UNIVERSIDADE FEDERAL DO PARANÁ

GISLAINE APARECIDA PERIÇARO

ALGORITMOS DE FILTRO GLOBALMENTE CONVERGENTES: TEORIA, IMPLEMENTAÇÃO E APLICAÇÃO

CURITIBA
2011

GISLAINE APARECIDA PERIÇARO

ALGORITMOS DE FILTRO GLOBALMENTE CONVERGENTES: TEORIA, IMPLEMENTAÇÃO E APLICAÇÃO

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. Ademir Alves Ribeiro

Co-orientadora:

Profa. Dra. Elizabeth Wegner Karas

CURITIBA

2011

TERMO DE APROVAÇÃO

GISLAINE APARECIDA PERIÇARO

ALGORITMOS DE FILTRO GLOBALMENTE CONVERGENTES: TEORIA, IMPLEMENTAÇÃO E APLICAÇÃO

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

_____

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

_____

Prof. Dr. Clóvis Caesar Gonzaga
Departamento de Matemática, UFSC

_____

Profa. Dra. Mael Sachine
Departamento de Matemática, UFPR

_____

Profa. Dra. Sandra Augusta Santos
Departamento de Matemática Aplicada, UNICAMP

Curitiba, 12 de dezembro de 2011.

Aos meus pais, João e Terezinha.

Agradecimentos

A Deus, pela vida, proteção e pelas graças recebidas.

Aos meus pais, irmãs, cunhados, sobrinhos e ao Júnior por todo amor, incentivo e por torcerem tanto pelo meu sucesso. Sem o apoio dessas pessoas tão especiais, nada disso seria possível.

Aos meus amigos Adriano, Juliano, Solange e Tatiane, pelo companheirismo, pelo apoio dado nos momentos difíceis e pelos momentos de descontração, principalmente aqueles que tornavam nossas cansativas viagens mais agradáveis.

À Solange, minha companheira de estudos desde as disciplinas do mestrado até a preparação para a primeira qualificação do doutorado. Com ela, além de momentos de alegria, dividi momentos de muita angústia e preocupações, mas sempre com a certeza de que iríamos superar tudo isso.

À D. Aparecida, por ter me recebido tão bem em sua casa durante os quatro anos de doutorado.

Ao meu orientador, professor Ademir, pelos ensinamentos, amizade, motivação, dedicação ao meu trabalho e, sobretudo, por ter me apresentado a este tema que a cada dia me encanta mais.

À minha co-orientadora, professora Elizabeth, pelo aprendizado proporcionado durante nossos seminários e pelas valiosas contribuições dadas ao trabalho.

Ao professor Anselmo Chaves Neto, pelo apoio e incentivo.

À Universidade Federal do Paraná, pela oportunidade de cursar o doutorado.

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

À Maristela Bandil, pela alegria e eficiência com as quais realiza seu trabalho.

À Universidade Estadual do Paraná, Campus Campo Mourão, por me proporcionar condições necessárias para concluir este curso.

À Fundação Araucária, pelo apoio financeiro.

Resumo

Discutimos neste trabalho métodos empregados para resolver problemas de programação não linear em que se deseja minimizar um função em uma determinada região do espaço multidimensional. Para solucionar tais problemas podemos empregar algoritmos iterativos que geram uma sequência de pontos, a qual esperamos convergir para um ponto estacionário. Uma forma de induzir a convergência é fazer uso do critério de filtro para verificar se um ponto tentativo deve ser aceito como próximo iterando. Para ser aceito pelo filtro, o ponto deve provocar uma redução na função objetivo ou na medida de inviabilidade considerada, quando comparado ao ponto corrente. O ponto pode ser testado por dois tipos de critérios de filtro, original ou inclinado, definidos de acordo com a regra que mede a redução no valor da função objetivo. Neste trabalho apresentamos um algoritmo geral de filtro, globalmente convergente, que não depende do método usado para o cálculo do passo e do critério de filtro considerado. A convergência é garantida desde que o passo satisfaça uma condição de eficiência que estabelece que perto de um ponto viável não estacionário a redução na função objetivo é relativamente grande. Mostramos que tal condição é satisfeita por pelo menos dois métodos empregados no cálculo do passo, um de Programação Quadrática Sequencial (PQS) e outro de Restauração Inexata (RI), para ambos os critérios de filtro. Para este primeiro método, apresentamos uma prova geral de que a condição de eficiência é satisfeita, sendo válida tanto para o critério de filtro original quanto inclinado. O algoritmo geral de filtro, bem como os algoritmos internos usados para determinar o passo foram implementados em MATLAB e testes numéricos foram realizados com problemas da coleção CUTEr. Para esses testes não foram observadas diferenças numéricas significativas entre os critérios de filtro, no entanto, o algoritmo de PQS mostrou-se mais robusto que RI e, ainda, mais eficiente em relação ao número de avaliações de funções e gradientes. Analisamos também a aplicabilidade dos algoritmos estudados a problemas práticos. Para isso, consideramos um problema de otimização que surge em análise de confiabilidade estrutural quando deseja-se determinar a probabilidade de falha de uma estrutura. Testes numéricos foram realizados com alguns problemas específicos da área de confiabilidade estrutural e os resultados indicaram que nosso algoritmo geral de filtro pode ser empregado nesse contexto.

Palavras-chave: Métodos de filtro, Convergência global, Implementação, Confiabilidade estrutural.

Abstract

We discuss in this work methods used to solve nonlinear programming problems in which one wishes to minimize a function into a particular region of the multidimensional space. To solve these problems we can use iterative algorithms that generate a sequence of points, which we hope to converge to a stationary point. A way to induce the convergence is to make use of the filter criterion to verify if a trial point should be accepted as the next iterate. To be accepted by the filter, the point should provide a decrease in the objective function or in the infeasibility measure considered, when compared to the current point. The point can be tested by two kinds of filter criteria, original or slanting, that are defined according to the rule that measures the reduction in the objective function value. In this work we present a general filter algorithm, globally convergent, which does not depend neither on the particular method used to calculate the step nor on the filter criterion adopted. The convergence is guaranteed under the assumption that the step satisfies an efficiency condition which establishes that near a feasible non-stationary point the decrease in the objective function is relatively large. We showed that such condition is satisfied for at least two methods used in the calculation of the step, one of them is based on Sequential Quadratic Programming (SQP) and the other is based on Inexact Restoration (IR), for both filter criteria. For the former method, we presented a general proof that the efficiency condition of the step is satisfied, being valid both for the original and for the slanting filter criterion. The general filter algorithm, as well as the internal algorithms used to determine the step were implemented in MATLAB and numerical experiments were performed with problems from the CUTEr collection. These tests have not presented significant numerical differences between the filter criteria, however, the SQP algorithm was more robust than IR and also more efficient when it comes to the number of functions and gradients evaluations. Furthermore, we also analyze the applicability of the studied algorithms to practical problems. For this porpose, we consider an optimization problem that arises in structural reliability analysis when it is desired to determine the failure probability of a structure. Numerical tests were performed with some particular problems of the structural reliability and the results indicated that our general filter algorithm can be used in this context.

Keywords: Filter methods, Global convergence, Implementation, Structural reliability.

Lista de Figuras

Lista de Tabelas

Sumário

Introdução

Estudamos neste trabalho métodos para solucionar problemas matemáticos da forma

$$(P) \qquad \begin{array}{ll} \text{minimizar} & f(x) \\ \text{sujeito a} & c_{\mathcal{E}}(x) = 0 \\ & c_{\mathcal{I}}(x) \leq 0, \end{array}$$

onde assumimos que as funções $f : \mathbb{R}^n \to \mathbb{R}$, $c_i : \mathbb{R}^n \to \mathbb{R}$, $i \in \mathcal{E} \cup \mathcal{I}$, são continuamente diferenciáveis. Problemas desse tipo surgem frequentemente em situações práticas de tomada de decisões e pertencem a uma classe de problemas denominada Programação Não Linear.

Os métodos de programação não linear que podem ser aplicados na resolução do problema (P) lidam com dois objetivos conflitantes: minimizar f e obter viabilidade. Estes dois objetivos podem ser combinados por meio de funções penalidades ou Lagrangiano Aumentado ou, ainda, podem ser tratados de forma independente, como nos Métodos de Filtro, introduzidos por Fletcher e Leyffer [12].

Baseados nos conceitos da otimização multiobjetivo, os métodos de filtro consideram otimalidade e viabilidade separadamente, priorizando este último objetivo, uma vez que a solução do problema deve ser um ponto viável. Dada uma medida de inviabilidade $h(\cdot)$, o filtro é definido como um conjunto de pares da forma $(f(x^j), h(x^j))$, em que nenhum par é dominado por outro, segundo a regra: o par $(f(x^j), h(x^j))$ é dito dominado pelo par $(f(x^l), h(x^l))$ se, e somente se, $f(x^j) \geq f(x^l)$ e $h(x^j) \geq h(x^l)$. De forma geral, os algoritmos de filtro consistem em, dado o ponto corrente, calcular um ponto tentativo, que será aceito como próximo iterando se o par (f, h) associado a ele não for dominado por nenhum elemento do filtro, ou seja, para um ponto tentativo ser aceito pelo filtro, este deve provocar uma redução suficiente em pelo menos uma das medidas de inviabilidade e otimalidade em relação aos demais pares do filtro.

Fletcher, Leyffer e Toint [13], apresentaram a primeira prova de convergência global de um método que combina filtro e região de confiança baseado em Programação Linear Sequencial (PLS). Em [14], estes mesmos autores estenderam a prova da convergência global no contexto da Programação Quadrática Sequencial (PQS), assumindo que a solução exata dos subproblemas quadráticos era encontrada. Exigindo apenas

uma solução aproximada dos subproblemas quadráticos, Fletcher *et al.* [10] estabeleceram convergência global de um algoritmo de filtro que decompõe o passo em duas componentes, normal e tangencial, baseados nos trabalhos de Byrd [3] e Omojokun [38]. Nesse método, a região de confiança é centrada no ponto corrente, limitando assim os passos de viabilidade (normal) e otimalidade (tangencial). Por algoritmo globalmente convergente, entende-se que a sequência gerada por ele tem pelo menos um ponto de acumulação estacionário.

Gonzaga, Karas e Vanti [16] propuseram um algoritmo geral de filtro globalmente convergente que mantém o cálculo do passo separado do algoritmo principal, e provaram que qualquer método para calcular o passo pode ser usado, desde que os pontos gerados sejam aceitos pelo filtro e que perto de um ponto viável não estacionário a redução da função objetivo seja relativamente grande. Mostraram ainda que o método de Restauração Inexata proposto por Martínez [34] e Martínez e Pilotta [35] satisfaz essa condição. Nesse método, os passos de viabilidade e otimalidade são tratados de forma mais independente do que em [10], uma vez que a região de confiança é centrada no passo obtido após a fase de viabilidade. Em [16] também foi provado que se uma modificação no critério de filtro for considerada, então todo ponto de acumulação da sequência gerada pelo algoritmo é estacionário.

Usando as mesmas ideias de [16], mas enfraquecendo a hipótese sobre o passo, Ribeiro, Karas e Gonzaga [43] provaram convergência global de métodos de filtro assumindo certas hipóteses que são válidas por pelo menos dois métodos para o cálculo do passo: PQS e Restauração Inexata.

Chin e Fletcher [7] e Fletcher, Leyffer e Toint [14] provaram que a sequência gerada por um algoritmo de filtro, cuja regra de dominação é um pouco diferente daquela proposta em [12], tem um ponto de acumulação estacionário, usando PLS e PQS, respectivamente, para calcular os iterandos. A regra de filtro considerada por esses autores é denominada filtro inclinado.

Empregando esse mesmo critério de filtro, Karas, Oening e Ribeiro [27] propuseram um algoritmo de filtro que usa Restauração Inexata para calcular o passo e provaram que todo ponto de acumulação da sequência gerada é estacionário, obtendo assim um resultado mais forte sobre convergência quando comparado aos demais trabalhos.

As técnicas de filtro também foram empregadas nos métodos dos Pontos Interiores, como apresentado nos trabalhos de Ulbrich, Ulbrich e Vicente [49] e Wächter e Biegler [51].

Os métodos de filtro podem ser usados também no contexto da otimização não diferenciável. Em [11], Fletcher e Leyffer aplicaram técnicas de filtro ao método dos feixes com região de confiança. Karas *et al.* [28] também combinaram filtro e método dos feixes, para solucionar problemas de otimização convexa não diferenciável e provaram convergência do algoritmo proposto para pontos estacionários.

Gould, Leyffer e Toint [17] propuseram um algoritmo que combina técnicas de filtro multidimensional e região de confiança para resolver sistemas de equações ou inequações não lineares. Em [20], os autores discutem os aspectos práticos do método proposto. Baseados em [17], Gould, Sainvitu e Toint [19] apresentaram um método de região de confiança com filtro para minimização irrestrita. Os autores provaram que o algoritmo é globalmente convergente e os resultados numéricos apresentados indicaram que o método proposto é competitivo com métodos clássicos de região de confiança.

Os métodos de filtro também têm sido empregados para resolver problemas de complementaridade, como apresentado em [30, 31].

Recentemente Shen, Xue e Chen [47] provaram convergência global de um algoritmo de filtro baseado em PQS com região de confiança que não requer qualquer procedimento de restauração, supondo uma condição de qualificação mais fraca que Mangasarian-Fromovitz.

Embora muitos trabalhos tenham sido elaborados considerando técnicas de filtro e região de confiança, alguns autores também propuseram a junção dos métodos de filtro com a estratégia de busca linear para globalização de métodos de programação não linear. Entre eles podemos citar Chin [5, 6], Wächter e Biegler [50, 51], Gu e Zho [24] e Wang *et al.* [52]. Em [48], Shen, Xue e Pu propuseram um método de filtro tridimensional, inspirado em [17], que calcula o passo por PQS e também emprega a estratégia de busca linear.

Neste trabalho apresentamos um algoritmo geral de filtro que não depende do método usado para calcular o passo. Assumimos que o passo satisfaz uma condição de eficiência, formalizada no próximo capítulo na Hipótese H3 e, deste modo, provamos que o algoritmo é globalmente convergente, independente do critério de filtro usado, original [12] ou inclinado [7]. Para completar nossa análise, apresentamos a prova de que o passo calculado por PQS satisfaz essa condição. No entanto, ao contrário de [43], que prova um resultado similar considerando o filtro original, não levamos em conta uma escolha particular do critério de filtro em nossa prova. Além disso, discutimos o emprego do método de Restauração Inexata, que também pode ser aplicado para determinar o passo. A fim de comparar o desempenho numérico dos algoritmos estudados, implementamos em MATLAB o algoritmo geral de filtro e os algoritmos de PQS e Restauração Inexata para o cálculo do passo. Após calibrar os parâmetros dos algoritmos, realizamos testes numéricos considerando problemas da coleção CUTEr [18]. Testamos ainda a aplicabilidade dos métodos de filtro a um problema de otimização que surge em Análise de Confiabilidade Estrutural.

O trabalho está organizado da seguinte forma. No Capítulo 1 apresentamos o algoritmo geral de filtro e a prova de que é globalmente convergente. Além disso, discutimos duas maneiras clássicas de calcular o passo e provamos que o algoritmo baseado em PQS satisfaz a Hipótese H3 independente da regra de filtro considerada. No Capítulo 2 descrevemos detalhes da implementação dos algoritmos abordados no

Capítulo 1. Os resultados dos testes realizados com problemas da coleção CUTEr são discutidos no Capítulo 3. Uma aplicação dos algoritmos estudados é apresentada no Capítulo 4.

Capítulo 1

Convergência global dos métodos de filtro

Discutimos neste capítulo a convergência global de métodos de filtro para solucionar o problema de programação não linear

$$\begin{array}{ll} \text{minimizar} & f(x) \\ \text{sujeito a} & c_{\mathcal{E}}(x) = 0 \\ & c_{\mathcal{I}}(x) \leq 0, \end{array} \tag{1.1}$$

onde os conjuntos de índices $\mathcal{E}$ e $\mathcal{I}$ se referem às restrições de igualdade e desigualdade, respectivamente. Considere m a cardinalidade de $\mathcal{E} \cup \mathcal{I}$ e assuma que as funções $f : \mathbb{R}^n \to \mathbb{R}$, $c_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, m$, são duas vezes continuamente diferenciáveis. As matrizes jacobianas de $c_{\mathcal{E}}$ e $c_{\mathcal{I}}$ são denotadas por $A_{\mathcal{E}}$ e $A_{\mathcal{I}}$, respectivamente.

Os métodos de programação não linear empregados na resolução de (1.1) têm por objetivo determinar um ponto $x^* \in \mathbb{R}^n$ que minimiza a função objetivo f no conjunto viável

$$\Omega = \{x \in \mathbb{R}^n \mid c_{\mathcal{E}}(x) = 0, c_{\mathcal{I}}(x) \leq 0\}.$$

Como os métodos apresentados nesse capítulo são iterativos e aceitam pontos inviáveis no decorrer das iterações, torna-se necessário definirmos uma função para medir o quanto um iterando está próximo do conjunto viável. Dessa forma, definimos como medida de inviabilidade, a função $h : \mathbb{R}^n \to \mathbb{R}_+$ dada por

$$h(x) = \left\| c^+(x) \right\|, \tag{1.2}$$

onde $\|\cdot\|$ é uma norma arbitrária e a função $c^+ : \mathbb{R}^n \to \mathbb{R}^m$ é definida por

$$c_i^+(x) = \begin{cases} c_i(x) & \text{se } \ i \in \mathcal{E} \\ \max\{0, c_i(x)\} & \text{se } \ i \in \mathcal{I}. \end{cases} \tag{1.3}$$

Os métodos de filtro, introduzidos por Fletcher e Leyffer em [12], definem uma *região proibida* em $\mathbb{R}^n$ associada aos pares $(f(x^j), h(x^j))$ escolhidos convenientemente das iterações anteriores, formando assim um conjunto de pares que denominamos *filtro*. Um ponto tentativo x^+ é aceito se o par $(f(x^+), h(x^+))$ não for dominado por nenhum elemento do filtro, segundo a regra: $(f(x^+), h(x^+))$ é dominado por $(f(x), h(x))$ se, e somente se, $f(x^+) \geq f(x)$ e $h(x^+) \geq h(x)$. Cada par do filtro possui a propriedade de não ser dominado por nenhum outro. No entanto, para garantir propriedades de convergência global dos métodos de filtro, esses mesmos autores sugerem que uma alteração seja feita nessa regra de dominação, criando-se uma margem em torno da região proibida, na qual os pontos também serão considerados proibidos.

Dessa forma, o método de filtro proposto em [12] evita pontos nas regiões

$$\mathcal{R}_j = \left\{ x \in \mathbb{R}^n \mid f(x) \geq f(x^j) - \alpha h(x^j) \;\text{ e }\; h(x) \geq (1-\alpha) h(x^j) \right\} \tag{1.4}$$

onde $\alpha \in (0,1)$ é uma constante dada. Temos também uma maneira um pouco diferente de definir a regra de dominação, proposta inicialmente por Chin [4], que considera as regiões

$$\mathcal{R}_j = \left\{ x \in \mathbb{R}^n \mid f(x) + \alpha h(x) \geq f(x^j) \;\text{ e }\; h(x) \geq (1-\alpha) h(x^j) \right\}. \tag{1.5}$$

O filtro baseado na regra (1.4) é denominado *filtro original* e aquele baseado em (1.5) é chamado *filtro inclinado*.

Na Figura 1.1 ilustramos as regiões em $\mathbb{R}^2$ formadas pelos pares $(f(x), h(x))$ associados aos pontos $x \in \mathcal{R}_j$, com $\mathcal{R}_j$ dado em (1.4) e (1.5), respectivamente. Tais pontos são recusados pelo filtro e, por esse motivo, denominamos cada uma dessas regiões de *região proibida no plano* $f \times h$. Nesta figura, e sempre que for conveniente, simplificamos a notação usando (f^j, h^j) para representar o par $(f(x^j), h(x^j))$.

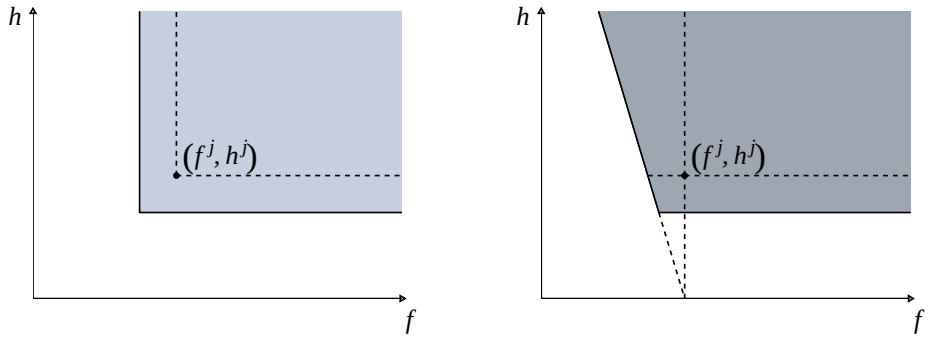

Figura 1.1: Regiões proibidas no plano $f \times h$.

Vamos apresentar na próxima seção nosso algoritmo geral de filtro.

1.1 O algoritmo geral de filtro

Apresentamos aqui um algoritmo geral de filtro que permite uma grande liberdade no cálculo do passo e na escolha do critério de filtro, original ou inclinado. Mostramos também, que este algoritmo é bem definido.

O algoritmo constrói uma sequência de conjuntos $F_0, F_1, \ldots, F_k$, compostos de pares $\left(f^j, h^j\right) \in \mathbb{R}^2$, onde F_k é denominado filtro corrente. Em nossa análise consideramos também o conjunto $\mathcal{F}_k$, que é uma região permanentemente proibida em $\mathbb{R}^n$ e uma região temporariamente proibida dada por $\bar{\mathcal{F}}_k = \mathcal{F}_k \bigcup \mathcal{R}_k$.

Na Figura 1.2 temos o filtro permanente, representado pelo conjunto

$$F_k = \left\{(f^i, h^i), (f^j, h^j), (f^l, h^l)\right\},$$

e o filtro temporário, dado por $\bar{F}_k = F_k \cup \left\{(f^k, h^k)\right\}$, para ambos os critérios, original e inclinado. As regiões hachuradas são formadas pelos pares $(f(x), h(x))$ correspondentes aos pontos $x \in \bar{\mathcal{F}}_k$.

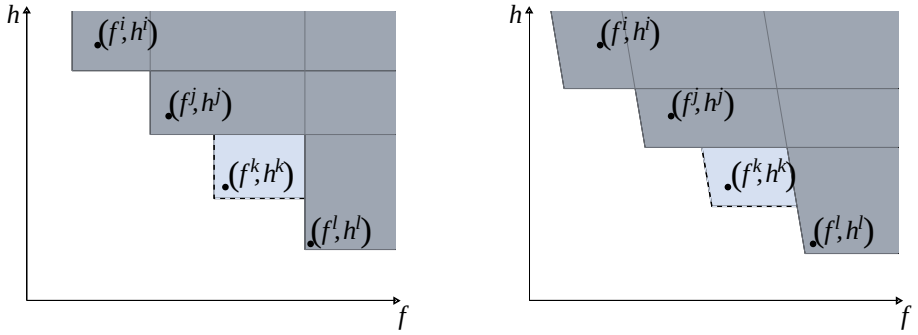

Figura 1.2: Filtro permanente.

Descrevemos agora nosso algoritmo geral de filtro.

Algoritmo 1.1 *Algoritmo geral de filtro*

Dados: $x^0 \in \mathbb{R}^n$, $F_0 = \emptyset$, $\mathcal{F}_0 = \emptyset$, $\alpha \in (0, 1)$.
$k = 0$
REPITA
Defina $\bar{F}_k = F_k \bigcup \{(f^k, h^k)\}$ e
$\bar{\mathcal{F}}_k = \mathcal{F}_k \bigcup \mathcal{R}_k$, com $\mathcal{R}_k$ dado em (1.4) ou (1.5)
Passo:
SE x^k é estacionário, pare com sucesso
SENÃO, calcule $x^{k+1} \notin \bar{\mathcal{F}}_k$.
Atualização do filtro:
SE $f(x^{k+1}) < f(x^k)$,
$F_{k+1} = F_k, \quad \mathcal{F}_{k+1} = \mathcal{F}_k \qquad$ (iteração f)

SENÃO,

$$F_{k+1} = \bar{F}_k \setminus \{(f^l, h^l) \in F_k \mid \mathcal{R}_l \subset \mathcal{R}_k\}, \quad \mathcal{F}_{k+1} = \bar{\mathcal{F}}_k \quad \text{(iteração } h\text{)}$$

$k = k + 1$.

No início de cada iteração, o par (f^k, h^k) é temporariamente introduzido no filtro, definindo a região proibida $\mathcal{R}_k$. Ao final da iteração, o par (f^k, h^k) se tornará permanente no filtro somente se a iteração não produzir uma redução em f, ou seja, se a iteração for do tipo h. Neste caso, removemos todos os pares $(f^l, h^l) \in F_k$ que definem uma região proibida contida na região associada ao par (f^k, h^k). Na iteração do tipo f o novo elemento é descartado, ou seja, não haverá atualização do filtro.

Note que se x^k é viável, então qualquer ponto x não proibido deve satisfazer $f(x) < f(x^k)$. A Figura 1.3 ilustra essa situação para ambos os critérios de filtro, original e inclinado.

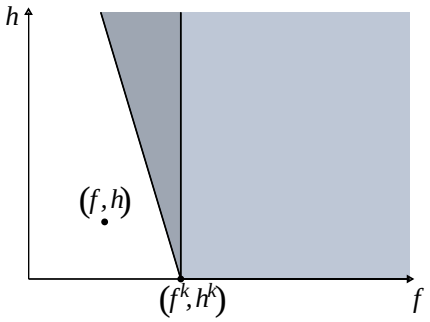

Figura 1.3: Caso em que x^k é viável.

O Lema 1.1, apresentado a seguir, estabelece que o Algoritmo 1.1 é bem definido, ou seja, se o ponto corrente é não estacionário, então um novo ponto não proibido pode ser escolhido.

Lema 1.1 *Considere o Algoritmo 1.1 e suponha que as restrições do problema* (1.1) *satisfazem uma condição de qualificação. Para todo* $k \in \mathbb{N}$ *tal que* x^k *é não estacionário, as seguintes afirmações são válidas:*

(*i*) $h^j > 0$, *para todo* $j \in \mathbb{N}$ *tal que* $(f^j, h^j) \in F_k$;

(*ii*) *Existe* $x^{k+1} \notin \bar{\mathcal{F}}_k$.

Demonstração. Vamos provar este lema por indução. Para $k = 0$, temos que $F_0 = \emptyset$ e $\bar{F}_0 = \{(f^0, h^0)\}$, logo (*i*) é válida. Para provar (*ii*), considere inicialmente que $h^0 > 0$. Nesse caso, podemos tomar x^1 como qualquer ponto viável. Por outro lado, se $h^0 = 0$, existe um ponto viável x^1 tal que $f(x^1) < f^0$, uma vez que x^0 não é um minimizador do problema (1.1). Em ambos os casos, concluímos que $x^1 \notin \bar{\mathcal{F}}_0$.

Agora, suponha que (*i*) e (*ii*) são válidas para $k - 1$. Se a iteração $k - 1$ é uma iteração f, então $F_k = F_{k-1}$ e consequentemente, pela hipótese de indução, temos

que a afirmação (i) é verdadeira para k. Caso contrário, $k-1$ é uma iteração h e $F_k = F_{k-1} \cup \{(f^{k-1}, h^{k-1})\}$. Nesse caso, é suficiente provar que $h^{k-1} > 0$. Suponha por contradição que $h^{k-1} = 0$. Pela hipótese de indução, existe $x^k \notin \bar{\mathcal{F}}_{k-1}$. Isto significa que $f(x^k) < f^{k-1}$, contradizendo o fato de que k é uma iteração h. Então, $h^{k-1} > 0$ e, deste modo, (i) é válida para k.

Resta provar (ii). Se $h^k > 0$, podemos tomar x^{k+1} como qualquer ponto viável. Por outro lado, se $h^k = 0$, como x^k não é um minimizador do problema (1.1), existe um ponto viável x^{k+1} tal que $f(x^{k+1}) < f^k$. Em ambos os casos, usando (i), concluímos que $x^{k+1} \notin \bar{\mathcal{F}}_k$. □

Dessa forma, vamos assumir que o Algoritmo 1.1 gera uma sequência infinita (x^k) e, na próxima seção, provamos que este algoritmo é globalmente convergente.

1.2 Convergência global

Assumindo uma hipótese sobre desempenho do passo, vamos provar nesta seção que qualquer sequência gerada pelo Algoritmo 1.1 tem pelo menos um ponto de acumulação estacionário. No decorrer dessa seção procuramos enfatizar as diferenças entre as propriedades de convergência que uma escolha particular da regra de filtro proporciona.

Primeiramente, vamos estabelecer as hipóteses necessárias para a análise de convergência do Algoritmo 1.1.

H1 *A sequência (x^k) permanece em um conjunto convexo e compacto $X \subset \mathbb{R}^n$.*

H2 *As funções f, c_i, $i = 1, \ldots, m$, são duas vezes continuamente diferenciáveis.*

H3 *Dado um ponto viável não estacionário $\bar{x} \in X$, existem $M > 0$ e uma vizinhança V de $\bar{x}$ tais que se $x^k \in V$, então*

$$f(x^k) - f(x^{k+1}) \geq M v_k,$$

onde $v_k = \min\left\{1, \min\left\{(1-\alpha)h^j \mid \left(f^j, h^j\right) \in F_k\right\}\right\}$ é definido como a altura do filtro.

As duas primeiras hipóteses são clássicas e, embora H1 seja uma hipótese sobre a sequência gerada pelo algoritmo, esta pode ser garantida incluindo restrições de caixa ao problema. Por outro lado, a Hipótese H3, proposta por Ribeiro, Karas e Gonzaga [43], assume que o passo deve ser eficiente no sentido de que, perto de um ponto viável não estacionário, a redução na função objetivo é relativamente grande. Esta condição se baseia no critério de Polak [40] para convergência global de algoritmos.

Considere o conjunto das iterações h dado por

$$\mathcal{K}_a = \left\{k \in \mathbb{N} \mid \left(f^k,\ h^k\right) \text{ é adicionado ao filtro}\right\}. \tag{1.6}$$

No lema a seguir vamos mostrar o que acontece quando este conjunto é infinito.

Lema 1.2 *Se o conjunto $\mathcal{K}_a$ é infinito, então*

$$h(x^k) \xrightarrow{\mathcal{K}_a} 0.$$

Demonstração. Assuma por contradição que, para algum $\delta > 0$, o conjunto

$$\mathcal{K} = \left\{k \in \mathcal{K}_a \mid h(x^k) \geq \delta\right\}$$

é infinito. A continuidade de (f, h), assegurada por H2, e a suposição de compacidade H1 garantem que existe uma subsequência convergente $(f^k, h^k)_{k\in\mathcal{K}_1}$, $\mathcal{K}_1 \subset \mathcal{K}$. Portanto, como $\alpha \in (0, 1)$, podemos tomar índices $j, k \in \mathcal{K}_1$, com $j < k$ tais que

$$\left\|(f^k, h^k) - (f^j, h^j)\right\| < \frac{\alpha\delta}{2} \leq \frac{\alpha h(x^j)}{2}.$$

Este resultado implica em $x^k \in \bar{\mathcal{F}}_j = \mathcal{F}_{j+1}$, o que é uma contradição, uma vez que, devido ao critério de atualização do filtro e à definição de $\bar{\mathcal{F}}$, temos que

$$x^k \notin \bar{\mathcal{F}}_{k-1} \supset \mathcal{F}_k \supset \mathcal{F}_{j+1}.$$

□

Vamos provar agora que a sequência (x^k) tem um ponto de acumulação viável.

Lema 1.3 *Considere a sequência $(x^k)_{k\in\mathbb{N}}$ gerada pelo Algoritmo 1.1. Então, existe um conjunto infinito $\mathbb{N}' \subset \mathbb{N}$ tal que $h(x^k) \xrightarrow{\mathbb{N}'} 0$.*

Demonstração. Se $\mathcal{K}_a$ é infinito, este resultado segue diretamente do Lema 1.2 e, nesse caso, $\mathbb{N}' = \mathcal{K}_a$. Por outro lado, se $\mathcal{K}_a$ é finito, existe $k_0 \in \mathbb{N}$ tal que toda iteração $k \geq k_0$ é uma iteração f. Deste modo, $(f(x^k))_{k\geq k_0}$ é decrescente e, pelas Hipóteses H1 e H2,

$$f(x^k) - f(x^{k+1}) \to 0. \tag{1.7}$$

Considere agora o conjunto

$$\mathcal{K}_1 = \left\{k \in \mathbb{N} \mid \alpha h(x^j) < f(x^k) - f(x^{k+1})\right\}$$

onde $j = k$ se usamos o filtro original e $j = k + 1$ se o filtro inclinado é usado.

Se $\mathcal{K}_1$ é finito, existe $k_1 \in \mathbb{N}$ tal que $h(x^{k+1}) < (1-\alpha)h(x^k)$ para todo $k \geq k_1$, o que implica em $h(x^k) \to 0$. Caso contrário, usando (1.7) concluímos que $h(x^k) \xrightarrow{\mathbb{N}'} 0$, com $\mathbb{N}' = \mathcal{K}_1$ ou $\mathbb{N}' = \{k + 1 \mid k \in \mathcal{K}_1\}$, dependendo da regra de filtro, original ou inclinado, respectivamente. De qualquer modo, $(x^k)_{k\in\mathbb{N}}$ tem um ponto de acumulação viável.

□

No lema a seguir apresentamos um resultado de convergência para pontos viáveis mais forte do que aquele apresentado no lema anterior. Esse resultado, cuja prova é dada em [27], estabelece que se a regra de filtro inclinado é usada, então qualquer ponto de acumulação da sequência gerada pelo algoritmo é viável. Este resultado também é provado por Chin e Fletcher [7] e por Fletcher, Leyffer e Toint [14], assumindo que um número infinito de pares (f^j, h^j) são adicionados ao filtro. Já Karas, Oening e Ribeiro [27] também apresentam esse resultado sem fazer esta exigência.

Lema 1.4 *Considere a sequência $(x^k)_{k\in\mathbb{N}}$ gerada pelo Algoritmo 1.1, onde $\mathcal{R}_k$ é definido por* (1.5)*. Então $h(x^k) \to 0$ e, consequentemente, qualquer ponto de acumulação da sequência (x^k) é viável.*

Demonstração. [27, Teorema 2.3]. □

O próximo lema mostra que se $\bar{x}$ é um ponto não estacionário, em uma vizinhança de $\bar{x}$, toda iteração k é uma iteração do tipo f.

Lema 1.5 *Seja $\bar{x} \in X$ um ponto não estacionário. Então nenhuma subsequência de $(x^k)_{k\in\mathcal{K}_a}$ converge para $\bar{x}$.*

Demonstração. Se $\bar{x}$ é um ponto viável, então pela Hipótese H3 exitem $M > 0$ e uma vizinhança V de $\bar{x}$ tais que para todo $x^k \in V$,

$$f(x^k) - f(x^{k+1}) \geq Mv_k.$$

Como $v_k > 0$, temos que $f(x^{k+1}) < f(x^k)$. Assim, $k \notin \mathcal{K}_a$.

Agora, assuma que $\bar{x}$ é inviável e suponha por contradição que existe um conjunto infinito $\mathcal{K} \subset \mathcal{K}_a$ tal que $x^k \overset{\mathcal{K}}{\to} \bar{x}$. Como h é contínua, temos que $h(x^k) \overset{\mathcal{K}}{\to} h(\bar{x})$. Por outro lado, o Lema 1.2 garante que $h(x^k) \overset{\mathcal{K}}{\to} 0$. Assim, $h(\bar{x}) = 0$, o que contradiz a hipótese de que $\bar{x}$ é inviável, completando a prova. □

Apresentamos a seguir a prova de que o Algoritmo 1.1 é globalmente convergente.

Teorema 1.6 *A sequência $(x^k)_{k\in\mathbb{N}}$ tem um ponto de acumulação estacionário.*

Demonstração. Se $\mathcal{K}_a$ é infinito, então pela Hipótese H1 existem $\mathcal{K}_1 \subset \mathcal{K}_a$ e $\bar{x} \in X$ tais que $x^k \overset{\mathcal{K}_1}{\to} \bar{x}$. Portanto, pelo Lema 1.5, $\bar{x}$ é estacionário. Caso contrário, existe $k_0 \in \mathbb{N}$ tal que toda iteração $k \geq k_0$ é uma iteração do tipo f. Deste modo, $\big(f(x^k)\big)_{k\geq k_0}$ é decrescente e por H1 e H2,

$$f(x^k) - f(x^{k+1}) \to 0. \tag{1.8}$$

Além disso, por construção do Algoritmo 1.1, $F_k = F_{k_0}$ para todo $k \geq k_0$. Portanto, a sequência $(v_k)_{k\in\mathbb{N}}$, definida em H3, satisfaz

$$v_k = v_{k_0} > 0 \tag{1.9}$$

para todo $k \geq k_0$.

Seja $\bar{x}$ um ponto de acumulação viável de (x^k), cuja existência é garantida pelo Lema 1.3. Vamos provar que este ponto é estacionário. Seja $\mathcal{K}$ um conjunto de índices tal que $x^k \overset{\mathcal{K}}{\to} \bar{x}$. Assuma por contradição que $\bar{x}$ é não estacionário. Pela Hipótese H3, existem $M > 0$ e uma vizinhança V de $\bar{x}$ tais que se $x^k \in V$, então

$$f(x^k) - f(x^{k+1}) \geq M v_k.$$

Como $x^k \overset{\mathcal{K}}{\to} \bar{x}$, então existe $k_1 > k_0$ tal que para todo $k > k_1$, $k \in \mathcal{K}$, $x^k \in V$. Portanto, para todo $k > k_1$, $k \in \mathcal{K}$, temos $f(x^k) - f(x^{k+1}) \geq M v_k = M v_{k_0} > 0$, contradizendo (1.8). □

O Teorema 1.6 estabelece que o Algoritmo 1.1 gera uma sequência infinita (x^k) que tem um ponto de acumulação estacionário. No entanto, se a regra de filtro inclinado é usada e, ainda, se o conjunto $\mathcal{K}_a$ é finito, podemos mostrar que qualquer ponto de acumulação da sequência gerada pelo algoritmo é estacionário. Provamos este resultado no próximo teorema.

Teorema 1.7 *Se $\mathcal{K}_a$ é finito e $\mathcal{R}_k$ é definido por* (1.5)*, então qualquer ponto de acumulação de (x^k) é estacionário.*

Demonstração. Do Lema 1.4, temos que qualquer ponto de acumulação da sequência (x^k) é viável. Dessa forma, pelos mesmos argumentos usados na prova do Teorema 1.6 quando $\mathcal{K}_a$ é finito, podemos concluir que qualquer ponto de acumulação de (x^k) é estacionário. □

Na próxima seção vamos descrever duas maneiras clássicas de determinar o passo do algoritmo geral de filtro.

1.3 Algoritmos internos

Apresentamos na seção anterior nossa principal hipótese, dada por H3, e mostramos que se ela é satisfeita, então o algoritmo geral de filtro é globalmente convergente. A partir de agora, discutiremos algoritmos internos que podem ser usados para calcular o ponto x^{k+1} de modo a satisfazer H3. Tais algoritmos estão baseados nos métodos de PQS [10] e Restauração Inexata [34, 35].

1.3.1 Programação Quadrática Sequencial

O algoritmo que descrevemos nesta seção é o mesmo apresentado por Ribeiro, Karas e Gonzaga [43], que foi inspirado no algoritmo de PQS com filtro proposto por Fletcher *et al.* [10]. Este algoritmo calcula o passo completo em duas fases. Primeiramente, temos a fase de viabilidade, cujo objetivo é reduzir a medida de inviabilidade h,

satisfazendo uma aproximação linear das restrições. Em seguida, a fase de otimalidade calcula um ponto tentativo minimizando um modelo quadrático da função objetivo na linearização do conjunto viável. Em [43] foi provado que se o filtro original é usado, esta abordagem satisfaz a Hipótese H3. Nesta seção provamos que esta hipótese é satisfeita quando consideramos tanto o filtro original como o inclinado.

Dado um ponto corrente x^k, consideramos o modelo da função objetivo

$$m_k(x^k + d) = f(x^k) + \nabla f(x^k)^T d + \frac{1}{2} d^T B_k d, \tag{1.10}$$

com $B_k \in \mathbb{R}^{n \times n}$ simétrica e uma linearização do conjunto viável, dada por

$$\mathcal{L}_k = \mathcal{L}(x^k) = \left\{ x^k + d \in \mathbb{R}^n \mid c_{\mathcal{E}}(x^k) + A_{\mathcal{E}}(x^k) d = 0,\ c_{\mathcal{I}}(x^k) + A_{\mathcal{I}}(x^k) d \leq 0 \right\}. \tag{1.11}$$

Se o conjunto $\mathcal{L}_k$ for não vazio, calculamos o passo resolvendo o problema quadrático

$$\begin{array}{ll} \text{minimizar} & m_k(x^k + d) \\ \text{sujeito a} & x^k + d \in \mathcal{L}_k \\ & \|d\| \leq \Delta, \end{array} \tag{1.12}$$

onde $\Delta > 0$ é o raio da região de confiança.

Uma solução de (1.12) fornece um ponto tentativo $x^k + d_\Delta$ que será avaliado pelo filtro. Para ser aceito como novo iterando este ponto não pode ser proibido pelo filtro corrente. Na verdade, vamos considerar o passo d_Δ como a soma de duas componentes: um passo de viabilidade n^k e um passo de otimalidade t_Δ.

O passo de viabilidade n^k deve satisfazer as restrições de (1.12) e tem por finalidade reduzir a medida de inviabilidade h. Esse passo pode ser obtido, por exemplo, por

$$n^k = P_{\mathcal{L}_k}(x^k) - x^k, \tag{1.13}$$

onde $P_{\mathcal{L}_k}(x^k)$ é a projeção ortogonal de x^k no conjunto $\mathcal{L}_k$. No entanto, não usamos essa escolha particular, mas assumimos uma certa eficiência nessa fase, descrita adiante na Hipótese H5. Além disso, o passo n^k é apenas útil se não estiver muito próximo da fronteira da região de confiança, pois, caso contrário, o passo de otimalidade poderá não produzir uma redução suficiente no modelo m_k. Dessa forma, dizemos que o subproblema (1.12) é compatível quando

$$\left\| n^k \right\| \leq \xi \Delta, \tag{1.14}$$

onde $\xi \in (0, 1)$ é uma constante. Em nossa análise vamos considerar

$$z^k = x^k + n^k \tag{1.15}$$

como o ponto obtido na fase de viabilidade. Note que, usando (1.10) e (1.15), temos

$$m_k(z^k) = m_k(x^k + n^k) = f(x^k) + \nabla f(x^k)^T n^k + \frac{1}{2} {n^k}^T B_k n^k. \tag{1.16}$$

Se o subproblema (1.12) for compatível, esperamos que o passo de otimalidade t_Δ produza uma redução razoável no modelo, sendo este passo obtido como uma solução aproximada do problema quadrático

$$\begin{array}{ll} \text{minimizar} & \left(\nabla f(x^k) + B_k n^k\right)^T t + \frac{1}{2} t^T B_k t \\ \text{sujeito a} & A_{\mathcal{E}}(x^k) t = 0 \\ & c_{\mathcal{I}}(x^k) + A_{\mathcal{I}}(x^k)(n^k + t) \leq 0 \\ & \left\| n^k + t \right\| \leq \Delta, \end{array} \tag{1.17}$$

obtido a partir de (1.12), com $d = n^k + t$. Assim, dado o ponto corrente x^k e um raio da região de confiança $\Delta > 0$, se (1.12) é compatível, o ponto tentativo é

$$x^k + d_\Delta = z^k + t_\Delta,$$

onde $z^k = x^k + n^k$ é o ponto que vem da fase de viabilidade e t_Δ é o passo de otimalidade.

Por outro lado, se $\mathcal{L}_k = \emptyset$ ou se o subproblema (1.12) for incompatível, o algoritmo chama um procedimento de restauração, cujo objetivo é obter um ponto $x^{k+1} \notin \bar{\mathcal{F}}_k$ com $h(x^{k+1}) < h(x^k)$, onde a função h é a medida de inviabilidade definida em (1.2). Existem vários algoritmos que podem ser empregados para obter x^{k+1} nesse caso, como o algoritmo de filtro multidimensional proposto por Gould, Leyffer e Toint [17], que será descrito no próximo capítulo, e o algoritmo proposto por Francisco [15], baseado em região de confiança.

A discussão apresentada até aqui está resumida no seguinte algoritmo para o cálculo do passo.

Algoritmo 1.2 *Cálculo de $x^{k+1} \notin \bar{\mathcal{F}}_k$ por PQS*

Dados: $x^k \in \mathbb{R}^n$, $\bar{\mathcal{F}}_k$, $0 < \Delta_{\min} < \Delta_{\max}$, $\Delta \in [\Delta_{\min}, \Delta_{\max}]$ e $c_p, \xi, \eta, \gamma \in (0,1)$.
SE $\mathcal{L}_k = \emptyset$,
    use o procedimento de restauração para obter $x^{k+1} \notin \bar{\mathcal{F}}_k$,
    obtenha B_{k+1} simétrica.
SENÃO
    calcule um passo de viabilidade n^k tal que $x^k + n^k \in \mathcal{L}_k$.
    REPITA (enquanto o ponto x^{k+1} não for obtido)
        SE $\left\| n^k \right\| > \xi \Delta$,
            use o procedimento de restauração para obter $x^{k+1} \notin \bar{\mathcal{F}}_k$,
            obtenha B_{k+1} simétrica.
        SENÃO,

calcule o passo de otimalidade t_Δ e defina $d_\Delta = n^k + t_\Delta$.
determine $ared = f(x^k) - f(x^k + d_\Delta)$ e $pred = m_k(x^k) - m_k(x^k + d_\Delta)$,
SE $\left\{x^k + d_\Delta \in \bar{\mathcal{F}}_k\right\}$ OU $\left\{pred \geq c_p\big(h(x^k)\big)^2 \quad \text{E} \quad ared < \eta\, pred\right\}$
    $\Delta = \gamma\Delta$
SENÃO
    $x^{k+1} = x^k + d_\Delta$
    obtenha B_{k+1} simétrica.
$\Delta_k = \Delta$

Assim, vemos que para um passo tentativo d_Δ ser aceito, além de $x^k + d_\Delta$ passar pelo critério de filtro, deve ser garantido também um decréscimo suficiente na função objetivo quando a redução predita pelo modelo não for muito pequena quando comparada com a violação das restrições.

Agora vamos provar que a Hipótese H3 é satisfeita se o Algoritmo 1.1 for aplicado ao problema (1.1) com o passo obtido pelo Algoritmo 1.2. Para tanto, apresentamos a função usada como medida de estacionaridade, sendo esta a mesma medida usada em [43].

Considere o conjunto $X \subset \mathbb{R}^n$ dado na Hipótese H1, $x, z \in X$ e $\mathcal{L}(x)$ definido em (1.11), chamamos o vetor

$$d^c(x,z) = P_{\mathcal{L}(x)}\big(z - \nabla f(x)\big) - z \tag{1.18}$$

de direção do gradiente projetado e usamos a função $\varphi : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$, dada por

$$\varphi(x,z) = \begin{cases} -\nabla f(x)^T \dfrac{d^c(x,z)}{\|d^c(x,z)\|} & \text{se } d^c(x,z) \neq 0, \\ 0 & \text{caso contrário,} \end{cases} \tag{1.19}$$

como medida de estacionaridade. De acordo com [16] temos que, em um ponto viável $\bar{x}$, as condições de KKT são equivalentes a $d^c(\bar{x},\bar{x}) = 0$. Além disso, se $\bar{x}$ não é estacionário, então $\varphi(\bar{x},\bar{x}) > 0$.

Considere o passo de Cauchy generalizado dado por $t^c = \lambda_c d_1^c$, onde

$$d_1^c = \frac{d^c(x^k, z^k)}{\|d^c(x^k, z^k)\|}$$

e

$$\lambda_c = \begin{cases} \underset{\lambda \geq 0}{\operatorname{argmin}} \left\{ m_k(z^k + \lambda d_1^c) \mid \left\| z^k + \lambda d_1^c - x^k \right\| \leq \Delta \right\} & \text{se } d^c(x^k, z^k) \neq 0, \\ 0 & \text{caso contrário,} \end{cases}$$

e assuma que as seguintes hipóteses são satisfeitas.

H4 *Todo ponto viável $\bar{x}$ satisfaz a condição de qualificação das restrições de Mangasarian-Fromovitz (MFCQ), a saber, os gradientes $\nabla c_i(\bar{x})$, $i \in \mathcal{E}$, são linearmente independentes, e existe uma direção $d \in \mathbb{R}^n$ tal que $A_{\mathcal{E}}(\bar{x})d = 0$ e $A_{\bar{\mathcal{I}}}(\bar{x})d < 0$, onde $\bar{\mathcal{I}} = \{i \in \mathcal{I} \mid c_i(\bar{x}) = 0\}$.*

H5 *Existem constantes $\delta_h > 0$ e $c_n > 0$ tais que para todo $k \geq 0$ com $h(x^k) \leq \delta_h$, um passo n^k pode ser calculado, satisfazendo*

$$\left\|n^k\right\| \leq c_n h(x^k).$$

H6 *Se o subproblema* (1.12) *for compatível, então a redução do modelo no passo de otimalidade t_Δ satisfaz*

$$m_k(z^k) - m_k(z^k + t_\Delta) \geq m_k(z^k) - m_k(z^k + t^c),$$

onde t^c é o passo de Cauchy generalizado.

H7 *As matrizes B_k são uniformemente limitadas, isto é, existe uma constante $\beta > 0$ tal que $\|B_k\| \leq \beta$ para todo $k \geq 0$.*

A Hipótese H4 é comum neste contexto e requer alguma regularidade das restrições. Já a Hipótese H5 significa que o passo de viabilidade deve ser razoavelmente escalado em relação às restrições. Em particular, $n^k = 0$ sempre que x^k é viável. Tal hipótese é discutida em detalhes por Martínez [34], que apresenta um algoritmo de viabilidade que a satisfaz sob condições razoáveis, como regularidade das restrições e ausência de ponto estacionário $\bar{x}$ para h, com $h(\bar{x}) \neq 0$. A Hipótese H6 diz que o passo de otimalidade deve ser pelo menos tão bom quanto o passo de Cauchy generalizado. Consideramos também uma condição clássica sobre as Hessianas B_k, descrita na Hipótese H7.

No próximo lema provamos que perto de um ponto viável, o conjunto $\mathcal{L}(x)$ é não vazio.

Lema 1.8 *Suponha que as Hipóteses H1, H2 e H4 sejam satisfeitas. Dado um ponto viável $\bar{x}$, existe uma vizinhança V_1 de $\bar{x}$ tal que $\mathcal{L}(x) \neq \emptyset$, para todo $x \in V_1$.*

Demonstração. Pela Hipótese H4, $A_{\mathcal{E}}(\bar{x})$ tem linhas linearmente independentes. Consequentemente, $A_{\mathcal{E}}(\bar{x})A_{\mathcal{E}}(\bar{x})^T$ é não singular e, por continuidade, existe uma vizinhança V_0 de $\bar{x}$ tal que para todo $x \in V_0$, $A_{\mathcal{E}}(x)A_{\mathcal{E}}(x)^T$ é não singular e

$$A_{\mathcal{E}}^+(x) = A_{\mathcal{E}}(x)^T \left(A_{\mathcal{E}}(x)A_{\mathcal{E}}(x)^T\right)^{-1}$$

é limitada em V_0. Assim, usando H1 e H2, temos que existe uma constante $M > 0$ tal que, para todo $i \in \mathcal{I}$ e $x \in V_0$,

$$\|\nabla c_i(x)^T A_{\mathcal{E}}^+(x)\| \leq M. \tag{1.20}$$

Também, por H4, temos que existe $d_0 \in \mathbb{R}^n$ tal que

$$A_{\mathcal{E}}(\bar{x})d_0 = 0 \qquad \text{e} \qquad A_{\bar{\mathcal{I}}}(\bar{x})d_0 < 0. \tag{1.21}$$

Defina

$$t = \frac{1}{2}\min\left\{ \frac{-c_i(\bar{x})}{\nabla c_i(\bar{x})^T d_0} \mid i \in \mathcal{I} \setminus \bar{\mathcal{I}} \ \text{ e } \ \nabla c_i(\bar{x})^T d_0 > 0 \right\}$$

e considere $\tilde{d} = td_0$. Deste modo, para $i \in \mathcal{I} \setminus \bar{\mathcal{I}}$, temos

$$c_i(\bar{x}) + \nabla c_i(\bar{x})^T \tilde{d} < 0. \tag{1.22}$$

Para $i \in \mathcal{E} \cup \bar{\mathcal{I}}$ temos que $c_i(\bar{x}) = 0$ e, de (1.21) e (1.22), segue que

$$c_{\mathcal{E}}(\bar{x}) + A_{\mathcal{E}}(\bar{x})\tilde{d} = 0 \qquad \text{e} \qquad c_{\mathcal{I}}(\bar{x}) + A_{\mathcal{I}}(\bar{x})\tilde{d} < 0.$$

Por continuidade, existem $\delta > 0$ e uma vizinhança $V_1 \subset V_0$ de $\bar{x}$ tais que para todo $x \in V_1$ e $i \in \mathcal{I}$

$$\left\| c_{\mathcal{E}}(x) + A_{\mathcal{E}}(x)\tilde{d} \right\| < \frac{\delta}{M} \qquad \text{e} \qquad c_i(x) + \nabla c_i(x)\tilde{d} < -\delta. \tag{1.23}$$

Dado $x \in V_1$, considere $d = \bar{d} + s$ com

$$\bar{d} = -A_{\mathcal{E}}^+(x)c_{\mathcal{E}}(x) \qquad \text{e} \qquad s = \big(I - A_{\mathcal{E}}^+(x)A_{\mathcal{E}}(x)\big)\tilde{d}.$$

Note que $s \in \mathcal{N}(A_{\mathcal{E}}(x))$. Assim,

$$c_{\mathcal{E}}(x) + A_{\mathcal{E}}(x)d = 0. \tag{1.24}$$

Para $i \in \mathcal{I}$, usando (1.23)

$$\begin{aligned} c_i(x) + \nabla c_i(x)^T d &= c_i(x) + \nabla c_i(x)^T \tilde{d} - \nabla c_i(x)^T A_{\mathcal{E}}^+(x)\big(c_{\mathcal{E}}(x) + A_{\mathcal{E}}(x)\tilde{d}\big) \\ &< -\delta - \nabla c_i(x)^T A_{\mathcal{E}}^+(x)\big(c_{\mathcal{E}}(x) + A_{\mathcal{E}}(x)\tilde{d}\big). \end{aligned}$$

Da desigualdade de Cauchy-Schwarz, (1.20) e (1.23), temos que

$$\left| \nabla c_i(x)^T A_{\mathcal{E}}^+(x)\big(c_{\mathcal{E}}(x) + A_{\mathcal{E}}(x)\tilde{d}\big) \right| \leq M\|c_{\mathcal{E}}(x) + A_{\mathcal{E}}(x)\tilde{d}\| < \delta$$

e, consequentemente,

$$c_{\mathcal{I}}(x) + A_{\mathcal{I}}(x)d < 0.$$

Este resultado, juntamente com (1.24), mostra que $x+d \in \mathcal{L}(x)$, completando a prova. □

Alguns dos resultados apresentados a seguir não dependem da regra de filtro considerada e já foram provados em [43]. No próximo lema avaliamos a medida de inviabilidade antes e depois de ser dado o passo tentativo.

Lema 1.9 *Suponha que as Hipóteses H1 e H2 sejam satisfeitas. Então, existe uma constante $c_h > 0$ tal que para quaisquer $x^k \in X$ e $\Delta > 0$ para os quais o passo tentativo d_Δ foi obtido pelo Algoritmo 1.2, temos*

$$h(x^k) \leq c_h\Delta \qquad e \qquad h(x^k + d_\Delta) \leq c_h\Delta^2.$$

Demonstração. [43, Lema 3.2]. □

No próximo lema avaliamos o quanto o modelo e a função objetivo podem piorar quando o passo de viabilidade é dado.

Lema 1.10 *Suponha que as Hipóteses H1, H2, H4, H5 e H7 sejam satisfeitas. Considere a constante δ_h e a vizinhança V_1 dadas pela Hipótese H5 e pelo Lema 1.8, respectivamente. Dado um ponto viável $\bar{x} \in X$, existem $N > 0$ e uma vizinhança $V_2 \subset V_1$ de $\bar{x}$ tais que se $x^k \in V_2$, então $h(x^k) \leq \delta_h$. Além disso, o passo n^k pode ser calculado e, para $z^k = x^k + n^k$,*

(i) $\left|m_k(x^k) - m_k(z^k)\right| \leq Nh(x^k)$;

(ii) $\left|f(x^k) - f(z^k)\right| \leq Nh(x^k)$.

Demonstração. [43, Lema 3.3]. □

Os próximos dois lemas estabelecem que os decréscimos do modelo e da função objetivo são relativamente grandes perto de um ponto viável não estacionário. O primeiro lema considera apenas o passo de otimalidade, enquanto que o segundo lema considera o passo completo.

Lema 1.11 *Suponha que as Hipóteses H1, H2, H4-H7 sejam satisfeitas. Seja $\bar{x} \in X$ um ponto viável não estacionário e $\bar{\eta} \in (0,1)$. Considere a vizinhança V_2 e a constante $\Delta_{\min}$ dadas pelo Lema 1.10 e Algoritmo 1.2, respectivamente. Então, existem constantes $\Delta_\rho \in (0, \Delta_{\min}]$, $\tilde{c} > 0$ e uma vizinhança $V_3 \subset V_2$ de $\bar{x}$ tais que se $x^k \in V_3$, $z^k = x^k + n^k$ e um passo de otimalidade tentativo t_Δ é obtido pelo algoritmo, temos*

(i) $m_k(z^k) - m_k(z^k + t_\Delta) \geq \tilde{c}\Delta'$ *para todos* Δ, Δ' *tais que* $\Delta' \leq \min\{\Delta, \Delta_\rho\}$;

(ii) $f(z^k) - f(z^k + t_\Delta) \geq \bar{\eta}\left(m_k(z^k) - m_k(z^k + t_\Delta)\right)$ *para todo* $\Delta \in (0, \Delta_\rho]$.

Demonstração. [43, Lema 3.4]. □

Lema 1.12 *Suponha que as Hipóteses H1, H2, H4-H7 sejam satisfeitas. Seja* $\bar{x} \in X$ *um ponto viável não estacionário. Considere as constantes* γ *e* η *dadas no Algoritmo 1.2,* c_h *dada no Lema 1.9 e* Δ_ρ, $\tilde{c}$ *e a vizinhança* V_3 *dadas no Lema 1.11. Se* $x^k \in V_3$, d_Δ *é um passo tentativo obtido pelo Algoritmo 1.2 com* $\Delta \leq \Delta_\rho$ *e*

$$\Delta \geq \gamma^2 \Delta_\rho \quad ou \quad h(x^k) \leq \frac{c_h \Delta^2}{\gamma^2},$$

então

(i) $m_k(x^k) - m_k(x^k + d_\Delta) \geq \frac{1}{2}\tilde{c}\Delta$;

(ii) $f(x^k) - f(x^k + d_\Delta) \geq \eta\left(m_k(x^k) - m_k(x^k + d_\Delta)\right)$.

Demonstração. Se $\Delta \geq \gamma^2 \Delta_\rho$, as afirmações (i) e (ii) seguem diretamente de [43, Lema 3.5]. Agora, suponha que $h(x^k) \leq \frac{c_h \Delta^2}{\gamma^2}$. Seja $\bar{\eta} \in (\eta, 1)$ e $\tau = \frac{\bar{\eta} - \eta}{\bar{\eta} + \eta}$. Considere a constante N dada pelo Lema 1.10. Podemos assumir, sem perda de generalidade, que

$$\Delta_\rho \leq \min\left\{\frac{\gamma^2 \tilde{c}}{2N c_h}, \frac{\tau \gamma^2 \tilde{c} \bar{\eta}}{N c_h}\right\}. \tag{1.25}$$

Dessa forma, se $x^k \in V_3$ e $\Delta \leq \Delta_\rho$, podemos aplicar o Lema 1.10, usando (1.25), para concluir que

$$\left|m_k(x^k) - m_k(z^k)\right| \leq N h(x^k) \leq N \frac{c_h \Delta^2}{\gamma^2} \leq \frac{1}{2}\tilde{c}\Delta.$$

Assim, desse resultado e do Lema 1.11(i), com $\Delta' = \Delta$, segue que

$$m_k(x^k) - m_k(x^k + d_\Delta) = m_k(x^k) - m_k(z^k) + m_k(z^k) - m_k(z^k + t_\Delta) \geq \frac{1}{2}\tilde{c}\Delta,$$

o que prova (i).

Para provar (ii), podemos aplicar novamente os Lemas 1.10 e 1.11 juntamente com (1.25), obtendo

$$\left|f(x^k) - f(z^k)\right| \leq N h(x^k) \leq N \frac{c_h \Delta^2}{\gamma^2} \leq \tau \bar{\eta} \tilde{c} \Delta \leq \tau\left(f(z^k) - f(z^k + t_\Delta)\right)$$

e

$$m_k(x^k) - m_k(z^k) \leq N h(x^k) < \tau \tilde{c} \Delta \leq \tau\left(m_k(z^k) - m_k(z^k + t_\Delta)\right).$$

Deste modo, temos que

$$\begin{aligned} f(x^k) - f(x^k + d_\Delta) &= f(x^k) - f(z^k) + f(z^k) - f(z^k + t_\Delta) \\ &\geq (1 - \tau)\left(f(z^k) - f(z^k + t_\Delta)\right) \end{aligned} \tag{1.26}$$

e

$$\begin{aligned} m_k(x^k) - m(x^k + d_\Delta) &= m_k(x^k) - m_k(z^k) + m_k(z^k) - m_k(z^k + t_\Delta) \\ &< (1+\tau)\left(m_k(z^k) - m_k(z^k + t_\Delta)\right). \end{aligned} \tag{1.27}$$

Portanto, de (1.26), usando o Lema 1.11(ii) e (1.27), obtemos

$$\begin{aligned} f(x^k) - f(x^k + d_\Delta) &\geq (1-\tau)\bar{\eta}\left(m_k(z^k) - m_k(z^k + t_\Delta)\right) \\ &> \frac{(1-\tau)}{1+\tau}\bar{\eta}\left(m_k(x^k) - m_k(x^k + d_\Delta)\right) \\ &= \eta\left(m_k(x^k) - m_k(x^k + d_\Delta)\right), \end{aligned} \tag{1.28}$$

completando a prova. □

Para apresentarmos o próximo resultado, precisamos definir a seguinte quantidade

$$\mathcal{H}_k = \max\left\{h(x^k), v_k\right\}, \tag{1.29}$$

onde v_k é a altura do filtro definida em H3.

A Figura 1.4 ilustra as duas possibilidades para $\mathcal{H}_k$ considerando o filtro inclinado. A figura à esquerda ilustra o caso $\mathcal{H}_k = v_k$, e a figura à direita mostra o caso $\mathcal{H}_k = h(x^k)$. As mesmas possibilidades ocorrem para o filtro original.

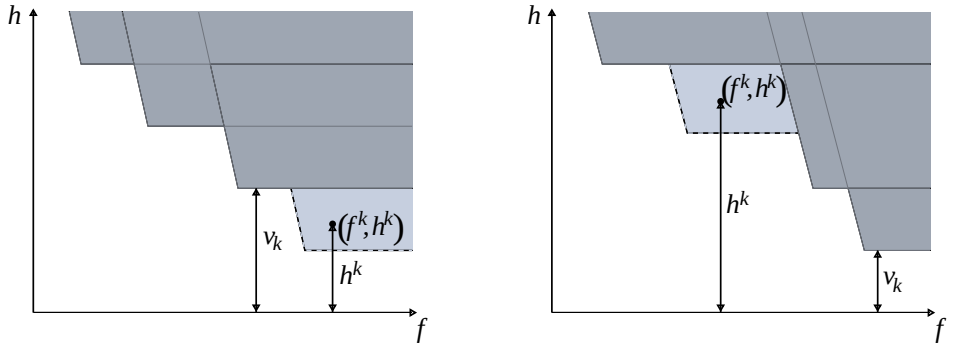

Figura 1.4: Quantidade $\mathcal{H}_k$.

O próximo lema estabelece que, perto de um ponto viável não estacionário, a rejeição de um passo se deve a um aumento da inviabilidade.

Lema 1.13 *Suponha que as Hipóteses H1, H2, H4-H7 sejam satisfeitas. Seja $\bar{x} \in X$ um ponto viável não estacionário. Considere a constante γ dada pelo Algoritmo 1.2, Δ_ρ e a vizinhança V_3 dada pelo Lema 1.11. Então, existe uma vizinhança $V \subset V_3$ de $\bar{x}$ tal que se $x^k \in V$ e d_Δ é o passo tentativo obtido pelo Algoritmo 1.1, temos*

$$h(x^k + d_\Delta) \geq \mathcal{H}_k$$

para qualquer $\Delta \in [\gamma^2\Delta_\rho, \Delta_\rho]$ que foi rejeitado pelo Algoritmo 1.2.

Demonstração. Sejam α, η, c_h e $\tilde{c}$ as constantes dadas pelos Algoritmos 1.1, 1.2 e Lemas 1.9, 1.11, respectivamente. Seja $V \subset V_3$ uma vizinhança de $\bar{x}$ tal que para todo $x \in V$, temos

$$\alpha h(x) \leq \frac{1}{2}\eta\tilde{c}\gamma^2\Delta_\rho. \tag{1.30}$$

Sem perda de generalidade podemos assumir que

$$\Delta_\rho \leq \frac{\eta\tilde{c}\gamma^2}{2\alpha c_h}. \tag{1.31}$$

Dessa forma, se $x^k \in V$ e $\Delta \in [\gamma^2\Delta_\rho, \Delta_\rho]$, podemos aplicar o Lema 1.12(i), obtendo

$$m_k(x^k) - m_k(x^k + d_\Delta) \geq \frac{1}{2}\tilde{c}\Delta \geq \frac{1}{2}\tilde{c}\gamma^2\Delta_\rho.$$

Usando (1.30), (1.31), o fato de que $\Delta_\rho \geq \Delta$ e o Lema 1.9, temos que

$$m_k(x^k) - m_k(x^k + d_\Delta) \geq \frac{\alpha}{\eta}\max\left\{h(x^k), h(x^k + d_\Delta)\right\}. \tag{1.32}$$

Por outro lado, do Lema 1.12(ii), segue que

$$f(x^k) - f(x^k + d_\Delta) \geq \eta\left(m_k(x^k) - m_k(x^k + d_\Delta)\right). \tag{1.33}$$

Temos ainda, de (1.32) e (1.33), que

$$f(x^k + d_\Delta) \leq f(x^k) - \alpha\max\left\{h(x^k), h(x^k + d_\Delta)\right\}. \tag{1.34}$$

Portanto, se o passo tentativo d_Δ foi rejeitado pelo Algoritmo 1.2, então de (1.33) temos que $x^k + d_\Delta \in \bar{\mathcal{F}}_k$. Deste modo, de (1.34), concluímos que

$$h(x^k + d_\Delta) \geq \max\left\{h(x^k), v_k\right\},$$

completando a prova. □

O próximo lema será útil para provarmos o principal resultado desta seção.

Lema 1.14 *Suponha que as Hipóteses H1 e H2 sejam satisfeitas. Se d_Δ é um passo tentativo obtido pelo Algoritmo 1.2 e $h(x^k + d_\Delta) \geq \mathcal{H}_k$, então*

$$h(x^k) \leq c_h\Delta^2 \quad e \quad \Delta \geq \frac{v_k}{\sqrt{c_h}}.$$

Demonstração. Do Lema 1.9, temos que

$$c_h\Delta^2 \geq h(x^k + d_\Delta) \geq \mathcal{H}_k.$$

Dessa forma, pela definição de $\mathcal{H}_k$ e pelo fato de que $v_k \leq 1$, concluímos que

$$c_h\Delta^2 \geq h(x^k) \quad \text{e} \quad c_h\Delta^2 \geq v_k \geq v_k^2,$$

completando a prova. □

Provaremos a seguir o principal resultado desta seção: se x^{k+1} for calculado pelo Algoritmo 1.2, então a Hipótese H3 é satisfeita. Como vimos no Teorema 1.6, essa hipótese foi fundamental na análise de convergência da Seção 1.1.

Para a análise que segue, vamos considerar o conjunto das iterações de restauração

$$\mathcal{K}_r = \left\{k \in \mathbb{N} \mid \mathcal{L}_k = \emptyset \text{ ou } \left\|n^k\right\| > \xi\Delta_k\right\}. \tag{1.35}$$

Teorema 1.15 *Suponha que o Algoritmo 1.1 seja aplicado ao problema* (1.1)*, com o novo iterando calculado pelo Algoritmo 1.2, e que as Hipóteses H1, H2, H4-H7 sejam satisfeitas. Dado um ponto viável não estacionário $\bar{x} \in X$, existem $M > 0$ e uma vizinhança V de $\bar{x}$ tais que se $x^k \in V$, então*

$$f(x^k) - f(x^{k+1}) \geq Mv_k.$$

Demonstração. Seja $\bar{x}$ um ponto viável não estacionário. Considere a vizinhança V dada pelo Lema 1.13 e a constante Δ_ρ dada pelo Lema 1.11. Sem perda de generalidade, podemos assumir que

$$\Delta_\rho \leq \min\left\{\frac{\gamma^2\xi}{c_hc_n}, \frac{\gamma^2\tilde{c}}{2c_hc_p}, \frac{\eta\tilde{c}\gamma^2}{2\alpha c_h}\right\}, \tag{1.36}$$

onde α é a constante dada no Algoritmo 1.1, c_p, ξ, η e γ são dadas no Algoritmo 1.2, c_n é dado na Hipótese H5, c_h e $\tilde{c}$ são dadas pelos Lemas 1.9 e 1.11, respectivamente. Pelo Lema 1.8 temos que se $x^k \in V$, então $\mathcal{L}_k \neq \emptyset$. Deste modo, o Algoritmo 1.2 inicia com o raio $\Delta \geq \Delta_{\min}$ e termina com $\Delta_k = \gamma^r\Delta$, onde r é o número de vezes que o raio foi reduzido no algoritmo. Vamos considerar dois casos, respectivamente com $\Delta_k \geq \gamma^2\Delta_\rho$ e $\Delta_k < \gamma^2\Delta_\rho$.

Primeiro caso: $\Delta_k \geq \gamma^2\Delta_\rho$. Nesse caso, usando a Hipótese H5 e restringindo a vizinhança V, se necessário, temos

$$\left\|n^k\right\| \leq c_nh(x^k) \leq \xi\gamma^2\Delta_\rho \leq \xi\Delta_k.$$

Então, o Algoritmo 1.2 não entra no procedimento de restauração durante a iteração k, isto é, $k \notin \mathcal{K}_r$. Portanto, aplicando o Lema 1.11(i) com $\Delta' = \gamma^2\Delta_\rho$, obtemos

$$m_k(z^k) - m_k(x^{k+1}) = m_k(z^k) - m_k(z^k + t_{\Delta_k}) \geq \tilde{c}\gamma^2\Delta_\rho. \tag{1.37}$$

Por outro lado, pelo Lema 1.10,

$$\left|m_k(x^k) - m_k(z^k)\right| \leq Nh(x^k). \tag{1.38}$$

Podemos restringir novamente a vizinhança V, se necessário, de modo que

$$Nh(x^k) \leq \frac{1}{2}\tilde{c}\gamma^2\Delta_\rho, \quad c_p\big(h(x^k)\big)^2 \leq \frac{1}{2}\tilde{c}\gamma^2\Delta_\rho \quad \text{e} \quad h(x^k) \leq 1. \tag{1.39}$$

De (1.37)-(1.39), temos que

$$m_k(x^k) - m_k(x^{k+1}) \geq \frac{1}{2}\tilde{c}\gamma^2\Delta_\rho \geq c_p\big(h(x^k)\big)^2.$$

Então, o mecanismo do Algoritmo 1.2 e o fato de que $v_k \leq 1$ implicam em

$$f(x^k) - f(x^{k+1}) \geq \eta\big(m_k(x^k) - m_k(x^{k+1})\big) \geq \frac{1}{2}\eta\tilde{c}\gamma^2\Delta_\rho \geq \frac{1}{2}\eta\tilde{c}\gamma^2\Delta_\rho v_k. \tag{1.40}$$

Segundo caso: agora, assuma que $\Delta_k < \gamma^2\Delta_\rho$. Neste caso, vamos analisar duas possibilidades relacionadas à quantidade $\mathcal{H}_k$, definida em (1.29). Na primeira, vamos supor que $h(x^k + d_\Delta) \geq \mathcal{H}_k$ para todo $\Delta \leq \gamma\Delta_\rho$ tal que o passo tentativo d_Δ foi calculado. Seja $\tilde{\Delta} = \dfrac{\Delta_k}{\gamma}$. Como $\Delta_k < \Delta_{\min}$, o passo tentativo $\tilde{d} = d_{\tilde{\Delta}}$ foi calculado. Além disso, $h(x^k + \tilde{d}) \geq \mathcal{H}_k$ porque $\tilde{\Delta} < \gamma\Delta_\rho$. Portanto, pelo Lema 1.14, temos

$$h(x^k) \leq c_h\tilde{\Delta}^2 = \frac{c_h\Delta_k^2}{\gamma^2} \quad \text{e} \quad \frac{\Delta_k}{\gamma} = \tilde{\Delta} \geq \frac{v_k}{\sqrt{c_h}}. \tag{1.41}$$

Pela Hipótese H5, (1.41) e (1.36), obtemos

$$\left\|n^k\right\| \leq c_n h(x^k) \leq \frac{c_n c_h \Delta_k^2}{\gamma^2} \leq \xi\Delta_k, \tag{1.42}$$

o que significa que o Algoritmo 1.2 não entra no procedimento de restauração durante a iteração k, isto é, $k \notin \mathcal{K}_r$. Portanto, pelo Lema 1.12(i) com $\Delta = \Delta_k$, (1.36), (1.41) e (1.39), temos

$$m_k(x^k) - m_k(x^{k+1}) \geq \frac{1}{2}\tilde{c}\Delta_k \geq \frac{c_p c_h}{\gamma^2}\Delta_k^2 \geq c_p h(x^k) \geq c_p\big(h(x^k)\big)^2. \tag{1.43}$$

Deste modo, pelo mecanismo do Algoritmo 1.2, de (1.43) e (1.41) obtemos

$$f(x^k) - f(x^{k+1}) \geq \eta\big(m_k(x^k) - m_k(x^{k+1})\big) \geq \frac{1}{2}\eta\tilde{c}\Delta_k \geq \frac{\eta\tilde{c}\gamma}{2\sqrt{c_h}}v_k. \tag{1.44}$$

Vamos analisar agora a segunda possibilidade, isto é, existe $\Delta \leq \gamma\Delta_\rho$ tal que $h(x^k + d_\Delta) < \mathcal{H}_k$. Seja $\hat{\Delta}$ o primeiro Δ satisfazendo tal condição. Vamos mostrar

que $\hat{\Delta} = \Delta_k$. Seja $\bar{d} = d_{\bar{\Delta}}$ o passo tentativo obtido com o raio $\bar{\Delta} = \dfrac{\hat{\Delta}}{\gamma}$. Afirmamos que

$$h(x^k + \bar{d}) \geq \mathcal{H}_k. \tag{1.45}$$

De fato, se $\bar{\Delta} \leq \gamma\Delta_\rho$, a definição de $\hat{\Delta}$ garante a afirmação. Por outro lado, se $\bar{\Delta} > \gamma\Delta_\rho$, então $\bar{\Delta} \in [\gamma^2\Delta_\rho, \Delta_\rho]$. Assim, aplicando o Lema 1.13 obtemos (1.45) e, pelo Lema 1.14, temos

$$h(x^k) \leq c_h\bar{\Delta}^2 = \frac{c_h\hat{\Delta}^2}{\gamma^2} \quad \text{e} \quad \frac{\hat{\Delta}}{\gamma} = \bar{\Delta} \geq \frac{v_k}{\sqrt{c_h}}. \tag{1.46}$$

Pelos mesmos argumentos utilizados para provar (1.42), obtemos $\left\|n^k\right\| \leq \xi\hat{\Delta}$. Logo, do Lema 1.12 com $\Delta = \hat{\Delta}$, obtemos

$$f(x^k) - f(x^k + d_{\hat{\Delta}}) \geq \eta\big(m_k(x^k) - m_k(x^k + d_{\hat{\Delta}})\big) \geq \frac{1}{2}\eta\tilde{c}\hat{\Delta} \tag{1.47}$$

que junto com (1.36), (1.46) e o Lema 1.9, implica em

$$f(x^k) - f(x^k + d_{\hat{\Delta}}) \geq \alpha\frac{c_h\hat{\Delta}^2}{\gamma^2} \geq \alpha\max\left\{h(x^k), h(x^k + d_{\hat{\Delta}})\right\}. \tag{1.48}$$

A definição de $\hat{\Delta}$ e (1.48) garantem que $x^k + d_{\hat{\Delta}}$ é aceito pelo filtro. Portanto, usando (1.47), concluímos que $x^k + d_{\hat{\Delta}} = x^{k+1}$. Além disso, de (1.47) e (1.46), segue que

$$f(x^k) - f(x^{k+1}) \geq \frac{1}{2}\eta\tilde{c}\hat{\Delta} \geq \frac{\eta\tilde{c}\gamma}{2\sqrt{c_h}}v_k. \tag{1.49}$$

Como (1.40), (1.44) e (1.49) esgotam todas as possibilidades, definindo

$$M = \min\left\{\frac{1}{2}\eta\tilde{c}\gamma^2\Delta_\rho, \frac{\eta\tilde{c}\gamma}{2\sqrt{c_h}}\right\},$$

completamos a demonstração. □

Assim, podemos concluir que o Algoritmo 1.1 com o passo calculado por PQS é globalmente convergente, independentemente do critério de filtro usado, original ou inclinado. A seguir, apresentamos outra forma de calcular o passo, proposta por Gonzaga, Karas e Vanti [16], que também satisfaz H3.

1.3.2 Restauração Inexata

O método de Restauração Inexata (RI) determina o passo em duas fases. A primeira é a fase de viabilidade, que tem por objetivo encontrar um ponto $z^k \notin \bar{\mathcal{F}}_k$ tal que $h(z^k) < (1-\alpha)h(x^k)$, onde x^k é o ponto corrente. Em seguida, se z^k não for estacionário, a fase de otimalidade busca reduzir o valor da função objetivo em relação a z^k. Nesta fase é aplicada uma estratégia de região de confiança para controlar o

tamanho do passo e, para não perder muito do ganho obtido na fase de viabilidade, o passo é determinado satisfazendo uma aproximação linear do conjunto viável, dada por

$$\mathcal{L}^{RI}(z^k) = \left\{ z^k + t \in \mathbb{R}^n \mid A_{\mathcal{E}}(z^k)t = 0, c_{\mathcal{I}}(z^k) + A_{\mathcal{I}}(z^k)t \leq c_{\mathcal{I}}^{+}(z^k) \right\}. \tag{1.50}$$

Na fase de viabilidade, podemos usar qualquer algoritmo que minimize a medida de inviabilidade h, podendo ser os mesmos citados para o procedimento de restauração do Algoritmo 1.2. No entanto, estes algoritmos podem falhar se h tiver um ponto estacionário inviável. Nesse caso, o método de filtro para sem sucesso.

Para determinar o passo de otimalidade t, Gonzaga, Karas e Vanti [16] propuseram um algoritmo de região de confiança que resolve aproximadamente o problema quadrático

$$\begin{array}{ll} \text{minimizar} & m_k(z^k + t) \\ \text{sujeito a} & z^k + t \in \mathcal{L}^{RI}(z^k) \\ & \|t\| \leq \Delta, \end{array} \tag{1.51}$$

onde

$$m_k(z^k + t) = f(z^k) + \nabla f(z^k)^T t + \frac{1}{2} t^T B_k t, \tag{1.52}$$

com B_k simétrica.

Agora vamos descrever o algoritmo baseado em Restauração Inexata que calcula o passo $x^{k+1} \notin \bar{\mathcal{F}}_k$.

Algoritmo 1.3 *Cálculo de $x^{k+1} \notin \bar{\mathcal{F}}_k$ por Restauração Inexata.*

Dados: $x^k \in \mathbb{R}^n$, $\bar{\mathcal{F}}_k$, $\Delta_{\min} > 0$, $\Delta \geq \Delta_{\min}$ e $\eta, \gamma \in (0, 1)$.
Fase de viabilidade:
SE $h(x^k) = 0$
faça $z^k = x^k$.
SENÃO
calcule $z^k \notin \bar{\mathcal{F}}_k$ tal que $h(z^k) < (1 - \alpha)h(x^k)$.
SE impossível, pare sem sucesso.
Fase de otimalidade:
SE z^k for estacionário
pare com sucesso.
SENÃO
REPITA
Encontre t, solução aproximada de (1.51).
Determine $ared = f(z^k) - f(z^k + t)$ e $pred = m_k(z^k) - m_k(z^k + t)$.
SE $z^k + t \notin \bar{\mathcal{F}}_k$ E $ared \geq \eta\, pred$,
faça $x^{k+1} = z^k + t$, $\Delta_k = \Delta$ e pare com sucesso.

SENÃO

$$\Delta = \gamma\Delta$$

Gonzaga, Karas e Vanti [16] provaram que se x^{k+1} é obtido pelo Algoritmo 1.3 e o filtro original é usado, então a seguinte condição é satisfeita:

Condição H: *Dado um ponto viável não estacionário $\bar{x} \in X$, existem $M > 0$ e uma vizinhança V de $\bar{x}$ tais que para qualquer iterando $x^k \in V$,*

$$f(x^k) - f(x^{k+1}) \geq M\sqrt{H_k},$$

onde $H_k = \min\left\{1, \min\left\{(1-\alpha)h^j \mid \left(f^j, h^j\right) \in F_k, f^j \leq f(x^k)\right\}\right\}$.

Por outro lado, Karas, Oening e Ribeiro [27] provaram esse mesmo resultado considerando o filtro inclinado.

Pela definição de H_k e v_k, temos que $\sqrt{H_k} \geq v_k$. Portanto, podemos concluir que se x^{k+1} é calculado pelo Algoritmo 1.3, a Hipótese H3 também é satisfeita.

Na próxima seção apresentamos exemplos gráficos que ilustram o funcionamento do Algoritmo 1.1 com o passo calculado por PQS e por Restauração Inexata.

1.4 Exemplos gráficos

Considere o problema bidimensional

$$\begin{aligned} \text{minimizar} \quad & (x_1+1)^2 + (x_2-2)^2 \\ \text{sujeito a} \quad & (x_1-2)^2 - x_2 + 2 \leq 0 \\ & -x_1 + x_2 - 2 \leq 0. \end{aligned} \tag{1.53}$$

A Figura 1.5 mostra o conjunto viável, as curvas de nível de f e o minimizador $x^* = (1, 3)$ do problema acima.

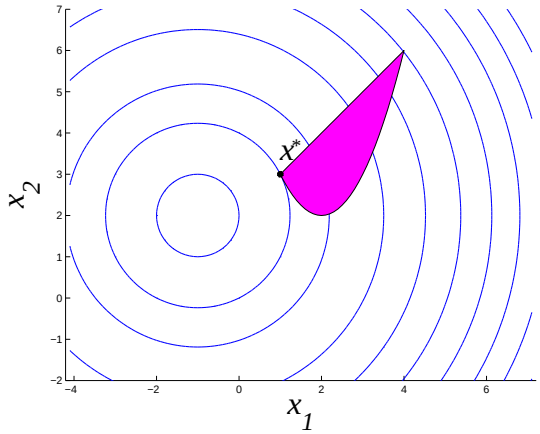

Figura 1.5: Conjunto viável.

Vamos ilustrar a seguir os mecanismos do Algoritmo 1.1 com o passo calculado pelos Algoritmos 1.2 e 1.3 utilizando o critério de filtro original, aplicado ao problema (1.53). Nosso objetivo com essa ilustração é mostrar cada uma das fases dos algoritmos de PQS e RI, bem como visualizar as regiões proibidas nos planos $x_1 \times x_2$ e $f \times h$. Para facilitar a compreensão das figuras que seguem, observe a Figura 1.6. À esquerda dessa figura indicamos um ponto corrente x^k pelo símbolo ★ e, à direita, esse mesmo símbolo representa o par (f^k, h^k), ou seja, o par ordenado correspondente às medidas de otimalidade e inviabilidade avaliadas no ponto corrente. Pelo mecanismo do nosso algoritmo geral de filtro, dado um ponto corrente, construímos uma região em $\mathbb{R}^2$ temporariamente proibida por esse ponto dada pelo conjunto $\mathcal{R}_k$, que poderá tornar-se uma região permanentemente proibida caso a iteração k seja uma iteração do tipo h. Na figura à esquerda, temos o conjunto $\mathcal{R}_k$, indicado pela região hachurada em azul.

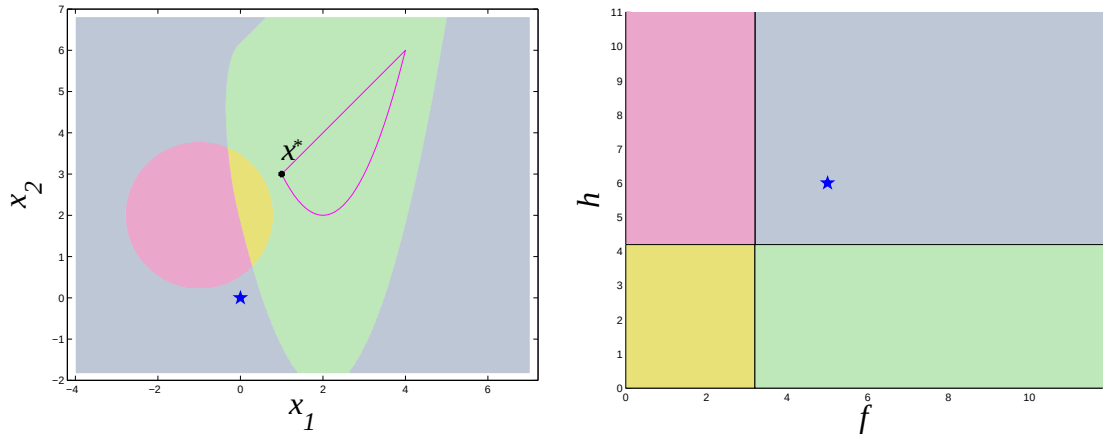

Figura 1.6: Região proibida pelo filtro original.

Na figura à direita observamos a região temporariamente proibida no plano $f \times h$, formada pelos pares (f, h) para os quais $f \geq f^k - \alpha h^k$ e $h \geq (1-\alpha)h^k$, indicada em azul. Nas demais regiões de ambos os planos temos os pontos que serão aceitos pelo filtro, uma vez que provocam uma redução em pelo menos uma das medidas de otimalidade e inviabilidade, em relação ao ponto corrente.

Na Figura 1.7 podemos observar algumas diferenças entre os critérios de filtro. O triângulo em vermelho na figura à direita indica a região aceita pelo filtro original e proibida pelo filtro inclinado. Já a região triangular hachurada em verde representa a região proibida pelo filtro original e aceita pelo filtro inclinado. As regiões correspondentes a essas no plano $x_1 \times x_2$ estão indicadas na figura à esquerda pelas mesmas cores. A figura sugere que o critério de filtro inclinado é menos tolerante do que o original.

Vamos apresentar agora algumas iterações do Algoritmo 1.1 considerando o critério de filtro original, com o passo sendo calculado por PQS e por RI.

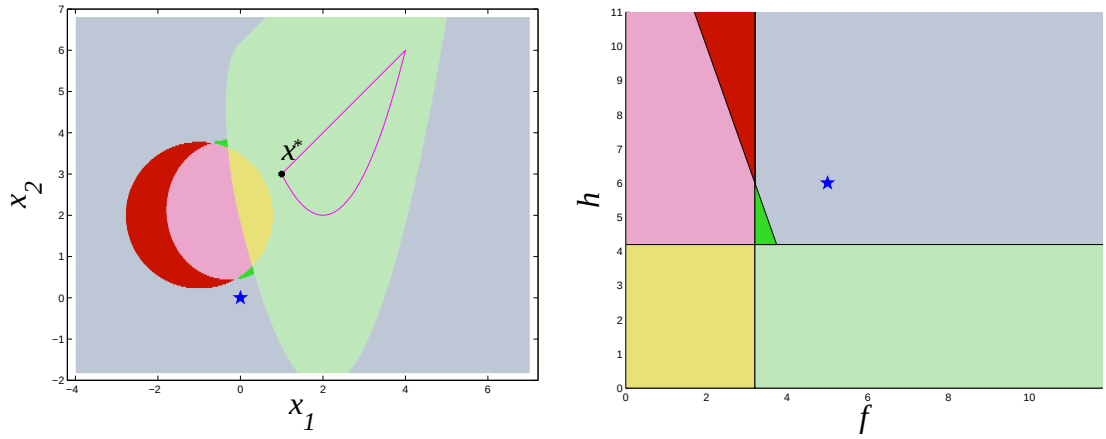

Figura 1.7: Comparação entre filtro original e inclinado.

1.4.1 Passo calculado por PQS

As Figuras 1.8-1.10 ilustram algumas iterações do algoritmo de filtro original, com o passo calculado por PQS (Algoritmo 1.2). Os pontos obtidos durante a execução do algoritmo estão representados pelos seguintes símbolos: ★ (ponto corrente); ● (ponto obtido após o passo de viabilidade); ♦ (ponto obtido após o passo de otimalidade); ▼ (ponto obtido pelo procedimento de restauração).

A Figura 1.8 ilustra a primeira iteração do Algoritmo 1.1. No lado esquerdo podemos observar os passos dados nesta iteração, partindo do ponto inicial $x^0 = (0, 1)$. As linhas tracejadas representam a fronteira da região de confiança obtida ao considerar a norma infinito nos subproblemas quadráticos. Observamos que o comprimento do passo de viabilidade, $\|n^0\|$, excedeu o raio $\Delta_0 = 1$ e, assim, x^1 foi obtido pelo procedimento de restauração. A linha em azul representa o comprimento do passo de restauração dado de x^0 (★) a x^1 (▼).

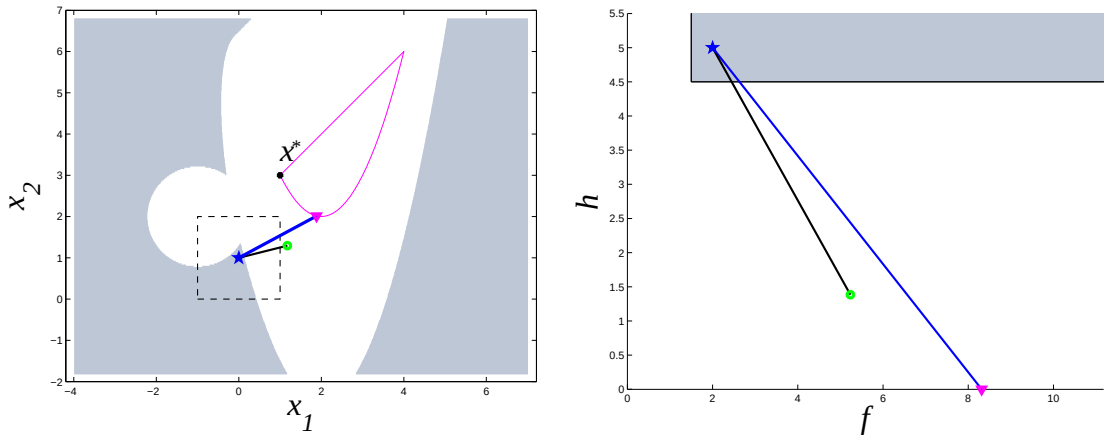

Figura 1.8: Primeira iteração - PQS.

Na Figura 1.8 ilustramos ainda as regiões proibidas em ambos os planos, $x_1 \times x_2$ e $f \times h$, indicadas pela parte hachurada. À esquerda temos o conjunto $\mathcal{R}_0$, formado pelos pontos em $\mathbb{R}^2$ temporariamente proibidos pelo ponto corrente x^0. Do outro lado

observamos a região no plano $f \times h$, ou seja, a região formada pelos pares (f, h) correspondentes aos pontos $x \in \mathcal{R}_0$, e usamos os símbolos ★ e ▼ para representar os pares (f^0, h^0) e (f^1, h^1), respectivamente. Iniciamos essa iteração com $F_0 = \emptyset$ e construímos o filtro temporário $\bar{F}_0 = \{(f^0, h^0)\}$. Como $f^1 > f^0$, concluímos que a primeira iteração é do tipo h e, assim, o filtro temporário se tornará permanente, ou seja, $F_1 = \{(f^0, h^0)\}$. Nas próximas figuras a região permanentemente proibida será hachurada em um tom mais escuro do que a região temporária.

Os passos dados na segunda iteração podem ser observados na Figura 1.9. Como o ponto x^1 é viável, o passo de viabilidade obtido nesta iteração é $n^1 = 0$. Assim, o ponto x^2 é a soma do ponto x^1 com o passo de otimalidade da segunda iteração. Novamente, o ponto tentativo é aceito pelo filtro e, também, pelo critério da redução suficiente na função objetivo. Observando a figura no plano $f \times h$ podemos notar que trata-se de uma iteração do tipo f e, portanto, não há atualização do filtro.

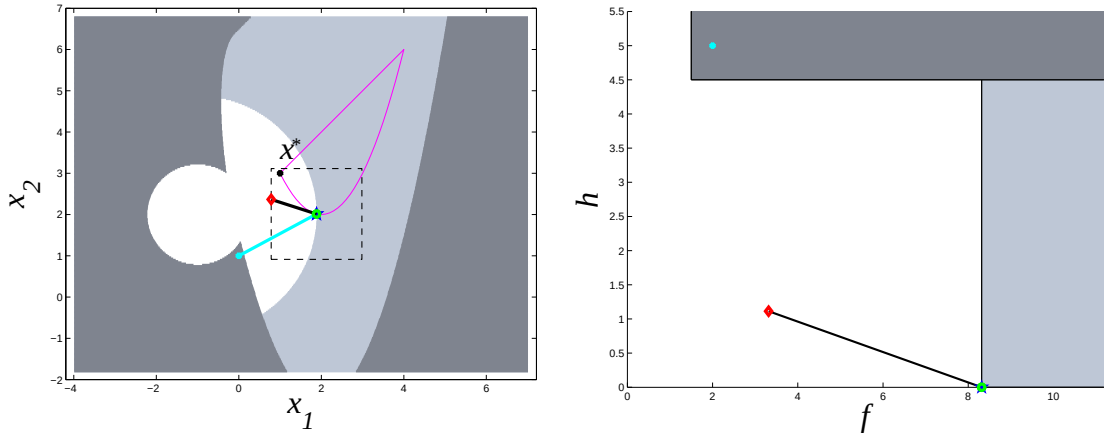

Figura 1.9: Segunda iteração - PQS.

Na Figura 1.10, as linhas em preto no plano $x_1 \times x_2$ indicam os passo de viabilidade e otimalidade, respectivamente, partindo de x^2. No início desta iteração temos $F_2 = \{(f^0, h^0)\}$ e $\bar{F}_2 = F_2 \cup \{(f^2, h^2)\}$. Como podemos observar no plano $f \times h$, a terceira iteração é do tipo h e, portanto, o filtro temporário $\bar{F}_2$ passará a ser permanente. Podemos notar ainda que o iterando x^3 está bem próximo do minimizador do problema (1.53). De fato, com mais três iterações o critério de parada é satisfeito.

1.4.2 Passo calculado por RI

Vamos ilustrar agora o desenvolvimento do Algoritmo 1.1 com o passo calculado por Restauração Inexata (Algoritmo 1.3). Para representar os pontos obtidos nas fases de viabilidade e otimalidade, usamos os símbolos ● e ♦, respectivamente. O ponto corrente está representado novamente por ★.

A Figura 1.11 ilustra a primeira iteração do Algoritmo 1.1 partindo do ponto inicial $x^0 = (1, 1)$. Nessa figura podemos facilmente observar uma importante diferença

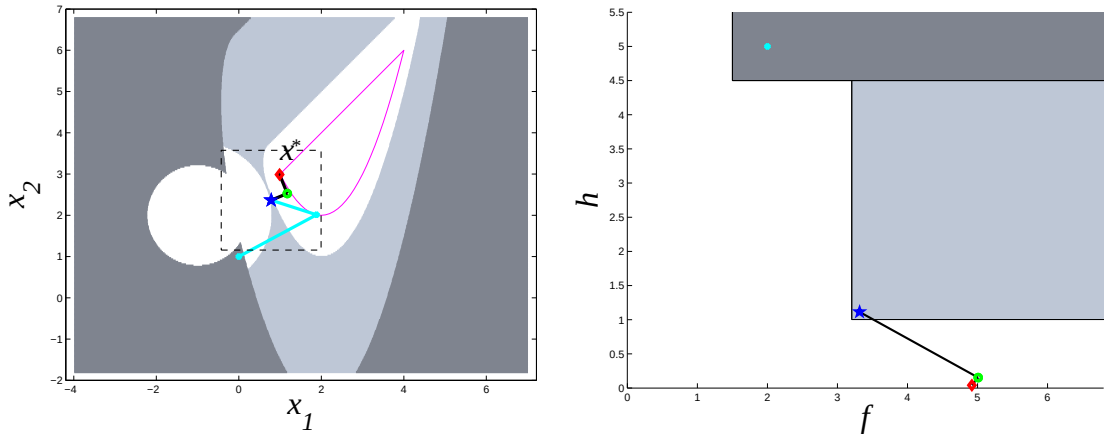

Figura 1.10: Terceira iteração - PQS.

entre os Algoritmos 1.2 e 1.3, a região de confiança deste último é centrada em z^k, ou seja, no ponto obtido após o passo de viabilidade e não no ponto corrente, como ocorre em PQS. Aqui novamente usamos a norma infinito na definição da região de confiança, cuja fronteira está representada pelas linhas tracejadas. No lado esquerdo temos os passos dados em cada uma das duas fases do algoritmo de Restauração Inexata, bem como a região temporariamente proibida em $\mathbb{R}^2$. Já a região proibida no plano $f \times h$ e o filtro temporário $\bar{F}_0 = \{(f^0, h^0)\}$ podem ser observados à direita, onde podemos notar também que a primeira iteração é do tipo f. Assim, não há atualização do filtro, ou seja, $F_1 = F_0 = \emptyset$.

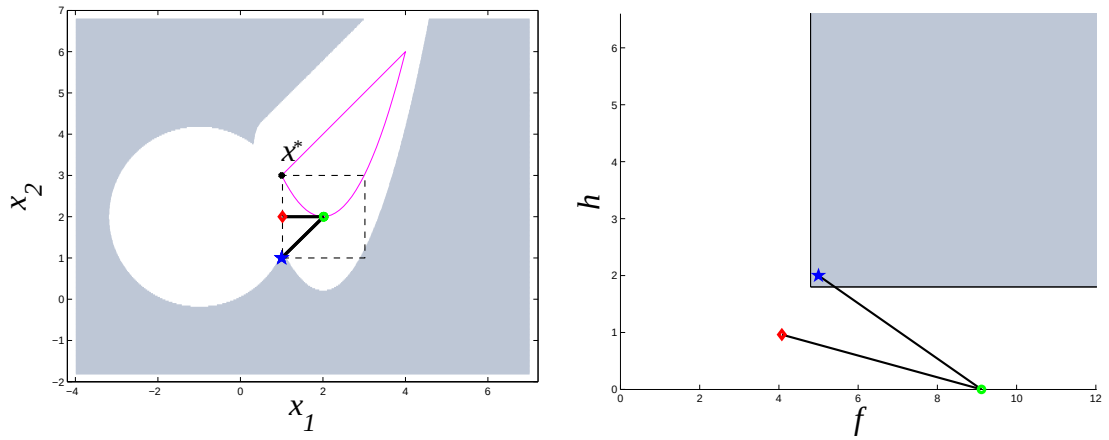

Figura 1.11: Primeira iteração - RI.

A segunda iteração está representada na Figura 1.12. Temos agora uma iteração do tipo h e, portanto, o filtro temporário passa a ser permanente, ou seja, $F_2 = \bar{F}_1 = \{(f^1, h^1)\}$.

Como o ponto x^2 obtido na iteração anterior encontra-se perto do minimizador x^*, os passos dados da terceira iteração são curtos, como podemos verificar na Figura 1.13, na qual são mostradas também as regiões temporária e permanentemente proibidas nos planos $x_1 \times x_2$ e $f \times h$. Temos novamente uma iteração do tipo h e, assim, o

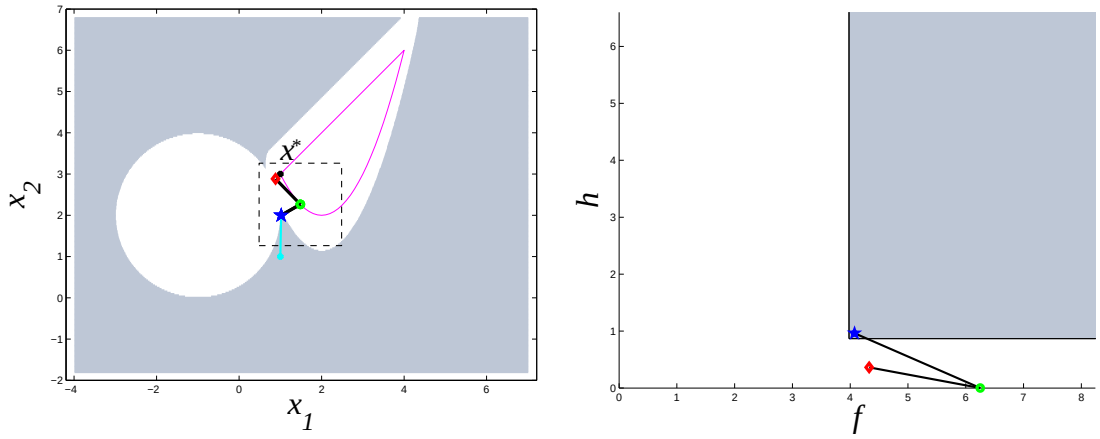

Figura 1.12: Segunda iteração - RI.

filtro temporário $\bar{F}_2 = \{(f^1, h^1), (f^2, h^2)\}$ se tornará permanente.

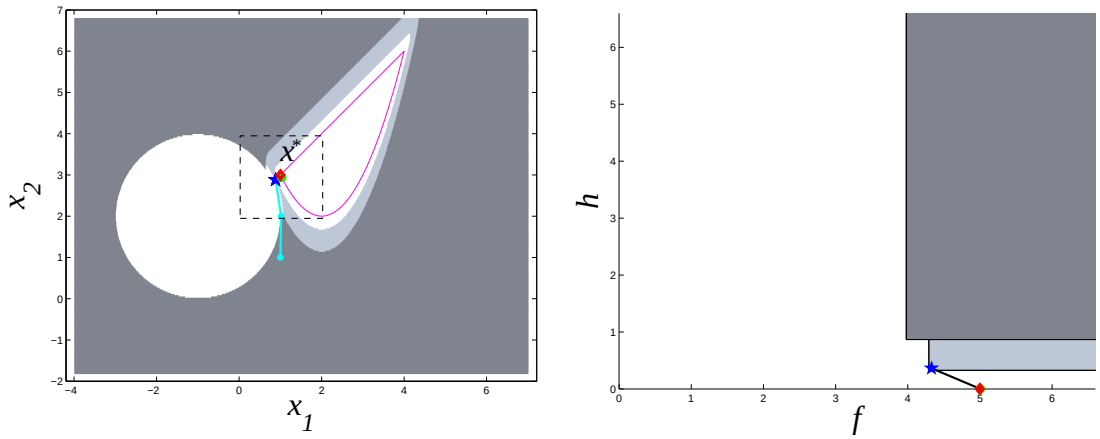

Figura 1.13: Terceira iteração - RI.

Notamos na Figura 1.13 que x^3 está bem próximo do minimizador. De fato, com mais duas iterações o problema (1.53) é solucionado.

Capítulo 2

Detalhes da implementação

Apresentamos neste capítulo detalhes da implementação do algoritmo geral de filtro, bem como dos algoritmos internos usados para o cálculo do passo, descritos no capítulo anterior. Os programas foram escritos em Matlab 7.8.0 (R2009a) para Linux e os testes numéricos foram realizados com problemas selecionados da coleção CUTEr [18], cujos resultados são analisados no próximo capítulo.

Inicialmente apresentamos as sub-rotinas utilizadas para testar se um ponto tentativo é aceito pelo filtro e para a atualização do filtro. Em seguida discutimos a implementação dos algoritmos internos utilizados para calcular o passo por PQS e por RI. Finalizamos o capítulo apresentando outras discussões acerca do critério de parada do algoritmo geral de filtro e da atualização da Hessiana do modelo quadrático a ser minimizado na fase de otimalidade.

2.1 Algoritmo geral de filtro

O algoritmo geral de filtro (Algoritmo 1.1) consiste em, dado um ponto inicial, definir o filtro temporário, calcular um ponto não proibido e atualizar o filtro, até que um ponto estacionário seja obtido. A definição do filtro temporário é simples. Começamos com $F_0 = \emptyset$ e no início de cada iteração o par (f^k, h^k) é temporariamente incluído no filtro. No final da iteração, se esta for do tipo h, esse elemento se tornará permanente e os pares (f^j, h^j) para os quais $\mathcal{R}_j \subset \mathcal{R}_k$ serão eliminados do filtro. Se a iteração for do tipo f, o par (f^k, h^k) será descartado, ou seja, não haverá atualização do filtro.

Para avaliar um ponto tentativo pelo critério de filtro original e realizar a atualização deste, implementamos as sub-rotinas apresentadas em [42]. Vamos apresentar agora as sub-rotinas utilizadas para testar se um ponto é proibido pelo filtro inclinado e para atualizá-lo quando necessário. Estas sub-rotinas também estão baseadas naquelas apresentadas em [42].

O Algoritmo 2.1 se caracteriza por não fazer comparações com todos os pares

do filtro corrente F_k ao verificar se um par tentativo (f^+, h^+) é proibido. Para isso, é necessário que os nF pares de F_k estejam ordenados pelas abscissas. Primeiramente verificamos se o par tentativo é aceito pelo par (f^k, h^k) e, em caso afirmativo, testamos a aceitação deste para os demais pares do filtro. A ordenação dos pares em F_k é realizada no momento em que há atualização do filtro, como veremos no Algoritmo 2.2.

Algoritmo 2.1 *Teste do filtro inclinado*

Dados: $F_k = \{(f^1, h^1), \ldots, (f^j, h^j), \ldots, (f^{nF}, h^{nF})\}$, (f^k, h^k), (f^+, h^+), $\alpha \in (0, 1)$.
SE $f^+ + \alpha h^+ \geq f^k$ E $h^+ \geq (1 - \alpha)h^k$,
    $proib = 1$
SENÃO
    $j = nF$;
        ENQUANTO $j > 0$ E $f^+ + \alpha h^+ < f^j$
            $j = j - 1$;
        SE $j > 0$ E $h^+ \geq (1 - \alpha)h^j$
            $proib = 1$;
        SENÃO
            $proib = 0$;

A próxima sub-rotina, utilizada para atualizar o filtro, inclui o par (f^k, h^k) no filtro e remove todos os pares (f^j, h^j) para os quais $f^j \geq f^k$ e $h^j \geq h^k$, o que evita comparações desnecessárias nas iterações posteriores. Além disso, os pares restantes são ordenados pelas abscissas.

Algoritmo 2.2 *Atualização do filtro inclinado*

Dados: $F_k = \{(f^1, h^1), \ldots, (f^j, h^j), \ldots, (f^{nF}, h^{nF})\}$, (f^k, h^k), $\alpha \in (0, 1)$
$j = nF$; $dom = 0$;
ENQUANTO $j > 0$ E $f^k \leq f^j$
    $j = j - 1$;
ENQUANTO $(j + dom + 1) \leq nF$ E $h^k \leq h^{j+dom+1}$
    $dom = dom + 1$;
$nF = nF + 1 - dom$;
SE $nF > j + 1$
    SE $dom = 0$
        PARA $i = nF$ ATÉ $j + 2$, PASSO=-1,
            $f^i = f^{i-1}$;
            $h^i = h^{i-1}$;
    SE $dom \geq 1$
        PARA $i = j + 2$ ATÉ nF,

$$f^i = f^{i+dom-1};$$
$$h^i = h^{i+dom-1};$$
$$f^{j+1} = f^k;$$
$$h^{j+1} = h^k;$$

A atualização do filtro inclinado feita por meio do Algoritmo 2.2 está de acordo com a que foi considerada no Algoritmo 1.1, uma vez que, dados os pares (f^j, h^j) e (f^k, h^k) tais que $f^j \geq f^k$ e $h^j \geq h^k$, temos que $\mathcal{R}_j \subset \mathcal{R}_k$, onde $\mathcal{R}_j$ e $\mathcal{R}_k$ são dados em (1.5). De fato, se $x \in \mathcal{R}_j$, então

$$f(x) + \alpha h(x) \geq f^j \geq f^k \quad \text{e} \quad h(x) \geq (1-\alpha)h^j \geq (1-\alpha)h^k,$$

o que significa que $x \in \mathcal{R}_k$.

É importante observar que quando consideramos o filtro original, dado em (1.4), a propriedade de inclusão estabelecida acima não é válida, ou seja, $f^j \geq f^k$ e $h^j \geq h^k$ não implicam em $\mathcal{R}_j \subset \mathcal{R}_k$. No entanto, o Algoritmo 2.2 também pode ser empregado na atualização do filtro original de modo que a propriedade da inclusão seja válida, desde que as entradas do filtro sejam definidas como os vértices da região proibida no plano $f \times h$, dados por $(\tilde{f}^j, \tilde{h}^j)$, onde $\tilde{f}^j = f^j - \alpha h^j$ e $\tilde{h}^j = (1-\alpha)h^j$, como apresentado em [42].

Na Figura 2.1 ilustramos o fato de que a propriedade de inclusão das regiões proibidas é válida para o filtro inclinado, mas não é satisfeita para o filtro original.

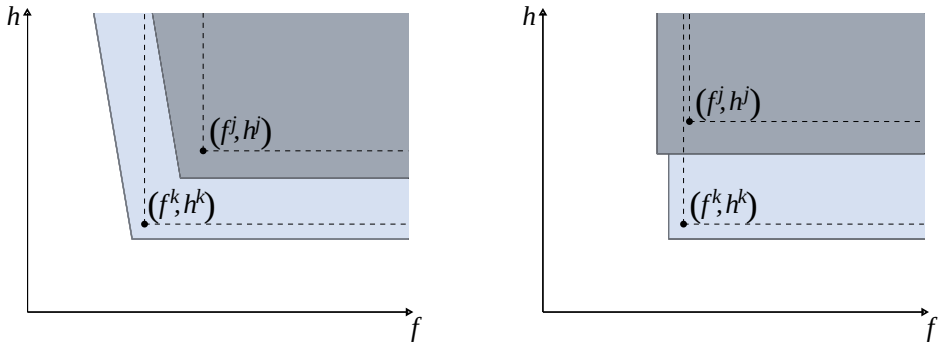

Figura 2.1: Inclusão das regiões proibidas.

Nos testes numéricos fizemos a contagem do número de atualizações do filtro, ou seja, do número de iterações do tipo h e também do número de pares pertencentes ao filtro final, uma vez que este não será necessariamente igual ao número de atualizações devido à limpeza do filtro realizada pelo Algoritmo 2.2. Os resultados observados serão discutidos no próximo capítulo.

2.2 Algoritmos internos

Vamos discutir agora detalhes da implementação dos Algoritmos 1.2 e 1.3, utilizados para calcular o passo a ser avaliado pelo filtro. Lembramos que no Algoritmo 1.2, baseado em PQS, o passo é calculado em duas etapas. Primeiro, calculamos um passo de viabilidade n^k que tem por objetivo reduzir a medida de inviabilidade h. Em seguida, determinamos um passo de otimalidade t_Δ que minimiza o modelo quadrático da função objetivo.

De forma similar, o Algoritmo 1.3 é dividido em duas fases: na primeira determinamos um passo que minimiza a função h e na segunda calculamos um passo de otimalidade, com a diferença de que a região de confiança é centrada no ponto obtido após o passo de viabilidade e não no ponto corrente. Assim, a forma descrita a seguir para determinar o passo de otimalidade é válida tanto para o algoritmo de PQS quanto de RI. Por outro lado, os passos de viabilidade são determinados de maneiras diferentes e, portanto, descrevemos o cálculo desses passos separadamente. Para o procedimento de restauração do Algoritmo 1.2 utilizamos o mesmo algoritmo empregado na fase de viabilidade do Algoritmo 1.3.

2.2.1 Passo de viabilidade

Apresentamos nesta seção a forma com que o passo de viabilidade foi determinado para cada um dos algoritmos de PQS e RI.

Passo de viabilidade do algoritmo de PQS.

O passo de viabilidade n^k deve ser determinado de forma a satisfazer a aproximação linear das restrições dada pelo conjunto

$$\mathcal{L}_k = \left\{ x^k + d \in \mathbb{R}^n \mid c_\mathcal{E}(x^k) + A_\mathcal{E}(x^k)d = 0, c_\mathcal{I}(x^k) + A_\mathcal{I}(x^k)d \leq 0 \right\}, \tag{2.1}$$

onde x^k é o iterando corrente.

A forma utilizada nesse trabalho para calcular o passo n^k é dada por

$$n^k = P_{\mathcal{L}_k}(x^k) - x^k,$$

em que $P_{\mathcal{L}_k}(x^k)$ é a projeção ortogonal de x^k no conjunto $\mathcal{L}_k$. Sendo assim, o passo de viabilidade pode ser obtido resolvendo-se o seguinte problema quadrático

$$\begin{aligned} \text{minimizar} \quad & \frac{1}{2}\|n\|^2 \\ \text{sujeito a} \quad & A_\mathcal{E}(x^k)n + c_\mathcal{E}(x^k) = 0 \\ & A_\mathcal{I}(x^k)n + c_\mathcal{I}(x^k) \leq 0. \end{aligned} \tag{2.2}$$

Para resolver o problema (2.2) utilizamos o comando `quadprog` do Matlab, que resolve

problemas do tipo

$$\begin{aligned} \text{minimizar} \quad & \frac{1}{2}x^T H x + c^T x \\ \text{sujeito a} \quad & Ax = a \\ & Bx \leq b \\ & b_l \leq x \leq b_u. \end{aligned}$$

Entre as possíveis saídas deste comando está `EXITFLAG` $= -2$, indicando que nenhum ponto viável foi encontrado. Nesse caso, concluímos que o conjunto $\mathcal{L}_k$ é vazio e, portanto, o algoritmo chama um procedimento de restauração, cujo objetivo é obter um ponto não proibido pelo filtro corrente que minimiza a medida de inviabilidade h. Tal procedimento será descrito adiante.

É importante citar que os problemas da coleção CUTEr são dados por

$$\begin{aligned} \text{minimizar} \quad & f(x) \\ \text{sujeito a} \quad & c_l \leq c(x) \leq c_u \\ & b_l \leq x \leq b_u. \end{aligned}$$

Dessa forma, foi necessário criar duas sub-rotinas auxiliares para separar as restrições de igualdade e desigualdade, transformando as restrições de desigualdade em restrições do tipo $c_{\mathcal{I}}(x) \leq 0$. Estas sub-rotinas retornam também as matrizes Jacobianas e transformam as restrições de caixa em restrições de desigualdade, embora a estrutura original dessas restrições também possa ser explorada na implementação sem fazer uso dessa transformação.

Passo de viabilidade do algoritmo de RI e procedimento de restauração do algoritmo de PQS.

No capítulo anterior vimos que a fase de viabilidade do Algoritmo 1.3 consiste em determinar um ponto z^k não proibido pelo filtro tal que $h(z^k) < (1-\alpha)h(x^k)$, onde x^k é o ponto corrente. Assim, nessa fase podemos resolver (aproximadamente) o problema

$$\begin{aligned} \text{minimizar} \quad & h(x) \\ \text{sujeito a} \quad & x \in \mathbb{R}^n. \end{aligned} \tag{2.3}$$

O mesmo ocorre quando o Algoritmo 1.2 chama um procedimento de restauração para determinar um ponto x^{k+1}, tal que $h(x^{k+1}) < h(x^k)$, devido ao fato de $\mathcal{L}_k = \emptyset$ ou à incompatibilidade do problema (1.12). Dessa forma, tanto a fase de viabilidade para RI como o procedimento de restauração para PQS podem ser vistos como um problema de viabilidade, ou seja, o problema de encontrar um vetor $x \in \mathbb{R}^n$ tal que

$$c_{\mathcal{E}}(x) = 0 \quad \text{e} \quad c_{\mathcal{I}}(x) \leq 0,$$

onde $c_{\mathcal{E}}(x)$ e $c_{\mathcal{I}}(x)$ são funções diferenciáveis de $\mathbb{R}^n$ em $\mathbb{R}^p$ e $\mathbb{R}^q$, respectivamente.

Para solucionar esse problema empregamos o algoritmo de filtro multidimensional proposto por Gould, Leyffer e Toint [17]. O objetivo desse método é encontrar um minimizador local da violação das restrições. Assim, o problema (2.3) pode ser reformulado como

$$\begin{aligned} \text{minimizar} \quad & \theta(x) \\ \text{sujeito a} \quad & x \in \mathbb{R}^n, \end{aligned} \tag{2.4}$$

onde

$$\theta(x) = \frac{1}{2} h(x)^2 = \frac{1}{2} \left\| c^+(x) \right\|^2 \tag{2.5}$$

e $\|\cdot\|$ denota a norma euclidiana.

Em [17] é apresentada a prova da convergência global para o algoritmo de filtro multidimensional que combina região de confiança e técnicas de filtro, considerando o caso particular em que há apenas restrições de igualdade. Porém, os autores citam que esse algoritmo também pode ser aplicado a problemas com restrições de desigualdade. Os aspectos práticos desse método são discutidos em [20], no qual os autores salientam que a técnica que leva em consideração as restrições de desigualdade é uma heurística e nenhuma garantia de convergência pode ser apresentada nesse caso. Entretanto, experimentos numéricos indicam que essa heurística funciona bem na prática.

O algoritmo consiste em, dado um ponto corrente x^k, calcular um passo s^k e definir o ponto tentativo

$$x^+ = x^k + s^k$$

que poderá ou não ser aceito como o próximo iterando x^{k+1}.

O passo s^k é obtido minimizando-se um modelo quadrático da função objetivo do problema (2.4) em uma vizinhança do iterando corrente. Neste trabalho consideramos o modelo de Gauss-Newton

$$m_k^{GN}(x^k + s) = \frac{1}{2} \left\| \bar{c}(x^k) + J_{\bar{c}}(x^k) s \right\|^2,$$

onde $\bar{c}(x^k)$ é o vetor formado por $c_{\mathcal{E}}(x^k)$ e pelas componentes não nulas de $c_{\mathcal{I}}^+(x^k)$ e $J_{\bar{c}}(x^k)$ é a jacobiana de $\bar{c}(x^k)$. Dessa forma, s^k é calculado resolvendo-se o subproblema

$$\begin{aligned} \text{minimizar} \quad & m_k^{GN}(x^k + s) \\ \text{sujeito a} \quad & \|s\| \leq \Delta_k, \end{aligned} \tag{2.6}$$

onde Δ_k é o raio da região de confiança.

Para a prova da convergência global do algoritmo, assim como nos demais métodos de região de confiança, é necessário que o passo s^k forneça um decréscimo suficiente no modelo, dado pela condição

$$m_k^{GN}(x^k) - m_k^{GN}(x^k + s^k) \geq \kappa \|g_k\| \min \left[\frac{\|g_k\|}{\beta_k}, \Delta_k \right], \tag{2.7}$$

onde $g_k = \nabla m_k^{GN}(x^k)$, $\kappa \in (0,1)$ e $\beta_k > 0$ é um limitante superior para a norma da Hessiana de m_k^{GN}.

Filtro multidimensional.

Ao contrário do que acontece no filtro bidimensional, no problema de viabilidade não consideramos uma função objetivo, mas ainda enfrentamos objetivos conflitantes, uma vez que desejamos levar cada uma das componentes $\{c_i^+(x)\}_{i=1}^m$ para zero, como uma questão independente. Dessa forma, é conveniente considerar um filtro multidimensional.

Para definir o filtro, precisamos primeiramente apresentar a regra de dominação no caso multidimensional. Nesse caso, dizemos que um ponto x domina um ponto y sempre que

$$\left|c_i^+(x)\right| \leq \left|c_i^+(y)\right| \quad \text{para todo} \quad i \in \{1,\ldots,m\}.$$

Assim, denotando o vetor m-dimensional $c^+(x^k)$ por c_k^+, definimos o filtro como uma lista $\mathcal{F}_m$ de vetores da forma $\left\{c_1^+, c_2^+, \cdots\right\}$ tal que, para cada par c_k^+, $c_l^+ \in \mathcal{F}_m$ com $k \neq l$,

$$\left|c_{j,k}^+\right| < \left|c_{j,l}^+\right| \quad \text{para pelo menos um} \quad j \in \{1,\ldots,m\},$$

onde $c_{j,k}^+$ é a $j-$ésima componente de c_k^+.

O ponto tentativo x^+ é aceito se não for dominado por nenhum outro ponto pertencente ao filtro. Porém, para tornar o algoritmo de filtro mais eficiente, assim como no caso bidimensional, uma leve modificação nesse critério de aceitação é introduzida. Assim, dizemos que um ponto tentativo x^+ é aceito pelo filtro $\mathcal{F}_m$ se, e somente se

$$\forall c_l^+ \in \mathcal{F}_m \quad \exists i \in \{1,\ldots,m\} \quad \text{tal que} \quad \left|c_i^+(x^+)\right| < \left[\left|c_{i,l}^+\right| - \gamma_\theta \|c_l^+\|\right]_+, \tag{2.8}$$

onde $\gamma_\theta \in \left(0, \frac{1}{\sqrt{m}}\right)$ e $[w]_+ = \max[0,w]$. Outros possíveis critérios de aceitação podem ser encontrados em [20].

A fim de evitar ciclos, um ponto tentativo que satisfaz (2.8) pode ser adicionado ao filtro. Além disso, para simplificar as comparações posteriores, os pontos dominados pela nova entrada do filtro são excluídos.

Se o ponto tentativo não é aceito pelo filtro, este ainda pode ser aceito pelo mecanismo usual de região de confiança, como será descrito no Algoritmo 2.3.

Algoritmo 2.3 *Fase de viabilidade*

1. *Inicialização*
   Dados: $x^0 \in \mathbb{R}^n$, $\Delta_0 > 0$, $0 < \gamma_0 \leq \gamma_1 < 1 \leq \gamma_2$, $\gamma_\theta \in \left(0, \frac{1}{\sqrt{m}}\right)$,
   $0 < \eta_1 < \eta_2 < 1$.
   Calcule $c^+(x^0)$ e defina $k = 0$ e $\mathcal{F}_m = \emptyset$.

2. *Critério de Parada*
   Se $h(x^k) \leq \varepsilon$ ou $\left\|\nabla\theta(x^k)\right\| \leq \varepsilon$ PARE.

3. *Determine o ponto tentativo*
   Determine s^k como uma solução aproximada para o problema (2.6).
   Calcule o ponto tentativo $x^+ = x^k + s^k$ e avalie $c^+(x^+)$.
   Defina
   $$\rho_k = \frac{\theta(x^k) - \theta(x^+)}{m_k^{GN}(x^k) - m_k^{GN}(x^+)}.$$

4. *Verifique se o ponto tentativo é aceito*

   (a) Se x^+ é aceito pelo filtro corrente:
   Faça $x^{k+1} = x^+$ e adicione $c^+(x^+)$ ao filtro se $\rho_k < \eta_1$

   (b) Se x^+ não é aceito pelo filtro corrente:
   Se $\rho_k \geq \eta_1$, faça $x^{k+1} = x^+$.
   Senão, faça $x^{k+1} = x^k$.

5. *Atualize o raio da região de confiança*

$$\Delta_{k+1} \in \begin{cases} [\gamma_0\Delta_k, \gamma_1\Delta_k] & \text{se } \rho_k < \eta_1, \\ [\gamma_1\Delta_k, \Delta_k] & \text{se } \rho_k \in [\eta_1, \eta_2), \\ [\Delta_k, \gamma_2\Delta_k] & \text{se } \rho_k \geq \eta_2. \end{cases}$$

Para resolver o problema (2.6), utilizamos o algoritmo de Gradiente Conjugado-Steihaug, como apresentado em [37].

Os valores escolhidos para as constantes do Algoritmo 2.3 foram:

$$\gamma_0 = 0,0625,\ \ \gamma_1 = 0,25,\ \ \gamma_2 = 2,\ \ \eta_1 = 0,01,\ \ \eta_2 = 0,9\ ,\ \ \Delta_0 = 1 \text{ e } \varepsilon = 10^{-6}.$$

Falha na fase de viabilidade.

O método apresentado aqui, bem como as demais técnicas que podem ser utilizadas no procedimento de restauração e na fase de viabilidade dos Algoritmos 1.2 e 1.3, respectivamente, garantem apenas a convergência para pontos críticos de primeira ordem da função h. Dessa forma, tal procedimento pode falhar quando o ponto crítico encontrado é inviável [10]. Os problemas apresentados a seguir ilustram essa situação.

Problema 1:

$$(P1) \qquad \begin{aligned} \text{minimizar} \quad & x \\ \text{sujeito a} \quad & c(x) = (x-1)^2 + 1 \leq 0 \\ & x \in \mathbb{R} \end{aligned}$$

Nesse caso, a medida de inviabilidade é dada por $h(x) = (x-1)^2 + 1$. Graficamente, temos

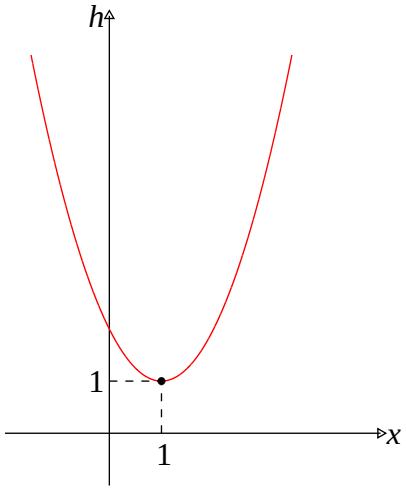

Figura 2.2: Medida de inviabilidade para o problema $(P1)$.

Observando a Figura 2.2, podemos concluir que trata-se de um problema incompatível e, portanto, é de se esperar que o algoritmo pare em um ponto que minimiza a medida de inviabilidade.

Ao rodar o Algoritmo 1.1 com o passo calculado por PQS, partindo de diferentes pontos iniciais, constatamos que o algoritmo se comporta da maneira esperada. Em poucas iterações a sequência de iterandos converge para o minimizador global de h, $x^* = 1$.

Problema 2:

$$(P2) \qquad \begin{array}{rl} \text{minimizar} & x \\ \text{sujeito a} & c(x) = 3 + \frac{1}{2}x - \frac{1}{6}x^3 \leq 0 \\ & x \in \mathbb{R} \end{array}$$

Para esse problema temos que $h(x) = \max\{0, c(x)\}$.

O gráfico de h é apresentado na Figura 2.3, onde

$$A = \left(-1, \frac{8}{3}\right), B = \left(1, \frac{10}{3}\right) \text{ e } C = (3, 0).$$

Podemos observar que a solução para o problema $(P2)$ é $x^* = 3$. Além disso, por meio da Figura 2.3 verificamos que h possui dois pontos estacionários inviáveis, A e B. Para verificar o comportamento do algoritmo nesse caso, escolhemos pontos iniciais nos intervalos $(-\infty, 1)$ e $(1, +\infty)$. Partindo de pontos localizados no primeiro intervalo, não muito próximos de $x = 1$, o algoritmo converge para o minimizador local de h, $x = -1$ e, tomando pontos iniciais no segundo intervalo, a sequência de iterandos converge para a solução do problema, $x^* = 3$. O algoritmo converge para

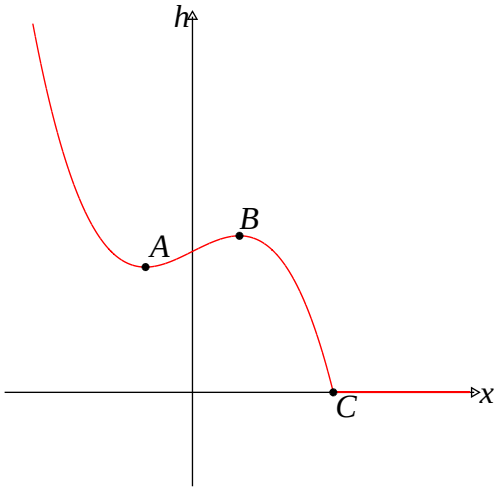

Figura 2.3: Medida de inviabilidade para o problema ($P2$).

o maximizador local de h, $x = 1$, apenas nos casos em que x^0 está suficientemente próximo deste ponto.

2.2.2 Passo de otimalidade

Vimos anteriormente que o passo de otimalidade, de ambos os algoritmos de PQS e RI, é determinado como uma solução (aproximada) do problema quadrático com região de confiança (1.17) e (1.51), respectivamente.

A norma utilizada para definir a região de confiança é arbitrária. Porém, escolhemos trabalhar com a norma $\|\cdot\|_\infty$, pois assim os problemas (1.17) e (1.51) continuam sendo de programação quadrática, o que não ocorreria se escolhêssemos a norma euclidiana, por exemplo. Assim, a restrição da região de confiança torna-se uma restrição de caixa, possibilitando novamente a utilização do comando `quadprog`.

Os multiplicadores associados à solução dos subproblemas quadráticos foram considerados como estimativas para os multiplicadores de Lagrange λ_k, necessários na atualização da matriz B_k, aproximação da Hessiana da Lagrangiana, e na verificação do critério de parada.

2.3 Outras discussões

Discutimos a seguir a atualização da Hessiana dos modelos quadráticos (1.10) e (1.52) considerados nos algoritmos de PQS e RI, respectivamente. Além disso, apresentamos o critério de parada utilizado na implementação.

Atualização da Hessiana dos modelos quadráticos.

Os Algoritmos 1.2 e 1.3 iniciam com $B_0 = I$, onde I é a matriz identidade com dimensão apropriada. A atualização dessa matriz em cada iteração é feita pela fórmula

BFGS [37], descrita a seguir.

Defina

$$s_k = x^{k+1} - x^k \quad \text{e} \quad \hat{y}_k = \nabla_x \mathcal{L}(x^{k+1}, \lambda_{k+1}) - \nabla_x \mathcal{L}(x^k, \lambda_{k+1}),$$

e determine

$$y_k = \begin{cases} \hat{y}_k, & \text{se } \hat{y}_k^T s_k \geq 0,2 s_k^T B_k s_k \\ \delta_k \hat{y}_k + (1-\delta_k) B_k s_k, & \text{caso contrário,} \end{cases}$$

onde δ_k é dado por

$$\delta_k = \frac{0,8 s_k^T B_k s_k}{s_k^T B_k s_k - s_k^T \hat{y}_k}.$$

A atualização de B_k é obtida por meio da seguinte fórmula

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{s_k^T y_k}.$$

De acordo com Shen, Xue e Chen [47], a Hessiana da Lagrangiana pode se tornar mal condicionada quando $\{\lambda_k\}$ é ilimitada e, para evitar o mau condicionamento de B_k, os autores impuseram um limite sobre $(y_k^T y_k)/(s_k^T y_k)$, de modo que a violação deste implicaria em $B_{k+1} = B_k$.

Nos testes numéricos iniciais para o algoritmo proposto nesse trabalho, observamos que o mau condicionamento da matriz B_k comprometia a execução do comando `quadprog`. Assim, seguindo a sugestão dos referidos autores, não atualizamos a matriz B_k nas iterações em que $(y_k^T y_k)/(s_k^T y_k) > 10^{10}$.

Critério de parada.

O algoritmo geral de filtro pára quando encontra um ponto cujo valor da medida de estacionaridade é suficientemente pequeno. Dessa forma, verificamos as seguintes condições

$$\left(\left\| \nabla f(x^k) + \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_{k,i} \nabla c_i(x^k) \right\|_\infty \leq \varepsilon \text{ ou } \left\| d^c(x^k, x^k) \right\|_\infty \leq \varepsilon \right) \qquad \text{e} \qquad h(x^k) \leq \varepsilon, \tag{2.9}$$

onde d^c é dada em (1.18) e $\varepsilon = 10^{-6}$ é a tolerância estabelecida. Além disso, a execução do algoritmo é encerrada se essas condições não forem satisfeitas até que o algoritmo geral de filtro atinja 1000 iterações ou 1 hora de tempo de CPU.

Capítulo 3

Resultados numéricos

Apresentamos neste capítulo os resultados dos testes numéricos realizados com 300 problemas da coleção CUTEr [18], para quatro variantes do Algoritmo 1.1 de acordo com o algoritmo empregado para determinar o passo e o critério de filtro utilizado: PQS-Original, PQS-Inclinado, RI-Original e RI-Inclinado. Os testes foram executados em um processador Intel(R) Core(TM)2 Duo CPU T5870 2,00GHz e 3,00GB de memória RAM. Analisamos aqui o desempenho de cada uma das variantes e comparamos os resultados.

No Apêndice A estão apresentados os nomes dos problemas selecionados e seus respectivos números de variáveis (n) e restrições de igualdade (p) e desigualdade (q). Os pontos iniciais considerados nos testes foram aqueles fornecidos pela CUTEr e os problemas possuem dimensões entre 2 e 203 variáveis.

Ao realizar os testes preliminares, identificamos sete situações nas quais foi necessário interromper a execução dos algoritmos. Assim, definimos um parâmetro de saída para as rotinas escritas em MATLAB, que pode assumir os seguintes valores:

-4: o algoritmo de filtro multidimensional alcançou o número máximo de iterações ($k = 1000$);
-3: um valor não numérico foi encontrado;
-2: o tempo limite (1 hora) foi alcançado;
-1: o número máximo de iterações ($k = 1000$) do algoritmo geral de filtro foi alcançado;
0: o critério de parada (2.9) foi satisfeito;
1: o raio da região de confiança tornou-se muito pequeno;
2: o procedimento de restauração retornou um ponto inviável, estacionário de h.

Nas próximas seções discutimos a escolha dos parâmetros usados na implementação dos Algoritmos 1.1, 1.2 e 1.3 e analisamos os resultados alcançados para os 300 problemas considerados.

3.1 Escolha dos parâmetros

Nos Algoritmos 1.1, 1.2 e 1.3 temos constantes cujos valores não foram definidos até o momento. Uma delas é o tamanho da margem considerada no filtro. A fim de verificar a influência dessa constante no desempenho dos algoritmos, testamos quatro valores diferentes para α: 10^{-1}, 10^{-2}, 10^{-3} e 10^{-4}. Para isso, escolhemos 93 problemas da coleção Hock-Schittkowski, que fazem parte dos 300 problemas selecionados da CUTEr. A Tabela 3.1 apresenta o número de problemas que alcançaram cada um dos possíveis valores do parâmetro de saída para o Algoritmo 1.1, com o passo calculado por PQS e por RI. Organizamos os dados em uma mesma tabela, pois não foram observadas diferenças entre os resultados para os quatro valores de α.

Saída	PQS		RI	
	Original	Inclinado	Original	Inclinado
-4	0	0	0	0
-3	3	3	3	3
-2	0	0	0	0
-1	0	0	0	0
0	84	84	79	79
1	1	1	4	4
2	5	5	7	7

Tabela 3.1: Saídas dos problemas.

Diante destes resultados, decidimos escolher o valor de α com base na análise do número de iterações e de avaliações de funções gastos por cada algoritmo. As diferenças observadas entre essas medidas de desempenho foram pequenas, mas os testes indicaram que valores menores de α fornecem melhores resultados. Assim, seguindo também a sugestão de Fletcher *et al.* [10], fixamos $\alpha = 0,0001$.

Para as demais constantes, escolhemos os valores:

$$\Delta_0 = 1,\ c_p = 10^{-4},\ \xi = 0,8,\ \eta = 0,01,\ \gamma = 0,5.$$

Ao iniciar a fase de otimalidade dos Algoritmos 1.2 e 1.3, definimos o raio da região de confiança como sendo o máximo entre Δ_0 e o dobro do raio obtido na iteração anterior.

3.2 Análise dos resultados

Na Tabela 3.2 apresentamos os resultados observados para os 300 problemas, de acordo com os valores do parâmetro de saída.

Podemos notar que embora os valores do parâmetro de saída não dependam da escolha do critério de filtro, estes são influenciados pelo algoritmo interno usado para

Saída	PQS		RI	
	Original	Inclinado	Original	Inclinado
-4	2	2	3	3
-3	5	5	5	5
-2	2	2	2	2
-1	2	2	3	3
0	278	278	252	252
1	2	2	9	9
2	9	9	26	26

Tabela 3.2: Valores do parâmetro de saída.

calcular o passo. Os algoritmos de PQS obtiveram sucesso, ou seja, atingiram o critério de parada (2.9), em $92,67\%$ dos problemas, enquanto que a porcentagem reduziu para 84% quando consideramos os algoritmos de RI. Todos os algoritmos falharam ao tentar resolver os problemas HIMMELBJ, HS101, HS102, HS103 e LIN, pois encontraram um valor não numérico (NAN) durante suas execuções.

Os problemas CRESC4 e POLAK2 não foram resolvidos com sucesso pelos algoritmos de PQS, pois o número máximo de iterações estipulado para o algoritmo de filtro multidimensional, usado no procedimento de restauração, foi atingido e assim a execução do algoritmo geral de filtro foi interrompida. O mesmo aconteceu na fase de viabilidade dos algoritmos de RI ao tentarem resolver os problemas CRESC100, POLAK2 e SPIRAL. Todos os algoritmos excederam o tempo máximo de CPU (1 hora) para os problemas HAIFAM e HYDROELS, embora tenham chegado próximo das soluções destes. Os problemas AVION2 e SMBANK alcançaram o número máximo de iterações do Algoritmo 1.1 com o passo calculado por PQS. Pela mesma razão, este algoritmo também falhou para os problemas BT1, MSS1 e SMBANK, quando o passo foi calculado por RI.

Para alguns problemas, apesar da solução corrente parecer ótima, os algoritmos não foram capazes de satisfazer o critério de parada (2.9). Para os algoritmos de PQS, os problemas que pararam nesse caso foram HS87 e S268 e, para RI foram AIRPORT, AVION2, DIXCHLNG, DNIEPER, EQC, HS107, HS69, HS87 e HS99. Estes são os problemas que apresentaram o valor 1 para o parâmetro de saída, ou seja, o raio da região de confiança ficou bem pequeno, a ponto de não ser observada redução suficiente na função objetivo.

A falha no procedimento de restauração e na fase de viabilidade, prevista na teoria e ilustrada no capítulo anterior, ocorreu para 9 problemas quando o passo foi determinado pelo Algoritmo 1.2, sendo eles: DIXCHLNG, HS61, HS88, HS89, HS90, HS92, LOOSTMA, POWELLSQ e S316-322. Já para os algoritmos de RI esta falha foi observada ao tentarmos resolver os problemas BT13, CHANDHEQ, CLUSTER, DEGENLPA, DEGENLPB, EIGMAXB, EIGMINB, HATFLDF, HS64, HS75, HS88, HS89, HS90, HS91, HS92, LAUNCH, LEAKNET, LOOTSMA, METHANB8,

POWELLBS, POWELLSQ, RECIPE, S277-280, S316-322, SWOPF e TRIGGER. Esta diferença entre o número de problemas que apresentaram saída 2 para PQS e RI pode ser justificada pelo fato de que o Algoritmo 1.3 chama o algoritmo de filtro multidimensional em toda iteração, a menos que o ponto corrente seja viável. Por outro lado, quando calculamos o passo por PQS, esse algoritmo é executado apenas quando é necessário realizar o procedimento de restauração, o que obviamente ocorre com menor frequência.

Para facilitar a comparação entre as quatro variantes do Algoritmo 1.1, construímos os gráficos de desempenho com relação ao número de iterações do algoritmo geral de filtro, número de avaliações de funções e tempo de processamento (CPU). Esta representação gráfica, introduzida por Dolan e Moré [9], fornece um meio de avaliar e comparar o desempenho de um conjunto $\mathcal{S}$ de algoritmos aplicados a um conjunto $\mathcal{P}$ de n_p problemas testes. Por exemplo, seja $t_{p,s}$ o tempo de processamento necessário para resolver o problema $p \in \mathcal{P}$ pelo algoritmo $s \in \mathcal{S}$ e $r_{p,s}$ o índice de desempenho dado por

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} \mid s \in \mathcal{S}\}}.$$

Considere um parâmetro $r_M \geq \max\{r_{p,s}\}$. Assim, definimos $r_{p,s} = r_M$ sempre que o algoritmo s não resolve o problema p.

Seja

$$\mathcal{K}(r_{p,s}, \tau) = \begin{cases} 1 & \text{se } r_{p,s} \leq \tau \\ 0 & \text{caso contrário,} \end{cases} \tag{3.1}$$

onde $\tau \in \mathbb{R}$. Dolan e Moré [9] definem

$$\rho_s(\tau) = \frac{1}{n_p} \sum_{j \in \mathcal{P}} \mathcal{K}(r_{j,s}, \tau) \tag{3.2}$$

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

De forma geral, considerando uma medida de desempenho arbitrária, $\rho_s(\tau)$ é a porcentagem de problemas que o algoritmo s resolve em τ vezes o valor da medida de desempenho do algoritmo mais eficiente. Nos gráficos de desempenho, os valores do fator τ são indicados no eixo das abscissas, enquanto que no eixo das ordenadas são representados os valores das respectivas probabilidades $\rho_s(\tau)$. Os autores sugerem ainda o uso de uma escala logarítmica a fim de obter informações mais completas sobre o desempenho dos algoritmos.

Os gráficos apresentados a seguir foram construídos considerando a escala $\log_2$, o que significa que $\rho_s(\tau)$ foi calculado substituindo-se os valores de $r_{p,s}$ por $\log_2(r_{p,s})$ em (3.1) e (3.2). Com isso, $\rho_s(\tau)$ deve ser interpretado como a porcentagem de problemas que o algoritmo s resolve em 2^τ vezes o valor da medida de desempenho do melhor algoritmo. O valor máximo observado no eixo das abscissas é o parâmetro r_M, dado

pelo valor máximo entre $\log_2(r_{p,s})$ para todo p e s, tal que o problema p tenha sido resolvido pelo algoritmo s. Assim, temos que $\rho_s(r_M) = 1$ e $\lim_{\tau \to r_M^-} \rho_s$ é a proporção de problemas resolvidos pelo algoritmo s.

Nas Figuras 3.1-3.3, temos os gráficos de desempenho para as quatro variantes do Algoritmo 1.1. Para construir os gráficos eliminamos dois problemas (BIGGSC4 e HS54) cujos minimizadores encontrados pelos algoritmos de PQS e RI foram diferentes, assim consideramos um total de 298 problemas. Na legenda das figuras identificamos o algoritmo interno usado para o cálculo do passo e o critério de filtro empregado.

A medida de desempenho considerada no gráfico apresentado na Figura 3.1 é o número de iterações do algoritmo geral de filtro. Podemos notar que não existe diferença significativa entre os resultados de acordo com o critério de filtro considerado. Por outro lado, vemos que os algoritmos de RI são mais eficientes que os algoritmos de PQS, uma vez PQS-Original e PQS-Inclinado resolveram aproximadamente 45% dos problemas com o menor número de iterações, enquanto que para RI-Original e RI-Inclinado a porcentagem foi de 73%. Observamos que as quatro variantes do Algoritmo 1.1 possuem desempenhos similares ao considerarmos um fator τ próximo de 1, o que significa que cada uma delas resolveu em torno de 83% dos problemas usando não mais que o dobro do número de iterações do algoritmo mais eficiente. Além disso, para valores de τ menores que 1, os algoritmos de RI apresentaram resultados superiores aos algoritmos de PQS. Notamos ainda que os algoritmos de PQS resolveram $92,6\%$ dos 298 problemas considerados e os algoritmos de RI resolveram $83,9\%$ deles.

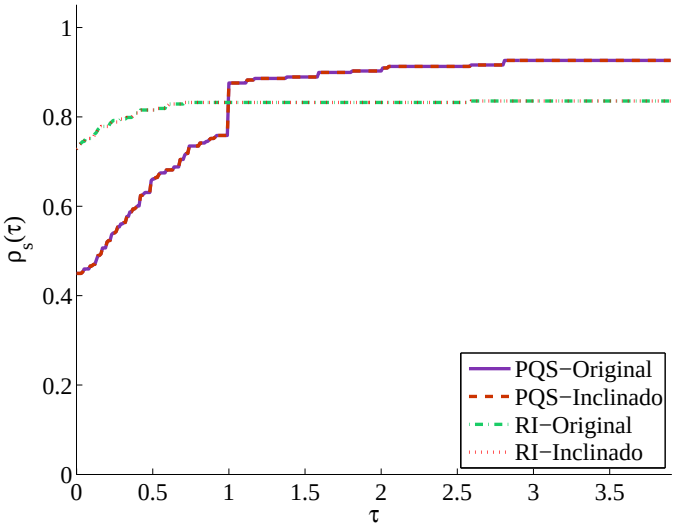

Figura 3.1: Gráfico de desempenho para o número de iterações na escala $\log_2$.

É importante ressaltar que o número de iterações não pode ser considerado como uma medida decisiva na escolha do algoritmo mais eficiente, uma vez que esta não leva em consideração o esforço computacional gerado pelos algoritmos internos usados nas fases de viabilidade e otimalidade dos algoritmos de PQS e RI. No entanto,

informações mais conclusivas podem ser obtidas se analisarmos o número de iterações em conjunto com outras medidas importantes como número de avaliação de funções e tempo de processamento dos algoritmos.

Para analisar o custo da avaliação de funções e derivadas, definimos uma medida que leva em consideração o número de avaliações da função objetivo ($\#f$), das restrições ($\#c$), do gradiente da função objetivo ($\#g$) e das jacobianas das restrições ($\#J$), dada por

$$nf = \#f + m\#c + 3\#g + 3m\#J,$$

onde $m = p + q$ é o número de restrições do problema. Escolhemos o peso 3 para o número de avaliações do gradiente da função objetivo e da jacobiana, pois de acordo com Griewank, Juedes e Utke [22], o custo da avaliação de derivadas é no máximo 5 vezes o custo da avaliação de funções .

Com base nessa medida de desempenho construímos o gráfico apresentado na Figura 3.2, no qual podemos notar que os algoritmos de PQS resolveram 58% dos problemas com o menor valor de nf e resolveram $92,6\%$ destes usando 2,71 vezes o número de avaliações de funções do melhor algoritmo, o que corresponde a um valor de τ de 1,44. Já os algoritmos de RI apresentaram o menor número de iterações para 36% dos problemas e para $\tau = 1,44$ resolveram aproximadamente 73% deles. Observamos ainda que RI-Original e RI-Inclinado atingem o valor máximo de ρ_s de $83,9\%$ para $\tau = 4,8$, ou seja, estes algoritmos resolvem um número máximo de 250 problemas usando aproximadamente 28 vezes o valor de nf do melhor algoritmo.

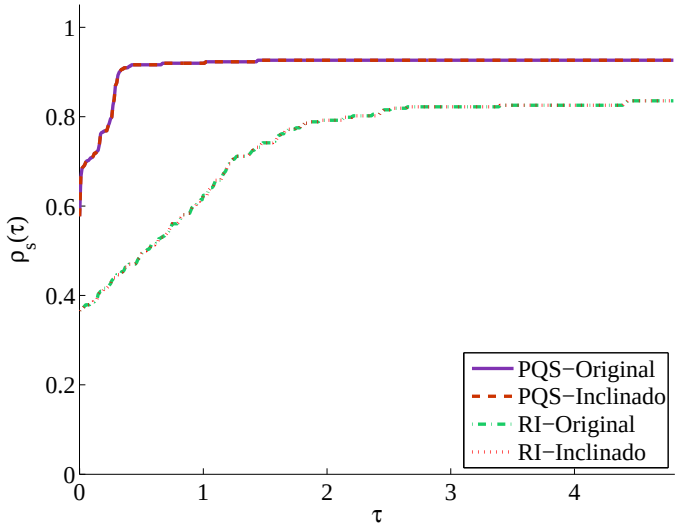

Figura 3.2: Gráfico de desempenho para avaliação de funções na escala $\log_2$.

O tempo de processamento é analisado no gráfico de desempenho da Figura 3.3. Nesta figura podemos observar uma diferença entre os critérios de filtro considerados. Os algoritmos PQS-Original e PQS-Inclinado resolveram 14% e 16% dos problemas com o menor tempo, respectivamente. O algoritmo RI-Inclinado resolveu

57% dos problemas no menor tempo, enquanto que RI-Original resolveu 17%. Os quatro algoritmos apresentam resultados próximos para valores de τ em torno de 1,2 resolvendo aproximadamente 81% dos problemas dentro desse fator.

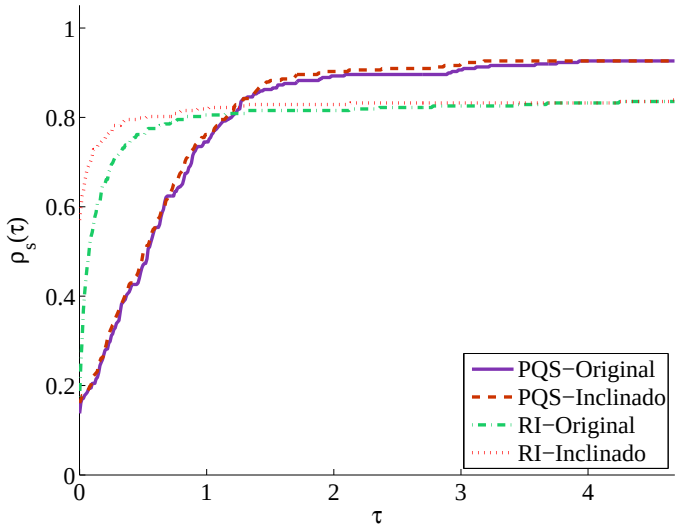

Figura 3.3: Gráfico de desempenho para o tempo de processamento na escala $\log_2$.

Uma outra questão a ser analisada aqui é o número de pares no filtro final dos quatro algoritmos. Como discutido anteriormente, em cada iteração do tipo h onde há atualização do filtro, fizemos uma limpeza deste por meio da remoção dos pares pertencentes a F_k que eram dominados pelo ponto corrente que foi incluído no filtro. Assim, mesmo que no decorrer das iterações haja um grande número de entradas no filtro, pode ser que no final da execução do algoritmo, este contenha uma quantidade pequena de pares.

No Apêndice B, podemos observar em detalhes como se deu o gerenciamento do filtro para as quatro variantes do Algoritmo 1.1, onde apresentamos o número de iterações do algoritmo geral, a quantidade delas que são do tipo h e o número de pares no filtro final. Por exemplo, o algoritmo PQS-Original resolveu o problema AIRPORT em 233 iterações sendo que em 112 delas houve atualização do filtro, mas no final da execução o filtro permaneceu com 78 pares. Dessa forma, no decorrer das iterações, 34 entradas do filtro foram descartadas.

A Tabela 3.3 resume os resultados observados para o número de pares no filtro final, considerando os 278 problemas resolvidos com sucesso pelos algoritmos PQS-Original e PQS-Inclinado.

Os 3 problemas que apresentaram mais de 20 pares no filtro, para ambos os algoritmos, foram AIRPORT, ELEC e MSS1, com 78, 74 e 81 pares, respectivamente.

Para os algoritmos RI-Original e RI-Inclinado, dos 252 problemas resolvidos com sucesso, 77, 78% deles terminaram com no máximo 2 pares no filtro e 20, 63% apresentaram de 3 a 10 pares. Apenas 4 problemas terminaram com mais de 10 pares

Número de pares	Número de problemas	
	PQS-Original	PQS-Inclinado
De 0 a 4	218	221
De 5 a 10	50	47
De 11 a 20	7	7
Mais de 20	3	3

Tabela 3.3: Quantidade de pares no filtro.

no filtro, sendo eles: BATCH (11 pares), ELEC (111 pares para RI-Original e 92 para RI-Inclinado), HS111 (14 pares) e TFI1 (16 pares).

Analisando os 249 problemas que foram resolvidos pelos quatro algoritmos, 202 apresentaram menos de 5 pares no filtro final quando o passo foi calculado por PQS e 231 apresentaram um número de pares nesse intervalo quando o passo foi calculado por RI, independentemente do critério de filtro usado. O número de problemas que tiveram de 5 a 10 elementos no filtro foi de 40 para PQS e de 14 para RI, restando apenas 7 e 4 problemas com mais de 10 pares, quando resolvidos por PQS e RI, respectivamente.

Dessa forma, podemos notar que, para os problemas selecionados, o número de pares remanescentes no filtro após a limpeza é baixo. Além disso, quando comparamos estes resultados com o número de iterações do tipo h (ver Apêndice B), que nos fornece o total de inclusões no filtro, vemos que a limpeza não apresenta vantagem na implementação, uma vez que para a maioria dos problemas, a diferença entre o número de pares no filtro final e a quantidade de iterações do tipo h é mínima.

3.3 Conclusões dos resultados numéricos

Com estes resultados numéricos podemos concluir que existe uma vantagem em calcular o passo pelo algoritmo baseado em PQS, uma vez que PQS-Original e PQS-Inclinado resolveram $92,67\%$ dos problemas corretamente e, além disso, dos 252 problemas resolvidos por RI-Original e RI-Inclinado, apenas 3 deles (CRESC4, HS61, S268) não foram resolvidos pelas demais variantes.

Apesar dos algoritmos de RI se mostrarem mais eficientes em relação ao número de iterações, estes apresentaram um custo maior no que se refere à avaliação de funções. Isto porque o custo de cada iteração do Algoritmo 1.1 com o passo calculado por RI é maior, devido ao algoritmo usado na fase de viabilidade. No entanto, apesar dos bons resultados para PQS, este algoritmo para o cálculo do passo requer a resolução de pelo menos dois sub-problemas quadráticos, um na fase de viabilidade e o(s) outro(s) na fase de otimalidade, o que também acarreta um alto custo computacional. Nos testes realizados notamos que, muitas vezes, a dificuldade na resolução dos problemas estava nessas fases. Isso foi um fator que limitou também nossa implementação, pois o comando `quadprog` apresenta restrições em relação ao número de variáveis do problema.

Finalizamos este capítulo concluindo que embora as regras de filtro original e inclinado proporcionem diferentes propriedades de convergência para o Algoritmo 1.1, numericamente não são observadas diferenças significativas nos resultados, pelo menos para o conjunto de problemas considerados neste trabalho.

Capítulo 4

Aplicação ao problema de confiabilidade estrutural

Neste capítulo apresentamos uma aplicação dos métodos de programação não linear, discutidos nos capítulos anteriores, a um problema de otimização que surge em Análise de Confiabilidade Estrutural, bem como uma breve descrição dessa metodologia.

Projetos de engenharia estrutural têm como propósito garantir um desempenho satisfatório do sistema, com segurança, funcionalidade e durabilidade, entre outros critérios. Porém, devido à presença de incertezas associadas às variáveis de projeto, existe um risco de que a estrutura não atenda a finalidade para a qual foi projetada. Para avaliar esse risco, denominado de probabilidade de falha, utiliza-se a metodologia Análise de Confiabilidade Estrutural, na qual 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*). Nesses 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 por Hasofer e Lind [26] e aperfeiçoado por Rackwitz e Fiessler [41], sendo este o algoritmo mais empregado na prática. Porém, na sua forma original, este algoritmo é instável, podendo não convergir em alguns casos, como veremos mais adiante. Isso tem impulsionado pesquisadores a proporem aperfeiçoamentos a esse algoritmo e novas metodologias para o cálculo da probabilidade de falha.

Liu e Kiureghian [29] avaliaram a aplicação dos métodos Gradiente Projetado,

Lagrangiano Aumentado e Programação Quadrática Sequencial no cálculo do ponto de projeto e compararam os resultados aos obtidos pelo HLRF e por uma melhoria proposta a esse algoritmo por meio da inclusão de uma função de mérito para induzir a convergência. No entanto, os autores deixam claro que essa melhoria não gera um algoritmo globalmente convergente, embora melhore os resultados do HLRF em alguns casos. Zhang e Kiureghian [54] propuseram uma melhoria ao algoritmo HLRF incluindo a regra de Armijo para selecionar o tamanho do passo, a qual provaram ser globalmente convergente. Santosh *et al.* [46] também apresentaram uma melhoria usando a mesma função de mérito de [29] e a regra de Armijo.

Santos e Matioli [45] propuseram a utilização do Método Duas Fases, desenvolvido por Luenberger [32], que combina os métodos de Penalização [53] e Gradiente Projetado [44] na determinação do ponto de projeto. Periçaro *et al.* [39] apresentaram uma comparação entre os algoritmos Duas Fases, HLRF e RI-Inclinado, que indicou um bom desempenho do algoritmo de filtro com Restauração Inexata quando aplicado ao problema de confiabilidade e isso nos motivou a verificar a aplicabilidade das demais variantes do nosso algoritmo geral de filtro nesse contexto.

A fim de mostrar essa possibilidade de aplicação prática dos algoritmos de filtro, fazemos inicialmente uma descrição do problema de otimização a ser solucionado em análise de confiabilidade estrutural, baseada nas referências [1, 25, 36], seguida por uma discussão sobre os algoritmos HLRF e iHLRF. Finalizamos o capítulo apresentando resultados de testes numéricos realizados considerando alguns problemas de confiabilidade disponíveis na literatura especializada.

4.1 Cálculo da probabilidade de falha

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,

$$\underline{X} = (X_1, X_2, \ldots, X_n)^T. \tag{4.1}$$

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 destas. Para tanto, é necessário estabelecer relações funcionais entre as variáveis básicas do sistema estrutural sob consideração. Matematicamente, esse relacionamento ou função de desempenho pode ser descrito como:

$$C(\underline{X}) = C(X_1, X_2, \ldots, X_n). \tag{4.2}$$

A superfície de falha ou equação de estado limite é definida como $C(\underline{X}) = 0$.

Essa superfície define o limite entre a região de segurança, $C(\underline{X}) > 0$, e a região de falha, $C(\underline{X}) < 0$. Dessa forma, a probabilidade de falha P_f é calculada por

$$P_f = \int \dots \int_{C(\underline{X})<0} f_{\underline{X}}(x_1, x_2, \dots, x_n)\, dx_1 dx_2 \cdots dx_n \tag{4.3}$$

onde $f_{\underline{X}}(x_1, x_2, \dots, x_n)$ é a função densidade de probabilidade conjunta das variáveis de projeto.

Resolver a integral múltipla dada em (4.3) é uma tarefa complexa, até mesmo porque nem sempre conhecemos a função densidade de probabilidade conjunta das variáveis aleatórias em questão. Por esse motivo, a probabilidade de falha é geralmente estimada por meio de métodos analíticos conhecidos como FORM e SORM, que consistem em aproximar a equação de estado limite por uma função linear e quadrática, respectivamente. O ponto sobre a equação de estado limite em que é feita a linearização ou aproximação quadrática é obtido por meio da resolução de um problema de otimização e a este ponto está relacionado um índice de confiabilidade que é fundamental na estimação de P_f. É neste momento, que surge a necessidade de aplicarmos um método de otimização à confiabilidade estrutural.

Para compreender o significado desse índice de confiabilidade, bem como sua relação com problemas de otimização, vamos considerar um problema bidimensional, em que a equação de estado limite é dada por

$$C(\underline{X}) = X_1 - X_2 = 0. \tag{4.4}$$

As variáveis X_1 e X_2 podem ser consideradas como a resistência e a solicitação impostas a uma estrutura, respectivamente. Assim, a falha ocorre quando $X_1 < X_2$, ou seja, quando $C(\underline{X}) < 0$.

Suponha que X_1 e X_2 sejam variáveis aleatórias independentes e normalmente distribuídas com médias μ_{X_1} e μ_{X_2}, e desvios padrão σ_{X_1} e σ_{X_2}, respectivamente. Então, $Z = C(\underline{X})$ também é uma variável aleatória normal com média $\mu_Z = \mu_{X_1} - \mu_{X_2}$ e desvio padrão $\sigma_Z = \sqrt{\sigma_{X_1}^2 + \sigma_{X_2}^2}$. Neste caso, a probabilidade de falha é dada por

$$P_f = P(Z < 0). \tag{4.5}$$

Seja Y a variável normal padrão, ou seja, variável normal com média 0 e desvio padrão 1, dada por

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

Temos que

$$P_f = P(Z < 0) = P\left(Y < \frac{0 - \mu_Z}{\sigma_Z}\right) = \Phi\left(-\frac{\mu_{X_1} - \mu_{X_2}}{\sqrt{\sigma_{X_1}^2 + \sigma_{X_2}^2}}\right),$$

onde Φ é a função de distribuição acumulada da variável normal padrão, definida como

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

Vemos que P_f depende da razão entre a média e o desvio padrão de Z, definida como índice de confiabilidade, que denotamos por

$$\beta = \frac{\mu_{X_1} - \mu_{X_2}}{\sqrt{\sigma_{X_1}^2 + \sigma_{X_2}^2}}. \tag{4.6}$$

As ideias apresentadas até aqui podem ser estendidas facilmente para um problema n-dimensional, em que a função de desempenho é dada por (4.2). No caso em que essa função é não linear, a média e o desvio padrão de $C(\underline{X})$, necessários para o cálculo de β, podem ser aproximados por meio da expansão de $C(\underline{X})$ em uma série de Taylor de primeira ordem, centrada em $\mu = (\mu_{X_1}, \mu_{X_2}, \ldots, \mu_{X_n})^T$, obtendo assim a equação linear $\tilde{C}(\underline{X}) = 0$. Assim, o índice de confiabilidade é dado por

$$\beta = \frac{E[\tilde{C}(\underline{X})]}{\sqrt{V[\tilde{C}(\underline{X})]}}, \tag{4.7}$$

em que $E[\tilde{C}(\underline{X})]$ e $V[\tilde{C}(\underline{X})]$ são aproximações de primeira ordem para média e variância de $C(\underline{X})$. Dessa forma, uma aproximação de primeira ordem para a probabilidade de falha pode ser obtida da seguinte forma

$$P_f \approx \Phi(-\beta), \tag{4.8}$$

onde β é dado em (4.7).

O método descrito anteriormente, que consiste em aproximar a função estado limite por sua linearização nos valores médios das variáveis aleatórias X_i, $i = 1, \ldots, n$, é denominado *FOSM* (*First Order Second Moment*), pois além de linearizar $C(\underline{X})$, leva em consideração apenas momentos de até segunda ordem, que são a média e o desvio padrão das variáveis de projeto. Porém, este método possui um inconveniente, a probabilidade de falha dada por (4.8) é exata apenas quando as variáveis de projeto são estatisticamente independentes, normalmente distribuídas e $C(\underline{X})$ é linear, ou ainda, quando as variáveis são independentes, seguem a distribuição de probabilidade lognormal e $C(\underline{X})$ é uma função multiplicativa das variáveis X_i, como mostrado em [25]. No

entanto, apesar da limitada aplicabilidade desse método, o *FOSM* é considerado a base teórica para vários outros métodos de confiabilidade, como o *FORM*, que descrevemos adiante.

Em 1974, Hasofer e Lind [26] propuseram o método *AFOSM* (*Advanced First Order Second Moment*) para contornar os inconvenientes do *FOSM*, sendo aplicável a problemas cujas variáveis são independentes e normalmente distribuídas. Inicialmente, as variáveis originais X_i são padronizadas, obtendo assim as variáveis Y_i dadas por

$$Y_i = \frac{X_i - \mu_{X_i}}{\sigma_{X_i}}. \tag{4.9}$$

Com essa transformação obtemos a equação de estado limite $c(\underline{Y}) = 0$ associada às variáveis aleatórias padronizadas ou reduzidas Y_i. Em confiabilidade estrutural, o sistema de coordenadas originais é comumente denominado "espaço original" e o sistema de coordenadas reduzidas é denominado "espaço normal padrão".

No que segue, usaremos a seguinte notação: x_i e y_i representam valores específicos que as variáveis aleatórias X_i e Y_i podem assumir, para $i = 1, 2, \ldots, n$. Assim, denotamos por $\mathbf{x}$ o vetor cujas componentes são $x_1, x_2, \ldots, x_n$ e, da mesma forma, $\mathbf{y} = (y_1, y_2, \ldots, y_n)^T$.

Considere novamente a equação (4.4). Por meio da padronização das variáveis originais, obtemos a seguinte equação de estado limite

$$c(\underline{Y}) = \sigma_{X_1} Y_1 - \sigma_{X_2} Y_2 + \mu_{X_1} - \mu_{X_2} = 0. \tag{4.10}$$

Na Figura 4.1 ilustramos a transformação de Hasofer e Lind. À esquerda temos a equação de estado limite (4.4) e algumas curvas de nível da função densidade de probabilidade conjunta das variáveis originais (normal bivariada), dada por

$$f_{\underline{X}}(\mathbf{x}) = \frac{1}{2\pi\sigma_{X_1}\sigma_{X_2}} e^{\left\{-\frac{1}{2}\left[\left(\frac{x_1-\mu_{X_1}}{\sigma_{X_1}}\right)^2+\left(\frac{x_2-\mu_{X_2}}{\sigma_{X_2}}\right)^2\right]\right\}}. \tag{4.11}$$

À direita ilustramos a equação de estado limite padronizada (4.10) e algumas curvas de nível da função densidade de probabilidade normal padrão bivariada

$$f_{\underline{Y}}(\mathbf{y}) = \frac{1}{2\pi} e^{-\frac{1}{2}\left(y_1^2+y_2^2\right)}. \tag{4.12}$$

Nesta figura podemos notar que quanto maior a distância entre a equação de estado limite e a origem do sistema de coordenadas Y_1Y_2, menor será a região de falha.

Assim, Hasofer e Lind [26] definiram o índice de confiabilidade β_{HL} como sendo a mínima distância entre a superfície de falha e a origem do sistema de coordenadas reduzidas, dado por

$$\beta_{HL} = \sqrt{\mathbf{y}^{*T}\mathbf{y}^*}, \tag{4.13}$$

em que $\mathbf{y}^*$ é o ponto sobre a superfície de falha mais próximo da origem, denominado ponto de projeto. Podemos notar que $\mathbf{y}^*$ é o ponto mais provável de falha, uma vez que à medida que o ponto se distancia da origem, o valor da função normal padrão (4.12) nesse ponto diminui. Ainda na Figura 4.1 denotamos o ponto de projeto no espaço original por $\mathbf{x}^*$. Este ponto representa a pior combinação das variáveis de projeto, pois é o ponto sobre a região de falha com maior probabilidade de ocorrência.

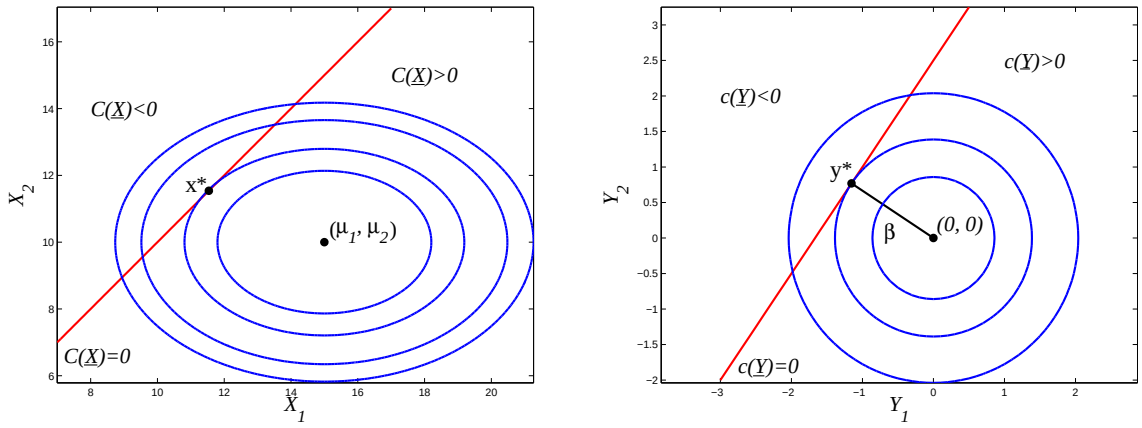

Figura 4.1: Transformação de Hasofer e Lind.

Dessa forma, usando uma simples fórmula da geometria, podemos calcular o índice de confiabilidade determinando a distância entre a reta (4.10) e a origem do sistema de coordenadas reduzidas, obtendo assim

$$\beta_{HL} = \frac{\mu_{X_1} - \mu_{X_2}}{\sqrt{\sigma_{X_1}^2 + \sigma_{X_2}^2}}, \tag{4.14}$$

que é idêntico àquele índice dado em (4.6), embora tenham sido determinados de maneiras diferentes. Isso significa que os métodos *FOSM* e *AFOSM* são equivalentes quando as variáveis são normais e a equação de estado limite é linear. Porém, na prática, as superfícies de falha costumam ser não lineares e neste caso, o cálculo do índice de confiabilidade torna-se um problema de otimização da forma

$$\begin{aligned} &\text{minimizar} && \frac{1}{2}\mathbf{y}^T\mathbf{y} \\ &\text{sujeito a} && c(\mathbf{y}) = 0 \end{aligned} \tag{4.15}$$

onde assumimos que a função $c : \mathbb{R}^n \to \mathbb{R}$ é continuamente diferenciável e cada ponto do conjunto $\{\mathbf{y} \in \mathbb{R}^n \mid c(\mathbf{y}) = 0\}$ cumpre a condição de qualificação de Mangasarian-Fromovitz.

Vamos mostrar agora que mesmo para o caso em que a equação de estado limite é não linear, uma aproximação para a probabilidade de falha pode ser obtida resolvendo-se o problema (4.15). Para tanto, primeiramente precisamos discutir a existência de solução para tal problema, o que será estabelecido no lema a seguir.

Lema 4.1 *O problema dado em* (4.15) *admite um minimizador global* $\mathbf{y}^*$.

Demonstração. Considere o conjunto $S = \{\mathbf{y} \in \mathbb{R}^n \mid c(\mathbf{y}) = 0\}$ e $\alpha = \inf\{\|\mathbf{y}\| \mid \mathbf{y} \in S\}$. Então, para todo $k \in \mathbb{N}$ existe $\mathbf{y}^k \in S$ tal que

$$\alpha \leq \left\|\mathbf{y}^k\right\| \leq \alpha + \frac{1}{k}. \tag{4.16}$$

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

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

Por outro lado, de (4.16), temos que $\left\|\mathbf{y}^k\right\| \to \alpha$, donde segue que $\|\mathbf{y}^*\| = \alpha$. Dessa forma, temos que

$$\|\mathbf{y}^*\| \leq \|\mathbf{y}\|$$

para todo $\mathbf{y} \in S$, completando a prova. □

Seja então $\mathbf{y}^*$ uma solução de (4.15), ou seja, $\mathbf{y}^*$ é um ponto de projeto. Pela regularidade da restrição c, temos que $\nabla c(\mathbf{y}^*) \neq 0$ e, portanto, existe $\lambda^* \in \mathbb{R}$ tal que as condições de KKT são satisfeitas, isto é, $c(\mathbf{y}^*) = 0$ e

$$\mathbf{y}^* = -\lambda^* \nabla c(\mathbf{y}^*). \tag{4.17}$$

Multiplicando ambos os membros de (4.17) por $\nabla c(\mathbf{y}^*)^T$, obtemos

$$\lambda^* = -\frac{\nabla c(\mathbf{y}^*)^T \mathbf{y}^*}{\|\nabla c(\mathbf{y}^*)\|^2}. \tag{4.18}$$

Assim, de (4.17) e (4.18) obtemos

$$\|\mathbf{y}^*\| = \pm\frac{\nabla c(\mathbf{y}^*)^T \mathbf{y}^*}{\|\nabla c(\mathbf{y}^*)\|}. \tag{4.19}$$

Note que $\nabla c(\mathbf{y}^*)^T \mathbf{y}^* < 0$, pois como a origem do sistema de coordenadas reduzidas pertence à região de segurança, uma vez que é obtida pela padronização dos valores médios das variáveis originais, os vetores $\nabla c(\mathbf{y}^*)$ e $\mathbf{y}^*$ possuem sinais opostos (Figura 4.2). Dessa forma, concluímos que a distância mínima da equação de estado limite $c(\mathbf{y}) = 0$ até a origem do espaço normal padrão é

$$\|\mathbf{y}^*\| = -\frac{\nabla c(\mathbf{y}^*)^T \mathbf{y}^*}{\|\nabla c(\mathbf{y}^*)\|}. \tag{4.20}$$

Uma vez calculado o ponto de projeto no espaço normal padrão, podemos obter as componentes do vetor $\mathbf{x}^*$, que representa o ponto de projeto no espaço original,

fazendo

$$x_i^* = \sigma_{X_i} y_i^* + \mu_{X_i}. \tag{4.21}$$

Vimos anteriormente que, para equações de estado limite não lineares, o método *FOSM* determina uma aproximação para a probabilidade de falha por meio da linearização de $C(\underline{X})$ nos valores médios das variáveis originais. Vamos mostrar agora que se a linearização for centrada no ponto de projeto $\mathbf{x}^*$, então o índice de confiabilidade β dado em (4.7) é equivalente ao índice de Hasofer e Lind, ou seja, equivale à mínima distância entre a origem do espaço normal padrão e a superfície de falha, dada em (4.20).

A expansão de $C(\underline{X})$ em série de Taylor de primeira ordem em torno de $\mathbf{x}^*$ é dada por

$$\tilde{C}(\underline{X}) = C(\mathbf{x}^*) + \nabla C(\mathbf{x}^*)^T(\underline{X} - \mathbf{x}^*), \tag{4.22}$$

ou, equivalentemente,

$$\tilde{C}(\underline{X}) = \sum_{i=1}^{n} (X_i - x_i^*) \left.\frac{\partial C}{\partial X_i}\right|_{\underline{X}=\mathbf{x}^*},$$

pois $C(\mathbf{x}^*) = 0$. De (4.9) e (4.21) temos que

$$X_i - x_i^* = \sigma_{X_i}(Y_i - y_i^*) \qquad \text{e} \qquad \left.\frac{\partial C}{\partial X_i}\right|_{\underline{X}=\mathbf{x}^*} = \frac{1}{\sigma_{X_i}} \cdot \left.\frac{\partial c}{\partial Y_i}\right|_{\underline{Y}=\mathbf{y}^*}.$$

Assim,

$$\tilde{C}(\underline{X}) = \nabla c(\mathbf{y}^*)^T(\underline{Y} - \mathbf{y}^*) = \tilde{c}(\underline{Y}), \tag{4.23}$$

onde $\tilde{c}(\underline{Y})$ é a aproximação linear de $c(\underline{Y})$. Na Figura 4.2 ilustramos, para o caso bidimensional, a equação de estado limite no espaço normal padrão e sua linearização no ponto de projeto $\mathbf{y}^*$, representada por $\tilde{c}(\underline{Y})$.

Usando (4.23) e lembrando que estamos assumindo que as variáveis são independentes, obtemos as seguintes aproximações de primeira ordem para média e variância de $C(\underline{X})$, respectivamente

$$E[\tilde{C}(\underline{X})] = E[\nabla c(\mathbf{y}^*)^T(\underline{Y} - \mathbf{y}^*)] = -\nabla c(\mathbf{y}^*)^T\mathbf{y}^*$$

e

$$V[\tilde{C}(\underline{X})] = V[\nabla c(\mathbf{y}^*)^T(\underline{Y} - \mathbf{y}^*)] = \nabla c(\mathbf{y}^*)^T\nabla c(\mathbf{y}^*) = \|\nabla c(\mathbf{y}^*)\|^2.$$

Portanto, substituindo esses resultados em (4.7), obtemos

$$\beta = -\frac{\nabla c(\mathbf{y}^*)^T\mathbf{y}^*}{\|\nabla c(\mathbf{y}^*)\|}. \tag{4.24}$$

Observe que o índice de confiabilidade dado em (4.24) corresponde à mínima distância entre a equação de estado limite e a origem do sistema de coordenadas re-

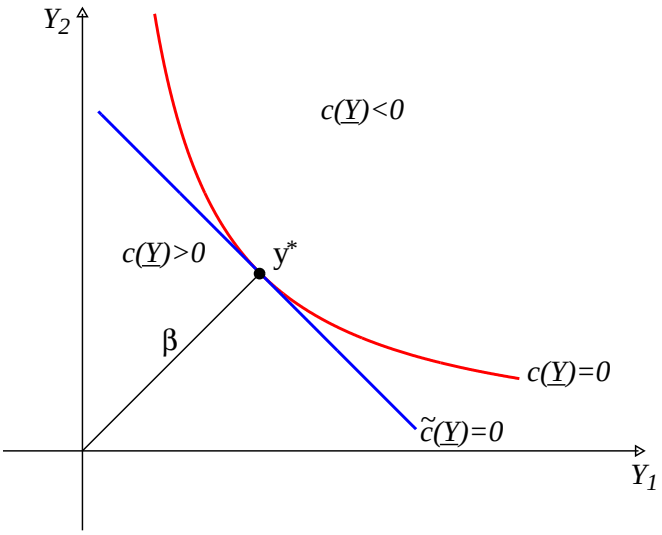

Figura 4.2: Equação de estado limite não linear.

duzidas dada em (4.20). Porém, é importante notar que essa equivalência se deve ao fato de que no método *AFOSM* a linearização de $C(\underline{X})$ foi feita no ponto de projeto. Por outro lado, no método *FOSM* o ponto considerado na linearização é a média das variáveis aleatórias, o que pode levar a significativos erros de aproximação, uma vez que este ponto não está necessariamente sobre a superfície de falha. Como o ponto de projeto é o ponto mais provável de falha, erros na aproximação de primeira ordem para P_f podem ser minimizados linearizando-se a equação de estado limite nesse ponto.

É importante frisar que o índice de confiabilidade (4.24) fornece a solução exata da probabilidade de falha apenas nos casos em que as variáveis de projeto são independentes e normalmente distribuídas e a superfície de falha é linear. Em qualquer outro caso, a solução de (4.15) fornece apenas uma aproximação para a probabilidade de falha. No entanto, as ideias dos métodos *FOSM* e *AFOSM* podem ser estendidas para problemas cujas variáveis são não normais e/ou correlacionadas.

Uma generalização dos métodos apresentados anteriomente definem um método conhecido como *FORM* (*First Order Reliability Method*) que, assim como o *AFOSM*, consiste em substituir a superfície de falha por sua linearização no ponto de projeto, levando em consideração todas as informações estatísticas relacionadas às variáveis originais. Embora o método *FORM* seja o mais empregado na prática, existe também o método *SORM* (*Second Order Reliability Method*), cuja diferença reside no fato de que a superfície de falha é aproximada por uma função quadrática, considerando assim a curvatura da estado limite nas proximidades do ponto de projeto, o que fornece melhores aproximações para a probabilidade de falha. Como o nosso objetivo nesse capítulo é empregar algoritmos de filtro para obter o ponto de projeto, que é útil em ambos os métodos, vamos nos limitar apenas ao método *FORM* e aos problemas que envolvem variáveis independentes, podendo ser não normais.

Variáveis aleatórias não normais são comuns nos problemas de engenharia,

sendo necessário realizar uma transformação das variáveis em normais equivalentes. Existem várias formas de efetuar essa transformação, uma delas utiliza o princípio da transformação normal, de Ditlevsen [8], que consiste em, para um dado ponto $\mathbf{x}$, estimar os parâmetros da distribuição normal equivalente para cada uma das variáveis, impondo que as funções de distribuição acumulada e densidade de probabilidade da variável original e da normal equivalente devem se igualar no ponto $\mathbf{x}$. Tal condição fornece os seguintes parâmetros

$$\mu_{X_i}^N = x_i - \Phi^{-1}[F_{X_i}(x_i)]\sigma_{X_i}^N \qquad \text{e} \qquad \sigma_{X_i}^N = \frac{\phi\left\{\Phi^{-1}[F_{X_i}(x_i)]\right\}}{f_{X_i}(x_i)},\ i = 1, \ldots, n \tag{4.25}$$

onde ϕ e Φ representam a função densidade de probabilidade e a função de distribuição acumulada da variável normal padrão, respectivamente, e f_{X_i} e F_{X_i}, denotam essas mesmas funções para a variável aleatória X_i.

Uma vez determinados os parâmetros da normal equivalente, a padronização de $\mathbf{x}$ é feita da seguinte forma

$$y_i = \frac{x_i - \mu_{X_i}^N}{\sigma_{X_i}^N}, i = 1, 2, \ldots, n. \tag{4.26}$$

Observe que os parâmetros dados em (4.25) são válidos apenas para o ponto $\mathbf{x}$ e, dessa forma, a cada novo ponto obtido pelo algoritmo usado para determinar o ponto de projeto, os parâmetros da normal equivalente devem ser recalculados.

As ideias do método *FORM* podem ser resumidas no seguinte algoritmo que pode ser empregado na busca pelo ponto de projeto de problemas de confiabilidade que envolvem variáveis aleatórias independentes, que seguem qualquer distribuição de probabilidade.

Algoritmo 4.1 *FORM*

Dado $\mathbf{x}^0 \in \mathbb{R}^n$ (geralmente o vetor de médias)
$k = 0$
REPITA (enquanto o critério de parada não for satisfeito)

1. Determinar os parâmetros da normal equivalente no ponto $\mathbf{x}^k$ usando (4.25).
2. Obter as coordenadas do ponto $\mathbf{y}^k$.
3. Calcular $\mathbf{y}^{k+1}$.
4. Voltar para o espaço original transformando $\mathbf{y}^{k+1}$ em $\mathbf{x}^{k+1}$.
5. Calcular o índice de confiabilidade: $\beta = \left\|\mathbf{y}^{k+1}\right\|$.

$k = k + 1$.

Para facilitar a implementação, a transformação das variáveis do espaço original para o espaço padronizado pode ser feita de forma matricial. Para tanto, conside-

ramos as matrizes de transformação

$$J_{\mathbf{xy}} = \text{diag}(\sigma_{X_1}^N, \ldots, \sigma_{X_n}^N), \qquad J_{\mathbf{yx}} = J_{\mathbf{xy}}^{-1} \qquad \text{e} \qquad \mathbf{m} = (\mu_{X_1}^N, \ldots, \mu_{X_n}^N)^T.$$

Dessa forma, as transformações necessárias nos passos *2* e *4* podem ser feitas, respectivamente, da seguinte forma

$$\mathbf{y} = J_{\mathbf{yx}} \cdot (\mathbf{x} - \mathbf{m}) \tag{4.27}$$

e

$$\mathbf{x} = J_{\mathbf{xy}} \cdot \mathbf{y} + \mathbf{m}. \tag{4.28}$$

Para problemas que envolvem variáveis correlacionadas, uma transformação extra é realizada a fim de obter variáveis independentes.

Observe que a função $C(\underline{X})$ não precisa ser reescrita como $c(\underline{Y})$. Para avaliar c em um ponto $\mathbf{y}^k$, basta obter $\mathbf{x}^k$ por meio de (4.28) e fazer $c(\mathbf{y}^k) = C(\mathbf{x}^k)$. Da mesma forma, $\nabla c(\mathbf{y}^k)$ pode ser determinado a partir das derivadas de C em $\mathbf{x}^k$ e $J_{\mathbf{xy}}$

$$\nabla c(\mathbf{y}^k) = J_{\mathbf{xy}} \cdot \nabla C(\mathbf{x}^k).$$

O critério de parada frequentemente adotado para o Algoritmo 4.1 é

$$1 - \frac{|\nabla c(\mathbf{y}^k)^T \mathbf{y}^k|}{\|\nabla c(\mathbf{y}^k)\| \, \|\mathbf{y}^k\|} < \varepsilon \qquad \text{e} \qquad |c(\mathbf{y}^k)| < \varepsilon. \tag{4.29}$$

Em nossa implementação escolhemos $\varepsilon = 10^{-4}$.

Vamos nos ater agora ao passo *3* do Algoritmo 4.1, que consiste em determinar o ponto de projeto y^{k+1}, resolvendo-se o problema (4.15). O algoritmo comumente usado nessa etapa é o algoritmo HLRF, porém qualquer outro algoritmo de otimização pode ser empregado. Na próxima seção vamos descrever os algoritmos HLRF e o iHLRF. Em seguida, discutimos a aplicação do nosso algoritmo geral de filtro nesse passo.

4.1.1 Cálculo do ponto de projeto

Vamos iniciar a discussão sobre o cálculo de $\mathbf{y}^{k+1}$ descrevendo o algoritmo HLRF que foi proposto inicialmente por Hasofer e Lind [26] e posteriormente aperfeiçoado por Rackwitz e Fiessler [41], que incluíram o tratamento de variáveis não normais ao algoritmo original. O algoritmo HLRF está baseado no método de Newton e consiste em substituir, a cada iteração, a equação de estado limite por sua linearização no ponto corrente, tomando como próximo iterando o ponto sobre a superfície de falha linearizada que está mais próximo da origem. Seguindo essas ideias podemos facilmente obter uma fórmula de recorrência para obter $\mathbf{y}^{k+1}$.

Considere a aproximação linear de $c(\mathbf{y})$ no ponto corrente, dada por

$$\tilde{c}(\mathbf{y}) = c(\mathbf{y}^k) + \nabla c(\mathbf{y}^k)^T(\mathbf{y} - \mathbf{y}^k). \tag{4.30}$$

Vamos substituir agora a equação de estado limite $c(\mathbf{y}) = 0$ por $\tilde{c}(\mathbf{y}) = 0$, ou seja,

$$c(\mathbf{y}^k) + \nabla c(\mathbf{y}^k)^T(\mathbf{y} - \mathbf{y}^k) = 0. \tag{4.31}$$

O ponto sobre o hiperplano dado por (4.31) que está mais próximo da origem é o ponto de intersecção com a reta perpendicular a ele, dada por

$$\mathbf{y} = s\nabla c(\mathbf{y}^k), \tag{4.32}$$

onde $s \in \mathbb{R}$. Substituindo (4.32) em (4.31), obtemos

$$s = \frac{\nabla c(\mathbf{y}^k)^T\mathbf{y}^k - c(\mathbf{y}^k)}{\|\nabla c(\mathbf{y}^k)\|^2}.$$

Dessa forma, a fórmula recursiva para determinar $\mathbf{y}^{k+1}$ pelo algoritmo HLRF é

$$\mathbf{y}^{k+1} = \frac{[\nabla c(\mathbf{y}^k)^T\mathbf{y}^k - c(\mathbf{y}^k)]\nabla c(\mathbf{y}^k)}{\|\nabla c(\mathbf{y}^k)\|^2}. \tag{4.33}$$

Vemos que existe uma semelhança entre o algoritmo HLRF e o método de PQS, uma vez que ambos consistem em simplificar o problema (4.15) substituindo a restrição por uma aproximação linear. O método de PQS minimiza, a cada iteração, um modelo quadrático da função objetivo sujeito à linearização da restrição do problema original. Assim, o problema (4.15) é substituído por

$$\begin{array}{ll} \text{minimizar} & \dfrac{1}{2}\mathbf{d}^T B_k \mathbf{d} + \mathbf{y}^{kT}\mathbf{d} \\ \text{sujeito a} & \nabla c(\mathbf{y}^k)^T\mathbf{d} + c(\mathbf{y}^k) = 0, \end{array} \tag{4.34}$$

em que B_k é uma aproximação da Hessiana do Lagrangiano associado ao problema (4.15). Sendo $\mathbf{d}^k$ uma solução de (4.34), o próximo iterando obtido pelo método de PQS é

$$\mathbf{y}^{k+1} = \mathbf{y}^k + \mathbf{d}^k. \tag{4.35}$$

Na verdade, o algoritmo HLRF é um caso particular de PQS em que B_k é aproximada pela matriz identidade. De fato, uma solução primal-dual $(\mathbf{d}^k, \xi^k)$ do subproblema quadrático (4.34), com $B_k = I_n$, deve satisfazer

$$\left\{ \begin{array}{rcl} \mathbf{d}^k + \mathbf{y}^k + \nabla c(\mathbf{y}^k)\xi^k & = & 0 \\ \nabla c(\mathbf{y}^k)^T\mathbf{d}^k + c(\mathbf{y}^k) & = & 0. \end{array} \right. \tag{4.36}$$

Resolvendo as equações de (4.36) simultaneamente, obtemos

$$\mathbf{d}^k = \frac{[\nabla c(\mathbf{y}^k)^T \mathbf{y}^k - c(\mathbf{y}^k)]\nabla c(\mathbf{y}^k)}{\|\nabla c(\mathbf{y}^k)\|^2} - \mathbf{y}^k \qquad \text{e} \qquad \xi^k = \frac{c(\mathbf{y}^k) - \nabla c(\mathbf{y}^k)^T \mathbf{y}^k}{\|\nabla c(\mathbf{y}^k)\|^2}.$$

Substituindo $\mathbf{d}^k$ em (4.35) obtemos exatamente (4.33), provando assim a equivalência entre PQS com $B_k = I_n$ e HLRF. Na Figura 4.3 ilustramos uma iteração do algoritmo HLRF.

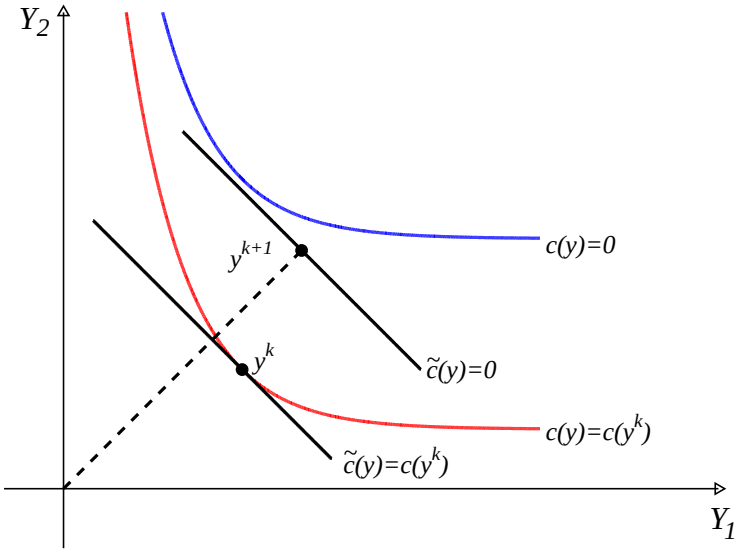

Figura 4.3: Uma iteração do algoritmo HLRF.

Ressaltamos que o algoritmo HLRF foi desenvolvido especificamente para ser aplicado ao problema (4.15) e devido à sua simplicidade e ao baixo custo computacional, tem sido amplamente utilizado em engenharia. No entanto, este algoritmo é instável, podendo não convergir em alguns casos, principalmente quando aplicado a problemas em que a superfície de falha é altamente não linear. Isto se deve ao fato de que nenhuma estratégia de globalização, como busca linear ou região de confiança, foi incluída no algoritmo original.

De acordo com Zhang e Kiureghian [54], para contornar o problema de convergência do algoritmo HLRF, vários pesquisadores propuseram melhorias a este, sem apresentar uma prova matemática formal de convergência global para os algoritmos propostos. Nesse trabalho, os autores apresentam uma forma de garantir a convergência empregando uma estratégia de busca linear com uma função de mérito que é decrescente na direção de busca

$$\mathbf{d}^k = \frac{[\nabla c(\mathbf{y}^k)^T \mathbf{y}^k - c(\mathbf{y}^k)]\nabla c(\mathbf{y}^k)}{\|\nabla c(\mathbf{y}^k)\|^2} - \mathbf{y}^k. \tag{4.37}$$

A melhoria proposta deu origem ao algoritmo iHLRF que consiste em determinar um tamanho de passo ótimo t_k a ser dado na direção (4.37). Assim, o próximo iterando é

dado por

$$\mathbf{y}^{k+1} = \mathbf{y}^k + t_k \mathbf{d}^k. \tag{4.38}$$

Observe que para $t_k = 1$, obtemos a fórmula recursiva do HLRF.

A fim de determinar t_k, a seguinte função de mérito é introduzida

$$m(\mathbf{y}) = \frac{1}{2}\mathbf{y}^T\mathbf{y} + \rho|c(\mathbf{y})|, \tag{4.39}$$

onde ρ é um parâmetro de penalidade. Zhang e Kiureghian [54] mostraram que se

$$\rho > \frac{\left\|\mathbf{y}^k\right\|}{\|\nabla c(\mathbf{y}^k)\|},$$

a direção de busca (4.37) é uma direção de descida para $m(\mathbf{y})$. Assim, a cada iteração t_k é escolhido de modo a satisfazer a condição de Armijo

$$m(\mathbf{y}^k + t_k\mathbf{d}^k) - m(\mathbf{y}^k) \leq \eta t_k \nabla m(\mathbf{y}^k)^T \mathbf{d}^k, \tag{4.40}$$

onde $\nabla m(\mathbf{y}^k) = \mathbf{y}^k + \rho \cdot \text{sign}(c(\mathbf{y}^k))\nabla c(\mathbf{y}^k)$.

Embora os autores garantam a convergência global do algoritmo, acreditamos que um estudo mais aprofundado sobre as propriedades de convergência deste ainda possa ser realizado.

Vamos apresentar agora o Algoritmo iHLRF, proposto em [54]. Não vamos descrever um algoritmo específico para o HLRF, pois este pode ser obtido apenas com a inclusão da fórmula recursiva (4.33) no passo *3* do Algoritmo 4.1.

Algoritmo 4.2 iHLRF

Dados: $\mathbf{y}^k \in \mathbb{R}^n$, $\gamma, \eta \in (0,1)$ e $\delta > 1$

1. Calcule a direção de busca $\mathbf{d}^k$ dada em (4.37).

2. Determine o parâmetro de penalidade ρ.

SE $|c(\mathbf{y}^k)| \geq \varepsilon_1$

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

SENÃO

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

3. Determinar t_k (busca linear).

Faça $t = 1$

ENQUANTO $m(\mathbf{y}^k + t\mathbf{d}^k) - m(\mathbf{y}^k) > \eta t \nabla m(\mathbf{y}^k)^T \mathbf{d}^k$

$t = \gamma t$

$t_k = t$

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

Em nossa implementação usamos os seguintes valores para as constantes do Algoritmo 4.2: $\gamma = 0,5$, $\eta = 0,5$, $\delta = 2$ e $\varepsilon_1 = 0,005|c(\mathbf{y}^0)|$.

É importante observar que qualquer algoritmo de otimização pode ser inserido no passo *3* do Algoritmo *FORM*. Quando o problema envolve apenas variáveis normais, o passo *1* é desnecessário, e neste caso, basta fazer a padronização das variáveis originais e chamar um algoritmo interno para o cálculo de $\mathbf{y}^{k+1}$. Neste caso, quando acoplamos uma das variantes do Algoritmo 1.1 no passo *3*, esta será executada apenas uma vez, retornando assim o ponto de projeto no espaço padronizado. Já os algoritmos HLRF e iHLRF dão apenas um passo em cada iteração do *FORM*, e nesse caso são executados até que o critério de parada (4.29) seja satisfeito.

Para problemas gerais que envolvem variáveis não normais, a interpretação do Algoritmo *FORM* é um pouco diferente. A cada iteração os parâmetros da normal equivalente são recalculados, pois $\mu_{X_i}^N$ e $\sigma_{X_i}^N$ assumem valores distintos em cada novo ponto $\mathbf{x}^k$. Assim, uma nova equação de estado limite no espaço padronizado é obtida, uma vez que essa depende dos parâmetros da normal equivalente. Mesmo neste caso, em que as variáveis podem seguir uma distribuição qualquer, os algoritmos HLRF e iHLRF dão apenas um passo em cada iteração do Algoritmo 4.1. Por outro lado, ao considerar um outro algoritmo de otimização, como o nosso algoritmo geral de filtro, este pode fornecer a cada iteração do *FORM* o ponto sobre a equação de estado limite em consideração com norma mínima. A convergência para o ponto de projeto é alcançada quando dois pontos de projeto consecutivos, $\mathbf{y}^k$ e $\mathbf{y}^{k+1}$, tornam-se suficientemente próximos.

A seguir vamos analisar a aplicabilidade dos algoritmos RI-Original, RI-Inclinado, PQS-Original e PQS-Inclinado ao problema de confiabilidade e compará-los aos algoritmos HLRF e iHLRF, por meio de testes numéricos realizados com problemas selecionados da literatura especializada.

4.2 Testes numéricos

A fim de comparar as quatro variantes do nosso algoritmo geral de filtro com os algoritmos HLRF e iHLRF, quando empregados no passo *3* do Algoritmo 4.1, selecionamos 20 problemas disponíveis na literatura que são frequentemente usados para avaliar o desempenho de novos métodos para o cálculo da probabilidade de falha, os quais descrevemos a seguir apresentando a função de desempenho no espaço original e as distribuições de probabilidade das variáveis de projeto.

Como citado anteriormente, todos os problemas selecionados envolvem variáveis estatisticamente independentes. Nos problemas de 1 a 6 todas as variáveis consideradas seguem a distribuição normal padrão. Por esse motivo, para esses problemas, vamos apresentar apenas a função de desempenho. Nos demais problemas, apresentamos as distribuições de probabilidade das variáveis de projeto, bem como os valores da

média e da variância destas. Como ponto inicial para os testes numéricos, escolhemos o vetor de médias das variáveis originais.

Problema 1 [2]: $C(\underline{X}) = 0,1(X_1 - X_2)^2 - \dfrac{(X_1 + X_2)}{\sqrt{2}} + 2,5.$

Problema 2 [2]: $C(\underline{X}) = -0,5(X_1 - X_2)^2 - \dfrac{(X_1 + X_2)}{\sqrt{2}} + 3.$

Problema 3 [23]: de $C(\underline{X}) = 2 - X_2 - 0,1X_1^2 + 0,06X_1^3$.

Problema 4 [23]: $C(\underline{X}) = 3 - X_2 + 256X_1^4$.

Problema 5 [21]: $C(\underline{X}) = 1 + \dfrac{(X_1 + X_2)^2}{4} - 4(X_1 - X_2)^2.$

Problema 6 [23]: $C(\underline{X}) = 2 + 0,015\sum_{i=1}^{9} X_i^2 - X_{10}.$

Problema 7 [46]: A função de desempenho é $C(\underline{X}) = X_1^3 + X_2^3 - 18$, onde as variáveis são normalmente distribuídas com parâmetros $\mu_{X_1} = \mu_{X_1} = 10$ e $\sigma_{X_1} = \sigma_{X_2} = 5$.

Problema 8 [46]: $C(\underline{X}) = X_1^3 + X_2^3 - 18$, onde as variáveis são normalmente distribuídas com médias $\mu_{X_1} = 10$ e $\mu_{X_1} = 9,9$ e desvios padrão $\sigma_{X_1} = \sigma_{X_2} = 5$.

Problema 9 [23]: O relacionamento entre as variáveis consideradas nesse problema é dado pela função $C(\underline{X}) = 2,5 - 0,2357(X_1 - X_2) + 0,0046(X_1 + X_2 - 20)^4$, onde X_1 e X_2 seguem a distribuição normal, com médias 10 e desvio padrão 3.

Problema 10 [46]: As variáveis aleatórias consideradas nesse problema são normalmente distribuídas com médias $\mu_{X_1} = 10$ e $\mu_{X_1} = 9,9$, e desvios padrão $\sigma_{X_1} = \sigma_{X_2} = 5$. A função de desempenho é $C(\underline{X}) = X_1^3 + X_2^3 - 67,5$.

Problema 11 [23]: A função de desempenho é dada por $C(\underline{X}) = X_1X_2 - 146,14$, onde X_1 e X_2 seguem a distribuição normal com médias $\mu_{X_1} = 78064,4$ e $\mu_{X_2} = 0,0104$ e desvios padrão $\sigma_{X_1} = 11709,7$ e $\sigma_{X_2} = 0,00156$.

Problema 12 [21]: A função de desempenho considerada nesse problema é

$$C(\underline{X}) = 2,2257 - \frac{0,025\sqrt{2}}{27}(X_1 + X_2)^3 + 0,2357(X_1 - X_2),$$

onde X_1 e X_2 seguem a distribuição normal, com média 10 e desvio padrão 3.

Problema 13 [46]: $C(\underline{X}) = X_1X_2 - 2000X_3$, onde as variáveis X_1 e X_2 são normalmente distribuídas com médias $0,32$ e 1400000, e desvios padrão $0,032$ e 70000, respectivamente. A variável X_3 segue a distribuição lognormal com média 100 e desvio padrão 40.

Problema 14 [25]: Neste problema consideramos as variáveis aleatórias X_1 e X_2 que seguem a distribuição lognormal com médias 38 e 54, e desvios padrão $3,8$ e $2,7$, respectivamente. A função de desempenho é dada por $C(\underline{X}) = X_1X_2 - 1140$.

Problema 15 [33]: Neste problema consideramos a função de desempenho linear, dada por

$$C(\underline{X}) = X_1 + 2X_2 + 3X_3 + X_4 - 5X_5 - 5X_6,$$

onde as variáveis aleatórias seguem a distribuição lognormal 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 [29]: Neste problema consideramos variáveis aleatóricas com as mesmas distribuições das variáveis consideradas no problema anterior e a seguinte função de desempenho

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

Problema 17 [33]: A função de desempenho é dada por

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

onde as variáveis X_i, $i = 1, \ldots, 3$, seguem a distribuição lognormal 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 [46]: $C(\underline{X}) = X_1X_2 - 78,12X_3$, onde X_1 e X_2 são variávies normais e X_3 segue a distribuição de valores extremos (máximos) tipo I. As médias 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 $\sigma_{X_1} = 0,5 \times 10^7$, $\sigma_{X_2} = 2 \times 10^{-5}$ e $\sigma_{X_3} = 1$.

Problema 19 [46]: A função de desempenho considerada nesse problema é a mesma do Problema 18, mas agora temos que X_1 e X_2 seguem a distribuição lognormal e X_3 segue a distribuição de valores extremos (máximos) tipo I com as mesmas médias e desvios padrão apresentados no problema anterior.

Problema 20 [29]: A função de desempenho considerada nesse problema é

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

onde 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 normais com médias $\mu_{X_2} = 25$ 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 aleatória lognormal com média $\mu_{X_4} = 0,0625$ e desvio padrão $\sigma_{X_4} = 0,0625$.

Nos testes realizados escolhemos $\alpha = 0,1$ para o Algoritmo 1.1, pois este foi o valor que apresentou melhores resultados nos testes iniciais. Além disso, ao considerar os problemas que envolvem variáveis não normais, optamos por interromper a execução do algoritmo geral de filtro assim que ocorresse a primeira iteração do tipo f. Essa escolha se deve ao fato de que nesse caso, como explicado anteriormente, a cada iteração do Algoritmo *FORM* uma nova equação de estado limite é definida devido à mudança nos parâmetros da normal equivalente. Portanto, executar o algoritmo geral de filtro até obter o ponto sobre a equação de estado limite em consideração mais próximo da origem acarreta cálculos desnecessários, aumentando assim o número de avaliações de funções e gradientes. Os testes indicaram melhores resultados para as variantes do Algoritmo 1.1 quando empregamos essa estratégia.

Nenhum dos 6 algoritmos resolveu o Problema 5, pois $\nabla c(\mathbf{y}^0) = 0$. Assim, erros numéricos ocorreram no cálculo da direção $\mathbf{d}^k$, dada em (4.37), para os algoritmos HLRF e iHLRF. Já para os algoritmos de filtro, a falha ocorreu pois o algoritmo de filtro multidimensional retornou um ponto inviável, estacionário para a medida de inviabilidade h. Além de não resolver o Problema 5, o algoritmo HLRF alcançou o número máximo de iterações do Algoritmo *FORM* ($k = 1000$) ao tentar resolver os problemas 8, 10 e 20. O Algoritmo iHLRF também não resolveu o Problema 16, pois embora tenha se aproximado da solução, o critério de parada não foi satisfeito, excedendo assim o número máximo de iterações. Os resultados para os 20 problemas testados podem ser observados no Apêndice C.

Em confiabilidade estrutural há uma preocupação com o número de avaliações da função de desempenho e de sua derivada, pois em muitos problemas práticos essa função é dada de forma numérica, como por exemplo por meio de um modelo de elementos finitos. Nesse caso, cada avaliação de c demanda um elevado custo computacional, sendo maior ainda para a avaliação do gradiente [1]. Dessa forma, a fim de comparar os algoritmos, construímos gráficos de desempenho para o número de avaliações de c e de sua derivada, que estão apresentados na Figura 4.4.

Podemos observar que as variantes do Algoritmo 1.1 apresentaram um desempenho inferior aos algoritmos HLRF e iHLRF em relação ao número de avaliações da função c e de ∇c. No entanto, os algoritmos de PQS e RI foram mais robustos, uma vez que resolveram 19 problemas com sucesso, enquanto que HLRF e iHLRF resolveram 16 e 18 problemas, respectivamente. Além disso, entre as variantes do nosso algoritmo geral de filtro, PQS-Original foi o que apresentou melhor desempenho.

Notamos ainda que, diferentemente do que aconteceu nos testes numéricos realizados com problemas da coleção CUTEr, os resultados para os problemas de confiabilidade apresentaram diferenças mais significativas entre os critérios de filtro, sendo que tanto para PQS quanto RI, o critério de filtro original foi o que apresentou melhores resultados. Isto pode ser justificado pela escolha de $\alpha = 0,1$ e pelo fato do filtro inclinado ser mais restritivo que o original. No entanto, observamos que essa

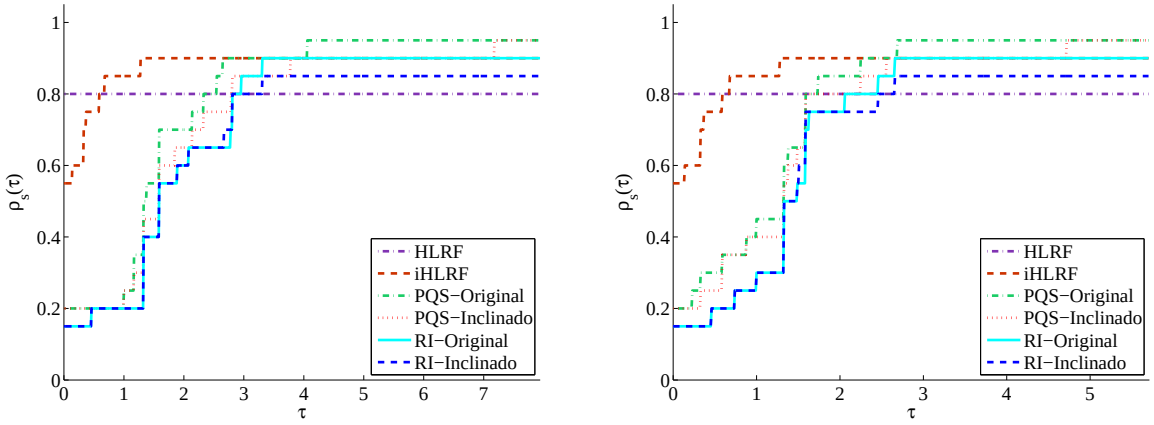

Figura 4.4: Gráfico de desempenho para o número de avaliações de c (à esquerda) e de ∇c (à direita), na escala $\log_2$.

diferença ocorreu apenas para os problemas que envolvem variáveis não normais. Em [39] mostramos que o desempenho do Algoritmo RI-Inclinado é semelhante ao do Algoritmo HLRF nestes casos, o que também ocorre para as demais variantes do algoritmo geral de filtro.

Os algoritmos de filtro apresentam uma vantagem em relação ao Algoritmo HLRF, sendo este o algoritmo mais usado em engenharia, que diz respeito à garantia de convergência. Como vimos, quando o algoritmo HLRF converge, a convergência é rápida e o custo computacional é baixo, porém, em muitos casos o algoritmo mostra-se instável, apresentando comportamento oscilatório, como observado nos problemas para os quais não obteve sucesso.

Concluímos este capítulo enfatizando que nosso objetivo com os testes apresentados aqui foi mostrar uma possibilidade de aplicação de nosso algoritmo geral de filtro a problemas práticos. O fato dos resultados terem sido inferiores aos alcançados pelos algoritmos HLRF e iHLRF, no que diz respeito ao número de avaliações da função de desempenho e seu gradiente, é justificado pela generalidade de nosso algoritmo, uma vez que este pode ser aplicado a problemas de otimização da forma (1.1). Por outro lado, os algoritmos HLRF e iHLRF foram desenvolvidos especificamente para o problema de confiabilidade, que possui uma estrutura muito particular, conforme apresentado em (4.15).

Nesse sentido, acreditamos que o desempenho dos algoritmos de filtro possam ser melhorados se a particularidade do problema em questão for levada em consideração e, ainda, se forem investigadas formas mais eficientes para acoplar o Algoritmo 1.1 ao Algoritmo *FORM*. Essa investigação foge ao escopo deste trabalho, mas deixa uma perspectiva para novos trabalhos a serem desenvolvidos nesse contexto. Além disso, é importante frisar que os testes numéricos foram realizados sobre um conjunto limitado de problemas de confiabilidade, o que nos impossibilita de tirar conclusões mais gerais sobre o desempenho dos algoritmos.

Conclusões

Apresentamos neste trabalho um algoritmo geral de filtro (Algoritmo 1.1) para resolver o problema de programação não linear (1.1) que permite uma grande liberdade no cálculo do passo e na definição da região proibida, dada pelo filtro original (1.4) ou inclinado (1.5). O passo pode ser calculado por qualquer método desde que seja eficiente no sentido de satisfazer a Hipótese H3, ou seja, de que perto de um ponto viável não estacionário, o decréscimo na função objetivo seja relativamente grande. Com esta hipótese estabelecemos a convergência global do algoritmo geral, de uma forma que nos parece bastante concisa.

Esse algoritmo já foi estudado em [16, 27, 43], no entanto nesses trabalhos os autores levam em consideração uma regra específica de filtro: Gonzaga, Karas e Vanti [16] e Ribeiro, Karas e Gonzaga [43] consideram o filtro original, enquanto que Karas, Oening e Ribeiro [27] trabalham com o filtro inclinado. Neste trabalho, diferentemente do que foi feito em [43], provamos que o Algoritmo 1.2, baseado em PQS com região de confiança, satisfaz a hipótese de eficiência do passo dada por H3, independente do critério de filtro considerado. Além disso, fundamentados em [16, 27], concluímos que esta hipótese também é válida se o passo for calculado por Restauração Inexata, como descrito no Algoritmo 1.3.

A fim de comparar os algoritmos estudados, implementamos os Algoritmos 1.1, 1.2 e 1.3 em MATLAB e realizamos testes numéricos com 300 problemas selecionados da coleção CUTEr. Os algoritmos para o cálculo do passo e os critérios de filtro deram origem a quatro variantes do algoritmo geral de filtro. Os resultados mostraram que, para esse conjunto de problemas, numericamente não existe diferença significativa entre os critérios de filtro original e inclinado, embora diferenças teóricas tenham sido observadas na análise de convergência. Além disso, os algoritmos para os quais o passo foi calculado por PQS mostraram-se mais robustos do que aqueles baseados em RI, uma vez que resolveram $92,67\%$ dos problemas com sucesso.

Para discutir a aplicabilidade dos métodos de filtro a problemas práticos, apresentamos um problema de otimização que surge em confiabilidade estrutural e realizamos testes numéricos com alguns problemas disponíveis na literatura especializada. Os testes indicaram que nosso algoritmo geral de filtro requer uma quantidade maior de avaliações de funções e gradientes que os algoritmos comumente empregados nesse contexto, HLRF e iHLRF. Porém, vimos que tal aplicação é possível e pode ser apri-

morada se levarmos em consideração as particularidades do problema em questão, na tentativa de tornar nosso algoritmo mais eficiente.

Sugestões para trabalhos futuros.

Os algoritmos internos usados para determinar o passo, baseados em PQS (Algoritmo 1.2) e RI (Algoritmo 1.3), empregam uma estratégia de região de confiança para controlar o tamanho do passo e induzir a convergência global. No entanto, existem na literatura métodos de filtro que ao invés de considerar regiões de confiança adotam uma estratégia de busca linear em que são determinados uma direção de busca que minimiza o modelo quadrático da função objetivo e o tamanho do passo a ser dado nessa direção. Como exemplo, temos os trabalhos de Chin [5, 6] e Wächter e Biegler [50, 51], que nos motivaram a pensar em um algoritmo interno baseado em busca linear para ser empregado no cálculo do passo do algoritmo geral de filtro.

Basicamente os métodos de busca linear utilizados para resolver o problema (1.1) consistem em, dado um ponto corrente x^k, determinar uma direção de busca d^k, obtendo um ponto tentativo

$$x^+ = x^k + td^k,$$

onde $t \in (0, 1]$ é o tamanho do passo a ser dado na direção d^k. Assim, utilizando uma estratégia *backtracking*, o tamanho do passo t é reduzido até que seja encontrado um valor para o qual o ponto tentativo x^+ satisfaça algum critério de aceitação de passo.

Estamos particularmente interessados em desenvolver um algoritmo de busca linear para o cálculo do passo que use o critério de filtro para avaliar o ponto tentativo e, assim como os Algoritmos 1.2 e 1.3, satisfaça a Hipótese H3.

Seguindo as ideias discutidas em [5], propomos um algoritmo que determina a direção d^k como solução do subproblema quadrático

$$\begin{array}{ll} \text{minimizar} & \nabla f(x^k)^T d + \frac{1}{2} d^T B_k d \\ \text{sujeito a} & x^k + d \in \mathcal{L}_k, \end{array} \tag{4.41}$$

onde B_k é uma matriz simétrica e $\mathcal{L}_k$ é dado em (1.11). Em seguida, calculamos o tamanho do passo por *backtracking*. Iniciamos com $t = 1$ e verificamos se $x^k + td^k \in \bar{\mathcal{F}}_k$. Em caso afirmativo, reduzimos o valor de t determinando assim um novo ponto tentativo que também será avaliado pelo filtro. No entanto, além de exigir que $x^k + td^k \notin \bar{\mathcal{F}}_k$ para ser aceito como próximo iterando, verificamos também uma condição de redução suficiente na função objetivo, assim como é feito nos métodos de região de confiança.

Sabemos que se d^k é uma direção de descida, então existe um $\delta > 0$ tal que a condição de Armijo, dada por

$$f(x^k + td^k) \leq f(x^k) + \eta t \nabla f(x^k)^T d^k, \tag{4.42}$$

é satisfeita para todo $t \in [0, \delta)$. Dessa forma, quando d^k for uma direção de descida ($\nabla f(x^k)^T d^k < 0$), mesmo que $x^k + td^k$ seja aceito pelo filtro, podemos impor uma condição de redução suficiente na função objetivo, dada por (4.42). Por meio desta condição podemos evitar que ocorra uma situação como aquela discutida por Wächter e Biegler [51], na qual é gerada uma sequência (x^k) que fornece apenas uma redução na medida de inviabilidade h e não em f, o que pode ocorrer quando confiamos apenas no critério de filtro.

Descrevemos a seguir o algoritmo de filtro com busca linear discutido acima.

Algoritmo *Cálculo de $x^{k+1} \notin \bar{\mathcal{F}}_k$ por busca linear*

Dados: $x^k \in \mathbb{R}^n$, $\bar{\mathcal{F}}_k$ e $t_{\min}$, $\gamma, \eta \in (0, 1)$.
SE $\mathcal{L}_k = \emptyset$,
    use o procedimento de restauração para obter $x^{k+1} \notin \bar{\mathcal{F}}_k$,
    obtenha B_{k+1} simétrica.
SENÃO
    calcule a direção de busca d^k, solução do problema (4.41).
    faça $t = 1$;
    REPITA (enquanto o ponto x^{k+1} não for obtido)
        $ared = f(x^k) - f(x^k + td^k)$
        SE $\left\{x^k + td^k \in \bar{\mathcal{F}}_k\right\}$ OU $\left\{\nabla f(x^k)^T d^k < 0 \ \text{ E } \ ared < -\eta t \nabla f(x^k)^T d^k\right\}$,
            $t = \gamma t$
            SE $t < t_{\min}$,
                use o procedimento de restauração para obter $x^{k+1} \notin \bar{\mathcal{F}}_k$;
                obtenha B_{k+1} simétrica.
        SENÃO
            $x^{k+1} = x^k + td^k$
            obtenha B_{k+1} simétrica.

Vemos que existem duas situações em que o algoritmo chama um procedimento de restauração cujo objetivo, assim como no Algoritmo 1.2, é determinar um ponto $x^{k+1} \notin \bar{\mathcal{F}}_k$ que minimize a medida de inviabilidade h. Isto é necessário quando o problema (4.41) é incompatível, ou seja, quando $\mathcal{L}_k = \emptyset$, e também quando for obtido um tamanho do passo $t < t_{\min}$, pois se a redução em t for realizada devido ao fato do ponto tentativo ser proibido pelo filtro, não há garantia de que encontraremos um valor de t para o qual $x^k + td^k \notin \bar{\mathcal{F}}_k$.

Deixamos como sugestão para trabalhos futuros, o desenvolvimento da prova de que se o passo for calculado pelo algoritmo de busca linear, então a Hipótese H3 será satisfeita, o que permitirá concluir que o nosso algoritmo geral de filtro com estratégia de busca linear é globalmente convergente. Testes numéricos podem ser realizados para comparar o desempenho deste algoritmo, considerando tanto o filtro original quanto o inclinado, com as demais variantes do Algoritmo 1.1.

Acreditamos ainda que o nosso algoritmo geral de filtro com o passo calculado pelo algoritmo de busca linear possa ser adaptado, a fim de se tornar aplicável ao problema de confiabilidade estrutural, uma vez que este algoritmo interno, assim como o iHLRF, emprega busca linear, mas possui a vantagem de não utilizar função de mérito.

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

[3] R. H. Byrd. Robust trust region methods for constrained optimization. Third SIAM Conference on Optimization, 1987.

[4] C. M. Chin. *A new trust region based SLP-filter algorithm which uses EQP active set strategy*. PhD thesis, Department of Mathematics, University of Dundee, Scotland, 2001.

[5] C. M. Chin. A global convergence theory of a filter line search method for nonlinear programming. Technical report, Numerical Optimization Report, Department of Statistics, University of Oxford, England, September 2002.

[6] C. M. Chin. A local convergence theory of a filter line search method for nonlinear programming. Technical report, Numerical Optimization Report, Department of Statistics, University of Oxford, England, January 2003.

[7] C. M. Chin e R. Fletcher. On the global convergence of an SLP-filter algorithm that takes EQP steps. *Mathematical Programming*, 96(1):161–177, 2003.

[8] D. Ditlevsen. Principle of normal tail approximation. *Journal of the Engineering Mechanics Division*, 107:1191–1208, 1981.

[9] E. D. Dolan e J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91:201–213, 2002.

[10] R. Fletcher, N. Gould, S. Leyffer, Ph. L. Toint e A. Wächter. Global convergence of a trust-region SQP-filter algorithm for general nonlinear programming. *SIAM Journal on Optimization*, 13(3):635–659, 2002.

[11] R. Fletcher e S. Leyffer. A bundle filter method for nonsmooth nonlinear optimization. Technical Report NA/195, Dundee University, Dept. of Mathematics, 1999.

[12] R. Fletcher e S. Leyffer. Nonlinear programming without a penalty function. *Mathematical Programming - Ser. A*, 91(2):239–269, 2002.

[13] R. Fletcher, S. Leyffer e Ph. L. Toint. On the global convergence of an SLP-filter algorithm. Technical Report NA/183, Dundee University, Dept. of Mathematics, 1998.

[14] R. Fletcher, S. Leyffer e Ph. L. Toint. On the global convergence of a filter-SQP algorithm. *SIAM Journal on Optimization*, 13(1):44–59, 2002.

[15] J. B. Francisco. *Viabilidade em programação não-linear: restauração e aplicações*. Tese de Doutorado, Universidade Estadual de Campinas, Campinas, São Paulo, 2005.

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

[17] N. I. M. Gould, S. Leyffer e Ph. L. Toint. A multidimensional filter algorithm for nonlinear equations and nonlinear least-squares. *SIAM Journal on Optimization*, 15(1):17–38, 2004.

[18] 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.

[19] N. I. M. Gould, C. Sainvitu e Ph. L. Toint. A filter-trust-region method for unconstrained optimization. *SIAM Journal on Optimization*, 16(2):341–357, 2006.

[20] N. I. M. Gould e Ph. L. Toint. FILTRANE, a fortran 95 filter-trust-region package for solving nonlinear least-squares and nonlinear feasibility problems. *ACM Transactions on Mathematical Software*, 33(1):3–25, 2007.

[21] R. V. Grandhi e L. Wang. Higher-order failure probability calculation using nonlinear approximations. *Computer methods in applied mechanics and engineering*, 168:185–206, 1999.

[22] A. Griewank, D. Juedes e J. Utke. Algorithm 755: Adol-c: A package for the automatic differentiation of algorithms written in c/c++. *ACM Transactions on Mathematical Software*, 22(2):135–167, 1996.

[23] F. Grooteman. Adaptive radial-base importance sampling method for structural reliability. *Structural Safety*, 30:533–542, 2008.

[24] C. Gu e D. Zhu. A secant algorithm with line search filter method for nonlinear optimization. *Applied Mathematical Modelling*, 35(2):879–894, 2011.

[25] A. Haldar e S. Mahadevan. *Probability, Reliability and Statistical Methods in Engineering Design*. John Wiley & Sons, New York, 2000.

[26] A. M. Hasofer e N. C. Lind. Exact and invariant second moment code format. *Jounal of Engineering Mechanics*, 100(1):111–121, 1974.

[27] 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.

[28] E. W. Karas, A. A. Ribeiro, C. Sagastizábal e M. Solodov. A bundle-filter method for nonsmooth convex constrained optimization. *Mathematical Programming, Ser. B*, 116:297–320, 2009.

[29] P. L. Liu e A. D. Kiureghian. Optimization algorithms for structural reliability. *Structural Safety*, 9:161–177, 1991.

[30] J. Long, C. Ma e P. Nie. A new filter method for solving nonlinear complementarity problems. *Applied Mathematics and Computation*, 185(1):705–718, 2007.

[31] J. Long e S. Zeng. A new Filter-Levenberg-Marquardt method with disturbance for solving nonlinear complementarity problems. *Applied Mathematics and Computation*, 216(2):677–688, 2010.

[32] D. G. Luenberger. A combined penalty function and gradient projection method for nonlinear programming. *Journal of Optimization Theory and Applications*, 14:477–495, 1974.

[33] S. Mahadevan e P. Shi. Multiple linearization method for nonlinear reliability analysis. *Journal of Engineering Mechanics*, 127(11):1165–1173, 2001.

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

[35] J. M. Martínez e E. A. Pilotta. Inexact restoration algorithm for constrained optimization. *Journal of Optimization Theory and Applications*, 104:135–163, 2000.

[36] R. E. Melchers. *Structural reliability analysis and prediction*. John Wiley & Sons, New York, 2nd edition, 1999.

[37] J. Nocedal e S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer-Verlag, 1999.

[38] E. Omojokun. *Trust Region Algorithms for Optimization with Nonlinear Equality and Inequality Constraints*. PhD thesis, Dept. of Computer Science, University of Colorado, 1991.

[39] 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. Em *XLIII Simpósio Brasileiro de Pesquisa Operacional*, Ubatuba, São Paulo, 2011.

[40] E. Polak. *Computational Methods in Optimization: A Unified Approach*. Academic Press, New York, 1971.

[41] R. Rackwitz e B. Fiessler. Structural reliability under combined load sequences. *Computers and Structures*, 9:489–494, 1978.

[42] A. A. Ribeiro. *Convergência global dos métodos de filtro para programação não linear*. Tese de Doutorado, Universidade Federal do Paraná, Curitiba, Paraná, 2005.

[43] A. A. Ribeiro, E. W. Karas e C. C. Gonzaga. Global convergence of filter methods for nonlinear programming. *SIAM Journal on Optimization*, 19(3):1231–1249, 2008.

[44] J. B. Rosen. The gradient projection method for nonlinear programming: part II, nonlinear constraints. *SIAM Journal on Applied Mathematics*, 9:514–532, 1961.

[45] S. R. Santos e L. C. Matioli. Desenvolvimento de algoritmos matemáticos aplicados a confiabilidade estrutural. In: E. N. Dvorkin, M. B. Goldschmit e M. A. Stori, editors, *Mecánica Computacional*, volume XXIX, pages 683–697, 2010.

[46] 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.

[47] C. Shen, W. Xue e X. Chen. Global convergence of a robust filter SQP algorithm. *European Journal of Operational Research*, 206:34–45, 2010.

[48] C. Shen, W. Xue e D. Pu. Global convergence of a tri-dimensional filter SQP algorithm based on line search method. *Applied Numerical Mathematics*, 59:235–250, 2009.

[49] M. Ulbrich, S. Ulbrich e L. N. Vicente. A globally convergent primal-dual interior-point filter method for nonlinear programming. *Mathematical Programming, Ser. A*, 100(2):379–410, 2004.

[50] A. Wächter e L. T. Biegler. Line search filter methods for nonlinear programming: Local convergence. *SIAM Journal on Optimization*, 16(1):32–48, 2005.

[51] A. Wächter e L. T. Biegler. Line search filter methods for nonlinear programming: Motivation and global convergence. *SIAM Journal on Optimization*, 16(1):1–31, 2005.

[52] X. Wang, Z. Zhu, S. Zuo e Q. Huang. An SQP-filter method for inequality constrained optimization and its global convergence. *Applied Mathematics and Computation*, 217(24):10224–10230, 2011.

[53] W. I. Zangwill. Nonlinear programming via penalty functions. *Management Science*, 13:344–358, 1967.

[54] Y. Zhang e A. D. Kiureghian. Finite element reliability methods for inelastic structures. Technical Report 1997, Department of Civil and Environmental Engineering, University of California, Berkeley.

Apêndice A

Problemas selecionados da coleção CUTEr

Apresentamos a seguir a relação dos 300 problemas selecionados da coleção CUTEr, com seus respectivos números de variáveis (n) e restrições de igualdade (p) e desigualdade (q).

Nome	n	p	q	Nome	n	p	q
AIRCRFTA	8	5	0	CANTILVR	5	0	1
AIRPORT	84	0	42	CB2	3	0	3
ALLINITC	4	1	0	CB3	3	0	3
ALSOTAME	2	1	0	CHACONN1	3	0	3
ARGTRIG	200	200	0	CHACONN2	3	0	3
AVGASA	8	0	10	CHANDHEQ	100	100	0
AVGASB	8	0	10	CLUSTER	2	2	0
AVION2	49	15	0	CONGIGMZ	3	0	5
BATCH	48	12	61	COSHFUN	10	0	3
BIGGSC4	4	0	7	CRESC100	6	0	200
BROWNALE	200	200	0	CRESC4	6	0	8
BT1	2	1	0	CSFI1	5	2	2
BT10	2	2	0	CUBENE	2	2	0
BT11	5	3	0	DALLASM	196	151	0
BT12	5	3	0	DECONVC	61	1	0
BT13	5	1	0	DEGENLPA	20	15	0
BT2	3	1	0	DEGENLPB	20	15	0
BT3	5	3	0	DEMYMALO	3	0	3
BT4	3	2	0	DIPIGRI	7	0	4
BT5	3	2	0	DISC2	29	17	6
BT7	5	3	0	DIXCHLNG	10	5	0
BT8	5	2	0	DNIEPER	61	24	0
BT9	4	2	0	DUAL1	85	1	0

DUAL2	96	1	0	HATFLDG	25	25	0
DUAL3	111	1	0	HATFLDH	4	0	7
DUAL4	75	1	0	HEART6	6	6	0
DUALC1	9	1	214	HEART8	8	8	0
DUALC2	7	1	228	HIMMELBA	2	2	0
DUALC5	8	1	277	HIMMELBC	2	2	0
DUALC8	8	1	502	HIMMELBE	3	3	0
EIGENA	110	110	0	HIMMELBI	100	0	12
EIGENB	110	110	0	HIMMELBJ	45	14	0
EIGENB2	110	55	0	HIMMELBK	24	14	0
EIGENBCO	110	55	0	HIMMELP2	2	0	1
EIGENC	30	30	0	HIMMELP3	2	0	2
EIGMAXA	101	101	0	HIMMELP4	2	0	3
EIGMAXB	101	101	0	HIMMELP5	2	0	3
EIGMAXC	22	22	0	HIMMELP6	2	0	5
EIGMINA	101	101	0	HONG	4	1	0
EIGMINB	101	101	0	HS10	2	0	1
EIGMINC	22	22	0	HS100	7	0	4
ELEC	75	25	0	HS101	7	0	5
EQC	9	0	3	HS102	7	0	5
EXPFITA	5	0	22	HS103	7	0	5
EXPFITB	5	0	102	HS104	8	0	5
EXPFITC	5	0	502	HS105	8	0	1
EXTRASIM	2	1	0	HS107	9	6	0
FCCU	19	8	0	HS108	9	0	13
FEEDLOC	90	19	240	HS11	2	0	1
FLETCHER	4	1	3	HS111	10	3	0
GENHS28	10	8	0	HS112	10	3	0
GIGOMEZ1	3	0	3	HS113	10	0	8
GIGOMEZ2	3	0	3	HS114	10	3	8
GIGOMEZ3	3	0	3	HS117	15	0	5
GMNCASE1	175	0	300	HS118	15	0	17
GMNCASE4	175	0	350	HS119	16	8	0
GOFFIN	51	0	50	HS20	2	0	3
GOTTFR	2	2	0	HS21	2	0	1
GRIDNETE	60	36	0	HS22	2	0	2
HAIFAM	99	0	150	HS23	2	0	5
HAIFAS	13	0	9	HS24	2	0	3
HALDMADS	6	0	42	HS26	3	1	0
HATFLDF	3	3	0	HS27	3	1	0

HS28	3	1	0	HS7	2	1	0
HS29	3	0	1	HS70	4	0	1
HS30	3	0	1	HS71	4	1	1
HS31	3	0	1	HS73	4	1	2
HS32	3	1	1	HS74	4	3	2
HS33	3	0	2	HS75	4	3	2
HS34	3	0	2	HS76	4	0	3
HS35	3	0	1	HS77	5	2	0
HS36	3	0	1	HS78	5	3	0
HS37	3	0	2	HS79	5	3	0
HS39	4	2	0	HS8	2	2	0
HS40	4	3	0	HS81	5	3	0
HS41	4	1	0	HS83	5	0	3
HS42	4	2	0	HS84	5	0	3
HS43	4	0	3	HS85	5	0	21
HS44	4	0	6	HS86	5	0	10
HS46	5	2	0	HS87	6	4	0
HS47	5	3	0	HS88	2	0	1
HS48	5	2	0	HS89	3	0	1
HS49	5	2	0	HS9	2	1	0
HS50	5	3	0	HS90	4	0	1
HS51	5	3	0	HS91	5	0	1
HS52	5	3	0	HS92	6	0	1
HS53	5	3	0	HS93	6	0	2
HS54	6	1	0	HS95	6	0	4
HS55	6	6	0	HS96	6	0	4
HS56	7	4	0	HS97	6	0	4
HS57	2	0	1	HS98	6	0	4
HS59	2	0	3	HS99	7	2	0
HS6	2	1	0	HUBFIT	2	0	1
HS60	3	1	0	HYDROELS	169	0	168
HS61	3	2	0	HYPCIR	2	2	0
HS62	3	1	0	KIWCRESC	3	0	2
HS63	3	2	0	LAUNCH	25	9	19
HS64	3	0	1	LEAKNET	156	153	0
HS65	3	0	1	LIN	4	2	0
HS66	3	0	2	LINSPANH	97	33	0
HS67	3	0	14	LISWET1	103	0	100
HS68	4	2	0	LISWET2	103	0	100
HS69	4	2	0	LISWET3	103	0	100

LISWET4	103	0	100	PFIT4	3	3	0
LISWET5	103	0	100	POLAK1	3	0	2
LISWET6	103	0	100	POLAK2	11	0	2
LOADBAL	31	11	20	POLAK3	12	0	10
LOOTSMA	3	0	2	POLAK4	3	0	3
LOTSCHD	12	7	0	POLAK5	3	0	2
LSNNODOC	5	4	0	POLAK6	5	0	4
LSQFIT	2	0	1	POWELL20	10	0	10
LUKVLE1	100	98	0	POWELLBS	2	2	0
MADSEN	3	0	6	POWELLSQ	2	2	0
MAKELA1	3	0	2	PRODPL0	60	20	9
MAKELA2	3	0	3	PRODPL1	60	20	9
MAKELA4	21	0	40	PT	2	0	501
MARATOS	2	1	0	QC	9	0	4
MATRIX2	6	0	2	QPCBLEND	83	43	31
METHANB8	31	31	0	QPNBLEND	83	43	31
MIFFLIN1	3	0	2	READING6	102	50	0
MIFFLIN2	3	0	2	RECIPE	3	3	0
MINMAXBD	5	0	20	RES	20	12	2
MINMAXRB	3	0	4	RK23	17	11	0
MISTAKE	9	0	13	ROBOT	14	2	0
MRIBASIS	36	9	46	ROSENMMX	5	0	4
MSS1	90	73	0	RSNBRNE	2	2	0
MWRIGHT	5	3	0	S268	5	0	5
OET1	3	0	202	S277-280	4	0	4
OET2	3	0	1002	S316-322	2	1	0
OET3	4	0	1002	SIMPLLPA	2	0	2
ODFITS	10	6	0	SIMPLLPB	2	0	3
OPTCNTRL	32	20	0	SINVALNE	2	2	0
OPTPRLOC	30	0	30	SMBANK	117	64	0
ORTHRDM2	203	100	0	SNAKE	2	0	2
ORTHREGA	133	64	0	SPIRAL	3	0	2
ORTHREGB	27	6	0	SUPERSIM	2	2	0
ORTHREGC	105	50	0	SWOPF	83	78	14
ORTHREGD	103	50	0	SYNTHES1	6	0	6
ORTHRGDM	23	10	0	SYNTHES2	11	1	13
PENTAGON	6	0	15	SYNTHES3	17	2	21
PFIT1	3	3	0	TAME	2	1	0
PFIT2	3	3	0	TFI1	3	0	101
PFIT3	3	3	0	TFI2	3	0	101

TFI3	3	0	101	WATER	31	10	0
TRIGGER	7	6	0	WOMFLET	3	0	3
TRUSPYR1	11	3	1	ZAMB2-8	138	48	0
TRUSPYR2	11	3	8	ZANGWIL3	3	3	0
TRY-B	2	1	0	ZECEVIC2	2	0	2
TWOBARS	2	0	2	ZECEVIC4	2	0	2
VANDERM1	100	100	99	ZY2	3	0	2

Apêndice B

Número de iterações para os problemas da coleção CUTEr

A seguir apresentamos os resultados para o número de iterações (ite) do Algoritmo 1.1, a quantidade de iterações do tipo h ($iteh$) e o número de pares (nF) no filtro final, para as quatro variantes PQS-Original, PQS-Inclinado, RI-Original e RI-Inclinado. Os problemas nos quais aparecem o símbolo $***$, são aqueles que apresentaram um valor para o parâmetro de saída diferente de 0.

Problema	PQS-Original			PQS-Inclinado			RI-Original			RI-Inclinado		
	ite	$iteh$	nF	ite	$iteh$	nF	ite	$iteh$	nF	ite	$iteh$	nF
AIRCRFTA	4	3	3	4	3	1	1	0	0	1	0	0
AIRPORT	233	112	78	233	112	78	***	***	***	***	***	***
ALLINITC	33	19	12	33	19	12	8	2	2	8	2	2
ALSOTAME	5	1	1	5	1	1	4	0	0	4	0	0
ARGTRIG	4	3	3	4	3	1	1	0	0	1	0	0
AVGASA	9	1	1	9	1	1	12	1	1	12	1	1
AVGASB	12	1	1	12	1	1	11	1	1	11	1	1
AVION2	***	***	***	***	***	***	***	***	***	***	***	***
BATCH	26	18	17	26	18	17	26	11	11	26	11	11
BIGGSC4	3	0	0	3	0	0	3	0	0	3	0	0
BROWNALE	2	1	1	2	1	1	1	0	0	1	0	0
BT1	157	1	1	157	1	1	***	***	***	***	***	***
BT10	7	6	6	7	6	6	1	0	0	1	0	0
BT11	13	8	5	13	8	5	9	4	3	9	4	3
BT12	8	1	1	8	1	1	7	1	1	7	1	1
BT13	20	11	8	20	11	1	***	***	***	***	***	***
BT2	14	0	0	14	0	0	9	0	0	9	0	0
BT3	8	0	0	8	0	0	8	0	0	8	0	0
BT4	9	4	4	9	4	4	7	5	5	7	5	5
BT5	7	3	3	7	3	3	5	3	3	5	3	3

BT7	10	3	3	10	3	3	10	5	5	10	5	5
BT8	12	0	0	12	0	0	2	0	0	2	0	0
BT9	13	12	12	13	12	12	6	3	3	6	3	3
CANTILVR	25	14	13	25	14	13	21	6	4	21	6	4
CB2	8	6	6	8	6	6	4	3	3	4	3	3
CB3	7	5	5	7	5	5	1	0	0	1	0	0
CHACONN1	7	5	5	7	5	5	5	3	3	5	3	3
CHACONN2	7	6	6	7	6	5	1	0	0	1	0	0
CHANDHEQ	12	11	5	12	11	1	***	***	***	***	***	***
CLUSTER	9	8	4	9	8	1	***	***	***	***	***	***
CONGIGMZ	3	2	2	3	2	2	1	0	0	1	0	0
COSHFUN	14	10	7	14	10	7	11	5	5	11	5	5
CRESC100	630	44	9	625	44	9	***	***	***	***	***	***
CRESC4	***	***	***	***	***	***	35	2	1	35	2	1
CSFI1	12	2	1	12	2	1	9	1	1	9	1	1
CUBENE	2	1	1	2	1	1	1	0	0	1	0	0
DALLASM	100	0	0	100	0	0	101	0	0	101	0	0
DECONVC	109	0	0	109	0	0	95	0	0	95	0	0
DEGENLPA	2	0	0	2	0	0	***	***	***	***	***	***
DEGENLPB	2	1	1	2	1	1	***	***	***	***	***	***
DEMYMALO	8	5	4	8	5	4	4	0	0	4	0	0
DIPIGRI	19	8	4	19	8	4	22	8	6	22	8	6
DISC2	18	8	5	18	8	5	17	6	6	17	6	6
DIXCHLNG	***	***	***	***	***	***	***	***	***	***	***	***
DNIEPER	7	2	2	7	2	2	***	***	***	***	***	***
DUAL1	153	1	1	153	1	1	156	1	1	156	1	1
DUAL2	195	1	1	195	1	1	215	1	1	215	1	1
DUAL3	226	1	1	226	1	1	248	1	1	248	1	1
DUAL4	85	2	2	85	2	2	63	1	1	63	1	1
DUALC1	12	1	1	12	1	1	14	1	1	14	1	1
DUALC2	12	1	1	12	1	1	15	1	1	15	1	1
DUALC5	26	1	1	26	1	1	26	1	1	26	1	1
DUALC8	15	2	2	15	2	2	11	1	1	11	1	1
EIGENA	2	1	1	2	1	1	1	0	0	1	0	0
EIGENB	2	1	1	2	1	1	1	0	0	1	0	0
EIGENB2	2	0	0	2	0	0	2	0	0	2	0	0
EIGENBCO	2	0	0	2	0	0	2	0	0	2	0	0
EIGENC	2	1	1	2	1	1	1	0	0	1	0	0
EIGMAXA	2	1	1	2	1	1	1	0	0	1	0	0
EIGMAXB	3	2	2	3	2	2	***	***	***	***	***	***

EIGMAXC	2	1	1	2	1	1	1	0	0	1	0	0
EIGMINA	2	0	0	2	0	0	1	0	0	1	0	0
EIGMINB	8	5	5	8	5	5	***	***	***	***	***	***
EIGMINC	2	0	0	2	0	0	1	0	0	1	0	0
ELEC	504	180	74	504	180	74	595	202	111	611	216	92
EQC	4	1	1	4	1	1	***	***	***	***	***	***
EXPFITA	23	0	0	23	0	0	23	0	0	23	0	0
EXPFITB	20	0	0	20	0	0	20	0	0	20	0	0
EXPFITC	19	0	0	19	0	0	19	0	0	19	0	0
EXTRASIM	2	1	1	2	1	1	2	1	1	2	1	1
FCCU	28	0	0	28	0	0	29	0	0	29	0	0
FEEDLOC	5	3	2	5	3	1	4	2	2	4	2	1
FLETCHER	9	5	5	9	5	5	6	5	5	6	5	5
GENHS28	6	0	0	6	0	0	8	0	0	8	0	0
GIGOMEZ1	9	5	4	9	5	4	4	0	0	4	0	0
GIGOMEZ2	9	6	5	9	6	5	7	4	4	7	4	4
GIGOMEZ3	7	5	5	7	5	5	2	0	0	2	0	0
GMNCASE1	24	5	3	24	5	3	25	1	1	25	1	1
GMNCASE4	2	1	1	2	1	1	1	0	0	1	0	0
GOFFIN	7	1	1	7	1	1	6	1	1	6	1	1
GOTTFR	3	2	2	3	2	1	1	0	0	1	0	0
GRIDNETE	36	1	1	36	1	1	35	1	1	35	1	1
HAIFAM	***	***	***	***	***	***	***	***	***	***	***	***
HAIFAS	9	5	5	9	5	5	8	4	4	8	4	4
HALDMADS	24	3	3	24	3	3	8	2	2	8	2	2
HATFLDF	2	1	1	2	1	1	***	***	***	***	***	***
HATFLDG	2	1	1	2	1	1	1	0	0	1	0	0
HATFLDH	5	0	0	5	0	0	3	0	0	3	0	0
HEART6	2	1	1	2	1	1	1	0	0	1	0	0
HEART8	2	1	1	2	1	1	1	0	0	1	0	0
HIMMELBA	2	1	1	2	1	1	1	0	0	1	0	0
HIMMELBC	2	1	1	2	1	1	1	0	0	1	0	0
HIMMELBE	2	1	1	2	1	1	1	0	0	1	0	0
HIMMELBI	246	5	3	246	5	3	249	0	0	249	0	0
HIMMELBJ	***	***	***	***	***	***	***	***	***	***	***	***
HIMMELBK	19	2	2	19	2	2	19	1	1	19	1	1
HIMMELP2	16	0	0	16	0	0	16	0	0	16	0	0
HIMMELP3	7	0	0	7	0	0	7	0	0	7	0	0
HIMMELP4	7	0	0	7	0	0	7	0	0	7	0	0
HIMMELP5	10	0	0	10	0	0	13	0	0	13	0	0

HIMMELP6	11	0	0	11	0	0	12	0	0	12	0	0
HONG	16	1	1	16	1	1	14	0	0	14	0	0
HS10	11	9	9	11	9	9	6	5	5	6	5	5
HS100	19	8	4	19	8	4	22	8	6	22	8	6
HS101	***	***	***	***	***	***	***	***	***	***	***	***
HS102	***	***	***	***	***	***	***	***	***	***	***	***
HS103	***	***	***	***	***	***	***	***	***	***	***	***
HS104	18	8	6	18	8	6	15	6	5	15	6	5
HS105	41	0	0	41	0	0	42	0	0	42	0	0
HS107	8	5	5	8	5	5	***	***	***	***	***	***
HS108	13	7	5	13	7	5	5	2	2	5	2	2
HS11	8	5	5	8	5	5	5	3	3	5	3	3
HS111	53	24	16	53	24	16	50	26	14	50	26	14
HS112	31	0	0	31	0	0	34	0	0	34	0	0
HS113	17	4	3	17	4	3	13	2	2	13	2	2
HS114	29	8	4	29	8	4	29	9	5	29	9	5
HS117	23	0	0	23	0	0	23	0	0	23	0	0
HS118	20	0	0	20	0	0	20	0	0	20	0	0
HS119	13	0	0	13	0	0	14	0	0	14	0	0
HS20	7	0	0	7	0	0	4	0	0	4	0	0
HS21	4	0	0	4	0	0	3	0	0	3	0	0
HS22	5	3	2	5	3	2	3	1	1	3	1	1
HS23	6	0	0	6	0	0	6	0	0	6	0	0
HS24	5	0	0	5	0	0	5	0	0	5	0	0
HS26	23	0	0	23	0	0	20	0	0	20	0	0
HS27	9	4	2	9	4	2	8	3	1	8	3	1
HS28	9	0	0	9	0	0	9	0	0	9	0	0
HS29	13	9	8	13	9	8	9	4	4	9	4	4
HS30	12	0	0	12	0	0	12	0	0	12	0	0
HS31	13	8	5	13	8	5	8	3	3	8	3	3
HS32	3	0	0	3	0	0	3	0	0	3	0	0
HS33	5	0	0	5	0	0	5	0	0	5	0	0
HS34	8	2	2	8	2	2	5	0	0	5	0	0
HS35	7	0	0	7	0	0	7	0	0	7	0	0
HS36	4	0	0	4	0	0	4	0	0	4	0	0
HS37	4	0	0	4	0	0	4	0	0	4	0	0
HS39	13	12	12	13	12	12	6	3	3	6	3	3
HS40	7	6	5	7	6	5	5	3	3	5	3	3
HS41	6	1	1	6	1	1	5	1	1	5	1	1
HS42	10	6	5	10	6	5	8	4	4	8	4	4

HS43	14	10	10	14	10	10	11	5	4	11	5	4
HS44	6	0	0	6	0	0	6	0	0	6	0	0
HS46	26	0	0	26	0	0	21	0	0	21	0	0
HS47	24	0	0	24	0	0	15	1	1	15	1	1
HS48	9	0	0	9	0	0	9	0	0	9	0	0
HS49	23	0	0	23	0	0	23	0	0	23	0	0
HS50	13	0	0	13	0	0	13	0	0	13	0	0
HS51	8	0	0	8	0	0	8	0	0	8	0	0
HS52	9	0	0	9	0	0	9	0	0	9	0	0
HS53	8	0	0	8	0	0	8	0	0	8	0	0
HS54	2	1	1	2	1	1	30	1	1	30	1	1
HS55	2	1	1	2	1	1	1	0	0	1	0	0
HS56	15	5	5	15	5	5	9	4	4	9	4	4
HS57	16	1	1	16	1	1	16	1	1	16	1	1
HS59	13	0	0	13	0	0	17	0	0	17	0	0
HS6	10	1	1	10	1	1	7	0	0	7	0	0
HS60	11	0	0	11	0	0	9	0	0	9	0	0
HS61	***	***	***	***	***	***	6	3	3	6	3	3
HS62	14	0	0	14	0	0	14	0	0	14	0	0
HS63	7	3	3	7	3	3	5	3	3	5	3	3
HS64	24	7	5	24	7	5	***	***	***	***	***	***
HS65	11	5	4	11	5	4	10	3	3	10	3	3
HS66	7	3	3	7	3	3	6	2	2	6	2	2
HS67	22	0	0	22	0	0	22	0	0	22	0	0
HS68	21	9	5	21	9	5	16	3	2	16	3	2
HS69	20	7	3	20	7	3	***	***	***	***	***	***
HS7	12	9	9	12	9	9	6	4	4	6	4	4
HS70	37	0	0	37	0	0	37	0	0	37	0	0
HS71	6	5	5	6	5	5	6	4	4	6	4	4
HS73	5	1	1	5	1	1	3	0	0	3	0	0
HS74	11	3	3	11	3	3	10	3	3	10	3	3
HS75	8	2	2	8	2	2	***	***	***	***	***	***
HS76	7	0	0	7	0	0	7	0	0	7	0	0
HS77	12	5	4	12	5	4	13	3	3	13	3	3
HS78	8	5	5	8	5	5	5	2	2	5	2	2
HS79	9	0	0	9	0	0	8	0	0	8	0	0
HS8	2	1	1	2	1	1	1	0	0	1	0	0
HS81	8	5	4	8	5	4	5	3	3	5	3	3
HS83	6	2	2	6	2	2	6	1	1	6	1	1
HS84	6	1	1	6	1	1	6	1	1	6	1	1

HS85	33	3	2	33	3	2	32	2	2	32	2	2
HS86	6	0	0	6	0	0	6	0	0	6	0	0
HS87	***	***	***	***	***	***	***	***	***	***	***	***
HS88	***	***	***	***	***	***	***	***	***	***	***	***
HS89	***	***	***	***	***	***	***	***	***	***	***	***
HS9	7	0	0	7	0	0	7	0	0	7	0	0
HS90	***	***	***	***	***	***	***	***	***	***	***	***
HS91	30	18	16	30	18	16	***	***	***	***	***	***
HS92	***	***	***	***	***	***	***	***	***	***	***	***
HS93	19	13	9	19	13	9	13	5	5	13	5	5
HS95	2	1	1	2	1	1	3	1	1	3	1	1
HS96	2	1	1	2	1	1	3	1	1	3	1	1
HS97	7	3	3	7	3	3	5	2	2	5	2	2
HS98	7	3	3	7	3	3	5	2	2	5	2	2
HS99	50	11	5	50	11	5	***	***	***	***	***	***
HUBFIT	7	0	0	7	0	0	7	0	0	7	0	0
HYDROELS	***	***	***	***	***	***	***	***	***	***	***	***
HYPCIR	2	1	1	2	1	1	1	0	0	1	0	0
KIWCRESC	10	7	6	10	7	6	6	4	3	6	4	3
LAUNCH	54	1	1	54	1	1	***	***	***	***	***	***
LEAKNET	36	5	3	36	5	3	***	***	***	***	***	***
LIN	***	***	***	***	***	***	***	***	***	***	***	***
LINSPANH	1	0	0	1	0	0	1	0	0	1	0	0
LISWET1	2	0	0	2	0	0	2	0	0	2	0	0
LISWET2	2	0	0	2	0	0	2	0	0	2	0	0
LISWET3	2	0	0	2	0	0	2	0	0	2	0	0
LISWET4	3	0	0	3	0	0	3	0	0	3	0	0
LISWET5	4	1	1	4	1	1	3	0	0	3	0	0
LISWET6	2	0	0	2	0	0	2	0	0	2	0	0
LOADBAL	105	0	0	105	0	0	105	0	0	105	0	0
LOOTSMA	***	***	***	***	***	***	***	***	***	***	***	***
LOTSCHD	6	1	1	6	1	1	6	1	1	6	1	1
LSNNODOC	5	0	0	5	0	0	5	0	0	5	0	0
LSQFIT	6	0	0	6	0	0	6	0	0	6	0	0
LUKVLE1	17	1	1	17	1	1	9	0	0	9	0	0
MADSEN	13	8	7	13	8	7	8	4	4	8	4	4
MAKELA1	7	4	4	7	4	4	4	2	2	4	2	2
MAKELA2	9	7	6	9	7	6	5	3	3	5	3	3
MAKELA4	20	1	1	20	1	1	17	1	1	17	1	1
MARATOS	4	3	3	4	3	3	4	2	2	4	2	2

MATRIX2	14	1	1	14	1	1	12	1	1	12	1	1
METHANB8	3	2	2	3	2	1	***	***	***	***	***	***
MIFFLIN1	7	4	4	7	4	4	6	3	3	6	3	3
MIFFLIN2	8	5	5	8	5	5	6	3	3	6	3	3
MINMAXBD	18	7	7	18	7	7	15	4	4	15	4	4
MINMAXRB	8	2	2	8	2	2	7	2	1	7	2	1
MISTAKE	10	5	5	10	5	5	9	3	3	9	3	3
MRIBASIS	8	2	2	8	2	1	6	1	1	6	1	1
MSS1	696	181	81	696	181	81	***	***	***	***	***	***
MWRIGHT	14	7	6	14	7	6	9	2	2	9	2	2
OET1	5	1	1	5	1	1	5	1	1	5	1	1
OET2	7	1	1	7	1	1	10	3	2	10	3	2
OET3	6	1	1	6	1	1	3	1	1	3	1	1
ODFITS	39	1	1	39	1	1	38	1	1	38	1	1
OPTCNTRL	3	2	2	3	2	2	2	1	1	2	1	1
OPTPRLOC	11	3	2	11	3	2	9	3	2	9	3	2
ORTHRDM2	12	4	4	12	4	4	13	3	3	13	3	3
ORTHREGA	45	3	3	45	3	3	32	2	2	32	2	2
ORTHREGB	7	1	1	7	1	1	9	1	1	9	1	1
ORTHREGC	44	3	3	44	3	3	67	5	5	67	5	5
ORTHREGD	14	4	4	14	4	4	17	4	4	17	4	4
ORTHRGDM	28	3	3	28	3	3	15	3	3	15	3	3
PENTAGON	34	0	0	34	0	0	31	0	0	31	0	0
PFIT1	2	1	1	2	1	1	1	0	0	1	0	0
PFIT2	2	1	1	2	1	1	1	0	0	1	0	0
PFIT3	2	1	1	2	1	1	1	0	0	1	0	0
PFIT4	2	1	1	2	1	1	1	0	0	1	0	0
POLAK1	12	6	5	12	6	5	11	5	4	11	5	4
POLAK2	***	***	***	***	***	***	***	***	***	***	***	***
POLAK3	19	12	8	19	12	8	15	7	7	15	7	7
POLAK4	10	6	3	10	6	2	8	3	2	8	3	1
POLAK5	13	3	2	13	3	2	8	2	2	8	2	2
POLAK6	20	9	9	20	9	9	11	5	5	11	5	5
POWELL20	2	1	1	2	1	1	1	0	0	1	0	0
POWELLBS	12	11	6	12	11	1	***	***	***	***	***	***
POWELLSQ	***	***	***	***	***	***	***	***	***	***	***	***
PRODPL0	7	2	2	7	2	2	5	1	1	5	1	1
PRODPL1	6	2	2	6	2	2	6	1	1	6	1	1
PT	4	1	1	4	1	1	3	1	1	3	1	1
QC	2	0	0	2	0	0	2	0	0	2	0	0

QPCBLEND	11	0	0	11	0	0	11	0	0	11	0	0
QPNBLEND	7	0	0	7	0	0	7	0	0	7	0	0
READING6	59	2	2	59	2	2	96	1	1	96	1	1
RECIPE	5	4	2	5	4	1	***	***	***	***	***	***
RES	1	0	0	1	0	0	1	0	0	1	0	0
RK23	6	3	3	6	3	3	6	2	2	6	2	2
ROBOT	2	1	1	2	1	1	1	0	0	1	0	0
ROSENMMX	18	10	10	18	10	10	11	3	3	11	3	3
RSNBRNE	2	1	1	2	1	1	1	0	0	1	0	0
S268	***	***	***	***	***	***	22	0	0	22	0	0
S277-280	10	1	1	10	1	1	***	***	***	***	***	***
S316-322	***	***	***	***	***	***	***	***	***	***	***	***
SIMPLLPA	2	1	1	2	1	1	2	1	1	2	1	1
SIMPLLPB	3	1	1	3	1	1	3	1	1	3	1	1
SINVALNE	2	1	1	2	1	1	1	0	0	1	0	0
SMBANK	***	***	***	***	***	***	***	***	***	***	***	***
SNAKE	5	2	1	5	2	1	5	1	1	5	1	1
SPIRAL	131	38	9	131	38	9	***	***	***	***	***	***
SUPERSIM	2	1	1	2	1	1	1	0	0	1	0	0
SWOPF	31	8	6	31	8	6	***	***	***	***	***	***
SYNTHES1	9	3	3	9	3	3	9	3	2	9	3	2
SYNTHES2	5	2	2	5	2	2	5	2	2	5	2	2
SYNTHES3	11	0	0	11	0	0	11	0	0	11	0	0
TAME	2	1	1	2	1	1	1	0	0	1	0	0
TFI1	12	4	3	12	4	3	72	24	16	72	24	16
TFI2	7	1	1	7	1	1	5	1	1	5	1	1
TFI3	4	0	0	4	0	0	4	0	0	4	0	0
TRIGGER	7	6	2	7	6	1	***	***	***	***	***	***
TRUSPYR1	24	8	5	24	8	5	22	6	4	22	6	4
TRUSPYR2	14	1	1	14	1	1	13	1	1	13	1	1
TRY-B	2	0	0	2	0	0	1	0	0	1	0	0
TWOBARS	9	7	7	9	7	7	10	5	4	10	5	4
VANDERM1	2	1	1	2	1	1	1	0	0	1	0	0
WATER	37	1	1	37	1	1	37	1	1	37	1	1
WOMFLET	24	8	2	24	8	1	18	5	1	18	5	1
ZAMB2-8	268	0	0	268	0	0	269	0	0	269	0	0
ZANGWIL3	2	1	1	2	1	1	1	0	0	1	0	0
ZECEVIC2	4	0	0	4	0	0	4	0	0	4	0	0
ZECEVIC4	9	1	1	9	1	1	8	0	0	8	0	0
ZY2	5	0	0	5	0	0	5	0	0	5	0	0

Apêndice C

Resultados para os problemas de confiabilidade estrutural

Na tabela a seguir apresentamos os resultados dos testes numéricos realizados com os problemas de confiabilidade apresentados no Capítulo 4. Indicamos por $\#c$, $\#\nabla c$ e β, o número de avaliações da função de desempenho e de seu gradiente, e o valor do índice de confiabilidade, respectivamente, obtidos para cada um dos 20 problemas. Os números 1 e 2 ao lado de PQS e RI representam o critério de filtro original e inclinado, respectivamente. O símbolo *** indica que o problema não foi resolvido com sucesso pelo algoritmo.

Problema		HLRF	iHLRF	PQS1	PQS2	RI1	RI2
1	$\#c$	2	2	5	5	5	5
	$\#\nabla c$	2	2	5	5	5	5
	β	2,5000	2,5000	2,5000	2,5000	2,5000	2,5000
2	$\#c$	2	2	6	6	6	6
	$\#\nabla c$	2	2	6	6	6	6
	β	3,0000	3,0000	3,0000	3,0000	3,0000	3,0000
3	$\#c$	2	2	5	5	5	5
	$\#\nabla c$	2	2	5	5	5	5
	β	2,0000	2,0000	2,0000	2,0000	2,0000	2,0000
4	$\#c$	2	2	6	6	6	6
	$\#\nabla c$	2	2	6	6	6	6
	β	3,0000	3,0000	3,0000	3,0000	3,0000	3,0000
5	$\#c$	***	***	***	***	***	***
	$\#\nabla c$	***	***	***	***	***	***
	β	***	***	***	***	***	***
6	$\#c$	2	2	5	5	5	5
	$\#\nabla c$	2	2	5	5	5	5
	β	2,0000	2,0000	2,0000	2,0000	2,0000	2,0000
7	$\#c$	8	10	18	18	11	11
	$\#\nabla c$	8	10	10	10	11	11
	β	2,2401	2,2401	2,2401	2,2401	2,2401	2,2401

8	$\#c$	***	16	26	26	29	29
	$\#\nabla c$	***	16	11	11	20	20
	β	***	2,2260	2,2260	2,2260	2,2260	2,2260
9	$\#c$	2	3	6	6	6	6
	$\#\nabla c$	2	3	6	6	6	6
	β	2,5000	2,5000	2,5000	2,5000	2,5000	2,5000
10	$\#c$	***	21	26	26	28	28
	$\#\nabla c$	***	21	11	11	19	19
	β	***	1,9003	1,9003	1,9003	1,9003	1,9003
11	$\#c$	6	6	30	30	42	42
	$\#\nabla c$	6	6	11	11	10	10
	β	5,428	5,428	5,428	5,428	5,428	5,428
12	$\#c$	2	2	4	4	5	5
	$\#\nabla c$	2	2	3	3	5	5
	β	2,2257	2,2257	2,2257	2,2257	2,2257	2,2257
13	$\#c$	6	6	35	866	41	1474
	$\#\nabla c$	6	6	20	158	25	314
	β	2,1911	2,1911	2,1911	2,1911	2,1911	2,1912
14	$\#c$	4	5	25	28	28	28
	$\#\nabla c$	4	5	19	19	22	22
	β	5,2127	5,2127	5,2127	5,2127	5,2127	5,2127
15	$\#c$	5	8	22	22	21	21
	$\#\nabla c$	5	8	13	13	10	10
	β	3,0434	3,0428	3,0432	3,0432	3,0427	3,0427
16	$\#c$	247	***	212	133	243	205
	$\#\nabla c$	247	***	41	31	86	65
	β	2,3483	***	2,3482	2,3483	2,3482	2,3482
17	$\#c$	12	29	27	43	93	76
	$\#\nabla c$	12	29	14	18	37	34
	β	0,8292	0,8292	0,8292	0,8292	0,8292	0,8292
18	$\#c$	10	11	26	69	37	37
	$\#\nabla c$	10	11	20	28	28	28
	β	3,3221	3,3221	3,3221	3,3221	3,3221	3,3221
19	$\#c$	7	9	116	96	69	69
	$\#\nabla c$	7	9	45	41	44	44
	β	4,4282	4,4282	4,4282	4,4282	4,4282	4,4282
20	$\#c$	***	11	20	20	34	34
	$\#\nabla c$	***	11	12	12	25	25
	β	***	1,3654	1,3657	1,3657	1,3677	1,3677