a segunda equação em (4.21):

$$\nabla_\lambda \psi_c(x,\lambda) = (I \circledast G(x)).$$

Desta forma, a matriz Jacobiana de ψ será dada por:

$$\nabla\psi(x,\lambda) = \begin{bmatrix} \nabla_x\psi_l(x,\lambda) & \nabla_\lambda\psi_l(x,\lambda) \\ \nabla_x\psi_c(x,\lambda) & \nabla_\lambda\psi_c(x,\lambda) \end{bmatrix} = \begin{bmatrix} \nabla_{xx}L(x,\lambda) & \nabla G(x) \\ (\Lambda \circledast I)\nabla G(x)^\top & (I \circledast G(x)). \end{bmatrix} \quad (4.22)$$

A fim de obter um menor custo computacional, se utiliza uma matriz simétrica definida positiva B no lugar da Hessiana do Lagrangeano $\nabla_{xx}L(x,\lambda)$. A matriz B pode ser uma aproximação quasi-Newton, a matriz identidade ou qualquer outra matriz definida positiva. Dessa forma, define-se a matriz do sistema linear W, que é dada por uma aproximação de (4.22) como segue:

$$W(x,B,\Lambda) = \begin{bmatrix} B & \bigtriangledown G(x) \\ (\Lambda \circledast I) \bigtriangledown G(x)^\top & (I \circledast G(x)) \end{bmatrix}. \quad (4.23)$$

É demostrado por [9] que $W(x,B,\Lambda)$ é invertível quando $B \succ 0$, Λ comuta com $G(x)$ e x é regular no sentido da definição (2.30).

Desta forma, o sistema linear de um método tipo Newton para encontrar as raízes de ψ é o seguinte:

$$\begin{bmatrix} B & \nabla G(x) \\ (\Lambda \circledast I)\nabla G(x)^\top & (I \circledast G(x)) \end{bmatrix} \begin{bmatrix} x_0 - x \\ \lambda_0 - \lambda \end{bmatrix} = - \begin{bmatrix} \nabla f(x) + \nabla G(x)\lambda \\ svec(sym(\Lambda G(x))) \end{bmatrix}. \quad (4.24)$$

Definindo a direção $d_0 = x_0 - x$ e simplificando (4.24) tem-se:

$$\begin{bmatrix} B & \nabla G(x) \\ (\Lambda \circledast I)\nabla G(x)^\top & (I \circledast G(x)) \end{bmatrix} \begin{bmatrix} d_0 \\ \lambda_0 \end{bmatrix} = \begin{bmatrix} -\nabla f(x) \\ 0 \end{bmatrix} \quad (4.25)$$

ou

$$\begin{aligned} Bd_0 + \nabla G(x)\lambda_0 &= -\nabla f(x) \\ (\Lambda \circledast I)\nabla G(x)^\top d_0 + (I \circledast G(x))\lambda_0 &= 0. \end{aligned} \quad (4.26)$$

Pode-se prova que se na solução de (4.26) resulta $d_0 = 0$, então x é um ponto de KKT.

Se d_0 não é nulo, foi provado em [9] que d_0 é de descida para f em x_k, no entanto, quando x_k está próximo da fronteira da região viável Ω, d_0 é tangente a esta região. Na próxima seção se mostra como calcular, usando a mesma matriz W, uma direção de descida e por sua vez viável.

4.3.3 Cálculo da direção de descida viável

Como apresentado por Herskovits em [23], para caracterizar a direção viável acrescenta-se uma quantidade negativa do lado direito da segunda equação de (4.26).

Com isso, define-se um outro sistema linear com a mesma matriz $W(x, B, \Lambda)$ na variáveis $(d, \bar{\lambda})$:

$$\begin{array}{rcl} Bd + \nabla G(x)\bar{\lambda} & = & -\nabla f(x) \\ (\Lambda \circledast I)\nabla G(x)^\top d + (I \circledast G(x))\bar{\lambda} & = & -\rho\lambda \end{array} \tag{4.27}$$

onde ρ um número real positivo e $\lambda = svec(\Lambda)$.

Com base nos sistemas lineares (4.26) e (4.27) vemos que para um $\rho > 0$, quanto menor for o seu valor mais a direção d tenderá a d_0. Por outro lado, se ao invés disso tivermos um valor cada vez maior para ρ, a tendência é que a direção d seja desviada para o interior da região viável [9]. Para garantir que esse desvio gere uma direção de descida, deve-se limitar superiormente o parâmetro ρ.

Uma forma de fazer isto é considerar:

$$d^\top \nabla f(x) \leqslant \xi d_0^\top \nabla f(x) \tag{4.28}$$

onde $\xi \in (0, 1)$.

O cálculo de d pode realizado como sendo

$$d = d_0 + \rho d_1 \tag{4.29}$$

onde d_1 é solução do seguinte sistema linear:

$$\begin{bmatrix} B & \nabla G(x) \\ (\Lambda \circledast I)\nabla G(x)^\top & (I \circledast G(x)) \end{bmatrix} \begin{bmatrix} d_1 \\ \lambda_1 \end{bmatrix} = \begin{bmatrix} 0 \\ -\lambda \end{bmatrix}. \tag{4.30}$$

Usando as mesmas ideia que no FAIPA, considera-se (4.28) e obtem-se:

$$\rho \leqslant (\xi - 1)\, \tfrac{(d_0)^\top \nabla f(x)}{(d_1)^\top \nabla f(x)}$$

quando $d_1^\top \nabla f(x) > 0$. Caso contrário, define-se:

$$\rho \leqslant \varphi \left\| d_0 \right\|^2.$$

4.3.4 Cálculo do arco

Utilizando informações estabelecidas na configuração x^k, definimos o fator de viabilidade $\tilde{\omega}$, que será obtido por meio de uma aproximação das derivadas segunda das restrições no ponto $x^k + d$, ao resolvermos um novo sistema linear com a mesma matriz W, como segue:

$$\tilde{\omega}^k = svec\left(G\left(x^k + d^k\right) - G\left(x^k\right) - smat\left(\nabla G^\top \left(x^k\right)^\top d\right)\right). \tag{4.31}$$

Como definido no FAIPA, o fator de viabilidade será acrescido no termo independente do sistema (4.30) de modo de obter $\tilde{d}$ como solução do seguinte sistema linear:

$$\begin{bmatrix} B^k & \nabla G(x^k) \\ (\Lambda^k \circledast I)\nabla G(x^k)^\top & (I \circledast G(x^k)) \end{bmatrix} \begin{bmatrix} \tilde{d}^k \\ \tilde{\lambda}^{k+1} \end{bmatrix} = \begin{bmatrix} 0 \\ -(\Lambda^k \circledast I)\tilde{\omega}^k \end{bmatrix} \tag{4.32}$$

ou, equivalentemente

$$\begin{array}{rcl} B^k\tilde{d}^k + \nabla G(x^k)\tilde{\lambda}^{k+1} & = & 0 \\ (\Lambda^k \circledast I)\nabla G(x^k)^\top \tilde{d}^k + (I \circledast G(x^k))\tilde{\lambda}^{k+1} & = & -(\Lambda^k \circledast I)\tilde{\omega}^k. \end{array} \tag{4.33}$$

Assim, define-se o seguinte arco tangente as direções d e $\tilde{d}$:

$$A(t) = x^k + td^k + t^2\tilde{d}^k. \tag{4.34}$$

Para encontrar uma nova estivativa x^{k+1} realiza-se uma busca linear em arco.

4.3.5 Busca linear

De forma análoga ao do FAIPA, o critério de Armijo define-se da seguinte forma: Escolher t como sendo o primeiro da sequência $\{1, \nu, \nu^2, \nu^3, \ldots\}$ tal que:

$$\begin{array}{rcl} f(A(t)) & \leqslant & f(x^k) + t\eta {d^k}^\top \nabla f(x^k) \\ G(A(t)) & \prec & 0 \end{array} \tag{4.35}$$

onde $\nu \in (0,1)$, $\eta \in (0,1)$ e A está definido em (4.34).

Considerando $\tilde{d} = 0$, este algoritmo se denomina Algoritmo de Pontos Interiores por Direções Viáveis (*Feasible Direction Interior Point Algorithm, FDIPA-SDP*. Este algoritmo foi proposto em [9].

4.3.6 Atualizações do FAIPA-SDP

Define-se agora um resultado que viabiliza a atualização dos parâmetros Λ e de B para o algoritmo FAIPA-SDP.

Em [9], é provado que sendo x^k um ponto regular do problema (2.9), $B \in \mathbb{S}^n_{++}$, $\Lambda \in \mathbb{S}^k_{++}$ e se Λ^k comutar com $G(x^k)$ em todas as iterações, teremos a garantia de que a matriz do sistema $W(x^k, B^k, \Lambda^k)$ é invertivel. Para a atualização de B pode ser empregada uma atualização Quasi-Newton.

Na próxima seção se enumeram os passos do FAIPA-SDP.

4.3.7 Algoritmo FAIPA-SDP

Parâmetro. $\xi, \eta, \nu \in (0,1)$ e $\varphi > 0$.

Dados. $x \in int(\Omega)$, $B \in \mathbb{S}^n_{++}$ e $\Lambda = \mu I \in \mathbb{S}^m_{++}$.

Passo 1. Cálculo da direção viável de descida d.

1.1 Resolver o sistema linear (4.26) para obter $d_0 \in \mathbb{R}^n$ e $\lambda_0 \in \mathbb{R}^{\bar{m}}$.

Se $d_0 = 0$, parar.
1.2 Resolver o sistema linear (4.30) para obter $d_1 \in \mathbb{R}^n$ e $\lambda_1 \in \mathbb{R}^{\bar{m}}$.

1.3 Se $d_1^\top \nabla f(x) > 0$ faça

$$\rho = min\left\{\varphi \left\|d_0\right\|^2, (\xi - 1) d_0^\top \nabla f(x) / d_1^\top \nabla f(x)\right\}. \tag{4.36}$$

Se $d_1^\top \nabla f(x) \leqslant 0$ define-se

$$\rho = \varphi \left\|d_0\right\|^2. \tag{4.37}$$

1.4 Calcule d como sendo

$$d = d_0 + \rho d_1. \tag{4.38}$$

Passo 2. Busca em arco.

2.1 Calcule

$$\tilde{\omega}^k = svec\left(G\left(x^k + d^k\right) - G\left(x^k\right) - smat\left(\nabla G^\top \left(x^k\right)^\top d\right)\right). \tag{4.39}$$

2.2 Resolva o sistema linear (4.32) para obter $\tilde{d}^k \in \mathbb{R}^n$.

2.3 Encontre t, o primeiro na sequência $\{1, \nu, \nu^2, \nu^3, \ldots\}$ que verifique

$$\begin{array}{c} f(x^k + td^k + t^2\tilde{d}^k) \leqslant f(x^k) + t\eta {d^k}^\top \nabla f(x^k) \\ g_i(x^k + td^k + t^2\tilde{d}^k) \prec 0. \end{array} \tag{4.40}$$

Passo 3. Atualizações.

3.1 Defina um novo ponto como sendo $x^{k+1} = x^k + td^k + t^2\tilde{d}^k$ e novos valores para $B \in \mathbb{S}^n_{++}$ e $\Lambda = \mu I \in \mathbb{S}^m_{++}$.

3.2 Voltar ao Passo 1.

Capítulo 5

Testes numéricos

No presente capítulo, se apresentam problemas resolvidos com o FDIPA-SDP e o FAIPA-SDP. São resolvidos problemas de programação semidefinida com restrições LMI e BMI.

5.1 Descrição dos problemas em termos do problema de programação semidefinida

Problema de mínimo volume

O problema de minimização do volume (3.30) tem $n = n_{elem}$ variáveis de projeto. O tamanho da função de restrição é $m = 1 + n_{dof} + n_{dof} + n_{elem}$. A função objetivo é $f(x) = \sum_{i=1}^{n_{elem}} x_i$ e a função de restrição $G^{minvol}(x) \in \mathbb{S}^m$ é

$$G^{minvol}(x) = G_0^{minvol} + \sum_{i=1}^{nelem} x_i G_i^{minvol}$$

onde:

$$G_0^{minvol} = \begin{bmatrix} \begin{bmatrix} -\bar{\gamma} & f^\top \\ f & 0 \end{bmatrix} & \begin{matrix} 0 & 0 \end{matrix} \\ \begin{matrix} 0 \\ 0 \end{matrix} & \begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix} \end{bmatrix} \tag{5.1}$$

e

$$G_i^{minvol} = \begin{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & -K_i \end{bmatrix} & \begin{matrix} 0 \\ 0 \end{matrix} & \begin{matrix} 0 \\ 0 \end{matrix} \\ \begin{matrix} 0 & 0 \end{matrix} & -K_i + \bar{\lambda} M_i & 0 \\ \begin{matrix} 0 & 0 \end{matrix} & 0 & e_i e_i^\top \end{bmatrix} \tag{5.2}$$

onde K_i é a matriz de rigidez do elemento i, M_i é a matriz de massa do elemento i e e_i é o i-ésimo vetor da base canônica de $\mathbb{R}^{n_{elem}}$.

A restrição $G^{minvol}(x) \preccurlyeq 0$ é LMI, portanto, $R(x,d)$ definido em (2.16), é nulo. Por este motivo $\tilde{\omega}^k$, definido em (4.31), também será nulo em toda iteração k. Assim, o FAIPA-SDP, quando aplicado ao problema (3.30) terá o mesmo número de iterações que o FDIPA-SDP.

Em vista disso, para resolver o problema (3.30) se executará apenas o FDIPA-SDP, pois não haverá diferença com o FAIPA-SDP.

Problema de maximização da frequência fundamental

O problema de maximização da frequência fundamental (3.31) tem $n = n_{elem} + 1$ variáveis de projeto (número de elementos mais a variável λ que servirá para minimizar a frequência fundamental). O tamanho da função de restrição é $m = 1 + n_{dof} + 1 + n_{dof} + n_{elem}$. A função objetivo é $f(x,\lambda) = -\lambda$ e a função de restrição $G^{maxfreq}(x,\lambda) \in \mathbb{S}^m$ é

$$G^{maxfreq}(x,\lambda) = G_0^{maxfreq} + \sum_{i=1}^{nelem} x_i G_i^{maxfreq} + \lambda \sum_{i=1}^{nelem} x_i \tilde{M}_i \in \mathbb{R}^m$$

onde:

$$G_0^{maxfreq} = \begin{bmatrix} \begin{bmatrix} -\bar{\gamma} & f^\top \\ f & 0 \end{bmatrix} & 0 & 0 & 0 \\ 0 & -\bar{V} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.3}$$

,

$$G_i^{maxfreq} = \begin{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & -K_i \end{bmatrix} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -K_i & 0 \\ 0 & 0 & 0 & e_i e_i^\top \end{bmatrix} \tag{5.4}$$

e

$$\tilde{M}_i = \begin{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & M_i & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.5}$$

onde K_i, M_i e e_i são os mesmos objetos descritos na seção anterior.

A restrição $G^{maxfreq}(x,\lambda) \preccurlyeq 0$ é BMI, portanto, $R(x,d)$ definido em (2.16), é não nulo. Por este motivo $\tilde{\omega}^k$, definido em (4.31), não será nulo em toda iteração k. Assim, o FAIPA-SDP, quando aplicado ao problema (3.31) poderá ter um número de iterações diferente que do FDIPA-SDP.

Neste sentido, será interessante executar o FDIPA-SDP ou o FAIPA-SDP para resolver o problema (3.31) pois podem apresentar número de iterações diferentes.

5.1.1 Dados

Nesta seção se apresentam os dados fixos de todos os testes numéricos.

Em todos os testes realizados os dados para ambos algoritmos de otimização foram os seguintes:

1. Cálculo do ρ
   - $\xi = 0.8$ $(0 < \xi < 1,\ \nabla f(x)^\top d \leqslant \xi \nabla f(x)^\top d_0)$;
   - $\phi = 1$ $(\phi > 0, \phi |d_0|^2)$.
2. Busca linear
   - $\eta = 0.1$ $(0 < \eta < 1,\ f(x + td) \leqslant f(x) + t\eta d^\top \nabla f(x))$;
   - $\nu = 0.7$ $(0 < \nu < 1,\ t = 1, \nu, \nu^2, ...)$.
3. Condição de parada
   - $TOL_{D0} = 10^{-4}$, tolerância da norma de d_0;
   - $TOL_{GLAG} = 10^{-4}$, tolerância da norma do Lagrangeano;
   - $TOL_{COMPL} = 10^{-4}$, tolerância da complementaridade.
4. Atualização de B: BFGS.
5. Atualização de Λ: $\Lambda_0 + constante * I$.

O algoritmo termina sua execução quando são verificados as seguintes condições, simultaneamente:

1. $|d_0^k| \leqslant TOL_{D0}$;
2. $|\nabla_x L(x^k, \Lambda_0^k)| \leqslant TOL_{GLAG}$;
3. $|tr(\Lambda_0^k G(x^k))| \leqslant TOL_{COMPL}$;

onde d_0 e Λ_0 são calculados no sistema linear (4.26).

Foram considerados os seguintes dados para todas as estruturas:

- $Area = 0.01$, área inicial de todos os elementos;
- $E = 10.0$, módulo de Young de todos os elementos;
- $\rho^{mat} = 1.0$, densidade de todos os elementos;

- $intensidade = 1.0$, intensidade da força no nó.

Os parâmetros dos problemas de otimização são:

- $\bar{\gamma} = 1.0$, complacência máxima admissível;
- $\bar{\lambda} = 0.05$, mínima frequência admissível.

5.1.2 Construção dos problemas de otimização

Os exemplos foram construidos para que a solução ótima do problema de máxima frequência x^* tenha $\lambda_{min}(x^*) = 0.05$. Para que a estrutura tenha esta frequência mínima como solução ótima, foi empregada a proposição (3.4.1). Se encontra a solução ótima do problema de mínimo volume (3.30) considerando os parâmetros $\bar{\gamma}$ e $\bar{\lambda}$ com os valores indicados na seção anterior. Resolvendo o problema de mínimo volume (3.30) se obtém V^*, o valor ótimo da função objetivo. A proposição (3.4.1) afirma que a solução ótima de (3.31) x^*, quando se consideram os parâmetros $\bar{\gamma}$ e $\bar{V} = V^*$, tem-se $\lambda_{min}(x^*) = \bar{\lambda}$.

5.1.3 Geometria e condições de contorno

Sejam as seguintes estruturas:

1. Exemplo 1: Estrutura em formato de L com 8 nós, $n_{elem} = 19$, $n_{dof} = 16 - 3$, fixa no nó inferior esquerdo, apoiada no nó inferior direito, força vertical para abaixo aplicada no nó inferior do lado direito.

2. Exemplo 2: Estrutura quadrada com 3×3 nós, $n_{elem} = 27$, $n_{dof} = 18 - 6$, fixa nos nós do lado esquerdo e força horizontal aplicada no nó central do lado direito.

3. Exemplo 3: Estrutura em formato de T com 10 nós, $n_{elem} = 28$, $n_{dof} = 20 - 3$, fixa no nó do lado inferior esquerdo, apoiada no nó inferior direito, forças verticais aplicada para abaixo no nó inferior do lado esquerdo e no nó inferior do lado direito.

4. Exemplo 4: Estrutura em formato retangular com 10 nós, $n_{elem} = 33$, $n_{dof} = 20 - 3$, fixa nos nós inferior esquerdo e direito, forças verticais para abaixo aplicada nos nós do lado inferior.

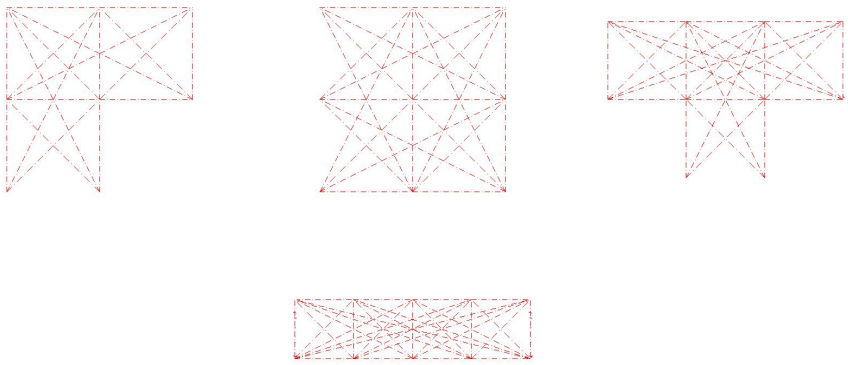

Figura 5.1: Geometria das treliças dos exemplos 1, 2, 3 e 4. As barras estão respresentadas por linhas tracejadas.

5.2 Execução dos algoritmos

A programação do FAIPA-SDP foi realizada modificando o código em MATLAB existente do FDIPA-SDP desenvolvido em [9] e [25]. Foi adicionado a este código: a resolução do terceiro sistema linear (4.12) e a modificação da busca linear de Armijo para que fosse feita no arco (4.34). Para o cálculo das matrizes de rigidez e de massa também foram feitas rotinas em MATLAB.

5.2.1 Ponto viável

Em quase todas as execuções dos testes foi necessário buscar um ponto viável para poder iniciar o processo de otimização.

Todas as execuções dos algoritmos foram iniciadas com os dados apresentados na seção (5.1.1).

Para encontrar um ponto viável em Ω, caso a estrutura viole alguma restrição do problema de otimização, resolve-se o seguinte problema auxiliar:

$$\begin{aligned} &\min_{x\in\mathbb{R}^n, z\in\mathbb{R}} z \\ &sujeito\ a: \quad G(x) - zI \preccurlyeq 0 \end{aligned} \tag{5.6}$$

onde G pode ser G^{minvol} ou $G^{maxfreq}$. Quando, no processo de otimização, z torna-se negativo, $G(x) \prec 0$, portanto x é um ponto viável.

5.2.2 Resultados

Todos os resultados apresentados correspodem ao problema de máxima frequência (3.31), otimizados com o FAIPA-SDP.

Emprega-se a seguinte simbologia:

- iter: número de iteração k;
- iter0: número de iterações para obter um ponto viável;
- iter FDIPA-SDP: número de iterações do FDIPA-SDP;
- iter FAIPA-SDP: número de iterações do FAIPA-SDP;
- L0min FDIPA-SDP: menor autovalor de Λ_0 do FDIPA-SDP;
- L0min FAIPA-SDP: menor autovalor de Λ_0 do FAIPA-SDP;
- f: valor da função objetivo $f(x^k)$;
- |d0|: norma de d_0;
- |glag|: norma de $\nabla L(x^k, \Lambda_0^k)$;
- |tr(L0*G)|: traço de $\Lambda_0^k G(x^k)$.

A matriz de multiplicadores de Lagrange Λ_0 obtida no final das execuções do FDIPA-SDP e do FAIPA-SDP foram todas indefinidas, porém o menor autovalor de Λ_0 é muito próximo de zero. Concretamente, os valores do menor autovalor de Λ_0 na última iteração para cada exemplo estão apresentadas na tabela (5.1).

Tabela 5.1: Convergência do FDIPA-SDP e FAIPA-SDP, menor autovalor de L0.

Exemplo	L0min FDIPA-SDP	L0min FAIPA-SDP
1	-3.15018e-010	-1.53521e-009
2	-3.72456e-010	-7.17332e-010
3	-1.16651e-009	-4.70084e-009
4	-2.18849e-011	-1.56644e-006

Tabela 5.2: Número de iterações do FDIPA-SDP e FAIPA-SDP.

Exemplo	iter0 FDIPA-SDP	iter0 FAIPA-SDP	iter FDIPA-SDP	iter FAIPA-SDP
1	18	44	18	44
2	17	18	17	18
3	14	14	56	56
4	68	76	45	117

Como se mostra na tabela (5.2), o FAIPA-SDP não realiza menos iterações que o FDIPA-SDP.

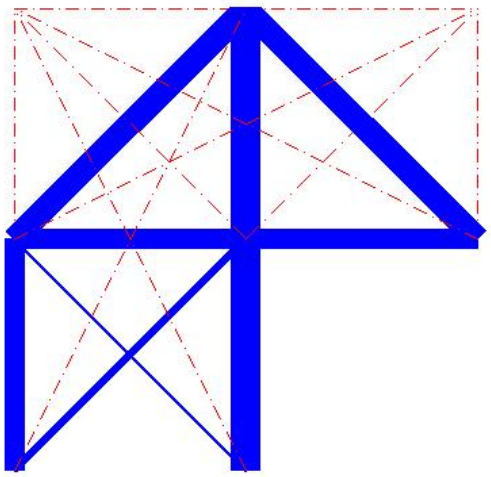

Figura 5.2: Solução ótima do exemplo 1.

Tabela 5.3: Exemplo 1: iterações da otimização.

iter	f	|d0|	|glag|	|tr(L0*G)|	t
1	0.290839				
2	0.205382	0.0907318	0.0907318	1.23497	1
3	0.131131	0.185004	0.185004	0.721042	0.49
4	0.0193374	0.224638	0.108055	0.376146	0.7
5	-0.0248899	0.104216	0.0812591	0.158317	1
6	-0.0371396	0.12153	0.109304	0.0773835	0.7
7	-0.0407547	0.139398	0.0533722	0.0166432	0.2401
8	-0.0424362	0.0819317	0.0359998	0.0108481	0.2401
9	-0.0446522	0.0687643	0.0343492	0.00455813	0.49
10	-0.0478858	0.0688725	0.0261309	0.00264104	1
11	-0.048491	0.0418386	0.0080073	0.000559502	0.7
12	-0.0487333	0.0231799	0.00432084	0.000232086	1
13	-0.0487671	0.00491268	0.00256133	2.89659e-005	1
14	-0.0487777	0.00323195	0.00187831	5.4695e-006	1
15	-0.0487829	0.00457874	0.0010845	1.34027e-006	0.7
16	-0.0487853	0.00392557	0.000931005	1.02789e-006	0.7
17	-0.0487869	0.00253081	0.000838254	6.16084e-007	1
18	-0.0487882	0.00223526	0.000620384	1.82632e-007	1
19	-0.0487888	0.00267638	0.000549818	6.67794e-008	0.49
20	-0.0487896	0.00283955	0.00061332	2.62904e-007	1
21	-0.0487904	0.00238394	0.000660757	5.93467e-008	1
22	-0.0487922	0.00478218	0.000720056	1.73746e-007	1
23	-0.0487985	0.016804	0.000980396	3.83011e-007	1
24	-0.0488121	0.0394454	0.00201194	1.15222e-006	1
25	-0.0488339	0.071065	0.00328689	1.22929e-006	1
26	-0.0488644	0.097064	0.00431011	2.81964e-006	1
27	-0.0489447	0.22382	0.00525091	6.20932e-006	1
28	-0.0490517	0.582222	0.00653471	1.99158e-005	0.49
29	-0.04931	0.647317	0.0062122	6.23601e-005	1
30	-0.0494109	0.79828	0.00741341	0.000221171	0.343
31	-0.0495645	0.323412	0.0046041	0.000145674	1
32	-0.0496888	0.710761	0.0057973	0.000275698	0.49
33	-0.0497841	0.150717	0.0028666	9.87405e-005	1
34	-0.0498399	0.197852	0.00358257	9.02564e-005	0.7
35	-0.0498634	0.0444985	0.00183062	3.51555e-005	0.7
36	-0.0498775	0.0134733	0.00156735	1.54741e-005	1
37	-0.0498805	0.00219713	0.000558158	3.0076e-006	1
38	-0.0498816	0.00354951	0.000329084	6.33036e-007	1
39	-0.0498824	0.00636991	0.000467146	2.2206e-007	0.7
40	-0.0498837	0.0103756	0.000760147	2.52192e-007	1
41	-0.0498847	0.00801075	0.000990268	1.0659e-007	1
42	-0.0498876	0.0195482	0.00112634	1.88968e-007	1
43	-0.0498998	0.0747895	0.00135894	7.88189e-007	1
44	-0.0499338	0.293573	0.00195691	3.94848e-006	0.7

Tabela 5.4: Exemplo 2: Iterações da otimização.

iter	f	|d0|	|glag|	|tr(L0*G)|	t
1	0.0237014				
2	0.00976814	0.303231	0.303231	0.0930998	0.057648
3	-0.0130335	0.0584369	0.0584369	0.0304335	0.49
4	-0.0245329	0.0211719	0.0935126	0.0393523	0.7
5	-0.0350955	0.0138335	0.282998	0.0480119	1
6	-0.0405581	0.0101071	0.103367	0.0194875	0.7
7	-0.0451457	0.00842121	0.0406135	0.0119428	0.7
8	-0.0480434	0.00529152	0.0641828	0.00536212	0.7
9	-0.0493714	0.00235241	0.0644739	0.00231581	0.7
10	-0.0500058	0.000996586	0.0672886	0.000979034	0.7
11	-0.0502412	0.000352148	0.0677007	0.000345164	0.7
12	-0.0503176	0.000112501	0.0688846	0.00010993	0.7
13	-0.0503412	3.76018e-005	0.0639819	3.36255e-005	0.7
14	-0.0503484	2.03345e-005	0.057541	1.01585e-005	0.7
15	-0.0503507	4.60899e-005	0.037672	3.07252e-006	0.7
16	-0.0503513	6.64586e-005	0.0122139	1.03285e-006	0.49
17	-0.0503517	5.49608e-005	0.00211868	6.96299e-007	0.7
18	-0.0503519	2.05856e-005	0.000171909	3.06248e-007	0.7

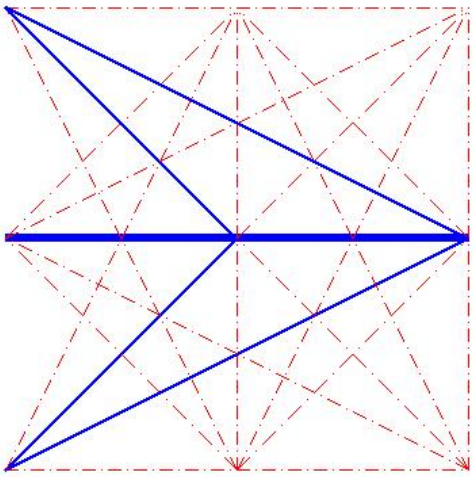

Figura 5.3: Solução ótima do exemplo 2.

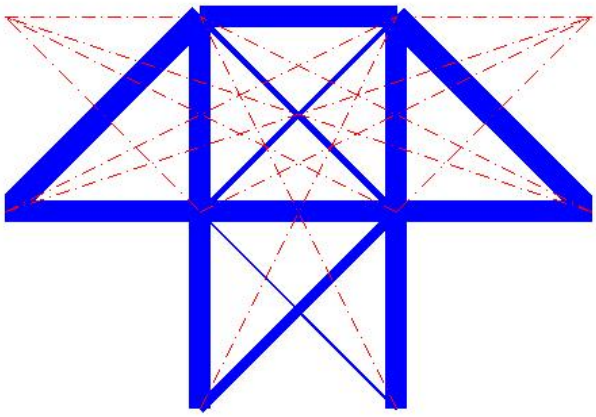

Figura 5.4: Solução ótima do exemplo 3.

Tabela 5.5: Exemplo 3: Iterações da otimização.

iter	f	|d0|	|glag|	|tr(L0*G)|	t
1	0.134206				
2	0.0935115	0.0415168	0.0415168	0.650868	1
3	0.0213009	0.076835	0.076835	0.337451	1
4	-0.0039895	4 0.093545	0.0783184	0.182148	0.7
5	-0.015972	0.0566584	0.0485504	0.0437697	1
6	-0.0209288	0.0885145	0.0329903	0.0156439	0.7
7	-0.0265478	0.101276	0.0239513	0.00502055	1
8	-0.0341431	0.150443	0.0319689	0.00399549	1
9	-0.0368105	0.0808517	0.0280265	0.0047001	0.7
10	-0.0382641	0.0473543	0.0316906	0.00204217	0.7
11	-0.0402253	0.0651743	0.0281751	0.00105883	1
12	-0.0414751	0.0892661	0.0209861	0.000582812	0.7
13	-0.041628	0.0621133	0.0146313	0.000451713	0.16807
14	-0.0417962	0.0357037	0.0189867	0.00037984	0.343
15	-0.0424461	0.0508753	0.0173514	0.000268067	1
16	-0.0426719	0.0582984	0.00594253	0.000134903	0.49
17	-0.0428172	0.0467894	0.0051934	0.000171624	0.7
18	-0.0428979	0.0227052	0.00411481	7.1728e-005	1
19	-0.0429225	0.0065563	0.00348518	1.182e-005	1
20	-0.0429582	0.0129771	0.00318155	4.79319e-006	1
21	-0.0429844	0.0237138	0.00312416	4.34151e-006	0.49
22	-0.0430248	0.0344146	0.00396385	1.34643e-005	0.7
23	-0.0430582	0.0244455	0.00397309	1.05943e-005	1
24	-0.0430995	0.0250389	0.00422559	3.46252e-006	1
25	-0.0431858	0.0490677	0.00566124	6.66189e-006	1
26	-0.0433021	0.0993277	0.00886928	1.24524e-005	0.7
27	-0.0436004	0.193349	0.0144765	3.05584e-005	1
28	-0.0438612	0.178566	0.0147089	3.49507e-005	1
29	-0.0443811	0.299917	0.018545	4.63711e-005	1
30	-0.0453182	0.456554	0.0233338	0.000137521	1
31	-0.0463712	0.371837	0.0157126	0.000428498	1
32	-0.0480705	0.778252	0.0375055	0.00106574	1
33	-0.0490005	0.230847	0.012789	0.000659335	1
34	-0.0491109	0.0531718	0.0102501	0.000253766	0.343
35	-0.0492952	0.0320564	0.0102269	0.000176277	0.7
36	-0.0495148	0.0380669	0.0344881	0.000101238	0.7
37	-0.0497044	0.0445994	0.033496	3.2547e-006	0.7
38	-0.0498109	0.0368918	0.0125461	0.000117385	0.7
39	-0.0498583	0.0252451	0.0025955	2.17678e-005	0.7
40	-0.0498893	0.0162466	0.00261421	1.78374e-005	1
41	-0.049903	0.00819687	0.00555332	5.89955e-006	1
42	-0.0499379	0.0215304	0.0084995	5.1315e-006	1
43	-0.0499612	0.0401529	0.0047613	9.89726e-006	0.343
44	-0.049973	0.0113626	0.00272985	1.73287e-005	0.49
45	-0.0499807	0.00571095	0.00199883	9.54259e-006	0.7

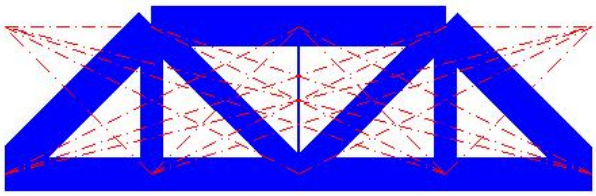

Figura 5.5: Solução ótima do exemplo 4.

Tabela 5.6: Exemplo 4: Iterações da otimização.

iter	f	|d0|	|glag|	|tr(L0*G)|	t
1	0.997401				
2	0.992585	0.0592839	0.0592839	2.32192	0.0823543
3	0.992515	0.0661156	0.0661156	2.29296	0.00113989
4	0.985652	0.148428	0.134925	2.1128	0.0096889
5	0.983023	0.00287603	0.00110067	2.34813	1
6	0.982465	0.482409	0.171868	1.91375	0.00113989
7	0.98033	0.566202	0.082383	2.02204	0.00332329
8	0.966706	0.123578	0.0174058	2.26905	0.117649
9	0.964613	0.576268	0.0921233	2.01614	0.00474756
10	0.959426	0.139297	0.0239154	2.24154	0.0403536
11	0.955313	0.571992	0.118774	1.98552	0.0096889
12	0.947247	0.165967	0.0358055	2.21842	0.057648
13	0.946387	0.514498	0.135728	1.99564	0.00162841
14	0.936934	0.134958	0.0383017	2.21971	0.0823543
15	0.935365	0.562104	0.140491	1.97171	0.00332329
16	0.919247	0.111936	0.0255836	2.19009	0.16807
17	0.917356	0.539801	0.114115	1.92269	0.00474756
18	0.908733	0.12291	0.0237699	2.17066	0.0823543
19	0.906137	0.463131	0.0794725	1.96838	0.00678223
20	0.904377	0.209655	0.0519502	2.0932	0.0096889
21	0.90138	0.374253	0.0768348	1.99924	0.0096889
22	0.900581	0.321358	0.0749315	2.02212	0.00332329
23	0.898362	0.388829	0.109914	1.98831	0.00678223
24	0.896137	0.293076	0.0925009	2.04103	0.0096889
25	0.892774	0.406594	0.14695	1.97952	0.0096889
26	0.891662	0.275218	0.0640238	2.04475	0.00474756
27	0.883371	0.164878	0.069281	2.13725	0.057648
28	0.880929	0.485572	0.062389	1.9427	0.00678223
29	0.875433	0.533684	0.0850108	1.90552	0.0138413
30	0.86284	0.123603	0.0362187	2.12636	0.117649
31	0.86156	0.514463	0.122705	1.89649	0.00332329
32	0.851329	0.100071	0.0201751	2.11843	0.117649
33	0.849155	0.492612	0.0798815	1.8984	0.00474756
34	0.843415	0.111329	0.0177454	2.08876	0.057648
35	0.840885	0.441122	0.0697086	1.92258	0.00678223
36	0.835941	0.213959	0.0458469	2.05047	0.0282475
37	0.833343	0.43603	0.114666	1.95207	0.00678223
38	0.82399	0.192212	0.0856131	2.08942	0.057648
39	0.820482	0.416202	0.118957	1.95666	0.0096889
...	...	...	...	...	...
117	-0.0500115	0.567081	0.00021476	4.11433e-005	0.7

Capítulo 6

Considerações finais

6.1 Conclusões

Neste trabalho foi desenvolvido o algoritmo FAIPA-SDP para resolver problemas de programação semidefinda. Este novo método foi baseado no algoritmo FAIPA proposto por Herskovits [6].

Mostra-se aqui que problemas de programação não linear com restrições vetoriais podem ser resolvidos através dos algoritmos FDIPA e FAIPA. Estes dois algoritmos partem de um ponto viável e geram uma sequência de pontos viáveis com redução do valor da função objetivo a cada iteração. No caso do FDIPA, a cada iteração, são resolvidos dois sistemas lineares com a mesma matriz de coeficientes. Para o FAIPA, além dos dois sistemas lineares, se resolve mais um sistema linear com a mesma matriz para definir um arco viável e de descida. Para obter um novo ponto, é realizada uma busca linear inexata ao longo deste arco.

Com base nos algoritmos citados acima desenvolve-se o algoritmo FAIPA-SDP. Essa técnica segue basicamente os mesmos passos do FAIPA para o cálculo da direção viável e de descida. No entanto, para expressar a matriz Jacobiana do sistema obtido pelo método de Newton utiliza-se o produto simétrico de Kronecker.

O presente trabalho se concentra na busca de soluções para problemas de programação semidefinida onde exitem restrições nos autovalores de funções matriciais.

Apresenta-se os seguintes problemas de otimização estrutural: minimização do volume e maximização da frequência natural de estruturas. Para estes problemas foram desenvolvidas duas formulações diferentes: a formulação clássica e a formulação de programação semidefinida. Além disso, mostra-se em (3.4.1) a relação existente entre estes dois problemas.

Para fins do nosso estudo são consideradas estruturas treliças bidimensionais. Essas estrutura são discretizada e analisada através do método dos elementos finitos.

Foram feitos testes númericos onde se resolvem os problemas (3.30) e (3.31). O

primeiro deles é um problema convexo, contudo o segundo é não convexo.

Todas as aplicações que apresentamos neste trabalho foram resolvidas utilizando o mesmo conjunto de parâmetros.

Os resultados numéricos obtidos mostram que o FAIPA-SDP converge para pontos de KKT dos problemas (3.30) e (3.31), pois as condições de parada foram verificadas em todos os testes. Adicionalmente observou-se, como indica a tabela (5.1), que o menor autovalor de Λ_0 é quase nulo. No entanto, o número de iterações do FAIPA-SDP não foi menor que o número de iterações do FDIPA-SDP como se mostra na tabela (5.2).

6.2 Sugestões para trabalhos futuros

Destacam-se os seguintes tópicos como sugestões para trabalhos futuros:

1. Investigar problemas de programação semidefinida onde a derivada segunda de G em relação as variáveis de projeto seja significativa. Talvez para estes problemas o FAIPA-SDP apresente melhor desempenho que o FDIPA-SDP, pois a busca linear no arco $A(t)$ poderia ser melhor que a busca linear na direção d uma vez que a curvatura na fronteira de Ω é maior para estes problemas.

2. Provar a convergência global do FAIPA-SDP.

3. Propor e resolver com o FAIPA-SDP, novos problemas de otimização de programação semidefinida com restrições mecânicas de deslocamento e/ou de tensão.

4. Propor uma implementação computacional eficiente do FAIPA-SDP para resolver problemas com um grande número de elementos. Isto significa estudar como resolver numericamente os sistemas lineares do FAIPA-SDP quando $n >> 1$ e $m >> 1$, ou seja resolver um problema de otimização de grande porte.

Referências Bibliográficas

[1] FARID, A. "Interior point methods in semidefinite programming with applications to combinatorial opitimization", *SIAM Journal on Optimization*, v. 42, n. 2, pp. 26–30, fev. 1993.

[2] BOYD, S., GHAOUI, L. E., FERON, E., et al. *Linear Matrix Inequalities in System and Control Theory*. 15 ed. Philadelphia, SIAM Studies in Applied Mathematics IAM, 1994.

[3] VANDENBERGHEY, L., BOYD, S. "Semidefinite Programming", *SIAM*, v. 38, pp. 8–9, mar. 1996.

[4] HERSKOVITS, J. "A view on Nonlinear Optimization", *KLUWER ACADEMIC PUBLISHERS - Holland*, pp. 1–41, nov. 1995.

[5] HERSKOVITS, J., SANTOS, G. *Feasible Arc Interior Point Algorithm for Nonlinear Optimization*. Computational mechanics, new trends and applications, CIMNE, Barcelona, 1998.

[6] HERSKOVITS, J., SANTOS, G. *A Feasible arc interior point algorithms for nonlinear optimization*. Computational Mechanics and New trends and Applications hal-00758803, CIMNE, Barcelona, Spain, 2014.

[7] AROZTEGUI, M., HERSKOVITS, J., ROCHE, J. R., et al. "A feasible direction interior point algorithm for nonlinear semidefinite programming." *Structural and Multidisciplinary Optimization*, nov. 2014.

[8] HERSKOVITS, J. "A Feasible Directions Interior-Point Technique for Nonlinear Optimization", *JOTA-Journal of Optimization Theory and Applications*, v. 99, pp. 53–58, 1998.

[9] AROZTEGUI, M. *Técnicas de programação semidefinida e aplicações em otimização de material*. D.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 2010.

[10] SHAPIRO, A. "First and Second Order Analysis of Nonlinear Semidefinite Programs", *Mathematical Programming*, v. 77, pp. 301–325, 1994.

[11] SHAPIRO, A. "On uniqueness of Lagrange multipliers in optimization problems subject to cone constraints", *SIAM Journal on Optimization*, v. 7, pp. 508 518, 1997.

[12] ACHTZIGER, W., KOCVARA, M. "Structural topology optimization with eigenvalues", *Institute of Applied Mathematics, University of Dortmund, Germany (SIAM Journal on Optimization)*, p. 36, 2006.

[13] MAURER, H., ZOWE, J. "First and Second-Order Necessary and Sufficient Optimality conditions for Infinite-Dimentional Programming Problems", *Mathematical Programming*, v. 16, pp. 98–110, 1979.

[14] STINGL, M. *On the Solution of Nonlinear Semidefinite Programs by Augmented Lagrangian Methods*. D.Sc., University of Erlangen Nurnberg, 2005.

[15] LUENBERGER, D. G. *Linear and Nonlinear programming*. 2 ed. Caixa Postal 68503, 21945-970 Rio de Janeiro, BRAZIL, Springer, 1984.

[16] ACHTZIGER, W., KOCVARA, M. "Structural topology optimization with eigenvalues", *Institute of Applied Mathematics, University of Dortmund, Germany (SIAM Journal on Optimization)*, p. 15, 2007.

[17] HUGHES, T. J. R. *The Finite Element Method, Linear Static and Dynamic Finite Element Analisys*. 1 ed. Englewood Cliffs, New Jersey 07632, Prentice-Hall, 1987.

[18] SAAD, Y. *Numerical Methods for Large Eigenvalue Problems*. Revised edition ed. Philadelphia, SIAM, 2011.

[19] CLARKE, F. H. *Optimization and nonsmooth analysis*. Wiley, New York, Canadian Mathematical Society Series in Mathematics, 1983.

[20] ALIZADEH, F., HAEBERLY, J., OVERTON, M. L. "Primal-Dual Interior-Point Methods for Semidefinite Programming: Convergence Rates, Stability and Numerical Results", *SIAM Journal on Optimization*, v. 8, pp. 746–768, 1998.

[21] VANDENBERGHEY, L., BOYD, S. "Applications of Semidefinite Programming", *SIAM*, ago. 1998.

[22] ROCHE, J. R., BAZAN, E., HERSKOVITS, J., et al. "A feasible direction algorithm for general nonlinear semidefinite programming." *Structural and Multidisciplinary Optimization*, ago. 2016.

[23] HERSKOVITS, J. *An Interior Point Technique for Nonlinear Optimization*. 1808, INRIA BP 105, Le Chesnay CEDEX, France, 1992.

[24] PANIER, E., TITS, A. "A QP-Free, Globally Convergent, Locally Superlinearly Convergent Algorithm for Inequality Constrained Optimization", *SIAM Journal on Control and Optimization*, v. 26, pp. 788–811, 1988.

[25] BAZAN, E. G. *Otimização estrutural mediante programação semidefinida*. M.Sc., UFRJ/COPPE, Rio de Janeiro, RJ, Brasil, 2012.