COMPARAÇÃO ENTRE ALGORITMOS DE PROGRAMAÇÃO NÃO LINEAR APLICADOS AO PROBLEMA DE CONFIABILIDADE ESTRUTURAL

Gislaine Aparecida Periçaro* gpericaro@fecilcam.br

Solange Regina dos Santos* solaregina@gmail.com

Ademir Alves Ribeiro[§] ademir.ribeiro@ufpr.br

Luiz Carlos Matioli[§] matioli@ufpr.br

*Programa de Pós Graduação em Métodos Numéricos em Engenharia, PPGMNE, UFPR
Universidade Estadual do Paraná - Campus Campo Mourão

[§] Universidade Federal do Paraná, Departamento de Matemática, Curitiba - PR

RESUMO

Apresentamos neste trabalho os Métodos Restauração Inexata e Duas Fases utilizados para resolver problemas gerais de programação não linear e analisamos a aplicabilidade destes ao problema de confiabilidade estrutural. Tal problema consiste em determinar o índice de confiabilidade, definido como a mínima distância de uma superfície de falha à origem do sistema de coordenadas, representadas pelas variáveis de projeto no espaço padronizado. Testes numéricos foram realizados e os resultados indicam que os métodos são eficientes e competitivos com o método desenvolvido especialmente para determinar o índice de confiabilidade, HLRF, mas que não possui garantia de convergência.

PALAVRAS-CHAVE. Confiabilidade Estrutural, Método de Restauração Inexata, Método Duas Fases.

PM - Programação Matemática.

ABSTRACT

This paper presents the Inexact Restoration and Two Phases Methods used to solve general nonlinear programming problems and analyzes their applicability to the reliability structural problem. This problem consists of determining the reliability index, defined as the minimum distance from the failure surface to the origin of the coordinate system, represented by design variables in standardized space. Numerical tests have been performed and the results indicate that the methods are efficient and competitive with the specially developed method to determine the reliability index, HLRF, which doesn't have guarantee convergence.

KEYWORDS. Structural Reliability. Inexact Restoration Method. Two Phases Method.

MP - Mathematical Programming.

1. Introdução

Todo projeto de engenharia estrutural tem como propósito garantir um desempenho satisfatório do sistema, com segurança, funcionalidade, durabilidade e outros critérios estabelecidos na fase de pré-projeto. Porém, devido à presença de incertezas associadas às propriedades dos materiais, aos parâmetros de resistência do solo, às propriedades geométricas das formas dos elementos estruturais projetados e aos carregamentos, existe um risco de que a estrutura não atenda a finalidade para a qual foi projetada. Para avaliar este risco, denominado de probabilidade de falha, utiliza-se a metodologia Análise de Confiabilidade Estrutural. Nesta metodologia são empregados métodos de simulação, métodos analíticos ou métodos aproximados. Os métodos analíticos são conhecidos como FORM (*First Order Reliability Method*) e SORM (*Second Order Reliability Method*). Nestes métodos um dos passos fundamentais é a determinação do ponto de projeto, ou ponto mais provável de falha. A determinação desse ponto é um problema de otimização no qual se busca localizar o ponto sobre a superfície de falha que está a uma menor distância da origem de um sistema de coordenadas que representam as variáveis de projeto. A solução desse problema enfrenta dificuldades clássicas de otimização, como por exemplo, a garantia de convergência.

Entre os vários algoritmos desenvolvidos para a determinação do ponto de projeto, destaca-se o algoritmo HLRF elaborado em 1974 por Hasofer e Lind e aperfeiçoado em 1978 por Rackwitz e Fiessler. Na sua forma original, este algoritmo é instável, podendo não convergir em alguns casos. Isto tem impulsionado pesquisadores a proporem aperfeiçoamentos a este algoritmo e novas metodologias para o cálculo da probabilidade de falha.

Santos e Matioli (2010), propuseram a utilização do Método Duas Fases, desenvolvido por Luenberger (1974) que combina os métodos de Penalização e Gradiente Projetado, na determinação do ponto de projeto. A primeira fase consiste em determinar um ponto que reduza a medida de inviabilidade, enquanto que a segunda fase, melhora a otimalidade.

Apresentamos neste trabalho uma metodologia alternativa para obtenção do ponto de projeto. Trata-se de um Algoritmo de Restauração Inexata que utiliza filtro como critério de avaliação do passo, proposto por Karas, Oening e Ribeiro (2008). O método de Restauração Inexata (RI), proposto inicialmente por Martínez e Pilotta (2000), possui caraterísticas semelhantes ao Método Duas Fases, embora sejam executados de forma diferente, pois calcula o passo em duas fases independentes. A primeira tem por objetivo reduzir a inviabilidade. Já na segunda fase, o valor da função objetivo é reduzido na aproximação tangencial do conjunto viável. Dessa forma, analisamos o desempenho deste método e do proposto por Santos e Matioli (2010) aplicados ao problema de Confiabilidade Estrutural e, por fim, os comparamos ao método HLRF.

2. Problema de Confiabilidade Estrutural

Em Confiabilidade Estrutural as grandezas físicas presentes em um projeto, denominadas de variáveis básicas ou variáveis de projeto, são consideradas variáveis aleatórias que podem ser tratadas por meio de um vetor aleatório,

$$\underline{X} = (X_1, X_2, \ldots, X_n) \tag{1}$$

e a probabilidade de falha de uma estrutura é obtida a partir da avaliação das incertezas inerentes às variáveis de projeto por meio das distribuições de probabilidade dessas variáveis aleatórias. Primeiramente, para a avaliação da probabilidade de falha é necessário estabelecer

relações funcionais entre as variáveis básicas do sistema estrutural sob consideração. Matematicamente, essa relação ou função de desempenho pode ser descrita como:

$$h(\underline{X}) = h(X_1, X_2, \ldots, X_n). \tag{2}$$

A superfície de falha ou equação de estado limite é definida como $h(\underline{X}) = 0$. Essa superfície define o limite entre a região de segurança, $h(\underline{X}) > 0$, e a região de falha, $h(\underline{X}) < 0$. Dessa forma, a probabilidade de falha P_f é calculada por

$$P_f = \int \ldots \int_{h(\underline{X})<0} f_{\underline{X}}(X_1, X_2, \ldots, X_n)\, dX_1 dX_2 \ldots dX_n \tag{3}$$

onde $f_{\underline{X}}(X_1, X_2, \ldots, X_n)$ é a função densidade de probabilidade conjunta das variáveis de projeto.

A obtenção da probabilidade de falha por meio da resolução de (3) é uma tarefa complexa e, por este motivo, (3) é geralmente resolvido por meio de métodos analíticos conhecidos como FORM e SORM, que consistem em estimar o índice de confiabilidade, denotado por β, no espaço normal padrão. De acordo com Haldar e Mahadevam (2000), esse índice é definido como a mínima distância da origem à superfície de falha no espaço normal padrão.

Em geral, o cálculo de β pode ser formulado como o seguinte problema de otimização restrito

$$\begin{array}{ll} \min & \beta = \sqrt{x^T x} \\ \text{sujeito a} & h(x) = 0 \end{array} \tag{4}$$

onde x denota a variável padronizada. A probabilidade de falha pode ser obtida a partir da relação:

$$P_f = 1 - \Phi(\beta) \tag{5}$$

onde,

$$\Phi(\beta) = \int_{-\infty}^{\beta} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} dx. \tag{6}$$

O algoritmo HLRF foi desenvolvido especificamente para resolver o problema (4) e consiste em substituir, a cada iteração, a função estado limite por sua linearização no ponto corrente. Este método é baseado na seguinte fórmula recursiva

$$x_{k+1} = \frac{1}{\|\nabla h(x_k)\|^2} \left[\nabla h(x_k)^T x_k - h(x_k)\right] \nabla h(x_k). \tag{7}$$

Santosh *et al.* (2006) propuseram uma melhoria ao algoritmo HLRF, incluindo uma regra de seleção do comprimento do passo, conhecida como Regra de Armijo. Além deste método, vários algoritmos de otimização disponíveis na literatura podem ser aplicados na resolução do problema (4). Liu e Der Kiureghian (1991) avaliaram a aplicação dos métodos Gradiente Projetado, Lagrangeano Aumentado e Programação Quadrática no cálculo do ponto de projeto e compararam os resultados aos obtidos pelo HLRF. Na próxima seção, apresentaremos um algoritmo de Restauração Inexata, que até então não foi estudado no contexto da Confiabilidade Estrutural, e o método proposto por Santos e Matioli (2010) para ser usado neste contexto.

3. Métodos de Programação Não Linear

Nesta seção vamos discutir dois métodos utilizados para resolver o problema de programação não linear

$$\begin{array}{ll} \text{minimizar} & f(x) \\ \text{sujeito a} & g(x) \leq 0 \\ & h(x) = 0, \end{array} \tag{8}$$

onde assumimos que as funções $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^r$ e $h : \mathbb{R}^n \to \mathbb{R}^m$ são continuamente diferenciáveis. As matrizes jacobianas de g e h são denotadas por $A_g(\cdot)$ e $A_h(\cdot)$, respectivamente.

Seja $c \in \mathbb{R}^{m+r}$ o vetor

$$c(x) = \begin{bmatrix} g^+(x) \\ h(x) \end{bmatrix}, \tag{9}$$

onde $g^+(x) = \max\{0, g(x)\}$. Dizemos que a i-ésima restrição, $i = 1, \ldots, m + r$, é satisfeita em $x \in \mathbb{R}^n$ se $c_i(x) = 0$. Assim, se $c(x) = 0$, então x é um ponto viável, ou seja, satisfaz as restrições de (8).

Os algoritmos de programação não linear que podem ser aplicados na resolução do problema (8) lidam com dois objetivos conflitantes: minimizar f e obter viabilidade. Estes dois objetivos podem ser combinados por meio de funções penalidades ou Lagrangeano aumentado ou, ainda, podem ser tratados de forma independente. A seguir apresentaremos dois métodos empregados na resolução de (8): Restauração Inexata e Duas Fases. O primeiro, pertence à classe de métodos em que otimalidade e viabilidade são tratadas como dois objetivos independentes. Já o segundo constrói uma função penalidade e, para minimizá-la, aplica um procedimento de duas fases de modo que, a primeira fase consiste essencialmente de um passo elaborado para melhorar a viabilidade, e a segunda fase melhora a otimalidade.

3.1 Método de Restauração Inexata com Filtro

Os métodos de Restauração Inexata descritos por Martínez e Pilotta (2000) e Martínez (2001) geram uma sequência x^k considerando viabilidade e otimalidade em diferentes fases de cada iteração. A fase de viabilidade tem por objetivo determinar um ponto intermediário, z^k, com o valor da medida de inviabilidade menor que a de x^k e na fase de otimalidade é calculado um ponto que diminui a função objetivo, em relação a z^k. Esse passo é dado em uma aproximação tangencial do conjunto viável, a fim de não perder demais o ganho em viabilidade obtido na fase anterior. Além disso, uma estratégia de região de confiança é aplicada em torno de z^k para controlar o tamanho do passo.

O ponto obtido na fase de otimalidade é um ponto tentativo que será aceito como próximo iterando se passar por algum critério de aceitação do passo. Um critério clássico de aceitação do passo é fazer uso de uma função de mérito, sendo este o critério usado pelos autores citados acima. Uma outra opção é usar o critério de filtro, proposto por Fletcher e Leyffer (2002).

O método de Restauração Inexata apresentado neste trabalho foi proposto por Karas, Oening e Ribeiro (2008), o qual utiliza o critério de filtro para avaliação do passo tentativo. A seguir discutiremos alguns aspectos teóricos deste método, descrevendo cada uma das fases citadas acima, e apresentaremos o algoritmo, cuja aplicabilidade ao problema de confiabilidade estrutural será investigada.

3.1.1 Fase de viabilidade

Vamos considerar aqui uma medida de inviabilidade dada pela função $\theta : \mathbb{R}^n \to \mathbb{R}$ definida por

$$\theta(x) = \|c(x)\|, \tag{10}$$

onde $\|\cdot\|$ representa uma norma arbitrária. O ponto obtido na fase de viabilidade, z^k, deve satisfazer a seguinte condição

$$\theta(z^k) < (1-\alpha)\theta(x^k), \tag{11}$$

onde $\alpha \in (0,1)$. Esta condição estabelece a necessidade de z^k ser melhor que x^k no que se refere a viabilidade.

A princípio, qualquer algoritmo que minimize θ pode ser usado nesta fase. Neste trabalho, optamos por utilizar um algoritmo de filtro multidimensional prosposto por Gould, Leyffer e Toint (2004), que combina técnicas de filtro e região de confiança, a fim de resolver sistemas de equações e/ou inequações não lineares. Este algoritmo, bem como outros que podem ser aplicados nesta fase, podem falhar quando θ possui um ponto estacionário inviável. Neste caso, o método de Restauração Inexata pára sem sucesso.

3.1.2 Fase de otimalidade

Como citado anteriormente, o passo obtido nesta fase é dado em uma aproximação tangencial do conjunto viável, representada por

$$\mathcal{L}(z^k) = \left\{ z^k + d \in \mathbb{R}^n \mid A_h(z^k)d = 0,\ g(z^k) + A_g(z^k)d \leq g^+(z^k) \right\}. \tag{12}$$

Dessa forma, o algoritmo usado na fase de otimalidade deve encontrar x^{k+1} em $\mathcal{L}(z^k)$ que satisfaça $f(x^{k+1}) \leq f(x^k)$ e algum critério de aceitação de passo. O algoritmo que Karas, Oening e Ribeiro (2008) empregam nessa fase foi proposto por Gonzaga, Karas e Vanti (2003) e consiste em resolver aproximadamente o problema

$$\begin{array}{ll} \text{minimizar} & m_k(z^k + s) = f(z^k) + \nabla f(z^k)^T s + \frac{1}{2} s^T B_k s \\ \text{sujeito a} & z^k + s \in \mathcal{L}(z^k) \\ & \|s\| \leq \Delta, \end{array} \tag{13}$$

onde B_k é uma matriz simétrica que pode ser uma aproximação de $\nabla^2 f(z^k)$, e Δ é o raio da região de confiança.

3.1.3 Critério de aceitação do passo

O algoritmo proposto por Karas, Oening e Ribeiro (2008) usa o critério de filtro para avaliar o passo tentativo. Neste algoritmo, uma região proibida em $\mathbb{R}^2$ é definida, armazenando pares $(f(x^j), \theta(x^j))$, escolhidos convenientemente das iterações anteriores. Um ponto tentativo é aceito quanto não for dominado por nenhum elemento do filtro, segundo a regra

$$x \text{ é dominado por } y \text{ se, e somente se } f(x) + \alpha\theta(x) \geq f(y) \text{ e } \theta(x) \geq (1-\alpha)\theta(y),$$

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

No início de cada iteração, o par $(f(x^k), \theta(x^k))$ é temporariamente introduzido no filtro e, ao término da iteração, este elemento se tornará permanente somente se a iteração não produzir um decréscimo em f. A figura 1 ilustra o conceito de filtro, segundo a regra de dominação definida anteriormente, onde a região pintada representa a região proibida. A

região mais escura representa o filtro permanente, enquanto que a região proibida pelo ponto corrente é a mais clara. Na figura, para simplificar a notação, usamos x para representar o par $(f(x), \theta(x))$. Note que o ponto x^+ é aceito pelo filtro, pois não pertence à região proibida.

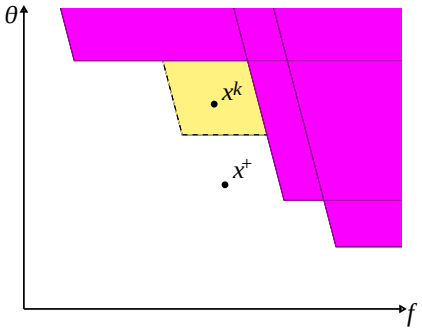

Figura 1: Filtro

3.1.4 Algoritmo de Restauração Inexata com filtro

As discussões apresentadas anteriormente acerca do Método de Restauração Inexata estão resumidas no seguinte algoritmo.

Algoritmo 3.1
Dados: $x^0 \in \mathbb{R}^n$, $F_0 = \emptyset$, $\mathcal{F}_0 = \emptyset$, $\Delta_0 > 0$, $\Delta = \Delta_0$, $\alpha \in (0,1)$, $\eta \in (0,1)$.
$k = 0$
REPITA
Defina os conjuntos $\bar{F}_k = F_k \bigcup \{(f(x^k), \theta(x^k))\}$ e
$\bar{\mathcal{F}}_k = \mathcal{F}_k \bigcup \left\{x \in \mathbb{R}^n \mid f(x) + \alpha\theta(x) \geq f(x^k) \text{ e } \theta(x) \geq (1-\alpha)\theta(x^k)\right\}$.
Fase de viabilidade:
SE $\theta(x^k) = 0$, então $z^k = x^k$.
SENÃO, calcule $z^k \notin \bar{\mathcal{F}}_k$ tal que $\theta(z^k) < (1-\alpha)\theta(x^k)$.
SE impossível, pare sem sucesso.
Fase de otimalidade:
SE z^k é estacionário, pare com sucesso.
SENÃO,
Defina $\Delta = \max\{\Delta_0, 2\Delta\}$.
REPITA (enquanto x^{k+1} não é obtido)
Encontre s, solução aproximada de (13).
Defina $ared = f(z^k) - f(z^k + s)$ e $pred = m_k(z^k) - m_k(z^k + s)$.
SE $z^k + s \notin \bar{\mathcal{F}}_k$ E $ared \geq \eta\, pred$,
Faça $x^{k+1} = z^k + s$, $\Delta_k = \Delta$ e pare com sucesso.
SENÃO $\Delta = \frac{\Delta}{2}$.
Atualização do filtro
SE $f(x^{k+1}) < f(x^k)$,
$F_{k+1} = F_k$, $\mathcal{F}_{k+1} = \mathcal{F}_k$ (o novo elemento é descartado)
SENÃO,
$F_{k+1} = \bar{F}_k$, $\mathcal{F}_{k+1} = \bar{\mathcal{F}}_k$ (o novo elemento é incluído no filtro)
$k = k + 1$.

Para compreender o significado da expressão "ponto estacionário", empregada no Algoritmo 3.1, vamos definir a direção do gradiente projetado associada a cada $z \in \mathbb{R}^n$ como

$$d_c(z) = P_{\mathcal{L}(z)}(z - \nabla f(z)) - z \tag{14}$$

onde $P_\Omega(w)$ denota a projeção ortogonal de $w \in \mathbb{R}^n$ no conjunto fechado $\Omega \subset \mathbb{R}^n$. Em um ponto viável z as condições de KKT são equivalentes a $d_c(z) = 0$ e, portanto, este ponto é chamado estacionário.

Karas, Oening e Ribeiro (2008) provaram que todo ponto de acumulação da sequência gerado pelo Algoritmo 3.1 é estacionário, ou seja, o algoritmo é globalmente convergente.

Para implementação desse algoritmo, escolhemos os seguintes valores para as constantes: $\alpha = 10^{-4}$, $\Delta_0 = 1$ e $\eta = 0,25$.

3.2 Método Duas Fases

O Método Duas Fases, proposto por Luenberger (1974) para a resolução do problema (8), combina as características do Método do Gradiente Projetado com o baixo esforço computacional por passo do Método de Penalização e utiliza apenas os valores das funções e de suas derivadas primeiras.

O Método do Gradiente Projetado é a generalização do Método de Cauchy, de forma que o gradiente negativo é projetado em direção a fronteira da região viável e a busca pelo minimizador é feita ao longo da curva resultante, gerando uma sequência de pontos viáveis convergindo para a solução do problema restrito. Esse Método possui uma desvantagem, a necessidade satisfazer continuamente a viabilidade durante o processo de minimização.

O Método Duas Fases possui a mesma taxa de convergência assintótica do Método do Gradiente Projetado, porém com a vantagem de contornar as dificuldades associadas com a manutenção da viabilidade, exigindo deste modo, menos cálculos a cada iteração.

Utilizando a abordagem do Método de Penalização o problema (8), no Método Duas Fases, é substituído por um problema alternativo dado por

$$\min q(x) = f(x) + \mu \left\|h(x)\right\|^2 + \mu \left\|g^+(x)\right\|^2 \tag{15}$$

em que μ é uma constante positiva grande.

Cada passo do Método Duas Fases é um procedimento para resolver o problema penalizado (15), de modo que, a primeira fase consiste na aplicação do Método de Newton à função q, com o objetivo de tentar mover-se para perto da região viável obtendo um ponto z_k que melhora a viabilidade do problema. O passo dado nesta fase é análogo a correção do passo exigido no Método do Gradiente Projetado. Já na segunda fase, um passo é dado na direção oposta ao gradiente de q em z_k com o intuito de reduzir o valor da função q a partir do ponto obtido na fase de viabilidade. Vale observar que uma busca linear usual, como por exemplo busca de Armijo, é necessária para a realização desta fase.

A figura 2 ilustra a aplicação do método para o caso de uma única restrição de igualdade em que M é o subespaço tangente definido por:

$$M = \left\{ y \in \mathbb{R}^n : \nabla^T h\,(x)\, y = 0 \right\}. \tag{16}$$

Na aplicação do Método de Newton é necessário o conhecimento da hessiana de $q(x)$. Luenberger (1974) determinou, para grandes valores de μ, uma aproximação da direção de

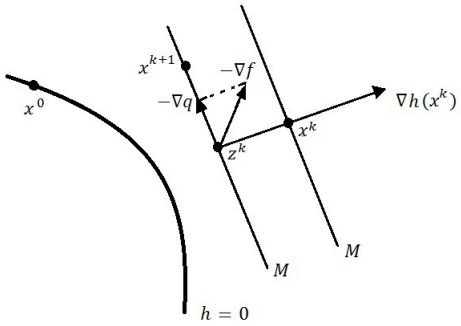

Figura 2: Método Duas Fases (Adaptado de Luenberger)

Newton, que utiliza somente informação de primeira ordem, dada por

$$d^k = -\frac{1}{2\mu}\left[\nabla c\left(x^k\right)\nabla^T c\left(x^k\right)\right]^{-2}\nabla c\left(x^k\right)\nabla q\left(x^k\right) \tag{17}$$

onde $\nabla c\left(x^k\right)$ é a transposta da matriz jacobiana de $c(x)$ com dimensão $(m+r)\times n$. Detalhes sobre a obtenção de d^k podem ser obtidos em Santos e Matioli (2010).

3.2.1 Algoritmo do Método Duas Fases

As ideias do Método Duas Fases, apresentadas anteriormente, são mostradas no seguinte algoritmo.

Algoritmo 3.2
Dados: $x^0 \in \mathbb{R}^n$, $\beta_0 = 1$, $\alpha_0 = 1$, $\mu^0 > 0$, $\epsilon \in (0,1)$, $\gamma \in (0,1)$.
$k = 0$
$\lambda^k = \left[\nabla c\left(x^k\right)\nabla^T c\left(x^k\right)\right]^{-1}\nabla c\left(x^k\right)\nabla f\left(x^k\right)$.
ENQUANTO o critério de parada não for satisfeito
Fase de viabilidade:
Calcular $d^k = -\frac{1}{2\mu}\left[\nabla c\left(x^k\right)\nabla^T c\left(x^k\right)\right]^{-2}\nabla c\left(x^k\right)\nabla q\left(x^k\right)$.
Determinar $\beta_k > 0$, tal que $q\left(x^k + \beta_k \nabla c\left(x^k\right)d^k\right) < q(x^k)$.
Fazer $z^k = x^k + \beta_k \nabla c\left(x^k\right)d^k$.
Fase de otimalidade:
Determinar $\alpha_k > 0$, por meio de busca linear, tal que $q\left(z^k - \alpha_k \nabla q\left(z^k\right)\right) < q(z^k)$.
Fazer $x^{k+1} = z^k - \alpha_k \nabla q\left(z^k\right)$.
Atualização do parâmetro de penalidade:
$\mu^{k+1} = 2\mu^k$.
Atualização do multiplicador de Lagrange:
$\lambda^{k+1} = \left[\nabla c\left(x^{k+1}\right)\nabla^T c\left(x^{k+1}\right)\right]^{-1}\nabla c\left(x^{k+1}\right)\nabla f\left(x^{k+1}\right)$.
Verificação do critério de parada:
$\frac{\left\|x^{k+1}-x^k\right\|}{\left\|x^{k+1}\right\|} \leq \varepsilon$ e $\frac{\left\|\lambda^{k+1}-\lambda^k\right\|}{\left\|\lambda^{k+1}\right\|} \leq \gamma$.
$k = k+1$.

Na implementação deste algoritmo, consideramos $\mu^0 = 100$, $\varepsilon = 10^{-6}$ (também usado como tolerância para o critério de parada dos demais métodos) e $\gamma = 10^{-6}$.

De acordo como Luenberger (1974), não é necessário encontrar β_k para garantir boas propriedades de convergência global do método, o que torna a implementação desse algoritmo relativamente simples. Já no caso do Método do Gradiente Projetado não há garantia de convergência global sem a introdução de métodos de perturbação, e, além disso, é necessário que o retorno ao conjunto viável, etapa que constitui a primeira fase do processo, seja executado dentro de uma tolerância aceitável. Neste sentido, o baixo esforço por passo no Método Duas Fases e a semelhança da taxa de convergência com o Método do Gradiente Projetado, indicam a vantagem desse método.

4. Testes numéricos

A fim de comparar os algoritmos apresentados na seção anterior com o HLRF, selecionamos 12 funções estado limite (EL) disponíveis na literatura. Em todos os casos, consideramos varáveis aleatórias normais e independentes. A seguir apresentamos as funções selecionadas, bem como os parâmetros das variáveis consideradas (média e variância).

EL 1: $h(\underline{X}) = X_1X_2 - 146,14$, onde $X_1 \sim N(78064,4; 11709,7^2)$ e $X_2 \sim N(0,0104; 0,00156^2)$.

EL 2: $h(\underline{X}) = 2 + 0,015\sum_{i=1}^{9} X_i^2 - X_{10}$, onde $X_{1,\ldots,10} \sim N(0,1)$.

EL 3: $h(\underline{X}) = 0,1(X_1 - X_2)^2 - \frac{(X_1+X_2)}{\sqrt{2}} + 2,5$, onde $X_{1,2} \sim N(0,1)$.

EL 4: $h(\underline{X}) = -0,5(X_1 - X_2)^2 - \frac{(X_1+X_2)}{\sqrt{2}} + 3$, onde $X_{1,2} \sim N(0,1)$.

EL 5: $h(\underline{X}) = 2 - X_2 - 0,1X_1^2 + 0,06X_1^3$, onde $X_{1,2} \sim N(0,1)$.

EL 6: $h(\underline{X}) = 2,5 - 0,2357(X_1 - X_2) + 0,0046(X_1 + X_2 - 20)^4$, onde $X_{1,2} \sim N(10,3^2)$.

EL 7: $h(\underline{X}) = 3 - X_2 + 256X_1^4$, onde $X_{1,2} \sim N(0,1)$

EL 8: $h(\underline{X}) = X_1^3 + X_2^3 - 18$, onde $X_1 \sim N(10,5^2)$ e $X_2 \sim N(9,9;5^2)$

EL 9: $h(\underline{X}) = X_1^3 + X_2^3 - 67,5$, onde $X_1 \sim N(10,5^2)$ e $X_2 \sim N(9,9;5^2)$.

EL 10: $h(\underline{X}) = 1 + \frac{(X_1+X_2)^2}{4} - 4(X_1 - X_2)^2$, onde $X_{1,2} \sim N(0,1)$.

EL 11: $h(\underline{X}) = 2,2257 - \frac{0,025\sqrt{2}}{27}(X_1 + X_2)^3 + 0,2357(X_1 + X_2)$, onde $X_{1,2} \sim N(10,3^2)$

EL 12: $h(\underline{X}) = X_1+2X_2+2X_3+X_4-5X_5-5X_6+0,001\sum_{i=1}^{6}\text{sen}(100X_i)$, onde $X_{1,\ldots,6} \sim N(0,1)$.

Os algoritmos 3.1, 3.2 e HLRF foram implementados em Matlab 7.8, versão R2009a, e os testes foram executados em um processador Intel(R) Core(TM)2 Duo CPU T5870 2,00GHz e 3,00GB de memória RAM. Para resolver o subproblema quadrático (13), referente à fase de otimalidade do Algoritmo 3.1, utilizamos o comando *Quadprog* do Matlab. Iniciamos este algoritmo com $B_0 = I$ e, para a atualização de B_k, implementamos a fórmula de atualização BFGS. Para determinar α_k na fase de otimalidade do Algoritmo 3.2, utilizamos a busca de Armijo. É importante observar que as variáveis originais foram padronizadas nos casos necessários e assim, os algoritmos foram executados considerando os pontos no espaço normal padrão.

A tabela 2 resume os resultados observados para cada um dos três algoritmos testados, onde β representa o valor do índice de confiabilidade, $\#h$ é o número de avaliações da função estado limite, $\#\nabla h$ representa o número de avaliações do gradiente de h e *ite* é o número de iterações. Os símbolos *, NC e – indicam que o algoritmo não resolveu o problema, não convergiu para um minimizador e atingiu o número máximo de iterações (1000), respectivamente.

Consideramos a média das variáveis padronizadas, ou seja, o vetor nulo, como ponto inicial para todos os problemas.

EL	Alg.	$\#h$	$\#\nabla h$	ite	β	EL	Alg.	$\#h$	$\#\nabla h$	ite	β
	3.1	621	161	47	5,3333		3.1	6	6	1	3,0000
1	3.2	159	10	4	5,4280	7	3.2	426	32	25	3,0000
	HLRF	6	6	5	5,4280		HLRF	3	3	2	3,0000
	3.1	5	5	1	2,0000		3.1	30	22	4	2,2259
2	3.2	432	32	15	2,0000	8	3.2	362	28	13	2,2260
	HLRF	3	3	2	2,0000		HLRF	NC	NC	NC	NC
	3.1	5	5	1	2,5000		3.1	29	21	4	1,9003
3	3.2	426	32	15	2,5000	9	3.2	304	24	11	1,9003
	HLRF	3	3	2	2,5000		HLRF	NC	NC	NC	NC
	3.1	6	6	1	3,0000		3.1	*	*	*	*
4	3.2	438	32	15	3,0000	10	3.2	*	*	*	*
	HLRF	3	3	2	3,0000		HLRF	*	*	*	*
	3.1	5	5	1	2,0000		3.1	38	38	1	2,2257
5	3.2	412	32	15	2,0000	11	3.2	560	38	18	3,9327
	HLRF	3	3	2	2,0000		HLRF	154	154	153	2,6557
	3.1	6	6	1	2,5000		3.1	367	247	51	2,6146
6	3.2	428	32	15	2,5000	12	3.2	–	–	–	–
	HLRF	3	3	2	2,5000		HLRF	–	–	–	–

Tabela 2: Resultados - Algoritmos 3.1, 3.2 e HLRF

Podemos observar que existem diferenças significativas entre o número de iterações realizadas em cada algoritmo. Porém, essa medida não pode ser considerada como um critério que define o melhor método. Isto porque os algoritmos possuem mecanismos diferentes para o cálculo do passo. Por exemplo, no Algoritmo 3.1 é necessário chamar dois algoritmos internos, que determinam o passo de viabilidade e otimalidade. O número de iterações para esse algoritmo mostrado na tabela 2 não leva em consideração os algoritmos internos. Podemos observar que o Algoritmo 3.1 resolveu aproximadamente 58% dos problemas em apenas uma iteração. Na verdade, nestes problemas, o passo de viabilidade obtido pelo algoritmo de filtro multidimensional, proposto por Gould, Leyffer e Toint (2005), é estacionário. Dessa forma, não foi necessário executar a fase de otimalidade. Para ter uma ideia do custo desse passo, apresentamos na tabela a seguir o número de iterações do algoritmo de viabilidade para os problemas resolvidos em uma iteração.

EL	EL 2	EL 3	EL 4	EL 5	EL 6	EL 7	EL 11
ite	2	2	3	2	3	3	35

Tabela 3: Número de iterações do algoritmo de viabilidade

Podemos notar, que mesmo no algoritmo de viabilidade o número de iterações foi baixo, exceto para a função estado limite 11.

Na tabela 2 observamos que o número de avaliações de funções realizadas pelo Algoritmo 3.2 é superior aos demais. Isto se justifica pelo fato de que, ao empregar o Método de Cauchy, é necessário realizar busca linear até que o comprimento α do passo dado na fase de

otimalidade seja obtido. A avaliação da função h é necessário na verificação da condição de Armijo em cada ponto tentativo.

Ao analisar o desempenho dos algoritmos na resolução do problema cuja restrição é dada por EL 1, observamos que os algoritmos 3.2 e HLRF alcançaram a solução em um pequeno número de iterações. Embora o Algoritmo 3.1 tenha demorado mais para atingir o critério de parada, este obteve um ponto cujo valor do índice de confiabilidade é menor do que aquele obtido pelos outros algoritmos, alcançando assim, um ponto sobre a função estado limite mais próximo da origem quando comparado com os demais. Vale lembrar os dois pontos obtidos, $(-5,0971; -1,5695)$ e $(-3,8383; -3,8381)$ são minimizadores locais do problema (4).

O algoritmo HLRF apresentou um comportamento oscilatório ao tentar resolver os problemas com EL 8 e EL 9. Tal comportamento permaneceu mesmo quando foram considerados outros pontos iniciais. Este é o grande inconveniente deste método. Vemos facilmente, que quando o algoritmo HLRF converge, a convergência é rápida e os cálculos são simples. Por outro lado, não há garantia de convergência global, uma vez que este é dependente do ponto inicial. Já os algoritmos 3.1 e 3.2 são globalmente convergentes, como apresentado em Karas, Oening e Ribeiro (2008) e Luenberger (1974), respectivamente.

Vamos justificar agora o fato dos três algoritmos falharem ao considerar EL 10. O Algoritmo 3.1 obteve falha na fase de restauração, aquela já prevista na teoria como possível de ocorrer, uma vez que foi verificado que o ponto inicial $x^0 = (0,0)$ é um ponto estacionário para a medida de inviabilidade θ. Já os algoritmos 3.2 e HLRF não foram capazes de resolver o problema partindo de $x^0 = (0,0)$, pois $\nabla h(x^0) = 0$ e, dessa forma, erros numéricos ocorreram ao calcular d^k, dado por (17), e x^{k+1} dado por (7). Dessa forma, executamos mais uma vez os algoritmos partido de $x^0 = (1,-1)$ e, neste caso, os algoritmos 3.1, 3.2 e HLRF alcançaram a solução $\beta = 0,3536$ em 1, 16 e 5 iterações, respectivamente.

A função estado limite dada por EL 12 é frequentemente considerada nos trabalhos que propõem novas metodologias para o cálculo da probabilidade de falha, por se tratar de uma situação desfavorável aos algoritmos, uma vez que contém um ruído artificial, como mencionado por Liu e Der Kiureghian (1991). Isto justifica a dificuldade encontrada pelos 3 algoritmos testados na resolução deste problema. O Algoritmo 3.2 converge para uma solução cuja distância até a origem é de 20,54, sendo assim, no contexto da confiabilidade estrutural, este resultado não é válido.

5. Conclusão

Analisamos neste trabalho o desempenho de dois métodos de Programação Não Linear, Restauração Inexata e Método Duas Fases, quando aplicados ao problema de confiabilidade estrutural e comparamos os resultados aos obtidos pelo HLRF. Os dois métodos possuem características semelhantes pois calculam o passo a cada iteração considerando duas fases, sendo que uma delas tem por objetivo reduzir a inviabilidade e a outra reduz a otimalidade. A diferença entre eles reside no fato de que o Método Duas Fases combina viabilidade e otimalidade em uma função penalidade. Já no Método de Restauração Inexata, viabilidade e otimalidade são consideradas separadamente. Neste método cada passo deve satisfazer o critério de filtro.

Os dois métodos de Programação Não Linear apresentados e o algoritmo HLRF foram implementados no ambiente Matlab e testes foram realizados considerando 12 problemas de confiabilidade estrutural, com variáveis aleatórias independentes e normalmente distribuídas. Os métodos mostraram-se eficientes na determinação do índice de confiabilidade β. Em dois problemas (EL 2 e EL 3) observamos que o algoritmo HLRF, amplamente utilizado na en-

genharia estrutural, não convergiu para um minimizador do problema (4) e em outro (EL 6), excedeu o número máximo de iterações. O algoritmo 3.1 de Restauração Inexata convergiu rapidamente na maioria dos casos, porém quando EL 4 foi considerada, apresentou falha na fase de viabilidade. O Método Duas Fases ultrapassou o número máximo de iterações ao resolver o problema (4) com $h(\underline{X})$ dada por EL 12. Em relação ao algoritmo HLRF, os outros dois algoritmos apresentam a vantagem de serem globalmente convergentes. Isto motiva a aplicação de tais métodos ao problema de determinar a probabilidade de falha de uma estrutura.

Referências

Fletcher, R. e Leyffer, S. (2002), Nonlinear programming without a penalty function, *Mathematical Programming*, 91(2), 239-269.

Gonzaga, C. C., Karas, E. W. e Vanti, M. (2003), A globally convergent filter method for nonlinear programming, *SIAM Journal on Optimization*, 14(3), 646-669.

Gould, N. I. M., Leyffer, S. e Toint, Ph., L. (2005), A multidimensional filter algorithm for nonlinear equations and nonlinear least-squares, *SIAM Journal on Optimization*, 15, 17-38.

Haldar, A. e Mahadevan, S., *Probability, reliability and statistical methods in engineering design*, John Wiley & Sons, New York, 2000.

Karas, E. W., Oening, A. P. e Ribeiro, A. A. (2008), Global convergence of slanting filter methods for nonlinear programming, *Applied Mathematics and Computation*, 200, 486-500.

Liu P.L. e Der Kiureghian A. (1991), Optimization algorithms for structural reliability, *Structural Safety*, 9, 161-177.

Luenberger, D. G. (1974), A combined penalty function and gradient projection method for nonlinear programming, *Journal of Optimization Theory and Applications*, 14, 477-495.

Martínez, J. M. e Pilotta, E. A. (2000), Inexact restoration algorithms for constrained optimization, *Journal of Optimization Theory and Applications*, 104, 135-163.

Martínez, J. M. (2001), Inexact-restoration method with Lagrangian tangent decrease and a new merit function for nonlinear programming, *Journal of Optimization Theory and Applications*, 111, 39-58.

Santos, S. R e Matioli L. C. (2010), Desenvolvimento de algoritmos matemáticos aplicados a confiabilidade estrutural, *International Journal of Pressure Vessels and Piping*, 83, 742-748.

Santosh, T., Saraf R., Ghosh A., e Kushwaka H. (2006), Optimum step length selection rule in modified HL–RF method for structural reliability, *International Journal of Pressure Vessels and Piping*, 83, 742-748.