UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA

DANILO RODRIGUES DE SOUZA

Método do gradiente conjugado Dai-Yuan modificado para otimização irrestrita

Goiânia
2019

sistema de bibliotecas ufg

PRPG
PRÓ-REITORIA DE PÓS-GRADUAÇÃO

UFG

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TESES E DISSERTAÇÕES NA BIBLIOTECA DIGITAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio da Biblioteca Digital de Teses e Dissertações (BDTD/UFG), regulamentada pela Resolução CEPEC nº 832/2007, sem ressarcimento dos direitos autorais, de acordo com a Lei nº 9610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou *download*, a título de divulgação da produção científica brasileira, a partir desta data.

1. Identificação do material bibliográfico: **[x] Dissertação** **[ ] Tese**

2. Identificação da Tese ou Dissertação:

Nome completo do autor: Danilo Rodrigues de Souza.

Título do trabalho: Método do gradiente conjugado Dai-Yuan modificado para otimização irrestrita.

3. Informações de acesso ao documento:

Concorda com a liberação total do documento [x] SIM [ ] NÃO[1]

Havendo concordância com a disponibilização eletrônica, torna-se imprescindível o envio do(s) arquivo(s) em formato digital PDF da tese ou dissertação.

Danilo Rodrigues de Souza.
Assinatura do(a) autor(a)[2]

Ciente e de acordo:

Assinatura do(a) orientador(a)[2]

Data: 19 / 03 / 2019

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. A extensão deste prazo suscita justificativa junto à coordenação do curso. Os dados do documento não serão disponibilizados durante o período de embargo.
Casos de embargo:
- Solicitação de registro de patente;
- Submissão de artigo em revista científica;
- Publicação como capítulo de livro;
- Publicação da dissertação/tese em livro.

[2] A assinatura deve ser escaneada.

DANILO RODRIGUES DE SOUZA

Método do gradiente conjugado Dai-Yuan modificado para otimização irrestrita

Dissertação apresentada ao Programa de Pós–Graduação do Instituto de Matemática e Estatística da Universidade Federal de Goiás, como requisito parcial para obtenção do título de Mestre em Programa de Pós-Graduação em Matemática.

Área de concentração: Otimização.

Orientador: Prof. Leandro da Fonseca Prudente

Goiânia
2019

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

Souza, Danilo Rodrigues de Souza
Método do gradiente conjugado Dai-Yuan modificado para otimização irrestrita [manuscrito] / Danilo Rodrigues de Souza Souza. - 2019.
71 f.: il.

Orientador: Prof. Dr. Leandro da Fonseca Prudente Fonseca.
Dissertação (Mestrado) - Universidade Federal de Goiás, Instituto de Matemática e Estatística (IME), Programa de Pós-Graduação em Matemática, Goiânia, 2019.
Bibliografia. Apêndice.
Inclui gráfico, tabelas, algoritmos.

1. Método do gradiente conjugado modificado.. 2. Otimização irrestrita. 3. Condições de Wolfe. 4. Algoritmo de busca linear. 5. Análise numérica. I. Fonseca, Leandro da Fonseca Prudente, orient. II. Título.

CDU 51

UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM MATEMÁTICA
Campus Samambaia – Caixa Postal 131 – CEP: 74.001-970 – Goiânia-GO.
Fones: (62) 3521-1208 e 3521-1137 www.ime.ufg.br

ATA DA REUNIÃO DA BANCA EXAMINADORA DA DEFESA DE DISSERTAÇÃO DE DANILO RODRIGUES DE SOUZA – Ao vigésimo oitavo dia do mês de fevereiro do ano de dois mil e dezenove (28/02/2019), às 15 horas, reuniram-se os componentes da Banca Examinadora: Prof. Leandro da Fonseca Prudente - Orientador, Prof. Luis Román Lucambio Pérez, Prof. Max Leandro Nobre Gonçalves e Prof. Douglas Soares Gonçalves para, sob a presidência do primeiro, e em sessão pública realizada na sala Geraldo Ávila do Instituto de Matemática e Estatística, procederem a avaliação da defesa de dissertação intitulada: **"Método do Gradiente conjugado Dai-Yuan modificado para otimização irrestrita"**, em nível de Mestrado, área de concentração em Otimização, de autoria de Danilo Rodrigues de Souza, discente do Programa de Pós-Graduação em Matemática da Universidade Federal de Goiás. A sessão foi aberta pelo Presidente da Banca, Prof. Leandro da Fonseca Prudente, que fez a apresentação formal dos membros da Banca. A seguir, a palavra foi concedida ao autor da dissertação que, em 45 minutos procedeu à apresentação de seu trabalho. Terminada a apresentação, cada membro da Banca arguiu o examinando, tendo-se adotado o sistema de diálogo sequencial. Terminada a fase de arguição, procedeu-se a avaliação da defesa. Tendo-se em vista o que consta na Resolução nº. 1513 do Conselho de Ensino, Pesquisa, Extensão e Cultura (CEPEC), que regulamenta o Programa de Pós-Graduação em Matemática e procedidas às correções recomendadas, a dissertação foi **APROVADA** por unanimidade, considerando-se integralmente cumprido este requisito para fins de obtenção do título de **MESTRE EM MATEMÁTICA**, na área de concentração em Otimização, pela Universidade Federal de Goiás. A conclusão do curso dar-se-á quando da entrega na secretaria do PPGM da versão definitiva da dissertação, com as devidas correções supervisionadas e aprovadas pelo orientador. Cumpridas as formalidades de pauta, às 17 horas a presidência da mesa encerrou esta sessão de defesa de dissertação e para constar eu, Ana Maria Pereira Pinto, secretária do PPGM, lavrei a presente Ata que, depois de lida e aprovada, será assinada pelos membros da Banca Examinadora em quatro vias de igual teor.

Prof. Dr. Leandro da Fonseca Prudente
Presidente - IME/UFG

Prof. Dr. Luis Román Lucambio Pérez
Membro – IME/UFG

Prof. Dr. Max Leandro Nobre Gonçalves
Membro – IME/UFG

Prof. Dr. Douglas Soares Gonçalves
Membro – CFM/UFSC

DANILO RODRIGUES DE SOUZA

MÉTODO DO GRADIENTE CONJUGADO DAI-YUAN MODIFICADO PARA OTIMIZAÇÃO IRRESTRITA

Dissertação defendida no Programa de Pós-Graduação do Instituto de Matemática e Estatística da Universidade Federal de Goiás como requisito parcial para obtenção do título de Mestre em Matemática, aprovada no dia 28 de fevereiro de 2019, pela Banca Examinadora constituída pelos professores:

Prof. Dr. Leandro da Fonseca Prudente
Instituto de Matemática e Estatística - UFG
Presidente da Banca

Prof. Dr. Luis Román Lucambio Pérez
Instituto de Matemática e Estatística – UFG

Prof. Dr. Max Leandro Nobre Gonçalves
Instituto de Matemática e Estatística – UFG

Prof. Dr. Douglas Soares Gonçalves
Centro de Ciências Físicas e Matemáticas - UFSC

Todos os direitos reservados. É proibida a reprodução total ou parcial do trabalho sem autorização da universidade, do autor e do orientador(a).

Danilo Rodrigues de Souza

Bacharel em Matemática pela Universidade Federal de Goiás - UFG.

Dedico esse trabalho a Deus e minha família.

Agradecimentos

Agradeço primeiramente a Deus, pois Ele sempre esteve comigo por toda minha trajetória de vida, e nos momento mais difíceis dos estudos, Ele segurou em minha mão.

Agradeço aos meus pais, Waldir Rodrigues de Souza e Maria de Lurdes Rodrigues, meus pilares, que me deram condições de sempre estudar e me apoiaram. Aconselhando em todas as minhas decisões profissionais e pessoais. Agradeço à minha esposa, Gabriele Ferreira de Souza Mendes, que me ajudou a ter coragem de retomar o mestrado em 2017, me proibindo veementemente de desistir novamente. Agradeço ao meu irmão, Leandro Rodrigues de Souza, que por várias vezes me aconselhou, me ajudou naquilo que precisei, sendo um braço direito em minha vida.

Agradeço ao meu orientador, Leandro da Fonseca Prudente, pois quando pedi para que me orientasse, se dispôs de imediato. Também sou grato pois, semanalmente fui à sua sala para tirar dúvidas, e ele pacientemente me ajudou.

Agradeço aos os professores do IME, que durante minha formação acadêmica foram de suma importância. Um agradecimento especial aos professores, Glaydston de Carvalho Bento, Max Leandro Nobre Gonçalves e Lidiane dos Santos Monteiro Lima. Agradeço também aos meus colegas matemáticos, Ana Maria Alves, Ângelo Guimarães, Lucas Brito, Bruno Ragonete, Fábio Sodré, Heric Marques, Leonardo Pacheco, Susane Gontijo, Nathanni Vieira, Cícero Rumão, Dhyogo Nunes, Ilton Ferreira, Pedro Bonfin, Flávio Pinto, Nielson Honório, Ana Paula de Melo, Raquel Pereira e Robson Lousa; Que por diversas vezes me aconselharam, ensinaram e ajudaram de várias formas.

Quero aqui agradecer aos meus grandes amigos, Allyssom Marques e Clauciney Pereira, por estarem ao meu lado em tantos momentos da vida, meus padrinhos de casamento e companheiros de fé.

Por fim quero esboçar minha gratidão à CAPES e CNPq, por financiarem meus estudos e pesquisas.

"Se Deus é por nós, quem será contra nós?"

Aos Romanos , 8:31,
BÍBLIA SAGRADA.

Resumo

Souza, D.R.. **Método do gradiente conjugado Dai-Yuan modificado para otimização irrestrita**. Goiânia, 2019. 67p. Dissertação de Mestrado. Instituto de Matemática e Estatística, Universidade Federal de Goiás.

Os métodos de gradiente conjugado não-lineares são métodos de primeira ordem reconhecidamente eficientes para resolver problemas de otimização irrestritos. Em particular, o método de Dai–Yuan (DY), introduzido nos final dos anos 90, é um dos mais populares. O objetivo deste trabalho é investigar o desempenho numérico do método DY com uma modificação no parâmetro conjugado. A cada iteração desse método, é necessário computar um tamanho de passo satisfazendo as condições padrões de Wolfe. Assim, descreveremos o algoritmo de busca linear de Moré e Thuente. Sob hipóteses usuais, a convergência global do método de DY modificado será provada. Testes numéricos serão apresentados utilizando a biblioteca de problemas *CUTEst*.

Palavras–chave

Método do gradiente conjugado modificado; Otimização irrestrita; Condições de Wolfe; Algoritmo de busca linear; Análise numérica.

Abstract

Souza, D.R.. **Modified Dai-Yuan gradient method for unrestricted optimization**. Goiânia, 2019. 67p. MSc. Dissertation. Instituto de Matemática e Estatística, Universidade Federal de Goiás.

Nonlinear conjugate gradient methods are efficient first-order algorithms for solving unconstrained optimization problems. In particular, the Dai-Yuan (DY) method, introduced in the 1990s, is one of the most popular. The objective of the present work is to investigate the numerical performance of the DY method with a modification in the conjugate parameter. At each iteration of this method, it is necessary to compute a step size satisfying the standard Wolfe conditions. Thus, we will describe the line search algorithm of Moré and Thuente. Under usual assumptions, the global convergence of the modified DY method will be provided. Numerical tests will be presented using the *CUTEst* problem library.

Keywords

Modified conjugate gradient method; Unrestricted optimization; Wolfe Conditions; Line seanch algorithms; Numerical analysis.

Sumário

CAPÍTULO 1

Introdução

O método do gradiente conjugado (CG) é um esquema iterativo, proposto inicialmente, para resolver sistemas lineares

$$Ax = b,$$

em que A é uma matriz $n \times n$ simétrica e definida positiva e $b \in \mathbb{R}^n$. Resolver tal sistema linear é equivalente a minimizar a função quadrática, estritamente convexa, definida por

$$f(x) = \frac{1}{2}x^T Ax - b^T x.$$

De modo geral, o método CG gera uma sequência de iterados de acordo com

$$x_{k+1} = x_k + \alpha_k d_k, \quad k = 1, 2, \dots, \tag{1-1}$$

em que d_k é a direção de busca e α_k é o tamanho de passo. A direção de busca é definida por

$$d_k = \begin{cases} -\nabla f(x_k), & \text{se } k = 1, \\ -\nabla f(x_k) + \beta_{k-1} d_{k-1}, & \text{se } k > 1, \end{cases} \tag{1-2}$$

em que o parâmetro β_{k-1} é tomado de maneira que as direções consecutivas sejam A-conjugadas, ou seja, $d_k^T A d_{k+1} = 0$ para todo k. O tamanho de passo α_k é definido por meio de uma busca linear exata, ou seja,

$$\alpha_k = \text{argmin}_{\alpha > 0}\{f(x_k + \alpha d_k)\},$$

que, neste caso específico, possui *fórmula fechada*. É possível mostrar que o método CG encontra a solução do sistema linear em no máximo n iterações. Discutiremos essas propriedades no Capítulo 2.

Naturalmente, surgiram propostas para estender o método para funções

gerais, procurando resolver

$$\text{Minimizar} \quad f(x), \quad x \in \mathbb{R}^n,$$

em que $f\colon \mathbb{R}^n \to \mathbb{R}$ é continuamente diferenciável. O método geral preserva o esquema (1-1)-(1-2) de maneira que escolhas distintas para o parâmetro conjugado β_{k-1} conduz à métodos distintos. O primeiro método CG não-linear foi proposto por Fletcher-Reeves [8] na década de 60. Destacamos ainda os métodos Hestenes-Stiefel [11], Polak-Ribière-Polyak [16, 17], Conjugate Descent [7] e Dai-Yuan [5]. No caso geral, a busca linear exata, por envolver técnicas de otimização global, pode ser computacionalmente impraticável. Usualmente, pode-se mostrar convergência assumindo que o tamanho de passo cumpra condições bem menos exigentes, como por exemplo, as condições de Wolfe. No Capítulo 3, exibiremos as condições de Wolfe, bem como discutiremos o algoritmo de Moré e Thuente [13], que foi desenhado para obter de um tamanho de passo que satisfaça as condições fortes de Wolfe.

Em um trabalho recente, Lucambio Pérez e Prudente [12], estenderam os métodos de gradiente conjugado não-lineares para problemas de otimização vetorial. A convergência do método de Dai-Yuan (DY) (combinado com as condições fortes de Wolfe) foi provada com uma modificação no parâmetro conjugado. O método com o novo parâmetro ficou denominado *gradiente conjugado Dai-Yuan modificado*. O parâmetro conjugado do método modificado, no contexto escalar, é dado por

$$\beta_k^{DY} = \frac{\|\nabla f(x_{k+1})\|^2}{\nabla f(x_{k+1})^T d_k - \tau_k \nabla f(x_k)^T d_k},$$

em que $\tau_k > 1$. No caso escalar, o parâmetro acima com $\tau_k = 1$ recupera o parâmetro DY original. O objetivo deste trabalho é investigar as propriedades teóricas e práticas do método DY modificado no contexto de otimização escalar, de modo que os resultamos teóricos serão mostrados supondo $\tau_k \geq 1$.

No Capítulo 4, mostraremos a boa definição e estudaremos a convergência do método DY modificado. Em particular, a convergência global pode ser obtida supondo que os tamanhos de passo satisfazem as condições padrões de Wolfe. No Capítulo 5, exibiremos testes numéricos utilizando a biblioteca de problemas *CUTEst* [10], para ilustrar o comportamento do método com diversas escolhas do parâmetro τ_k.

CAPÍTULO 2

O método de gradiente conjugado linear

Neste capítulo vamos definir o método do gradiente conjugado linear, e discutir suas propriedades essenciais. O método GC linear, proposto por Hestenes–Stiefel [11], na década de 50, é um esquema iterativo para resolver sistemas lineares com matrizes associadas, definidas positivas. Esse método surgiu como uma alternativa para métodos diretos como, por exemplo, métodos que utilizam a fatoração de Cholesky. Métodos iterativos são, em geral, indicados para sistemas esparsos e de grande escala. O método de gradiente conjugado é um esquema iterativo para resolver sistemas lineares,

$$Ax = b, \tag{2-1}$$

em que A é uma matriz $n \times n$ simétrica e definida positiva, e $b \in \mathbb{R}^n$. Encontrar a solução do sistema (2-1), equivale a resolver o problema de otimização,

$$\text{Minimizar} \quad f(x), \quad x \in \mathbb{R}^n,$$

em que

$$\begin{array}{rccl} f: & \mathbb{R}^n & \to & \mathbb{R} \\ & x & \mapsto & \dfrac{1}{2}x^T Ax - b^T x. \end{array} \tag{2-2}$$

A propriedade de A ser simétrica e definida positiva garante que f é estritamente convexa e, consequente, o único ponto estacionário x^* é minimizador global de f.

Métodos de GC lineares, usam apenas informações da função objetivo e do seu gradiente. A convergência desse método é, em geral, mais rápida que do método de Cauchy e com custo computacional menor do que o método de Newton. Vamos mostrar que o método GC linear, minimiza a função quadrática (2-2), em no máximo n passos.

2.1 Busca exata e direções conjugadas

O método GC linear é um método iterativo de descida. Desse modo, dado um iterado x_k, o método computa uma direção d_k que decresce f a partir de x_k e calcula um tamanho passo α_k, obtendo um novo ponto x_{k+1}. Em síntese,

$$x_{k+1} = x_k + \alpha_k d_k, \quad k = 1, 2, \dots . \tag{2-3}$$

Nesta seção mostraremos como o método GC linear computa o comprimento do passo α_k e como determina as direções de busca d_k. Para determinar o comprimento de passo a cada iteração, vamos fixar $x_k \in \mathbb{R}^n$ e d_k uma direção de descida. O passo é calculado por:

$$\alpha_k = \operatorname{argmin}_{\alpha>0}\{f(x_k + \alpha d_k)\}.$$

Seja

$$\begin{array}{rccc} \varphi: & \mathbb{R} & \to & \mathbb{R} \\ & \alpha & \mapsto & f(x_k + \alpha d_k). \end{array}$$

Como f é estritamente convexa, φ também é. Assim, α_k é o único ponto estacionário de φ. Logo, estamos interessados em obter α_k tal que $\varphi'(\alpha_k) = 0$. Como

$$\varphi'(\alpha_k) = \nabla f(x_k + \alpha_k d_k)^T d_k = \nabla f(x_{k+1})^T d_k,$$

então

$$\nabla f(x_{k+1})^T d_k = 0. \tag{2-4}$$

Pela definição de f em (2-2),

$$\begin{aligned} \nabla f(x_k + \alpha_k d_k) &= A(x_k + \alpha_k d_k) - b \\ &= Ax_k - b + \alpha_k A d_k. \end{aligned}$$

Logo,

$$\nabla f(x_{k+1}) = \nabla f(x_k) + \alpha_k A d_k. \tag{2-5}$$

Calculando o produto interno de (2-5) com d_k, obtemos:

$$\nabla f(x_{k+1})^T d_k = \nabla f(x_k)^T d_k + \alpha_k (Ad_k)^T d_k.$$

Por (2-4), $\nabla f(x_k)^T d_k + \alpha_k (Ad_k)^T d_k = 0$ e, portanto,

$$\alpha_k = \frac{-\nabla f(x_k)^T d_k}{d_k^T A d_k}. \tag{2-6}$$

As direções do método GC seguem duas exigências: serem de descida e A-conjugadas. Vamos definir direções A-conjugadas e mostrar duas propriedades importantes dessas direções.

Definição 2.1 *Seja $A \in \mathbb{R}^{n\times n}$ uma matriz definida positiva. Dizemos que as direções $d_1, \ldots, d_k \in \mathbb{R}^n \setminus \{0\}$ são A-conjugadas se*

$$d_i^T A d_j = 0,$$

$\forall\, i, j = 1, \ldots, k$ *com* $i \neq j$.

Observação 2.1 *Se A é a matriz identidade, então os vetores A-conjugados são ortogonais.*

Mostraremos a seguir que um conjunto de direções A-conjugadas é linearmente independente.

Lema 2.1 *[19, Lema 5.13] Seja $A \in \mathbb{R}^{n\times n}$ uma matriz simétrica e definida positiva. Um conjunto qualquer de vetores A-conjugados é linearmente independente.*

Demonstração. Sejam $d_1, \ldots, d_k \in \mathbb{R}^n \setminus \{0\}$ direções A-conjugadas. Considere constantes $a_1, \ldots, a_k \in \mathbb{R}$ tais que:

$$a_1 d_1 + \cdots + a_k d_k = 0.$$

Agora, $\forall i \in \{1, \ldots, k\}$, podemos fazer o produto do vetor $d_i^T A$, com a combinação acima. Assim,

$$a_1 d_i^T A d_1 + \cdots + a_i d_i^T A d_i + \cdots + a_k d_i^T A d_k = 0. \tag{2-7}$$

Como as direções são A-conjugadas, então $d_i^T A d_j = 0$ para $i \neq j$. Consequentemente, da combinação linear (2-7), temos:

$$a_i d_i^T A d_i = 0.$$

Como $d_i^T A d_i \neq 0$, pois A é definida positiva, temos $a_i = 0$. Logo $a_i = 0\ \forall i \in \{1, \ldots, k\}$.

■

O lema a seguir é uma propriedade fundamental para a eficácia do método.

Lema 2.2 *[19, Lema 5.15] Sejam $d_1, \ldots, d_k$ direções A-conjugadas, conforme a Definição 2.1. Dado $x_1 \in \mathbb{R}^n$, considere a sequência finita definida em (2-3), de modo que α_k é dado por (2-6). Então*

$$\nabla f(x_{k+1})^T d_j = 0,$$

para $j = 1, \ldots, k$.

Demonstração. Por (2-4), temos $\nabla f(x_{k+1})^T d_k = 0$. Assim se $j = k$, é trivial. Agora suponha que $j < k$. Note que

$$\begin{aligned} \nabla f(x_{k+1})^T d_j &= (A(x_k + \alpha_k d_k) - b)^T d_j = (Ax_k - b + \alpha_k A d_k)^T d_j \\ &= \nabla f(x_k)^T d_j + \alpha_k d_k^T A d_j = \nabla f(x_k)^T d_j, \end{aligned} \tag{2-8}$$

pois as direções são A-conjugadas, ou seja, $\alpha_k d_k^T A d_j = 0$. Como (2-4) vale para $\forall k \in \{1, \ldots, n-1\}$, se $j = k-1$, então a relação acima garante que

$$\nabla f(x_{k+1})^T d_j = \nabla f(x_k)^T d_j = \nabla f(x_k)^T d_{k-1} = 0.$$

Se, $j < k-1$, então aplicando (2-8) recorrentemente, temos

$$\nabla f(x_{k+1})^T d_j = \nabla f(x_k)^T d_j = \cdots = \nabla f(x_{j+1})^T d_j = 0,$$

donde a última igualdade segue de (2-4)

■

Para que o método gere direções d_k que sejam A-conjugadas, em particular, tomaremos combinações das direções $-\nabla f(x_k)$ e d_{k-1}. Iniciando com a direção $d_1 = -\nabla f(x_1)$, as direções subsequentes devem ser tais que

$$d_k^T A d_{k-1} = 0, \ \forall k > 1.$$

Considere $g_k = \nabla f(x_k)$. As direções são definidas por:

$$d_k = \begin{cases} -g_k, & \text{se } k = 1, \\ -g_k + \beta_{k-1} d_{k-1}, & \text{se } k > 1, \end{cases} \tag{2-9}$$

em que β_{k-1} é um parâmetro algorítmico que garante que direções consecutivas, sejam A-conjugadas. Seja $k > 1$. Fazendo o produto da direção d_k por $d_{k-1}^T A$, obtemos:

$$d_{k-1}^T A d_k = d_{k-1}^T A(-g_k) + \beta_{k-1} d_{k-1}^T A d_{k-1}.$$

Como estamos exigindo que d_k seja A-conjugada com d_{k-1}, então $d_{k-1}^T A d_k = 0$. Consequentemente,

$$\beta_{k-1} = \frac{d_{k-1}^T A g_k}{d_{k-1}^T A d_{k-1}}. \tag{2-10}$$

A definição de β_{k-1} em (2-10) também garante que as direções geradas por (2-9), são de descida. De fato, se $\|g_k\| \neq 0$, então

$$\begin{aligned} g_k^T d_k &= g_k^T(-g_k + \beta_{k-1} d_{k-1}) = \\ &= -\|g_k\|^2 + \beta_{k-1} g_k^T d_{k-1} \\ &= -\|g_k\|^2, \end{aligned}$$

pois (2-4) nos dá $g_k^T d_{k-1} = 0$.

2.2 Esquema geral do método

Nesta seção trataremos do esquema iterativo do método de gradiente conjugados linear, juntamente com suas principais propriedades.

Algoritmo 1 *Seja $x_1 \in \mathbb{R}^n$. Calcule $g_1 = \nabla f(x_1)$ e inicie $k \leftarrow 1$.*

Passo 1 *Se $g_k = 0$, então PARE.*

Passo 2 *Defina:*

$$d_k = \begin{cases} -g_k, & \text{se } k = 1, \\ -g_k + \beta_{k-1} d_{k-1}, & \text{se } k > 1, \end{cases}$$

de modo que $\beta_{k-1} = \dfrac{d_{k-1}^T A g_k}{d_{k-1}^T A d_{k-1}}$.

Passo 3 *Faça $x_{k+1} = x_k + \alpha_k d_k$, de modo que* $\alpha_k = \dfrac{-g_k^T d_k}{d_k^T A d_k}$.

Passo 4 *Calcule g_{k+1}, faça $k \leftarrow k+1$ e vá ao **Passo 1**.*

O resultado a seguir mostra que as direções geradas pelo Algoritmo 1, são A-conjugadas.

Teorema 2.2 *[19, Teorema 5.17] Se x_k e d_k foram geradas pelo Algoritmo 1, então para todo $j = 1, \ldots, k-1$ temos,*

$$g_k^T g_j = 0 \;\; e \;\; d_k^T A d_j = 0.$$

Demonstração. Vamos usar indução sobre k para mostrar esse resultado. Seja $k = 1$. A condição (2-4) garante $g_2^T g_1 = -g_2^T d_1 = 0$. Além disso, as direções d_1 e d_2 são A-conjugadas, logo $d_2^T A d_1 = 0$. Suponha agora que o resultado vale para $k > 1$, vamos provar para $k+1$. A hipótese de indução garante que as

direções $d_1, \ldots, d_k$ são A-conjugadas. Consequentemente, o Lema 2.2 mostra que $g_{k+1}^T d_j = 0, \forall\ j = 1, \ldots, k$. Usando (2-9), para $j = 1, \ldots, k$, obtemos:

$$g_{k+1}^T g_j = g_{k+1}^T \left(-d_j + \beta_{j-1} d_{j-1}\right) = -g_{k+1}^T d_j + \beta_{j-1} g_{k+1}^T d_{j-1} = 0. \tag{2-11}$$

A definição de β_k nos dá $d_{k+1}^T A d_k = 0$. Para $j < k$, a hipótese de indução garante

$$d_{k+1}^T A d_j = \left(-g_{k+1} + \beta_k d_k\right)^T A d_j = -g_{k+1}^T A d_j + \beta_k d_k^T A d_j = -g_{k+1}^T A d_j.$$

Por (2-5) e (2-11), concluímos que

$$d_{k+1}^T A d_j = -g_{k+1}^T A d_j = -g_{k+1}^T \left(\frac{g_{j+1} - g_j}{\alpha_j}\right) = 0.$$

■

O resultado a seguir, garante a convergência do método para x^*, minimizador da quadrática, em no máximo n passos.

Teorema 2.3 *[19, Teorema 5.14] Considere a função quadrática dada por (2-2) e seu minimizador x^*. Dado $x_1 \in \mathbb{R}^n$, a sequência finita de iterados, gerada pelo Algoritmo 1, cumpre $x_{n+1} = x^*$.*

Demonstração. Pelo Teorema 2.2 e Lema 2.1, $\{d_1, \ldots, d_n\}$ é uma base de $\mathbb{R}^n$. Assim existem $t_i \in \mathbb{R}$, para $i = 1, \ldots, n$ tais que:

$$x^* - x_1 = \sum_{i=1}^{n} t_i d_i. \tag{2-12}$$

Tome $k \in \{1, \ldots, n\}$. Fazendo o produto de $d_k^T A$ com (2-12), obtemos:

$$d_k^T A(x^* - x_1) = \sum_{i=1}^{n} t_i d_k^T A d_i = t_k d_k^T A d_k.$$

Logo

$$t_k = \frac{d_k^T A(x^* - x_1)}{d_k^T A d_k}. \tag{2-13}$$

Por outro lado,

$$x_k = x_{k-1} + \alpha_{k-1} d_{k-1} = x_{k-2} + \alpha_{k-2} d_{k-2} + \alpha_{k-1} d_{k-1},$$

e, recursivamente, temos

$$x_k = x_1 + \alpha_1 d_1 + \cdots + \alpha_{k-1} d_{k-1}. \tag{2-14}$$

Fazendo o produto de (2-14) com $d_k^T A$, obtemos:

$$d_k^T A x_k = d_k^T A x_1. \tag{2-15}$$

Como x^* é o minimizador da f, então

$$\nabla f(x^*) = 0 \Rightarrow Ax^* = b. \tag{2-16}$$

Substituindo (2-15) e (2-16) em (2-13), para todo $k \in \{1, \ldots, n\}$, temos:

$$t_k = \frac{d_k^T A x^* - d_k^T A x_1}{d_k^T A d_k} = \frac{d_k^T b - d_k^T A x_k}{d_k^T A d_k} = \frac{-d_k^T (A x_k - b)}{d_k^T A d_k} = \frac{-\nabla f(x_k)^T d_k}{d_k^T A d_k} = \alpha_k.$$

Logo, por (2-12), obtemos:

$$x^* - x_1 = \sum_{i=1}^{n} t_i d_i = \sum_{i=1}^{n} \alpha_i d_i$$

e, portanto,

$$x^* = x_1 + \sum_{i=1}^{n} \alpha_i d_i = x_{n+1}.$$

■

O resultado subsequente mostra que o Algoritmo 1 minimiza f em subespaços gerados pelas direções A-conjugadas. Na prática, isso também garante o término em n passos. De fato, o Lema 2.1 garante que as direções A-conjugadas formam uma base L.I. Assim, no n-ésimo passo, o método encontrará o minimizador de f em todo $\mathbb{R}^n$.

Teorema 2.4 *[19, Teorema 5.16] Dado $x_1 \in \mathbb{R}^n$, considere a sequência finita gerada pelo Algoritmo 1. O ponto x_{k+1} minimiza f sobre subespaço $S = x_1 + Spam[d_1, \ldots, d_k]$, com $k = 1, \ldots, n$.*

Demonstração. Note que

$$x_{k+1} = x_k + \alpha_k d_k = \cdots = x_1 + \alpha_1 d_1 + \alpha_2 d_2 + \cdots + \alpha_k d_k.$$

Assim $x_{k+1} \in S$. Seja v um elemento de S, logo

$$v - x_{k+1} \in \text{Spam}[d_1, \ldots, d_k].$$

Pelo Lema 2.2,

$$\nabla f(x_{k+1})^T (v - x_{k+1}) = 0. \tag{2-17}$$

Como f é convexa, a condição (2-17) é suficiente para garantir que x_{k+1} minimiza f em S.

■

Vale ressaltar que a terminação em n passos pode ser vista de outro modo. Se o Algoritmo 1 gera x_{n+1} após n passos, então o Teorema 2.2 garante que os gradientes $\nabla f(x_i)$ formam uma base ortogonal de $\mathbb{R}^n$. Como

$$\nabla f(x_{n+1})^T \nabla f(x_j) = 0,$$

pelo mesmo teorema, então

$$\nabla f(x_{n+1}) = 0.$$

Pois, caso contrário, teríamos $n+1$ vetores formando uma base ortogonal de $\mathbb{R}^n$.

Nos próximos capítulos, trataremos da generalização do método GC para funções gerais. Nesse caso, perderemos a noção de direções conjugadas e também será necessário o estudo de algoritmos de busca linear.

CAPÍTULO 3

Busca linear

Algoritmos de busca linear são importantes ferramentas para globalizar métodos de descida do tipo

$$x_{k+1} = x_k + \alpha_k d_k, \tag{3-1}$$

em que d_k é uma direção de descida a partir de x_k. Idealmente, a cada iteração, o tamanho de passo deve ser calculado como

$$\alpha_k = \text{argmin}_{\alpha \geq 0} f(x_k + \alpha d_k),$$

ou seja, minimizar f a partir de x_k ao longo de d_k. Por envolver técnicas de otimização global, tal estratégia é inviável na prática quando f não é convexa. Para funções gerais a expressão "busca linear exata", significa obter α_k como um ponto estacionário do problema de minimizar f em x_k ao longo de d_k, implicando que

$$\nabla f(x_k + \alpha_k d_k)^T d_k = 0.$$

Entretanto, mesmo a exigência de se obter pontos estacionários, usualmente resulta em uma busca linear cara e impraticável. Condições bem menos exigentes podem ser combinadas com métodos específicos para obter convergência global. Usualmente, para métodos do tipo GC, exige-se que o tamanho de passo satisfaça as condições de Wolfe. Essas condições associadas ao lema de Zoutendijk, que apresentaremos nesse capítulo, podem garantir a convergência de métodos de descida.

Na Seção 3.1 desse capítulo, traremos as condições de Wolfe, juntamente com a existência de passos satisfazendo tais condições. A convergência de métodos de descida também será discutida nessa seção. Na Seção 3.2, descreveremos o método proposto por Moré–Thuente [13] para obter passos satisfazendo as condições fortes de Wolfe.

3.1 Condições de Wolfe

Fixando o ponto $x_k \in \mathbb{R}^n$ e a direção descida d_k a partir de x_k, temos $\nabla f(x_k)^T d_k < 0$. Definindo

$$\begin{array}{rccc} \varphi: & \mathbb{R} & \to & \mathbb{R} \\ & \alpha & \mapsto & f(x_k + \alpha d_k). \end{array}$$

A primeira condição, conhecida como decréscimo suficiente, exige que α seja tal que

$$f(x_k + \alpha d_k) \leq f(x_k) + \delta\alpha\nabla f(x_k)^T d_k := l(\alpha), \tag{3-2}$$

ou equivalentemente,

$$\varphi(\alpha) \leq \varphi(0) + \delta\varphi'(0)\alpha, \tag{3-3}$$

tal que $0 < \delta < 1$. Essa desigualdade também é chamada de *Condição de Armijo*. Geometricamente, vide Figura 3.1, os passos admissíveis devem ser tais que o gráfico $\varphi(\alpha)$ deve estar abaixo da reta $l(\alpha)$.

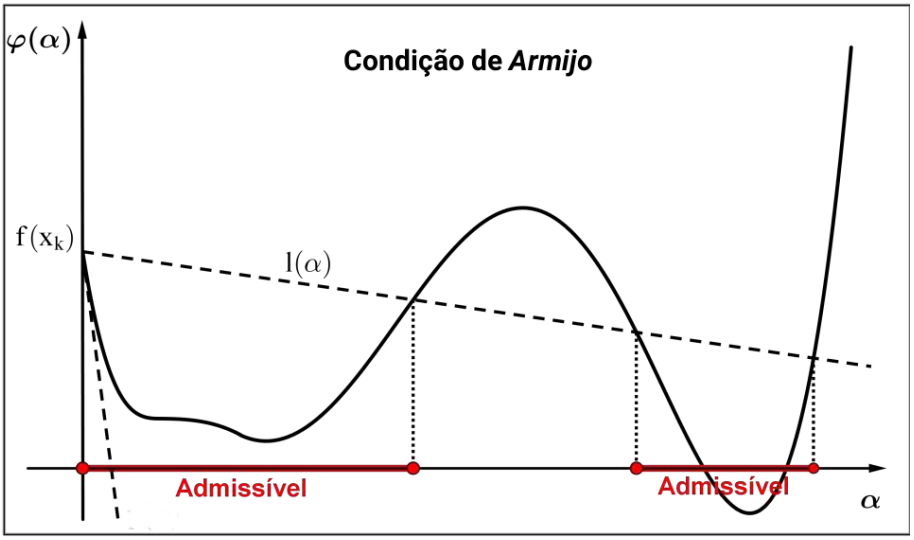

Figura 3.1: *Retirada de [15]*

Se o α escolhido for muito pequeno a condição (3-2) será satisfeita, porém isso é obviamente indesejável. Uma alternativa para evitar tais escolhas é fazer uma exigência adicional, conhecida como condição de curvatura, exigindo que

$$\nabla f(x_k + \alpha d_k)^T d_k \geq \sigma\nabla f(x_k)^T d_k, \tag{3-4}$$

em que $0 < \sigma < 1$. Deste modo, valores de α próximos a zero serão descartados pela condição de curvatura. Quando $0 < \delta < \sigma < 1$ as desigualdades (3-2) e (3-4), combinadas, compõe as *Condições padrões de Wolfe*, que podem ser vistas geometricamente na Figura 3.2. Além de evitar passos excessivamente pequenos, as condições padrões de Wolfe sempre induzem a escolha de passos maiores. Passos com curvaturas negativas indicam que podemos decrescer ainda mais o valor de φ com passos maiores. A escolha de passos grandes são desejáveis para métodos de descida, pois isso reduz o custo computacional.

A condição (3-4) restringe o quanto a curvatura pode ser negativa, mas não o quão ela pode ser positiva, o que pode ser alcançado exigindo que,

$$|\nabla f(x_k + \alpha d_k)^T d_k| \leq -\sigma \nabla f(x_k)^T d_k, \tag{3-5}$$

ou em termos de φ,

$$|\varphi'(\alpha)| \leq \sigma |\varphi'(0)|, \tag{3-6}$$

de modo que $0 < \sigma < 1$. Unindo as condições (3-2) e (3-5), quando $0 < \delta < \sigma < 1$, temos as *condições fortes de Wolfe* e geometricamente podem ser vistas na Figura 3.3. Se um passo satisfaz as condições fortes de Wolfe então ele satisfaz as padrões. Assim além de evitar passos excessivamente pequenos e induzir a escolha de passos grandes, ao restringir o quanto a curvatura pode ser positiva estamos exigindo que o passo esteja próximo a um ponto estacionário.

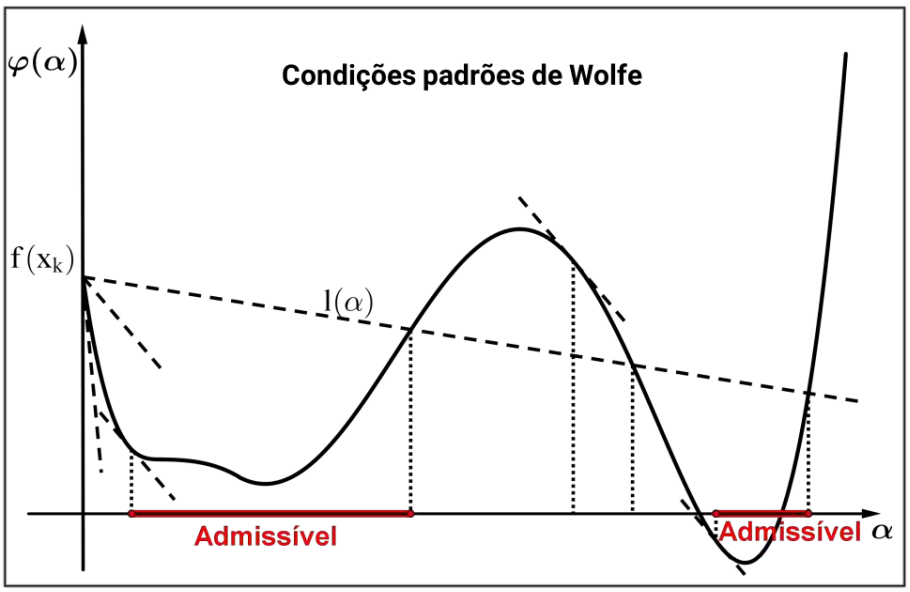

Figura 3.2: *Retirada de [15]*

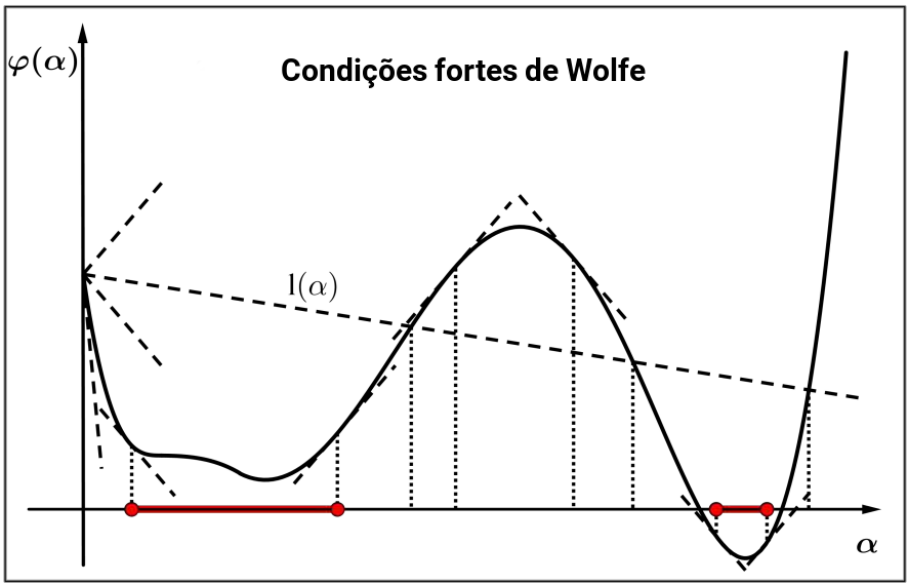

Figura 3.3: *Retirada de [15]*

A fim de mostrar a existência de um α que satisfaça as condições fortes de Wolfe, é necessário assumir algumas hipóteses sobre a função objetivo e seu gradiente.

Hipóteses 1

(i) Suponha que f é limitada inferiormente em $\mathbb{R}^n$, e é continuamente diferenciável em uma vizinhança $\mathcal{N}$ do conjunto de nível $\mathcal{L} = \{x \in \mathbb{R}^n : f(x) \leq f(x_1)\}$.

(ii) O gradiente $\nabla f(x)$ é Lipschitz contínuo em $\mathcal{N}$, isto é, existe uma constante $L > 0$ tal que

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$$

$\forall x, y \in \mathcal{N}$.

Com tais hipóteses, podemos provar a existência de um intervalo com passos satisfazendo as condições fortes de Wolfe, e consequentemente as condições padrões.

Teorema 3.1 *[14, Lema 3.1] Seja $f\colon \mathbb{R}^n \to \mathbb{R}$ continuamente diferenciável, d_k uma direção de descida em x_k e suponha f limitada inferiormente ao longo de $\{x_k + \alpha d_k | \alpha > 0\}$. Se $0 < \delta < \sigma < 1$, então existe intervalo em que os comprimentos de passo contidos nele, satisfazem as condições fortes de Wolfe.*

Demonstração. Por hipótese, $\varphi(\alpha) = f(x_k + \alpha d_k)$ é limitada inferiormente para todo $\alpha > 0$. Note que o coeficiente da reta $l(\alpha)$, definida em (3-2), é negativo e, portanto, essa reta é ilimitada inferiormente. Consequentemente $l(\alpha)$ intercepta

o gráfico de φ em pelo menos um ponto. Seja $\alpha' > 0$ a menor imagem inversa dos pontos de intersecção, isto é,

$$\varphi(\alpha') = f(x_k + \alpha' d_k) = f(x_k) + \alpha' \delta \nabla f(x_k)^T d_k = l(\alpha). \tag{3-7}$$

Assim todos os comprimentos de passo menores que α', satisfazem a condição (3-2). Como f é diferenciável, o teorema do valor médio, aplicado à φ, garante a existência de $\alpha'' \in (0, \alpha')$ tal que

$$\frac{f(x_k + \alpha' d_k) - f_k}{\alpha'} = \nabla f(x_k + \alpha'' d_k)^T d_k,$$

agora, utilizando (3-7), temos

$$\alpha' \delta \nabla f_k^T d_k = f(x_k + \alpha' d_k) - f_k = \alpha' \nabla f(x_k + \alpha'' d_k)^T d_k.$$

Logo

$$|\nabla f(x_k + \alpha'' d_k)^T d_k| = -\delta \nabla f_k^T d_k < -\sigma \nabla f_k^T d_k$$

pois $\nabla f_k^T d_k < 0$ e $\delta < \sigma$. Consequentemente α'' satisfaz (3-2) e (3-5). Como f é contínua, existe um intervalo centrado em α'' no qual seus pontos satisfazem as condições fortes de Wolfe.

■

Para que possamos obter convergência global de um método de descida do tipo (3-1), devemos ter não apenas comprimentos de passo bem escolhidos, mas também direções de busca d_k bem escolhidas. Para tais direções, devemos nos ater ao cosseno do ângulo θ_k entre d_k e $-\nabla f(x_k)$, definido por,

$$\cos\theta_k = \frac{-\nabla f(x_k)^T d_k}{\|\nabla f(x_k)\| \|d_k\|}. \tag{3-8}$$

O resultado a seguir, proposto por Zoutendijk, possui amplas consequências. Mostra, por exemplo, que o método do gradiente é globalmente convergente, usando passos que satisfazem Wolfe. Para outros métodos, como GC, mostra o quanto a direção de descida d_k pode se distanciar de $-\nabla f(x_k)$ e mesmo assim ser globalmente convergente. Outras condições podem ser usadas para os comprimentos de passo, porém faremos o uso somente das condições de Wolfe.

Lema 3.1 *[5, Lema 3.2, Zoutendijk] Supondo que x_1 é um ponto inicial que satisfaz as Hipóteses 1. Considere algum método da forma (3-1), na qual d_k é uma direção de descida*

e α_k satisfaz as condições padrões de Wolfe. Assim temos:

$$\sum_{k\geq 1} \frac{(g_k^T d_k)^2}{\|d_k\|^2} < \infty, \tag{3-9}$$

ou equivalentemente,

$$\sum_{k\geq 1} \cos^2\theta_k \|g_k\|^2 < \infty.$$

Demonstração. Para uma iteração k-ésima qualquer, pela condição de curvatura (3-4) temos,

$$g_{k+1}^T d_k - g_k^T d_k \geq \sigma g_k^T d_k - g_k^T d_k = (\sigma - 1) g_k^T d_k.$$

Usando a desigualdade de Schwarz no lado esquerdo da inequação anterior e sabendo que o gradiente ∇f é Lipschitz contínuo, obtemos:

$$(g_{k+1} - g_k)^T d_k \leq \|g_{k+1} - g_k\| \|d_k\| \leq \alpha L \|d_k\|^2,$$

e assim,

$$\alpha \geq \frac{(g_{k+1} - g_k)^T d_k}{L\|d_k\|^2} \geq \frac{(\sigma - 1) g_k^T d_k}{L\|d_k\|^2}.$$

Pela condição de decréscimo suficiente (3-2),

$$f_k - f_{k+1} \geq -\delta\alpha g_k^T d_k \geq \frac{-\delta(\sigma - 1)}{L} \frac{(g_k^T d_k)^2}{\|d_k\|^2},$$

se fizermos $c = \dfrac{\delta(1-\sigma)}{L}$, então

$$f_{k+1} \leq f_k - c\frac{(g_k^T d_k)^2}{\|d_k\|^2}.$$

De maneira recursiva aplicando a relação acima em $f_k, f_{k-1}, \ldots, f_2$, temos:

$$f_{k+1} \leq f_k - c\frac{(g_k^T d_k)^2}{\|d_k\|^2} \leq f_{k-1} - c\frac{(g_{k-1}^T d_{k-1})^2}{\|d_{k-1}\|^2} - c\frac{(g_k^T d_k)^2}{\|d_k\|^2} \leq \cdots \leq f_1 - c\sum_{j=1}^{k} \frac{(g_j^T d_j)^2}{\|d_j\|^2}.$$

Portanto, como f é limitada inferiormente, existe uma constante M tal que $-f < -M$ e assim temos

$$c\sum_{j=1}^{k} \frac{(g_j^T d_j)^2}{\|d_j\|^2} \leq f_1 - f_{k+1} \leq f_1 - M,$$

concluindo a demonstração.

■

De modo similar o resultado pode ser demonstrado exigindo as condições fortes de Wolfe. Vamos denominar a inequação (3-9) por *condição de Zoutendijk*.

A condição de Zoutendijk (3-9) implica que

$$\lim_{k\to\infty} \cos^2\theta_k \|g_k\|^2 = 0.$$

Se o método que estivermos tratando garantir que o ângulo θ_k definido em (3-8) é limitado a partir de 90°, ou seja, se existe uma constante $\lambda > 0$ tal que

$$\cos\theta_k \geq \lambda > 0 \; \forall k,$$

então teríamos imediatamente que

$$\lim_{k\to\infty} \|\nabla f(x_k)\| = 0.$$

Desse modo provamos que um método de descida é convergente, desde que a direção de busca nunca esteja muito próxima a ortogonalidade com a direção do gradiente e o passo satisfaça as condições de Wolfe. No Capítulo 4 faremos o uso do condição de Zoutendijk para mostrar que o método GC Dai-Yuan modificado converge.

3.2 O algoritmo de busca de Moré e Thuente

A eficiência de métodos de descidas dependem, muitas vezes, da formulação de eficientes métodos de busca linear. O algoritmo proposto por Moré em Thuente em [13] é bastante conhecido, bem testado e usado em diversos trabalhos, como por Zhu–Byrd–Nocedal [24]. Vamos tratar nessa seção do algoritmo proposto por Moré e Thuente, que visa encontrar um passo que satisfaz as condições fortes de Wolfe.

Os resultados que exibiremos na Subseção 3.2.1, mostram que o algoritmo de busca pode ser usado para encontrar um α satisfazendo (3-3) e (3-6) com $\delta \leq \sigma$. Na Subseção 3.2.2, discutiremos estratégias para buscar um minimizador local da função φ. Por fim, detalhes técnicos da escolha de passos testes, através de interpolações de cúbicas e quadráticas, serão discutidos na Subseção 3.2.3.

Para o algoritmo que apresentaremos nessa seção, vamos supor φ, que foi definida na Seção 3.1, continuamente diferenciável. Como a direção d_k é de descida, então $\varphi'(0) < 0$. Nosso objetivo é encontrar $\alpha > 0$ tal que as condições (3-3) e (3-6) sejam satisfeitas. Uma exigência natural é pedir que α satisfaça os

limites

$$0 \leq \alpha_{min} \leq \alpha \leq \alpha_{max},$$

em que α_{min} e α_{max} são parâmetros algorítmicos. O limite inferior é usado devido aspectos técnicos do algoritmo, enquanto o superior é necessário para garantir o término finito na presença de uma função φ limitada inferiormente.

O problema principal que será tratado nessa seção é encontrar um passo α aceitável que pertença ao conjunto

$$T(\delta) \equiv \{\alpha > 0 : \varphi(\alpha) \leq \varphi(0) + \delta\varphi'(0)\alpha, \ \ |\varphi'(\alpha)| \leq \delta|\varphi'(0)|\}.$$

A formulação dos resultados em termos de $T(\delta)$ tem algumas vantagens, a primeira é que o algoritmo de busca não se torna dependente de σ e a segunda é que $T(\delta)$ é não vazio quando φ é limitada inferiormente.

3.2.1 Esquema geral

Para encontrar um α em $T(\delta)$, vamos assumir que φ é continuamente diferenciável em $[0, \alpha_{max}]$, que $\varphi'(0) < 0$ e $\delta \in (0,1)$. Várias buscas lineares assumem que $\delta < \frac{1}{2}$, pois se φ é uma quadrática, $a\alpha^2 + b\alpha + c$, com concavidade para cima, $\varphi''(0) > 0$ e $b > 0$, então o minimizador global é $\alpha^* = -\frac{b}{2a}$ e assim satisfaz

$$\varphi(\alpha^*) = a\left(-\frac{b}{2a}\right)^2 + b\left(-\frac{b}{2a}\right) + c = \frac{b^2}{4a} - \frac{b^2}{2a} + \varphi(0)$$

$$= \varphi(0) - \frac{b^2}{4a} = \varphi(0) + \alpha^* b\frac{1}{2},$$

logo

$$\varphi(\alpha^*) = \varphi(0) + \frac{1}{2}\alpha^*\varphi'(0) \leq \varphi(0) + \delta\alpha^*\varphi'(0),$$

pois $\varphi'(0) = b$ e $\varphi(0) = c$. Assim α^* satisfaz (3-3) e (3-6).

Dado um $\alpha_0 \in [\alpha_{min}, \alpha_{max}]$, o algoritmo de busca gera uma sequência de intervalos I_k e de iterados $\alpha_k \in I_k \cap [\alpha_{min}, \alpha_{max}]$, da seguinte maneira:

Algoritmo de Busca. Seja $I_0 = [0, \infty]$.

Para $k = 0, 1, ...$

Escolha um passo salva-guardado $\alpha_k \in I_k \cap [\alpha_{min}, \alpha_{max}]$.

Teste para convergência.

Atualize o intervalo I_k.

As salva-guardas impostas ao passo α_k serão importantes para garantir a convergência do algoritmo. Essa regras serão discutidas ao longo da subseção em momentos oportunos. A seguir vamos tratar da atualização do I_k.

O intuito da atualização dos intervalos é encontrar um I_k tal que $T(\delta) \cap I_k \neq \emptyset$, e refinar os intervalos de modo que a interseção continue não vazia. Para tal, vamos descrever condições nos pontos da fronteira de um intervalo I que vão garantir a interseção não vazia. Essas condições nos pontos da fronteira são dadas em termos da função auxiliar ψ definida por:

$$\psi(\alpha) \equiv \varphi(\alpha) - \varphi(0) - \delta\varphi'(0)\alpha,$$

e sua derivada:

$$\psi'(\alpha) = \varphi'(\alpha) - \delta\varphi'(0).$$

Teorema 3.2 *Seja I um intervalo fechado com pontos de fronteira α_l e α_u. Se esses pontos satisfazem*

$$\psi(\alpha_l) \leq \psi(\alpha_u), \;\; \psi(\alpha_l) \leq 0 \; e \;\; \psi'(\alpha_l)(\alpha_u - \alpha_l) < 0, \tag{3-10}$$

então existe um $\alpha^* \in I$ com $\psi(\alpha^*) \leq \psi(\alpha_l)$ e $\psi'(\alpha^*) = 0$. Em particular, $\alpha^* \in (T(\delta) \cap I)$.

Demonstração. Observe que α_l e α_u não são ordenados. Suponha, sem perda de generalidade, que $\alpha_l < \alpha_u$. Defina

$$\alpha_m = Sup\{\alpha \in [\alpha_l, \alpha_u] : \psi(\beta) \leq 0, \; \beta \in [\alpha_l, \alpha]\}.$$

Como $\psi'(\alpha_l) < 0$, então $\alpha_l < \alpha_m$. Afirmamos que $\psi(\alpha_l) \leq \psi(\alpha_m)$. De fato, se $\alpha_m = \alpha_u$, então a hipótese garante $\psi(\alpha_l) \leq \psi(\alpha_m)$. Se $\alpha_m < \alpha_u$ então a definição de α_m implica que $\psi(\alpha_m) = 0$ e, por hipótese, temos $\psi(\alpha_l) \leq 0 = \psi(\alpha_m)$.

Sendo ψ contínua e $[\alpha_l, \alpha_m]$ compacto, garantimos a existência de um minimizador global α^* e afirmamos que ele está no interior do intervalo. De fato, $\alpha^* \neq \alpha_l$ pois a hipótese garante que $\psi'(\alpha_l) < 0$ e isso implica que ψ decresce numa vizinhança de α_l, logo α^* não poderia ser minimizador global. Agora, $\alpha^* \neq \alpha_m$ pois $\psi(\alpha_l) \leq \psi(\alpha_m)$. Assim $\psi'(\alpha^*) = 0$ e isso implica,

$$0 = \psi'(\alpha^*) = \varphi'(\alpha^*) - \delta\varphi'(0) \Rightarrow |\varphi'(\alpha^*)| = \delta|\varphi'(0)|.$$

Como $\psi(\alpha) \leq 0 \; \forall \; \alpha \in [\alpha_l, \alpha_m]$, então α^* satisfaz (3-3) e, consequentemente, $\alpha^* \in T(\delta)$.

■

O Teorema 3.2 mostra que se os pontos de fronteira de I satisfazem (3-10), então o algoritmo de busca pode ser visto como um procedimento para encontrar

um minimizador para ψ. As hipóteses do Teorema 3.2 podem ser reescritas como: α_l é o ponto de fronteira com menor valor de imagem, α_l satisfaz a condição de decréscimo suficiente e $\alpha_u - \alpha_l$ é uma direção de descida para ψ em α_l de modo que $\psi(\alpha) < \psi(\alpha_l)$, para todo $\alpha \in I$ suficientemente próximo à α_l.

A seguir descreveremos como atualizar um intervalo I, que satisfaz as condições do Teorema 3.2, de modo que o novo intervalo I_+ também satisfaça essas hipóteses.

Algoritmo Atualizador. Seja I um intervalo com pontos de fronteira α_l e α_u. Dado um valor de teste $\alpha_t \in I$, os pontos de fronteira α_l^+ e α_u^+ do novo intervalo I_+, são determinados da seguinte maneira:
Caso U1: Se $\psi(\alpha_t) > \psi(\alpha_l)$, então $\alpha_l^+ = \alpha_l$ e $\alpha_u^+ = \alpha_t$.
Caso U2: Se $\psi(\alpha_t) \leq \psi(\alpha_l)$ e $\psi'(\alpha_t)(\alpha_l - \alpha_t) > 0$, então $\alpha_l^+ = \alpha_t$ e $\alpha_u^+ = \alpha_u$.
Caso U3: Se $\psi(\alpha_t) \leq \psi(\alpha_l)$ e $\psi'(\alpha_t)(\alpha_l - \alpha_t) < 0$, então $\alpha_l^+ = \alpha_t$ e $\alpha_u^+ = \alpha_l$.

Corolário 1 *Seja I um intervalo com os pontos de fronteira α_l e α_u. Se esses pontos satisfazem (3-10), então os novos pontos de fronteira α_l^+ e α_u^+, fornecidos pelo algoritmo atualizador, também vão satisfazer.*

Demonstração. É claro que se ocorrer $\psi'(\alpha_t) = 0$ e $\psi(\alpha_t) \leq \psi(\alpha_l)$, então não existe a necessidade de atualizar I, pois $\alpha_t \in T(\delta)$. Como α_l e α_u satisfazem (3-10), então

$$\psi(\alpha_l) \leq \psi(\alpha_u), \;\; \psi(\alpha_l) \leq 0, \;\; \psi'(\alpha_l)(\alpha_u - \alpha_l) < 0.$$

Se vale o caso U1, então $\alpha_l^+ = \alpha_l$ e $\alpha_u^+ = \alpha_t$, portanto $\psi(\alpha_u^+) > \psi(\alpha_l^+)$ e pela hipótese $\psi(\alpha_l^+) \leq 0$. Se $\psi'(\alpha_l^+) < 0$ então a hipótese garante $\alpha_u > \alpha_l$ e isso implica que $\alpha_t > \alpha_l$, o que equivale a $\alpha_u^+ > \alpha_l^+$. Agora se $\psi'(\alpha_l^+) > 0$ então $\alpha_u < \alpha_l$ e consequentemente $\alpha_u^+ < \alpha_l^+$. Assim, independentemente do sinal de $\psi'(\alpha_l^+)$, I_+ satisfaz (3-10). Se o caso U2 vale, então $\alpha_l^+ = \alpha_t$ e $\alpha_u^+ = \alpha_u$, logo

$$\psi(\alpha_l^+) \leq \psi(\alpha_l) \leq \psi(\alpha_u) = \psi(\alpha_u^+) \quad , \; \psi(\alpha_l^+) \leq \psi(\alpha_l) \leq 0 \quad \text{e} \quad \psi'(\alpha_l^+)(\alpha_l - \alpha_l^+) > 0.$$

Se $\psi'(\alpha_l^+) > 0$ então $\alpha_l - \alpha_l^+ > 0$ e consequentemente $\alpha_u < \alpha_l^+ < \alpha_l$, logo $\psi'(\alpha_l^+)(\alpha_u^+ - \alpha_l^+) < 0$. Por outro lado, se $\psi'(\alpha_l^+) < 0$ então $\alpha_l < \alpha_l^+$ e portanto $\alpha_l^+ < \alpha_u = \alpha_u^+$ implicando que $\psi'(\alpha_l^+)(\alpha_u^+ - \alpha_l^+) < 0$. Assim, no caso U2, I_+ também satisfaz (3-10). Por fim, se vale o caso U3, é imediato que o novo intervalo também irá satisfazer as hipóteses do Teorema 3.2

■

Vimos que o algoritmo atualizador gera novos intervalos que satisfazem (3-10), se iniciarmos com um intervalo que também satisfaz tais hipóteses. Agora

vamos considerar $\alpha_l = 0$ e $\alpha_u = \infty$, seja algum valor de teste $\alpha_t \in [\alpha_{min}, \alpha_{max}]$. Se vale o caso U1, então $\alpha_l^+ = \alpha_l$ e $\alpha_u^+ = \alpha_t$, logo $\psi(\alpha_u^+) > \psi(\alpha_l^+)$ e como $\psi(0) = 0$ e $\psi'(0) = \varphi'(0) - \delta\varphi'(0) \leq 0$, pois $\delta \leq 1$ e $\varphi'(0) < 0$, então α_l^+ e α_u^+ satisfazem (3-10). Se vale o caso U3 é imediato ver que os pontos de fronteira do novo intervalo vão satisfazer as condições do Teorema 3.2 Agora, se valer o caso U2, o processo deve ser repetido para algum $\alpha_t^+ \in [\alpha_t, \alpha_{max}]$, até que U1 ou U3 seja válido. Porém se o caso U2 ocorrer, então devemos eventualmente usar α_{max} como valor de teste, e isso pode ser feito exigindo que:

$$\alpha_t^+ \in [\min\{\lambda_{max}\alpha_t, \alpha_{max}\}, \alpha_{max}] \tag{3-11}$$

em que $\lambda_{max} > 1$.

Sendo inicialmente $\alpha_l = 0$ e como $\psi(0) = 0$, a sequência $\alpha_0, \alpha_1, \ldots$ de valores testes é crescente e

$$\psi(\alpha_k) \leq 0 \text{ e } \psi'(\alpha_k) < 0,\ k = 0, 1, \ldots, \tag{3-12}$$

enquanto valer U2. Um critério de parada razoável para α_{max} é quando $\psi(\alpha_{max}) \leq 0$ e $\psi'(\alpha_{max}) < 0$, pois, caso contrário, o Teorema 3.2 garante a existência de um $\alpha^* \in T(\delta)$ com $\alpha^* \leq \alpha_{max}$. Deste modo, após um número finito de valores testes, ou o algoritmo termina com α_{max}, que significa um fracasso do método, ou gera um intervalo com os pontos de fronteira satisfazendo (3-10).

Lema 3.2 *Seja $I_0 = [0, \infty]$. Se $I_{k_0} \subset [0, \alpha_{max}]$ é o intervalo gerado pelo algoritmo atualizador, e seus pontos de fronteira satisfazem (3-10). Então após um número finito de valores testes, ou o algoritmo de busca gera um intervalo $J \subset [\alpha_{min}, \alpha_{max}]$ com os pontos de fronteira satisfazendo (3-10), ou a sequência de valores testes é decrescente.*

Demonstração. Suponha que após um número finito de valores testes o algoritmo não gerou $J \subset [\alpha_{min}, \alpha_{max}]$. Assim temos uma sequência de valores testes $\{\alpha_k\}$ com as seguintes propriedades,

$$\psi(\alpha_k) > 0 \text{ ou } \psi'(\alpha_k) \geq 0,\ k = 0, 1, \ldots\ . \tag{3-13}$$

De fato, se as inequações (3-13) não valerem para algum valor teste, ou seja, $\psi(\alpha_t) \leq 0$ e $\psi'(\alpha_t) < 0$. Então o caso U2 vale, logo o algoritmo atualizador nos dará $\alpha_l^+ = \alpha_t$ e $\alpha_u^+ = \alpha_u$. Como $I_{k_0} \subset [0, \alpha_{max}]$ e $\alpha_t \in [\alpha_{min}, \alpha_{max}]$, mostramos a existência de um intervalo $J \equiv [\alpha_l^+, \alpha_u^+] \subset [\alpha_{min}, \alpha_{max}]$. Contrariando nossa primeira premissa e consequentemente os valores testes satisfazem (3-13). É imediato ver assim que essa sequência é decrescente.

■

Agora com esse resultado podemos estabelecer outro critério de parada para o algoritmo. Se ele não encontrar um intervalo $I \subset [\alpha_{min}, \alpha_{max}]$ de modo que seus pontos de fronteira satisfaçam (3-10), então a sequência de valores testes é decrescente. Devemos assim usar α_{min} como um valor de teste em um número finito de passos. Isso é feito escolhendo

$$\alpha_t^+ \in [\alpha_{min}, \max\{\lambda_{mim}\alpha_t, \alpha_{min}\}] \tag{3-14}$$

de modo que $\lambda_{mim} < 1$. Assim o algoritmo termina com α_{min} se $\psi(\alpha_{min}) > 0$ ou $\psi'(\alpha_{min}) \geq 0$. Este critério de terminação é razoável, pois o Teorema 3.2 mostra que, nesse caso, existe um $\alpha^* \in T(\delta)$ tal que $\alpha^* \leq \alpha_{min}$. Assim parar em α_{min}, valendo a condição (3-13), significa o fracasso da busca.

As escolhas (3-11) e (3-14) são duas das regras de salva-guardas que mencionamos anteriormente. A escolha (3-11) é feita somente quando (3-12) vale, do mesmo modo a escolha (3-14) é feira se (3-13) valer. Quando $I_{k_0} \subset [\alpha_{min}, \alpha_{max}]$, para algum $k_0 \in \mathbb{N}$, precisamos de uma terceira regra de salva-guarda para garantir que o comprimento de I_k tenda a zero. Na implementação isso é feito pelo monitoramento do comprimento de I_k. Se o comprimento de $I_k = [a, b]$ não for decrescente por um fator $\lambda < 1$, após duas iterações, então um passo bisseção é usado para o próximo valor teste α_t, isto é

$$\alpha_t = \frac{(a+b)}{2}.$$

Teorema 3.3 *O algoritmo de busca produz uma sequência $\{\alpha_k\} \in [\alpha_{min}, \alpha_{max}]$ de modo que após um número finito de valores testes, umas das seguintes condições valem:*

A busca termina com α_{max}, a sequência de valores testes é crescente e vale (3-12).

A busca termina com α_{min}, a sequência de valores testes é decrescente e (3-13) *vale.*

Um intervalo $I_k \subset [\alpha_{min}, \alpha_{max}]$ é gerado.

Demonstração. Sejam $\alpha_l^{(k)}$ e $\alpha_u^{(k)}$ os pontos de fronteira de I_k e defina

$$\gamma_l^{(k)} = \min\{\alpha_l^{(k)}, \alpha_l^{(k)}\} \quad \text{e} \quad \gamma_u^{(k)} = \max\{\alpha_l^{(k)}, \alpha_l^{(k)}\}.$$

O ponto esquerdo $\gamma_l^{(k)}$ é não decrescente, enquanto que o direito, $\gamma_u^{(k)}$, é não crescente.

Vamos mostrar que $\gamma_u^{(k)} = \infty$ não vale $\forall\, k \geq 0$. Caso $\gamma_u^{(k)} = \infty\ \forall\, k \geq 0$, somente o caso U2 do algoritmo atualizador pode valer, afinal, nos outros dois casos ambos os pontos de fronteira são finitos. Ocorrendo apenas o caso U2, é claro que (3-12) vale e consequentemente (3-11) também se verifica. Assim α_{max} é usado como valor de teste e, portanto, ou o algoritmo termina em α_{max} ou $\gamma_u^{(k)} = \alpha_{max}$.

De modo análogo, vamos mostrar que $\gamma_l^{(k)} = 0$ não vale $\forall\ k \geq 0$. Caso $\gamma_l^{(k)}\ \ \forall\, k \geq 0$, então somente os casos U1 e U3 ocorrem, pois no caso U2, ambos os pontos de fronteira são positivos. Consequentemente (3-13) vale e o valor de teste é escolhido como em (3-14). Se $\alpha_{min} = 0$, então (3-13) não é verificado para todo k, pois $\psi(0) = 0$ e $\psi'(0) < 0$. Agora se $\alpha_{min} > 0$, então eventualmente α_{min} é usado como valor de teste e ,portanto, ou o algoritmo termina em α_{min} ou $\gamma_l^{(k)} = \alpha_{min}$.

Mostramos com isso, que após um número finito de valores testes, ou a busca termina em um dos dois limites, α_{min} ou α_{max}, ou $\gamma_l^{(k)} > 0$ e $\gamma_u^{(k)} < \infty$. Neste último caso $I_k \subset [\alpha_{min}, \alpha_{max}]$.

■

O Teorema 3.3 estabelece três possíveis cenários para o algoritmo após um números finito de valores testes. Os dois primeiros cenários estabelecem terminação para a busca em α_{min} ou α_{max}. Ambos critérios de parada estão relacionados com o fracasso do método. No terceiro caso, o mais interessante, o algoritmo gera um intervalo $I_k \subset [\alpha_{min}, \alpha_{max}]$. Neste caso, as regras de salva-guarda garantem que os comprimentos dos intervalos I_k tendem a zero. Consequentemente, a sequência $\{\alpha_k\}$ converge para um $\alpha^* \in T(\delta)$.

É possível descartar o término finito em α_{min} ou α_{max} assumindo que

$$\psi(\alpha_{min}) \leq 0 \text{ e } \psi'(\alpha_{min}) < 0 \tag{3-15}$$

e

$$\psi(\alpha_{max}) > 0 \text{ ou } \psi'(\alpha_{max}) \geq 0. \tag{3-16}$$

Uma maneira da condição (3-15) ser satisfeita é ,por exemplo, se $\alpha_{mim} = 0$, pois

$$\psi(0) = 0 \text{ e } \psi'(\alpha_{min}) = \varphi'(0) - \delta\varphi'(0) = (1-\delta)\varphi'(0) < 0.$$

Suponha que φ seja limitada. Uma condição suficiente para o cumprimento da

inequação a esquerda em (3-16) é tomar

$$\alpha_{max} \geq \frac{1}{\delta}\left(\frac{\varphi(0)-\varphi_{mim}}{-\varphi'(0)}\right) \tag{3-17}$$

em que φ_{mim} é um limite inferior estrito para $\varphi(\alpha)$. De fato,

$$\psi(\alpha_{max}) = \varphi(\alpha_{max}) - \varphi(0) - \delta\varphi'(0)\alpha_{max} \geq \varphi(\alpha_{max}) - \varphi(0) - \delta\varphi'(0)\frac{1}{\delta}\left(\frac{\varphi(0)-\varphi_{mim}}{-\varphi'(0)}\right) =$$

$$= \varphi(\alpha_{max}) - \varphi(0) + \varphi(0) - \varphi_{mim},$$

$$\Rightarrow \psi(\alpha_{max}) \geq \varphi(\alpha_{max}) - \varphi_{mim} > 0.$$

Agora temos as ferramentas para mostrar que o algoritmo de busca, exceto para casos patológicos, termina com um número finito de iterações, em um ponto admissível. Quando o algoritmo não termina em número finito de passos, então $\{\alpha_k\}$ converge para um ponto $\alpha^* \in T(\delta)$ e a função ψ' muda sinal um número infinito de vezes em uma vizinhança de α^*. No sentido que existe uma sequência $\{\mu_k\}$ que converge para α^* tal que $\psi'(\mu_k)\psi'(\mu_{k+1}) < 0$. Tal situação, por ser atípica em problemas práticos, caracteriza um caso patológico.

Teorema 3.4 *Se os limites α_{min} e α_{max} satisfazem respectivamente (3-15) e (3-16), então o algoritmo de busca termina em um número finito de passos com um $\alpha_k \in T(\delta)$, ou os iterados $\{\alpha_k\}$ convergem para algum $\alpha^* \in T(\delta)$ com $\psi'(\alpha^*) = 0$. Se o algoritmo de busca não terminar em um número finito de passos, então existe um índice k_0 tal que os pontos de fronteira $\alpha_l^{(k)}$, $\alpha_u^{(k)}$ do intervalo I_k satisfazem $\alpha_l^{(k)} < \alpha^* < \alpha_u^{(k)}$. Além disso, se $\psi(\alpha^*) = 0$ então ψ' muda o sinal em $[\alpha_l^{(k)}, \alpha^*]$ para todo $k \geq k_0$, enquanto que se $\psi(\alpha^*) < 0$, então ψ' muda de sinal em $[\alpha_l^{(k)}, \alpha^*]$ ou $[\alpha^*, \alpha_u^{(k)}]$ para todo $k \geq k_0$.*

Demonstração. Suponha que $\alpha_k \notin T(\delta)$ para todos os iterados gerados pelo algoritmo de busca. Desde que os intervalos I_k sejam uniformemente limitados e seus comprimentos tendam a zero, qualquer sequência $\{\theta_k\}$ com $\theta_k \in I_k$ deve convergir para um limite comum α^*. O Teorema 3.2 garante que existe um $\theta_k \in (T(\delta) \cap I_k)$ com $\varphi'(\theta_k) = \delta\varphi'(0)$, pois $\alpha_l^{(k)}$ e $\alpha_u^{(k)}$ satisfazem (3-10) por hipótese e $\psi'(\theta_k) = 0$ implica que $\varphi'(\theta_k) - \delta\varphi'(0) = 0$. Assim $\alpha^* \in T(\delta)$ e pela continuidade da φ', temos

$$\lim_{k\to\infty} \varphi'(\theta_k) = \varphi'(\alpha^*) \Rightarrow \varphi'(\alpha^*) = \delta\varphi'(0) < 0,$$

em particular, $\psi'(\alpha^*) = 0$.

Se o algoritmo de busca não terminar em um número finito de passos, então como a φ é contínua e $\varphi'(\alpha^*) < 0$, vamos definir k_0 tal que $\varphi'(\alpha) < 0 \; \forall \; \alpha \in I_k$,

onde $k \geq k_0$. Como $\psi(\alpha_l^{(k)}) \leq 0$ e $\alpha_l^{(k)} \notin T(\delta)$, então $|\varphi'(\alpha_l^{(k)})| > \delta|\varphi'(0)|$, pois $T(\delta) = \{\alpha > 0 : \varphi(\alpha) \leq \varphi(0) + \delta\varphi'(0)\alpha, \quad |\varphi'(\alpha)| \leq \delta|\varphi'(0)|\}$. Por I_k ser fechado, $\varphi'(\alpha_l^{(k)}) < 0$ para $k \geq k_0$, e como $\alpha_l^{(k)} \notin T(\delta)$ então

$$-\varphi'(\alpha_l^{(k)}) > \delta|\varphi'(0)| \Rightarrow \varphi'(\alpha_l^{(k)}) < \delta\varphi'(0),$$

logo $\psi'(\alpha_l^{(k)}) < 0$. A condição (3-10) nos pontos de fronteira nos dá $\psi'(\alpha_l^{(k)})(\alpha_u^{(k)} - \alpha_l^{(k)}) < 0$, portanto $\alpha_l^{(k)} < \alpha_u^{(k)}$, particularmente, $\alpha_l^{(k)} < \alpha^* < \alpha_u^{(k)}$.

Agora, se $\psi(\alpha^*) = 0$, não devemos ter $\psi'(\alpha) \leq 0 \ \forall \alpha \in [\alpha_l^{(k)}, \alpha^*]$, pois isso implicaria que $\psi(\alpha_l^{(k)}) > \psi(\alpha^*) = 0$, que contradiz o fato $\psi(\alpha_l^{(k)}) \leq 0$. Portanto deve existir um $\mu_k \in [\alpha_l^{(k)}, \alpha^*]$ tal que $\psi'(\mu_k) > 0$. Como mostramos que $\psi'(\alpha_l^{(k)}) < 0$, então ψ' muda de sinal em $[\alpha_l^{(k)}, \alpha^*]$.

Por fim, se $\psi(\alpha^*) < 0$, vamos tomar $k_0 > 0$ tal que $\psi(\alpha_u^{(k)}) < 0$ para todo $k \geq k_0$. Suponha que $\psi'(\alpha_u^{(k)}) \geq 0$, então $\varphi'(\alpha_u^{(k)}) \geq \delta\varphi'(0)$. Tomando um novo k_0, se necessário, temos que a continuidade de φ' nos dá $\varphi'(\alpha_u^{(k)}) < 0$, implicando que $\alpha_u^{(k)} \in T(\delta)$, contrariando a hipótese $\alpha_k \notin T(\delta) \ \forall \ k > 0$. Logo $\psi'(\alpha_u^{(k)}) < 0$. Assim, não podemos ter $\psi'(\alpha) \geq 0 \ \forall \ \alpha \in [\alpha_l^{(k)}, \alpha_u^{(k)}]$, pois isso implicaria $\psi(\alpha_l^{(k)}) > \psi(\alpha_u^{(k)})$ que é uma contradição. Logo deve existir algum $\mu_k \in [\alpha_l^{(k)}, \alpha_u^{(k)}]$ tal que $\psi'(\mu_k) > 0$. Portanto ψ' muda de sinal em $[\alpha_l^{(k)}, \alpha^*]$ ou $[\alpha^*, \alpha_u^{(k)}]$.

■

3.2.2 Busca por um minimizador local

A princípio o algoritmo é estruturado para buscar minimizadores de $\psi(\alpha)$. Entretanto gostaríamos, no caso ideal, de encontrar minimizadores para $\varphi(\alpha)$. Nesta subseção veremos uma condição suficiente para garantir a existência de um minimizador de $\varphi(\alpha)$. Assim mostraremos como esse resultado pode ser incorporado no algoritmo, de maneira que busquemos minimizadores de $\varphi(\alpha)$ e não de $\psi(\alpha)$. A garantia de terminação finita do Teorema 3.4 é dada quando $\delta \leq \sigma$. Um dos problemas de deixar a escolha do σ livre e buscar mínimo de φ desde o início, tomando $\sigma \approx 0$, é a possibilidade de não existir $\alpha \geq 0$ que satisfaça (3-3) e (3-6), quando $\sigma < \delta$, mesmo que $T(\delta) \neq \emptyset$. Vamos ilustrar esse acontecimento com uma modificação de um exemplo de Al-Baali e Fletcher, proposto em [2]. Defina

$$\varphi(\alpha) = \begin{cases} \frac{1}{2}(1-\beta)\alpha^2 - \alpha, & 0 \leq \alpha \leq 1 \\ \frac{1}{2}(\beta - 1) - \beta\alpha, & \alpha \geq 1, \end{cases} \tag{3-18}$$

tal que $\sigma < \beta < \delta$. A função acima é continuamente diferenciável pois é definida por duas funções de classe C^1 e no ponto $\alpha = 1$ as derivadas pela esquerda e pela direita têm o mesmo valor. De fato

$$\varphi'(\alpha) = \begin{cases} (1-\beta)\alpha - 1, & 0 \leq \alpha \leq 1 \\ -\beta, & \alpha \geq 1, \end{cases}$$

logo $\varphi'(1) = -\beta$. Temos também que, se $0 \leq \alpha \leq 1$ então

$$\alpha \leq \frac{1-\beta}{1-\beta} \Rightarrow (1-\beta)\alpha - 1 \leq -\beta.$$

Assim

$$|\varphi'(\alpha)| \geq \beta > \sigma, \ \forall \ \alpha \geq 0.$$

Além disso, se $\delta < \frac{1}{2}$, então

$$T(\delta) = \left[\frac{1-\delta}{1-\beta}, \ \frac{1-\beta}{2(\delta-\beta)}\right].$$

Logo $T(\delta) \neq \emptyset$, com $\alpha = 1$ em seu interior.

Vamos mostrar que, se a busca para $T(\delta)$ encontrar um valor de teste α_k tal que $\psi(\alpha_k) \leq 0$ e $\psi'(\alpha_k) \geq 0$, então $\alpha_k \in T(\delta)$ ou teremos identificado um intervalo que contém pontos satisfazendo (3-3) e (3-6), para qualquer $\sigma \in (0,1)$.

Teorema 3.5 *Suponha que os limites α_{min} e α_{max} satisfaçam (3-15) e (3-16). Seja $\{\alpha_k\}$ a sequência gerada pelo algoritmo de busca e $\alpha_l^{(k)}$ e $\alpha_u^{(k)}$ os pontos de fronteira do intervalo I_k gerado. Se α_k é o primeiro iterado que satisfaz*

$$\psi(\alpha_k) \leq 0 \ e \ \psi'(\alpha_k) \geq 0, \tag{3-19}$$

então $\alpha_l^{(k)} < \alpha_u^{(k)}$. Além disso, $\alpha_k \in T(\delta)$ ou $\varphi'(\alpha_k) > 0$. Se $\varphi'(\alpha_k) > 0$ então o intervalo

$$I^* \equiv [\alpha_l^{(k)}, \alpha_k]$$

contém um α^* que satisfaz (3-3) e $\varphi'(\alpha^*) = 0$. Ainda mais, qualquer $\alpha \in I^*$ com $\varphi(\alpha) \leq \varphi(\alpha_k)$ também satisfaz (3-3).

Demonstração. Afirmamos que $\psi'(\alpha_l^{(k)}) < 0$. De fato, caso contrário, teríamos $\psi'(\alpha_l^{(k)}) \geq 0$ e, pela construção dos I_k, $\psi(\alpha_l^{(k)}) \leq 0$. Contrariando a hipótese de que α_k é o primeiro a satisfazer (3-19), afinal $\alpha_l^{(k)}$ foi um valor teste α_j para algum $j < k$. Portanto $\psi'(\alpha_l^{(k)}) < 0$. Consequentemente por (3-10), $\alpha_l^{(k)} < \alpha_u^{(k)}$ e, particularmente, $\alpha_l^{(k)} < \alpha_k$, deixando I^* bem definido. Pela hipótese $\psi'(\alpha_k) \geq 0$. Se $\varphi'(\alpha_k) \leq 0$, então

temos

$$0 \leq \psi'(\alpha_k) = \varphi'(\alpha_k) - \delta\varphi'(0) \Rightarrow \varphi'(\alpha_k) \geq \delta\varphi'(0) \Rightarrow |\varphi'(\alpha_k)| \leq \delta|\varphi'(0)|.$$

Como $\psi(\alpha_k) \leq 0$, logo $\alpha_k \in T(\delta)$.

Agora suponha que $\varphi'(\alpha_k) > 0$. Como φ é contínua e I^* é compacto, tome α^* um minimizador global de φ. Temos $\alpha_k \neq \alpha^*$, pois numa vizinhança de α_k a função φ é crescente, ou seja, os pontos na vizinhança à esquerda de α_k têm imagens de menor valor que $\varphi(\alpha_k)$. Uma vez que $\psi'(\alpha_l^{(k)}) < 0$ temos:

$$\varphi'(\alpha_l^{(k)}) - \delta\varphi'(0) < 0 \Rightarrow \varphi'(\alpha_l^{(k)}) < \delta\varphi'(0) < 0.$$

Logo $\alpha_l^{(k)} \neq \alpha^*$, pois os pontos numa vizinhança à direita de $\alpha_l^{(k)}$ têm valores de imagens menores do que $\varphi(\alpha_l^{(k)})$. Consequentemente α^* está no interior de I^* e $\varphi'(\alpha^*) = 0$. Finalmente, suponha que $\varphi(\alpha) \leq \varphi(\alpha_k)$ para algum $\alpha \in I^*$, então

$$\varphi(\alpha) \leq \varphi(\alpha_k) \leq \varphi(0) + \delta\varphi'(0)\alpha_k \leq \varphi(0) + \delta\varphi'(0)\alpha,$$

logo α satisfaz (3-3).

■

A existência de α_k tal que $\psi(\alpha_k) \leq 0$ e $\varphi'(\alpha_k) > 0$ é a chave para o algoritmo identificar um intervalo contendo minimizadores de φ. Entretanto, não existem garantias que o algoritmo gera um ponto satisfatório tendo tais condições. Um exemplo é a função que definimos em (3-18), para qual $\psi'(\alpha) < 0$ para todo $\alpha \geq 0$. Mesmo se φ tiver um minimizador α^* que satisfaça a condição de decréscimo suficiente, o algoritmo pode não encontrá-lo, convergindo para algum ponto em $T(\delta)$. Vamos ilustrar essa situação com a função

$$\varphi(\alpha) = -\frac{3\alpha}{\alpha^2 + 2} - \delta\alpha, \tag{3-20}$$

tomando $\delta = 0.03$. Na Figura 3.4, o intervalo no eixo das abcissas, em destaque, é o conjunto $T(\delta)$. Se $\sigma > \delta$, o intervalo de valores que satisfazem (3-3) e (3-6) é maior que $T(\delta)$, se $\sigma < \delta$ então o intervalo será menor. O Teorema 3.4 garante terminação finita em um $\alpha_k \in T(\delta)$, mas esse ponto pode não estar próximo ao minimizador α^*.

O algoritmo será modificado para buscar um ponto α_j próximo de um minimizador local de φ, quando $\psi(\alpha_k) \leq 0$ e $\varphi'(\alpha_k) > 0$ para algum k. Antes, necessitamos do seguinte resultado técnico.

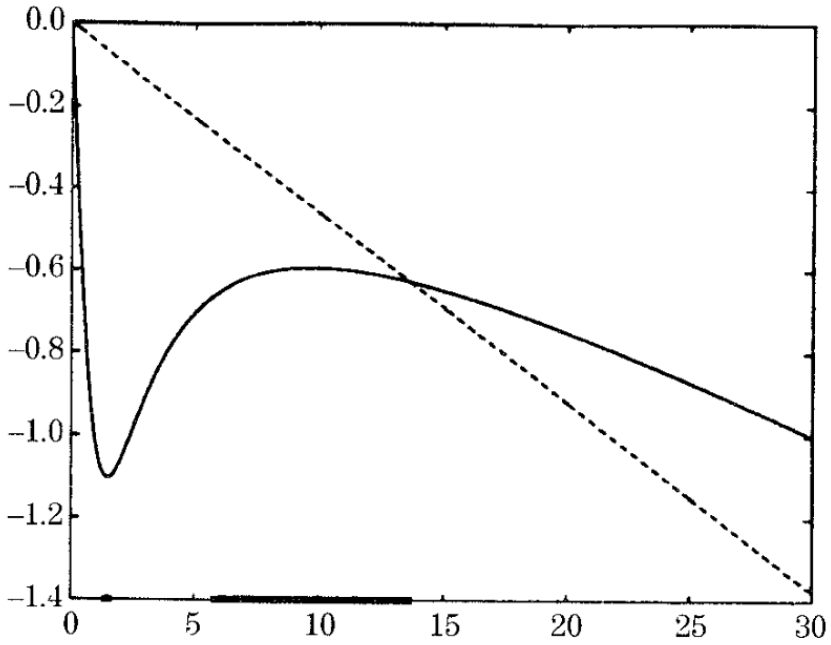

Figura 3.4: *Retirada de [13]. A reta plotada é* $l(\alpha) = \varphi(0) + \delta\varphi'(0)\alpha$.

Teorema 3.6 *Seja I um intervalo fechado com os pontos de fronteira α_l e α_u. Se os pontos de fronteira satisfazem*

$$\varphi(\alpha_l) \leq \varphi(\alpha_u) \text{ e } \varphi'(\alpha_l)(\alpha_u - \alpha_l) < 0,$$

então existe $\alpha^* \in I$ com $\varphi(\alpha^*) \leq \varphi(\alpha_l)$ e $\varphi'(\alpha^*) = 0$.

Demonstração. Seja α^* o minimizador global da função contínua φ no conjunto compacto I. Suponha que $\alpha^* = \alpha_u$, logo $\varphi(\alpha^*) = \varphi(\alpha_u)$ e portanto a hipótese garante $\varphi(\alpha_l) = \varphi(\alpha_u)$. Se $\alpha_u < \alpha_l$ então, por hipótese, $\varphi'(\alpha_l) > 0$ e consequentemente existe um vizinhança de α_l em que a função é crescente, logo as imagens dos pontos na vizinhança à esquerda de α_l são menores que $\varphi(\alpha_l)$, que contraria o fato de $\varphi(\alpha^*)$ ser mínimo. Se $\alpha_u > \alpha_l$ então $\varphi'(\alpha_l) < 0$ e de modo análogo vamos obter pontos à direita de α_l que têm suas imagens menores que $\varphi(\alpha_l)$, novamente uma contradição. Como $\alpha_l = \alpha_u$ não pode ocorrer pela construção, então $\alpha^* \neq \alpha_u$.

Supondo que $\alpha^* = \alpha_l$, através de argumentos semelhantes, chegamos a uma contradição, concluindo que α^* está no interior de I e portanto $\varphi'(\alpha^*) = 0$.

■

Vamos mostrar agora que o intervalo I^*, especificado no Teorema 3.5, satisfaz as hipóteses do Teorema 3.6. O Teorema 3.5 garante que $\varphi'(\alpha_k) > 0$ e $\alpha_l^{(k)} < \alpha_u^{(k)}$. Portanto as hipóteses (3-10) sobre os pontos de fronteira de I_k implicam

que $\psi'(\alpha_l^{(k)}) < 0$, consequentemente $\varphi'(\alpha_l^{(k)}) < \delta\varphi'(0) < 0$. Assim afirmamos que I^* satisfaz as hipóteses do Teorema 3.6. De fato, se $\alpha_l^{(k)} = \alpha_l$ e $\alpha_k = \alpha_u$ então $\varphi'(\alpha_l)(\alpha_u - \alpha_l) = \varphi'(\alpha_l^{(k)})(\alpha_k - \alpha_l^{(k)}) < 0$, pois $\alpha_k - \alpha_l^{(k)} > 0$. Se $\alpha_l^{(k)} = \alpha_u$ e $\alpha_k = \alpha_l$ então $\varphi'(\alpha_l)(\alpha_u - \alpha_l) = \varphi'(\alpha_k)(\alpha_l^{(k)} - \alpha_k) < 0$, pois $\alpha_l^{(k)} - \alpha_k < 0$, como queríamos mostrar.

Uma vez obtido α_k tal que $\psi(\alpha_k) \leq 0$ e $\varphi'(\alpha_k) > 0$, o algoritmo é modificado de forma a garantir término finito em um iterado satisfazendo (3-3) e (3-6) para qualquer $\sigma \in (0,1)$. A modificação é simples, apenas trocamos ψ por φ no algoritmo atualizador.

Algoritmo Atualizador Modificado. Dado um valor de teste $\alpha_t \in I$, os pontos de fronteira α_l^+ e α_u^+ do novo intervalo I_+, são determinados da seguinte maneira:
Caso M1: Se $\varphi(\alpha_t) > \varphi(\alpha_l)$, então $\alpha_l^+ = \alpha_l$ e $\alpha_u^+ = \alpha_t$.
Caso M2: Se $\varphi(\alpha_t) \leq \varphi(\alpha_l)$ e $\varphi'(\alpha_t)(\alpha_l - \alpha_t) > 0$, então $\alpha_l^+ = \alpha_t$ e $\alpha_u^+ = \alpha_u$.
Caso M3: Se $\varphi(\alpha_t) \leq \varphi(\alpha_l)$ e $\varphi'(\alpha_t)(\alpha_l - \alpha_t) < 0$, então $\alpha_l^+ = \alpha_t$ e $\alpha_u^+ = \alpha_l$.

Mostramos que o intervalo I^* dado pelo Teorema 3.5 satisfaz as hipóteses do Teorema 3.6. Vamos mostrar agora que o algoritmo atualizador modificado mantém essa propriedade, ou seja, se I é um intervalo qualquer que satisfaz as hipóteses do Teorema 3.6, então o novo intervalo gerado pelo algoritmo atualizador modificado irá satisfazer as hipóteses do Teorema 3.6. De fato, sejam α_l e α_u pontos de fronteira do intervalo I e suponha, sem perda de generalidade, que $\alpha_l < \alpha_u$. Se vale M1, ou seja, $\varphi(\alpha_t) > \varphi(\alpha_l)$, então $\alpha_l^+ = \alpha_l$ e $\alpha_u^+ = \alpha_t$, satisfazendo a primeira condição do Teorema 3.6. Como $\varphi'(\alpha_l)(\alpha_u - \alpha_l) < 0$ e $\alpha_u - \alpha_l > 0$ por hipótese, então $\varphi'(\alpha_l) < 0$ e por $\alpha_t \in I$, temos $\varphi'(\alpha_l^+)(\alpha_u^+ - \alpha_l^+) < 0$, obtendo assim a segunda condição. Se vale M2, logo $\varphi(\alpha_t) \leq \varphi(\alpha_l)$, $\varphi'(\alpha_t)(\alpha_l - \alpha_t) > 0$, $\alpha_l^+ = \alpha_t$ e $\alpha_u^+ = \alpha_u$, então $\varphi(\alpha_l^+) \leq \varphi(\alpha_u^+)$, pois $\varphi(\alpha_l) \leq \varphi(\alpha_u)$ por hipótese, e também temos $\varphi'(\alpha_l^+) < 0$, pois $\alpha_l < \alpha_t$, consequentemente $\varphi'(\alpha_l^+)(\alpha_u^+ - \alpha_l^+) < 0$. Se vale M3, é imediato ver que I_+ satisfaz as hipóteses do Teorema 3.6. O caso $\alpha_l > \alpha_u$ é análogo.

O algoritmo de busca usará o algoritmo atualizador modificado de uma maneira muito simples. Se em alguma iteração α_k satisfazer $\psi(\alpha_k) \leq 0$ e $\varphi'(\alpha_k) > 0$, então a busca passará a atualizar todos os intervalos, dali em diante, usando o algoritmo atualizador modificado. A garantia de que esse algoritmo de busca atualizado irá encontrar um α satisfazendo (3-3) e (3-6) para qualquer $\sigma \in (0,1)$, é dada pelo próximo resultado.

Teorema 3.7 *Suponha que α_{min} e α_{max} satisfaçam (3-15) e (3-16) respectivamente. Se o algoritmo de busca modificado gerar um iterado tal que $\psi(\alpha_k) \leq 0$ e $\varphi'(\alpha_k) > 0$, então o método termina em um α_j que satisfaz (3-3) e (3-6), para qualquer $\sigma \in (0,1)$.*

Demonstração. Seja α_k um iterado tal que $\psi(\alpha_k) \leq 0$ e $\varphi'(\alpha_k) > 0$, logo o Teorema 3.5 garante que $\alpha_k > \alpha_l^{(k)}$. Se valer os casos M1 e M3, então o intervalo fornecido pelo algoritmo atualizador modificado é:

$$I_{k+1} = [\alpha_l^{(k)}, \alpha_k].$$

O caso M2 não ocorre pois $\varphi'(\alpha_k)(\alpha_l^{(k)} - \alpha_k) > 0$, contrariando os fatos $\varphi'(\alpha_k) > 0$ e $\alpha_k > \alpha_l^{(k)}$. O Teorema 3.5 também garante que qualquer $\alpha \in I_{k+1}$ com $\varphi(\alpha) \leq \varphi(\alpha_k)$ satisfaz (3-3). Afirmamos que para qualquer iteração $j > k$, o ponto de fronteira $\alpha_l^{(j)}$ satisfaz (3-3). Vamos provar essa afirmação por indução, para $j = k+1$. Suponha que vale o caso M1, logo $\alpha_l^{(k+1)} = \alpha_l$ assim se $\alpha_l = \alpha_l^{(k)}$ então $\varphi(\alpha_l^{(k+1)}) = \varphi(\alpha_l^{(k)}) \leq \varphi(\alpha_k)$, ou se $\alpha_l = \alpha_k$ então $\varphi(\alpha_l^{(k+1)}) = \varphi(\alpha_k)$. Se vale M2, ou seja, $\varphi(\alpha_t) \leq \varphi(\alpha_l)$ então $\alpha_l^{(k+1)} = \alpha_t$, agora se $\alpha_l = \alpha_l^{(k)}$ então $\varphi(\alpha_l^{(k+1)}) \leq \varphi(\alpha_l^{(k)}) \leq \varphi(\alpha_k)$, e se $\alpha_l = \alpha_k$ então $\varphi(\alpha_l^{(k+1)}) \leq \varphi(\alpha_k)$. O caso M3 é análogo ao M2, e portanto a afirmação vale para $j = k+1$. Suponha a seguinte hipótese de indução, $\varphi(\alpha_l^{(j)}) \leq \varphi(\alpha_k)$ para algum $j > k$, e vamos provar para $j+1$. Se vale M1 então $\alpha_l^{(j+1)} = \alpha_l^{(j)}$, logo $\varphi(\alpha_l^{(j+1)}) \leq \varphi(\alpha_k)$. Se vale M2 então $\alpha_l^{(j+1)} = \alpha_t$ e portanto $\varphi(\alpha_l^{(j+1)}) \leq \varphi(\alpha_l^{(j)}) \leq \varphi(\alpha_k)$. O caso M3 é análogo ao caso M2 e assim provamos a afirmação para todo $j > k$. Agora notemos o seguinte fato: qualquer sequencia $\{\theta_k\}$ com $\theta_k \in I_k$ deve convergir para um limite comum, α^*. Pelo Teorema 3.6, existe um $\theta_{k+1} \in I_{k+1}$ tal que $\varphi'(\theta_{k+1}) = 0$ e pela continuidade de φ', obtemos $\varphi'(\alpha^*) = 0$. Tomando $j > k$ suficientemente grande, temos que $\alpha_l^{(j)}$ satisfaz (3-6), para qualquer $\sigma \in (0,1)$. Assim provamos que o algoritmo de busca modificado termina em uma iteração que satisfaz (3-3) e (3-6), para qualquer $\sigma \in (0,1)$.

■

3.2.3 A escolha do passo de teste

Seja I um intervalo nos quais α_l e α_u são seus pontos de fronteira, e α_t é um valor teste em I, o algoritmo atualizador gera um novo intervalo I_+. Nesta subseção, vamos mostrar como selecionar um novo valor teste $\alpha_t^+ \in I_+$. Vamos representar os valores funcionais nesses pontos por f_l, f_u, f_t, e das derivadas como g_l, g_u e g_t. Os valores funcionais f_l, f_u, f_t ou das derivadas g_l, g_u e g_t, podem ser tanto da função objetivo φ, quanto da auxiliar ψ. Os valores funcionais e das derivadas da função ψ são utilizados até que o algoritmo gere um iterado α_k que satisfaça $\psi(\alpha_k) \leq 0$ e $\varphi'(\alpha_k) > 0$, consequentemente, será utilizado os valores funcionais e das derivadas da função objetivo φ.

Seja α_q o minimizador da quadrática que interpola f_l, f_t e g_l, definida por: $q(\alpha) = a\alpha^2 + b\alpha + c$. As condições de interpolações são,

$$\begin{aligned} q(\alpha_l) &= a\alpha_l^2 + b\alpha_l + c = f_l, \\ q(\alpha_t) &= a\alpha_t^2 + b\alpha_t + c = f_t \\ q'(\alpha_l) &= 2a\alpha_l + b = g_l. \end{aligned}$$

Resolvendo o sistema, temos:

$$\begin{aligned} a &= \frac{f_l - f_t - g_l * (\alpha_l - \alpha_t)}{-(\alpha_l - \alpha_t)^2}, \\ b &= g_l + \frac{2\alpha_l(f_l - f_t - g_l * (\alpha_l - \alpha_t))}{(\alpha_l - \alpha_t)^2}. \end{aligned}$$

Assim,

$$\begin{aligned} \alpha_q &= -\frac{b}{2a} \\ &= \alpha_l - \frac{1}{2}\frac{(\alpha_l - \alpha_t)g_l}{g_l - \frac{f_l - f_t}{\alpha_l - \alpha_t}}. \end{aligned}$$

Seja α_s o minimizador de uma quadrática que interpola f_l, g_t e g_l, com as seguintes condições de interpolação:

$$\begin{aligned} q(\alpha_l) &= a\alpha_l^2 + b\alpha_l + c = f_l, \\ q'(\alpha_l) &= 2a\alpha_l + b = g_l, \\ q'(\alpha_t) &= 2a\alpha_t + b = g_t. \end{aligned}$$

De modo análogo, obtemos,

$$\begin{aligned} \alpha_s &= -\frac{b}{2a} \\ &= \alpha_l - \frac{(\alpha_l - \alpha_t)g_l}{g_l - g_t}. \end{aligned}$$

Por fim, seja α_c o minimizador da cúbica que interpola f_l, f_t, g_l e g_t, definida por:

$$p(\alpha) = c_1(\alpha - \alpha_l)^3 + c_2(\alpha - \alpha_l)^2 + c_3(\alpha - \alpha_l) + c_4,$$

de modo que os coeficientes c_i são escolhidos tais que:

$$\begin{aligned} &p(\alpha_l) = c_4 = f_l, \\ &p'(\alpha_l) = c_3 = g_l, \\ &p(\alpha_t) = c_1(\alpha_t - \alpha_l)^3 + c_2(\alpha_t - \alpha_l)^2 + g_l(\alpha_t - \alpha_l) + f_l = f_t, \\ &p'(\alpha_t) = 3c_1(\alpha_t - \alpha_l)^2 + 2c_2(\alpha_t - \alpha_l) + g_l = g_t. \end{aligned}$$

Para obtermos c_1 e c_2, resolveremos o sistema abaixo:

$$\begin{cases} c_1(\alpha_t-\alpha_l)^3 + c_2(\alpha_t-\alpha_l)^2 = f_t - g_l(\alpha_t-\alpha_l) - f_l \\ 3c_1(\alpha_t-\alpha_l)^2 + 2c_2(\alpha_t-\alpha_l) = g_t - g_l. \end{cases} \tag{3-21}$$

Como o determinante da matriz associada é não nulo, a solução do sistema (3-21) é dada por:

$$\begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} \frac{-2}{(\alpha_t-\alpha_l)^3} & \frac{1}{(\alpha_t-\alpha_l)^2} \\ \frac{3}{(\alpha_t-\alpha_l)^2} & \frac{-1}{(\alpha_t-\alpha_l)} \end{bmatrix} * \begin{bmatrix} f_t - g_l(\alpha_t-\alpha_l) - f_l \\ g_t - g_l \end{bmatrix}.$$

Assim o minimizador da cúbica é:

$$\alpha_c = \alpha_l - \frac{g_l}{c_2 + \sqrt{c_2^2 - 3c_1 g_l}}.$$

Os cálculos acima, com maior riqueza de detalhes, podem ser vistos em [23].

Vamos dividir o processo de escolha de valores testes em quatro casos. Nos dois primeiros, iremos escolher α_t^+ pela interpolação dos valores funcionais em α_l e α_t, de modo que o novo valor esteja entre esse dois pontos. Vamos definir α_t^+ em termos de α_q, α_s e α_c. No terceiro caso escolheremos α_t^+ extrapolando os valores funcionais em α_l e α_t, assim o valor teste α_t^+ estará do lado de fora do intervalo com α_t e α_l sendo os pontos de fronteira. Definiremos α_t^+ em termos de α_c e α_s. No último caso, as informações disponíveis em α_l e α_t indicam que a função está diminuindo rapidamente na direção do passo, mas não parece haver uma boa maneira de escolher α_t^+ a partir das informações disponíveis.

Caso 1: $f_t > f_l$. Calcule α_c, α_q e faça

$$\alpha_t^+ = \begin{cases} \alpha_c, & \text{se } |\alpha_c-\alpha_l| < |\alpha_q-\alpha_l|, \\ \frac{1}{2}(\alpha_q+\alpha_c), & \text{caso contrário.} \end{cases}$$

Como α_c e α_q estão em I_+, ambos são candidatos para α_t^+. Por α_l ser um ponto com menor valor funcional, em relação a α_t, desejamos que a escolha seja próxima de α_l. Os pontos α_c e α_q são relativamente próximos a α_l, pois

$$|\alpha_c-\alpha_l| \leq \frac{2}{3}|\alpha_u-\alpha_l| \;\text{ e }\; |\alpha_q-\alpha_l| \leq \frac{1}{2}|\alpha_u-\alpha_l|.$$

Assim, se $\alpha_t^+ = \alpha_c$ então

$$|\alpha_t^+-\alpha_l| \leq |\alpha_q-\alpha_l| \leq \frac{1}{2}|\alpha_u-\alpha_l|,$$

ou se $\alpha_t^+ = \frac{1}{2}(\alpha_q + \alpha_c)$ então

$$\left|\frac{\alpha_q}{2} + \frac{\alpha_c}{2} - \alpha_l\right| \leq \left|\frac{\alpha_q}{2} - \frac{\alpha_l}{2}\right| + \left|\frac{\alpha_c}{2} + \frac{\alpha_l}{2}\right| \leq \frac{1}{4}|\alpha_u - \alpha_l| + \frac{2}{6}|\alpha_u - \alpha_l|.$$

Para qualquer uma das escolhas de α_t^+, temos

$$|\alpha_t^+ - \alpha_l| \leq \frac{7}{12}|\alpha_u - \alpha_l|.$$

Quando f_t é muito maior que f_l, então a escolha de um passo próximo de α_l é claramente desejado. Neste caso, o passo quadrático está mais próximo de α_l do que o passo cúbico. De fato, se $\alpha_q(f_t)$ é o valor de α_q em função de f_t, então

$$\lim_{f_t \to \infty} \alpha_q(f_t) = \alpha_l.$$

Por outro lado temos

$$\lim_{f_t \to \infty} \alpha_c(f_t) = \alpha_l + \frac{2}{3}(\alpha_u - \alpha_l),$$

logo o ponto médio entre α_c e α_q é uma escolha razoável.

Caso 2: Se $f_t \leq f_l$ e $g_t g_l < 0$, calcule α_c, α_s e faça

$$\alpha_t^+ = \begin{cases} \alpha_c, & \text{se } |\alpha_c - \alpha_t| \geq |\alpha_s - \alpha_t|, \\ \alpha_s, & \text{caso contrário,} \end{cases}$$

como α_c e α_s estão em I_+, ambos são candidatos para α_t^+. Sendo $g_t g_l < 0$, então o teorema do valor intermediário nos garante que existe um minimizador entre α_l e α_t. A escolha de um passo que esteja mais longe de α_t pode fazer com que a escolha ultrapasse o minimizador e assim os próximos passos provavelmente vão satisfazer este caso.

Caso 3: $f_t \leq f_l$, $g_t g_l \geq 0$ e $|g_t| \leq |g_l|$. Neste caso a cúbica que interpolamos pode não ter um minimizador, e mesmo que existir α_c, ele pode estar na direção errada. Por exemplo, podemos ter $\alpha_t > \alpha_l$, porém $\alpha_c < \alpha_t$. Por outro lado, o passo α_s existe e sempre está na direção correta.

Se a cúbica tende ao infinito na direção do passo e o mínimo dela está além de α_t, defina

$$\alpha_t^+ = \begin{cases} \alpha_c, & \text{se } |\alpha_c - \alpha_t| < |\alpha_s - \alpha_t|, \\ \alpha_s, & \text{caso contrário,} \end{cases}$$

caso isso não ocorra, defina $\alpha_t^+ = \alpha_s$. Esta escolha é baseada em observação, de modo que durante a extrapolação é sensato ser cauteloso e escolher o passo mais

próximo de α_t.

O valor teste definido acima pode estar do lado de fora do intervalo com os pontos de fronteira α_t e α_u, ou ele pode estar neste intervalo, porém próximo a α_u. Qualquer dessas situações são indesejadas, assim vamos redefinir α_t^+ por

$$\alpha_t^+ = \begin{cases} \min\{\alpha_t + \lambda(\alpha_u - \alpha_t), \alpha_t^+\}, & \text{se } \alpha_t > \alpha_l, \\ \max\{\alpha_t + \lambda(\alpha_u - \alpha_t), \alpha_t^+\} & \text{caso contrário,} \end{cases}$$

para algum $\lambda < 1$.

Caso 4: $f_t \leq f_l$, $g_t g_l \geq 0$, e $|g_t| > |g_l|$. Nesse caso vamos escolher α_t^+ como o minimizador da cúbica que interpola f_u, f_t, g_u e g_t.

CAPÍTULO 4

O método gradiente conjugado Dai-Yuan modificado

Os métodos de gradiente conjugado (GC) não-lineares são métodos de primeira ordem reconhecidamente eficientes para resolver problemas de otimização irrestritos, ou seja,

$$\text{Minimizar} \quad f(x), \quad x \in \mathbb{R}^n,$$

em que $f\colon \mathbb{R}^n \to \mathbb{R}$ é continuamente diferenciável. Devido ao baixo uso de recursos computacionais, esses métodos são amplamente aplicados à problemas de larga escala. O método, de modo geral, é definido da seguinte maneira:

Algoritmo 2 *Seja $x_1 \in \mathbb{R}^n$. Calcule $g_1 = \nabla f(x_1)$ e inicie $k \leftarrow 1$.*

Passo 1 *Se $g_k = 0$, então PARE.*

Passo 2 *Defina:*

$$d_k = \begin{cases} -g_k, & \text{se } k = 1, \\ -g_k + \beta_{k-1} d_{k-1}, & \text{se } k > 1, \end{cases}$$

de modo que β_{k-1} é um parâmetro algorítmico.

Passo 3 *Calcule um $\alpha_k > 0$ satisfazendo as condições de Wolfe (fortes ou padrões). Faça $x_{k+1} = x_k + \alpha_k d_k$.*

Passo 4 *Calcule g_{k+1}, faça $k \leftarrow k+1$ e vá ao **Passo 1**.*

Historicamente, o primeiro método de GC não-linear foi proposto por Fletcher e Reeves na década de 60, [8], com parâmetro β_{k-1} definido por:

$$\beta_{k-1}^{FR} = \frac{g_k^T g_k}{g_{k-1}^T g_{k-1}}.$$

A convergência do Algoritmo 2 com β_{k-1}^{FR}, supondo que os comprimentos de passos α_k satisfaçam (3-2) e (3-5), foi mostrada em [1]. Após esse trabalho precursor,

outros métodos GC não-lineares surgiram propondo uma modificação no parâmetro β_{k-1}, alguns deles são: Polak–Ribière–Polyak [16,17]

$$\beta_k^{PRP} = \frac{g_{k+1}^T(g_{k+1} - g_k)}{g_k^T g_k},$$

Hestenes-Stiefel [11]

$$\beta_k^{HS} = \frac{g_{k+1}^T(g_{k+1} - g_k)}{(g_{k+1} - g_k)^T d_k}.$$

A convergência do PRP ou HS pode ser obtida tomando β_k, respectivamente, como $\beta_k = \max\{0, \beta_k^{PRP}\}$ e $\beta_k = \max\{0, \beta_k^{HS}\}$, vide [9]. Foi mostrado em [18] que o Algoritmo 2 com o parâmetro β_k^{PRP} ou com β_k^{HS}, mesmo usando busca exata, pode não convergir. Foi mostrado em [4] que o parâmetro Conjugate Descent [7], definido por:

$$\beta_k^{CD} = -\frac{g_{k+1}^T g_{k+1}}{g_k^T d_k},$$

pode não convergir, mesmo com passos satisfazendo as condições fortes de Wolfe. Por fim, temos o parâmetro Dai–Yuan [5],

$$\beta_k^{DY} = \frac{\|g_{k+1}\|^2}{g_{k+1}^T d_k - g_k^T d_k},$$

que possui convergência supondo que o comprimento de passo satisfaça as condições padrões de Wolfe. Todos os parâmetros propostos recuperam o parâmetro original (2-10), quando f é quadrática convexa.

Neste capítulo, vamos enfatizar o método proposto por Dai–Yuan. Em um trabalho recente, Lucambio Pérez e Prudente [12], estenderam os métodos de gradiente conjugado para o contexto de otimização vetorial. Para o método Dai–Yuan, foi necessária uma modificação no parâmetro algorítmico para demonstrar a convergência do método. Essa modificação foi denominada *método GC Dai–Yuan modificado*. O parâmetro, intitulado β_k^{mDY}, é definido por:

$$\beta_k^{mDY} = \frac{\|g_{k+1}\|^2}{g_{k+1}^T d_k - \tau_k g_k^T d_k}, \tag{4-1}$$

de modo que $\{\tau_k\}_{k\geq 1}$ é uma sequência real, tal que $\tau_k > 1$. Observe que $\tau_k = 1\ \forall k$, recupera o parâmetro original. Vale ressaltar que, sem a modificação, ainda não foi possível mostrar a convergência do método DY para contexto vetorial, permanecendo um problema em aberto.

A ênfase nesse método é devida ao interesse em investigar o desempenho numérico, no contexto escalar, da modificação proposta por Lucambio Pérez

e Prudente, que trataremos no Capítulo 5. Neste capítulo vamos investigar as propriedades teóricas do método DY modificado, no caso escalar.

4.1 Análise teórica

Nesta seção vamos investigar as propriedades teóricas do método DY modificado. Mostraremos, inicialmente, que o parâmetro está bem definido. Para isso devemos verificar se o Algoritmo 2, com o novo parâmetro (4-1), garante que o denominador de β_k^{mDY} é diferente de zero e as direções geradas a cada iteração são de descida. Diferentemente do caso vetorial, tomaremos $\tau_k \geq 1$. Assim, os resultamos teóricos demonstrados nesta seção, generalizam os resultados para o método DY clássico. Para os resultados de convergência, iremos supor que $g_k \neq 0 \ \forall k$. O passo α_k será calculado de forma a satisfazer as condições padrões de Wolfe, (3-2) e (3-4).

Lema 4.1 *Considere o Algoritmo 2 com β_{k-1} definido por (4-1) e α_k satisfazendo as condições (3-2) e (3-4), então o algoritmo está bem definido.*

Demonstração. Vamos provar por indução sobre k que $\beta_k^{mDY} > 0$ e $g_k^T d_k < 0$. Para $k = 1$ temos:

$$d_1^T g_1 = -g_1^T g_1 = -\|g_1\|^2 < 0.$$

Agora a condição padrão de Wolfe (3-4) nos dá:

$$g_2^T d_1 > \sigma g_1^T d_1,$$

em que $0 < \sigma < 1$ e $g_1^T d_1 < 0$. Assim,

$$g_2^T d_1 - \tau_1 g_1^T d_1 > \sigma g_1^T d_1 - \tau_1 g_1^T d_1 = (\sigma - \tau_1) g_1^T d_1 > 0.$$

Portanto o denominador de β_1^{mDY} é positivo. Suponha agora que d_k seja direção de descida para algum $k > 1$, isto é:

$$g_k^T d_k < 0.$$

Pela condição padrão de Wolfe (3-4) temos

$$g_{k+1}^T d_k > \sigma g_k^T d_k,$$

logo,

$$g_{k+1}^T d_k - \tau_k g_k^T d_k > \sigma g_k^T d_k - \tau_k g_k^T d_k = (\sigma - \tau_k) g_k^T d_k > 0.$$

Note que o denominador de β_k^{mDY} é positivo, consequentemente $\beta_k^{mDY} > 0$. Agora

$$d_{k+1} = -g_{k+1} + \beta_k^{mDY} d_k \tag{4-2}$$

está bem definido. Fazendo o produto interno de (4-2) com g_{k+1}, obtemos:

$$d_{k+1}^T g_{k+1} = -\|g_{k+1}\|^2 + \beta_k^{mDY} d_k^T g_{k+1}.$$

Por (4-1) temos:

$$\begin{aligned} d_{k+1}^T g_{k+1} &= -\|g_{k+1}\|^2 + \frac{\|g_{k+1}\|^2}{g_{k+1}^T d_k - \tau_k g_k^T d_k} d_k^T g_{k+1} \\ &= \frac{-\|g_{k+1}\|^2 (g_{k+1}^T d_k - \tau_k g_k^T d_k) + \|g_{k+1}\|^2 d_k^T g_{k+1}}{g_{k+1}^T d_k - \tau_k g_k^T d_k} \\ &= \frac{\tau_k \|g_{k+1}\|^2 d_k^T g_k}{g_{k+1}^T d_k - \tau_k g_k^T d_k} \end{aligned}$$

Portanto

$$d_{k+1}^T g_{k+1} = \tau_k \beta_k^{mDY} d_k^T g_k. \tag{4-3}$$

Como $\tau_k > 1$ e $\beta_k^{mDY} > 0$, a hipótese de indução nos dá:

$$d_{k+1}^T g_{k+1} < 0,$$

como queríamos demonstrar.

■

De (4-3), obtemos uma nova formulação para β_k^{mDY} que será usada posteriormente:

$$\beta_k^{mDY} = \frac{d_{k+1}^T g_{k+1}}{\tau_k d_k^T g_k}. \tag{4-4}$$

Se o Algoritmo 2, com o parâmetro β_{k-1} definido como em (4-1) e o passo α_k satisfazendo as condições padrões de Wolfe, gera uma sequência de direções d_k, então vale condição de Zoutendijk (3-9). No resultado a seguir, veremos que a condição de Zoutendijk é um importante resultado para provar a convergência do método.

Teorema 4.1 *Suponha que x_1 é um ponto inicial conforme (i) nas Hipóteses 1. Seja $\{x_k\}_{k\geq 1}$, a sequência gerada pelo Algoritmo 2, com β_{k-1} definido como em (4-1) e o passo α_k satisfazendo (3-2) e (3-4). Então o Algoritmo 2, ou termina em um ponto estacionário*

ou converge globalmente no seguinte sentido:

$$\liminf_{k\to\infty} \|g_k\| = 0.$$

Demonstração. Suponha que o algoritmo não termina em um número finito de iterações, logo

$$\|g_k\| > 0 \ , \forall k \geq 1.$$

Pela definição de d_{k+1} temos

$$\langle d_{k+1} + g_{k+1}, d_{k+1} + g_{k+1} \rangle = \langle \beta_k^{mDY} d_k, \beta_k^{mDY} d_k \rangle,$$

assim

$$\|d_{k+1}\|^2 + 2d_{k+1}^T g_{k+1} + \|g_{k+1}\|^2 = (\beta_k^{mDY})^2 \|d_k\|^2.$$

Dividindo a igualdade anterior por $(g_{k+1}^T d_{k+1})^2 > 0$, obtemos

$$\frac{\|d_{k+1}\|^2}{(g_{k+1}^T d_{k+1})^2} = \frac{(\beta_k^{mDY})^2 \|d_k\|^2}{(g_{k+1}^T d_{k+1})^2} - \frac{2}{g_{k+1}^T d_{k+1}} - \frac{\|g_{k+1}\|^2}{(g_{k+1}^T d_{k+1})^2}.$$

Por (4-4), temos

$$\frac{\|d_{k+1}\|^2}{(g_{k+1}^T d_{k+1})^2} = \frac{\|d_k\|^2}{\tau_k^2 (g_k^T d_k)^2} - \left(\frac{1}{\|g_{k+1}\|} + \frac{\|g_{k+1}\|}{g_{k+1}^T d_{k+1}} \right)^2 + \frac{1}{\|g_{k+1}\|^2}.$$

Portanto

$$\frac{\|d_{k+1}\|^2}{(g_{k+1}^T d_{k+1})^2} \leq \frac{\|d_k\|^2}{\tau_k^2 (g_k^T d_k)^2} + \frac{1}{\|g_{k+1}\|^2},$$

Assim, de modo recursivo,

$$\begin{aligned} \frac{\|d_k\|^2}{(g_k^T d_k)^2} &\leq \frac{\|d_{k-1}\|^2}{\tau_k^2 (g_{k-1}^T d_{k-1})^2} + \frac{1}{\|g_k\|^2} \leq \\ &\leq \left(\frac{\|d_{k-2}\|^2}{\tau_k^2 (g_{k-2}^T d_{k-2})^2} + \frac{1}{\|g_{k-1}\|^2} \right) \frac{1}{\tau_k^2} + \frac{1}{\|g_k\|^2} \leq \cdots \leq \\ &\leq \frac{\|d_1\|^2}{(g_1^T d_1)^2 \tau_k^{2(k-1)}} + \sum_{j=2}^{k} \frac{1}{\|g_j\|^2 \tau_k^{2(k-j)}}. \end{aligned}$$

Assim

$$\frac{\|d_k\|^2}{(g_k^T d_k)^2} = \sum_{j=1}^{k} \frac{1}{\|g_j\|^2 \tau_k^{2(k-j)}}, \tag{4-5}$$

pois $d_1 = -g_1$. Como o algoritmo não termina em número finito de iterações,

então existe uma constante positiva c tal que

$$\|g_k\| \geq c > 0, \ \forall k \geq 1.$$

Por (4-5), temos

$$\frac{\|d_k\|^2}{(g_k^T d_k)^2} \leq \sum_{j=1}^{k} \frac{1}{c^2 \tau_k^{2(k-j)}} \leq \sum_{j=1}^{k} \frac{1}{c^2} = \frac{k}{c^2}$$

o que implica:

$$\sum_{k\geq 1} \frac{(g_k^T d_k)^2}{\|d_k\|^2} = \infty$$

contrariando a condição de Zoutendijk (3-9), concluindo a demonstração.

■

CAPÍTULO 5

Experimentos numéricos

No capítulo anterior desenvolvemos a base teórica do método DY modificado, mostrando a convergência do método para o contexto escalar. Nesse capítulo, investigaremos a escolha do parâmetro τ_k no método DY modificado, do ponto de vista numérico. Trataremos da implementação do método GC não-linear, com parâmetro β_k^{mDY} e passo α_k satisfazendo as condições padrões de Wolfe.

5.1 Implementação

O linguagem escolhida para implementar o método foi Julia, veja [3]. Julia é uma linguagem livre, recentemente desenvolvida no *Massachusetts Institute of Technology – MIT*. Além de se tratar de uma linguagem livre, destacamos que existem vários pacotes disponíveis para aplicações diversas. Um pacote fundamental para a implementação é a biblioteca de função para testes *CUTEst* [10], utilizando a interface com Julia desenvolvida em [22]. Tendo em vista a necessidade de fazer comparações entre os algoritmos, essa biblioteca contém problemas de otimização restrita e irrestrita. A biblioteca *CUTEst* é amplamente usada para performar testes computacionais na área de otimização.

O código da implementação está dividido em quatro sub-rotinas, dessas, três são de nossa autoria. A quarta foi retirada e adaptada do pacote *LineSearches.jl*, vide [20]. Chamada de *morethuente.jl*, essa sub-rotina, escrita originalmente em *Fortran*, é a tradução para Julia do algoritmo de Moré e Thuente, apresentado no Capítulo 3. Originalmente, a sub-rotina é implementada para encontrar um passo que satisfaça as condições fortes de Wolfe, (3-2) e (3-5). Para nossa implementação, modificamos a busca para condições padrões de Wolfe, (3-2) e (3-4). Para tal, mudamos o critério de parada da sub-rotina. De modo que, para cada passo computado, testamos se esse passo satisfaz as condições padrões de Wolfe. Vale ressaltar, que se um passo satisfaz as condições fortes, então também satisfaz as condições padrões de Wolfe.

No Apêndice A, encontram-se as três sub-rotinas por nós implementadas. A sub-rotina *myfunction.jl* é responsável por avaliar a função objetivo e seu gradiente, também a função unidimensional da busca linear juntamente com sua derivada. A sub-rotina *gradiente.jl*, é a própria implementação do Algoritmo 2 com o parâmetro β_{k-1} definido por (4-1) e o passo α_k satisfazendo (3-2) e (3-4). Na implementação desse algoritmo, usamos a sequência $\{\tau_k\}_{k\geq 1}$ constante. Note que se $\tau_k = 1,\ \forall k \geq 1$, então temos o método DY original. Além disso, definimos alguns parâmetros para a sub-rotina, como as constantes das condições de Wolfe, $\delta = 10^{-4}$ e $\sigma = 0.9$. Nessa sub-rotina, também informamos os critérios de parada. Se na k-ésima iteração ocorrer

$$\frac{\|g_k\|}{\|g_1\|} \leq 10^{-6},$$

o algoritmo para e temos sucesso. Se o número máximo de iterações ultrapassar quinhentas vezes o número de variáveis, o algoritmo para e temos fracasso.

Outra importante função dessa sub-rotina é acionar o método de busca linear, *morethuente.jl*. Para tal, a sub-rotina deve fornecer alguns parâmetros essenciais, e entre eles, está o passo tentativa inicial. Em nossa implementação usamos o passo recomendado por Shanno e Phua em [21], definido por:

$$\alpha_{ini}^k = \begin{cases} \dfrac{1}{\|g_k\|} & \text{se } k = 1, \\ \alpha_{k-1}.\dfrac{d_{k-1}^T g_{k-1}}{d_k^T g_k} & \text{caso contrário.} \end{cases} \tag{5-1}$$

Note que, α_{k-1} é o passo inicial na iteração anterior. Um cuidado que tivemos ao definir esse passo inicial, foi evitar valores muito grandes ou pequenos. Para isso, além da escolha acima, nós definimos o passo da seguinte maneira:

$$\alpha_{ini} = \min\{\max\{\alpha_{ini}^k, 10^{-2}\}, 10^2\}.$$

Quando *gradiente.jl* aciona a busca, a sub-rotina *morethuente.jl* pode retornar sucesso ou fracasso. Caso seja sucesso, a iteração continua. Quando é fracasso, temos dois possíveis cenários: (I) Um erro de execução que fez a busca falhar, consequentemente não é possível continuar iterando; (II) O cumprimento de um critério de parada que não seja a detecção de um ponto satisfazendo as condições de Wolfe, situação que pode ser remediada reiniciando o método. Neste último caso, o algoritmo toma a direção de máxima descida na próxima iteração.

A última sub-rotina é *main.jl*, responsável por fazer a implementação funcionar. Nela nós acionamos alguns pacotes básicos: os mais importantes são

CUTEst e *NLPModels*. Esse pacotes são necessários para acessar a biblioteca de problemas testes que utilizamos. Com essa biblioteca, temos acesso ao valor da função e de seu gradiente, em qualquer ponto do domínio. Essa coleção também fornece, para cada problema, o ponto inicial. Selecionamos para os testes numéricos, todos os problemas irrestritos da coleção *CUTEst*, totalizando 234 problemas. Dentre esses, o problema com maior dimensão tem 192627 variáveis, porém nenhum τ_k testado foi capaz de resolvê-lo.

5.2 *Performance Profiles*

A comparação entre os métodos implementados foi através de *performance profiles*. Esse tipo de análise foi proposta por Dolan e Moré em [6]. O *performance profile* possibilita a comparação simultânea de diversos algoritmos quando aplicamos a um conjunto de problemas. A fim de estabelecer um critério de comparação, usaremos o tempo de processamento e o número avaliações de função, demandados para resolver os problemas. Para cada valor de τ_k adotado, consideramos um método distinto.

Considere que tenhamos um conjunto de métodos e queremos determinar qual deles é o melhor. Se aplicarmos esses métodos em um único problema, o melhor será aquele que resolveu mais rapidamente, ou com o menor número de avaliações. Essa análise, mesmo que simples, deixa de ser viável quando o número de problemas cresce. Afinal, podemos ter um método superior para o problema A, mas inferior no problema B. A análise que o *performance profile* propõe é capaz de fornecer dados que permitem a comparação.

Para fixarmos ideia, vamos considerar que estamos interessados em classificar os métodos tomando o tempo de processamento como medida de desempenho. Assim, para cada problema, selecionamos qual dos métodos resolveu com menor tempo e dividimos os tempos demandados por cada método, por este melhor tempo. Por exemplo, para o problema A, se o menor tempo demandado foi 10 segundos, dividiremos por 10, todos os tempos gastos por cada método para resolver o problema A. Logo, ao melhor método é atribuído o valor 1 e aos demais, valores proporcionais. Procedendo dessa maneira para todos os problemas, podemos calcular a porcentagem de problemas resolvidos por um método, para um determinado valor atribuído. O número de problemas que um método resolveu, com valor atribuído 1, nos informa a eficiência desse método. Tomando o método que resolveu maior número de problemas, com valor atribuído 1, temos o mais eficiente dos métodos. Se o intuito é buscar um método que soluciona o máximo de problemas, então estamos interessados na robustez do método. Nesse

caso, devemos olhar para o extremo direito dos gráficos de *performance profiles*. Vale ressaltar que adicionamos um critério de empate para comparação de eficácia. Se um tempo é igual a até 105% do melhor, então nossa comparação considera empate.

Em nossa implementação, testamos dezessete parâmetros τ_k distintos. Dentre esses, $\tau_k = 1$, que recupera o método original DY. Inicialmente, usamos o tempo máximo de um minuto para cada problema. Assim, coletamos os dados e aplicamos o *performance profiles*. Os dados que obtemos, Tabela 5.1, sugerem que quando τ_k cresce muito, o método perde eficiência e robustez. Selecionamos os valores para τ_k que obtiveram melhor eficiência e robustez para melhor análise. Realizamos novos testes com um maior tempo limite, 15 minutos para cada problema. Na Tabela 5.2, está reportado eficiência e robustez dos τ_k selecionados, incluindo o original.

	Eficiência		Robustez
τ_k	Tempo(%)	Avaliações(%)	–
1.0	25.6410	23.0769	67.0940
1.01	26.4957	22.2222	75.2137
1.02	26.0684	22.6496	72.6496
1.03	18.3761	13.6752	73.9316
1.04	20.5128	15.3846	74.3590
1.05	16.6667	13.6752	71.7949
1.06	18.3761	12.3932	73.9316
1.065	15.8120	11.5385	72.2222
1.07	15.8120	12.8205	72.2222
1.075	16.2393	13.2479	71.7949
1.08	17.9487	13.2479	69.6581
1.09	13.2479	12.3932	68.8034
1.1	15.8120	12.8205	68.3761
1.2	12.8205	11.9658	65.3846
1.3	12.8205	13.2479	64.1026
1.4	11.1111	11.9658	61.5385
2.0	12.3932	12.8205	54.7009

Tabela 5.1: *Eficiência e robustez dos métodos, com tempo máximo de 60 segundos para cada problema.*

Os gráficos nas Figuras 5.1 a 5.4, mostram os *performance profiles*. No eixo das abscissa, temos os valores atribuídos a cada método. No eixo das ordenadas, temos a porcentagem de problemas resolvidos por um método, para cada valor atribuído. Nas Figuras 5.2 e 5.4, diminuímos o eixo das abscissas para melhor visualização da eficiência dos métodos que obtiveram melhor desempenho, segundo a Tabela 5.1. Analisando as Figuras 5.1 a 5.4 e a Tabela 5.2, podemos concluir que $\tau_k = 1.01$ é a melhor escolha para o parâmetro. Pela Tabela 5.2, no-

	Eficiência		Robustez
τ_k	Tempo(%)	Avaliações(%)	–
1.0	29.4872	29.4872	70.9402
1.01	33.3333	31.1966	79.9145
1.03	28.2051	22.2222	77.3504
1.04	25.2137	23.9316	77.7778
1.06	28.6325	29.0598	76.4957

Tabela 5.2: *Eficiência e robustez dos métodos, com tempo máximo de 900 segundos para cada problema.*

tamos que essa escolha conduz o método a maior eficiência e robustez. Dentre os problemas não resolvidos pelo método com parâmetro $\tau_k = 1.01$, 8.11% foi por exceder o número máximo de interações, 6.41% foi por exceder o tempo máximo de 15 minutos e os 5.6% devido a um erro no código *morethuente.jl*.

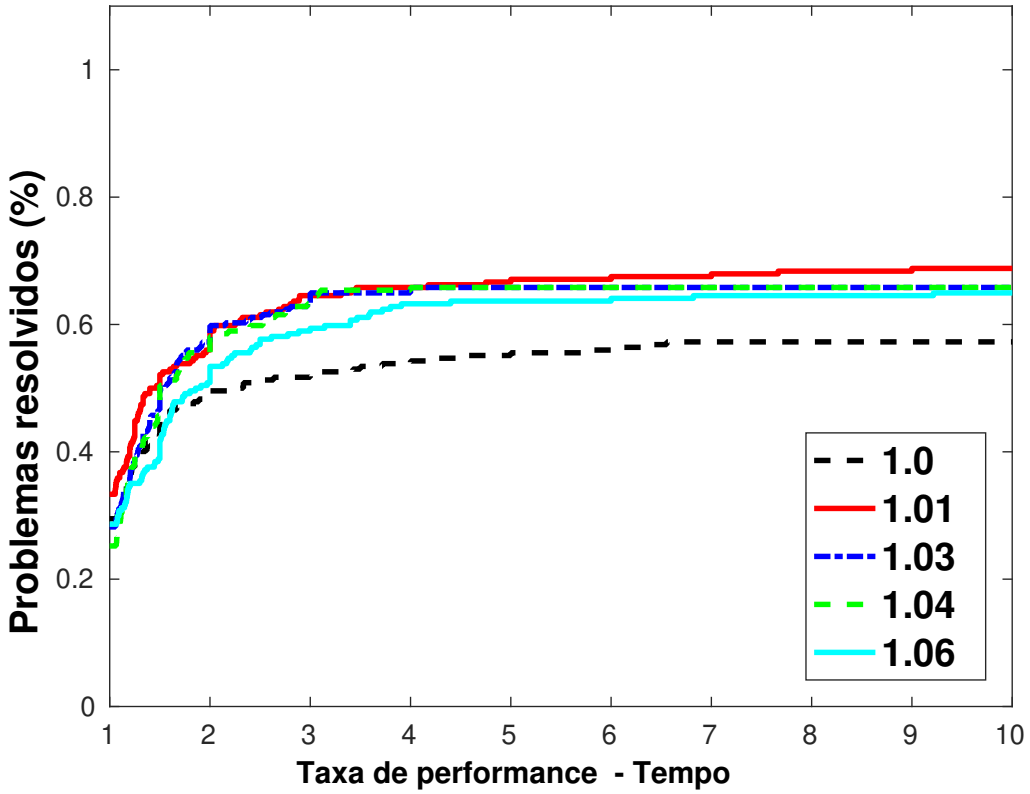

Figura 5.1

A escolha de $\tau_k = 1.01$ do método modificado torna o método de DY, que é um método clássico, mais competitivo. Além de estar bem definido no sentido vetorial, como mostrado em [12], no contexto escalar foi possível melhorar o desempenho numérico do método, dentro conjunto de problemas testes da biblioteca *CUTEst*.

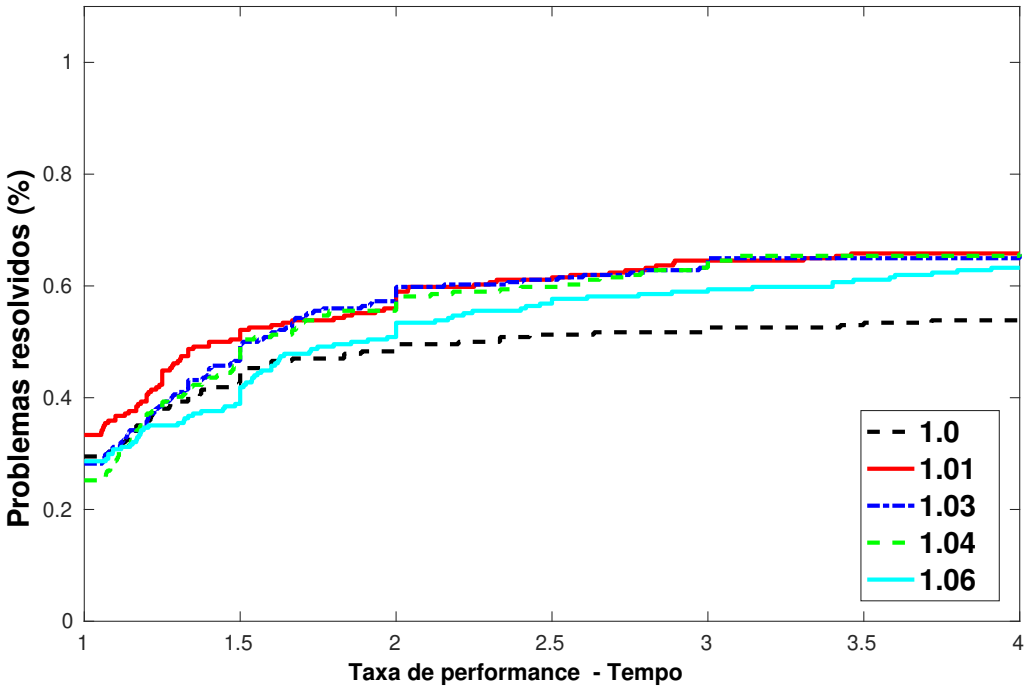

Figura 5.2

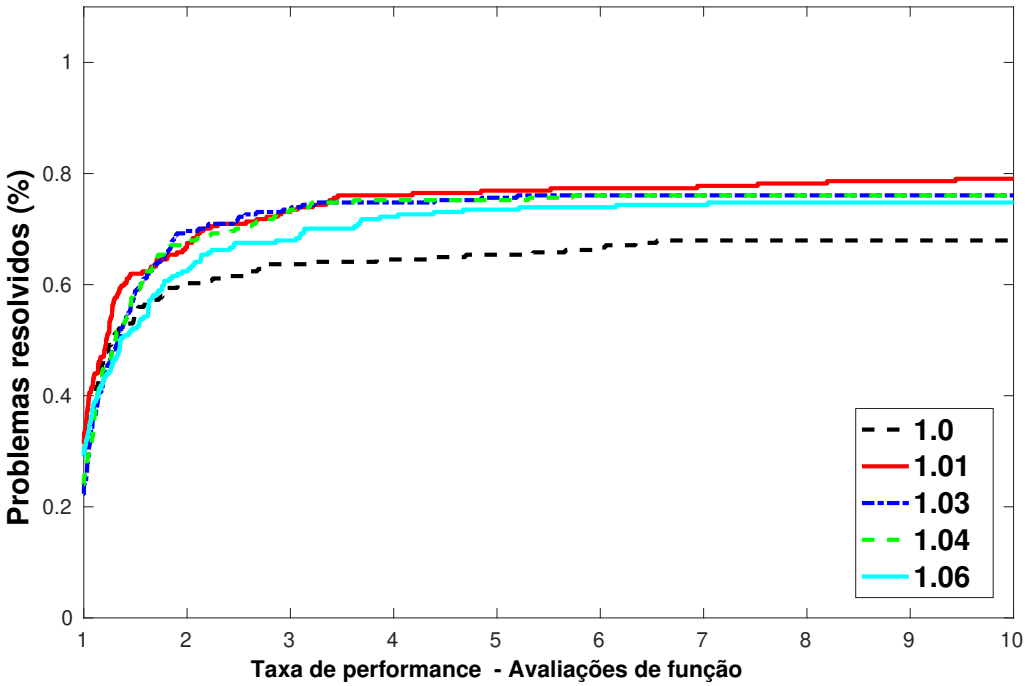

Figura 5.3

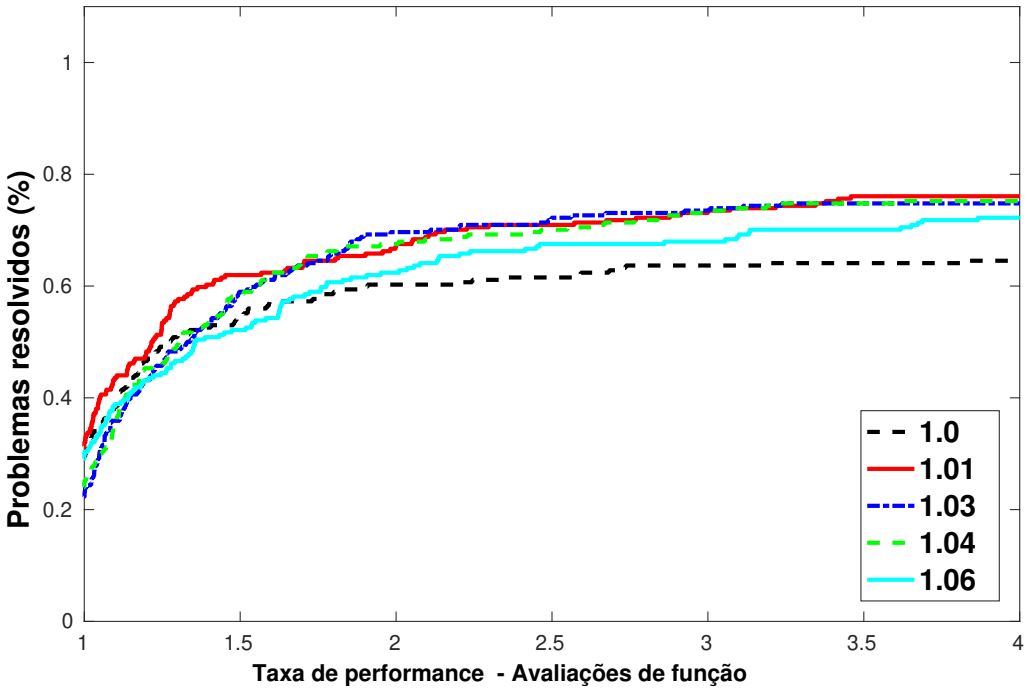
Problemas resolvidos (%)
0
0.2
0.4
0.6
0.8
1
1
1.5
2
2.5
3
3.5
4
Taxa de performance - Avaliações de função
1.0
1.01
1.03
1.04
1.06

Figura 5.4

CAPÍTULO 6

Conclusão e considerações finais

Neste trabalho, investigamos o método DY modificado no contexto de otimização escalar. Mostramos que o método DY modificado, usando passos que satisfazem as condições padrões Wolfe, está bem definido e é globalmente convergente. Testes numéricos mostraram que o método DY modificado, com parâmetro $\tau_k = 1.01$, pode ser considerado, no conjunto de problemas da biblioteca *CUTEst*, mais eficiente e robusto do que a versão original. Dessa maneira, o trabalho aqui generaliza a teoria de convergência do método Dai–Yuan e potencializa sua aplicabilidade.

Finalmente, como trabalho futuro, deixamos a possibilidade de formulação de um método DY adaptativo, no qual o parâmetro τ_k é escolhido a cada iteração.

APÊNDICE A

Código

O código da implementação está dividido em quatro sub-rotinas, dessas, três são de nossa autoria. A quarta foi retirada e adaptada de um pacote de otimização, vide [20]. Nós modificamos essa sub-rotina, para que a busca encontre passo satisfazendo as condições padrões de Wolfe, (3-2) e (3-4). Neste apêndice, encontram-se as três sub-rotinas que criamos.

A.1 myfunction.jl

```
#################################################################
function evalf(x)
    global nfev,nlp,sf

    nfev += 1
    f = obj(nlp,x)
    f = sf * f
    return f
end
#################################################################

function evalg(x)
    global ngev,nlp,sf

    ngev += 1
    g = grad(nlp,x)
    g = sf .* g
    return g
end

#################################################################

function evalphi(alpha)
    global x, d
```

```
    evalf( x + alpha * d )
end

################################################################

function evaldphi(alpha)
    global x, d

    g = evalg( x + alpha * d )
    return dot(g,d)
end

################################################################

function evalphidphi(alpha)

    phi  = evalphi(alpha)
    dphi = evaldphi(alpha)

    return (phi,dphi)
end
```

A.2 gradiente.jl

```
function mDY(tau::T) where T

    global x,d,nfev,ngev,nlp,sf,timelim,maxtime,start

    # Start timing
    start = time()

    # Define initial parameters
    tol::T     = 1e-6
# c_1 Wolfe sufficient decrease condition:
    ftol::T    = 1e-4
# c_2 Wolfe curvature condition (Recommend 0.1 for GradientDescent):
    gtol::T    = 0.9
    itmax::Int = 500 * nlp.meta.nvar
    scale      = true
    timelim    = true
# Maximum time for each problem
    maxtime    = 900.0
    #maxtime    = 60.0
```

```
# Set the initial point
x = nlp.meta.x0

# Initialize arrays
dprev = similar(x)
gprev = similar(x)

# Counters
it   = 0
nfev = 0
ngev = 0

# Evaluate the objective function and its gradient
f = obj(nlp,x)
g = grad(nlp,x)
nfev += 1
ngev += 1

# Calculate the infity norm of the gradient
normg = norm(g,Inf)

# Define the scale factor
if scale
    sf = 1.0 / max(1.0,normg)
    f = sf * f
    g = sf .* g
else
    sf = 1.0
end

#-------------------------------------------
# Main Loop
#-------------------------------------------

while true
    global infoMT,beta,alpha

    # Print information
    if mod(it,10) == 0
        @printf("\n %-6s %-11s %-9s %-2s\n","it","f","normg","LS")
    end

    if it == 0
        @printf("%-6d %11.4e %9.6f %2s\n",it, f, normg,"-")
    else
        @printf("%-6d %11.4e %9.6f %2d\n",it, f, normg,infoMT)
```

```
    end

#-------------------------------------------------------------
# Test the stopping criterea
#-------------------------------------------------------------

    # Test optimality
    if normg <= tol

        # Stop timing
        tempo = time() - start
        solinfo = 0

        println( "\n Solution was found")
        @printf("\n Number of iterations: %-5d",it)
        @printf("\n Number of function evaluations:%-10d",nfev)
        @printf("\n Number of gradient evaluations:%-10d\n",ngev)
        @printf("\n Total CPU time in seconds:%8.2f\n",tempo)
        return x,it,solinfo,nfev,ngev,tempo
    end

    # Test whether the number of iterations is exhausted
    if it >= itmax

        # Stop timing
        tempo = time() - start

        solinfo = 2
        println("\n Maximum of allowed iterations reached\n")
        @printf("\n Number of iterations: %-5d",it)
        @printf("\n Number of function evaluations:%-10d",nfev)
        @printf("\n Number of gradient evaluations:%-10d\n",ngev)
        @printf("\n Total CPU time in seconds:%8.2f\n",tempo)
        return x,it,solinfo,nfev,ngev,tempo
    end

    if timelim
       tempo = time() - start
       if tempo > maxtime
          solinfo = 3
          println("\n Maximum of allowed time reached\n")
          @printf("\n Number of iterations:%-5d",it)
          @printf("\n Number of function evaluations:%-10d",nfev)
          @printf("\n Number of gradient evaluations:%-10d\n",ngev)
          @printf("\n Total CPU time in seconds:%8.2f\n",tempo)
          return x,it,solinfo,nfev,ngev,tempo
```

```
    end
end

#-------------------------------------------------------------
# Prepare the iteration
#-------------------------------------------------------------

# Check for restart
restart = false
if it > 0 && infoMT != 1
    restart = true
end

# Update the conjugate parameter beta
if it > 0 && !restart
    beta = dot(g,g) / (dot(dprev,g) - tau * dot(dprev,gprev))
end

# Define the search direction
if it == 0 || restart
    d = copy(-g)
else
    d .= - g .+ beta .* d
end

#-------------------------------------------------------------
# Iterate
#-------------------------------------------------------------

#-------------------------------------------------------------
# Compute the initial trial step size
# by the Shanno and Phua recommendation
#-------------------------------------------------------------
dg = dot(d,g)

if it == 0
    alpha = 1.0 / normg
    alpha = max( alpha, 1e-2 )
    alpha = min( alpha, 1e+2 )
else
    alpha = alpha * dot(dprev,gprev) / dg
    alpha = max( alpha, 1e-2 )
    alpha = min( alpha, 1e+2 )
end

# Increment the iteration counter
```

```
        it += 1

        # Call the linesearch
        alpha, infoMT = MoreThuente(alpha,f,dg,ftol,gtol)

        # Check for error in MoreThuete code
        if infoMT == 6
            tempo = time() - start
            solinfo = 4
            println("\n Error in MoreThuete code\n")
            return x,it,solinfo,nfev,ngev,tempo
        end

        # Save previous data
        dprev = copy(d)
        gprev = copy(g)
        # Update x
        x .= x .+ alpha .* d

        # Calculate the objective function at the new iterate
        f = evalf(x)

        # Calculate the gradient at the new iterate
        g = evalg(x)

        # Calculate the infity norm of the gradient
        normg = norm(g,Inf)
    end

end
```

A.3 main.jl

```
#-------------------------
#Basic packages
#-------------------------

using CUTEst, Printf
using NLPModels
using LinearAlgebra

#-------------------------
#Subroutines
#-------------------------
```

```
include("myfunction.jl")
include("gradiente.jl")
include("morethuente.jl")

function main()

    global nlp

    #-----------------------------
    #Files directory
    #-----------------------------
    maindir = "/home/danilo/Dropbox/Danilo-Leandro/Danilo/"
    taudir = "/home/danilo/Dropbox/Danilo-Leandro/Danilo/tau"

    #-----------------------------
    #Select the tau values
    #-----------------------------
    tau_choices = [1.0,1.01,1.03,1.04,1.06]
    numtau = length(tau_choices)

    #-----------------------------------
    #Selects unrestricted CUTEst library issues
    #-----------------------------------
    problems = CUTEst.select(contype="unc")
    numprob = length(problems)


    for j = 1:numtau
        tau = tau_choices[j]
        outfile = "$tau"
        cd(taudir)

        if !isfile(outfile)
                cd(maindir)
                fid = open(outfile,"w")
        else
                @printf("File %10s already exists,
                                number = %3d\n",outfile,j)

                @goto escape_label
        end
        for i = 1:numprob


                nlp = CUTEstModel(problems[i])
```

```
                @printf("Problem: %10s \n\n",problems[i])

                x, it, solinfo , nfev, ngev, tempo = mDY(tau)

                #-----------------------------------------------------
                #Write the data in the files for each tau value
                #-----------------------------------------------------
                @printf(fid,"%.2f %1d %10d %20d %8.2f\n",
                                tau,solinfo,it,nfev,tempo)


                finalize(nlp)

        end

        run(`mv $outfile $taudir`)
        close(fid)
        @label escape_label
    end
end
```

Referências Bibliográficas

[1] AL-BAALI, M. **Descent Property and Global Convergence of the Fletcher-Reeves Method with Inexact Line Search**. *IMA Journal of Numerical Analysis*, 5(1):121–124, 1985.

[2] AL-BAALI, M.; FLETCHER, R. **An efficient line search for nonlinear least squares**. *Journal of Optimization Theory and Applications*, 48(3):359–377, Mar 1986.

[3] BEZANSON, J.; EDELMAN, A.; KARPINSKI, S.; SHAH, V. B. **Julia: A fresh approach to numerical computing–https://julialang.org/research/**. *SIAM Review*, 59:65–98, 2017.

[4] DAI, Y. H.; .YUAN, Y. **Convergence properties of the conjugate descent method**. *Advances in Mathematics*, 25(6):552–562, 1996.

[5] DAI, Y. H.; YUAN, Y. **A Nonlinear Conjugate Gradient Method with a Strong Global Convergence Property**. *SIAM Journal on Optimization*, 10(1):177–182, 1999.

[6] DOLAN, E. D.; MORÉ, J. J. **Benchmarking optimization software with performance profiles**. *Mathematical programming*, p. 201–213, Oct. 2002.

[7] FLETCHER, R. **Practical Method of Optimization, Unconstrained Optimization, vol. 1**, 1980.

[8] FLETCHER, R.; REEVES, C. M. **Function minimization by conjugate gradients**. *The Computer Journal*, 7(2):149–154, 1964.

[9] GILBERT, J. C.; NOCEDAL, J. **Global Convergence Properties of Conjugate Gradient Methods for Optimization**. *SIAM Journal on Optimization*, 2(1):21–42, 1992.

[10] GOULD, N. I. M.; ORBAN, D.; TOINT, P. L. **https://github.com/ralna/cutest/wiki – CUTEst: A constrained and unconstrained testing environment with safe threads**.

[11] HESTENES, M. R.; STIEFEL, E. **Methods of conjugate gradients for solving linear systems**, volume 49. Journal of Research of the National Bureau of Standards, 1952.

[12] LUCAMBIO PÉREZ, L. R.; PRUDENTE, L. F. **Non-linear conjugate gradient methods for vector optimization**. *SIAM Journal on Optimization*, 2018.

[13] MORÉ, J. J.; THUENTE, D. J. **Line Search Algorithms with Guaranteed Sufficient Decrease**. *ACM Trans. Math. Softw.*, 20(3):286–307, Sept. 1994.

[14] NOCEDAL, J.; WRIGHT, S. **Numerical optimization**. Springer Science & Business Media, 2006.

[15] PANONCELI, D. M. **Um estudo de buscas unidimensionais aplicadas ao método BFGS**. p. 32–37, 2015.

[16] POLAK, E.; RIBIÈRE, G. **Note sur la convergence de méthodes de directions conjuguées**. *Revue française d'informatique et de recherche opérationnelle, série rouge*, 3(1):35–43, 1969.

[17] POLYAK, B. T. **The conjugate gradient method in extremal problems**. *USSR Computational Mathematics and Mathematical Physics*, 9(4):94 – 112, 1969.

[18] POWELL, M. J. D. **Nonconvex minimization calculations and the conjugate gradient method**. In: *Numerical analysis*, p. 122–141. Springer, 1984.

[19] RIBEIRO, A. A.; KARAS, E. W. **Um curso de otimização**. p. 75–77, 2011.

[20] RISETH, A. N. **https://github.com/julianlsolvers/linesearches.jl – This package provides an interface to line search algorithms implemented in julia**.

[21] SHANNO, D. F.; PHUA, K. H. **Remark on algorithm 500: Minimization of unconstrained multivariate functions**. *ACM Transactions on Mathematical Software*, 6(4):618–622, 1980.

[22] SIQUEIRA, A. S. **https://github.com/juliasmoothoptimizers/cutest.jl – CUTEst.jl: Julia's CUTEst interface**.

[23] SUN, W.; YUAN, Y.-X. **Optimization theory and methods: Nonlinear programming**. *Optimization and Its Applications - Springer*, p. 89 – 100, 2006.

[24] ZHU, C.; BYRD, R. H.; LU, P.; NOCEDAL, J. **Algorithm 778: L-BFGS-B: Fortran Subroutines for Large-scale Bound-constrained Optimization**. *ACM Trans. Math. Softw.*, 23(4):550–560, Dec. 1997.