UNIVERSIDADE FEDERAL DO PARANÁ

SOLANGE REGINA DOS SANTOS

DESENVOLVIMENTO DE ALGORITMOS MATEMÁTICOS APLICADOS A CONFIABILIDADE ESTRUTURAL

CURITIBA
2012

SOLANGE REGINA DOS SANTOS

DESENVOLVIMENTO DE ALGORITMOS MATEMÁTICOS APLICADOS A CONFIABILIDADE ESTRUTURAL

Tese apresentada ao Programa de Pós-Graduação em Métodos Numéricos em Engenharia, Área de Concentração em Programação Matemática, dos Setores de Tecnologia e de Ciências Exatas da Universidade Federal do Paraná, como requisito parcial à obtenção do título de Doutor em Ciências.

Orientador:

Prof. Dr. Luiz Carlos Matioli

CURITIBA

2012

TERMO DE APROVAÇÃO

SOLANGE REGINA DOS SANTOS

DESENVOLVIMENTO DE ALGORITMOS MATEMÁTICOS APLICADOS A CONFIABILIDADE ESTRUTURAL

Tese aprovada como requisito parcial para a obtenção do grau de Doutor em Ciências, no Programa de Pós-Graduação em Métodos Numéricos em Engenharia - Programação Matemática da Universidade Federal do Paraná, pela seguinte banca examinadora:

Orientador: _____

Prof. Dr. Luiz Carlos Matioli
Programa de Pós-Graduação em Métodos Numéricos em Engenharia - PPGMNE, UFPR

_____

Prof. Dr. Ademir Alves Ribeiro
Programa de Pós-Graduação em Métodos Numéricos em Engenharia - PPGMNE, UFPR

_____

Prof. Dr. André Teófilo Beck
Departamento de Engenharia de Estruturas, USP

_____

Prof. Dr. Anselmo Chaves Neto
Programa de Pós-Graduação em Métodos Numéricos em Engenharia - PPGMNE, UFPR

_____

Prof. Dr. Ricardo Biloti
Departamento de Matemática Aplicada, UNICAMP

Curitiba, 27 de abril de 2012.

A minha mãe e, acima de tudo, minha amiga, Aurora.
Ao meu namorado e companheiro, Marcio.

Agradecimentos

A Deus pela proteção na caminhada e pelas graças concedidas.

À minha mãe, Aurora, pelos ensinamentos, exemplo de força, de determinação e, acima de tudo, pelo amor dedicado às filhas e por cumprir com sabedoria o papel de pai e mãe.

Ao meu namorado, Marcio, por todo amor, pela compreensão e pelo apoio na conquista desta etapa tão importante em minha vida.

Ao meu sobrinho, Paulo Henrique, e às minhas irmãs, Sandra e Suellen, pelo incentivo.

Aos meus amigos, Adriano, Gislaine, Juliano e Tatiane, pela amizade, pelo incentivo, pelos momentos de desconstração e por tornarem as viagens à Curitiba mais agradáveis e divertidas.

À minha avó Aparecida, por ter acolhido a mim e aos meus amigos em sua casa durante todo o doutorado. Sem a sua hospitalidade e generosidade, a realização desta etapa seria ainda mais díficil.

À Gislaine, por ter compartilhado comigo os desafios, as incertezas e os momentos de estudos desde a época do mestrado.

Ao meu orientador, professor Matioli, pelos ensinamentos, motivação, persistência, incentivo e, sobretudo por acreditar no meu "amadurecimento" acadêmico. Não posso deixar de agradecer à sua dedicação ao trabalho, pois mesmo quando esteve tão distante, na China, esteve sempre presente norteando a pesquisa.

Ao professor Anselmo, pelo auxílio e por ter nos apresentado este tema.

Ao professor André Teófilo Beck, pela participação na pesquisa.

Aos professores Ademir Alves Ribeiro e Ricardo Biloti, por aceitarem participar da banca examinadora dessa tese e pelas valiosas observações que fizeram, tornando o trabalho melhor.

Ao doutorando André Jacomel Torii, por sua contribuição.

À Universidade Federal do Paraná e ao Programa de Pós-Graduação em Métodos Numéricos em Engenharia, pela oportunidade de cursar o doutorado.

Aos professores do Programa de Pós-Graduação em Métodos Numéricos em Engenharia, pelos valiosos ensinamentos transmitidos.

À querida e prestativa Maristela Bandil, pelo seu trabalho e amizade.

À Universidade Estadual do Paraná, Campus Campo Mourão, por me possibilitar condições necessárias para a conclusão deste curso.

À Fundação Araucária, pelo apoio financeiro.

Lista de Figuras

Lista de Tabelas

Resumo

A resolução de problemas de confiabilidade estrutural pelo Método de Primeira Ordem requer a utilização de algoritmos de otimização para encontrar a menor distância entre a função de estado limite e a origem do espaço normal padrão. O algoritmo Hasofer-Lind-Rackwitz-Fiessler (HLRF), desenvolvido especificamente para este fim, tem sido eficiente, mas não robusto, pois não converge para um número significativo de problemas. Métodos de programação não linear, como por exemplo, Gradiente Projetado, Lagrangeano aumentado e Programação Quadrática Sequencial, já foram empregados na resolução de problemas de confiabilidade estrutural. Entretanto, diversas pesquisas e vários métodos de otimização foram desenvolvidos recentemente, especialmente com relação aos métodos de Lagrangeano aumentado, em que novas funções de penalidade e métodos modernos têm sido elaborados. Neste trabalho, três novos algoritmos de otimização são apresentados. O primeiro, denominado nHLRF, foi desenvolvido especificamente para aplicação ao problema de confiabilidade estrutural. Tal algoritmo baseia-se no algoritmo HLRF, no entanto, utiliza uma nova função de mérito diferenciável com as condições de Wolfe para a seleção do comprimento do passo na busca linear. Sob certas hipóteses, o algoritmo proposto gera uma sequência que converge para um minimizador local do problema. Os outros dois algoritmos propostos baseiam-se em Lagrangeanos aumentados. Ambos utilizam penalidades quadráticas e a estrutura dos métodos de Lagrangeano aumentado modernos empregados na resolução de problemas com restrições de desigualdade. Para este caso, foi mostrado que sob hipóteses convencionais a função Lagrangeano aumentado gerada pelos dois métodos tem um minimizador local. Os métodos de Lagrangeano aumentado com a penalidade clássica, bem como com as novas funções de penalidade, foram implementados em Matlab e os testes numéricos executados com problemas da coleção CUTEr. Para esses testes os algoritmos de Lagrangeano aumentado propostos mostraram-se mais eficientes do que o método de Lagrangeano aumentado clássico. O desempenho e a robustez dos novos algoritmos também foram comparados na resolução de problemas de confiabilidade estrutural. O novo algoritmo nHLRF mostrou-se mais robusto do que HLRF e iHLRF (versão melhorada do HLRF), e tão eficiente quanto o algoritmo iHLRF. Os dois métodos Lagrangeano aumentado propostos foram mais eficientes que o método clássico e mais robustos que os métodos baseados no algoritmo HLRF, principalmente na resolução de problemas com um número maior de variáveis aleatórias.

Palavras-chave: Confiabilidade estrutural, Métodos de Lagrangeano aumentado, algoritmos baseados em HLRF.

Abstract

Solution of structural reliability problems by the First Order method requires the use of optimization algorithms to find the smallest distance between a limit state function and the origin of standard Gaussian space. The Hasofer-Lind-Rackwitz-Fiessler (HLRF) algorithm, developed specifically for this purpose, has been shown to be efficient but not robust, as it fails to converge for a significant number of problems. Nonlinear programming methods, such as Projected Gradient, Augmented Lagrangian and Sequential Quadratic Programming, have already been tested in application to structural reliability problems. However, many studies and several optimization methods have been developed recently, specially about the augmented Lagrangian methods, in which new penalty functions and modern methods have been developed. In the present work, three new optimization algorithms are presented. The first algorithm, named nHLRF, was developed specifically for structural reliability problems. This algorithm is based on the HLRF, although it uses a new differentiable merit function with Wolfe conditions for step length selection in linear search. It is shown that, under certain assumptions, the proposed algorithm generates a sequence that converges to the local minimizer of the problem. The other two proposed algorithms are based on augmented Lagrangian. Both use quadratic penalties and the structure of modern augmented Lagrangian methods to solve problems with inequality constraints. For this case, it is shown that under conventional hypotheses, the augmented Lagrangian function generated by the two methods has a local minimizer. The augmented Lagrangian methods with the classical penalty as well as with the new penalty functions, were implemented in Matlab and numerical tests were performed with problems from CUTEr collection. For these tests, the proposed Augmented Lagrangian algorithms proved to be more efficient than the classical augmented Lagrangian method. Performance and robustness of the new algorithms were also compared in solving structural reliability problems. The new algorithm nHLRF proved to be more robust than HLRF and iHLRF (improved version of HLRF), and as efficient as the iHLRF algorithm. The two proposed augmented Lagrangian methods proved to be more efficient than the classical augmented Lagrangian method and more robust than the methods based on the HLRF algorithm, mainly in solving problems with large number of random variables.

Key-words: Structural reliability, augmented Lagrangian methods, HLRF-based algorithm.

Sumário

Introdução

Com o intuito de facilitar a compreensão de nossa pesquisa, apresentamos primeiramente os principais conceitos relacionados à confiabilidade estrutural e um breve resgate histórico referente às metodologias difundidas na literatura para a resolução do problema de confiabilidade estrutural.

Observamos que diversos são os interesses da engenharia estrutural, entre eles, destacamos o desenvolvimento de projetos que garantam um desempenho satisfatório, com funcionalidade, durabilidade, econômicos e, acima de tudo, seguros. Porém, existem incertezas associadas as variáveis de projeto que faz com que a estrutura corra o risco de falhar e não cumprir a finalidade para a qual foi projetada. Deste modo, para avaliar a possibilidade de falha são empregadas ferramentas da análise de confiabilidade estrutural, que é uma metodologia que permite avaliar com base nas variáveis de projeto e nas distribuições de probabilidade dessas variáveis a probabilidade de falha de uma estrutura.

Com base na formulação convencional da confiabilidade estrutural, para a obtenção da probabilidade de falha de uma estrutura é essencial definir o vetor de variáveis aleatórias $\mathbf{X}$, dado por

$$\mathbf{X} = (X_1, X_2, \ldots, X_n)^T,$$

que corresponde às incertezas associadas ao projeto, como por exemplo, à solicitação imposta à estrutura, à resistência, à geometria e os materiais.

A função desempenho

$$h(\mathbf{X}) = h(X_1, X_2, \ldots, X_n)$$

estabelece um limite entre o domínio de falha e o domínio seguro, ou seja,

$$\begin{aligned} D_f &= \{\mathbf{X} | h(\mathbf{X}) \leq 0\} \text{ é o domínio de falha,} \\ D_s &= \{\mathbf{X} | h(\mathbf{X}) > 0\} \text{ é o domínio de segurança.} \end{aligned}$$

A equação de estado limite é definida por $h(\mathbf{X}) = 0$. Consequentemente, a probabilidade de falha pode ser avaliada por

$$P_f = \int \ldots \int_{h(\mathbf{X}) \leq 0} f_{\mathbf{X}}(x_1, x_2, \ldots, x_n)\, dx_1 dx_2 \ldots dx_n,$$

onde $f_{\mathbf{X}}(x_1, x_2, \ldots, x_n)$ é a função densidade de probabilidade conjunta para os vetores de variáveis aleatórias e a integração é realizada sobre o domínio de falha, $h(\mathbf{X}) \leq 0$. O

cálculo da probabilidade de falha por meio da avaliação da integral múltipla não é fácil, pois: 1. envolve uma integral multi-dimensional; 2. a forma exata da função densidade de probabilidade conjunta das variáveis aleatórias raramente é conhecida; 3. a equação de estado limite $h(\mathbf{X}) = 0$ nem sempre é dada de forma analítica, mas como a solução de algum algoritmo numérico.

A solução direta da integral múltipla via Monte Carlo somente é possível quando a probabilidade de falha, P_f, não é muito pequena e/ou a função de estado limite é dada de forma analítica.

Soluções aproximadas podem ser obtidas eficientemente usando os Métodos de Confiabilidade de Primeira Ordem (*First Order Reliability Method* - FORM), ou de Segunda Ordem (*Second Order Reliability Method* - SORM). Para apresentarmos seus conceitos estabelecemos a seguinte notação: $\mathbf{X}$ e $\mathbf{Y}$ são vetores do $\mathbb{R}^n$ cujas componentes X_i e Y_i são, respectivamente, variáveis aleatórias do espaço original $\mathbb{X}$ (ou "espaço de projeto") e do espaço reduzido $\mathbb{Y}$; $\mathbf{x}$ e $\mathbf{y}$ são vetores do $\mathbb{R}^n$ cujas componentes x_i e y_i representam, respectivamente, valores específicos das variáveis aleatórias X_i e Y_i. No espaço de projeto ou espaço original as variáveis são dimensionais, enquanto que, no espaço reduzido as variáveis são adimensionais. A palavra "espaço" não é empregada aqui no sentido matemático.

Os métodos FORM e SORM envolvem a transformação das variáveis aleatórias do espaço original em variáveis aleatórias normais (ou Gaussianas) padrão. Na literatura é possível encontrar trabalhos, como por exemplo Haldar e Mahadevan [16] e Melchers [33], que fornecem detalhes sobre as transformações necessárias para mapear pontos do espaço original $\mathbb{X}$ para o espaço reduzido $\mathbb{Y}$. Existem formas diferenciadas de efetuar essa transformação, uma delas utiliza o Princípio da aproximação normal, proposto por Ditlevsen [13], que consiste em estimar os parâmetros de uma distribuição normal equivalente para cada variável no espaço de projeto. Considerações sobre estas transformações serão feitas no Capítulo 1.

Uma das principais etapas dos métodos FORM e SORM é a obtenção do ponto de projeto $\mathbf{y}^*$, ou seja, o ponto sob a superfície de falha mais próximo da origem, no espaço $\mathbb{Y}$. A distância (no espaço $\mathbb{Y}$) do ponto de projeto à origem do sistema é denominada de índice de confiabilidade e denotada por β.

Desta forma, a determinação de $\mathbf{y}^*$ consiste na resolução do seguinte problema de otimização restrito

$$(P) \qquad \begin{array}{ll} \text{minimizar} & \dfrac{1}{2}\mathbf{y}^T\mathbf{y} \\ \text{sujeito a} & h(\mathbf{y}) = 0, \end{array}$$

em que $h : \mathbb{R}^n \to \mathbb{R}$ é a superfície de falha e $h \in C^1$.

Uma vez que o ponto de projeto tenha sido localizado, a solução de primeira ordem

representa uma linearização da função de estado limite no ponto de projeto, resultando na estimativa linear da probabilidade de falha, dada por

$$P_f = \Phi(-\beta) = 1 - \Phi(\beta),$$

em que $\Phi(\cdot)$ é a função distribuição acumulada da variável normal padrão definida por

$$\Phi(\beta) = \int_{-\infty}^{\beta} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}y^2}\, dy.$$

A solução do problema (P) tem motivado o desenvolvimento de vários algoritmos, como por exemplo, o algoritmo HLRF desenvolvido por Hasofer e Lind [17] e Rackwitz e Fiessler [41].

O algoritmo HLRF tem se mostrado eficiente, mas não converge para muitos problemas [25, 26]. Na verdade, não há prova matemática de convergência para o algoritmo HLRF.

Liu e Kiureghian [26] propuseram uma versão modificada do algoritmo HLRF (denotada M-HLRF), incluindo uma função de mérito para induzir a convergência. Os autores mostraram que o algoritmo M-HLRF possui um desempenho superior ao algoritmo HLRF, apesar de nem sempre ser globalmente convergente.

Santosh *et al.* [50] apresentaram uma melhoria ao algoritmo HLRF usando a mesma função de mérito proposta por Liu e Kiureghian [26], combinada com busca linear utilizando a regra de Armijo. A prova de convergência para esse algoritmo também não foi apresentada.

Liu e Kiureghian [25, 26] compararam o desempenho de diferentes algoritmos na resolução do problema de confiabilidade (P). Os métodos Gradiente Projetado, Lagrangeano aumentado e Programação Quadrática Sequencial foram testados pelos autores e comparados com os algoritmos HLRF e M-HLRF. Nestes testes foram identificados a falta de robustez do algoritmo HLRF e do método de Lagrangeano aumentado, além disso, os métodos de Programação Quadrática Sequencial e M-HLRF mostraram-se mais eficientes do que os outros métodos para os problemas testados pelos autores.

Zhang e Kiureghian [58] desenvolveram uma melhoria ao algoritmo HLRF, denotada iHLRF (i refere-se a *improved*), introduzindo uma função de mérito não diferenciável e usando a regra de Armijo na seleção do comprimento do passo na busca linear. Os autores estabeleceram as condições sob as quais o algoritmo gera uma sequência que converge para um minimizador do problema (P).

Santos e Matioli [47] analisaram a eficiência do Método Duas Fases, desenvolvido por Luenberger [27], que combina os métodos de Penalização [57] e Gradiente Projetado [45] na determinação do ponto de projeto. A primeira fase do método consiste na aplicação do Método de Newton com o objetivo de reduzir a medida de inviabilidade, enquanto que

na segunda fase, o método de Cauchy é aplicado com o objetivo de melhorar a medida de otimalidade. Sendo assim, a cada iteração apenas um passo de Newton e um passo de Cauchy são realizados, tornando o método simples de ser implementado. A direção de Newton é substituída por uma direção relaxada que utiliza apenas informação de primeira ordem das funções envolvidas, motivando a aplicação desse método em problemas de confiabilidade estrutural.

Periçaro, Santos, Ribeiro e Matioli [37] apresentaram um comparação entre os algoritmos Duas Fases, HLRF e o algoritmo de Restauração Inexata que utiliza filtro como critério de avaliação do passo, proposto por Karas, Oening e Ribeiro [23]. O método de Restauração Inexata, proposto inicialmente por Martínez e Pilota [30], possui características semelhantes ao Método Duas Fases, embora sejam executados de forma diferente.

Neste trabalho propomos três novos algoritmos para a resolução do problema (P). O primeiro, baseado no algoritmo HLRF e denotado nHLRF (n refere-se a *new*), emprega uma função de mérito diferenciável e as condições de Wolfe na determinação do comprimento do passo. Ao contrário dos estudos anteriores, [25, 26, 50], apresentamos resultados de convergência para o algoritmo nHLRF. Os outros dois algoritmos, denotados LAPM (M refere-se a Matioli) e LAPB (B refere-se a Bertsekas), baseiam-se nos métodos de Lagrangeano aumentado. É importante mencionar que Liu e Kiureghian [26] atribuíram um desempenho insatisfatório ao método de Lagrangeano aumentado desenvolvido por Hestenes [18] e Powell [40], o qual utiliza uma função de penalidade quadrática clássica, denotado LAPC (C referente a clássico), para a resolução de problemas de confiabilidade estrutural. Porém, desde a publicação de [26], em 1991, muitas pesquisas foram realizadas sobre os métodos de Lagrangeano aumentado, como por exemplo [2, 7, 9, 14, 19, 20, 21, 24, 29, 32, 44, 51, 53] e, novas funções de penalidade e métodos de Lagrangeano aumentado modernos foram desenvolvidos. Sendo assim, há um espaço significativo para a aplicação dos métodos de Lagrangeano aumentado na solução de problemas de confiabilidade estrutural.

O trabalho está organizado da seguinte forma: no Capítulo 1 apresentamos inicialmente o problema fundamental de confiabilidade estrutural, em seguida, mostramos que para um caso mais geral, o índice de confiabilidade pode ser obtido pela resolução de um problema de otimização restrito e, por fim, apresentamos os métodos que podem ser empregados para transformar as variáveis do espaço original $\mathbb{X}$ para o espaço reduzido $\mathbb{Y}$. No Capítulo 2 apresentamos as versões clássica e melhorada do algoritmo HLRF e, também o novo algoritmo nHLRF, juntamente com os resultados de convergência. O Capítulo 3 é destinado a apresentação dos métodos de Lagrangeano aumentado com as novas funções de penalidade. Os resultados dos experimentos numéricos realizados com os problemas da coleção CUTEr também são apresentados nesse capítulo. E, para finalizar o trabalho, no Capítulo 4, descrevemos os resultados dos novos algoritmos de otimização aplicados a um conjunto de problemas de confiabilidade estrutural.

Capítulo 1

Confiabilidade Estrutural

Por simplicidade optamos por apresentar, inicialmente nesse capítulo, o problema fundamental de confiabilidade estrutural e os conceitos necessários para a sua resolução. Em seguida, estendemos a resolução do problema fundamental para o caso $n-$dimensional, considerando a equação de estado limite linear e não linear. Em seguida, apresentamos as técnicas empregadas para a transformação das variáveis originais em variáveis normais padrão e independentes. Para finalizar o capítulo, apresentamos o método iterativo, FORM, desenvolvido unicamente para o cálculo da probabilidade de falha.

Os conceitos apresentados a seguir estão baseados principalmente nos trabalhos de Beck [1], Haldar e Mahadevan [16] e Melchers [33].

1.1 Problema fundamental de confiabilidade estrutural

Com o objetivo de compreender o significado do índice de confiabilidade, bem como de sua relação com o problema de otimização restrito (P), consideramos inicialmente um problema bidimensional em que a equação de estado limite é dada por

$$h(\mathbf{X}) = X_1 - X_2 = 0. \tag{1.1}$$

Na equação (1.1) as variáveis X_1 e X_2 representam a resistência e a carga impostas a uma determinada estrutura, respectivamente. Deste modo, a falha ocorre quando $X_1 < X_2$, ou seja, quando $h(\mathbf{X}) < 0$. Valores positivos para $h(\mathbf{X})$ indicam segurança, enquanto que, um valor nulo corresponde à condição de estado limite da estrutura, ou seja, uma condição sob a qual uma estrutura ou elemento estrutural torna-se inadequado para desempenhar a função proposta.

Considerando que a probabilidade de falha é uma medida da propensão à violação de estados limites, o cálculo da probabilidade de falha é dado por

$$P_f = P[\{\mathbf{X} \in D_f\}] = P[\{h(\mathbf{X}) \leq 0\}].$$

Portanto, para a equação (1.1) temos que

$$P_f = P[\{X_1 \leq X_2\}] = P[\{X_1 - X_2 \leq 0\}]. \tag{1.2}$$

A avaliação de (1.2) constitui o denominado *problema fundamental de confiabilidade estrutural*.

Suponha que as variáveis aleatórias X_1 e X_2 sejam independentes e normalmente distribuídas com médias μ_{X_1} e μ_{X_2}, e desvios padrão σ_{X_1} e σ_{X_2}, respectivamente. Assim, $Z = h(\mathbf{X})$ também é uma variável aleatória, com média e variância iguais a $\mu_Z = \mu_{X_1} - \mu_{X_2}$ e $\sigma_Z^2 = \sigma_{X_1}^2 + \sigma_{X_2}^2$, respectivamente. Os valores de μ_Z e σ_Z^2 são obtidos aplicando a regra da adição (subtração) de variáveis aleatórias.

Deste modo, a equação (1.2) torna-se

$$P_f = P(Z \leq 0). \tag{1.3}$$

Para o desenvolvimento da equação (1.3), considere agora uma variável aleatória normal padrão Y, ou seja, uma variável aleatória com média nula e desvio padrão unitário obtida pela padronização de Z, ou seja,

$$Y = \frac{Z - \mu_Z}{\sigma_Z}.$$

Temos que,

$$P_f = P(Z \leq 0) = P\left(Y \leq \frac{0 - \mu_Z}{\sigma_Z}\right) = \Phi\left(-\frac{\mu_Z}{\sigma_Z}\right),$$

onde $\Phi(\cdot)$ é a função distribuição acumulada da variável normal padrão, definida como

$$\Phi(y) = \int_{-\infty}^{y} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}t^2}\, dt.$$

Observa-se que na variável normal padronizada Y obtemos uma medida para P_f que depende da razão entre a média e o desvio padrão de Z. Tal razão é representada por β e definida como o índice de confiabilidade, isto é

$$\beta = \frac{\mu_Z}{\sigma_Z}. \tag{1.4}$$

Assim, podemos reescrever a equação (1.3) da seguinte forma

$$P_f = \Phi(-\beta).$$

Observa-se que o índice de confiabilidade β é de extrema importância na confiabilidade estrutural e está diretamente relacionado à probabilidade de falha. No caso da

equação de estado limite (1.1) o índice de confiabilidade é dado por

$$\beta = \frac{\mu_{X_1} - \mu_{X_2}}{\sqrt{\sigma_{X_1}^2 + \sigma_{X_2}^2}}, \tag{1.5}$$

e é somente válido quando as variáveis X_1 e X_2 são Gaussianas.

Os conceitos apresentados até aqui também podem ser utilizados no cálculo da probabilidade de falha para problemas envolvendo equações de estado limite com um número maior de variáveis. Sendo assim, apresentamos nas Seções 1.2 e 1.3 a extensão da solução do problema fundamental para o caso $n-$dimensional considerando o caso não linear e linear, respectivamente.

1.2 Extensão da solução do problema fundamental: função desempenho não linear $n-$dimensional

Nessa seção vamos considerar a função desempenho, $h(\mathbf{X})$, não linear. De acordo com a equação (1.5) o índice de confiabilidade é definido como a razão entre a média e o desvio padrão de $h(\mathbf{X})$.

No caso em que a função desempenho é não linear, a média e o desvio padrão de $h(\mathbf{X})$ podem ser aproximados por meio da expansão de $h(\mathbf{X})$ em série de Taylor de primeira ordem, centrada nos valores médios das variáveis aleatórias $\mu = (\mu_{X_1}, \ldots, \mu_{X_n})^T$. Obtemos assim, a equação de estado limite linear

$$\tilde{h}(\mathbf{X}) = h(\mu) + \sum_{i=1}^{n} \frac{\partial h}{\partial X_i} (X_i - \mu_{X_i}). \tag{1.6}$$

Utilizando a equação (1.6) o índice de confiabilidade é dado por

$$\beta = \frac{E\left[\tilde{h}(\mathbf{X})\right]}{\sqrt{V\left[\tilde{h}(\mathbf{X})\right]}}, \tag{1.7}$$

em que $E\left[\tilde{h}(\mathbf{X})\right]$ e $V\left[\tilde{h}(\mathbf{X})\right]$ são aproximações de primeira ordem para média e variância de $h(\mathbf{X})$, respectivamente, dadas por

$$\begin{aligned} E\left[\tilde{h}(\mathbf{X})\right] &= E\left[h(\mu) + \sum_{i=1}^{n} \frac{\partial h}{\partial X_i} (X_i - \mu_{X_i})\right] = E\left[h(\mu)\right] + \sum_{i=1}^{n} E\left[\frac{\partial h}{\partial X_i} (X_i - \mu_{X_i})\right] = \\ &= h(\mu) + \sum_{i=1}^{n} \frac{\partial h}{\partial X_i} \left[\mu_{X_i} - \mu_{X_i}\right] = h(\mu). \end{aligned} \tag{1.8}$$

$$\begin{aligned} V\left[\tilde{h}(\mathbf{X})\right] &= V\left[h(\mu) + \sum_{i=1}^{n} \frac{\partial h}{\partial X_i}(X_i - \mu_{X_i})\right] = \\ &= V\left[\frac{\partial h}{\partial X_1}(X_1 - \mu_{X_1}) + \ldots + \frac{\partial h}{\partial X_n}(X_n - \mu_{X_n})\right] = \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n}\left(\frac{\partial h}{\partial X_i}\right)\left(\frac{\partial h}{\partial X_j}\right) cov(X_i, X_j). \end{aligned} \tag{1.9}$$

Substituindo as equações (1.8) e (1.9) em (1.7), temos a seguinte aproximação para o índice de confiabilidade

$$\beta \approx \frac{h(\mu)}{\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}\left(\frac{\partial h}{\partial X_i}\right)\left(\frac{\partial h}{\partial X_j}\right) cov(X_i, X_j)}}. \tag{1.10}$$

Tomando a expressão (1.10), obtemos a seguinte estimativa de primeira ordem para a probabilidade de falha

$$P_f \approx \Phi(-\beta).$$

Essa abordagem denomina-se FOSM (*First Order Second Moment*), pois considera no seu desenvolvimento apenas os momentos de até segunda ordem, ou seja, a média e o desvio padrão das variáveis aleatórias. De acordo com Haldar e Mahadevan [16] existem alguns inconvenientes com relação ao método FOSM: 1. a linearização da função desempenho nos valores médios das variáveis aleatórias acarreta significativos erros no cálculo da probabilidade de falha; 2. o índice de confiabilidade não é constante para diferentes formulações da função estado limite sob as mesmas condições de falha; 3. sua aplicação é limitada, pois a probabilidade de falha dada por (1.10) é exata somente quando as variáveis aleatórias são estatisticamente independentes e normalmente distribuídas ou quando $h(\mathbf{X})$ é uma função multiplicativa das variáveis aleatórias, que por sua vez, são variáveis independentes com distribuição lognormal.

Apesar de todas as limitações, o método FOSM é considerado fundamental na elaboração de outros métodos de confiabilidade estrutural, como por exemplo, o método proposto em 1974 por Hasofer e Lind [17], que descrevemos a seguir.

1.3 Extensão da solução do problema fundamental: função desempenho linear $n-$dimensional

Visando contornar os erros acarretados pelo método FOSM no cálculo da probabilidade de falha, Hasofer e Lind [17] propuseram uma nova abordagem aplicável a problemas cujas variáveis aleatórias são independentes e normalmente distribuídas, denominada AFOSM (*Advanced First Order Second Moment*).

O método AFOSM é baseado na transformação de Hasofer e Lind [17], dada por

$$Y_i = \frac{X_i - \mu_{X_i}}{\sigma_{X_i}}. \tag{1.11}$$

A equação (1.11) transforma variáveis aleatórias normais X_i em variáveis aleatórias normais padrão Y_i, ou seja, variáveis aleatórias com média zero e desvio padrão unitário. Sendo assim, essa transformação fornece a equação de estado limite, $h(\mathbf{Y}) = 0$, no espaço reduzido $\mathbb{Y}$.

Para facilitar a compreensão dos conceitos vamos considerar um exemplo bidimensional dado pela a equação de estado limite (1.1) do problema fundamental. Deste modo, aplicando-se a transformação (1.11) a equação de estado limite (1.1), obtemos a seguinte equação de estado limite no espaço das variáveis reduzidas Y_i

$$h(\mathbf{Y}) = \sigma_{X_1} Y_1 - \sigma_{X_2} Y_2 + \mu_{X_1} - \mu_{X_2} = 0. \tag{1.12}$$

Na Figura 1.1 ilustramos a transformação de Hasofer e Lind. A Figura 1.1(a) representa a equação de estado limite (1.1), no espaço original, juntamente com curvas de nível da função densidade de probabilidade normal bivariada, definida por

$$f_{\mathbf{X}}(\mathbf{x}) = \frac{1}{2\pi\sigma_{X_1}\sigma_{X_2}} e^{\left\{-\frac{1}{2}\left[\left(\frac{x_1-\mu_{X_1}}{\sigma_{X_1}}\right)^2 + \left(\frac{x_2-\mu_{X_2}}{\sigma_{X_2}}\right)^2\right]\right\}},$$

enquanto que a Figura 1.1(b) ilustra a equação de estado limite (1.12), no espaço reduzido, juntamente com curvas de nível da função densidade de probabilidade normal padrão bivariada, definida por

$$f_{\mathbf{Y}}(\mathbf{y}) = \frac{1}{2\pi} e^{-\frac{1}{2}\left(y_1^2 + y_2^2\right)}. \tag{1.13}$$

A posição da equação de estado limite no sistema de coordenadas do espaço reduzido, Figura 1.1(b), define uma medida de confiabilidade do sistema, pois quanto mais próximo da origem estiver a equação de estado limite, maior a região de falha.

Hasofer e Lind [17] generalizaram esse resultado estabelecendo a seguinte definição:

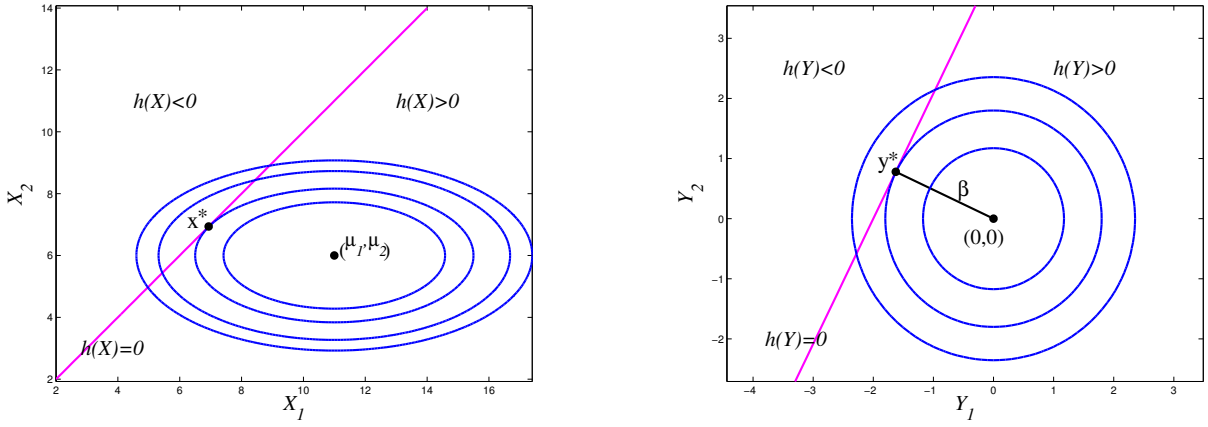
Definição 1.1. *O índice de confiabilidade, denotado por β, corresponde à mínima distância entre a equação de estado limite e a origem do espaço normal padrão.*

Nestes termos

$$\beta = \|\mathbf{y}^*\|,$$

em que $\mathbf{y}^*$ é o ponto sobre a equação de estado limite, $h(\mathbf{Y}) = 0$, mais próximo da origem e denominado de ponto de projeto[1]. Devido ao aspecto geométrico da função densidade

[1] O ponto de projeto é convencionalmente indicado na literatura por um asterisco (*).



(a) Espaço original

(b) Espaço reduzido

Figura 1.1: Transformação das variáveis X_1 e X_2 em variáveis normais padrão.

de probabilidade normal padrão bivariada, dada pela equação (1.13), a medida em que o ponto $\mathbf{y}$ no sistema de coordenadas normal padrão se afasta da origem, o valor da função (1.13) diminui. Sendo assim, na Figura 1.1(b), nota-se que o ponto de projeto é o ponto de interseção da equação de estado limite e a curva de nível da função de probabilidade normal padrão bivariada, com o maior conteúdo de probabilidade. Portanto, o ponto de projeto $\mathbf{y}^*$ é o ponto mais provável de falha, ou seja, é o ponto sobre a região de falha com maior probabilidade de ocorrência.

Segundo Beck [1], como o cálculo da probabilidade de falha depende de β, pois $P_f = \Phi(-\beta)$, o índice de confiabilidade é uma medida geométrica da probabilidade de falha.

Com base na definição (1.1) é possível determinar o índice de confiabilidade utilizando conceitos geométricos simples, ou seja, por meio do cálculo da distância euclidiana entre reta, dada por (1.12), e ponto (origem do sistema de coordenadas reduzidas). Obtemos assim

$$\beta_{HL} = \frac{\mu_{X_1} - \mu_{X_2}}{\sqrt{\sigma_{X_1}^2 + \sigma_{X_2}^2}}. \tag{1.14}$$

Observe que as expressões (1.5) e (1.14) são iguais. Isso indica que os métodos FOSM e AFOSM são equivalentes quando a equação de estado limite é linear e as variáveis são Gaussianas e estatisticamente independentes.

Porém, em geral as equações de estado limite são não lineares, nessas situações, o índice de confiabilidade pode ser obtido por meio da resolução de um problema de otimização restrito, conforme apresentamos na Seção 1.4, ou ainda, por meio da linearização da equação de estado limite no ponto de projeto, conforme descrevemos na Seção 1.5.

1.4 Função desempenho não linear: obtenção do ponto de projeto via resolução do problema de otimização

Conforme mencionado na Seção 1.3, o ponto de projeto corresponde ao ponto sobre a superfície de falha mais próximo da origem no espaço normal padrão. Sendo assim, podemos determiná-lo a partir da resolução do problema de otimização restrito (P) e que, por conveniência, apresentamos novamente a seguir

$$\begin{array}{ll} \text{minimizar} & \dfrac{1}{2}\mathbf{y}^T\mathbf{y} \\ \text{sujeito a} & h(\mathbf{y}) = 0 \end{array} \tag{1.15}$$

em que $h : \mathbb{R}^n \to \mathbb{R}$ é a superfície de falha com $h \in C^1$.

No caso em que a equação de estado limite é não linear uma aproximação para a probabilidade de falha sempre é possível, pois de acordo com o Lema 1.2, desenvolvido a partir das ideias de Periçaro [36], a existência de solução para o problema (1.15) está garantida.

Lema 1.2. *O problema de otimização restrito* (1.15) *admite um minimizador global.*

Demonstração. Considere o conjunto $\Omega = \{\mathbf{y} \in \mathbb{R}^n | h(\mathbf{y}) = 0\}$ e $\alpha = \inf\{\|\mathbf{y}\| \,|\mathbf{y} \in \Omega\}$ (o qual existe pelo fato de que $\|\mathbf{y}\| \geq 0$ para todo $\mathbf{y} \in \Omega$). Então, para todo $k \in \mathbf{N}$ existe $\mathbf{y}^k \in \Omega$ tal que

$$\alpha \leq \left\|\mathbf{y}^k\right\| \leq \alpha + \frac{1}{k}. \tag{1.16}$$

Em particular, $\left\|\mathbf{y}^k\right\| \leq \alpha + 1$, para todo $k \in \mathbb{N}$. Logo, existe uma subsequência convergente, $\mathbf{y}^k \xrightarrow{\mathbb{N}'} \mathbf{y}^*$. Como a função h é contínua, o conjunto $\mathbb{S}$ é fechado. Portanto, $\mathbf{y}^* \in \Omega$. Além disso,

$$\left\|\mathbf{y}^k\right\| \xrightarrow{\mathbb{N}'} \|\mathbf{y}^*\|.$$

Por outro lado, de (1.16), temos que $\left\|\mathbf{y}^k\right\| \xrightarrow{\mathbb{N}'} \alpha$, donde segue que $\|\mathbf{y}^*\| = \alpha$. Dessa forma, temos que $\|\mathbf{y}^*\| \leq \|\mathbf{y}\|$ para todo $\mathbf{y} \in \Omega$, completando a prova. □

A seguir apresentamos alguns resultados importantes para o desenvolvimento do trabalho.

Primeiramente, considere Ω o conjunto viável para o problema (1.15), ou seja

$$\Omega = \{\mathbf{y} \in \mathbb{R}^n | h(\mathbf{y}) = 0\}.$$

No Teorema 1.4, que enunciamos a seguir, usamos a suposição de que o problema (1.15) satisfaz a condição de qualificação LICQ (*Linear independence constraint qualification*), definida da seguinte forma:

Definição 1.3. *Um ponto $\bar{\mathbf{y}} \in \Omega$ é dito ser um ponto regular se o conjunto formado pelos gradientes de h em $\bar{\mathbf{y}}$ é linearmente independente. Essa condição é denominada de condição de qualificação LICQ.*

No caso do problema (1.15) em que $h : \mathbb{R}^n \to \mathbb{R}$, ou seja, o problema possui uma única restrição, basta que $\nabla h(\bar{\mathbf{y}}) \neq 0$ para que $\bar{\mathbf{y}}$ seja um ponto regular.

Teorema 1.4 (**Condições de otimalidade de Lagrange**). *Suponha que $f : \mathbb{R}^n \to \mathbb{R}$ seja diferenciável no ponto $\mathbf{y}^* \in \mathbb{R}^n$ e que $h : \mathbb{R}^n \to \mathbb{R}$ seja continuamente diferenciável em uma vizinhança deste ponto. Se $\mathbf{y}^*$ é um minimizador local do problema* (1.15) *e a condição de qualificação LICQ* (*dada na Definição 1.3*) *é satisfeita, então existe um único $\lambda^* \in \mathbb{R}$ tal que*

$$\nabla \ell_{\mathbf{y}}(\mathbf{y}^*, \lambda^*) = \nabla f(\mathbf{y}^*) + \lambda^* \nabla h(\mathbf{y}^*) = 0. \tag{1.17}$$

Demonstração. Izmailov e Solodov [22, Teorema 2.2.1]. □

Definição 1.5. *Dizemos que $\mathbf{y}^* \in \mathbb{R}^n$ é um ponto estacionário para o problema* (1.15)*, quando $\mathbf{y}^* \in \Omega$ e existe $\lambda^* \in \mathbb{R}$ tal que a condição* (1.17) *é satisfeita. Além disso, λ^* é chamado multiplicador de Lagrange associado ao ponto estacionário $\mathbf{y}^*$.*

O sistema de equações formado por

$$\nabla \ell_{\mathbf{y}}(\mathbf{y}^*, \lambda^*) = 0 \qquad \text{e} \qquad h(\mathbf{y}^*) = 0$$

nas variáveis $(\mathbf{y}^*, \lambda^*)$ chama-se *sistema de Karush-Kuhn-Tucker*, ou ainda, *condições de KKT* e caracteriza os pontos estacionários do problema (1.15) e o multiplicador de Lagrange associado.

Com base no resultado apresentado no Lema 1.2, considere $\mathbf{y}^*$ um ponto estacionário para (1.15) e suponha que seja válida a condição de qualificação LICQ nesse ponto, ou seja $\nabla h(\mathbf{y}^*) \neq 0$. De acordo com o Teorema 1.4, existe $\lambda^* \in \mathbb{R}$ tal que as condições de KKT são satisfeitas, ou seja

$$\mathbf{y}^* = -\lambda^* \nabla h(\mathbf{y}^*) \tag{1.18}$$

e

$$h(\mathbf{y}^*) = 0.$$

O multiplicador de Lagrange λ^* é obtido multiplicando a equação (1.18) por $\nabla h(\mathbf{y}^*)^T$, assim

$$\lambda^* = -\frac{\nabla h(\mathbf{y}^*)^T \mathbf{y}^*}{\|\nabla h(\mathbf{y}^*)\|^2}. \tag{1.19}$$

A partir das equações (1.18) e (1.19), obtemos

$$\|\mathbf{y}^*\| = \frac{\left|\nabla h(\mathbf{y}^*)^T \mathbf{y}^*\right|}{\|\nabla h(\mathbf{y}^*)\|}.$$

Observando a Figura 1.1(b), percebemos que a origem do sistema de coordenadas reduzidas, obtida pela padronização dos valores médios das variáveis aleatórias originais, pertence à região de segurança, ou seja, $h(\mathbf{y}) > 0$. Como $\nabla h(\mathbf{y}^*)$ indica a direção de máximo crescimento da função desempenho no ponto $\mathbf{y}^*$, os vetores $\nabla h(\mathbf{y}^*)$ e $\mathbf{y}^*$ possuem sinais opostos, logo $\nabla h(\mathbf{y}^*)^T \mathbf{y}^* < 0$. Portanto, concluímos que a mínima distância da origem do sistema de coordenadas reduzidas à equação de estado limite é

$$\|\mathbf{y}^*\| = -\frac{\nabla h(\mathbf{y}^*)^T \mathbf{y}^*}{\|\nabla h(\mathbf{y}^*)\|}. \tag{1.20}$$

Portanto, podemos afirmar que a equação (1.20) fornece o valor para o índice de confiabilidade β.

1.5 Função desempenho não linear: solução via linearização da equação de estado limite

Na seção 1.2 mostramos que o método FOSM apresenta alguns inconvenientes no cálculo da probabilidade de falha quando a linearização de $h(\mathbf{X})$ é realizada nos valores médios das variáveis aleatórias originais. Dessa forma, vamos mostrar nessa seção que se a linearização for centrada no ponto de projeto $\mathbf{x}^*$, o índice de confiabilidade equivale ao índice de Hasofer e Lind, dado por (1.14).

Considere a expansão em série de Taylor de primeira ordem no ponto de projeto $\mathbf{x}^*$, ou seja,

$$\tilde{h}(\mathbf{X}) = h(\mathbf{x}^*) + \nabla h(\mathbf{x}^*)^T (\mathbf{X} - \mathbf{x}^*).$$

Como $\mathbf{x}^*$ está localizado na superfície de falha, temos que

$$\tilde{h}(\mathbf{X}) = \nabla h(\mathbf{x}^*)^T (\mathbf{X} - \mathbf{x}^*) = \sum_{i=1}^{n} (X_i - x_i^*) \left.\frac{\partial h}{\partial X_i}\right|_{\mathbf{X}=\mathbf{x}^*}. \tag{1.21}$$

Usando a transformação de Hasofer e Lind, dada por (1.11), temos as componentes do vetor $\mathbf{x}^*$ no espaço original

$$x_i^* = \sigma_{X_i} y_i^* + \mu_{X_i}. \tag{1.22}$$

Das equações (1.11) e (1.22), temos que

$$X_i - x_i^* = \sigma_{X_i} (Y_i - y_i^*) \tag{1.23}$$

e

$$\left.\frac{\partial h}{\partial X_i}\right|_{\mathbf{X}=\mathbf{x}^*} = \frac{1}{\sigma_{X_i}} \cdot \left.\frac{\partial h}{\partial Y_i}\right|_{\mathbf{Y}=\mathbf{y}^*}. \tag{1.24}$$

Portanto, substituindo as equações (1.23) e (1.24) em (1.21), obtemos

$$\tilde{h}(\mathbf{X}) = \sum_{i=1}^{n}(Y_i - y_i^*)\left.\frac{\partial h}{\partial Y_i}\right|_{\mathbf{Y}=\mathbf{y}^*} = \nabla h(\mathbf{y}^*)^T(\mathbf{Y} - \mathbf{y}^*) = \tilde{h}(\mathbf{Y}). \tag{1.25}$$

Lembrando que as variáveis aleatórias Y_i pertencem ao espaço normal padrão e são estatisticamente independentes, sendo assim, as aproximações de primeira ordem para a média e variância da equação de estado limite são dadas, respectivamente, por

$$E\left[\tilde{h}(\mathbf{X})\right] = E\left[\nabla h(\mathbf{y}^*)^T(\mathbf{Y} - \mathbf{y}^*)\right] = -\nabla h(\mathbf{y}^*)^T\mathbf{y}^* \tag{1.26}$$

e

$$V\left[\tilde{h}(\mathbf{X})\right] = V\left[\nabla h(\mathbf{y}^*)^T(\mathbf{Y} - \mathbf{y}^*)\right] = \left\|\nabla h(\mathbf{y}^*)\right\|^2. \tag{1.27}$$

Substituindo as equações (1.26) e (1.27) em (1.7), segue que

$$\beta = -\frac{\nabla h(\mathbf{y}^*)^T\mathbf{y}^*}{\|\nabla h(\mathbf{y}^*)\|}. \tag{1.28}$$

A equação (1.28) é idêntica à equação (1.20), que corresponde a mínima distância entre a equação de estado limite à origem do sistema de coordenadas reduzidas.

Conforme mencionado, o ponto de projeto é o ponto sobre o domínio de falha com maior probabilidade de ocorrência o que, de acordo com Beck [1], significa que o maior conteúdo de probabilidade no domínio de falha está na vizinhança desse ponto. Dessa forma, erros na aproximação de primeira ordem para a probabilidade de falha podem ser minimizados linearizando a equação de estado limite sobre o ponto de projeto, uma vez que a precisão da aproximação de primeira ordem depende do grau de não linearidade de $h(\mathbf{y})$ no ponto $\mathbf{y}^*$.

Os métodos FOSM e AFOSM fornecem a solução exata da probabilidade de falha apenas nos casos em que a equação de estado limite é linear e as variáveis aleatórias são Gaussianas e independentes. Essas hipóteses são extremamente limitantes para aplicação em situações práticas. No entanto, tais métodos constituem a base para a construção de dois métodos mais robustos, FORM (*First Order Reliability Method*) e SORM (*Second Order Reliability Method*), aplicados na resolução de problemas cujas variáveis aleatórias sejam correlacionadas e/ou seguem uma distribuição aleatória qualquer.

O método FORM, assim como o método AFOSM, consiste em substituir a superfície de falha por sua linearização no ponto de projeto, considerando todas as informações estatísticas das variáveis de projeto. Já no método SORM, a superfície de falha é aproximada por uma função quadrática levando em consideração o aspecto da curvatura da

superfície de falha nas proximidades do ponto de projeto. Neste trabalho limitamo-nos ao estudo do método FORM.

A solução dos problemas de confiabilidade via método FORM, tanto para equações de estado limite lineares quanto para não lineares, envolve, para a obtenção do ponto de projeto, a resolução de um problema de otimização, dado por (1.15), podendo ser aplicado qualquer algoritmo de otimização.

Um algoritmo conhecido na literatura é o algoritmo HLRF, desenvolvido em 1974 por Hasofer e Lind [17] e aperfeiçoado em 1978 por Rackwitz e Fiessler [41] . Este algoritmo é popular devido a sua simplicidade, embora não seja necessariamente eficiente para problemas fortemente não lineares. No Capítulo 2 apresentamos as versões clássica e melhorada do algoritmo HLRF, bem como um dos algoritmos que estamos propondo para o cálculo da probabilidade de falha, denominado nHLRF, cuja prova de convergência também é apresentada.

As informações estatísticas utilizadas pelos métodos abordados até aqui se limitam ao primeiro e segundo momento das variáveis aleatórias, isso equivale a considerar todas as variáveis do problema com distribuição Gaussiana. Porém, o método FORM, que será apresentado mais adiante, permite lidar com distribuições de probabilidade não Gaussianas e, também, possibilita o tratamento de variáveis aleatórias correlacionadas, exigindo uma transformação adicional à de Hasofer e Lind, dada por (1.11). Para o estudo dos métodos de transformação tomamos principalmente como referência os trabalhos [1], [13], [16] e [33].

1.6 Métodos de transformação

Os métodos de confiabilidade foram desenvolvidos com o objetivo de calcular o índice de confiabilidade de uma estrutura com base nas variáveis de projeto. Em consequência da Definição 1.1, o índice de confiabilidade será corretamente obtido se as variáveis de projeto forem mapeadas para o espaço normal padrão e forem não correlacionadas.

Sendo assim, primeiramente apresentamos na Seção 1.6.1 a situação mais simples, ou seja, quando as variáveis de projeto são Gaussianas e estatisticamente independentes. Já na Seção 1.6.2, apresentamos a transformação composta utilizando o modelo de Nataf, que consiste em transformar variáveis com uma distribuição qualquer e possivelmente correlacionadas em variáveis normais equivalentes e independentes.

1.6.1 Transformação de Hasofer e Lind na forma matricial

Em geral, a equação de estado limite é formulada e avaliada no espaço original, no entanto, a busca pelo ponto de projeto é feita no espaço normal padrão. Sendo assim, a maior parte do trabalho para a resolução dos problemas de confiabilidade, quando as

variáveis de projeto $\mathbf{X}$ são independentes e normalmente distribuídas, é a transformação de $\mathbf{X}$ para o espaço reduzido. Neste caso, basta padronizarmos as variáveis de projeto por meio da transformação de Hasofer e Lind, dada pela equação (1.11), conforme apresentado na Seção 1.3.

Contudo, para problemas envolvendo um grande número de variáveis aleatórias é conveniente trabalhar utilizando operações matriciais. Assim, denotando o vetor de médias por $\mathbf{M}$ e a matriz diagonal de desvios padrão por $\mathbf{D}$, ou seja,

$$\mathbf{M} = (\mu_{X_1}, \ldots, \mu_{X_n})^T \qquad \text{e} \qquad \mathbf{D} = \operatorname{diag}(\sigma_{X_1}, \sigma_{X_2}, \ldots, \sigma_{X_n}),$$

a transformação de Hasofer e Lind pode ser escrita matricialmente da seguinte forma

$$\mathbf{y} = \mathbf{D}^{-1} \cdot \{\mathbf{x} - \mathbf{M}\} \tag{1.29}$$

e

$$\mathbf{x} = \mathbf{D} \cdot \mathbf{y} + \mathbf{M}, \tag{1.30}$$

onde $\mathbf{D}^{-1}$ é a matriz inversa de $\mathbf{D}$.

Para facilitar estas transformações, uma forma alternativa é trabalhar com matrizes jacobianas. Desse modo, considerando a tranformação de Hasofer e Lind, dada por (1.11), para variáveis aleatórias não-correlacionadas, temos

$$\mathbf{J_{yx}} = \frac{\partial y_i}{\partial x_j} = \begin{cases} \frac{1}{\sigma_{X_i}}, & \text{para } \; i = j \\ 0, & \text{para } \; i \neq j \end{cases}$$

$$\mathbf{J_{xy}} = \frac{\partial x_i}{\partial y_j} = \begin{cases} \sigma_{X_i}, & \text{para } \; i = j \\ 0, & \text{para } \; i \neq j \end{cases}$$

com $i = 1, \ldots, n$ e $j = 1, \ldots, n$.

Observe que existe uma relação entre as matrizes $\mathbf{D}$ e $\mathbf{J_{xy}}$ e entre $\mathbf{D}^{-1}$ e $\mathbf{J_{yx}}$. Logo, as equações (1.29) e (1.30) podem ser reescritas, respectivamente, como

$$\mathbf{y} = \mathbf{J_{yx}} \{\mathbf{x} - \mathbf{M}\} \tag{1.31}$$

e

$$\mathbf{x} = \mathbf{J_{xy}} \cdot \mathbf{y} + \mathbf{M}. \tag{1.32}$$

Como as equações de estado limite são escritas no espaço das variáveis originais, as derivadas também podem ser avaliadas nesse mesmo espaço. Assim, o vetor $\nabla h(\mathbf{y})$ é

obtido da seguinte forma

$$\nabla h(\mathbf{y}) = \left\{ \frac{\partial h}{\partial y_i} \right\}_{i=1,\ldots,n} = \left\{ \sum_{j=1}^{n} \frac{\partial h}{\partial x_i} \frac{\partial x_i}{\partial y_j} \right\}_{i=1,\ldots,n} = (\mathbf{J_{xy}})^T \nabla h(\mathbf{x}).$$

Portanto, as equações (1.31) e (1.32) constituem, respectivamente, as transformações do espaço de projeto para o espaço reduzido, $\mathbb{X} \rightarrow \mathbb{Y}$, e do espaço reduzido para o espaço de projeto, $\mathbb{Y} \rightarrow \mathbb{X}$.

1.6.2 Transformação composta

A Transformação Composta recebe tal denominação por consistir de três etapas: 1. a transformação das variáveis originais em variáveis normais equivalentes; 2. a determinação dos coeficientes de correlação equivalentes, utilizando para isto o modelo de Nataf; 3. a eliminação da correlação entre as variáveis aleatórias.

Essa transformação revela-se importante no contexto do método FORM, uma vez que, tal método permite incorporar na determinação da probabilidade de falha informações sobre funções de distribuição de probabilidade das variáveis originais, bem como, a correlação entre as mesmas, envolvendo a construção de uma função conjunta de distribuição de probabilidade $f_{\mathbf{X}}(\mathbf{x})$ e a transformação desta para o espaço normal padrão $f_{\mathbf{Y}}(\mathbf{y})$.

Para obtenção da função conjunta de distribuição de probabilidade, $f_{\mathbf{X}}(\mathbf{x})$, é necessário registros simultâneos de todas as variáveis envolvidas no problema. Contudo, as informações estatísticas a respeito das variáveis aleatórias do problema limitam-se às funções de distribuição de probabilidade marginais

$$f_{X_i}(x_i), \quad i = 1, \ldots, n,$$

e aos coeficientes de correlação entre pares de variáveis aleatórias, dados pela matriz de correlação

$$\mathbf{R_X} = \begin{bmatrix} 1 & \rho_{X_{12}} & \cdots & \rho_{X_{1n}} \\ \rho_{X_{21}} & 1 & \cdots & \rho_{X_{2n}} \\ \vdots & \vdots & \cdots & \vdots \\ \rho_{X_{n1}} & \rho_{X_{n2}} & \cdots & 1 \end{bmatrix}$$

sendo n é o número de variáveis aleatórias do problema.

Segundo Beck [1], a transformação de $f_{\mathbf{X}}(\mathbf{x})$ para $f_{\mathbf{Y}}(\mathbf{y})$ representa um mapeamento um-a-um que leva pontos do espaço original para o espaço reduzido. Teoricamente, esse mapeamento pode ser realizado por meio da Transformação de Rosenblatt [46], sendo inapropriada na prática, por envolver integrações multi-dimensionais e distribuições condicionais que dificilmente são conhecidas.

A seguir apresentamos os procedimentos utilizados na primeira etapa da transformação composta, ou seja, a transformação das distribuições marginais originais em distribuições normais equivalentes.

O princípio da aproximação normal

O princípio da aproximação normal também denominado "princípio da cauda normal", proposto por Ditlevsen [13], consiste em aproximar a cauda da distribuição original pela cauda de uma distribuição normal equivalente e, dessa forma, determinar para um ponto x_i^* uma distribuição normal equivalente que preserve o conteúdo de probabilidade da distribuição original neste ponto, $F_{X_i}(x_i^*)$.

Os parâmetros da distribuição normal equivalente, média $\mu_{X_i}^{neq}$ e desvio padrão $\sigma_{X_i}^{neq}$, são obtidos impondo duas condições,

$$F_{X_i}^{neq}(x_i^*) = F_{X_i}(x_i^*) \tag{1.33}$$

e

$$f_{X_i}^{neq}(x_i^*) = f_{X_i}(x_i^*). \tag{1.34}$$

Utilizando a transformação de Hasofer e Lind (1.11), é possível obter um conjunto de variáveis, possivelmente correlacionadas, com distribuição normal padrão, denotadas por $\mathbf{Z} = (Z_1, \ldots, Z_n)$, onde

$$z_i^* = \frac{x_i^* - \mu_{X_i}^{neq}}{\sigma_{X_i}^{neq}}. \tag{1.35}$$

Reescrevendo as equações (1.33) e (1.34) em termos de z_i^*, obtemos

$$F_{X_i}^{neq}(x_i^*) = \Phi\left(\frac{x_i^* - \mu_{X_i}^{neq}}{\sigma_{X_i}^{neq}}\right) = \Phi(z_i^*) \tag{1.36}$$

e

$$f_{X_i}^{neq}(x_i^*) = \frac{1}{\sigma_{X_i}^{neq}\sqrt{2\pi}} e^{\left\{-\frac{1}{2}\left(\frac{x_i^* - \mu_{X_i}^{neq}}{\sigma_{X_i}^{neq}}\right)^2\right\}} = \frac{\phi(z_i^*)}{\sigma_{X_i}^{neq}}, \tag{1.37}$$

em que $\Phi(z_i^*)$ e $\phi(z_i^*)$ são, respectivamente, função de distribuição acumulada normal padrão e função densidade de probabilidade normal padrão da variável z_i^*.

A partir da equação (1.37) obtemos uma expressão para o desvio padrão da distribuição normal equivalente, dada por

$$\sigma_{X_i}^{neq} = \frac{\phi(z_i^*)}{f_{X_i}(x_i^*)}, \tag{1.38}$$

sendo z_i^* obtido a partir da equação (1.36),

$$z_i^* = \Phi^{-1}\left(F_{X_i}^{neq}(x_i^*)\right).$$

Utilizando a equação (1.35) obtemos uma expressão para a média da distribuição normal equivalente

$$\mu_{X_i}^{neq} = x_i^* - z_i^* \sigma_{X_i}^{neq}. \tag{1.39}$$

A transformação é realizada para cada uma das distribuições marginais e é válida somente no ponto $\mathbf{x}^*$. Deste modo, a medida que o algoritmo calcula um novo iterando na busca do ponto de projeto, a transformação deve ser refeita.

Assim como a transformação de Hasofer e Lind, a transformação das variáveis de projeto $\mathbf{X}$ para variáveis normais padrão $\mathbf{Z}$, também pode ser escrita matricialmente a partir de um vetor de médias $\mathbf{M}^{neq}$ e de uma matriz diagonal de desvios padrão $\mathbf{D}^{neq}$, contendo os parâmetros das distribuições normais equivalentes

$$\mathbf{M}^{neq} = \left(\mu_{X_1}^{neq}, \ldots, \mu_{X_n}^{neq}\right)^T \qquad \text{e} \qquad \mathbf{D}^{neq} = \operatorname{diag}\left(\sigma_{X_1}^{neq}, \sigma_{X_2}^{neq}, \ldots \sigma_{X_n}^{neq}\right). \tag{1.40}$$

A inversa da matriz diagonal de desvios padrão é denotada por $(\mathbf{D}^{neq})^{-1}$.

Utilizando a equação (1.35), introduzimos as matrizes jacobianas

$$\mathbf{J_{zx}} = \frac{\partial z_i}{\partial x_j} = \begin{cases} \frac{1}{\sigma_{X_i}^{neq}}, & \text{para } \; i = j \\ 0, & \text{para } \; i \neq j \end{cases}$$

$$\mathbf{J_{xz}} = \frac{\partial x_i}{\partial z_j} = \begin{cases} \sigma_{X_i}^{neq}, & \text{para } \; i = j \\ 0, & \text{para } \; i \neq j \end{cases}$$

com $i = 1, \ldots, n$ e $j = 1, \ldots, n$.

Assim, como mencionado na seção 1.6.1, existe uma relação entre as matrizes $\mathbf{D}^{neq}$ e $\mathbf{J_{xz}}$ e entre $(\mathbf{D}^{neq})^{-1}$ e $\mathbf{J_{zx}}$.

Logo as respectivas transformações $\mathbb{X} \to \mathbb{Z}$ e $\mathbb{Z} \to \mathbb{X}$, são dadas matricialmente por

$$\mathbf{z} = \mathbf{J_{zx}}\left\{\mathbf{x} - \mathbf{M}^{neq}\right\} \tag{1.41}$$

e

$$\mathbf{x} = \mathbf{J_{xz}} \cdot \mathbf{z} + \mathbf{M}^{neq}. \tag{1.42}$$

É importante destacar que, caso exista correlação entre as variáveis originais $\mathbf{X}$, a transformação dada pela equação (1.41), não elimina a correlação entre as variáveis normais padrão $\mathbf{Z}$. Deste modo, a correlação entre pares de variáveis originais deve ser imposta na distribuição conjunta normal padrão $f_{\mathbf{Z}}(\mathbf{z})$, isto é, $f_{\mathbf{Z}}(\mathbf{z}) = \phi_n(\mathbf{z}, \rho_{\mathbf{z}})$ onde $\rho_{\mathbf{Z}}$ é a matriz de correlação a ser determinada. Isto pode ser feito por meio da transformação

de Nataf, que refere-se a segunda etapa da transformação composta.

Transformação de Nataf

A transformação proposta por Nataf [34], consiste em construir uma aproximação para a função conjunta de densidade de probabilidade $f_{\mathbf{X}}(\mathbf{x})$ a partir da função de densidade normal padrão n-dimensional com matriz de correlação $\rho_{\mathbf{Z}}$

$$f_{\mathbf{X}}(\mathbf{x}) = \phi_n(\mathbf{z}, \rho_{\mathbf{z}}) \frac{f_{X_1}(x_1) f_{X_2}(x_2) \dots f_{X_n}(x_n)}{\phi_{Z_1}(z_1)\phi_{Z_2}(z_2) \dots \phi_{Z_n}(z_n)}, \tag{1.43}$$

em que $f_{X_i}(x_i)$ e $\phi_{Z_i}(z_i)$ são, respectivamente, a distribuição marginal de X_i e a função densidade de probabilidade normal padrão, com $i = 1, \dots, n$. Além disso, $\rho_{\mathbf{Z}}$ é a matriz de correlação das variáveis normais padrão $\mathbf{Z}$ e $\phi_n(\mathbf{z}, \rho_{\mathbf{z}})$ é a função densidade de probabilidade normal padrão n-dimensional com variáveis correlacionadas, dada por

$$\phi_n(\mathbf{z}, \rho_{\mathbf{z}}) = \frac{1}{\sqrt{(2\pi)^n \left|\rho_{\mathbf{z}}\right|}} e^{-\frac{1}{2}\mathbf{z}^T \rho_{\mathbf{z}}^{-1} \mathbf{z}},$$

em que $|\rho_{\mathbf{z}}|$ e $\rho_{\mathbf{z}}^{-1}$ são, respectivamente, o determinante e a inversa da matriz $\rho_{\mathbf{z}}$. Observe que $\rho_{\mathbf{z}}^{-1}$ deve existir para que $\phi_n(\mathbf{z}, \rho_{\mathbf{z}})$ esteja adequadamente definida.

Se o coeficiente de correlação entre X_i e X_j, denotado por $\rho_{X_{ij}}$, impõe uma certa tendência na distribuição conjunta $f_{X_i X_j}(x_i, x_j)$, é necessário encontrar um coeficiente de correlação $\rho_{Z_{ij}}$ que imponha a mesma tendência na distribuição conjunta $f_{Z_i Z_j}(z_i, z_j)$. Para isto, considere duas variáveis aleatórias (X_i, X_j) não-normais com coeficiente de correlação $\rho_{X_{ij}}$, dado por

$$\rho_{X_{ij}} = \frac{\operatorname{cov}(X_i, X_j)}{\sigma_{X_i}\sigma_{X_j}} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \frac{(x_i - \mu_{X_i})}{\sigma_{X_i}} \frac{\left(x_j - \mu_{X_j}\right)}{\sigma_{X_j}} f_{X_i X_j}(x_i, x_j) dx_i dx_j. \tag{1.44}$$

De acordo com Melchers [33] utilizando a equação (1.43), para $n = 2$, a equação (1.44) torna-se

$$\rho_{X_{ij}} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} z_i z_j \phi_2(z_i, z_j, \rho_{Z_{ij}}) dz_i dz_j. \tag{1.45}$$

Esta última expressão (1.45) permite calcular iterativamente a matriz de correlação $\rho_{Z_{ij}}$ a partir de $\rho_{X_{ij}}$, porém este cálculo pode ser bastante trabalhoso, pois o termo desconhecido está dentro da integral dupla. Para contornar essa dificuldade, Liu e Kiureghian [25] desenvolveram um conjunto de fórmulas analíticas que fornecem uma relação $\mathbf{R}_{ij}$ entre $\rho_{Z_{ij}}$ e $\rho_{X_{ij}}$ para várias combinações de distribuições, dada por

$$\mathbf{R}_{ij} = \frac{\rho_{Z_{ij}}}{\rho_{X_{ij}}}.$$

Valores para $\mathbf{R}_{ij}$ podem ser obtidos na literatura, como por exemplo em Melchers

[33], e variam de acordo com a distribuição de probabilidade das variáveis de projeto. De acordo com Beck [1], como na prática o coeficiente de correlação entre duas variáveis $\rho_{X_{ij}}$ é obtido de forma subjetiva é possível aproximar $\rho_{Z_{ij}}$ por $\rho_{X_{ij}}$, embora exista diferença teórica entre $\rho_{Z_{ij}}$ e $\rho_{X_{ij}}$.

Conforme mencionado, o princípio da aproximação normal e a transformação de Nataf permitem a obtenção de um conjunto de variáveis aleatórias normais padrão $\mathbf{Z}$ podendo ser correlacionadas, ou seja, com distribuição normal padrão n-dimensional $f_{\mathbf{Z}}(\mathbf{z}) = \phi_n(\mathbf{z}, \rho_{\mathbf{z}})$. Para aproveitar as propriedades de simetria da distribuição normal padrão n-dimensional é necessário eliminar a correlação entre as variáveis $\mathbf{Z}$. Para tanto, primeiramente precisamos analisar a relação entre a matriz de correlação $\rho_{\mathbf{Z}}$ e a matriz de covariância $\mathbf{C_Z}$ das variáveis normais padrão $\mathbf{Z}$ que, por sua vez, será estabelecida no lema a seguir.

Lema 1.6. *A matriz de correlação das variáveis normais padrão $\rho_{\boldsymbol{Z}}$ é igual à sua matriz de covariância $\boldsymbol{C_Z}$.*

Demonstração. O coeficiente de correlação entre as variáveis normais padrão (Z_i, Z_j), denotado por $\rho_{Z_{ij}}$, é definido da seguinte forma

$$\rho_{Z_{ij}} = \frac{\operatorname{cov}(Z_i, Z_j)}{\sigma_{Z_i}\sigma_{Z_j}}, \tag{1.46}$$

em que $\operatorname{cov}(Z_i, Z_j)$ indica a covariância entre as variáveis Z_i e Z_j.

Como $\sigma_{Z_i} = \sigma_{Z_j} = 1$, a equação (1.46) torna-se

$$\rho_{Z_{ij}} = \operatorname{cov}(Z_i, Z_j).$$

Portanto, a matriz de correlação das variáveis reduzidas $\rho_{\mathbf{Z}}$ é igual a sua matriz de covariância $\mathbf{C_Z}$. □

A eliminação da correlação das variáveis normais padrão $\mathbf{Z}$, transformando-as em variáveis aleatórias normais padrão e independentes, $\mathbf{Y} = (Y_1, \ldots, Y_n)$, constitui o terceiro passo da transformação composta. Com base no resultado do Lema 1.6, isso pode ser feito por meio da decomposição ortogonal ou da decomposição de Cholesky da matriz de covariância $\mathbf{C_z}$/ou correlação $\rho_{\mathbf{Z}}$, conforme abordamos a seguir.

Transformação ortogonal da matriz de correlação

Considere a seguinte transformação linear

$$\mathbf{Y} = \mathbf{T}^T\mathbf{Z}, \tag{1.47}$$

em que $\mathbf{Z}$ é o vetor de variáveis normais padrão correlacionadas, $\mathbf{Y}$ é o vetor de variáveis normais padrão independentes e $\mathbf{T}$ é uma matriz de transformação a ser determinada pela decomposição ortogonal.

Seja $\mathbf{C_Y}$ a matriz de covariância das variáveis $\mathbf{Y}$, ou seja,

$$\mathbf{C_Y} = E\left[(\mathbf{Y} - \mu_\mathbf{Y})(\mathbf{Y} - \mu_\mathbf{Y})^T\right]. \tag{1.48}$$

Temos que,

$$\mu_\mathbf{Y} = E(\mathbf{Y}) = E\left(\mathbf{T}^T\mathbf{Z}\right) = \mathbf{T}^T E(\mathbf{Z}) = 0. \tag{1.49}$$

Das equações (1.48) e (1.49), obtemos

$$\mathbf{C_Y} = E\left[\mathbf{Y}\mathbf{Y}^T\right]. \tag{1.50}$$

Substituindo a equação (1.47) em (1.50), segue que

$$\mathbf{C_Y} = E\left[\left(\mathbf{T}^T\mathbf{Z}\right)\left(\mathbf{T}^T\mathbf{Z}\right)^T\right] = \mathbf{T}^T E\left[\mathbf{Z}\mathbf{Z}^T\right]\mathbf{T}. \tag{1.51}$$

Considere a matriz de covariância das variáveis $\mathbf{Z}$, dada por

$$\mathbf{C_Z} = E\left[(\mathbf{Z} - \mu_\mathbf{Z})(\mathbf{Z} - \mu_\mathbf{Z})^T\right]. \tag{1.52}$$

Como $\mu_\mathbf{Z} = 0$, então a equação (1.52) torna-se

$$\mathbf{C_Z} = E\left[\mathbf{Z}\mathbf{Z}^T\right]. \tag{1.53}$$

Substituindo a equação (1.53) na equação (1.51), temos

$$\mathbf{C_Y} = \mathbf{T}^T\mathbf{C_Z}\mathbf{T}. \tag{1.54}$$

A matriz de covariância $\mathbf{C_Z}$ é real e simétrica, de forma que, recorrendo aos conceitos de álgebra linear concluímos que $\mathbf{C_Z}$ é ortogonalmente diagonalizável, ou seja, existe uma matriz ortogonal $\mathbf{P}$, $\left(\mathbf{P}^{-1} = \mathbf{P}^T\right)$, e uma matriz diagonal $\mathbf{Q}$, tais que

$$\mathbf{P}^{-1}\mathbf{C_Z}\mathbf{P} = \mathbf{Q}. \tag{1.55}$$

A matriz diagonal $\mathbf{Q}$ é composta pelos autovalores da matriz de covariância $\mathbf{C_Z}$ e a matriz ortogonal $\mathbf{P}$ é composta pelos autovetores normalizados associados aos autovalores da matriz $\mathbf{C_Z}$. Substituindo a matriz de transformação $\mathbf{T}$, na equação (1.54), pela matriz $\mathbf{P}$, a matriz de covariância $\mathbf{C_Y}$ torna-se uma matriz diagonal, ou seja,

$$\mathbf{C_Y} = \mathbf{P}^T\mathbf{C_Z}\mathbf{P} = \mathbf{Q}. \tag{1.56}$$

Da equação (1.56), concluímos que as variáveis $\mathbf{Y}$ são independentes, pois $\text{cov}(Y_i, Y_j) = 0$, para todo $i \neq j$. Além disso, $\sigma^2_{Z_i} = \lambda_i$, onde λ_i com $i = 1, \ldots, n$, são os autovalores da matriz $\mathbf{C_Z}$. Portanto, as variâncias das variáveis $\mathbf{Y}$ resultantes desta transformação podem ser diferentes da unidade. No entanto, isso é inconveniente, pois exige uma transformação adicional como a de Hasofer e Lind para obter variáveis reduzidas com variância unitária. Para evitar essa transformação adicional é necessário determinar uma matriz de transformação $\mathbf{T}$, tal que $\mathbf{T}^T\mathbf{C_Z}\mathbf{T} = \mathbf{I}$, onde $\mathbf{I}$ é a matriz identidade. Para tanto, precisamos apresentar primeiramente alguns resultados importantes.

Teorema 1.7. *A matriz de covariância de um vetor aleatório $\boldsymbol{X} \in \mathbb{R}^n$, denotada por $\boldsymbol{C_X}$, é semi-definida positiva.*

Demonstração. Seja o vetor $a \in \mathbb{R}^n \setminus \{0\}$ e $\mathbf{C_X}$ a matriz de covariância do vetor aleatório $\mathbf{X} \in \mathbb{R}^n$, dada por

$$\mathbf{C_X} = E\left[(\mathbf{X} - \mu_\mathbf{X})(\mathbf{X} - \mu_\mathbf{X})^T\right]. \tag{1.57}$$

Tomando a equação (1.57), temos que

$$\begin{aligned}
a^T\mathbf{C_X}a &= a^T E\left[(\mathbf{X} - \mu_\mathbf{X})(\mathbf{X} - \mu_\mathbf{X})^T\right] a \\
&= E\left[a^T(\mathbf{X} - \mu_\mathbf{X})(\mathbf{X} - \mu_\mathbf{X})^T a\right] \\
&= E\left\{\left[(\mathbf{X} - \mu_\mathbf{X})^T a\right]^T\left[(\mathbf{X} - \mu_\mathbf{X})^T a\right]\right\} \\
&= E\left\{\left[(\mathbf{X} - \mu_\mathbf{X})^T a\right]^2\right\} \\
&= E\left\{\left[\mathbf{X}^T a - \mu_\mathbf{X}^T a\right]^2\right\} \\
&= V\left(\mathbf{X}^T a - \mu_\mathbf{X}^T a\right) + \left[E\left(\mathbf{X}^T a - \mu_\mathbf{X}^T a\right)\right]^2 \\
&= V\left(\mathbf{X}^T a\right) + V\left(\mu_\mathbf{X}^T a\right) + \left[E\left(\mathbf{X}^T a\right) - E\left(\mu_\mathbf{X}^T a\right)\right]^2.
\end{aligned} \tag{1.58}$$

Como $V\left(\mu_\mathbf{X}^T a\right) = 0$ e

$$E\left(\mathbf{X}^T a\right) = E\left(a_1X_1 + \ldots + a_nX_n\right) = a_1E\left(X_1\right) + \ldots + a_nE\left(X_n\right) = \left[E(\mathbf{X})\right]^T a = \mu_\mathbf{X}^T a,$$

a equação (1.58) torna-se

$$a^T\mathbf{C_X}a = V\left(\mathbf{X}^T a\right) + \left(\mu_\mathbf{X}^T a - \mu_\mathbf{X}^T a\right)^2 = V\left(\mathbf{X}^T a\right).$$

Como $V\left(\mathbf{X}^T a\right) \geq 0$, $\mathbf{C_X}$ é semi-definida positiva. □

Teorema 1.8. *A matriz de covariância $\boldsymbol{C_X}$ de um vetor aleatório $\boldsymbol{X} \in \mathbb{R}^n$ é definida positiva se, e somente se, é inversível.*

Demonstração. Seja a matriz de covariância $\mathbf{C_X}$ definida positiva. Então, todos os determinantes das submatrizes de $\mathbf{C_X}$ são positivos. Logo, $\mathbf{C_X}$ é inversível.

Seja $\mathbf{C_X}$ inversível. Então, todos os autovalores de $\mathbf{C_X}$ são não nulos. Como $\mathbf{C_X}$ é semi-definida positiva (Teorema 1.7) os seus autovalores são todos positivos. Portanto, a matriz $\mathbf{C_X}$ é definida positiva. □

De acordo com o Teorema 1.8, se a matriz de covariância $\mathbf{C_Z}$ for inversível, os elementos da matriz diagonal $\mathbf{Q}$ (autovalores da matriz $\mathbf{C_Z}$) serão todos positivos e a equação (1.55) pode ser escrita da seguinte forma:

$$\mathbf{Q}^{-\frac{1}{2}}\mathbf{P}^T\mathbf{C_Z}\mathbf{P}\mathbf{Q}^{-\frac{1}{2}} = \mathbf{I}.$$

Lembrando que a matriz $\mathbf{P}$ é ortogonal.

Dessa forma, para transformar as variáveis normais padrão correlacionadas $\mathbf{Z}$ em variáveis normais padrão independentes $\mathbf{Y}$, basta tomar a matriz de transformação na equação (1.47) como $\mathbf{T} = \mathbf{P}\mathbf{Q}^{-\frac{1}{2}}$.

As matrizes jacobianas para a transformação (1.47), ou seja, $\mathbf{Y} = \mathbf{T}^T\mathbf{Z}$, são

$$\mathbf{J_{yz}} = \mathbf{T}^T = \mathbf{Q}^{-\frac{1}{2}}\mathbf{P}^T \tag{1.59}$$

e

$$\mathbf{J_{zy}} = \left(\mathbf{T}^T\right)^{-1} = \left(\mathbf{Q}^{-\frac{1}{2}}\mathbf{P}^T\right)^{-1} = \mathbf{P}\mathbf{Q}^{\frac{1}{2}}. \tag{1.60}$$

Portanto, utilizando as equações (1.59) e (1.60) obtemos a expressão para as transformações do espaço $\mathbb{Z} \to \mathbb{Y}$ e do espaço $\mathbb{Y} \to \mathbb{Z}$, respectivamente

$$\mathbf{y} = \mathbf{J_{yz}} \cdot \mathbf{z} \tag{1.61}$$

e

$$\mathbf{z} = \mathbf{J_{zy}} \cdot \mathbf{y}. \tag{1.62}$$

O inconveniente da transformação ortogonal é o custo computacional da decomposição. Uma transformação que apresenta vantagens com relação a este inconveniente, principalmente quando as matrizes $\mathbf{C_Z}$ e $\rho_\mathbf{Z}$ são esparsas, é a transformação por decomposição de Cholesky.

Transformação por decomposição de Cholesky

Considere novamente a transformação dada por (1.47). Do mesmo modo que na transformação ortogonal, o objetivo da transformação de Cholesky é obter uma matriz de transformação $\mathbf{T}$ tal que

$$\mathbf{C_Y} = \mathbf{T}^T\mathbf{C_Z}\mathbf{T} = \mathbf{I}, \tag{1.63}$$

em que $\mathbf{I}$ é a matriz identidade.

Pré-multiplicando a equação (1.63) por $\left(\mathbf{T}^T\right)^{-1}$ e, em seguida, pós-multiplicando por $(\mathbf{T})^{-1}$, obtemos

$$\mathbf{C_Z} = \left(\mathbf{T}^T\right)^{-1} \mathbf{T}^{-1}. \tag{1.64}$$

Considerando o fato de que $\left(\mathbf{T}^T\right)^{-1} = \left(\mathbf{T}^{-1}\right)^T$ e denotando $\left(\mathbf{T}^T\right)^{-1}$ pela matriz $\mathbf{L}$, podemos reescrever a equação (1.64) da seguinte forma

$$\mathbf{C_Z} = \mathbf{L}\mathbf{L}^T.$$

Se $\mathbf{C_Z}$ for uma matriz definida positiva, então a matriz de tranformação $\mathbf{T}$ poderá ser obtida por meio da decomposição de Cholesky, em que $\mathbf{L}$ é uma matriz triangular inferior com diagonal positiva.

Como $\mathbf{T}^T = \mathbf{L}^{-1}$, as matrizes jacobianas da transformação são dadas, respectivamente, por

$$\mathbf{J_{yz}} = \mathbf{L}^{-1} \tag{1.65}$$

e

$$\mathbf{J_{zy}} = \mathbf{L}. \tag{1.66}$$

Portanto, as transformações do espaço $\mathbb{Z} \rightarrow \mathbb{Y}$ e do espaço $\mathbb{Y} \rightarrow \mathbb{Z}$ são dadas, respectivamente, por

$$\mathbf{y} = \mathbf{J_{yz}} \cdot \mathbf{z} \tag{1.67}$$

e

$$\mathbf{z} = \mathbf{J_{zy}} \cdot \mathbf{y}. \tag{1.68}$$

Note que as transformações (1.67) e (1.68), são idênticas as transformações (1.61) e (1.62), repectivamente, ou seja, as transformações via decomposição ortogonal ou decomposição de Cholesky são equivalentes.

União das etapas da transformação composta

Nesse momento, acabamos de apresentar as etapas do mapeamento de um conjunto de variáveis aleatórias de um espaço $\mathbb{X}$ para um espaço $\mathbb{Z}$, e do espaço $\mathbb{Z}$ para o espaço $\mathbb{Y}$.

É possível simplicar a implementação da transformação composta combinando as suas três etapas, utilizando para isto, as matrizes jacobianas obtidas por meio da aplicação da regra da cadeia, assim

$$\mathbf{J_{yx}} = \mathbf{J_{yz}} \cdot \mathbf{J_{zx}} \tag{1.69}$$

$$\mathbf{J_{xy}} = \mathbf{J_{xz}} \cdot \mathbf{J_{zy}}. \tag{1.70}$$

Considerando o fato de que $\mathbf{J_{xz}} = \mathbf{D}^{neq}$ e $\mathbf{J_{zx}} = (\mathbf{D}^{neq})^{-1}$ e, além disso, utilizando

as matrizes jacobianas (1.59) e (1.60) da decomposição ortogonal, as equações (1.69) e (1.70) tornam-se

$$\begin{aligned} \mathbf{J_{yx}} &= \mathbf{Q}^{-\frac{1}{2}}\mathbf{P}^T \cdot (\mathbf{D}^{neq})^{-1} \\ \mathbf{J_{xy}} &= \mathbf{D}^{neq} \cdot \mathbf{P}\mathbf{Q}^{\frac{1}{2}}. \end{aligned} \tag{1.71}$$

Por outro lado, utilizando as matrizes jacobianas (1.65) e (1.66) da decomposição de Cholesky, as equações (1.69) e (1.70) tornam-se

$$\begin{aligned} \mathbf{J_{yx}} &= \mathbf{L}^{-1} \cdot (\mathbf{D}^{neq})^{-1} \\ \mathbf{J_{xy}} &= \mathbf{D}^{neq} \cdot \mathbf{L}. \end{aligned} \tag{1.72}$$

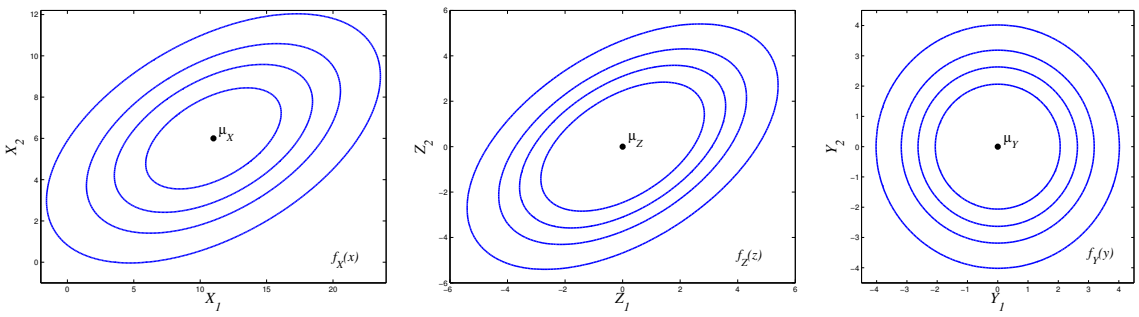
De forma geral, a transformação das variáveis originas $\mathbf{X}$ para variáveis normais padrão independentes $\mathbf{Y}$ é obtida pela seguinte transformação

$$\mathbf{y} = \mathbf{J_{yx}} \{\mathbf{x} - \mathbf{M}^{neq}\} \tag{1.73}$$

e, consequentemente, a transformação das variáveis normais padrão independentes, $\mathbf{Y}$, para o espaço das variáveis originais, é dada por

$$\mathbf{x} = \mathbf{J_{xy}} \cdot \mathbf{y} + \mathbf{M}^{neq}. \tag{1.74}$$

A Figura 1.2 ilustra as etapas da transformação composta considerando o caso em que as variáveis aleatórias $\mathbf{X}$ seguem a distribuição normal bivariada com correlação igual a 0,5.



(a) Espaço original $\mathbb{X}$ (b) Espaço reduzido $\mathbb{Z}$ cujas v.a.'s são correlacionadas (c) Espaço reduzido $\mathbb{Y}$ cujas v.a.'s são independentes

Figura 1.2: Transformação das variáveis X_1 e X_2 em variáveis normais padrão independentes.

Diante do que foi exposto, finalizamos o Capítulo 1 apresentando as ideias do algoritmo FORM na busca pelo ponto de projeto para problemas de confiabilidade estrutural, cujas variáveis envolvidas são possivelmente correlacionadas e seguem qualquer distribuição de probabilidade.

Algoritmo 1.1. *FORM*

Dados: $\mathbf{x}^0 \in \mathbb{R}^n$ *(geralmente o vetor de médias), as matrizes jacobianas* $\mathbf{J_{yz}}$ *e* $\mathbf{J_{zy}}$ *em* $\mathbf{x}^0$ *e a matriz de correlação equivalentes* $\rho_{\mathbf{Z}}$ *ou covariância* $\mathbf{C_Z}$.
$k = 0$
REPITA *enquanto o critério de parada não for satisfeito*

1. *Determinar os parâmetros das distribuições normais equivalentes em* $\mathbf{x}^k$, *dados em* (1.40), *cujas componentes são definidas em* (1.38) *e* (1.39).
2. *Atualizar as matrizes jacobianas* $\mathbf{J_{yx}}$ *e* $\mathbf{J_{xy}}$, *usando as equações* (1.71) *ou* (1.72).
3. *Transformar o ponto* $\mathbf{x}^k$ *em* $\mathbf{y}^k$ *no espaço* $\mathbb{Y}$, *por meio da equação* (1.73).
4. *Calcular o novo ponto* $\mathbf{y}^{k+1}$.
5. *Retornar para o espaço* $\mathbb{X}$ *transformando o ponto* $\mathbf{y}^{k+1}$ *em* $\mathbf{x}^{k+1}$, *utilizando* (1.74).
6. *Calcular o índice de confiabilidade* $\beta = \left\|\mathbf{y}^{k+1}\right\|$.

$k = k + 1$

O critério de parada frequentemente utilizado para o Algoritmo 1.1 é

$$\left|h(\mathbf{y}^k)\right| < \varepsilon \quad \text{e} \quad 1 - \frac{\left|\nabla h(\mathbf{y}^k)^T \mathbf{y}^k\right|}{\|\nabla h(\mathbf{y}^k)\| \, \|\mathbf{y}^k\|} < \varepsilon, \tag{1.75}$$

sendo $\varepsilon = 10^{-4}$.

Vários algoritmos de otimização podem ser empregados para o cálculo de $\mathbf{y}^{k+1}$ (passo *4* do Algoritmo 1.1). Porém, o algoritmo comumente usado nessa etapa é o HLRF. No Capítulo 2 vamos descrever o algoritmo HLRF, em sua versão clássica, e em uma versão melhorada, denominada iHLRF. Para finalizar o capítulo apresentamos o algoritmo nHLRF que estamos propondo.

Capítulo 2

Obtenção do ponto de projeto

Nesse capítulo discutimos principalmente os métodos utilizados para o cálculo do ponto $\mathbf{y}^{k+1}$ que corresponde, especificamente, ao passo *4* do Algoritmo 1.1 mostrado no capítulo anterior.

Para tanto propomos um novo algoritmo, baseado no algoritmo HLRF, que utiliza uma nova função de mérito diferenciável juntamente com as condições de Wolfe na seleção do comprimento do passo na busca linear. Mostramos que sob determinadas circunstâncias o algoritmo proposto gera um sequência de pontos que converge para um minimizador local do problema.

Apresentamos também, nesse capítulo, os algoritmos HLRF e iHLRF que serão utilizados nesse trabalho para fins de comparação.

As informações contidas nesse capítulo e no Capítulo 4, referem-se ao conteúdo do artigo de Santos, Matioli e Beck [48], intitulado "*New optimization algorithms for structural reliability analysis*" publicado na revista "*Computer Modeling in Engineering and Science*".

2.1 Algoritmo HLRF e melhorias

Ao introduzir a ideia de linearização da superfície de falha no ponto mais próximo à origem no espaço padrão normal, Hasofer e Lind [17] desenvolveram um algoritmo iterativo para calcular o ponto de projeto. Em um estudo posterior, Rackwitz e Fiessler [41] propuseram o mesmo algoritmo para a obtenção do ponto de projeto, juntamente com uma técnica de transformação para variáveis aleatórias não Gaussianas, permitindo assim, considerá-las na resolução dos problemas de confiabilidade estrutural. Este algoritmo tem grande popularidade no meio científico devido à sua simplicidade. Em reconhecimento ao trabalho original de Hasofer e Lind e do esforço de Rackwitz e Fiessler para a popularização do algoritmo, Liu e Kiureghian [25] nomearam-no de algoritmo HLRF.

O algoritmo HLRF baseia-se no método de Newton, no sentido de que, a cada iteração k a equação de estado limite é substituída por sua linearização no ponto corrente

$\mathbf{y}^k$, de forma que, o próximo iterando $\mathbf{y}^{k+1}$ é o ponto sobre a equação de estado limite linearizada, dada por $h(\mathbf{y}^k) + \nabla h(\mathbf{y}^k)^T(\mathbf{y} - \mathbf{y}^k) = 0$, que está mais próximo da origem do sistema de coordenadas reduzidas.

A Figura 2.1 ilustra uma iteração do algoritmo HLRF considerando, para isto, uma função desempenho não linear. Note que, o ponto inicial e o ponto gerado não precisam necessariamente ser viáveis.

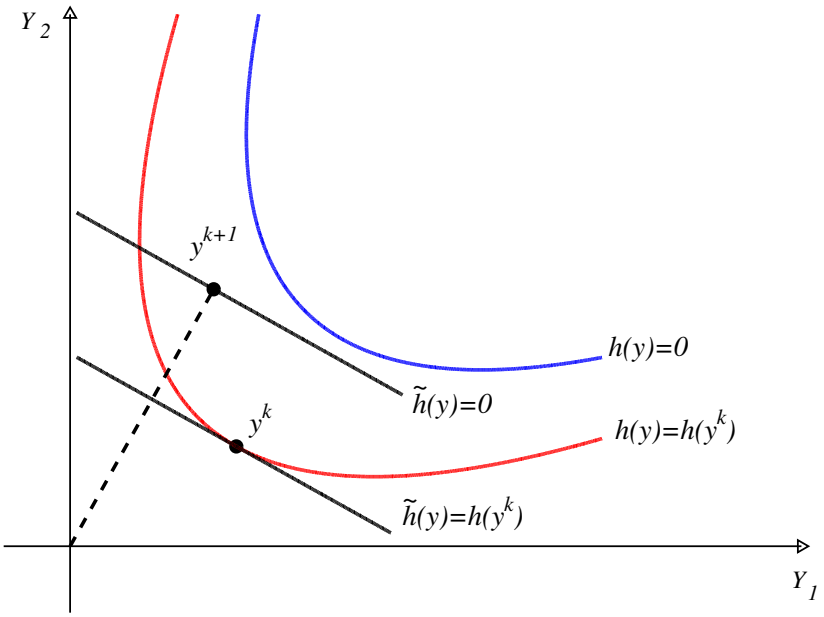


Figura 2.1: Ilustração de uma iteração do algoritmo HLRF.

Diante do exposto, a fórmula de recorrência para $\mathbf{y}^{k+1}$ pode ser obtida considerando inicialmente a aproximação linear de $h(\mathbf{y})$ no ponto corrente, isto é,

$$\tilde{h}(\mathbf{y}) = h(\mathbf{y}^k) + \nabla h(\mathbf{y}^k)^T(\mathbf{y} - \mathbf{y}^k). \tag{2.1}$$

Em seguida, substituindo a equação de estado limite $h(\mathbf{y}) = 0$ por sua linearização, $\tilde{h}(\mathbf{y})$, segue que

$$h(\mathbf{y}^k) + \nabla h(\mathbf{y}^k)^T(\mathbf{y} - \mathbf{y}^k) = 0. \tag{2.2}$$

O ponto sobre a equação de estado limite linearizada mais próximo da origem é obtido pela interseção de (2.2) com a reta perpendicular a ela, dada por

$$\mathbf{y} = t\nabla h(\mathbf{y}^k), \quad \text{com } t \in \mathbb{R}. \tag{2.3}$$

Assim, calculando a interseção entre (2.2) e (2.3), temos

$$t = \frac{\nabla h(\mathbf{y}^k)^T\mathbf{y}^k - h(\mathbf{y}^k)}{\|\nabla h(\mathbf{y}^k)\|^2} \tag{2.4}$$

e, finalmente, substituindo a equação (2.4) em (2.3), obtemos a fórmula recursiva do algoritmo HLRF

$$\mathbf{y}^{k+1} = \frac{\left[\nabla h(\mathbf{y}^k)^T \mathbf{y}^k - h(\mathbf{y}^k)\right] \nabla h(\mathbf{y}^k)}{\left\|\nabla h(\mathbf{y}^k)\right\|^2}, \tag{2.5}$$

onde $\mathbf{y} \in \mathbb{R}^n$ é o vetor das variáveis definidas no espaço normal reduzido e $h : \mathbb{R}^n \to \mathbb{R}$ é a superfície de falha.

Optamos por não apresentar especificamente o algoritmo HLRF, pois o mesmo pode ser obtido simplesmente acrescentando a sua fórmula recursiva, dada por (2.5), no passo *4* do Algoritmo 1.1, descrito no Capítulo 1.

A popularidade do algoritmo HLRF na resolução do problema (1.15) esta relacionada diretamente à sua simplicidade e eficiência, pois necessita de poucas operações pelo fato de utilizar apenas produtos de vetores e, também, ao baixo custo computacional. Porém, não há garantia de convergência, principalmente em problemas cuja superfície de falha é altamente não linear. De acordo com Liu e Kiureghian [25, 26], o algoritmo é conhecido por falhar em um número considerável de problemas.

Liu e Kiureghian [25, 26] apresentaram um algoritmo alternativo denominado M-HLRF para contornar os problemas de convergência do algoritmo HLRF. Os autores combinaram a direção de busca $\mathbf{d}^k$ gerada pelo algoritmo HLRF, dada por

$$\mathbf{d}^k = \frac{1}{\left\|\nabla h\left(\mathbf{y}^k\right)\right\|^2} \left[\nabla h\left(\mathbf{y}^k\right)^T \mathbf{y}^k - h\left(\mathbf{y}^k\right)\right] \nabla h\left(\mathbf{y}^k\right) - \mathbf{y}^k, \tag{2.6}$$

com uma busca linear nessa direção. A busca linear é realizada até que uma redução suficiente em uma função de mérito, $p(\mathbf{y})$, seja alcançada. A seguinte função de mérito foi considerada

$$p(\mathbf{y}) = \frac{1}{2} \left\| \mathbf{y} - \frac{\nabla h\left(\mathbf{y}\right)^T \mathbf{y}}{\left\|\nabla h\left(\mathbf{y}\right)\right\|^2} \nabla h\left(\mathbf{y}\right) \right\|^2 + \frac{1}{2} c \cdot h\left(\mathbf{y}\right)^2. \tag{2.7}$$

Apesar da robustez ter sido melhorada, Liu e Kiureghian [25] reconheceram que a convergência do algoritmo M-HLRF não pode ser garantida, uma vez que a função de mérito $p(\mathbf{y})$ pode ter mínimos que não são soluções para o problema original (1.15). Além disso, $\mathbf{d}^k$ pode não ser uma direção de descida para a função de mérito, dada por (2.7), em alguns casos.

Um algoritmo melhorado (iHLRF) foi apresentado por Zhang e Kiuregian [58], também baseado em uma busca linear na direção HLRF, de forma que o próximo iterando é dado por

$$\mathbf{y}^{k+1} = \mathbf{y}^k + \alpha^k \mathbf{d}^k.$$

Note que a fórmula recursiva (2.5) do algoritmo de HLRF é obtida quando o passo completo é usado, ou seja, $\alpha^k = 1$.

A seguinte função de mérito não-diferenciável é usada pelos autores para deter-

minar α^k na busca linear

$$p(\mathbf{y}) = \frac{1}{2}\mathbf{y}^T\mathbf{y} + c\,|h(\mathbf{y})|\,. \tag{2.8}$$

O passo apropriado α^k é o primeiro valor que satisfaz uma condição de decréscimo suficiente na função de mérito, dada por

$$p(\mathbf{y}^{k+1}) \leq p(\mathbf{y}^k) + m_1\alpha^k\nabla p(\mathbf{y}^k)^T\mathbf{d}^k, \tag{2.9}$$

onde $\nabla p(\mathbf{y}^k) = \mathbf{y}^k + c \cdot sign(h(\mathbf{y}^k))\nabla h(\mathbf{y}^k)$ e $m_1 \in (0,1)$. A desigualdade (2.9) define a conhecida regra de Armijo.

Zhang e Kiureghian [58] mostraram que a direção $\mathbf{d}^k$, dada pela equação (2.6), é uma direção de descida para $m(\mathbf{y})$, dada em (2.8), em $\mathbf{y}^k \in \mathbb{R}^n$ desde que a seguinte condição seja satisfeita

$$c > \frac{\|\mathbf{y}^k\|}{\|\nabla h\,(\mathbf{y}^k)\|}. \tag{2.10}$$

A fim de satisfazer a condição (2.10), Zhang e Kiureghian [58] estabeleceram as seguintes regras para atualização de c:

Se $\left|h(\mathbf{y}^k)\right| \geq \Delta$

$$c = \eta \max\left\{\frac{\|\mathbf{y}^k\|}{\|\nabla h(\mathbf{y}^k)\|}, \frac{1}{2}\frac{\|\mathbf{y}^k + \mathbf{d}^k\|^2}{|h(\mathbf{y}^k)|}\right\};$$

caso contrário,

$$c = \eta\frac{\|\mathbf{y}^k\|}{\|\nabla h\,(\mathbf{y}^k)\|}.$$

Os valores dos parâmetros Δ e η sugeridos por [58] são $10^{-3}\,|h(\mathbf{y}^0)|$ e 2, respectivamente.

De acordo com Zhang e Kiureghian [58], o algoritmo iHLRF tem se mostrado mais eficiente e mais confiável do que os algoritmos HLRF e M-HLRF. No entanto, uma importante desvantagem do algoritmo iHLRF é o fato da função de mérito não ser diferenciável, isto porque, na busca linear é necessário verificar iterativamente se houve uma redução suficiente na função de mérito e, para isto, é preciso avaliar o seu gradiente, que por sua vez pode não estar definido em todos os pontos do domínio.

A seguir apresentamos o algoritmo iHLRF proposto em [58] sendo que este, assim como o algoritmo HLRF, deve ser inserido no passo *4* do Algoritmo 1.1 para a obtenção do ponto $\mathbf{y}^{k+1}$.

Algoritmo 2.1. *iHLRF*

Dados: $\mathbf{y}^k \in \mathbb{R}^n$, $\Delta > 0$, $\eta > 1$, $\alpha \in (0, 1]$, $0 < m_1 < 1$

1. *Calcule a direção de busca* $\mathbf{d}^k$ *dada em* (2.6)
2. *Determine o parâmetro de penalidade* c *da função de mérito* (2.8)

   SE $\left|h(\mathbf{y}^k)\right| \geq \Delta$

   $$c = \eta \max\left\{ \frac{\left\|\mathbf{y}^k\right\|}{\left\|\nabla h(\mathbf{y}^k)\right\|}, \frac{1}{2}\frac{\left\|\mathbf{y}^k + \mathbf{d}^k\right\|^2}{|h(\mathbf{y}^k)|} \right\}$$

   SENÃO

   $$c = \eta \frac{\left\|\mathbf{y}^k\right\|}{\left\|\nabla h\left(\mathbf{y}^k\right)\right\|}$$

3. *Determine* α^k *usando a desigualdade* (2.9)

   ENQUANTO $p(\mathbf{y}^k + \alpha\mathbf{d}^k) - p(\mathbf{y}^k) > m_1\alpha\nabla p(\mathbf{y}^k)^T\mathbf{d}^k$

   FAÇA $\alpha = \dfrac{\alpha}{2}$

   $\alpha^k = \alpha$

4. *Faça* $\mathbf{y}^{k+1} = \mathbf{y}^k + \alpha^k\mathbf{d}^k$

É importante ressaltar que no processo iterativo, tanto os algoritmos HLRF, iHLRF como o algoritmo nHLRF, que será apresentado mais adiante, realizam apenas um passo em cada iteração do Algoritmo 1.1, até que o critério de parada, dado em (1.75), seja atendido.

2.2 Algoritmo nHLRF

Nessa seção propomos um novo algoritmo denominado nHLRF usando, para isto, uma função de mérito diferenciável que decresce a cada iteração ao longo da busca linear na direção HLRF, dada por (2.6). A função de mérito é a seguinte

$$m(\mathbf{y}) = \frac{1}{2}\mathbf{y}^T\mathbf{y} + \frac{c}{2}h\left(\mathbf{y}\right)^2 \tag{2.11}$$

onde $c > 0$.

Para estabelecer a prova de convergência do algoritmo nHLRF, é necessário que a direção de busca HLRF, dada pela equação (2.6), seja uma direção de descida para a função de mérito (2.11).

A seguir, um importante resultado deste trabalho é apresentado.

Teorema 2.1. *A direção de busca HLRF, dada em* (2.6)*, é uma direção de descida no ponto* $\mathbf{y}^k \in \mathbb{R}^n$ *para a função de mérito* (2.11)*, desde que*

$$c > -\frac{1}{h(\mathbf{y}^k)} \cdot \frac{\left(\mathbf{y}^k\right)^T \nabla h(\mathbf{y}^k)}{\left\|\nabla h(\mathbf{y}^k)\right\|^2} \tag{2.12}$$

em que $h(\mathbf{y}^k) \neq 0$.

Demonstração. Devemos provar que $\nabla m(\mathbf{y}^k)^T \mathbf{d}^k < 0$, para todo $\mathbf{y}^k \in \mathbb{R}^n$. Temos que,

$$\nabla m(\mathbf{y}^k) = \mathbf{y}^k + c \cdot h(\mathbf{y}^k) \nabla h(\mathbf{y}^k). \tag{2.13}$$

Das equações (2.6) e (2.13), obtemos

$$\nabla m(\mathbf{y}^k)^T \mathbf{d}^k = -\left(\left(\mathbf{y}^k\right)^T \mathbf{y}^k - \frac{(\nabla h(\mathbf{y}^k)^T \mathbf{y}^k)^2}{\left\|\nabla h(\mathbf{y}^k)\right\|^2} \right) - \left(c \cdot h(\mathbf{y}^k)^2 + \frac{h(\mathbf{y}^k)\left(\mathbf{y}^k\right)^T \nabla h(\mathbf{y}^k)}{\left\|\nabla h(\mathbf{y}^k)\right\|^2} \right). \tag{2.14}$$

Usando a desigualdade de Schwartz: $\left|\nabla h(\mathbf{y}^k)^T \mathbf{y}^k\right| \leq \left\|\nabla h(\mathbf{y}^k)\right\| \cdot \left\|\mathbf{y}^k\right\|$, concluímos que

$$\left(\mathbf{y}^k\right)^T \mathbf{y}^k - \frac{(\nabla h(\mathbf{y}^k)^T \mathbf{y}^k)^2}{\left\|\nabla h(\mathbf{y}^k)\right\|^2} \geq 0. \tag{2.15}$$

Considerando a condição (2.12) na avaliação de (2.14)

$$c > -\frac{1}{h(\mathbf{y}^k)} \cdot \frac{\left(\mathbf{y}^k\right)^T \nabla h(\mathbf{y}^k)}{\left\|\nabla h(\mathbf{y}^k)\right\|^2}, \tag{2.16}$$

podemos garantir que

$$c \cdot h(\mathbf{y}^k)^2 + \frac{h(\mathbf{y}^k)\left(\mathbf{y}^k\right)^T \nabla h(\mathbf{y}^k)}{\left\|\nabla h(\mathbf{y}^k)\right\|^2} > 0. \tag{2.17}$$

Consequentemente, combinando as equações (2.15) e (2.17), segue que

$$\nabla m(\mathbf{y}^k)^T \mathbf{d}^k < 0.$$

Além disso, observe que $\nabla m(\mathbf{y}^k)^T \mathbf{d}^k = 0$ apenas se

$$h(\mathbf{y}^k) = 0 \text{ e } \ {\mathbf{y}^k}^T \mathbf{y}^k - \frac{(\nabla h(\mathbf{y}^k)^T \mathbf{y}^k)^2}{\left\|\nabla h(\mathbf{y}^k)\right\|^2} = 0,$$

o que equivale a dizer que $\mathbf{y}^k \in \mathbb{R}^n$ satisfaz as condições de KKT para o problema (1.15), sendo portanto um ponto estacionário para (1.15). □

O Teorema 2.1 garante que a função de mérito (2.11) decresce na direção $\mathbf{d}^k$ a partir de $\mathbf{y}^k$. Contudo, uma simples redução da função de mérito, $(m(\mathbf{y}^{k+1}) < m(\mathbf{y}^k))$, não garante a convergência do algoritmo.

A fim de garantir a convergência, uma primeira condição a ser imposta é a existência de α, ou seja, do comprimento do passo satisfazendo a condição expressa por (2.9). De qualquer maneira, tal condição pode levar a um desempenho insatisfatório, uma vez

que permite a escolha de passos muito curtos. Para evitar passos curtos, ou seja, α pequeno, a seguinte condição de curvatura é considerada

$$\nabla m(\mathbf{y}^k + \alpha \mathbf{d}^k)^T \mathbf{d}^k \geq m_2 \nabla m(\mathbf{y}^k)^T \mathbf{d}^k, \tag{2.18}$$

para algum $m_2 \in (m_1, 1)$ e $m_1 \in (0, 1)$.

O decréscimo suficiente, dado por (2.9), e a condição de curvatura, dada por (2.18), são conhecidos coletivamente como condições de Wolfe, em que $0 < m_1 < m_2 < 1$.

O Teorema 2.2 assegura que para qualquer direção de descida, $\mathbf{d}^k$, existem pontos $\mathbf{y}^k + \alpha^k \mathbf{d}^k$ satisfazendo as condições (2.9) e (2.18).

Teorema 2.2. *Seja $m : \mathbb{R}^n \to \mathbb{R}$ continuamente diferenciável. Suponha que, para todo $\mathbf{y}^k \in \mathbb{R}^n$ e $\mathbf{d}^k \in \mathbb{R}^n$, $\nabla m(\mathbf{y}^k)^T \mathbf{d}^k < 0$ e assuma que $\left\{m(\mathbf{y}^k + \alpha \mathbf{d}^k) | \alpha > 0\right\}$ é limitada inferiormente. Então, se $0 < m_1 < m_2 < 1$, existe $\alpha_u > \alpha_l > 0$ tal que $\mathbf{y}^{k+1} = \mathbf{y}^k + \alpha^k \mathbf{d}^k$ satisfaz as condições de Wolfe, dadas em* (2.9) *e* (2.18)*, se $\alpha^k \in (\alpha_l, \alpha_u)$.*

Demonstração. Dennis [12, Teorema 6.3.2]. □

O Teorema 2.3 estabelece as condições para as quais o algoritmo nHLRF, apresentado a seguir, converge para um ponto estacionário de $m(\mathbf{y}^k)$, ou seja, $\nabla m(\mathbf{y}^k) \to 0$.

Teorema 2.3. *Considere uma iteração da forma $\mathbf{y}^{k+1} = \mathbf{y}^k + \alpha^k \mathbf{d}^k$, onde $\mathbf{d}^k \in \mathbb{R}^n$ é uma direção de descida e α^k satisfaz as condições de Wolfe* (2.9) *e* (2.18)*. Suponha que $m(\mathbf{y})$ é limitada por baixo em $\mathbb{R}^m$ e que $m(\mathbf{y})$ é continuamente diferenciável em um conjunto aberto $\mathcal{S}$ contendo o conjunto de nível $\mathcal{N} = \{\mathbf{y} : m(\mathbf{y}) \leq m(\mathbf{y}^0)\}$, onde $\mathbf{y}^0$ é o ponto inicial da iteração. Assuma também que $\nabla m(\mathbf{y})$ é Lipschitz contínua em $\mathcal{S}$, isto é, existe uma constante $L > 0$ tal que*

$$\|\nabla m(\mathbf{y}) - \nabla m(\overline{\mathbf{y}})\| \leq L \|\mathbf{y} - \overline{\mathbf{y}}\|,$$

para todo $\mathbf{y}$, $\overline{\mathbf{y}} \in \mathcal{S}$.
Então,

$$\lim_{k \to \infty} \left\|\nabla m(\mathbf{y}^k)\right\| = 0.$$

Demonstração. Nocedal e Wright [35, Teorema 3.2]. □

Diante do exposto, concluímos que, dado um ponto inicial $\mathbf{y}^0 \in \mathbb{R}^n$ e a direção $\mathbf{d}^k$, dada em (2.6), se as hipóteses dos Teoremas 2.1, 2.2 e 2.3 são válidas, o algoritmo nHLRF irá gerar uma sequência de pontos que satisfaz as condições de Wolfe e que converge para um ponto estacionário de $m(\mathbf{y})$, que é também solução para o problema (1.15).

Após apresentar os resultados teóricos que estabelecem convergência do método, apresentamos a seguir o algoritmo nHLRF.

Algoritmo 2.2. *nHLRF*

Dados: $\mathbf{y}^k \in \mathbb{R}^n$, $\eta > 0$, $\alpha \in (0, 1]$, $0 < m_1 < m_2 < 1$

1. *Calcule a direção de busca* $\mathbf{d}^k$ *dada em* (2.6)
2. *Determine o parâmetro de penalidade* c *da função de mérito* (2.11)

   SE $h(\mathbf{y}^k) = 0$

   Escolha $c > 0$

   SENÃO

   $$c = \eta \left| \frac{1}{h(\mathbf{y}^k)} \frac{\left(\mathbf{y}^k\right)^T \nabla h(\mathbf{y}^k)}{\|\nabla h(\mathbf{y}^k)\|^2} \right|$$

3. ENQUANTO *as condições de Wolfe, definidas em* (2.9) *e* (2.18)*, não são satisfeitas*

   SE $m(\mathbf{y}^k + \alpha \mathbf{d}^k) - m(\mathbf{y}^k) > m_1 \alpha \nabla m(\mathbf{y}^k)^T \mathbf{d}^k$

   FAÇA $\alpha = \dfrac{\alpha}{2}$

   SE $\nabla m(\mathbf{y}^k + \alpha \mathbf{d}^k)^T \mathbf{d}^k < m_2 \nabla m(\mathbf{y}^k)^T \mathbf{d}^k$

   FAÇA $\alpha = 2\alpha$

   $\alpha^k = \alpha$
4. *Faça* $\mathbf{y}^{k+1} = \mathbf{y}^k + \alpha^k \mathbf{d}^k$

No Algoritmo 2.2 as condições para calcular o parâmetro de penalidade c da função de mérito (2.11), foram estabelecidas com o objetivo de contornar os problemas numéricos causados quando $h(\mathbf{y}^k) = 0$, garantindo assim que a direção $\mathbf{d}^k$, dada em (2.6), decresça para qualquer $\mathbf{y}^k \in \mathbb{R}^n$. Detalhes a respeito da escolha dos parâmetros para a implementação são dados no Capítulo 4.

No capítulo seguinte apresentamos dois métodos de Lagrangeano aumentado aplicados a problemas de programação não linear com restrições de igualdade. Ambos utilizam penalidade quadrática e a estrutura dos métodos modernos para problemas com restrições de desigualdade. Portanto, podem ser vistos como Lagrangeanos aumentados aplicados a problemas com restrições de desigualdade estendidos para problemas com restrições de igualdade, sem acréscimo de variáveis de folga. No principal resultado desse capítulo, mostramos que sob hipóteses convencionais as funções Lagrangeano aumentado geradas pelos dois métodos possuem minimizador local, como no caso do método proposto por Hestenes [18] e Powell [40]. Para finalizar o capítulo, experimentos numéricos com problemas da coleção CUTEr são apresentados, ilustrando o desempenho dos algoritmos.

Capítulo 3

Métodos de Lagrangeano aumentado

Nos métodos de Lagrangeano aumentado clássicos a penalidade quadrática introduzida por Hestenes [18] e Powell [40] é utilizada para problemas com restrições de igualdade, no formato do problema (1.15), mas com m restrições de igualdade

$$\begin{array}{ll} \text{minimizar} & f(x) \\ \text{sujeito a} & h(x) = 0 \\ & x \in \mathbb{R}^n \end{array} \tag{3.1}$$

onde $f : \mathbb{R}^n \to \mathbb{R}$ e $h : \mathbb{R}^n \to \mathbb{R}^m$ são funções continuamente diferenciáveis.

A função Lagrangeano aumentado proposta por estes autores é definida como

$$\mathcal{L}(x, \lambda, \rho) = f(x) + \sum_{i=1}^{m} \left\{ \frac{\rho}{2} [h_i(x)]^2 + \lambda_i h_i(x) \right\} \tag{3.2}$$

em que $x \in \mathbb{R}^n$, $\lambda \in \mathbb{R}^m$ é o vetor de multiplicadores de Lagrange e $\rho > 0$ é o parâmetro de penalidade. É conhecido que, para $\rho > \bar{\rho} > 0$ o método de Lagrangeano aumentado converge para uma solução do problema (3.1).

Neste trabalho, propomos dois novos métodos de Lagrangeano aumentado para resolução do problema de confiabilidade estrutural (1.15). Contudo, apresentaremos as discussões teóricas dos métodos de Lagrangeano aumentado para o problema (3.1), uma vez que (1.15) é uma particularização de (3.1) com uma única restrição, ou seja, $m = 1$.

Os métodos que estamos propondo são extensões dos resultados apresentados por Matioli e Gonzaga [32] e Tseng e Bertsekas [53] para problemas de programação não linear com restrições de desigualdade

$$\begin{array}{ll} \text{minimizar} & f(x) \\ \text{sujeito a} & g(x) \leq 0 \\ & x \in \mathbb{R}^n \end{array} \tag{3.3}$$

em que $f : \mathbb{R}^n \to \mathbb{R}$ e $g : \mathbb{R}^n \to \mathbb{R}^p$ são funções continuamente diferenciáveis.

Quase na totalidade, os métodos de Lagrangeano aumentado propostos para resolver o problema (3.3) utilizam a seguinte metodologia: definem o problema dual associado ao problema (3.3), que está no formato primal, e o algoritmo de ponto proximal associado ao problema dual.

Devido à estrutura do problema dual ser mais simples do que a do primal, do ponto de vista teórico, a convergência é mais fácil de ser tratada no dual. Por outro lado, a computação é mais simples de ser tratada para o problema primal. Portanto, usando alguma medida de distância, por exemplo, norma Euclidiana, ϕ-divergências, distância de Bregman, é possível desenvolver métodos aplicados ao problema dual, com boas propriedades de convergência e que sob algumas hipóteses são equivalentes ao método de Lagrangeano aumentado no primal.

Rockafellar [42, 43] foi um dos pioneiros a utilizar essa metodologia através do algoritmo de pontos proximais para operador monótono maximal.

Muitos artigos foram publicados sobre os métodos de Lagrangeano aumentado aplicados ao problema (3.3), por exemplo, [2, 6, 7, 9, 11, 14, 19, 20, 21, 24, 29, 32, 39, 44, 51, 53]. Inicialmente, vamos nos ater aos trabalhos de Matioli e Gonzaga [32] e Tseng e Bertsekas [53], pois estamos estendendo estes para problemas no formato de (3.1) sem a introdução de variáveis de folga.

A seguir, relacionamos alguns fatos importantes sobre esses métodos de Lagrangeano aumentado para problemas com restrições de desigualdade (3.3), que nos motivaram a generalizá-los também para problemas com restrições de igualdade (3.1).

- A convergência do método é garantida mesmo se o parâmetro de penalidade for mantido constante, contudo, se o parâmetro decrescer, a convergência será mais rápida [32, 53];

- A existência de uma certa relação de equivalência entre região de confiança e o kernel dual proposto, de maneira que, aumentando o parâmetro de penalidade a região de confiança é diminuída [32];

- Para o caso particular do problema de programação linear e dependendo da escolha do parâmetro de penalidade, o comportamento do algoritmo de ponto proximal (olhando no dual) é similar ao algoritmo afim escala, o qual é conhecido possuir boas propriedades de convergência [32];

- Testes numéricos com diversos problemas têm mostrado que estes métodos apresentam bom desempenho, inclusive para problemas de grande porte incluindo limites nas variáveis. Birgin, Castillo e Martinez [4], testaram 65 implementações de diferentes métodos de Lagrangeano aumentados. Para a classe de problemas testados, os autores concluíram que os métodos clássicos apresentam superioridade aos métodos

modernos, reforçando nossa motivação em introduzir novos métodos de Lagrangeano aumentado similares aos clássicos, mas com propriedades dos métodos modernos.

Na seção 3.1, descrevemos as funções Lagrangeano aumentado com as novas funções de penalidade associadas ao problema (3.1). Na seção 3.2, mostramos a existência de um minimizador local estrito das funções Lagrangeano aumentado, assim como, os algoritmos dos métodos propostos. O primeiro refere-se as novas funções propostas e o segundo, com o propósito de comparação, ao método clássico de Hestenes [18] e Powell [40].

Testes numéricos com problemas da coleção CUTEr [10], foram desenvolvidos para comparar o desempenho dos métodos apresentados e os resultados são descritos e analisados na Seção 3.3.

3.1 Introdução das novas penalidades

Considere a função quadrática de uma variável real

$$y \in \mathbb{R} \to \theta(y) = y^2. \tag{3.4}$$

Associado ao problema (3.1) definimos as novas funções Lagrangeano aumentado, as quais são extensões de [32] e [53] para o problema com restrições de desigualdade, ou seja,

$$(x, \lambda, \gamma) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^m_{++} \mapsto \mathcal{L}(x, \lambda, \gamma) = f(x) + \sum_{i=1}^{m} \lambda_i h_i(x) + \frac{1}{2} \sum_{i=1}^{m} \theta(\sqrt{\gamma_i} h_i(x)) \tag{3.5}$$

e

$$(x, \lambda, \gamma) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^m_{++} \mapsto \mathcal{L}(x, \lambda, \omega) = f(x) + \sum_{i=1}^{m} \lambda_i h_i(x) + \frac{1}{2} \sum_{i=1}^{m} \omega_i \theta(h_i(x)). \tag{3.6}$$

Substituindo a função (3.4) nas equações (3.5) e (3.6), temos

$$(x, \lambda, \gamma) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^m_{++} \mapsto \mathcal{L}(x, \lambda, \gamma) = f(x) + \sum_{i=1}^{m} \lambda_i h_i(x) + \frac{1}{2} \sum_{i=1}^{m} \gamma_i h_i(x)^2 \tag{3.7}$$

e

$$(x, \lambda, \omega) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^m_{++} \mapsto \mathcal{L}(x, \lambda, \omega) = f(x) + \sum_{i=1}^{m} \lambda_i h_i(x) + \frac{1}{2} \sum_{i=1}^{m} \omega_i h_i(x)^2. \tag{3.8}$$

em que λ é o vetor de multiplicadores de Lagrange associado às restrições h e, γ e ω são os parâmetros de penalidade. Os parâmetros de penalidade γ, em (3.7), e ω, em (3.8),

são definidos, respectivamente, como

$$\gamma_i = \frac{\lambda_i^2}{r_i}, \quad \text{com} \quad r_i > 0 \tag{3.9}$$

e

$$\omega_i = \frac{\lambda_i}{r_i}, \quad \text{com} \quad r_i \neq 0, \tag{3.10}$$

para todo $i = 1, \ldots, m$. Como os parâmetros de penalidade devem ser positivos, escolhemos r_i tal que isso ocorra, por exemplo, em (3.10), $r_i < 0$ se $\lambda_i < 0$ e $r_i > 0$, caso contrário.

Note a presença de γ_i em (3.7) e de ω_i em (3.8) no termo que penaliza as restrições, ambos são dependentes dos multiplicadores de Lagrange. Essa é uma das diferenças em relação a penalidade clássica (3.2). Além disso, no método clássico o parâmetro de penalidade é um escalar, enquanto que para as penalidades propostas é vetorial. Essas diferenças e sua importância serão detalhadas mais adiante.

Matioli e Gonzaga [32] e Tseng e Bertsekas [53] com metodologias similares a (3.5) e (3.6), respectivamente, apresentaram uma família $\mathcal{P}$ de funções penalidade variando a função θ. Ambos construíram Lagrangeanos aumentado aplicados a problemas com restrições de desigualdade. A função Lagrangeano aumentado (3.5), com θ dada pela função barreira de Polyak [38], é usada por Matioli e Gonzaga [32] para apresentar métodos de Lagrangeano aumentado equivalentes a pontos proximais com distância de Bregman no dual. A função Lagrangeano aumentado (3.6), com θ sendo uma função exponencial, é usada por Tseng e Bertsekas [53] para métodos de Lagrangeano aumentado equivalentes a pontos proximais com ϕ-divergências.

Nos métodos de Lagrangeano aumentado o parâmetro de penalidade é um dos poucos que temos a liberdade de escolha para sua atualização. No Lema 3.1, mostramos que para uma escolha particular do parâmetro de penalidade, as três funções Lagrangeano aumentado são equivalentes. Esse fato será importante na atualização do parâmetro de penalidade nos algoritmos de Lagrangeano aumentado que serão apresentados adiante.

Lema 3.1. *Se $\lambda_i \neq 0$, para todo $i = 1, \ldots, m$, e os parâmetros de penalidade γ, em* (3.5)*, e ω, em* (3.6)*, são atualizados como $\gamma_i = \dfrac{\lambda_i^2}{r_i}$ (dado por 3.9) com*

$$r_i = \frac{\lambda_i^2}{\rho} \tag{3.11}$$

e $\omega_i = \dfrac{\lambda_i}{r_i}$ (dado por 3.10) com

$$r_i = \frac{\lambda_i}{\rho} \tag{3.12}$$

onde $\rho > 0$ (dado por 3.2), então as funções (3.2)*,* (3.5) *e* (3.6) *coincidem.*

Demonstração. A prova é imediata. Iremos provar para um caso, pois para o outro é idêntico.

De fato, substituindo $\gamma_i = \dfrac{\lambda_i^2}{r_i}$ com $r_i = \dfrac{\lambda_i^2}{\rho}$ em (3.5) e usando a função θ dada por (3.4), segue que (3.2) e (3.5) coincidem. □

A Figura 3.1 mostra geometricamente, em um caso unidimensional com $r = 1$, o comportamento da penalidade clássica, $\theta(y) = y^2$, e das novas penalidades, $\theta(\sqrt{\gamma}y) = \lambda^2 y^2$ e $\omega\theta(y) = \lambda y^2$, sendo θ uma função quadrática real definida em (3.4). Nos primeiros três gráficos, consideramos $\lambda = 2$ e nos três seguintes, $\lambda = 1/2$. Note a influência de λ na escala dos gráficos, vê-se que $\theta(\sqrt{\gamma}y)$ tem a característica de penalizar mais que as outras duas no caso em que $|\lambda| > 1$ e menos se $|\lambda| < 1$. Por outro lado, $\omega\theta(y)$ é intermediária as outras duas. Se $|\lambda| = 1$ as três coincidem, pois $\gamma = 1$ e $\omega = 1$.

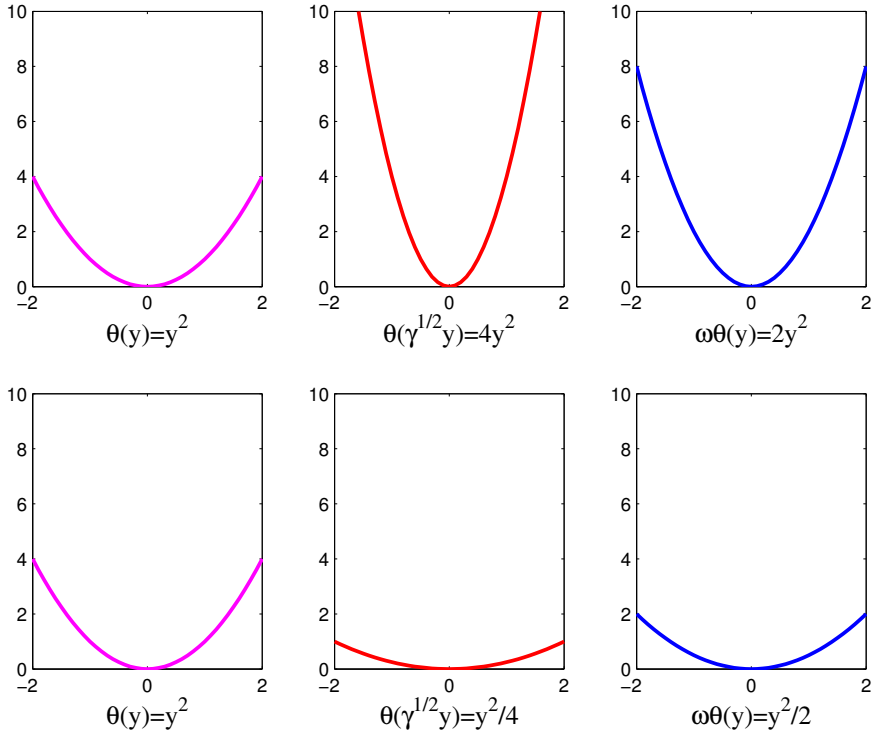


Figura 3.1: Penalidades $\theta(y)$, $\theta(\sqrt{\gamma}y)$ e $\omega\theta(y)$ para dois valores distintos de λ.

Em [32] os autores mostram que para o problema linear com restrições de desigualdade, os algoritmos de pontos proximais com núcleos quadráticos são equivalentes, no dual, ao método afim escala. Estes são alguns pontos que consideramos relevantes em relação à penalidade clássica. Intuitivamente e observando os aspectos geométricos das penalidades propostas, acreditamos que existem atualizações do parâmetro de penalidade em termos dos multiplicadores de Lagrange que aumenta a velocidade de convergência. Esse fato foi usado, embora intuitivamente, nos métodos que implementamos.

3.2 Análise de convergência do Lagrangeano aumentado com as novas funções de penalidade

Nessa seção analisamos as propriedades que são satisfeitas pelas funções (3.7) ou (3.5) e (3.8) ou (3.6). Mostramos que, basta um valor finito de γ e ω para que $\bar{x}$, um ponto que satisfaz a condição suficiente de segunda ordem para o problema (3.1), seja um minimizador local estrito das funções Lagrangeano aumentado (3.7) e (3.8).

Primeiramente, considere Ω o conjunto viável para o problema (3.1), em outras palavras

$$\Omega = \{x \in \mathbb{R}^n : h(x) = 0\}. \tag{3.13}$$

A seguir apresentamos um resultado que generaliza um lema conhecido na literatura e que será utilizado como ferramenta para a demonstração do principal resultado deste capítulo.

Lema 3.2. *Sejam $G \in \mathbb{R}^{n\times n}$ e $A \in \mathbb{R}^{m\times n}$ tais que $G = G^T$ e $z^TGz > 0$ para todo $z \in \mathcal{N}(A)$, $z \neq 0$. Então, existe $\bar{\rho} \geq 0$ tal que $G + A^TDA > 0$ para toda matriz $D = diag(\rho)$ com $\rho_i \geq \bar{\rho}$, $i = 1, \ldots, m$.*

Demonstração. Suponha por absurdo que para todo $k \in \mathbb{N}$, existam $x^k \in \mathbb{R}^n$ e $D_k = diag(\rho^k)$ tais que $\rho_i^k \geq k$, $i = 1, \ldots, m$ e

$$(x^k)^TGx^k + (x^k)^TA^TD_kAx^k \leq 0. \tag{3.14}$$

Sem perda de generalidade, podemos supor que $\left\|x^k\right\| = 1$ para todo $k \in \mathbb{N}$ e $x^k \to \bar{z}$. Definindo $y^k = Ax^k$, concluímos de (3.14) que existe uma constante $M > 0$ tal que

$$(y^k)^TD_ky^k \leq M.$$

Portanto, para cada $j = 1, \ldots, m$, temos

$$\rho_j^k(y_j^k)^2 \leq \sum_{i=1}^{m} \rho_i^k(y_i^k)^2 = (y^k)^TD_ky^k \leq M.$$

Como $\rho_j^k \to \infty$, segue que $y^k \to 0$. Mas $y^k = Ax^k \to A\bar{z}$, o que implica que $A\bar{z} = 0$. Por outro lado, de (3.14) segue que $(x^k)^TGx^k \leq 0$, fornecendo $\bar{z}^TG\bar{z} \leq 0$, uma contradição. □

No próximo Teorema, usamos a hipótese de que o problema (3.1) satisfaz a condição de qualificação LICQ, dada na Definição 1.3.

Teorema 3.3 (Condições suficientes de segunda ordem). *Suponha que para qualquer ponto viável $\bar{x} \in \mathbb{R}^n$, existe um vetor de multiplicadores de Lagrange $\bar{\lambda}$ tal que as condições de KKT são satisfeitas para o problema* (3.1)*. Suponha também que, $z^T\nabla^2_{xx}\ell(\bar{x}, \bar{\lambda})z > 0$,*

para todo $z \in \mathcal{N}(\nabla h(\bar{x}))$, $z \neq 0$. *Então* $\bar{x}$ *é um minimizador local estrito para o problema* (3.1).

Demonstração. Nocedal [35, Teorema 12.6]. □

A convergência do método de Lagrangeano aumentado clássico é discutida por Bertsekas em [3], onde o autor mostra, sob as hipóteses do Teorema 3.3, as condições sob as quais a convergência é garantida e também a velocidade de convergência. Aqui, faremos um resumo destes resultados.

O primeiro resultado fornece uma estimativa para os pontos minimizadores da função Lagrangeano aumentado (3.2). Assim, dado um conjunto limitado $\mathbb{C} \subset \mathbb{R}^m$ existe um escalar $\bar{\rho} \geq 0$ tal que para todo $\rho > \bar{\rho}$ e para todo $\lambda \in \mathbb{C}$ a função $\mathcal{L}(x, \lambda, \rho)$ tem um único minimizador $x(\lambda, \rho)$ em alguma bola aberta centrada em $\overline{x}$. Contudo, para algum escalar $M > 0$ segue que

$$\|x(\lambda, \rho) - \overline{x}\| \leq \frac{M \left\|\lambda - \overline{\lambda}\right\|}{\rho}, \quad \forall \rho > \overline{\rho}, \quad \lambda \in \mathbb{C} \tag{3.15}$$

e

$$\left\|\overline{\lambda}(\lambda, \rho) - \overline{\lambda}\right\| \leq \frac{M \left\|\lambda - \overline{\lambda}\right\|}{\rho}, \quad \forall \rho > \overline{\rho}, \quad \lambda \in \mathbb{C}. \tag{3.16}$$

O segundo resultado estabelece a velocidade de convergência do método, o qual diz que se o parâmetro de penalidade for mantido finito, $\rho^k \to \bar{\rho} > 0$, a convergência é linear. Por outro lado, se for permitido que o parâmetro de penalidade tenda a infinito, $\rho^k \to \infty$, então a convergência é superlinear.

Após apresentar os resultados anteriores, vamos mostrar o nosso principal resultado.

Teorema 3.4. *Suponha que* $\bar{x}$ *é uma solução local do problema* (3.1) *no qual a condição de qualificação LICQ e as condições do Teorema 3.3 são satisfeitas. Então, existe um valor limite* $\bar{\gamma} \geq 0$ *tal que para todo* $\gamma_i > \bar{\gamma}, i = 1, 2, \ldots, m$, $\bar{x}$ *é um minimizador da função Lagrangeano aumentado* $\mathcal{L}(x, \bar{\lambda}, \gamma)$, *dada em* (3.7).

Demonstração. Seja $\mathcal{L}$ a função Lagrangeano aumentado (3.7). Vamos provar esse resultado mostrando que $\bar{x} \in \Omega$ satisfaz as condições suficientes de segunda ordem para $\mathcal{L}(x, \bar{\lambda}, \gamma)$, isto é

$$\nabla_x \mathcal{L}(\bar{x}, \bar{\lambda}, \gamma) = 0 \ \text{ e } \ \nabla^2_{xx} \mathcal{L}(\bar{x}, \bar{\lambda}, \gamma) > 0.$$

Segue que,

$$\nabla_x \mathcal{L}(\bar{x}, \bar{\lambda}, \gamma) = \nabla f(\bar{x}) + \sum_{i=1}^{m} \bar{\lambda}_i \nabla h_i(\bar{x}) + \sum_{i=1}^{m} \gamma_i h_i(\bar{x}) \nabla h_i(\bar{x})$$

ou

$$\nabla_x \mathcal{L}(\bar{x}, \bar{\lambda}, \gamma) = \nabla_x \ell(\bar{x}, \bar{\lambda}) + \sum_{i=1}^{m} \gamma_i h_i(\bar{x}) \nabla h_i(\bar{x}).$$

Por hipótese, as condições de KKT em $\bar{x} \in \Omega$ são satisfeitas, então $\nabla_x \ell(\bar{x}, \bar{\lambda}) = 0$ e $h(\bar{x}) = 0$. Portanto, $\nabla_x \mathcal{L}(\bar{x}, \bar{\lambda}, \gamma) = 0$. Desse modo, $\bar{x}$ é também um ponto estacionário para a função (3.7).

Temos que,

$$\nabla^2_{xx} \mathcal{L}(\bar{x}, \bar{\lambda}, \gamma) = \nabla^2 f(\bar{x}) + \sum_{i=1}^{m} \bar{\lambda}_i \nabla^2 h_i(\bar{x}) + \sum_{i=1}^{m} \gamma_i \nabla h_i(\bar{x}) \nabla h_i(\bar{x})^T + \sum_{i=1}^{m} \gamma_i h_i(\bar{x}) \nabla^2 h_i(\bar{x})$$

ou

$$\nabla^2_{xx} \mathcal{L}(\bar{x}, \bar{\lambda}, \gamma) = \nabla^2_{xx} \ell(\bar{x}, \bar{\lambda}) + \sum_{i=1}^{m} \gamma_i \nabla h_i(\bar{x}) \nabla h_i(\bar{x})^T + \sum_{i=1}^{m} \gamma_i h_i(\bar{x}) \nabla^2 h_i(\bar{x}).$$

Como $\bar{x} \in \Omega$, segue que $h(\bar{x}) = 0$ e assim

$$\nabla^2_{xx} \mathcal{L}(\bar{x}, \bar{\lambda}, \gamma) = \nabla^2_{xx} \ell(\bar{x}, \bar{\lambda}) + \nabla h(\bar{x})^T diag(\gamma) \nabla h(\bar{x}).$$

Por hipótese, $\nabla^2_{xx} \ell(\bar{x}, \bar{\lambda}) > 0$. Portanto, pelo Lema 3.2 a prova está completa. □

Nota 3.5. *O Teorema 3.4 também é válido para a função Lagrangeano aumentado* (3.8)*. Contudo, uma adaptação é necessária na demonstração para que a Hessiana de* $\mathcal{L}$ *seja definida positiva. De fato, seja* $\mathcal{L}$ *a função* (3.8)*. Repetindo os passos da prova do Teorema 3.4, obtemos a seguinte expressão para a hessiana de* $\mathcal{L}$

$$\nabla^2_{xx} \mathcal{L}(\bar{x}, \bar{\lambda}, \omega) = \nabla^2_{xx} \ell(\bar{x}, \bar{\lambda}) + \nabla h(\bar{x})^T diag(\omega) \nabla h(\bar{x}). \tag{3.17}$$

Sendo $\omega_i = \dfrac{\lambda_i}{r_i}$*, tomamos* $r_i < 0$ *se* $\lambda_i < 0$ *e* $r_i > 0$*, caso contrário. Essa foi a estratégia usada no algoritmo de Lagrangeano aumentado para* $\mathcal{L}$ *dada pela função* (3.8)*.*

Nota 3.6. *Os resultados de convergência, discutidos anteriormente e apresentados em Bertsekas [3], podem em parte ser aplicados aos métodos que estamos propondo. Assim, considere as hipóteses do Teorema 3.3. Então, para um dado conjunto limitado* $\mathbb{C} \subset \mathbb{R}^m$ *existe um escalar* $\overline{\gamma} \geq 0$ *tal que para todo* $\gamma_i > \overline{\gamma}$*,* $i = 1, 2, \ldots, m$*, e para todo* $\lambda \in \mathbb{C}$ *a função* (3.5) *admite um único minimizador* $x(\lambda, \gamma)$ *em alguma bola aberta centrada em* $\overline{x}$*. Contudo, para algum escalar* $M > 0$ *temos*

$$\|x(\lambda, \gamma) - \overline{x}\| \leq \frac{M \left\|\lambda - \overline{\lambda}\right\|}{\gamma_m}, \quad \forall \lambda \in \mathbb{C} \text{ e } \gamma_m = \max\{\gamma_i, i = 1, \ldots, m\} \tag{3.18}$$

e

$$\left\|\overline{\lambda}(\lambda, \gamma) - \overline{\lambda}\right\| \leq \frac{M \left\|\lambda - \overline{\lambda}\right\|}{\gamma_m}, \quad \forall \lambda \in \mathbb{C} \text{ e } \gamma_m = \max\{\gamma_i, i = 1, \ldots, m\}. \tag{3.19}$$

Quanto ao resultado da velocidade de convergência não podemos garantir a validade com as mesmas hipóteses para o método clássico mostrado em Bertsekas [3], devido ao fato de que nos métodos propostos o parâmetro de penalidade é vetorial e dependente dos multiplicadores de Lagrange. Dessa forma, o estudo sobre a velocidade de convergência não será tratado nesse trabalho, ficando para trabalhos futuros.

A seguir apresentamos o algoritmo de Lagrangeano aumentado com as funções de penalidade propostas neste trabalho.

Algoritmo 3.1. *LA com novas funções de penalidade*

Dados: $x^0 \in \mathbb{R}^n$, $\lambda^0 \in \mathbb{R}^m$, $\gamma^0 \in \mathbb{R}^m_{++}$, $\omega^0 \in \mathbb{R}^m_{++}$

$k = 0$

REPITA *enquanto o critério de parada não for satisfeito*

1. *Determinar*

   $x^{k+1} \in$ *argmin* $\left\{\mathcal{L}(x, \lambda^k, \gamma^k) \quad ou \quad \mathcal{L}(x, \lambda^k, \omega^k) : x \in \mathbb{R}^n\right\}$

2. *Atualizar o multiplicador de Lagrange*

   $\lambda_i^{k+1} = \lambda_i^k + \gamma_i^k h_i(x^{k+1}), \quad i = 1, \ldots, m \qquad$ (*para a função 3.7*)

   ou

   $\lambda_i^{k+1} = \lambda_i^k + \omega_i^k h_i(x^{k+1}), \quad i = 1, \ldots, m \qquad$ (*para a função 3.8*)

3. *Atualizar o parâmetro de penalidade, componente a componente*

   SE $\left\|h(x^{k+1})\right\| \geq \delta \left\|h(x^k)\right\|$

   Atualize $\gamma_i^{k+1}, \quad i = 1, \ldots, m \qquad$ (*para a função 3.7*)

   ou

   Atualize $\omega_i^{k+1}, \quad i = 1, \ldots, m \qquad$ (*para a função 3.8*)

   SENÃO

   $\gamma_i^{k+1} = \gamma_i^k$

   ou

   $\omega_i^{k+1} = \omega_i^k$

   FIM

$k = k + 1$

No passo 1 do Algoritmo 3.1 a função $\mathcal{L}$ refere-se a função Lagrangeano aumentado (3.7) ou (3.8). Detalhes com relação a escolha e atualização dos parâmetros serão fornecidos na seções seguintes, após o Algoritmo 3.2.

Para comparar a eficiência dos métodos propostos, também implementamos o método clássico de Hestenes [18] e Powell [40].

Algoritmo 3.2. *LA com penalidade clássica*

Dados: $x^0 \in \mathbb{R}^n$, $\delta \in (0,1)$, $\lambda^0 \in \mathbb{R}^m$, $\rho^0 \in \mathbb{R}_{++}$

$k = 0$

REPITA *enquanto o critério de parada não for satisfeito*

1. *Determinar*

$$x^{k+1} \in \operatorname{argmin} \left\{ f(x) + \sum_{i=1}^{m} \left[\frac{\rho^k}{2} \left(h_i(x)\right)^2 + \lambda_i^k h_i(x) \right] \right\}$$

2. *Atualizar o multiplicador de Lagrange*

$$\lambda_i^{k+1} = \lambda_i^k + \rho^k h_i(x^{k+1}), \quad i = 1, \ldots, m$$

3. *Atualizar o parâmetro de penalidade*

SE $\left\|h(x^{k+1})\right\| \geq \delta \left\|h(x^k)\right\|$

Faça $\rho^{k+1} > \rho^k$

SENÃO

Faça $\rho^{k+1} = \rho^k$

FIM

$k = k + 1$

3.3 Aspectos da implementação

Apresentamos nessa seção aspectos da implementação dos Algoritmos 3.1 e 3.2, descritos na seção anterior, bem como o método utilizado para a resolução do subproblema interno, apresentado no *passo 1* desses algoritmos.

3.3.1 Passo Interno

Denominamos de passo interno, o *passo 1* dos Algoritmos 3.1 e 3.2. Nesse passo o objetivo é determinar a cada iteração, k, o novo ponto, x^{k+1}, que é o minimizador do subproblema irrestrito, cuja função objetivo é dada pela função Lagrangeano aumentado.

Diversos são os métodos de otimização utilizados na resolução dos subproblemas irrestritos. Diante disso e considerando que, o nosso principal objetivo é constituir novas metodologias, e nesse caso especificamente métodos de Lagrangeano aumentado que possam ser aplicados na resolução de problemas de confiabilidade estrutural, optamos por não focar nossa atenção no estabelecimento de uma única metodologia para a minimização dos subproblemas irrestritos. Assim, uma rotina interna do Matlab denominada *fminunc* que determina o minimizador de uma função escalar de várias variáveis a partir de uma estimativa inicial foi empregada.

Na utilização da *fminunc*, se o gradiente analítico é fornecido pelo usuário durante o processo iterativo, então a rotina utiliza o método BFGS (método quasi-newton) juntamente com um procedimento de busca linear. Essa foi a situação considerada em nossa

implementação. Caso contrário, a rotina emprega o método do Gradiente Conjugado na minimização de problemas irrestritos.

Na rotina *fminunc*, estabelecemos um limite máximo para o número de iterações ($maxit$=500) e uma tolerância de 10^{-8} para a diferença entre os iterandos e seus respectivos valores funcionais.

3.3.2 Passo Externo

Os Algoritmos 3.1 e 3.2 consistem em, dado um ponto inicial, calcular um novo iterando por meio do passo interno e, em seguida, atualizar os parâmetros, até que um ponto estacionário seja obtido. As etapas relativas às atualizações dos parâmetros (*passos* 2 e 3 dos Algoritmos 3.1 e 3.2) consistem o que chamamos de passo externo.

Deste modo, o maior esforço computacional dos métodos de Lagrangeano aumentado está associado com a solução do subproblema irrestrito (passo interno), ou seja, na minimização da função lagrangeano aumentado, pois no passo externo, é realizada somente a atualização do parâmetro de penalidade e dos multiplicadores de Lagrange, conforme descrevemos a seguir.

Atualização dos multiplicadores de Lagrange

No Algoritmo clássico 3.2, Hestenes [18] e Powell [40] atualizaram o multiplicador de Lagrange, λ, forçando satisfazer as condições de KKT. Na iteração k, são conhecidos λ^k e ρ^k e, o algoritmo determina x^{k+1}. Se $\lambda_i^{k+1} = \lambda_i^k + \rho^k h_i(x^{k+1})$, então

$$\nabla_x \mathcal{L}(x^{k+1}, \lambda^{k+1}, \rho^k) = \nabla_x \ell(x^{k+1}, \lambda^{k+1}) = 0 \tag{3.20}$$

desde que x^{k+1} seja uma solução do subproblema irrestrito definido no *passo 1* do Algoritmo 3.2.

Vamos usar esse fato para justificar a escolha de λ^{k+1} no Algoritmo 3.1. Apresentamos a seguir somente o caso da função (3.7), pois o processo é análogo para a função (3.8).

De fato, tomando a derivada da função (3.7) com relação a x e mantendo λ^k e γ^k fixos, obtemos

$$\nabla_x \mathcal{L}(x, \lambda^k, \gamma^k) = \nabla f(x) + \sum_{i=1}^{m} \left(\lambda_i^k + \gamma_i^k h_i(x)\right) \nabla h_i(x).$$

E, o gradiente avaliado em x^{k+1} resulta em

$$\nabla_x \mathcal{L}(x^{k+1}, \lambda^k, \gamma^k) = \nabla f(x^{k+1}) + \sum_{i=1}^{m} \left(\lambda_i^k + \gamma_i^k h_i(x^{k+1})\right) \nabla h_i(x^{k+1}).$$

Portanto, se $\lambda_i^{k+1} = \lambda_i^k + \gamma_i^k h_i(x^{k+1})$ e x^{k+1} é a solução do subproblema irrestrito definido no *passo 1* do Algoritmo 3.1, então $\nabla_x \mathcal{L}(x^{k+1}, \lambda^{k+1}, \gamma^k) = \nabla_x \ell(x^{k+1}, \lambda^{k+1}) = 0$.

Isso significa que a cada iteração k, do Algoritmo 3.1, o ponto gerado, x^{k+1}, satisfaz uma das condições de KKT, ou seja, $\nabla_x \ell(x^{k+1}, \lambda^{k+1}) = 0$. Já a condição de KKT referente à viabilidade, $h(x) = 0$, é forçada ser satisfeita pela atualização do parâmetro de penalidade.

Atualização do parâmetro de penalidade

Nos métodos de Lagrangeano aumentado o procedimento usado para atualizar o parâmetro penalidade é baseado na medida de inviabilidade do problema, pois não há necessidade de penalizar em toda iteração. Deste modo, se $\left\|h(x^{k+1})\right\| \geq \delta \left\|h(x^k)\right\|$ com $\delta \in (0,1)$, é porque não houve ganho significativo na viabilidade e então aumenta-se a penalidade. Em nossa implementação escolhemos $\delta = 0,1$.

No Algoritmo 3.2 consideramos um aumento no valor do parâmetro de penalidade fazendo $\rho^{k+1} = 2\rho^k$ e escolhemos os seguintes valores iniciais para ρ e λ:

$$\rho^0 = 1 \quad \text{e} \quad \lambda_i^0 = 1, \ \text{com} \ i = 1, \ldots, m$$

sendo m o número de restrições de igualdade do problema.

Para a atualização dos parâmetros de penalidade γ e ω, que compõe o *passo 3* do Algoritmo 3.1, desenvolvemos heurísticas que são apresentadas a seguir:

Heurística 3.1. *Atualização do parâmetro de penalidade* γ^{k+1}

SE $\left\|h(x^{k+1})\right\| \geq \delta \left\|h(x^k)\right\|$

$\bar{\lambda} = \max\left\{\lambda_{\min}, \min\left\{\lambda_{\max}, \lambda^{k+1}\right\}\right\}$

SE $\left|\bar{\lambda}_i\right| < 1$

$r_i^{k+1} = 10^{-2}\left(\bar{\lambda}_i\right)^2$

SENÃO

$r_i^{k+1} = 0,5 \cdot r_i^k$

FIM

$\gamma_i^{k+1} = \dfrac{\left(\bar{\lambda}_i\right)^2}{r_i^{k+1}}$

SE $\gamma_i^{k+1} = 0$

$\gamma_i^{k+1} = \max\limits_i \{\gamma_i\}$

FIM

FIM

Heurística 3.2. *Atualização do parâmetro de penalidade* ω^{k+1}

SE $\left\|h(x^{k+1})\right\| \geq \delta \left\|h(x^{k})\right\|$

$\bar{\lambda} = \max\left\{\lambda_{\min}, \min\left\{\lambda_{\max}, \lambda^{k+1}\right\}\right\}$

SE $\left|\bar{\lambda}_i\right| < 1$

$r_i^{k+1} = 10^{-2}\left|\bar{\lambda}_i\right|$

SENÃO

$r_i^{k+1} = 0,5 \cdot r_i^{k}$

FIM

$\omega_i^{k+1} = \dfrac{\left|\bar{\lambda}_i\right|}{r_i^{k+1}}$

SE $\omega_i^{k+1} = 0$

$\omega_i^{k+1} = \max\limits_{i}\{\omega_i\}$

FIM

FIM

Na Figura 3.1, apresentada anteriormente, destacamos a influência de λ no comportamento das novas penalidades, $\theta(\sqrt{\gamma}y) = \lambda^2 y^2$ e $\omega\theta(y) = \lambda y^2$, para o caso unidimensional, sendo θ uma função quadrática real definida em (3.4). Observamos que a penalidade $\theta(\sqrt{\gamma}y)$ tem a característica de penalizar mais que as outras duas, $\theta(y)$ e $\omega\theta(y)$, quando $|\lambda| > 1$ e de penalizar menos, quando $|\lambda| < 1$.

Se a penalização for muito rigorosa os subproblemas gerados pelo algoritmo podem ficar difícies de resolver, por tornarem-se mal condicionados. Por outro lado, se os pontos gerados estão muito inviáveis e a penalização é muito branda, os subproblemas gerados podem ser fáceis de resolver e o algoritmo pode gastar muitas iterações na resolução do problema. Diante disso e considerando o aspecto geométrico das penalidades é que elaboramos as Heurísticas 3.1 e 3.2 para a atualização dos parâmetros de penalidade γ e ω.

A Figura 3.2 exibe o comportamento das heurísticas adotadas para a atualização dos parâmetros de penalidade γ e ω. Desde que nas novas metodologias os parâmetros de penalidade são dependentes dos multiplicadores de Lagrange, plotamos um caso particular unidimensional considerando $\lambda = 1/2$. Assim, se $r = 1$ tem-se os seguintes parâmetros de penalidade: $\gamma = 1/4$ para $\theta(\sqrt{\gamma}y)$, $\omega = 1/2$ para $\omega\theta(y)$ e $\rho = 1$ para $\theta(y)$, o qual não depende de λ. Note na geometria da figura uma penalização muito branda para as penalidades das novas metodologias comparadas com a clássica. Por isso, decidimos através das heurísticas adotadas ser mais rigorosos aumentando a penalização, como mostrado nos três últimos gráficos da Figura 3.2. Obviamente o caso em que $\rho = r = \lambda = 1$ tem-se $\gamma = \omega = 1$ e as três metodologias conduzem à mesma penalidade $\theta(y) = y^2$.

No caso em que o parâmetro de penalidade tende ao infinito, é possível que a penalização seja muito rigorosa, sendo assim, procuramos contornar esse inconveniente

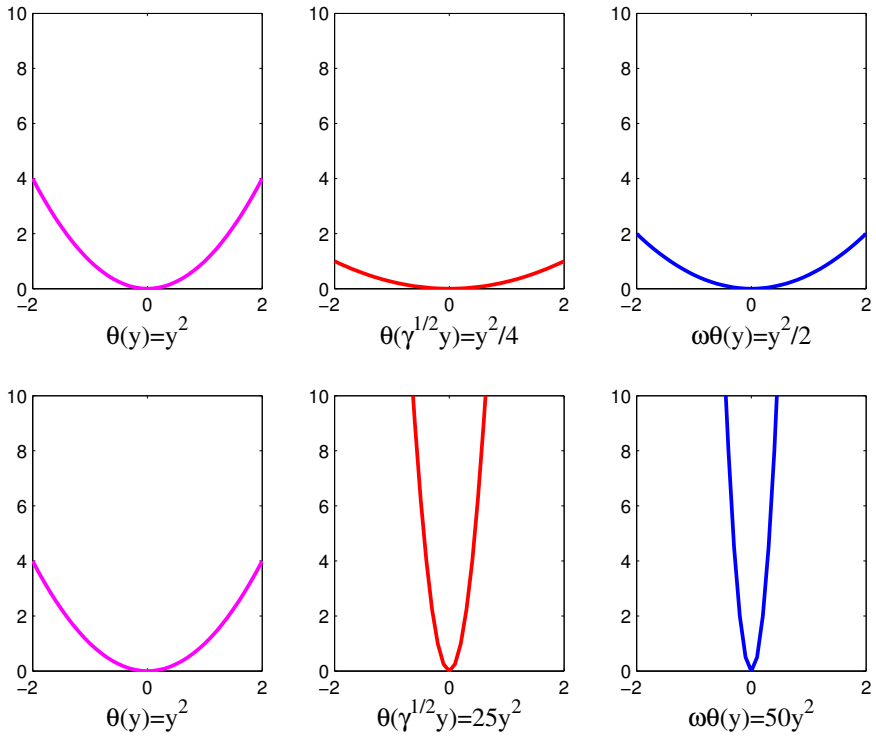


Figura 3.2: Penalidades $\theta(y)$, $\theta(\sqrt{\gamma}y)$ e $\omega\theta(y)$ para dois valores distintos de r.

limitando os possíveis valores para o multiplicador de Lagrange, fazendo

$$\bar{\lambda} = \max\left\{\lambda_{\min}, \min\left\{\lambda_{\max}, \lambda^{k+1}\right\}\right\},$$

em que $\lambda_{\min} = -10^5$ e $\lambda_{\max} = 10^5$.

Para o Algoritmo 3.1, escolhemos os seguintes valores iniciais para os parâmetros:

$$\lambda_i^0 = 1, \;\; r_i^0 = 1, \;\; \gamma_i^0 = 10 \;\text{ e }\; \omega_i^0 = 10, \;\; \text{com } \; i = 1, \ldots, m$$

onde m é o número de restrições de igualdade do problema.

3.3.3 Critério de parada

Os Algoritmos 3.1 e 3.2 são interrompidos quando encontram um ponto cujo valor da medida de estacionaridade, definida por meio das condições de KKT, é suficientemente pequeno. Dessa forma, estabelecemos as seguintes condições

$$\left\| \nabla f(x^k) + \sum_{i=1}^{m} \lambda_i^k \nabla h_i(x^k) \right\| \leq \varepsilon \qquad \text{e} \qquad \left\| h(x^k) \right\| \leq \varepsilon, \tag{3.21}$$

em que $\|\cdot\|$ é a norma euclidiana e $\varepsilon = 10^{-4}$ é a tolerância estabelecida. Se essas condições não forem atendidas até que o algoritmo externo atinja 100 iterações ou 1 hora de tempo de CPU, as execuções dos Algoritmos 3.1 e 3.2 também são interrompidas sem obter a

solução do problema.

3.3.4 Análise dos resultados numéricos

Essa seção destina-se a apresentação e análise dos resultados dos testes numéricos realizados com 124 problemas da coleção CUTEr para duas variantes do Algoritmo 3.1, de acordo com a função Lagrangeano aumentado (3.7 e 3.8), e para o Algoritmo 3.2, sendo nomeados da seguinte forma:

- Algoritmo LAPM, refere-se ao método de Lagrangeano aumentado utilizando a função (3.7), descrito pelo Algoritmo 3.1;
- Algoritmo LAPB, refere-se ao método de Lagrangeano aumentado utilizando a função (3.8), descrito pelo Algoritmo 3.1;
- Algoritmo LAPC, refere-se ao método de Lagrangeano aumentado utilizando a função (3.2), descrito pelo Algoritmo 3.2.

Os algoritmos foram implementados em Matlab 7.8.0 (R2009a) para Linux. Os testes foram executados em um processador Intel(R) Core(TM)2 Duo CPU T5870 2,00GHz e 3,00GB de memória RAM.

No Apêndice A estão dispostos os nomes dos problemas selecionados e seus respectivos números de variáveis (n) e restrições de igualdade (m). Consideramos problemas de dimensões variadas, com pelo menos 2 variáveis e 1 restrição e no máximo 1005 variáveis e 998 restrições. Os pontos iniciais considerados nos testes foram aqueles definidos na coleção CUTEr.

Durante os testes iniciais percebemos a necessidade de interromper a execução dos algoritmos considerando quatro critérios distintos. Dessa forma, definimos um parâmetro de saída para as rotinas escritas em Matlab, que pode assumir os seguintes valores:

0: o critério de parada (3.21) foi atendido;
1: número máximo de iterações do algoritmo, no passo externo (it.ext=100), foi alcançado;
2: o tempo limite (1 hora) foi ultrapassado;
3: um valor não numérico (NaN) foi encontrado.

Na Tabela 3.1 apresentamos os resultados dos 124 problemas testados considerando os valores do parâmetro de saída. Assim, observando a primeira linha verifica-se que o critério de parada (3.21) foi atendido para 100 problemas, quando executados os algoritmos LAPC e LAPM, e para o algoritmo LAPB o mesmo critério foi atendido para 101 problemas. Esses resultados indicam que os Algoritmos LAPC e LAPM encontraram a solução em $80,65\%$ dos problemas, enquanto que, o algoritmo LAPB obteve sucesso em $81,45\%$ dos problemas.

Tabela 3.1: Número de problemas de acordo com o parâmetro de saída.

Saída	LAPC	LAPM	LAPB
0	100	100	101
1	22	23	22
2	1	1	1
3	1	0	0

No Apêndice B são apresentados detalhadamente os resultados dos testes com os problemas da coleção CUTEr para cada um dos algoritmos, LAPC, LAPB e LAPM.

Observamos que todos os algoritmos falharam na resolução dos seguintes problemas: DIXCHLNG, EIGENB, HEART6, HYDCAR6, HYDCAR20, LUKVLE6 (99,49), LUKVLE6 (999,499), LUKVLE13 (49,30), LUKVLE13 (98,64), LUKVLE13 (998,664), LUKVLE15 (999,747), ORTHRDM2 (53,25), ORTHRDM2 (203,100), ORTHREGA (133, 64), ORTHREGC (505,250), ORTHREGC (1005,500), ORTHREGD, ORTHRGDM (103, 50), ORTHRGDM (155,76) e POWELLBS. O primeiro número entre parênteses em frente ao nome do problema indica o número de variáveis e, o segundo, o número de restrições de igualdade. Para o problema LUKVLE15 (999,747), o tempo computacional estabelecido (1 hora) foi ultrapasado e, para os demais problemas, o número máximo de iterações no passo externo ($k = 100$) foi atingido.

O algoritmo LAPC também falhou ao tentar resolver o problema MSS1, pois encontrou um valor não numérico (NaN) durante a sua execução. Além disso, para os problemas COOLHANS, EIGENC (30,30), EIGENC (462,462), o algoritmo LAPC ultrapassou o número máximo de iterações no passo externo. Considerando esse mesmo critério, o algoritmo LAPM não obteve a solução dos problemas EIGENC (462,462), EIGENC2 (462,231), LUKVLE1(1000,998) e LUKVLE6(9,4), e o algoritmo LAPB não obteve a solução dos problemas COOLHANS, EIGENB2 (6,3) e LUKVLE1(1000,998).

A fim de facilitar a análise dos resultados, construímos tabelas com informações específicas. Sendo assim, as Tabelas 3.2, 3.3 e 3.4 referem-se ao número de vezes que cada algoritmo, LAPC, LAPB e LAPM, apresentou melhor desempenho levando em consideração o número de iterações no passo externo (*it.ext*), o número de iterações do algoritmo no passo interno (*it.int*), o número de avaliações da função (*fun*) e de derivadas (*grad*) e o tempo computacional (*tempo*). Os valores relacionados ao *empate* mostram a quantidade de problemas cujos desempenhos coincidiram.

A Tabela 3.2 apresenta uma comparação entre os algoritmos LAPC e LAPM. Nesta comparação, excluímos 21 problemas que não foram resolvidos por ambos os algoritmos.

Os resultados apresentados na Tabela 3.2 indicam que o algoritmo LAPM foi mais eficiente que o algoritmo LAPC para os problemas testados, pois apresentou melhores resultados com relação aos cinco critérios mencionados, *it.ext*, *it.int*, *fun*, *grad* e

Tabela 3.2: Comparação do desempenho entre os algoritmos LAPC e LAPM para os problemas da coleção CUTEr.

Método	it.ext	it.int	fun	grad	tempo
LAPC	13	35	40	40	2
LAPM	60	66	59	59	101
empate	30	2	4	4	0

tempo computacional, em $58,25\%$, $64,08\%$, $57,28\%$, $57,28\%$ e $98,06\%$ dos problemas, respectivamente.

Na Tabela 3.3 apresentamos uma comparação entre os algoritmos LAPC e LAPB. Para a construção dessa tabela consideramos 103 dos 124 problemas, que foram resolvidos por pelo menos um dos dois algoritmos.

Tabela 3.3: Comparação do desempenho entre os algoritmos LAPC e LAPB para os problemas da coleção CUTEr.

Método	it.ext	it.int	fun	grad	tempo
LAPC	16	40	44	44	1
LAPB	56	61	55	55	102
empate	31	2	4	4	0

Os resultados da Tabela 3.3 indicam que o algoritmo LAPB superou o algoritmo LAPC em: $54,37\%$ dos problemas com relação ao número de iterações externas, em $59,22\%$ dos problemas com relação ao número de iterações internas e em $53,40\%$ dos problemas para o número de avaliações de funções e de derivadas.

Além disso, verifica-se que os algoritmos LAPM e LAPB quase na totalidade dos problemas, apresentaram menor tempo computacional durante suas execuções, com relação ao algoritmo LAPC. De qualquer modo, é importante ressaltar que o tempo computacional não pode ser considerado como uma única medida decisiva na escolha do algoritmo mais eficiente. Porém, observando os demais critérios em conjunto, tanto para o algoritmo LAPB como para o LAPM, nota-se que os algoritmos modernos foram mais eficientes com relação a quantidade de iterações, de avaliações de funções e de derivadas, e esse desempenho fez com que o tempo computacional dispendido também fosse menor, levando-nos a crer que os métodos modernos foram, de uma forma geral, mais eficientes que o clássico na resolução dos problemas testados.

Para a construção da Tabela 3.4, que apresenta uma comparação entre os algoritmos LAPM e LAPB, também desconsideramos 21 problemas que não foram resolvidos por nenhum dos dois algoritmos. Nessa análise percebemos que, ao contrário do que aconteceu nas duas comparações anteriores, houve empate nos resultados dos melhores desempenhos para grande parte dos problemas. Assim, para os 103 problemas solucionados pelos algoritmos, houve empate em 73, considerando o número de iterações no passo

Tabela 3.4: Comparação do desempenho entre os algoritmos LAPM e LAPB para os problemas da coleção CUTEr.

Método	it.ext	it.int	fun	grad	tempo
LAPM	27	33	34	34	39
LAPB	3	10	8	8	26
empate	73	60	61	61	38

externo, em 60 problemas, considerando o número de iterações no passo interno, em 61 problemas, considerando o número de avaliação de funções e de derivadas e em 38 problemas com relação ao tempo computacional. Para a maior parte dos problemas restantes (que não aparecem no empate), o algoritmo LAPM foi mais eficiente do que o algoritmo LAPB com relação à todos os critérios considerados.

Conclusão dos resultados numéricos

Nessa seção apresentamos dois novos métodos de Lagrangeano aumentado que utilizam parâmetros de penalidade vetoriais e que são dependentes dos multiplicadores de Lagrange. Essas são as principais diferenças dessas penalidades com relação à penalidade clássica, proposta por Hestenes [18] e Powell [40].

Mostramos que, tomando um certo valor para r, os métodos de Lagrangeano aumentado propostos coincidem com o método clássico existente. Assim, procuramos desenvolver heurísticas de atualização que apresentassem um desempenho satisfatório para a resolução dos problemas de minimização com restrições de igualdade.

Os resultados que acabamos de descrever, nessa seção, compravam que as novas metodologias são promissoras e mais eficientes do que o método clássico, pois apresentaram resultados superiores com relação aos critérios considerados.

Diante disso, vamos discutir no próximo capítulo a aplicação dos métodos de Lagrangeano aumentado, que propomos, para a resolução dos problemas de confiabilidade estrutural e, em seguida, comparar os resultados aos obtidos pelos algoritmos HLRF, iHLRF e nHLRF, descritos no Capítulo 2.

Capítulo 4

Aplicação dos novos algoritmos de otimização ao problema de confiabilidade

No Capítulo 3 descrevemos os métodos de Lagrangeano aumentado com as duas novas funções de penalidade e com a função de penalidade clássica, e analisamos o desempenho desses métodos na resolução de 124 problemas de otimização com m restrições de igualdade, selecionados da coleção CUTEr. Nosso objetivo agora é investigar o desempenho desses métodos e dos algoritmos HLRF, iHLRF e nHLRF na resolução de problemas de confiabilidade estrutural.

4.1 Lagrangeano aumentado com as novas funções de penalidade aplicado ao problema de confiabilidade estrutural

Nessa seção vamos discutir os aspectos teóricos da aplicação dos dois novos métodos de Lagrangeano aumentado ao problema de confiabilidade estrutural. Sendo assim, considerando o formato do problema de confiabilidade estrutural, dado por (1.15), a função Langrangeano aumentado com penalidade clássica, proposta por Hestenes [18] e Powell [40], é definida como

$$\mathcal{L}(\mathbf{y}, \lambda, \rho) = f(\mathbf{y}) + \lambda h(\mathbf{y}) + \frac{\rho}{2} h(\mathbf{y})^2 \tag{4.1}$$

em que $\mathbf{y} \in \mathbb{R}^n$, $\lambda \in \mathbb{R}$ é o multiplicador de Lagrange associado a restrição de igualdade e $\rho > 0$ é o parâmetro de penalidade.

Já as funções lagrangeano aumentado com as novas funções de penalidade, apli-

cadas na resolução do problema (1.15), são definidas como

$$(\mathbf{y}, \lambda, \gamma) \in \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}_{++} \rightarrow \mathcal{L}(\mathbf{y}, \lambda, \gamma) = f(\mathbf{y}) + \lambda h(\mathbf{y}) + \frac{\gamma}{2} h(\mathbf{y})^2 \tag{4.2}$$

e

$$(\mathbf{y}, \lambda, \omega) \in \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}_{++} \rightarrow \mathcal{L}(\mathbf{y}, \lambda, \omega) = f(x) + \lambda h(\mathbf{y}) + \frac{\omega}{2} h(\mathbf{y})^2, \tag{4.3}$$

em que γ e ω são os parâmetros de penalidade definidos, respectivamente, por

$$\gamma = \frac{\lambda^2}{r}, \quad \text{com} \quad r > 0 \tag{4.4}$$

e

$$\omega = \frac{\lambda}{r}, \quad \text{com} \quad r \neq 0. \tag{4.5}$$

Em geral, os problemas de confiabilidade estrutural possuem dimensão menor quando comparados aos problemas da coleção CUTEr, isto é, menor número de variáveis e apenas uma restrição de igualdade. Diante desse fato e observando os resultados de alguns testes numéricos, optamos por não utilizar na resolução de tais problemas as heurísticas (3.1) e (3.2), desenvolvidas para a atualização dos parâmetros de penalidade γ e ω, apresentadas no capítulo anterior. Os critérios para atualização dos parâmetros de penalidade são mostrados mais adiante, na Seção 4.2.1.

Conforme mencionamos no Capítulo 1, qualquer algoritmo de otimização pode ser inserido no passo *4* do Algoritmo 1.1. Deste modo, apresentamos a seguir os algoritmos referentes aos métodos de Lagrangeano aumentado, usando as funções de penalidade (4.2) e (4.3) e a função de penalidade clássica (4.1), aplicados aos problemas de confiabilidade estrutural.

Quando acoplamos o algoritmo LAPC, ou LAPM, ou ainda, LAPB, no passo *4* do Algoritmo 1.1, primeiramente executa-se o passo interno dos algoritmos de Lagrangeano aumentado, até que o critério de parada estabelecido na rotina *fminunc* seja atendido, retornando um novo iterando, $\mathbf{y}^{k+1}$, no espaço reduzido. Em seguida, o passo externo é executado apenas uma vez, para que sejam atualizados o multiplicador de Lagrange e o parâmetro de penalidade. Na sequência, retoma-se o passo *5* do Algoritmo 1.1 até que o critério de parada seja satisfeito.

Já os algoritmos HLRF, iHLRF e nHLRF, quando acoplados no passo *4* do Algoritmo 1.1, realizam apenas um passo em cada iteração do FORM, até que o critério de parada seja atendido.

Os aspectos considerados em nossa implementação, como por exemplo o critério de parada e os valores dos parâmetros, são fornecidos mais adiante, na Seção 4.2.1.

Algoritmo 4.1. *LAPM e LAPB aplicados ao problema de confiabilidade estrutural*

Dados: $\mathbf{y}^k \in \mathbb{R}^n$, $\lambda^k \in \mathbb{R}$, $\gamma^k \in \mathbb{R}_{++}$, $\omega^k \in \mathbb{R}_{++}$

1. *Determinar*
   $\mathbf{y}^{k+1} \in \operatorname{argmin} \left\{ \mathcal{L}(\mathbf{y}, \lambda^k, \gamma^k) \quad ou \quad \mathcal{L}(\mathbf{y}, \lambda^k, \omega^k) : \mathbf{y} \in \mathbb{R}^n \right\}$
2. *Atualizar o multiplicador de Lagrange*
   $\lambda^{k+1} = \lambda^k + \gamma^k h(\mathbf{y}^{k+1})$ (*para a função 4.2*)
   ou
   $\lambda^{k+1} = \lambda^k + \omega^k h(\mathbf{y}^{k+1})$ (*para a função 4.3*)
3. *Atualizar o parâmetro de penalidade*
   SE $\left\| h(\mathbf{y}^{k+1}) \right\| \geq \delta \left\| h(\mathbf{y}^k) \right\|$
   Faça $r^{k+1} < r^k$
   SENÃO
   Faça $r^{k+1} = r^k$
   FIM
   $\gamma^{k+1} = \dfrac{\left(\lambda^{k+1}\right)^2}{r^{k+1}}$ (*para a função 4.2*)
   ou
   $\omega^{k+1} = \dfrac{\left|\lambda^{k+1}\right|}{r^{k+1}}$ (*para a função 4.3*)

Algoritmo 4.2. *LAPC aplicado ao problema de confiabilidade estrutural*

Dados: $\mathbf{y}^k \in \mathbb{R}^n$, $\delta \in (0,1)$, $\lambda^k \in \mathbb{R}^m$, $\rho^k \in \mathbb{R}_{++}$

1. *Determinar*
   $\mathbf{y}^{k+1} \in \operatorname{argmin} \left\{ f(\mathbf{y}) + \lambda h(\mathbf{y}) + \dfrac{\rho}{2} h(\mathbf{y})^2 \right\}$
2. *Atualizar o multiplicador de Lagrange*
   $\lambda^{k+1} = \lambda^k + \rho^k h(\mathbf{y}^{k+1})$
3. *Atualizar o parâmetro de penalidade*
   SE $\left\| h(\mathbf{y}^{k+1}) \right\| \geq \delta \left\| h(\mathbf{y}^k) \right\|$
   Faça $\rho^{k+1} > \rho^k$
   SENÃO
   Faça $\rho^{k+1} = \rho^k$
   FIM

Na próxima seção, alguns problemas numéricos clássicos de confiabilidade são apresentados. Estes problemas são utilizados para comparar a robustez e a eficiência dos métodos de Lagrangeano aumentado clássicos e os propostos, e dos algoritmos HLRF, iHLRF e nHLRF.

4.2 Problemas numéricos

Selecionamos 25 problemas da literatura que são apresentados na sequência. Com exceção dos problemas 22 e 23, todas as variáveis envolvidas são estatisticamente independentes. Para cada problema, as distribuições de probabilidade e os momentos (média e desvio padrão) das variáveis de projeto são apresentados, assim como a função desempenho $h(\mathbf{X})$. Para os problemas 1 a 6 todas as variáveis seguem a distribuição Gaussiana (ou normal) e, para esses problemas, apenas as funções desempenho são apresentadas.

Problema 1 (Borri e Speranzini [5]): $h(\mathbf{X}) = 0,1(X_1 - X_2)^2 - \dfrac{(X_1 + X_2)}{\sqrt{2}} + 2,5.$

Problema 2 (Borri e Speranzini [5]): $h(\mathbf{X}) = -0,5(X_1 - X_2)^2 - \dfrac{(X_1 + X_2)}{\sqrt{2}} + 3.$

Problema 3 (Grooteman [15]): $h(\mathbf{X}) = 2 - X_2 - 0,1X_1^2 + 0,06X_1^3.$

Problema 4 (Grooteman [15]): $h(\mathbf{X}) = 3 - X_2 + 256X_1^4.$

Problema 5 (Wang e Grandhi [55]): $h(\mathbf{X}) = 1 + \dfrac{(X_1 + X_2)^2}{4} - 4(X_1 - X_2)^2.$

Problema 6 (Grooteman [15]): $h(\mathbf{X}) = 2 + 0,015 \sum_{i=1}^{9} X_i^2 - X_{10}.$

Problema 7 (Santosh *et al.* [50]): A função desempenho é $h(\mathbf{X}) = X_1^3 + X_2^3 - 18$, e as variáveis de projeto são Gaussianas e estatisticamente independentes, com parâmetros $\mu_{X_1} = \mu_{X_2} = 10$ e $\sigma_{X_1} = \sigma_{X_2} = 5$.

Problema 8 (Santosh *et al.* [50]): $h(\mathbf{X}) = X_1^3 + X_2^3 - 18$, as variáveis seguem a distribuição Gaussiana com valores médios $\mu_{X_1} = 10$ e $\mu_{X_2} = 9,9$, e desvios padrão $\sigma_{X_1} = \sigma_{X_2} = 5$.

Problema 9 (Grooteman [15]): O relacionamento entre as variáveis consideradas neste problema é dado pela função $h(\mathbf{X}) = 2,5 - 0,2357(X_1 - X_2) + +0,0046(X_1 + X_2 - 20)^4$, onde X_1 e X_2 são Gaussianas com valores médios iguais a 10 e desvios padrão iguais a 3.

Problema 10 (Santosh *et al.* [50]): As variáveis aleatórias consideradas neste problema possuem distribuição Gaussiana com valores médios $\mu_{X_1} = 10$ e $\mu_{X_2} = 9,9$, e desvios padrão $\sigma_{X_1} = \sigma_{X_2} = 5$. A função desempenho é $h(\mathbf{X}) = X_1^3 + X_2^3 - 67,5$.

Problema 11 (Grooteman [15]): A função desempenho é dada por

$$h(\mathbf{X}) = X_1 X_2 - 146,14$$

em que X_1 e X_2 são variáveis Gaussianas com média $\mu_{X_1} = 78064,4$ e $\mu_{X_2} = 0,0104$, e desvios padrão $\sigma_{X_1} = 11709,7$ e $\sigma_{X_2} = 0,00156$.

Problema 12 (Wang e Grandhi [54]): A função desempenho considerada neste problema

é $h(\mathbf{X}) = 2,2257 - \dfrac{0,025\sqrt{2}}{27}(X_1 + X_2)^3 + 0,2357(X_1 - X_2)$, onde X_1 e X_2 seguem a distribuição Gaussiana, com valores médios iguais a 10 e desvios padrão iguais a 3.

Problema 13 (Santosh *et al.* [50]): A função desempenho é $h(\mathbf{X}) = X_1X_2 - 2000X_3$, onde as variáveis X_1 e X_2 são Gaussianas com valores médios $0,32$ e 1400000, e desvios padrão $0,032$ e 70000, respectivamente. A variável X_3 é uma variável lognormal com média 100 e desvio padrão 40.

Problema 14 (Haldar e Mahadevan [16]): Nesse problema X_1 e X_2 são variáveis lognormais com médias 38 e 54, e desvios padrão $3,8$ e $2,7$, respectivamente. A função desempenho é dada por $h(\mathbf{X}) = X_1X_2 - 1140$.

Problema 15 (Mahadevan e Pan [28]): Consideramos nesse problema uma função desempenho linear dada por, $h(\mathbf{X}) = X_1 + 2X_2 + 3X_3 + X_4 - 5X_5 - 5X_6$, cujas variáveis são lognormais com médias $\mu_{X_i} = 120$, para $i = 1, \ldots, 4$, $\mu_{X_5} = 50$ e $\mu_{X_6} = 40$, e desvios padrão $\sigma_{X_i} = 12$, para $i = 1, \ldots, 4$, $\sigma_{X_5} = 15$ e $\sigma_{X_6} = 12$.

Problema 16 (Liu e Kiureghian [26]): Neste problema as variáveis aleatórias seguem a mesma distribuição das variáveis do Problema 15 e a função desempenho considerada é a seguinte:

$$h(\mathbf{X}) = X_1 + 2X_2 + 2X_3 + X_4 - 5X_5 - 5X_6 + 0,001\sum_{i=1}^{6}\text{sen}(100X_i). \tag{4.6}$$

Problema 17 (Mahadevan e Pan[28]): A função desempenho é dada por

$$\begin{aligned} h(\mathbf{X}) \;=\; & -240758,1777 + 10467,364X_1 + 11410,63X_2 + 3505,3015X_3 - \\ & -246,81X_1^2 - 285,3275X_2^2 - 195,46X_3^2 \end{aligned} \tag{4.7}$$

onde $X_i, i = 1 \ldots, 4$ são lognormais com médias $\mu_{X_1} = 21,2$, $\mu_{X_2} = 20$ e $\mu_{X_3} = 9,2$, e desvios padrão $\sigma_{X_1} = 0,1$, $\sigma_{X_2} = 0,2$ e $\sigma_{X_3} = 0,1$.

Problema 18 (Santosh *et al.* [50]): A função desempenho é $h(\mathbf{X}) = X_1X_2 - 78,12X_3$, onde X_1 e X_2 são variáveis Gaussianas e X_3 segue a distribuição de valores extremos (máximo) Tipo-I. Para esse problema, os valores médios das variáveis são $\mu_{X_1} = 2 \times 10^7$, $\mu_{X_2} = 10^{-4}$ e $\mu_{X_3} = 4$, e os desvios padrão são $\sigma_{X_1} = 0,5 \times 10^7$, $\sigma_{X_2} = 0,2 \times 10^{-4}$ e $\sigma_{X_3} = 1$.

Problema 19 (Santosh *et al.* [50]): A função desempenho considerada nesse problema é a mesma do Problema 18, mas agora X_1 e X_2 seguem a distribuição lognormal e X_3 segue a distribuição de valores extremos (máximo) Tipo-I, com os mesmos valores médios e desvios padrão para as variáveis do problema 18.

Problema 20 (Liu e Kiureghian [26]): A função desempenho considerada nesse problema

é a seguinte:

$$\begin{aligned} h(\mathbf{X}) &= 1,1 - 0,00115X_1X_2 + 0,00117X_1^2 + 0,00157X_2^2 + \\ &+0,0135X_2X_3 - 0,0705X_2 - 0,00534X_1 - 0,0149X_1X_3 - \\ &-0,0611X_2X_4 + 0,0717X_1X_4 - 0,226X_3 + 0,0333X_3^2 - \\ &-0,558X_3X_4 + 0,998X_4 - 1,339X_4^2. \end{aligned} \tag{4.8}$$

Nesse caso, X_1 segue a distribuição de valores extremos (máximos) Tipo-II com média $\mu_{X_1} = 10$ e desvio padrão $\sigma_{X_1} = 5$; X_2 e X_3 são variáveis aleatórias Gaussianas com valores médios $\mu_{X_2} = 25$ e $\mu_{X_3} = 0,8$, e desvios padrão $\sigma_{X_2} = 5$ e $\sigma_{X_3} = 0,2$; e X_4 é uma variável lognormal com média $\mu_{X_4} = 0,0625$ e desvio padrão $\sigma_{X_4} = 0,0625$.

Problema 21 (Wang e Grandhi [56]): A função desempenho considerada é

$$h(\mathbf{X}) = X_1^4 + 2X_2^4 - 20.$$

Nesse problema, as variáveis são Gaussianas com valores médios $\mu_{X_1} = \mu_{X_2} = 10$ e desvios padrão $\sigma_{X_2} = \sigma_{X_2} = 5$.

Problema 22 (Haldar e Mahadevan [16]): A função desempenho e as distribuições de probabilidades são as mesmas consideradas no Problema 14, mas agora X_1 e X_2 são variáveis aleatórias dependentes com correlação igual a 0,3.

Problema 23 (Liu e Kiureghian [26]): Nesse problema a confiabilidade do pórtico elástico linear de três vãos e cinco andares, conforme ilustra a Figura 4.1, é examinado e 21 variáveis randômicas básicas são consideradas: 3 carregamentos aplicados, 2 módulos de elasticidade, 8 momentos de inércia e 8 áreas de seção transversal. Assume-se que a estrutura falha caso o deslocamento horizontal no nó 1 exceda 0,2 pés. A função desempenho é expressa como $h(\mathbf{X}) = 0,2 - u_1(\mathbf{X})$. Informações referentes às distribuições de probabilidade e a matriz de correlação das variáveis são apresentadas no Anexo.

Problem 24 (Torii e Lopez [52]): Esse problema refere-se à análise da rede de distribuição de água representada pela Figura 4.2. Neste caso, todas as tubulações possuem 50 metros de comprimento, exceto a tubulação 12 que mede 100m de comprimento. O nó 1 possui uma pressão prescrita equivalente a 15m de coluna d'água, enquanto todos os demais nós possuem demandas de $0,005m^3/s$. Além disso, todas as tubulações possuem diâmetro igual a $0,11m$, com rugosidade relativa do material $ks = 0,046mm$ e o fluído é água a 15°C. É importante ressaltar que a obtenção das pressões na rede requer a solução de um sistema de equações não lineares. Este exemplo é descrito em maiores detalhes em [52]. A função de estado limite é dada por $h(\mathbf{X}) = 5 - h_{17}(\mathbf{X}) \leq 0$, onde h_{17} é a pressão no nó 17.

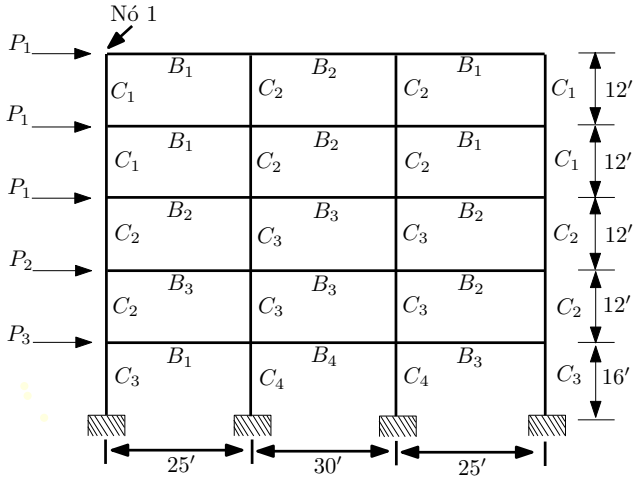


Figura 4.1: Exemplo de estrutura de pórtico para o Problema 23.

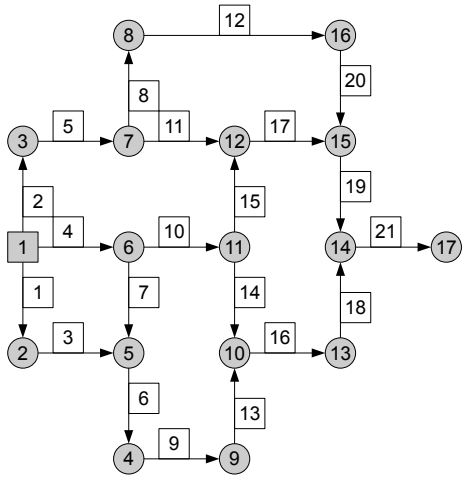


Figura 4.2: Exemplo de rede de tubulações com as tubulações e nós numerados para o Problema 24.

Problem 25 (Torii e Lopez [52]): Nesse problema a rede de distribuição, representada na Figura 4.3, é analisada. Neste caso, todas as tubulações possuem comprimento igual a 50m. O nó 10 possui uma pressão prescrita equivalente a 20m de coluna d'água, enquanto todos os demais nós possuem uma demanda de $0,005m^3/s$. Além disso, todas as tubulações possuem diâmetro igual a $0,20m$ e rugosidade relativa do material $ks = 0,002mm$ e o fluído é água a 15°C. Este exemplo é modelado e discutido em [52]. Aqui assume-se que as condições de contorno são variáveis com distribuição Gaussiana. Para a pressão prescrita no nó 10 foi utilizado um desvio padrão igual a 1m. Para as demandas nos nós foi utilizado um desvio padrão igual a $0,00125m^3/s$. A função de estado limite é dada por $h(\mathbf{X}) = 12 - \min g_i(\mathbf{X}) \leq 0$, onde $\min g_i$ é a menor pressão nodal observada na rede.

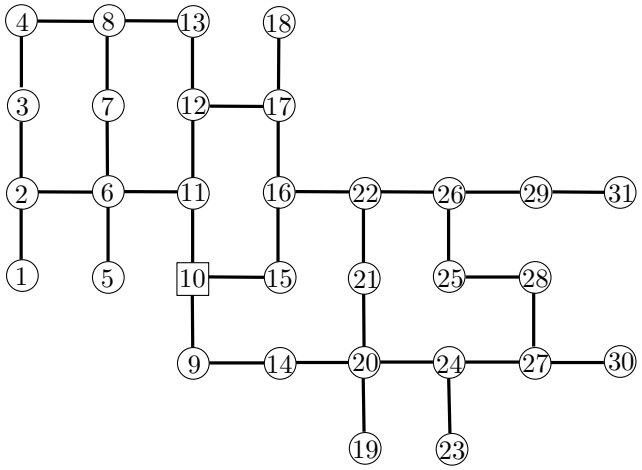


Figura 4.3: Exemplo de rede de tubulações com as tubulações e nós numerados para o Problema 25.

4.2.1 Critérios adotados na implementação dos algoritmos

Para efeitos de comparação da robustez e da eficiência dos três algoritmos propostos com o Lagrangeano aumentado clássico e com os algoritmos HLRF e iHLRF, os seis algoritmos foram implementados em Matlab R2009a versão 7.8. Todos os problemas foram resolvidos usando um processador Intel (R) Core (TM) 2 Duo CPU T5870 de 2 GHz e com 3GB de memória RAM. Os critérios utilizados na implementação desses algoritmos são apresentados a seguir.

Critério de parada

Na execução dos seis algoritmos estabelecemos um tempo computacional máximo de 1 hora e um limite máximo de 100 iterações, lembrando que, para o caso dos métodos de Lagrangeano aumentado esse limite refere-se ao número de iterações executadas no passo externo. Além disso, os algoritmos também foram interrompidos quando

$$\left|h(\mathbf{y}^k)\right| < \varepsilon \quad \text{e} \quad 1 - \frac{\left|\nabla h(\mathbf{y}^k)^T \mathbf{y}^k\right|}{\|\nabla h(\mathbf{y}^k)\| \, \|\mathbf{y}^k\|} < \varepsilon,$$

onde $\varepsilon = 10^{-4}$.

Para os métodos de Lagrangeano aumentado, assim como fizemos para os problemas da coleção CUTEr, utilizamos a rotina *fminunc* do Matlab na obtenção da solução do subproblema irrestrito. Estabelecemos, no passo interno, um número máximo de 1000 iterações, um limite de 1000 avaliações de funções e uma tolerância de 10^{-15} para a diferença entre os iterandos e seus respectivos valores funcionais.

Escolha dos parâmetros

Escolhemos os seguintes valores para as constantes dos algoritmos:

- iHLRF: $\eta = 10$, $\alpha^0 = 1$ e $m_1 = 0,1$.
- nHLRF: $\eta = 10$, $c^0 = 100$, $\alpha^0 = 1$, $m_1 = 0,1$ e $m_2 = 0,9$.
- LAPC: $\lambda^0 = 1$, $\rho^0 = 1$ e $\delta = 0,1$
- LAPM e LAPB: $\lambda^0 = 1$, $r^0 = 1$ e $\delta = 0,1$

Atualização do parâmetro de penalidade

Para o Algoritmo 4.2 consideramos um aumento no valor do parâmetro de penalidade do mesmo modo que havíamos feito na resolução dos problemas da coleção CUTEr, ou seja, $\rho^{k+1} = 2\rho^k$.

Por outro lado, para o Algoritmo 4.1 decidimos empregar uma estratégia de atualização para o parâmetro de penalidade diferentemente do que havíamos feito anteriormente, em que consideramos as Heurísticas 3.1 e 3.2, uma vez que os problemas de confiabilidade estrutural apresentam apenas uma restrição e, nesse caso, os testes indicaram melhores resultados quando atualizamos o parâmetro de penalidade em toda iteração externa, fazendo

$$\gamma^{k+1} = \frac{\left(\lambda^{k+1}\right)^2}{r^{k+1}} \qquad \text{e} \qquad \omega^{k+1} = \frac{\left|\lambda^{k+1}\right|}{r^{k+1}}, \qquad \text{sendo } r^{k+1} = 0,01r^k.$$

Escolha do ponto inicial

Em nossa implementação definimos como ponto inicial o ponto médio das variáveis originais. Entretanto, com essa escolha observamos que nenhum dos algoritmos foi capaz de resolver os problemas 5, 24 e 25, devido a ocorrência de erros numéricos durante a execução dos algoritmos. No caso do problema 5, os algoritmos HLRF, iHLRF e nHLRF apresentaram erros numéricos no cálculo da direção de descida $\mathbf{d}^k$, dado pela equação (2.6), pois $\nabla h(\mathbf{y}^0) = 0$. Para os métodos de Lagrangeano aumentado, os erros ocorreram devido as funções (4.1), (4.2) e (4.3) tornarem-se constantes em $\mathbf{y}^0$, de modo que o problema não poderia ser resolvido. No caso dos problemas 24 e 25, o erro foi ocasionado por uma divisão por zero na avaliação do gradiente da função desempenho, que é obtido por diferenças finitas. Assim, para contornar tais problemas decidimos escolher aleatoriamente os seguintes pontos iniciais (no espaço reduzido): $\mathbf{y} = (0,1)^T$ para o problema 5 e $\mathbf{y} = (0,1;\ldots;0,1)^T$ para os poblemas 24 e 25.

Comparação dos algoritmos com base nos gráficos de desempenho

Em geral, a comparação entre diversos algoritmos é feita selecionando-se um conjunto de problemas testes que são resolvidos por cada um dos algoritmos. Para cada par de problema-algoritmo são medidos diversos critérios, como por exemplo, número de iterações.

Para auxiliar a análise dos resultados utilizamos uma técnica gráfica, desenvolvida por Dolan e Moré [8], denominada de *performance profile*, em português gráfico de desempenho que permite avaliar e comparar de forma compactada o desempenho de um conjunto $\mathcal{S}$ de algoritmos aplicados a um conjunto $\mathcal{P}$ de n_p problemas testes, de acordo com um determinado critério.

Para compreensão da construção do gráfico de desempenho, vamos definir como medida de desempenho o tempo computacional. Dolan e Moré [8] definem o seguinte índice de desempenho

$$r_{p,s} = \frac{t_{p,s}}{\{\min\{t_{p,s} | \, s \in \mathcal{S}\}}, \tag{4.9}$$

em que, neste caso, $t_{p,s}$ é o tempo de processamento necessário para a resolução do problema $p \in \mathcal{P}$ pelo algoritmo $s \in \mathcal{S}$.

Considere um parâmetro $r_M \geq \max\{r_{p,s}\}$. Sempre que um algoritmo s não resolve o problema p, o índice de desempenho $r_{p,s}$ é definido como $r_{p,s} = r_M$.

Seja

$$\mathcal{K}(r_{p,s}, \tau) = \begin{cases} 1, & \text{se } r_{p,s} \leq \tau \\ 0, & \text{caso contrário} \end{cases} \tag{4.10}$$

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

Dolan e Moré [8] estabelecem

$$\rho_s(\tau) = \frac{1}{n_p} \sum_{j \in \mathcal{P}} \mathcal{K}(r_{p,s}, \tau) \tag{4.11}$$

como a probabilidade do índice $r_{p,s}$ estar dentro de um fator τ do melhor índice possível. Deste modo, $\rho_s(1)$ é a proporção de problemas que o algoritmo s resolve no menor tempo.

Considerando uma medida de desempenho arbitrária, $\rho_s(\tau)$ representa a porcentagem de problemas que o algoritmo $s \in \mathcal{S}$ resolve em τ vezes o valor da medida de desempenho do algoritmo mais eficiente.

Deste modo, no gráfico de desempenho, o eixo das ordenadas representa o percentual de problemas cujo o algoritmo $s \in \mathcal{S}$ obteve o melhor desempenho para o conjunto de problemas $\mathcal{P}$, ou seja, os valores das respectivas probabilidades $\rho_s(1)$. Enquanto que, os valores do fator τ são indicados no eixo das abscissas e representam a quantidade de vezes que o algoritmo s superou o melhor desempenho, considerando todos os problemas do conjunto $\mathcal{P}$, fixado uma medida de desempenho.

Para facilitar a visualização, Dolan e Moré [8] sugerem a utilização de uma escala logarítmica, $\log_2(r_{p,s})$. Neste caso, $\rho_s(\tau)$ é calculado substituindo-se os valores de $r_{p,s}$ por $\log_2(r_{p,s})$ nas equações (4.10) e (4.11), e é interpretado como a porcentagem de problemas que o algoritmo s resolve em 2^τ vezes o valor da melhor medida de desempenho.

Apresentamos na próxima seção os resultados e as análises dos testes numéricos.

4.3 Resultados numéricos

Para cada problema, a Tabela 4.1 apresenta os resultados referentes ao algoritmo HLRF na primeira linha, para o iHLRF na segunda linha e para o algoritmo nHLRF na terceira linha. Quatro medidas foram selecionadas para a comparação: *iter* que representa o número de iterações, *fun* que refere-se ao número total de avaliações da função desempenho, *grad* que representa o número de avaliações do gradiente da função desempenho e *tempo* que é o tempo computacional gasto na execução do algoritmo. Em cada caso, o índice de confiabilidade (β) também é apresentado.

Tabela 4.1: Comparação do desempenho entre os algoritmos HLRF, iHLRF e nHLRF.

Problema	*Método*	*iter*	*fun*	*grad*	*tempo (segundos)*	β
1	HLRF	1	2	2	0,1074	2,5000
	iHLRF	1	3	2	0,0213	2,5000
	nHLRF	1	3	3	0,0353	2,5000
2	HLRF	1	2	2	0,0003	3,0000
	iHLRF	1	3	2	0,0054	3,0000
	nHLRF	1	3	3	0,0093	3,0000
3	HLRF	1	2	2	0,0002	2,0000
	iHLRF	1	3	2	0,0035	2,0000
	nHLRF	1	3	3	0,0106	2,0000
4	HLRF	1	2	2	0,0001	3,0000
	iHLRF	1	3	2	0,0032	3,0000
	nHLRF	1	3	3	0,0071	3,0000
5	HLRF	4	5	5	0,0007	0,3536
	iHLRF	4	9	5	0,0045	0,3536
	nHLRF	4	9	9	0,0104	0,3536
6	HLRF	1	2	2	0,0001	2,0000
	iHLRF	1	3	2	0,0042	2,0000
	nHLRF	1	3	3	0,0084	2,0000
7	HLRF	7	8	8	0,0008	2,2401
	iHLRF	7	15	8	0,0067	2,2401
	nHLRF	7	15	15	0,0169	2,2401
8	HLRF	100	100	100	0,0119	1,1656
	iHLRF	32	133	33	0,0366	2,2260
	nHLRF	32	133	133	0,0459	2,2260
9	HLRF	1	2	2	0,0003	2,5000
	iHLRF	1	3	2	0,0028	2,5000
	nHLRF	1	3	3	0,0093	2,5000
10	HLRF	100	100	100	0,0128	1,1466
	iHLRF	47	189	48	0,0554	1,9003
	nHLRF	37	138	138	0,0488	1,9003
11	HLRF	5	6	6	0,0007	5,4280

continua na próxima página

Tabela 4.1 – continuação da página anterior

Problema	*Método*	*iter*	*fun*	*grad*	*time (segundos)*	β
	iHLRF	5	11	6	0,0051	5,4280
	nHLRF	5	11	11	0,0093	5,4280
12	HLRF	1	2	2	0,0002	2,2257
	iHLRF	1	3	2	0,0043	2,2257
	nHLRF	1	3	3	0,0143	2,2257
13	HLRF	5	6	6	0,0240	2,1911
	iHLRF	5	11	6	0,0064	2,1911
	nHLRF	5	11	11	0,0115	2,1911
14	HLRF	3	4	4	0,0494	5,2127
	iHLRF	3	7	4	0,0057	5,2127
	nHLRF	3	7	7	0,0159	5,2127
15	HLRF	13	14	14	0,0261	3,0483
	iHLRF	15	35	16	0,0286	3,0483
	nHLRF	15	82	82	0,0194	3,0483
16	HLRF	100	100	100	0,0487	2,3481
	iHLRF	100	97175	101	28,560	2,3480
	nHLRF	64	2022	2022	0,2063	2,3482
17	HLRF	11	12	12	0,0228	0,8292
	iHLRF	34	140	35	0,0403	0,8292
	nHLRF	14	34	34	0,0302	0,8292
18	HLRF	9	10	10	0,0102	3,3221
	iHLRF	12	30	13	0,0212	3,3221
	nHLRF	12	31	31	0,0157	3,3221
19	HLRF	6	7	7	0,0159	4,4282
	iHLRF	6	13	7	0,0124	4,4282
	nHLRF	6	13	13	0,0171	4,4282
20	HLRF	100	100	100	0,0388	1,1643
	iHLRF	10	30	11	0,0120	1,3651
	nHLRF	13	42	42	0,0210	1,3653
21	HLRF	100	100	100	0,0111	0,9863
	iHLRF	39	196	40	0,0366	2,3655
	nHLRF	56	276	276	0,0737	2,3655
22	HLRF	4	5	5	0,0194	4,5297
	iHLRF	4	9	5	0,0595	4,5297
	nHLRF	4	9	9	0,0534	4,5297
23	HLRF	3	4	4	0,7738	2,6666
	iHLRF	3	7	4	0,4114	2,6666
	nHLRF	4	12	12	0,5347	2,6666
24	HLRF	não convergiu				
	iHLRF	não convergiu				
	nHLRF	não convergiu				
25	HLRF	não convergiu				
	iHLRF	não convergiu				

continua na próxima página

Tabela 4.1 – continuação da página anterior

Problema	*Método*	*iter*	*fun*	*grad*	*time (segundos)*	β
	nHLRF	não convergiu				

De acordo com a Tabela 4.1, observa-se que os algoritmos HLRF, iHLRF e nHLRF resolveram com sucesso 72%, 88% e 92% dos problemas, respectivamente. Logo, o algoritmo nHLRF resolveu uma quantidade maior de problemas, e portanto pode ser considerado mais robusto do que os algoritmos HLRF e iHLRF. Os resultados mostram que nenhum dos três algoritmos (HLRF, iHLRF e nHLRF) foi capaz de solucionar os problemas 24 e 25. Isso aconteceu porque durante o processo iterativo $\nabla h(\mathbf{y}^k) = 0$, o que ocasionou erros numéricos no cálculo da direção de busca $\mathbf{d}^k$. Nota-se também que o algoritmo HLRF atingiu o número máximo de iterações ($k = 100$) na resolução dos problemas 8, 10, 16, 20 e 21, e isso também ocorreu com o iHLRF para o Problema 16. Embora os algoritmos HLRF e iHLRF não tenham satisfeito o critério de parada para o Problema 16, os valores encontrados para os índices de confiabilidade estão muito próximos da solução.

A fim de simplificar a análise dos resultados, construímos tabelas menores com informações específicas. As Tabelas 4.2, 4.3 e 4.4 referem-se ao número de vezes que cada algoritmo, HLRF, iHLRF e nHLRF, obteve melhor desempenho levando em consideração o número de iterações (*iter*), o número de avaliações da função desempenho (*fun*), o número de avaliações do gradiente (*grad*) e o tempo computacional (*tempo*). Os valores relacionados ao "*empate*" indicam a quantidade de problemas cujo o desempenho dos algoritmos coincidiu. Para a construção da Tabela 4.2, analisamos os resultados de 22 problemas que foram resolvidos por pelo menos um dos dois algoritmos, HLRF ou iHLRF. Nessa análise, os problemas 16, 24 e 25 não foram considerados.

Tabela 4.2: Comparação do desempenho entre os algoritmos HLRF e iHLRF.

Método	*iter*	*fun*	*grad*	*tempo*
HLRF	3	18	3	13
iHLRF	4	4	4	9
empate	15	0	15	0

Na construção das Tabelas 4.3 e 4.4, os problemas 24 e 25 não foram considerados, pois não foram resolvidos por nenhum dos três algoritmos (HLRF, iHLRF e nHLRF).

Tabela 4.3: Comparação do desempenho entre os algoritmos HLRF e nHLRF.

Método	*iter*	*fun*	*grad*	*tempo*
HLRF	4	18	18	13
nHLRF	5	5	5	10
empate	14	0	0	0

Os resultados apresentados nas Tabelas 4.2 e 4.3 indicam que, quando o algoritmo HLRF converge, seu desempenho supera o desempenho dos algoritmos iHLRF e nHLRF,

em termos de, número de avaliações de funções, número de avaliações de gradientes e tempo computacional. De fato, uma vez que os algoritmos iHLRF e nHLRF executam uma busca linear para calcular o comprimento do passo, α^k, esses algoritmos naturalmente realizam um número maior de avaliações de funções e de gradientes. O pior desempenho, no entanto, é compensado pela robustez do algoritmos iHLRF e nHLRF.

Consideramos que os algoritmos iHLRF e nHLRF são mais robustos, pois resolveram 22 e 23 problemas dos 25 testados, respectivamente, enquanto que o algoritmo HLRF resolveu apenas 18 problemas. Este resultado reforça a observação de que o algoritmo HLRF é eficiente quando converge, mas não há qualquer garantia de convergência. O algoritmo nHLRF aqui apresentado, por outro lado, possui garantia de convergência.

Tabela 4.4: Comparação do desempenho entre os algortimos iHLRF e nHLRF.

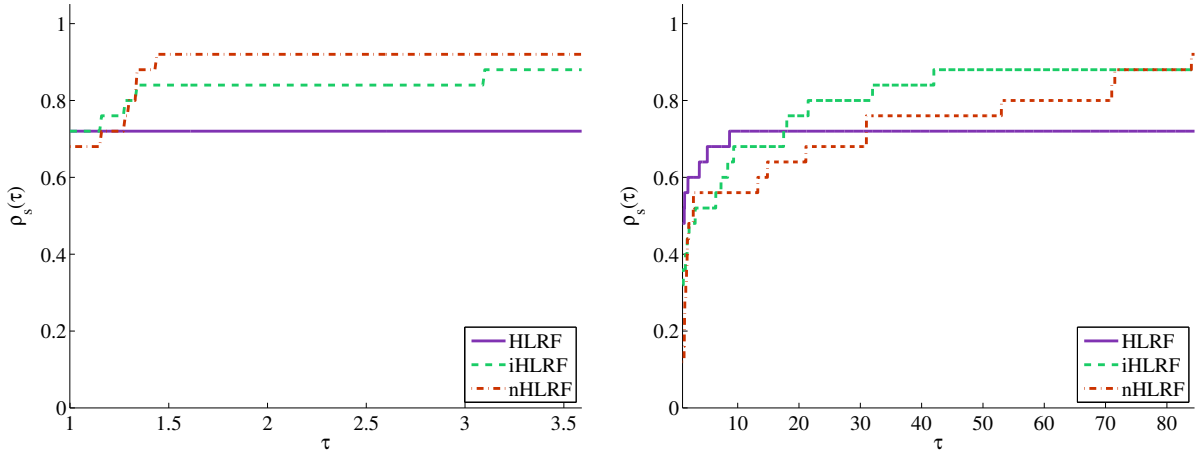
Método	*iter*	*fun*	*grad*	*tempo*
iHLRF	3	5	21	17
nHLRF	3	3	2	6
empate	17	15	0	0

Os resultados mostrados na Tabela 4.4 indicam que os algoritmos iHLRF e nHLRF possuem desempenhos equivalentes com relação ao número de iterações. Contudo, o algoritmo iHLRF apresenta melhor desempenho com relação a *fun*, *grad* e *tempo*. Isso acontece porque no algoritmo iHLRF, α^k é obtido pela regra de Armijo, enquanto que no algoritmo nHLRF, são utilizadas as condições de Wolfe. As condições de Wolfe requerem na minimização da função de mérito a avaliação de sua derivada, consequentemente um número mais elevado de funções h e gradientes ∇h é calculado. Em contrapartida, o benefício para a menor eficiência é a garantia de convergência do algoritmo nHLRF. Enquanto os algoritmos iHLRF e HLRF atingiram o número máximo de iterações na resolução de problemas 16 e 8, 10, 16, 20 e 21, respectivamente, o algoritmo nHLRF foi capaz de resolver esses problemas dentro do número máximo de iterações.

Para finalizar a comparação entre os algoritmos baseados em HLRF, construímos também gráficos de desempenho na escala decimal para o número iterações, tempo computacional, número de avaliações da função h e de sua derivada, que são apresentados nas Figuras 4.4 e 4.5.

No gráfico apresentado na Figura 4.4(a) à medida de desempenho considerada é o número de iterações. Podemos notar que não existe diferença significativa entre os desempenhos dos algoritmos HLRF, iHLRF e nHLRF com base nesse critério, uma vez que os algoritmos HLRF e iHLRF resolveram 72% dos problemas com o menor número de iterações, o que equivale a 18 problemas, enquanto que para nHLRF a porcentagem foi de 68%, o que equivale a 17 problemas do total de 25 problemas testados. Lembrando que, os algoritmos HLRF, iHLRF e nHLRF resolveram 18, 22 e 23 dos 25 problemas testados com sucesso, respectivamente.

Analisando o eixo das abscissas do gráfico apresentado na Figura 4.4(a), é possível perceber que o algoritmo nHLRF resolveu 92% dos problemas testados, utilizando aproximadamente 1,5 vezes o menor número de iterações, diferentemente do algoritmo iHLRF que necessitou de aproximadamente 3,1 vezes o melhor desempenho para resolver 88% dos problemas. É importante observar que, o algorimto HLRF apresentou melhor desempenho para todos os problemas, para o qual foi capaz de solucionar, o que também ocorreu com relação ao número de avaliações da função desempenho e de sua derivada, conforme pode ser visto nas Figura 4.5(a) e 4.5(b).



(a) Número de iterações

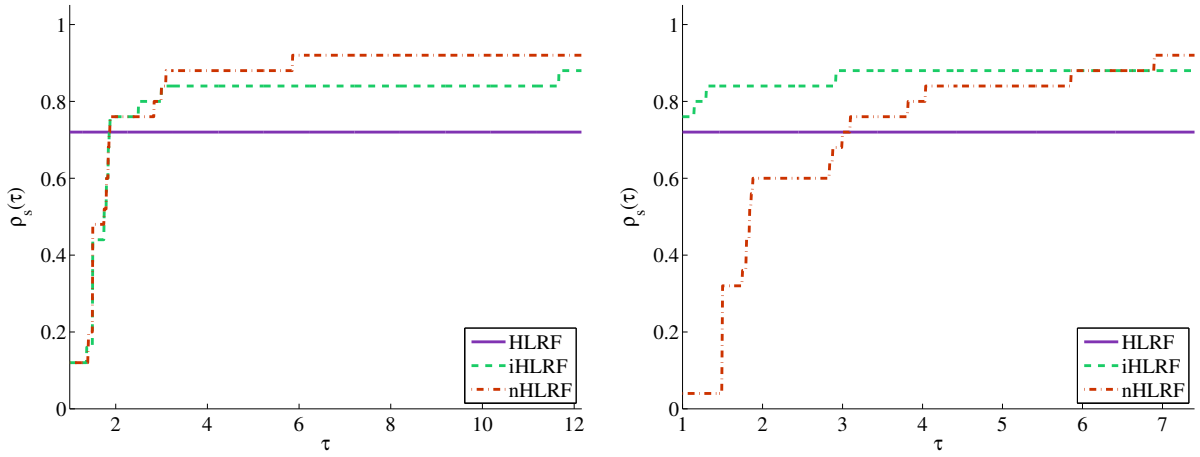
(b) Tempo computacional

Figura 4.4: Gráficos de desempenho para os algoritmos baseados em HLRF com relação ao número de iterações e tempo computacional.

Com relação ao tempo computacional, o gráfico apresentado na Figura 4.4(b) indica que o algoritmo HLRF foi mais eficiente em 48% dos problemas analisados, enquanto que os algoritmos iHLRF e nHLRF em 32% e 12% dos casos, respectivamente.

Com relação ao número de avaliações da função desempenho, o gráfico da Figura 4.5(a) indica que o algoritmo HLRF resolveu 72% dos problemas com um número menor de avaliações da função h, enquanto que para os algorimtos iHLRF e nHLR a porcentagem foi de 12%. Além disso, o algoritmo nHLRF resolveu 92% dos problemas testados utilizando aproximadamente 6 vezes o menor número de avaliações de h, sendo que o algoritmo iHLRF necessitou de aproximadamente 11,7 vezes o melhor desempenho para resolver 88% dos problemas.

Para o número de avaliações da derivada da função desempenho, o gráfico da Figura 4.5(b) mostra que os algoritmos HLRF e iHLRF resolveram 72% e 76% dos problemas, respectivamente, com um número menor de avaliação de ∇h, enquanto que para o algoritmo nHLRF a porcentagem foi de 4%. Além disso, o algoritmo nHLRF resolveu 92% dos problemas testados utilizando aproximadamente 6,6 vezes o menor número de avaliações de ∇h, sendo que o algoritmo iHLRF necessitou de aproximadamente 3 vezes o melhor desempenho para resolver 88% dos problemas. Conforme abordado anteriormente,



(a) Número de avaliações da função desempenho (b) Número de avaliações do gradiente da função desempenho

Figura 4.5: Gráficos de desempenho para os algoritmos baseados em HLRF com relação ao número de avaliações da função desempenho e do seu gradiente.

esses resultados são justificados pela necessidade de realização da busca linear durante o processo iterativo. Tais resultados são compensados pelo fato do algoritmo nHLRF ser mais robusto que os algoritmos HLRF e iHLRF.

Os resultados da aplicação dos métodos de Lagrangeano aumentado são apresentados na Tabela 4.5. Para cada problema, os resultados para os algoritmos LAPC, LAPM e LAPB são mostrados, respectivamente, na primeira, na segunda e terceira linhas da tabela. Para os métodos de Lagrangeano aumentado, *it.ext* representa a quantidade de iterações realizadas no passo externo, *it.int* é o número de iterações realizadas pela rotina *fminunc* no passo interno, *fun* é o número total de avaliações da função desempenho, *grad* é o número total de avaliações do gradiente da função desempenho e β é o valor para índice de confiabilidade obtido em cada caso. É importante notar que os valores de *fun* e *grad* são os mesmos, por esta razão estão listados em uma mesma coluna.

Tabela 4.5: Comparação do desempenho entre os algoritmos LAPM, LAPB e LAPC.

Problema	*Método*	*it.ext*	*it.int*	*fun/grad*	*tempo (segundos)*	β
1	LAPC	5	9	18	0,6023	2,5000
	LAPM	2	4	6	0,1686	2,5000
	LAPB	2	4	6	0,0698	2,5000
2	LAPC	5	9	18	0,2371	3,0000
	LAPM	2	4	6	0,0506	3,0000
	LAPB	2	4	6	0,0482	3,0000
3	LAPC	5	8	17	0,1386	2,0000
	LAPM	2	3	6	0,0471	2,0000
	LAPB	2	3	6	0,0475	2,0000

continua na próxima página

Tabela 4.5 – continuação da página anterior

Problema	*Método*	*it.ext*	*it.int*	*fun/grad*	*tempo (segundos)*	β
4	LAPC	5	9	17	0,1094	3,0000
	LAPM	2	4	6	0,0473	3,0000
	LAPB	2	4	6	0,0504	3,0000
5	LAPC	1	13	15	0,0299	0,3536
	LAPM	1	13	15	0,0439	0,3536
	LAPB	1	13	15	0,0443	0,3536
6	LAPC	5	8	17	0,0708	2,0000
	LAPM	2	3	6	0,0488	2,0000
	LAPB	2	3	6	0,0499	2,0000
7	LAPC	2	20	25	0,0425	2,2401
	LAPM	4	29	40	0,0924	2,2401
	LAPB	3	22	31	0,0752	2,2401
8	LAPC	2	46	52	0,0535	2,2260
	LAPM	4	52	66	0,1070	2,2260
	LAPB	3	43	55	0,0939	2,2260
9	LAPC	5	9	18	0,0705	2,5000
	LAPM	2	4	6	0,0461	2,5000
	LAPB	2	4	6	0,0527	2,5000
10	LAPC	2	37	46	0,0518	1,9003
	LAPM	4	45	59	0,1099	1,9003
	LAPB	3	44	55	0,0876	1,9003
11	LAPC	1	68	90	0,0626	5,3333
	LAPM	1	68	90	0,0820	5,3333
	LAPB	1	68	90	0,0950	5,3333
12	LAPC	5	9	18	0,0718	2,2257
	LAPM	2	3	6	0,0454	2,2257
	LAPB	2	3	6	0,0502	2,2257
13	LAPC	1	64	75	0,1975	2,2046
	LAPM	6	132	178	0,2423	2,1911
	LAPB	3	137	177	0,1712	2,1911
14	LAPC	4	71	95	0,2296	5,2127
	LAPM	4	64	75	0,1724	5,2127
	LAPB	4	62	87	0,1402	5,2127
15	LAPC	9	78	362	0,7052	3,0486
	LAPM	6	63	274	0,3319	3,0483
	LAPB	8	48	344	0,3170	3,0494
16	LAPC	4	72	160	0,2266	2,3484
	LAPM	5	69	171	0,2346	2,3486
	LAPB	4	88	190	0,2252	2,3483
17	LAPC	2	53	71	0,0787	0,8292
	LAPM	2	38	57	0,1766	0,8292
	LAPB	3	58	80	0,1261	0,8292
18	LAPC	1	88	107	0,1377	3,3225

continua na próxima página

Tabela 4.5 – continuação da página anterior

Problema	*Método*	*it.ext*	*it.int*	*fun/grad*	*tempo (segundos)*	β
	LAPM	4	120	159	0,2193	3,3221
	LAPB	3	123	155	0,1774	3,3221
19	LAPC	5	309	407	0,4854	4,4129
	LAPM	5	194	241	0,3088	4,4169
	LAPB	4	219	272	0,3718	4,4103
20	LAPC	13	146	165	0,4636	1,3656
	LAPM	2	18	21	0,1173	1,7198
	LAPB	3	57	61	0,1737	1,3704
21	LAPC	2	64	75	0,0654	2,3655
	LAPM	4	68	87	0,1833	2,3655
	LAPB	3	70	84	0,1362	2,3655
22	LAPC	3	51	67	0,2757	4,5297
	LAPM	5	60	84	0,2082	4,5297
	LAPB	3	39	55	0,6174	4,5297
23	LAPC	11	88	106	7,8007	2,6670
	LAPM	2	14	16	4,9509	2,6670
	LAPB	3	63	73	6,7781	2,6676
24	LAPC	34	179	344	163,2402	1,8103
	LAPM	3	34	47	24,9473	1,8106
	LAPB	5	34	60	30,5753	1,8106
25	LAPC	25	262	626	535,6507	2,2016
	LAPM	3	35	83	77,4204	2,2004
	LAPB	3	39	85	83,3869	2,2004

Com base nos resultados da Tabela 4.5, observa-se que os algoritmos LAPC, LAPM e LAPB resolveram com sucesso 100% dos problemas testados. Assim, os métodos de Lagrangeano aumentado podem, portanto, ser considerados mais robustos do que os algorimtos HLRF, iHLRF e nHLRF que resolveram um menor número de problemas, como mostrado na Tabela 4.1.

Os desempenhos dos algoritmos LAPC, LAPB e LAPM são comparados nas Tabelas 4.6, 4.7 e 4.8. A comparação leva em conta o número de iterações do algoritmo no passo externo (*it.ext*), o número de iterações realizadas no passo interno (*it.int*), o número de avaliações da função desempenho (*fun*), o número de avaliações do gradiente da função desempenho (*grad*) e o tempo computacional (*tempo*). Nesta análise, os 25 problemas foram considerados, pois foram resolvidos por cada um dos algoritmos.

Tabela 4.6: Comparação do desempenho entre os algoritmos LAPC e LAPM.

Método	*it.ext*	*it.int*	*fun/grad*	*tempo*
LAPC	8	7	8	10
LAPM	12	16	15	15
empate	5	2	2	0

De acordo com os dados da Tabela 4.6, pode-se observar que o algoritmo LAPM apresentou melhor desempenho na resolução dos problemas de confiabilidade estrutural do que o algoritmo LAPC, com respeito a todas as medidas de comparação consideradas *it.ext*, *it.int*, *fun*, *grad* e *tempo*. O mesmo ocorreu com o método LAPB, como pode ser visto na Tabela 4.7.

Tabela 4.7: Comparação do desempenho entre os algoritmos LAPC e LAPB.

Método	*it.ext*	*it.int*	*fun/grad*	*tempo*
LAPC	7	7	8	9
LAPB	13	16	15	16
empate	5	2	2	0

Assim, os resultados apresentados aqui indicam que os métodos de Lagrangeano aumentado propostos (LAPM e LAPB) não são apenas mais robustos, mas também mais eficientes do que o "clássico" LAPC. Nota-se que o mau desempenho atribuído ao método de Lagrangeano aumentado, por Liu e Kiureghian [26], refere-se ao método clássico, de modo que há um espaço significativo para a consideração dos métodos modernos de Lagraneano aumentado na resolução de problemas de confiabilidade estrutural.

Tabela 4.8: Comparação do desempenho entre os algoritmos LAPM e LAPB.

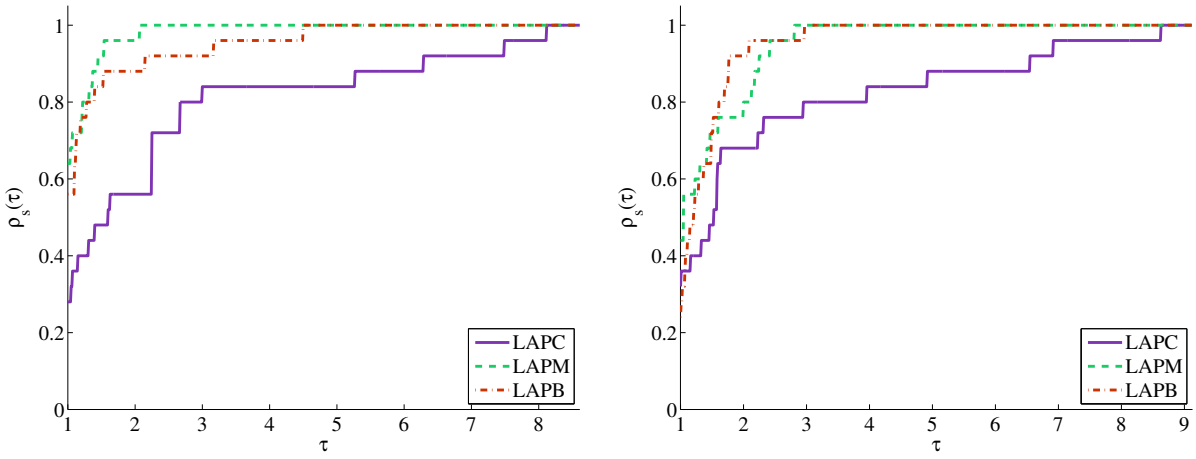
Método	*it.ext*	*it.int*	*fun/grad*	*tempo*
LAPM	5	9	9	13
LAPB	9	6	7	12
empate	11	10	9	0

A comparação entre LAPM e LAPB apresentada na Tabela 4.8 indica um empate no desempenho desses algoritmos para a maioria dos problemas testados com relação aos critérios avaliados. Para o restante dos problemas em que não houve empate, o algoritmo LAPB foi mais eficiente com relação ao número de iterações realizadas no passo externo (*it.ext*). No entanto, é preciso observar que o maior esforço computacional dos métodos de Lagrangeano aumentado está associado a resolução dos subproblemas irrestritos no passo interno, gerados pela minimização da função Lagrangeano aumentado, uma vez que no passo externo do algoritmo, é realizada somente a atualização do parâmetro penalidade e do multiplicador Lagrange.

Já o algoritmo LAPM foi mais eficiente do que LAPB com relação a *it.int*, *fun* e *grad*. Com relação ao tempo computacional consideramos a eficiência dos dois algoritmos similar. Os resultados da comparação entre os algoritmos LAPM e LAPB foram semelhantes aos apresentados para os problemas da coleção CUTEr.

No gráfico apresentado na Figura 4.6(a), analisamos o desempenho dos algoritmos de Lagrangeano aumentado com relação ao número de iterações realizadas no passo interno. O gráfico indica que os algoritmos LAPC, LAPM e LAPB resolveram 48%, 64%

e 56% dos problemas com o menor número de iterações no passo interno, o que equivale a 7, 16 e 14 problemas, respectivamente, dos 25 problemas testados. Observando o eixo das abscissas, é possível notar que, o algoritmo LAPM resolveu os problemas utilizando até 2,1 vezes o menor número de iterações, diferentemente dos algoritmos LAPB e LAPC, que necessitaram aproximadamente até 4,5 e 8,2 vezes, respectivamente, o melhor desempenho.



(a) Número de iterações no passo interno

(b) Tempo computacional

Figura 4.6: Gráficos de desempenho para os algoritmos de Lagrangeano aumentado com relação ao número de iterações no passo interno e tempo computacional.

A Figura 4.6(b) mostra que o algoritmo LAPM foi mais eficiente com relação ao tempo computacional em 44% dos problemas testados. Já os algoritmos LAPC e LAPB, foram mais eficientes com relação a este critério em 32% e 24% dos casos, respectivamente. Além disso, é possível perceber que, o algoritmo LAPM resolveu 100% dos casos utilizando aproximadamente 2,9 vezes o menor tempo computacional, enquanto que os algoritmos LAPB e LAPC necessitaram aproximadamente até 3 e 8,7 vezes o melhor desempenho, respectivamente.

O gráfico da Figura 4.7, apresenta o desempenho dos algoritmos com relação ao número de avaliações da função desempenho. Esta mesma análise pode ser estendida para o número de avaliações de ∇h, já que os resultados são os mesmos . O gráfico indica que os algoritmos LAPC, LAPM e LAPB resolveram 9%, 17% e 10% dos problemas, respectivamente, com o menor número de avaliações da função h. Observando o fator τ, é possível notar que, o algoritmo LAPM resolveu todos problemas utilizando até 2,4 vezes o menor número de avaliações da função desempenho, já os algoritmos LAPB e LAPC, utilizaram até 4,6 e 7,9 vezes, respectivamente, o melhor desempenho para esta medida.

Desta forma, é possível verificar que, assim como ocorreu com os problemas da coleção CUTEr, os resultados para os problemas de confiabilidade estrutural indicam que os algoritmos de Lagrangeano aumentado propostos são mais eficientes que o algoritmo de Lagrangeano aumentado clássico, proposto por Hestenes [18] e Powell [40].

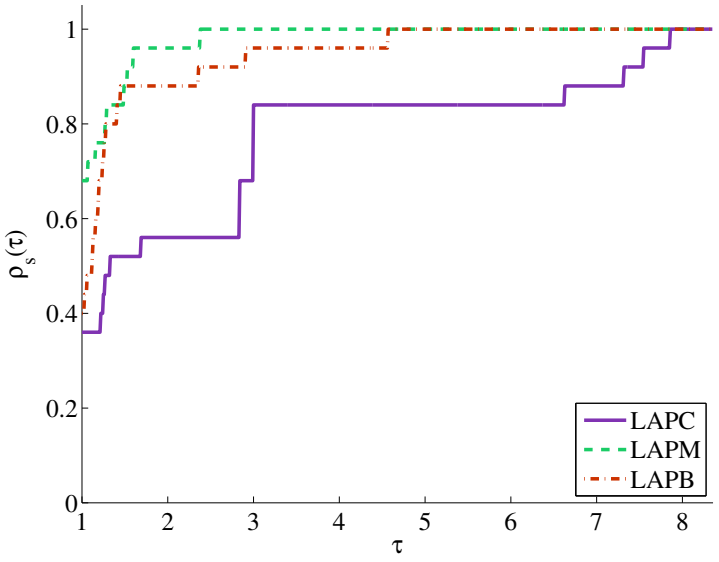


Figura 4.7: Gráficos de desempenho para os algoritmos de Lagrangeano aumentado com relação ao número de avaliações da função desempenho.

Em problemas práticos de confiabilidade estrutural, há uma preocupação com relação ao número de avaliações da função desempenho e de sua derivada, pois na maioria dos casos, a função desempenho é apresentada de forma implícita, de modo que, tais avaliações são realizadas por um modelo de elementos finitos e possuem um alto custo computacional.

Sendo assim, apresentamos nas Tabelas 4.9, 4.10 e 4.11 os resultados da comparação do desempenho entre os algoritmos, considerando o número total de chamadas da função de estado limite (*fun*) e a quantitdade de cálculos de gradiente (*grad*).

Tabela 4.9: Comparação do desempenho entre os algoritmos LAPC e HLRF, iHLRF, nHLRF.

Método	*fun*	*grad*	*Método*	*fun*	*grad*	*Método*	*fun*	*grad*
LAPC	6	7	LAPC	7	4	LAPC	7	6
HLRF	19	18	iHLRF	18	21	nHLRF	18	19
empate	0	0	empate	0	0	empate	0	0

Tabela 4.10: Comparação do desempenho entre os algoritmos LAPM e HLRF, iHLRF, nHLRF.

Método	*fun*	*grad*	*Método*	*fun*	*grad*	*Método*	*fun*	*grad*
LAPM	6	7	LAPM	7	3	LAPM	7	6
HLRF	19	18	iHLRF	18	22	nHLRF	18	19
empate	0	0	empate	0	0	empate	0	0

Comparando os resultados dos algoritmos baseados em HLRF e dos algoritmos de Lagrangeano aumentado, com relação ao número total de avaliações da função desempenho e de suas derivadas, nota-se que os algoritmos baseados em HLRF são mais

Tabela 4.11: Comparação do desempenho entre os algoritmos LAPB e HLRF, iHLRF, nHLRF.

Método	*fun*	*grad*	*Método*	*fun*	*grad*	*Método*	*fun*	*grad*
LAPB	6	7	LAPB	7	3	LAPB	6	6
HLRF	19	18	iHLRF	18	22	nHLRF	19	19
empate	0	0	empate	0	0	empate	0	0

eficientes, como pode ser visto nas Tabelas 4.9, 4.10, 4.11. Isto é justificado pela generalidade dos métodos de Lagrangeano aumentado, que podem ser aplicados na resolução de problemas de otimização com restrições de igualdade e, até mesmo, com restrições de desigualdades, como discutido em [3] e [32], além disso são robustos e podem ser aplicados a problemas com centenas e milhares de restrições, enquanto que os algoritmos HLRF, iHLRF e nHLRF foram desenvolvidos especificamente para problemas de confiabilidade estrutural.

Entretanto, é importante salientar que os métodos de Lagrangeano aumentado apresentaram melhores resultados na resolução de problemas com um número maior de variáveis, como no caso dos problemas de 16, 24 e 25. A propósito, consideramos esse fato uma vantagem importante dos métodos de Lagrangeano aumentado, já que a maioria dos problemas práticos envolve um grande número de variáveis aleatórias.

Quanto ao número de problemas resolvidos, os algoritmos LAPC, LAPM e LAPB são mais robustos, pois resolveram 100% dos problemas testados, seguidos pelo algoritmo nHLRF, que resolveu 92% dos problemas, e pelos algoritmos iHLRF e HLRF, que solucionaram apenas 88% e 72% dos problemas testados, respectivamente.

4.4 Conclusão dos resultados numéricos

Nessa seção apresentamos a aplicação dos três novos algoritmos de otimização que estamos propondo a problemas de confiabilidade estrutural, e os resultados foram comparados aos obtidos pelos algoritmos LAPC, HLRF e iHLRF.

Os resultados numéricos mostrados aqui, indicam que o algoritmo nHLRF possui um desempenho semelhante ao iHLRF, com relação ao número de iterações e avaliações da função desempenho. Além disso, o algoritmo nHLRF pode ser considerado mais robusto que o HLRF e o iHLRF, já que o mesmo resolveu uma quantidade maior de problemas.

Já os métodos de Lagrangeano aumentado propostos (LAPM e LAPB) mostraram-se mais eficientes do que o método de Lagrangeano aumentado clássico (LAPC), pois resolveram um maior número de problemas com um menor número de iterações, de avaliações de funções e tempo computacional. E, as duas metodologias (LAPM e LAPB) comparadas entre si, apresentaram desempenhos semelhantes para os critérios avaliados. Tais resultados indicam que a eficiência dos métodos de Lagrangeano aumentado se man-

teve semelhante a observada na resolução dos 124 problemas da coleção CUTEr, conforme apresentado no capítulo anterior.

Comparando os algoritmos baseados em HLRF com os algoritmos de Lagrangeano aumentado, notou-se que os baseados em HLRF são mais eficientes, principalmente no que diz respeito ao número de avaliações da função desempenho e de gradientes. Entretanto, os métodos de Lagrangeano aumentado são mais indicados para a resolução de problemas que envolvem grande número de variáveis aleatórias, pois ao contrário dos algoritmos HLRF, iHLRF e nHLRF, obtiveram a solução para os problemas com maior número de variáveis. Além disso, nos algoritmos baseados em HLRF é necessário que o gradiente da função desempenho, que é avaliado iterativamente, seja não nulo. Caso essa suposição não seja atendida, ou seja, $\nabla h(\mathbf{y}^k) = 0$, os algoritmos HLRF, iHLRF e nHLRF não são capazes de determinar a solução. Isso ocorreu na resolução dos Problemas 24 e 25. Sendo assim, para a resolução destes problemas é imprescindível a utilização de métodos alternativos, que não dependam da suposição de que $\nabla h(\mathbf{y}^k) \neq 0$, o que torna relevante a utilização dos métodos de Lagrangeano aumentado na obtenção do índice de confiabilidade estrutural.

Portanto, o resultados numéricos indicam que os novos métodos aplicados ao problema de confiabilidade estrutural são competitivos, promissores e mais robustos do que os demais métodos apresentados aqui.

Conclusão

Nesse trabalho três novos algoritmos de otimização foram apresentados. Dois dos novos algoritmos referem-se ao método de Lagrangeano aumentado e possuem características mais gerais, no sentido de que são utilizados para a resolução de problemas com diversas restrições de igualdade. O terceiro algoritmo, no entanto, baseia-se no algoritmo HLRF e foi desenvolvido especificamente para a resolução de problemas de confiabilidade estrutural.

Os dois novos métodos de Lagrangeano aumentado (Algoritmos LAPB e LAPM) são baseados nos resultados apresentados por Matioli e Gonzaga [32] e Tseng e Bertsekas [53] para problemas com restrições de desigualdade. Os resultados apresentados por Birgin, Castillo e Martínez [4] indicando que o método de Lagrangeano aumentado clássico tem superioridade sobre os métodos modernos, e o fato de que as novas funções de penalidade, propostas nesse trabalho, apresentarem bom desempenho tanto do ponto de vista teórico como computacional, encorajou-nos a introduzir novos métodos de Lagrangeano aumentado com características semelhantes aos clássicos, mas com propriedades dos métodos modernos.

Para os métodos de Lagrangeano aumentado propostos, provamos que, sob condições suficientes de segunda ordem a função Lagrangeano aumentado tem minimizador local. Sendo assim, os novos métodos constituem metodologias alternativas para a resolução de problemas de minimização com restrições de igualdade.

Os resultados numéricos para os 124 problemas da coleção CUTEr de dimensões variadas, apresentados no Capítulo 3, indicaram que os métodos de Lagrangeano aumentado modernos, estabelecidos por meio dos algoritmos LAPM e LAPB, são competitivos e promissores quando comparados com método clássico (algoritmo LAPC), pois apresentaram melhores resultados com base nos cinco critérios avaliados: número de iterações no passo externo, número de iterações no passo interno, número de avaliações de funções e de gradientes, e tempo computacional.

Outro aspecto importante a destacar é sobre o parâmetro de penalidade. Mostramos que, para uma certa escolha dos parâmetros de penalidade as metodologias propostas coincidem com o método clássico de Hestenes [18] e Powell [40]. Além disso, observando o comportamento geométrico das penalidades, desenvolvemos heurísticas para a atualização dos parâmetros de penalidade, que por sua vez forneceram bons resultados numéricos em

relação ao método convencional.

Conforme mencionamos inicialmente, neste trabalho também propomos um novo algoritmo (algoritmo nHLRF) baseado no algoritmo HLRF. O algoritmo nHLRF faz uma busca linear na direção HLRF e com base nas condições Wolfe, minimiza uma nova função de mérito diferenciável para a obtenção do comprimento do passo que será dado na direção de busca, para a determinação do próximo iterando. Mostramos que, sob certas suposições, o algoritmo gera uma sequência de pontos que converge para um minimizador local do problema. Essas suposições, no algoritmo, foram convertidas em regras para a atualização dos parâmetros nHLRF.

Nota-se que na literatura, especificamente nos trabalhos de Hasofer e Lind [17], Rackwitz e Fiessler [41] e Santosh *et al.* [50], não são apresentados resultados de convergência para os respectivos algoritmos, HLRF e iHLRF, o que torna os resultados teóricos do algoritmo nHLRF ainda mais relevantes.

Com o objetivo de investigar a eficiência e a robustez dos novos algoritmos apresentamos no Capítulo 4 a aplicação destes a problemas de confiabilidade estrutural, e os resultados foram comparados aos obtidos com os algoritmos LAPC, HLRF e iHLRF. Observamos que o algoritmo nHLRF apresentou um desempenho semelhante ao algoritmo iHLRF com relação ao número de iterações e avaliações da função desempenho. Entretanto, o algoritmo nHLRF solucionou um maior número de problemas, comparado com os algoritmos HLRF e iHLRF.

Também pudemos perceber que os algoritmos LAPM e LAPB foram mais eficientes, com base nos critérios avaliados, na resolução dos problemas de confiabilidade estrutural do que o algoritmo de Lagrangeano aumentado clássico, LAPC. Isso nos leva a concluir que as penalidades quadráticas modernas, introduzidas através dos dois novos métodos de Lagrangeano aumentado, são mais robustas e mais eficientes do que a penalidade "clássica", considerada no estudo comparativo de Liu e Kiureghian [26]. Contudo, isso torna os resultados apresentados em [26] obsoletos em relação ao desempenho comparativo dos métodos de Lagrangeano aumentado na resolução de problemas de confiabilidade estrutural.

Os testes também indicaram que os algoritmos LAPC, LAPM e LAPB requerem uma quantidade maior de avaliações de funções e gradientes que os algoritmos comumente empregados nesse contexto, HLRF e iHLRF. Porém, nenhuma conclusão definitiva com respeito a robustez e o desempenho dos métodos propostos pode ser feita, pois os resultados numéricos apresentados aqui são baseados em um conjunto limitado de problemas de confiabilidade estrutural. Entretanto, vimos que aplicação dos métodos de Lagrangeano aumentado é mais vantajosa para os problemas que envolvem grande número de variáveis aleatórias e, também, na resolução de problemas em que durante o processo iterativo $\nabla h(\mathbf{y}^k) = 0$, $\forall k \in \mathbb{N}$, já que os algoritmos baseados em HLRF necessitam atender à suposição de que o gradiente da função desempenho no ponto corrente seja não nulo. Tais

condições tornam os métodos de Lagrangeano aumentado um atrativo para problemas práticos de engenharia.

Para finalizar, gostaríamos de observar que as informações contidas nos capítulos 2 e 4 desse trabalho, referem-se essencialmente ao conteúdo do artigo de Santos, Matioli e Beck [48] intitulado "*New optimization algorithms for structural reliability analysis*" e que foi aceito para publicação no início de 2012 pela revista "*Computer Modeling in Engineering and Science*".

Sugestões para futuros trabalhos

Como pesquisas futuras, sugerimos a investigação de outras escolhas para a atualização do parâmetro de penalidade, incluindo estudos teóricos que embasam as declarações acerca do desempenho dos métodos modernos que não puderam ser comprovadas neste trabalho. Além disso, recomendamos um estudo sobre algoritmos mais adequados para a resolução dos subproblemas irrestritos gerados na abordagem dos métodos de Lagrangeano aumentado pois, acreditamos que, a eficiência de tais métodos possa ser significativamente melhorada.

Referências Bibliográficas

[1] A. T. Beck. *Curso de confiabilidade estrutural*. Universidade de São Paulo - Escola de Engenharia de São Carlos: Notas de aula, 2010.

[2] A. Ben-Tal e M. Zibulevsky. Penalty-barrier multiplier methods for convex programming problems. *SIAM Journal on Optimization*, 7:347–366, 1997.

[3] D. P. Bertsekas. Multiplier methods: a survey. *Automatica*, 12:133–145, 1997.

[4] E. G. Birgin, R. Castillo e J. M. Martínez. Numerical comparison of augmented lagrangian algorithms for nonconvex problems. *Computational Optimization and Applications*, 31:31–56, 2005.

[5] A. Borri e E. Speranzini. Structural reliability analysis using a standard deterministic finite element code. *Structural Safety*, 19:361–282, 1997.

[6] L. Bregman. The relaxation method for finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computation Mathematics and Mathematical Physics*, 7:200–217, 1967.

[7] Y. Censor e S. Zenios. The proximal minimization algorithm with d-functions. *Journal Optimization Theory and Applications*, 73:451–464, 1992.

[8] E. D. Dolang e J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91:201–2013, 2002.

[9] C. Chen e M. Teboulle. Convergence analysis of a proximal-like minimization algorithm using bregman function. *SIAM Journal on Optimization*, 3:538–543, 1993.

[10] N. I. M. Gould, D. Orban e Ph. L. Toint. CUTEr, a constrained and unconstrained testing environment, revisited. *ACM Transactions on Mathematical Software*, 29(4):373–394, 2003.

[11] A. De Pierro e A. Iusem. A relaxed version of Bregman's method for convex programming. *Journal of Optimization Theory and Applications*, 5:421–440, 1986.

[12] J. E. Dennis e R. D. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*. Society for Industrial and Applied Mathematics, Philadelphia, 1996.

[13] O. Ditlevsen. Principle of normal tail approximation. *Journal of the Engineering Mechanics Division*, 107(EM6):1191–1207, 1981.

[14] J. Eckstein. Nonlinear proximal algorithms using bregman functions with applications to convex programming. *Mathematics of Operations Research*, 18:202–226, 1993.

[15] F. Grooteman. Adaptive radial-base importance sampling method for structural reliability. *Structural Safety*, 30:533–542, 2008.

[16] A. Haldar e S. Mahadevan. *Probability, Reliability and Statistical Methods in Engineering Design*. John Wiley & Sons, New York, 2000.

[17] A. M. Hasofer e N. C. Lind. Exact and invariant second moment code format. *Journal of Engineering Mechanics*, 100(1):111–121, 1974.

[18] M. Hestenes. Multiplier and gradient methods. *Journal Optimization Theory and Applications*, 4:303–320, 1969.

[19] A. Iusem e M. Teboulle. On the convergence rate of entropic proximal optimization methods. *Computational and Applied Mathematics*, 12:153–168, 1993.

[20] A. Iusem e M. Teboulle. Convergence rate of nonquadratic proximal point methods for convex and linear programming. *Mathematics of Operations Research*, 20:657–677, 1995.

[21] A. Iusem, M. Teboulle e B. Svaiter. Entropy-like proximal methods in convex programming. *Mathematics of Operations Research*, 19(4):790–814, 1994.

[22] A. Izmailov e M. Solodov. *Otimização volume 1 - Condições de otimalidade, elementos de análise convexa e de dualidade*. IMPA, Rio de Janeiro, 2005.

[23] E. W. Karas, A. P. Oening e A. A. Ribeiro. Global convergence of slanting filter methods for nonlinear programming. *Applied Mathematics and Computation*, 200(2):486–500, 2008.

[24] K. Kiwiel. Proximal minimization methods with generalized bregman functions. *SIAM Journal on Control and Optimization*, 35(4):1142–1168, 1997.

[25] P. L. Liu e A. Der Kiureghian. Optimization algorithms for structural reliability analysis. Technical Report UCB/SESM 86/09, Department of Civil Engineering, University of California, Berkeley, 1986.

[26] P. L. Liu e A. Der Kiureghian. Optimization algorithms for structural reliability. *Structural Safety*, 9:161–177, 1991.

[27] D. G. Luenberger. *Linear and Nonlinear Programming*. Addison - Wesley Publishing Company, New York, 1986.

[28] S. Mahadevan e P. Shi. Multiple linearization method for nonlinear reliability analysis. *Journal of Engineering Mechanics*, 127(11):1165–1173, 2001.

[29] J. M. Martínez. Box-quacan and the implementation of Augmented Lagrangian algorithms for minimization with inequality constraints. *Computational and Applied Mathematics*, 19:31–56, 2000.

[30] J. M. Martínez e E. A. Pilotta. Inexact restoration algorithm for constrained optimization. *Journal of Optimization Theory and Applications*, 104:135–163, 2000.

[31] J. M. Martínez e S. A. Santos. *Métodos Computacionais de Otimização*. 20º Colóquio Brasileiro de Matemática, Rio de Janeiro, IMPA, 1995.

[32] L. C. Matioli e C. C. Gonzaga. A new family of penalty for Augmented Lagrangian methods. *Numerical linear algebra with applications*, 15:925–944, 2008.

[33] R. E. Melchers. *Structural reliability analysis and prediction*. John Wiley & Sons, New York, 2nd edition, 1999.

[34] A. Nataf. Determination des distribution dont les marges sont donnees. *Comptes Rendus de l'Academie des Sciences*, 225:42–43, 1962.

[35] J. Nocedal e S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer-Verlag, New York, 1999.

[36] G. A. Periçaro. *Algoritmos de filtro globalmente convergentes: teoria, implementação e aplicação*. Tese de doutorado, Programa de Pós-Graduação em Métodos Numéricos em Engenharia, Programação Matemática, Universidade Federal do Paraná, 2011.

[37] G. A. Periçaro, S. R. Santos, A. A. Ribeiro e L. C. Matioli. Comparação entre algoritmos de programação não linear aplicados ao problema de confiabilidade estrutural. *XLIII Simpósio Brasileiro de Pesquisa Operacional*, Ubatuba, São Paulo, 2011.

[38] R. Polyak. Modified barrier functions: theory and methods. *Mathematical Programming*, 54:177–222, 1992.

[39] R. Polyak e M. Teboulle. Nonlinear rescaling and proximal-like methods in convex optimization. *Mathematical Programming*, 76:265–284, 1997.

[40] M. J. D. Powell. A method for nonlinear constraints in minimizations problems in optimization. *R. Fletcher*, 283–298, 1969.

[41] R. Rackwitz e B. Fiessler. Structural reliability under combined load sequences. *Computers & Structures*, 9:489–494, 1978.

[42] R. T. Rockafellar. Augmented Lagrangians and applications of the proximal point algorithm in convex programming. *Mathematics of Operations Research*, 2(1):97–116, 1976.

[43] R. T. Rockafellar. Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization*, 14(5):877–898, 1976.

[44] A. C. Rômulo e C. C. Gonzaga. A nonlinear programming algorithm based on non-coercive penalty functions. *Mathematical Programming*, 96(1):87–101, 2003.

[45] J. B. Rosen. The gradient projection method for nonlinear programming: part II, nonlinear constraints. *SIAM Journal on Applied Mathematics*, 9:514–532, 1961.

[46] M. Rosenblatt. Remarks on a multivariate transformation. *Annals of Mathematical Statistics*, 23:470–472, 1952.

[47] S. R. Santos e L. C. Matioli. Desenvolvimento de algoritmos matemáticos aplicados a confiabilidade estrutural. *Mecánica Computacional*, volume XXIX, 683–697, 2010.

[48] S. R. Santos, L. C. Matioli e A. T. Beck. New optimization algorithms for structural reliability analysis. *Computer Modeling in Engineering and Science*, 83(1):23–56, 2012.

[49] S. R. Santos e L. C. Matioli. Two new Augmented Lagrangian algorithms with quadratic penalty for equality problems. Technical report, Department of Mathematics, Federal University of Paraná, http://people.ufpr.br/ matioli/minhahome/pesquisa.html, 2011.

[50] T. Santosh, R. Saraf, A. Ghosh e H. Kushwaka. Optimum step length selection rule in modified HL–RF method for structural reliability. *International Journal of Pressure Vessels and Piping*, 83:742–748, 2006.

[51] M. Teboulle. Entropic proximal mappings with applications to nonlinear programming. *Mathematics of Operations Research*, 17:97–116, 1992.

[52] A. J. Torii e R. H. Lopez. Reliability analysis of water distribution networks using the adaptive response surface approach. *Journal of Hydraulic Engineering, DOI:10.1061/(ASCE)HY.1943-7900.0000504*, 2012.

[53] P. Tseng e D. Bertsekas. On the convergence of exponential multiplier method for convex programming. *Mathematical Programming*, 60:1–19, 1993.

[54] L. Wang e R. V. Grandhi. Safety index calculations using intervening variables for structural reliability. *Computers and Structures*, 59:1139–1148, 1996.

[55] L. Wang e R. V. Grandhi. Higher-order failure probability calculation using nonlinear approximations. *Computer methods in applied mechanics and engineering*, 168:185–206, 1999.

[56] L. Wang e R. V. Grandhi. Structural reliability analysis and optimization: use of approximations. Technical Report 209154, NASA/CR, 1999.

[57] W. I. Zangwill. Nonlinear programming via penalty functions. *Management Science*, 13:344–358, 1967.

[58] Y. Zhang e A. D. Kiureghian. Finite element reliability methods for inelastic structures. Technical report, Department of Civil and Environmental Engineering , University of California, Berkeley, 1997.

Apêndice A

Problemas selecionados da coleção CUTEr

Apresentamos a seguir a relação dos 124 problemas selecionados da coleção CUTEr, juntamente com seus respectivos números de variáveis (n) e de restrições de igualdade (m).

Problema	n	m	**Problema**	n	m	**Problema**	n	m
ARGTRIG	200	200	GRIDNETE	924	484	LUKVLE6	9	4
BROWNALE	200	200	GRIDNETH	264	144	LUKVLE6	99	49
BT1	2	1	GRIDNETH	924	484	LUKVLE6	999	499
BT2	3	1	HEART6	6	6	LUKVLE7	10	4
BT3	5	3	HEART8	8	8	LUKVLE7	50	4
BT4	3	2	HIMMELBA	2	2	LUKVLE7	100	4
BT5	3	2	HIMMELBC	2	2	LUKVLE9	10	6
BT6	5	2	HIMMELBE	3	3	LUKVLE9	50	6
BT7	5	3	HS6	2	1	LUKVLE9	100	6
BT8	5	2	HS7	2	1	LUKVLE11	49	30
BT9	4	2	HS8	2	2	LUKVLE11	98	64
BT10	2	2	HS9	2	1	LUKVLE11	998	664
BT11	5	3	HS11	2	1	LUKVLE13	49	30
BT12	5	3	HS26	3	1	LUKVLE13	98	64
BYRDSPHR	3	2	HS27	3	1	LUKVLE13	998	664
CLUSTER	2	2	HS28	3	1	LUKVLE15	57	42
COOLHANS	9	9	HS39	4	2	LUKVLE15	97	72
CUBENE	2	2	HS40	4	3	LUKVLE15	997	747
DIXCHLNG	10	5	HS42	4	2	LUKVLE16	57	42
EIGENA2	6	3	HS46	5	2	LUKVLE16	97	72
EIGENA2	110	55	HS47	5	3	LUKVLE16	197	147

continua na próxima página

continuação da página anterior

Problema	n	m	**Problema**	n	m	**Problema**	n	m
EIGENB	110	110	HS48	5	2	MARATOS	2	1
EIGENB2	6	3	HS49	5	2	MSS1	90	73
EIGENB2	110	55	HS50	5	3	MWRIGHT	5	3
EIGENB2	420	210	HS51	5	3	ORTHRDM2	53	25
EIGENBCO	6	3	HS52	5	3	ORTHRDM2	203	100
EIGENBCO	110	55	HS56	7	4	ORTHREGA	133	64
EIGENBCO	650	325	HS61	3	2	ORTHREGB	27	6
EIGENC	30	30	HS77	5	2	ORTHREGC	25	10
EIGENC	462	462	HS78	5	3	ORTHREGC	105	50
EIGENC2	30	15	HS79	5	3	ORTHREGC	505	250
EIGENC2	462	231	HS100LNP	7	2	ORTHREGC	1005	500
ELEC	75	25	HS111LNP	10	3	ORTHREGD	103	50
ELEC	150	50	HYDCAR6	29	29	ORTHRGDM	103	50
ELEC	300	100	HYDCAR20	99	99	ORTHRGDM	155	76
GENHS28	5	3	HYPCIR	2	2	POWELLBS	2	2
GENHS28	10	8	LUKVLE1	100	98	POWELLSQ	2	2
GENHS28	15	13	LUKVLE1	1000	998	RECIPE	3	3
GOTTFR	2	2	LUKVLE3	50	2	RSNBRNE	2	2
GRIDNETE	60	36	LUKVLE3	100	2	S316-322	2	1
GRIDNETE	180	100	LUKVLE3	1000	2	SINVALNE	2	2
GRIDNETE	612	324						

Apêndice B

Resultados para os problemas da coleção CUTEr

A seguir apresentamos os resultados dos Algoritmos LAPC, LAPM e LAPB para os problemas da coleção CUTEr. Para cada problema, os resultados do algortimo LAPC são mostrados na primeira linha, do LAPM na segunda linha e do LAPB na terceira linha. Usamos *it.ext* para representar o número de iterações no passo externo, *it.int* para representar o número de iterações no passo interno, *fun* o número total de avaliações de funções, *grad* o número total de avaliações de derivadas, $f(\bar{x})$ para indicar o valor da função objetivo na solução ou no último iterando (caso o problema não tenha sido solucionado) e *tempo* refere-se ao tempo computacional dispendido na resolução do problema pelo algoritmo em questão. Valores atribuídos para *saída* indicam o critério de parada atendido, conforme mencionado na seção 3.3.4. Por exemplo, para o primeiro problema, ARGTRIG, os três algoritmos satisfizeram o critério de parada (3.21), pois o valor atribuído para *saída* é zero.

Os problemas nos quais aparecem o símbolo ***, são aqueles que apresentaram um valor não numérico (NaN) durante o processo iterativo e, neste caso, o valor atribuído ao parâmetro de saída é 3.

Problema (n, m)	*Método*	*it.ext*	*it.int*	*fun*	*grad*	*tempo* (*segundos*)	$f(\bar{x})$	*saída*
ARGTRIG	LAPC	2	915	921	921	42,7	0	0
(200,200)	LAPM	15	940	1050	1050	24,2	0	0
	LAPB	15	940	1050	1050	21,6	0	0
BROWNALE	LAPC	3	113	151	151	5,2	0	0
(200,200)	LAPM	3	73	94	94	1,7	0	0
	LAPB	3	73	94	94	1,4	0	0
BT1	LAPC	9	27	56	56	3	-1	0
(2,1)	LAPM	4	17	32	32	1	-1	0
	LAPB	6	17	40	40	0,7	-1	0
BT2	LAPC	3	98	118	118	1,3	2	0

Continuação na próxima página

continuação da página anterior

Problema (n, m)	*Método*	*it.ext*	*it.int*	*fun*	*grad*	*tempo* (*segundos*)	$f(\bar{x})$	*saída*
(3,1)	LAPM	4	109	150	150	0,5	2	0
	LAPB	4	109	150	150	0,6	2,2	0
BT3	LAPC	9	131	166	166	2,6	4	0
(5,3)	LAPM	7	127	155	155	0,7	4	0
	LAPB	8	116	155	155	0,8	4,1	0
BT4	LAPC	8	129	160	160	2,4	-46	0
(3,2)	LAPM	5	59	84	84	0,4	-46	0
	LAPB	7	76	104	104	0,6	-45,5	0
BT5	LAPC	4	40	63	63	1,2	962	0
(3,2)	LAPM	3	41	58	58	0,2	962	0
	LAPB	3	41	58	58	0,3	961,7	0
BT6	LAPC	5	176	199	199	2,7	0	0
(5,2)	LAPM	4	141	167	167	0,5	0	0
	LAPB	4	141	167	167	0,6	0,3	0
BT7	LAPC	18	253	334	334	5,5	306	0
(5,3)	LAPM	8	140	187	187	0,9	306	0
	LAPB	13	197	270	270	1,2	306,5	0
BT8	LAPC	5	25	37	37	1,1	1	0
(5,2)	LAPM	2	60	74	74	0,2	1	0
	LAPB	2	60	74	74	0,2	1	0
BT9	LAPC	8	114	153	153	2,4	-1	0
(4,2)	LAPM	6	101	127	127	0,6	-1	0
	LAPB	6	101	127	127	0,6	-1	0
BT10	LAPC	8	85	119	119	2,1	-1	0
(2,2)	LAPM	6	73	98	98	0,5	-1	0
	LAPB	6	73	98	98	0,5	-1	0
BT11	LAPC	8	151	174	174	2,5	1	0
(5,3)	LAPM	8	217	248	248	1	1	0
	LAPB	9	239	279	279	1,1	0,8	0
BT12	LAPC	4	111	136	136	1,6	6	0
(5,3)	LAPM	3	118	138	138	0,4	6	0
	LAPB	3	118	138	138	0,5	6,2	0
BYRDSPHR	LAPC	6	62	85	85	1,7	-5	0
(3,2)	LAPM	4	47	72	72	0,3	-5	0
	LAPB	4	47	72	72	0,4	-4,7	0
CLUSTER	LAPC	101	1013	1754	1754	28,2	0	1
(2,2)	LAPM	3	508	568	568	1,5	0	0
	LAPB	101	634	946	946	8	0	1
COOLHANS	LAPC	12	74	98	98	2,6	0	0
(9,9)	LAPM	10	81	111	111	0,8	0	0
	LAPB	10	81	111	111	0,7	0	0
CUBENE	LAPC	2	40	54	54	0,6	0	0

Continuação na próxima página

continuação da página anterior

Problema (n, m)	*Método*	*it.ext*	*it.int*	*fun*	*grad*	*tempo* (*segundos*)	$f(\bar{x})$	*saída*
(2,2)	LAPM	2	49	65	65	0,2	0	0
	LAPB	2	49	65	65	0,2	0	0
DIXCHLNG	LAPC	101	487	1164	1164	22,8	0	1
(10,5)	LAPM	101	969	1380	1380	8,4	0	1
	LAPB	101	884	1297	1297	8,1	0	1
EIGENA2	LAPC	6	77	96	96	1,7	0	0
(6,3)	LAPM	4	45	62	62	0,3	0	0
	LAPB	4	45	62	62	0,4	0	0
EIGENA2	LAPC	10	1077	1296	1296	20,6	0	0
(110,55)	LAPM	14	1707	1783	1783	10,5	0	0
	LAPB	14	1392	1485	1485	9,3	0	0
EIGENB	LAPC	101	22197	22677	22677	397,3	0	1
(110,110)	LAPM	101	2030	2291	2291	18,7	0	1
	LAPB	101	10407	10743	10743	66,4	0	1
EIGENB2	LAPC	7	87	153	153	2,1	0	0
(6,3)	LAPM	16	399	658	658	2,3	0	0
	LAPB	101	131	627	627	6,7	0	1
EIGENB2	LAPC	9	703	808	808	13,4	0	0
(110,55)	LAPM	6	1000	1079	1079	6,1	0	0
	LAPB	6	943	1016	1016	5,7	0	0
EIGENB2	LAPC	9	2606	2721	2721	270,4	0	0
(420,210)	LAPM	7	2267	2342	2342	137,8	0	0
	LAPB	7	2096	2171	2171	106,7	0	0
EIGENBCO	LAPC	4	123	175	175	1,7	0	0
(6,3)	LAPM	8	83	119	119	0,7	1	0
	LAPB	8	103	139	139	0,7	1	0
EIGENBCO	LAPC	6	2659	2748	2748	45,6	0	0
(110,55)	LAPM	4	1400	1468	1468	8,6	0	0
	LAPB	4	1401	1469	1469	8,3	0	0
EIGENBCO	LAPC	11	5415	5500	5500	1212,1	0	0
(650,325)	LAPM	13	5890	6007	6007	475,4	0	0
	LAPB	13	5899	6007	6007	480,3	0	0
EIGENC	LAPC	101	17885	20188	20188	178,7	0	1
(30,30)	LAPM	2	228	239	239	0,7	0	0
	LAPB	2	228	239	239	0,6	0	0
EIGENC	LAPC	101	22119	22673	22673	3267,3	0	1
(462,462)	LAPM	101	3328	3782	3782	209,6	0	1
	LAPB	11	4474	4735	4735	225,2	0	0
EIGENC2	LAPC	9	582	657	657	6,8	0	0
(30,15)	LAPM	8	195	225	225	0,9	0	0
	LAPB	8	195	225	225	1,1	0	0
EIGENC2	LAPC	13	3645	4166	4166	477,2	0	0

Continuação na próxima página

continuação da página anterior

Problema (n, m)	*Método*	*it.ext*	*it.int*	*fun*	*grad*	*tempo* (*segundos*)	$f(\bar{x})$	*saída*
(462,231)	LAPM	101	2474	2788	2788	149,4	119	1
	LAPB	44	4241	4421	4421	257	0	0
ELEC	LAPC	9	460	491	491	7,2	244	0
(75,25)	LAPM	8	390	424	424	2	244	0
	LAPB	9	358	391	391	2	243,8	0
ELEC	LAPC	10	544	575	575	15,5	1055	0
(150,50)	LAPM	7	425	450	450	6,4	1055	0
	LAPB	9	427	459	459	4,9	1055,2	0
ELEC	LAPC	11	3162	3221	3221	212,6	4448	0
(300,100)	LAPM	8	728	771	771	22,1	4448	0
	LAPB	10	834	880	880	33,1	4448,4	0
GENHS28	LAPC	5	112	124	124	1,8	0	0
(5,3)	LAPM	5	96	112	112	0,4	0	0
	LAPB	5	96	112	112	0,5	0,4	0
GENHS28	LAPC	6	211	226	226	2,7	1	0
(10,8)	LAPM	5	166	182	182	0,6	1	0
	LAPB	5	166	182	182	0,7	0,9	0
GENHS28	LAPC	5	216	228	228	2,4	1	0
(15,13)	LAPM	5	198	214	214	0,7	1	0
	LAPB	5	198	214	214	0,8	1,5	0
GOTTFR	LAPC	2	40	47	47	0,6	0	0
(2,2)	LAPM	2	27	34	34	0,1	0	0
	LAPB	2	27	34	34	0,2	0	0
GRIDNETE	LAPC	7	840	855	855	9,3	40	0
(60,36)	LAPM	6	769	786	786	2,7	40	0
	LAPB	7	782	800	800	3	39,6	0
GRIDNETE	LAPC	8	1836	1854	1854	58,7	51	0
(180,100)	LAPM	7	1799	1832	1832	34,5	51	0
	LAPB	7	1846	1870	1870	24,9	50,6	0
GRIDNETE	LAPC	10	3142	3171	3171	644	76	0
(612,324)	LAPM	8	3116	3171	3171	275	76	0
	LAPB	8	3160	3315	3315	252,8	75,6	0
GRIDNETE	LAPC	10	3622	3652	3652	1926,8	87	0
(924,484)	LAPM	9	3736	3830	3830	607,5	87	0
	LAPB	9	3735	4016	4016	507,7	87,3	0
GRIDNETH	LAPC	8	2115	2134	2134	157,1	57	0
(264,144)	LAPM	7	2126	2157	2157	65,6	57	0
	LAPB	7	2284	2350	2350	54,2	57,1	0
GRIDNETH	LAPC	10	3533	3565	3565	1886,2	87	0
(924,484)	LAPM	9	3716	3811	3811	646,4	87	0
	LAPB	9	3736	4013	4013	507,2	87,3	0
HEART6	LAPC	101	488	1081	1081	21,3	0	1

Continuação na próxima página

continuação da página anterior

Problema (n, m)	*Método*	*it.ext*	*it.int*	*fun*	*grad*	*tempo* (*segundos*)	$f(\bar{x})$	*saída*
(6,6)	LAPM	101	540	942	942	7,7	0	1
	LAPB	101	621	1078	1078	7,9	0	1
HEART8	LAPC	5	1622	2016	2016	15,3	0	0
(8,8)	LAPM	5	1303	1616	1616	3,7	0	0
	LAPB	7	2058	2567	2567	5,9	0	0
HIMMELBA	LAPC	2	11	16	16	0,5	0	0
(2,2)	LAPM	2	11	17	17	0,1	0	0
	LAPB	2	11	17	17	0,1	0	0
HIMMELBC	LAPC	2	16	20	20	0,5	0	0
(2,2)	LAPM	2	14	20	20	0,2	0	0
	LAPB	2	14	20	20	0,1	0	0
HIMMELBE	LAPC	2	23	28	28	0,5	0	0
(3,3)	LAPM	2	21	24	24	0,1	0	0
	LAPB	2	21	24	24	0,1	0	0
HS6	LAPC	2	46	56	56	0,7	0	0
(2,1)	LAPM	2	58	86	86	0,2	0	0
	LAPB	2	58	86	86	0,2	0	0
HS7	LAPC	4	32	39	39	1	-2	0
(2,1)	LAPM	3	29	42	42	0,2	-2	0
	LAPB	3	29	42	42	0,2	-1,7	0
HS8	LAPC	2	15	21	21	0,5	-1	0
(2,2)	LAPM	2	14	21	21	0,1	-1	0
	LAPB	2	14	21	21	0,1	-1	0
HS9	LAPC	2	11	21	21	0,5	-1	0
(2,1)	LAPM	2	8	22	22	0,1	-1	0
	LAPB	2	8	22	22	0,1	-0,5	0
HS11	LAPC	6	50	66	66	1,6	-8	0
(2,1)	LAPM	6	46	65	65	0,4	-8	0
	LAPB	6	46	65	65	0,4	-8,5	0
HS26	LAPC	2	84	92	92	1	0	0
(3,1)	LAPM	2	81	97	97	0,3	0	0
	LAPB	2	81	97	97	0,3	0	0
HS27	LAPC	3	59	66	66	0,9	0	0
(3,1)	LAPM	3	57	68	68	0,3	0	0
	LAPB	3	57	68	68	0,3	0	0
HS28	LAPC	2	18	24	24	0,5	0	0
(3,1)	LAPM	2	21	33	33	0,1	0	0
	LAPB	2	21	33	33	0,1	0	0
HS39	LAPC	8	114	153	153	2,4	-1	0
(4,2)	LAPM	6	101	127	127	0,5	-1	0
	LAPB	6	101	127	127	0,5	-1	0
HS40	LAPC	7	124	164	164	2,2	0	0

Continuação na próxima página

continuação da página anterior

Problema (n, m)	*Método*	*it.ext*	*it.int*	*fun*	*grad*	*tempo* (*segundos*)	$f(\bar{x})$	*saída*
(4,3)	LAPM	4	63	76	76	0,3	0	0
	LAPB	4	63	76	76	0,3	-0,2	0
HS42	LAPC	7	42	63	63	1,7	14	0
(4,2)	LAPM	6	36	57	57	0,4	14	0
	LAPB	7	40	63	63	0,5	13,9	0
HS46	LAPC	2	104	114	114	1,1	0	0
(5,2)	LAPM	2	129	137	137	0,4	0	0
	LAPB	2	129	137	137	0,4	0	0
HS47	LAPC	9	294	342	342	4,3	0	0
(5,3)	LAPM	6	211	253	253	0,8	0	0
	LAPB	7	252	305	305	0,9	0	0
HS48	LAPC	2	23	26	26	0,5	0	0
(5,2)	LAPM	2	38	43	43	0,2	0	0
	LAPB	2	38	43	43	0,3	0	0
HS49	LAPC	3	96	103	103	1,3	0	0
(5,2)	LAPM	2	87	94	94	0,3	0	0
	LAPB	2	87	94	94	0,2	0	0
HS50	LAPC	2	53	61	61	0,8	0	0
(5,3)	LAPM	2	56	63	63	0,2	0	0
	LAPB	2	56	63	63	0,2	0	0
HS51	LAPC	7	115	139	139	2,6	0	0
(5,3)	LAPM	5	92	117	117	0,4	0	0
	LAPB	5	92	117	117	0,6	0	0
HS52	LAPC	8	154	174	174	2,7	5	0
(5,3)	LAPM	7	150	167	167	0,7	5	0
	LAPB	8	160	181	181	0,7	5,3	0
HS56	LAPC	5	77	93	93	1,5	-3	0
(7,4)	LAPM	5	95	114	114	0,4	-3	0
	LAPB	6	114	136	136	0,6	-3,5	0
HS61	LAPC	5	50	64	64	1,4	-144	0
(3,2)	LAPM	4	38	54	54	0,3	-144	0
	LAPB	4	38	54	54	0,3	-143,6	0
HS77	LAPC	5	140	171	171	2,3	0	0
(5,2)	LAPM	4	145	171	171	0,6	0	0
	LAPB	4	145	171	171	0,5	0,2	0
HS78	LAPC	5	71	88	88	1,6	-3	0
(5,3)	LAPM	4	55	71	71	0,3	-3	0
	LAPB	4	55	71	71	0,3	-2,9	0
HS79	LAPC	5	128	143	143	1,9	0	0
(5,3)	LAPM	4	120	133	133	0,5	0	0
	LAPB	4	120	133	133	0,5	0,1	0
HS100LNP	LAPC	4	183	213	213	2,2	681	0

Continuação na próxima página

continuação da página anterior

Problema (n, m)	*Método*	*it.ext*	*it.int*	*fun*	*grad*	*tempo* (*segundos*)	$f(\bar{x})$	*saída*
(7,2)	LAPM	4	163	192	192	0,6	681	0
	LAPB	4	163	192	192	0,6	680,6	0
HS111LNP	LAPC	8	612	642	642	6	-48	0
(10,3)	LAPM	4	330	347	347	1	-48	0
	LAPB	5	396	416	416	1,2	-47,8	0
HYDCAR6	LAPC	101	6534	7484	7484	78,6	0	1
(29,29)	LAPM	101	13509	14704	14704	41,8	0	1
	LAPB	101	10634	11766	11766	33,4	0	1
HYDCAR20	LAPC	101	20786	21225	21225	354,9	0	1
(99,99)	LAPM	101	3553	3782	3782	23,9	0	1
	LAPB	101	8476	8794	8794	48,5	0	1
HYPCIR	LAPC	2	12	17	17	0,4	0	0
(2,2)	LAPM	2	15	20	20	0,1	0	0
	LAPB	2	15	20	20	0,1	0	0
LUKVLE1	LAPC	8	170	231	231	4,5	0	0
(100,98)	LAPM	9	138	183	183	1,3	6	0
	LAPB	10	163	217	217	1,5	6,2	0
LUKVLE1	LAPC	8	170	241	241	135,7	0	0
(1000,998)	LAPM	101	239	916	916	66,1	6	1
	LAPB	101	261	1063	1063	60,9	6,2	1
LUKVLE3	LAPC	12	505	542	542	6,6	694	0
(50,2)	LAPM	7	367	405	405	1,5	28	0
	LAPB	9	401	440	440	1,8	27,6	0
LUKVLE3	LAPC	12	467	504	504	8,7	694	0
(100,2)	LAPM	7	332	369	369	1,9	28	0
	LAPB	9	366	404	404	2,1	27,6	0
LUKVLE3	LAPC	12	464	501	501	234,1	694	0
(1000,2)	LAPM	7	338	374	374	60,1	28	0
	LAPB	9	373	410	410	55	27,6	0
LUKVLE6	LAPC	13	417	463	463	5,3	378	0
(9,4)	LAPM	101	342	1068	1068	6,8	378	1
	LAPB	11	332	380	380	1,4	377,7	0
LUKVLE6	LAPC	101	672	1545	1545	31,3	6038	1
(99,49)	LAPM	101	461	1171	1171	8,1	6038	1
	LAPB	101	538	1379	1379	8,8	6037,7	1
LUKVLE6	LAPC	101	594	1450	1450	424,3	62638	1
(999,499)	LAPM	101	407	1014	1014	81,8	62638	1
	LAPB	101	475	20536	20536	585,9	98345	1
LUKVLE7	LAPC	7	246	276	276	3,2	-2	0
(10,4)	LAPM	7	281	316	316	1	-2	0
	LAPB	7	274	311	311	1	-1,6	0
LUKVLE7	LAPC	9	283	330	330	4,4	-14	0

Continuação na próxima página

continuação da página anterior

Problema (n, m)	*Método*	*it.ext*	*it.int*	*fun*	*grad*	*tempo* (*segundos*)	$f(\bar{x})$	*saída*
(50,4)	LAPM	7	352	406	406	1,5	-14	0
	LAPB	9	402	470	470	1,7	-13,8	0
LUKVLE7	LAPC	10	285	346	346	6,1	-26	0
(100,4)	LAPM	8	378	433	433	2,2	-26	0
	LAPB	9	414	478	478	2,5	-25,9	0
LUKVLE9	LAPC	9	512	565	565	5,7	1	0
(10,6)	LAPM	10	481	537	537	1,7	1	0
	LAPB	10	486	541	541	1,7	1,3	0
LUKVLE9	LAPC	8	546	601	601	6,5	5	0
(50,6)	LAPM	9	563	626	626	2,1	5	0
	LAPB	9	556	622	622	2	5,2	0
LUKVLE9	LAPC	9	543	603	603	9,3	10	0
(100,6)	LAPM	9	565	623	623	3,1	10	0
	LAPB	10	567	630	630	3,1	10,2	0
LUKVLE11	LAPC	8	463	486	486	5,9	0	0
(49,30)	LAPM	7	482	502	502	1,7	0	0
	LAPB	7	482	502	502	1,7	0	0
LUKVLE11	LAPC	9	661	692	692	11,5	0	0
(98,64)	LAPM	7	683	703	703	3,5	0	0
	LAPB	7	683	703	703	3,4	0	0
LUKVLE11	LAPC	10	787	817	817	485,3	0	0
(998,664)	LAPM	8	924	950	950	172,9	0	0
	LAPB	8	924	950	950	146,2	0	0
LUKVLE13	LAPC	101	1580	2189	2189	32,8	320	1
(49,30)	LAPM	101	1598	1995	1995	10,7	323	1
	LAPB	101	2210	2727	2727	12,3	323,1	1
LUKVLE13	LAPC	101	2661	3248	3248	58,1	777	1
(98,64)	LAPM	101	2382	2656	2656	17,4	775	1
	LAPB	101	3706	4158	4158	22	772,4	1
LUKVLE13	LAPC	101	4097	4771	4771	2870,8	6734	1
(998,664)	LAPM	101	2676	2893	2893	487,9	9113	1
	LAPB	101	13571	14069	14069	2533,4	9007,3	1
LUKVLE15	LAPC	6	845	865	865	9,5	0	0
(57,42)	LAPM	6	1260	1282	1282	4,1	0	0
	LAPB	6	1260	1282	1282	4,1	0	0
LUKVLE15	LAPC	8	1890	1957	1957	29,8	0	0
(97,72)	LAPM	6	1462	1493	1493	6,8	0	0
	LAPB	6	1462	1493	1493	6,8	0	0
LUKVLE15	LAPC	12	5921	6000	6000	3743,8	0	2
(997,747)	LAPM	42	20719	21000	21000	3677,5	0	2
	LAPB	38	18471	19000	19000	3698,8	0	2
LUKVLE16	LAPC	9	745	772	772	8,8	0	0

Continuação na próxima página

continuação da página anterior

Problema (n, m)	*Método*	*it.ext*	*it.int*	*fun*	*grad*	*tempo* (*segundos*)	$f(\bar{x})$	*saída*
(57,42)	LAPM	7	639	664	664	2,4	0	0
	LAPB	7	639	664	664	2,3	0	0
LUKVLE16	LAPC	9	804	831	831	13	0	0
(97,72)	LAPM	7	762	787	787	3,8	0	0
	LAPB	7	762	787	787	3,7	0	0
LUKVLE16	LAPC	9	992	1019	1019	56,6	0	0
(197,147)	LAPM	8	811	843	843	14	0	0
	LAPB	8	811	843	843	14,7	0	0
MARATOS	LAPC	5	26	36	36	1,1	-1	0
(2,1)	LAPM	3	16	25	25	0,2	-1	0
	LAPB	3	16	25	25	0,2	-1	0
MSS1	LAPC	***	***	***	***	***	***	3
(90,73)	LAPM	11	88	166	166	1,2	-9	0
	LAPB	17	115	218	218	1,6	-9	0
MWRIGHT	LAPC	7	190	223	223	5,7	1,2884	0
(5,3)	LAPM	5	91	106	106	0,4	25	0
	LAPB	6	114	136	136	0,5	25	0
ORTHRDM2	LAPC	101	201	703	703	21	1,7973	1
(53,25)	LAPM	101	181	1108	1108	6,5	2	1
	LAPB	101	181	1108	1108	6,5	1,8	1
ORTHRDM2	LAPC	101	199	740	740	36,2	7,7758	1
(203,100)	LAPM	101	257	1192	1192	10	8	1
	LAPB	101	257	1192	1192	11,3	7,8	1
ORTHREGA	LAPC	101	1769	2284	2284	75,1	413,4881	1
(133,64)	LAPM	101	2090	2345	2345	23,6	415	1
	LAPB	101	2512	2852	2852	32,8	408,3	1
ORTHREGB	LAPC	2	84	95	95	0,9	0	0
(27,6)	LAPM	3	86	100	100	0,4	0	0
	LAPB	3	86	100	100	0,4	0	0
ORTHREGC	LAPC	7	499	520	520	5,4	0,399	0
(25,10)	LAPM	6	430	448	448	1,3	0	0
	LAPB	6	430	448	448	1,3	0,4	0
ORTHREGC	LAPC	7	1380	1412	1412	22	1,9755	0
(105,50)	LAPM	6	1309	1338	1338	6,6	2	0
	LAPB	6	1309	1338	1338	6,4	2	0
ORTHREGC	LAPC	101	2033	2599	2599	359	9,582	1
(505,250)	LAPM	101	2339	2946	2946	128,6	10	1
	LAPB	101	2214	2820	2820	184,4	9,6	1
ORTHREGC	LAPC	101	2285	2880	2880	1366	18,7907	1
(1005,500)	LAPM	101	2346	3069	3069	399,4	19	1
	LAPB	101	2334	3039	3039	550,3	18,8	1
ORTHREGD	LAPC	101	265	783	783	24,5	15,5906	1

Continuação na próxima página

continuação da página anterior

Problema (n, m)	*Método*	*it.ext*	*it.int*	*fun*	*grad*	*tempo* (*segundos*)	$f(\bar{x})$	*saída*
(103,50)	LAPM	101	339	1343	1343	8	16	1
	LAPB	101	339	1343	1343	8,6	15,6	1
ORTHRGDM	LAPC	101	259	811	811	22,2	15,3701	1
(103,50)	LAPM	101	443	1493	1493	8,8	15	1
	LAPB	101	443	1493	1493	9,8	15,4	1
ORTHRGDM	LAPC	101	296	882	882	31,5	23,3348	1
(155,76)	LAPM	101	237	1149	1149	8,8	23	1
	LAPB	101	237	1149	1149	9,4	23,3	1
POWELLBS	LAPC	101	545	1363	1363	23,3	0	1
(2,2)	LAPM	101	314	1739	1739	7,7	0	1
	LAPB	101	314	1739	1739	8,2	0	1
POWELLSQ	LAPC	3	32	58	58	0,9	0	0
(2,2)	LAPM	3	73	124	124	0,4	0	0
	LAPB	3	73	124	124	0,4	0	0
RECIPE	LAPC	2	52	59	59	0,7	0	0
(3,3)	LAPM	2	52	60	60	0,2	0	0
	LAPB	2	52	60	60	0,2	0	0
RSNBRNE	LAPC	2	49	65	65	0,7	0	0
(2,2)	LAPM	2	48	64	64	0,2	0	0
	LAPB	2	48	64	64	0,3	0	0
S316-322	LAPC	12	52	71	71	3,8	334,2967	0
(2,1)	LAPM	7	27	45	45	0,4	334	0
	LAPB	9	29	51	51	0,8	334,3	0
SINVALNE	LAPC	2	74	100	100	1,2	0	0
(2,2)	LAPM	2	42	63	63	0,2	0	0
	LAPB	2	42	63	63	0,2	0	0

Anexo

Informações referentes ao problema 23 de confiabilidade estrutural

As informações apresentadas a seguir referem-se aos dados extraídos do trabalho de Liu e Kiureghian [25] para o Problema 23 de confiabilidade estrutural, apresentado no Capítulo 4. Apresentamos na Tabela 1 os elementos da estrutura de pórtico, na Tabela 2 as distribuições de probabilidade com seus respectivos parâmetros e, por fim, na Tabela 3 os coeficientes de correlação das 21 variáveis de projeto.

Tabela 1: Elementos da estrutura de pórtico.

Elemento	Módulo de elasticidade	Momento de inércia	Área de seção transversal
B_1	E_4	I_{10}	A_{18}
B_2	E_4	I_{11}	A_{19}
B_3	E_4	I_{12}	A_{20}
B_4	E_4	I_{13}	A_{21}
C_1	E_5	I_6	A_{14}
C_2	E_5	I_7	A_{15}
C_3	E_5	I_8	A_{18}
C_4	E_5	I_9	A_{17}

Tabela 2: Distribuições de probabilidade das variáveis de projeto.

Variável	Distribuição de probabilidade	Média	Desvio padrão
P_1	Rayleigh	30	9,00
P_2	Rayleigh	20	8,00
P_3	Rayleigh	16	6,40
E_4	Gaussiana	454000	40000
E_5	Gaussiana	497000	40000
I_6	Gaussiana	0,94	0,12
I_7	Gaussiana	1,33	0,15
I_8	Gaussiana	2,47	0,30

continua na próxima página

Tabela 2 – continuação da página anterior

Variável	Distribuição de probabilidade	Média	Desvio padrão
I_9	Gaussiana	3,00	0,35
I_{10}	Gaussiana	1,25	0,30
I_{11}	Gaussiana	1,63	0,40
I_{12}	Gaussiana	2,69	0,65
I_{13}	Gaussiana	3,00	0,75
A_{14}	Gaussiana	3,36	0,60
A_{15}	Gaussiana	4,00	0,80
A_{16}	Gaussiana	5,44	1,00
A_{17}	Gaussiana	6,00	1,20
A_{18}	Gaussiana	2,72	1,00
A_{19}	Gaussiana	3,13	1,10
A_{20}	Gaussiana	4,01	1,30
A_{21}	Gaussiana	4,50	1,50

Tabela 3: Coeficientes de correlação das variáveis de projeto do Problema 23.

Variável	P_1	P_2	P_3	E_4	E_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}	I_{13}	A_{14}	A_{15}	A_{16}	A_{17}	A_{18}	A_{19}	A_{20}	A_{21}
P_1	1																				
P_2	0,5	1																			
P_3	0,5	0,5	1																		
E_4	0	0	0	1																	
E_5	0	0	0	0,9	1																
I_6	0	0	0	0	0	1															
I_7	0	0	0	0	0	0,13	1														
I_8	0	0	0	0	0	0,13	0,13	1													
I_9	0	0	0	0	0	0,13	0,13	0,13	1												
I_{10}	0	0	0	0	0	0,13	0,13	0,13	0,13	1											
I_{11}	0	0	0	0	0	0,13	0,13	0,13	0,13	0,13	1										
I_{12}	0	0	0	0	0	0,13	0,13	0,13	0,13	0,13	0,13	1									
I_{13}	0	0	0	0	0	0,13	0,13	0,13	0,13	0,13	0,13	0,13	1								
A_{14}	0	0	0	0	0	0,95	0,13	0,13	0,13	0,13	0,13	0,13	0,13	1							
A_{15}	0	0	0	0	0	0,13	0,95	0,13	0,13	0,13	0,13	0,13	0,13	0,13	1						
A_{16}	0	0	0	0	0	0,13	0,13	0,95	0,13	0,13	0,13	0,13	0,13	0,13	0,13	1					
A_{17}	0	0	0	0	0	0,13	0,13	0,13	0,95	0,13	0,13	0,13	0,13	0,13	0,13	0,13	1				
A_{18}	0	0	0	0	0	0,13	0,13	0,13	0,13	0,95	0,13	0,13	0,13	0,13	0,13	0,13	0,13	1			
A_{19}	0	0	0	0	0	0,13	0,13	0,13	0,13	0,13	0,95	0,13	0,13	0,13	0,13	0,13	0,13	0,13	1		
A_{20}	0	0	0	0	0	0,13	0,13	0,13	0,13	0,13	0,13	0,95	0,13	0,13	0,13	0,13	0,13	0,13	0,13	1	
A_{21}	0	0	0	0	0	0,13	0,13	0,13	0,13	0,13	0,13	0,13	0,95	0,13	0,13	0,13	0,13	0,13	0,13	0,13	1