Pontifícia Universidade Católica de Goiás

Escola de Ciência Exatas e da Computação

LICENCIATURA EM MATEMÁTICA

UMA INTRODUÇÃO A OTIMIZAÇÃO CONTÍNUA:

estudo dos métodos de *Polak-Ribiere* e de Newton truncado para a minimização de funções

JOÁS RODRIGUES DA MATA

GOIÂNIA, GO

2021

JOÁS RODRIGUES DA MATA

UMA INTRODUÇÃO A OTIMIZAÇÃO CONTÍNUA:

estudo dos métodos de *Polak-Ribiere* e de Newton truncado para a minimização de funções

Monografia apresentada a Escola de Ciências Exatas e da Computação como requisito parcial para obtenção do título de graduado em Matemática da Pontifícia Universidade Católica de Goiás.

Orientador: Prof. Dr. Adelino Candido Pimenta

GOIÂNIA – GO

2021

JOÁS RODRIGUES DA MATA

UMA INTRODUÇÃO A OTIMIZAÇÃO CONTÍNUA:

estudo dos métodos de *Polak-Ribiere* e de Newton truncado para a minimização de funções

Esta Monografia foi julgada adequada para obtenção do título de graduado em Matemática, e aprovado em sua forma final pela Escola de Ciências Exatas e da Computação, da Pontifícia Universidade Católica de Goiás, Em: ____/____/_____

_____
Prof. Dr. Adelino Candido Pimenta
Orientador

Banca Examinadora

_____
Prof. Dr. Adelino Candido Pimenta
Orientador

_____

Prof. Dra. Maria José Pereira Dantas
Banca

_____

Prof. Dra. Vanda Domingues Vieira
Banca

_____
Prof. Msc. Bercholina Honorato Alves
Suplente

Dedico este trabalho primeiramente à Deus, que mantém todas as coisas pelo poder da sua palavra. Também à meus pais e minha família, pois são a rocha sob a qual me sustento.

AGRADECIMENTOS

Agradeço a minha família que tanto me ajudou neste percurso de quase 4 anos de estudo, pois são o sustento sob o qual me apoio.

Agradeço também à professora Maria José Pereira Dantas que, durante meu período de iniciação cientifica me orientou com sua melhor disposição e me fez tomar gosto pelo assunto tratado neste trabalho. E também aos amigos que fiz no laboratório de estatística e modelagem matemática da PUC-GO, o qual a professora Maria José coordena com tanto cuidado.

Aos diversos professores, amigos e professores amigos que fiz durante meu período na PUC Goiás. Destes amigos, ressalto duas que muito me apoiaram, Nadya e Jordanna.

“O sentimentalismo é o progenitor, o avô e a parteira da brutalidade” (Theodore Dalrymple)

“...todas as coisas estão cheias de cansaço; ninguém o pode exprimir: os olhos não se fartam de ver, nem os ouvidos se enchem de ouvir. O que tem sido isso é o que há de ser; e o que se tem feito, isso se tornará a fazer; nada há que seja novo debaixo do sol. Há alguma coisa de que se possa dizer: Vê, isto é novo? ela já existiu nos séculos que foram antes de nós. Já não há lembrança das gerações passadas; nem das gerações futuras haverá lembrança entre os que virão depois delas.” (Eclesiastes 1:7-11).

Resumo

Nesta monografia tratamos do tema de otimização continua fazendo um estudo dos métodos de ordem um, Polak-Ribière, e de ordem dois, Newton truncado. Buscou-se entender com este trabalho as vantagens de cada um dos dois métodos quando utilizados para minimizar alguns tipos de funções multidimensionais presentes na literatura de otimização, assim como pretendeu-se propiciar uma introdução não tão formal ao assunto da otimização, mais especificamente o caso da minimização. Primeiramente foi conduzido um estudo bibliográfico dos fundamentos de otimização contínua, objetivando revisar o ferramental matemático necessário ao entendimento tema proposto. Em seguida foi conduzido outro estudo bibliográfico detalhado os principais aspectos algorítmicos e matemáticos dos dois métodos supramencionados. Por fim, foi conduzido um teste onde simulamos a minimização das mesmas 5 funções para ambos os métodos buscando comparar seus desempenhos, utilizando para isso a implementação destes métodos oferecida pela biblioteca S*cipy* da linguagem de programação Python. As métricas utilizadas para esta comparação, que são oferecidas no final da iteração pela própria biblioteca Scipy, foram *número de testes da função*, *número de iteração do método*, *número de testes do jacobiano.* Os resultados apontaram para um melhor desempenho do método de Polak-Ribière em três dos cinco casos das funções propostas, porém o método Newton Trunca se mostrou também muito eficiente na entrega da solução. Concluímos com este trabalho que o assunto otimização, assim como os dois métodos específicos tratados aqui são uma boa iniciação para os interessados em otimização.

Palavras chave: Polak-Ribière, Newton truncado, Minimização, Python.

Abstract

In this monograph we deal with the theme of continuous optimization, making a study of the order one, Polak-Ribière, and order two, Newton truncated methods. This work sought to understand the advantages of each of the two methods when used to minimize some types of multidimensional functions present in the optimization literature, as well as providing a not so formal introduction to the subject of optimization, more specifically the case of minimization. Firstly, a bibliographic study of the fundamentals of continuous optimization was conducted, aiming to review the mathematical tools necessary to understand the proposed theme. Then, another detailed bibliographic study was conducted on the main algorithmic and mathematical aspects of the two methods mentioned above. Finally, a test was conducted where we simulated the minimization of the same 5 functions for both methods in order to compare their performance, using the implementation of these methods offered by the Scipy library of the Python programming language. The metrics used for this comparison, which are offered at the end of the iteration by the Scipy library, were number of function tests, method iteration number, number of Jacobian tests. The results pointed to a better performance of Newton's Truncated method in all cases of the proposed functions, but the Polak-Ribière method was also very fast in delivering the solution. We conclude with this work that the subject of optimization, as well as the two specific methods discussed here, are a good introduction for those interested in optimization.

Keywords: Polak-Ribière, Truncated Newton, Minimization, Python.

Lista de tabelas

Tabela 2.1- Lista de Símbolos sugestivos 17
Tabela 2.2 - Comparativo de função e sua aproximação de Taylor de 2° ordem – adaptado de (PARKINSON;BALLING; HEDENGREN, 2018) 25
Tabela 3.1 - Comparação dos algoritmos PR-CG e $PR+$ - Adaptada de Nocedal e Wright (2006, p.124) 33
Tabela 3.2 - Comparativo entre os métodos PR-CG e $PR+$. 33
Tabela 5.1 - Características das funções de teste 47
Tabela 5.2 - Pontos de mínimo e valores mínimos da função 53
Tabela 5.3 - Comparação dos métodos na minimização das funções. 53
Tabela 5.4 - Comparativo dos erros entre valor estimado (M_func) e valor teórico (M_teórico) 55

FIGURAS

Figura 1 - Gráfico Rosenbrook Function 49
Figura 2 - Curvas de nível da Rosenbrook Function..... 49
Figura 3 - Gráfico da Ackley 2 Function 50
Figura 4 - Curvas de nível da Ackley 2 Function 50
Figura 5 - Gráfico da Beale Function 51
Figura 6 - Curvas de nível da Beale Function 51
Figura 7 - Gráfico da Booth Funcition 52
Figura 8 - Curvas de nível da Booth Funcition 52

SUMÁRIO

1 INTRODUÇÃO 13

1.1. Reflexão antropológica sobre a otimização 14

2 FUNDAMENTOS MATEMÁTICOS DA OTIMIZAÇÃO CONTÍNUA 16

2.1. Notações usadas no texto 16

2.2. Vetor Gradiente 19

2.3. Algumas propriedades do vetor gradiente 19

2.4. Matriz hessiana 19

2.5. Matrizes simétricas definidas 20

2.6. Checando se uma matriz é positiva definida usando matrizes menores principais. 22

2.7. Série de Taylor 23

2.8. Exemplo de uma aproximação utilizando o polinômio de Taylor de 2° ordem. 24

2.9. Funções quadráticas 25

2.10. Aproximação de Taylor de 2° ordem para a função 25

2.11. Propriedades da função quadrática 26

2.12. Condições para otimização irrestrita 27

3 MÉTODOS DE ORDEM UM - POLAK-RIBIÈRE 29

3.1. Breve histórico dos métodos de direção conjugada 29

3.2. Direções conjugadas 29

3.3. Método Polak-Ribière 29

3.4. Exemplo 34

4 MÉTODO DE ORDEM DOIS: NEWTON TRUNCADO 38

4.1. Condição de Wolfe 38

4.2. Condição de Goldstein 39

4.3. Relação entre método de Newton-puro e Newton-CG (ou Método de Newton Truncado) 42

5 SIMULAÇÕES DE MINIMIZAÇÃO DE ALGUMAS FUNÇÕES COM OS MÉTODOS POLAK-RIBIÈRE E NEWTON TRUNCADO 47

6 Conclusão 57

REFERÊNCIAS 58

APENDICE 1 – CÓDIGO EM PYTHON DOS TESTES DE MINIMIZAÇÃO 59

APENDICE 2 – RESULTADOS BRUTOS DA EXECUÇÃO DO CÓDIGO 61

1 INTRODUÇÃO

Formalmente a otimização é campo de estudo da matemática, que permeia tanto o que se chama de matemática pura quanto matemática aplicada. É pura porque envolve conceitos de outros campos notoriamente tidos como abstratos, como a análise real, topologia e análise funcional, entre outros. É também matemática aplicada, pois se ocupa de, por meio de sua metodologia, de resolver problemas de cunho prático que encontramos no dia-a-dia. Porém, o fato é que direta ou indiretamente a otimização está presente em nosso meio, em problemas de engenharia, economia e na tomada de decisão (RIBEIRO; KARAS, 2013). Na maior parte dos problemas de otimização o que se pretende é minimizar ou maximizar certa variável. Pode ser, por exemplo a variável K=”kilometro por litro(km/l)", onde o óbvio seria maximizar K. Ou poderia ser R="gastos com reparos de caminhões", em que para uma empresa de logística terrestre, em exemplo, o objetivo seria minimizar K.

Quando se fala em otimizar é necessário destacar que existem diversos tipos de problemas de otimização, que são tipificados segundo características especificas da função que se pretende maximizar ou minimizar. Alguns tipos de otimização são: programação linear, otimização combinatória e otimização contínua, que é a tratada neste trabalho.

A otimização continua lida com problemas em que a variável de controle x pode assumir uma infinidade de valores em um dado intervalo I, no qual o problema esteja definido. Estes problemas são modelados como funções de uma ou várias variáveis, podendo estas funções ter diversas características que determinarão os métodos mais adequados de lidar com a busca do valor ótimo. Problemas que podem ser modelados como funções fortemente convexas e quadráticas, podem ser otimizados por métodos baseados no gradiente da função ou na matriz hessiana, que para esse tipo de problema apresentam alta eficácia, ou mais formalmente dizendo, uma convergência rápida.

1.1. Reflexão antropológica sobre a otimização

Se você pensa o mundo surgiu na revolução americana, francesa ou industrial. Se acredita que a grande evolução do ser humano foi proporcionada pela revolução sexual dos anos 60 e 70 , ou para alguns mais jovens, que tem fé que a revolução proporcionada pelos computadores e pela internet foi, até agora, o ápice do ser humano. Esqueça isso. Foi provavelmente a revolução agrícola ou revolução neolítica, como é conhecida na literatura especializada, que mudou a história da humanidade. Antes da ascensão da agricultura, o ser humano era uma espécie pouco diferente das demais no aspecto de aquisição de comida. É claro que nesse momento a revolução cognitiva do homem, como é conhecida, já havia acontecido por volta de 50 a 30 mil anos antes, mas não tocaremos nesse assunto aqui.

A prática de caçar e coletar frutos para se alimentar todos os dias era a regra para espécie homo sapiens, vindo dessa prática a nomenclatura caçador-coletor. Éramos nômades, andarilhos, pois quando uma região tinha suas fontes de alimentação exauridas, nos deslocávamos em busca de um novo local que propiciasse recursos para nossa sobrevivência. A partir de certo momento, a espécie humana passa a cultivar, claro, de um modo rustico, certas culturas de plantas que coletava e consumia diariamente. Com o passar do tempo, esse modelo cultivo primitivo evoluiu e se tornou mais complexo e especializado, com desenvolvimento de técnicas de irrigação e plantio seletivo, o que provavelmente levou ao aumento da produção de alimentos por área. Foi nesse período que o homem também começou a domesticar certos animais para utilização em serviço ou mesmo para sua alimentação. Com o desenvolvimento dessa prática agrícola, o homem passa a produzir seu próprio alimento, não precisando mais se deslocar em busca de novos ambientes em busca de recursos para sua sobrevivência. Essa transição feita de um comportamento nômade para um agrícola e sedentário fixo, foi o que tornou viável o crescimento populacional assimétrico que tivemos nos últimos milênios. A revolução agrícola, ao permitir o aumento da população e a fixação de tribos em um local e ali permanecer indefinidamente, foi o que impulsionou a criação de povos e civilizações, culminando também no desenvolvimento da escrita como meio de registro e transmissão de conhecimento e informações. Esse período, conhecido também com

transição demográfica neolítica em conjunto com a revolução cognitiva, foi com certeza os dois maiores saltos no desenvolvimento humano que tivemos.

Nós evoluímos no sentido de que não precisamos mais agir como caçadores-coletores, porém essa pratica esteve presente em todo o percurso do homo sapiens, antes mesmo da revolução cognitiva que permitiu nossa espécie se sobressair sobre todas os outros *homos*. Então por centenas de milhares de anos agimos assim caçando e coletando para se alimentar. Ou seja, mesmo após milênios de evolução, ainda existem traços nômades em nós, em nível psicológico. E o que isso tem a ver com o tema otimização, podemos perguntar. Tudo. Pois, como foi dito a otimização surge das necessidades encontradas no dia a dia.

Em uma pesquisa universidade de Genebra na Suíça conduzida pelo psicólogo Boris Cheval sobre os benefícios da atividade física na mente humana, foi constatado que mediante a alguma atividade nosso cérebro envia sinais ao nosso organismo para que ele economize energia. Concluíram que este é um dos motivos pelos quais a prática de exercícios físicos é tão difícil para pessoas que não estão acostumadas. Essa constatação explica muitas coisas, inclusive, o porquê da necessidade de se otimizar diversos aspectos e problemas que surgem no dia a dia.

Pense por exemplo no período da colheita do milho no início da revolução agrícola , o que os responsáveis pela plantação faziam para colher o milho no mínimo de dias possíveis. Aumentavam a quantidade de pessoas, se trabalhava mais horas por dia? As opções são muitas, mas o objetivo era o mesmo, diminuir os dias gasto na colheita. Como esse, existem vários exemplos de otimização que são comuns ou não ao nosso dia a dia, indo desde uma tarefa muito comum como reduzir o tempo gasto no banho e, indo a até a otimização de problemas complexos como a minimização de resíduos resultantes de processos químicos.

2 FUNDAMENTOS MATEMÁTICOS DA OTIMIZAÇÃO CONTÍNUA

Neste capítulo pretendemos expor os fundamentos matemáticos necessários à boa compreensão dos métodos de otimização contínua. Os aspectos matemáticos advêm principalmente das áreas do cálculo multivariável, álgebra linear e análise real. É fundamental que ideias como o de ***vetor gradiente, matriz hessiana***, ***espaços vetoriais***, **convergência de sequências e séries, função objetivo** entre outros, sejam claras ao leitor, pois são estes que estruturam matematicamente o funcionamento dos métodos utilizados neste trabalho.

2.1. Notações usadas no texto

Vamos adotar as seguintes notações para uma escrita concisa e precisa dos resultados e conceitos aqui presentes. Sempre que utilizar-se uma letra como x e esta estiver em *itálico* e for minúscula estará se referindo a um valor escalar. Quando utilizar-se uma letra como $\mathbf{x}$ minúscula e em **negrito** está será um vetor. Denotaremos uma n-upla de variáveis de controle como um vetor n-dimensional $\mathbf{x}$**,** ou seja, a ordem é importante. Assumimos também que esse vetor está posicionado na forma de um vetor-coluna, ou seja, na vertical. O vetor $\mathbf{x}^T$ é dito vetor transposto de $\mathbf{x}$, e está posicionado na forma de um vetor-linha, ou seja, na horizontal. Indicamos por $\mathbf{x}^*$o ponto ótimo procurado. Na Tabela 2.1 apresentamos um resumo das notações utilizadas neste trabalho.

Tabela 2.1- Lista de Símbolos sugestivos

Símbolo	Significado	Símbolo	Significado
$\mathbf{A}$	Uma matriz A qualquer.	$\mathbf{x}$ $\mathbf{x}^k$ $\mathbf{x}^*$	Vetor das variáveis de controle Vetor avaliado na iteração k Vetor no ponto ótimo
$\mathbf{a}$	Um vetor qualquer	$x_1, x_2, \dots, x_n$	Elementos do vetor $\mathbf{x}$
$\mathbf{a}_i, \quad \boldsymbol{i} = \mathbf{1}, \mathbf{2}, \dots$	Coluna i de uma matriz qualquer $\mathbf{A}$	$\mathbf{s}$ $\mathbf{s}^k$	Direção de busca, Direção de busca na iteração k
$\mathbf{I}$	A matriz identidade. Matriz cujos elementos da diagonal principal $a_{ij} = 1, com\ i = j$	$\alpha,$ α^k α^*	Tamanho do passo Tamanho do passo na iteração $k,$ Tamanho do passo no ponto de mínimo ao longo da direção de busca.

$\mathbf{A}^{\mathbf{T}}, \mathbf{a}^{\mathbf{T}}$	Matriz transposta e vetor transposto	$f(\mathbf{x})$ $f(\mathbf{x}^k)$, f^k	Função objetivo Função objetivo avaliada em $\mathbf{x}^k$
$\boldsymbol{\nabla} f(\mathbf{x})$ ou $\boldsymbol{\nabla} f(\mathbf{x}^k)$, $\boldsymbol{\nabla} f^k$	Vetor gradiente da função $f(x)$, Vetor gradiente calculado no ponto $\mathbf{x}^k$	$\nabla^2 f(\mathbf{x}^k), \nabla^2 f^k$ $\mathbf{H}(\mathbf{x}^k), \mathbf{H}^k$	Matriz hessiana avaliada em $\mathbf{x}^k$
$\|\|\mathbf{x}\|\|$	Norma do vetor $\mathbf{x}$	$\|x\|$	Valor absoluto do escalar x
$\Delta\mathbf{x}^k = \mathbf{x}^{k+1} - \mathbf{x}^k$	Acréscimo ou diferença entre os vetores.	$\mathbf{x} \in R^n$	Indica que o vetor $\mathbf{x}$ é um elemento do espaço euclidiano de dimensão n
$\gamma^k = \nabla f^{k+1} - \nabla f^k$	Diferença nos gradientes	$\mathbf{N}^k$	Matriz de direção avaliada em $\mathbf{x}^k$

A seguir discutiremos alguns dos principais termos contidos na tabela 1, assim como utilizaremos estes conceitos preliminares para introduzir novas ideias e termos.

2.2. Vetor Gradiente

O gradiente de uma função $f(x)$ é comumente denotado por $\nabla f(\mathbf{x})$. O gradiente é definido como um vetor coluna que contém as derivadas parciais de primeira ordem de $f(\mathbf{x})$:

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} \tag{2.1}$$

Objetivando entender claramente o cálculo de um vetor gradiente apresentaremos a seguir um simples exemplo.

Exemplo: Calcular o vetor gradiente da função $f(\mathbf{x}) = 8 - 5x_1 + x_2 + 2x_1^2 + 3x_1x_2 + x_2^2$ e avalia-lo no ponto $\mathbf{x}^0 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$.

$\nabla f = \begin{bmatrix} -5 + 4x_1 + 3x_2 \\ 1 + 3x_1 + 2x_2 \end{bmatrix}$ que quando calculado em $\mathbf{x}^0 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$ resulta em $\nabla f(\mathbf{x}^0) = \begin{bmatrix} -3 \\ 1 \end{bmatrix}$

2.3. Algumas propriedades do vetor gradiente

(P1) O vetor gradiente $\nabla f(\mathbf{x})$ é ortogonal ao contorno da função e aponta na direção de maior crescimento da mesma. O gradiente negativo $-\nabla f(\mathbf{x})$ aponta na direção em que a função apresenta maior decrescimento.

(P2) O produto escalar de um vetor ortogonal $\mathbf{v}$ com $\nabla f(\mathbf{x})$ é zero, ou seja, $\mathbf{v}^{\mathrm{T}} \nabla f(\mathbf{x}) = 0$.

2.4. Matriz hessiana

A matriz hessiana $\mathbf{H}(\mathbf{x}), ou\ \nabla^2 f$ é definida como a matriz quadrada das derivadas parciais de segunda ordem. A forma resumida dessa matriz é dada por:

$$\left(\mathbf{H}_f\right)_{i,j} = \frac{\partial^2 f}{\partial x_i \partial x_j} \tag{2.2}$$

Adequando a fórmula (3.2) a forma matricial temos o seguinte:

$$\mathbf{H}(\mathbf{x}) = f(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \qquad (2.3)$$

Vamos agora, para esclarecer o procedimento de cálculo da matriz hessiana, desenvolver um exemplo que nos permita sistematizar de forma clara o processo de obtenção da mesma.

Encontre a matriz hessiana relacionada a função $f(\mathbf{x}) = 12 - 7x_2^2 x_1 + 5x_1 + 6x_1^2 + 7x_2^2$

Neste caso, como tem-se apenas 2 variáveis, x_1 e x_2, a hessiana será uma matriz quadrada de ordem 2×2. Assim sendo, terá apenas quatro elementos. Utilizando a forma resumida (2.2) para determinar cada termo teremos:

$$\begin{matrix} \frac{\partial^2 f}{\partial x_1^2} = 12 & \frac{\partial^2 f}{\partial x_1 \partial x_2} = -14x_2 \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} = -14x_2 & \frac{\partial^2 f}{\partial x_2^2} = 14 - 14x_1 \end{matrix} \quad \rightarrow \quad \mathbf{H}(\mathbf{x}) = f(\mathbf{x}) = \begin{bmatrix} 12 & -14x_2 \\ 14x_2 & 14 - 14x_1 \end{bmatrix}$$

2.5. Matrizes simétricas definidas

Definiremos aqui o que vem a ser uma matriz simétrica definida, mais especificamente o que vem a ser uma matriz simétrica definida positiva e negativa. Primeiro devemos conceituar o que vem a ser uma matriz simétrica. Abaixo temos a definição

Matriz simétrica: Em álgebra linear, uma matriz $\mathbf{A}$ é dita simétrica se esta coincidir com a sua $transposta$, ou seja, $\mathbf{A} = \mathbf{A}^{\mathrm{T}}$. Para concluir a introdução da matriz simétrica tem-se abaixo algumas propriedades de uma matriz simétrica.

- Se $\mathbf{A}$ é uma matriz simétrica, então para qualquer escalar $k \in \mathbb{R}$, temos que $k\mathbf{A}$ também é uma matriz simétrica.
- A soma de $\mathbf{A}$ com sua $transposta$ $\mathbf{A}^{\mathrm{T}}$ também é transposta.
- Se $\mathbf{A}$ é uma matriz simétrica, então todos os seus *autovalores* são reais.

Matriz simétrica positiva definida(Numerical optimization, Nocedal and Wright, p.16): diz-se que uma matriz $\mathbf{A}, \mathrm{n}_{\mathrm{x}}\mathrm{n}$ é *simétrica positiva definida* se, para qualquer vetor não-nulo $\mathbf{x} \in \mathbb{R}^n$ temos que:

$$\mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x} > 0 \tag{2.4}$$

Matriz simétrica negativa definida: diz-se que uma matriz $\mathbf{A}, \mathrm{n}_{\mathrm{x}}\mathrm{n}$ é *simétrica negativa definida* se, para qualquer vetor não-nulo $\mathbf{x} \in \mathbb{R}^n$ temos que:

$$\mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x} < 0 \tag{2.5}$$

É interessante destacar que a matriz que mais nos interessa no contexto da otimização é a matriz hessiana $\mathbf{H}(\mathbf{x})$. Caso a hessiana seja positiva definida isso indica que a função objetivo $f(\mathbf{x})$ tem sempre curvatura positiva e se for negativa definida a curvatura será sempre negativa. Essa é uma importante condição para que uma função apresente mínimo ou máximo respectivamente.

Para verificar se uma dada matriz $\mathbf{A}$ é positiva definida ou negativa definida é necessário um método que não inclua a verificação de todos os vetores $\mathbf{x} \in \mathbb{R}^n$, pois isso será impossível, temos infinitos vetores no espaço $\mathbb{R}^n$. Existem algumas técnicas que se utilizam das teorias e resultados da álgebra linear sobre matrizes, tais como autovalores, determinantes, decomposição de Cholesky entre outros. Abaixo discutiremos algumas formas de verificar se a matriz é positiva definida ou negativa definida, fazendo uso destes recursos. Vejamos abaixo algumas formas:

- Uma matriz simétrica $\mathbf{A}$ é positiva definida se todos os seus autovalores são positivos
- Uma matriz simétrica é positiva definida se o determinante de cada uma de suas matrizes menores principais é positivo. Um menor é uma submatriz que é obtida da matriz $\mathbf{A}$ eliminando-se a $linha\ i\ e\ coluna\ j$.
- Uma matriz simétrica $\mathbf{A}$ é positiva definida se podemos decompô-la como $\mathbf{A} = \mathbf{L}\mathbf{L}^{\mathrm{T}}$ em que $\mathbf{L}$ é uma matriz triangular inferior com elementos positivos na diagonal. Esse procedimento de reescrita da matriz pode ser executado utilizando-se a decomposição de Cholesky.

Vamos desenvolver um exemplo utilizando o método de verificação por matrizes menores principais, pois é o modo mais simples. Nos restringiremos aqui a operar uma matriz $3\ x\ 3$, caso se tomasse ordens maiores de matrizes o processo de verificação seria dispendioso, e também existem diversos softwares de computação numérica que executam facilmente estes cálculos e verificações. Alguns exemplos de softwares são o MATLAB, Wolfram Alpha, bibliotecas do Python entre outros.

2.6. Checando se uma matriz é positiva definida usando matrizes menores principais.

Considere a matriz $\mathbf{B} = \begin{bmatrix} 15 & 7 & -2 \\ 1 & 2 & 3 \\ 4 & 5 & 1 \end{bmatrix}$. As matrizes menores principais dela são:

$\mathbf{B_1} = [15]$, tal que $\det \mathbf{B_1} = 15$

$\mathbf{B_2} = \begin{bmatrix} 15 & 7 \\ 1 & 2 \end{bmatrix}$, cujo $\det \mathbf{B_2} = 23$

$\mathbf{B_3} = \mathbf{B} = \begin{bmatrix} 15 & 7 & -2 \\ 1 & 2 & 3 \\ 4 & 5 & 1 \end{bmatrix}$, cujo $\det \mathbf{B_3} = \det \mathbf{B} = -\ 112$

Como um dos determinantes é negativo a matriz $\mathbf{B}$ não é **positiva definida.** Podemos também verificar os autovalores dessa matriz, pois sendo de ordem $\mathbf{3\ x\ 3}$ não é tão dispendioso e computacionalmente complexo o seu cálculo, vejamos então.

Recorrendo à definição de autovetor que diz: que $\mathbf{v}$ **é um autovetor** associado à um autovalor λ com relação a matriz $\mathbf{B}$ se

$$\mathbf{Bv} = \lambda\mathbf{v} \tag{2.6}$$

Pela relação (2.6) pode-se entender que $\mathbf{v}$ é um autovetor se, e somente se,

$$\mathbf{Bv} - \lambda\mathbf{v} = \mathbf{Bv} - (\lambda\mathbf{I})\mathbf{v} = (\mathbf{B} - \lambda\mathbf{I})\mathbf{v} = \mathbf{0} \tag{2.7}$$

O que nos leva a concluir que

$$det(\mathbf{B} - \lambda\mathbf{I}) = \mathbf{0} \tag{2.8}$$

é a condição necessária e suficiente.

Vejamos então o desenvolvimento de (2.8) aplicado ao problema de determinar se a matriz $\mathbf{B}$ é positiva definida.

$$\det\left(\begin{bmatrix} 15 & 7 & -2 \\ 1 & 2 & 3 \\ 4 & 5 & 1 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right) = 0$$

$$\det\left(\begin{bmatrix} 15-\lambda & 7 & -2 \\ 1 & 2-\lambda & 3 \\ 4 & 5 & 1-\lambda \end{bmatrix}\right) = -\lambda^3 + 18\lambda^2 - 33\lambda - 112 = 0$$

Basta agora determinar as raízes da equação $-\lambda^3 + 18\lambda^2 - 33\lambda - 112 = 0$. Que são as seguintes:

$\lambda_1 = -1{,}69$

$\lambda_2 = 4{,}31$

$\lambda_3 = 15{,}38$

Como pode-se observar o autovalor λ_1 é negativo, o que confirma que $\mathbf{B}$ não é positiva definida.

2.7. Série de Taylor

A série de Taylor é uma série de funções, utilizada para fazer a expansão, ou aproximação de uma função f em um ponto especifico $\mathbf{x}^k$.

No caso de se aproximar uma função de uma variável em torno de um dado ponto x_0 a série é definida como o polinômio de $ordem\ n$ a seguir:

$$p_n(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!}(x - x_0))^n \qquad (2.9)$$

Vamos como exemplo, aproximar a função $f(x) = x^2 sen(x)$ em torno do ponto $x_0 = 1$ determinando os 4 primeiros termos. De modo geral, teríamos o seguinte:

$$p_3(x) = \frac{f(x_0)}{0!}(x - x_0)^0 + \frac{f^{(1)}(x_0)}{1!}(x - x_0)^1 + \frac{f^{(2)}(x_0)}{2!}(x - x_0)^2 + \frac{f^{(3)}(x_0)}{3!}(x - x_0)^3$$

+...

$$p_3(x) = \frac{x_0^2 sen(a)}{0!}(x - x_0)^0 + \frac{2x_0\, sen(x_0) + x_0^2 \cos(x_0)}{1!}(x - x_0)^1 -$$

$$\frac{+x_0^2 sen(x_0) - 4x_0 \cos(x_0) - 2sen(x_0)}{2!}(x - x_0)^2 - \frac{x_0^2 \cos(x_0) + 6x_0\, sen(x_0) - 6\cos(x_0)}{3!}(x - x_0)^3 + \cdots$$

Aplicando $a = 1$ no exemplo acima temos o seguinte polinômio:

$$p_3(x) = 1 + (x - 1) - \frac{1}{2}(x - 1)^2 + 2(x - 1)^2 + \frac{5}{6}(x - 1)^3$$

$$f(x) = \frac{5x^3 - 6x^2 + 3x + 4}{6}$$

A seguir veremos a série de Taylor no caso multivariável. Vale ressaltar antes, que a maioria dos problemas de otimização contínua utiliza apenas a aproximação de Taylor de primeira e segunda ordem, pois além de serem efetivas para os casos que

se aplicam, fazer aproximações de ordem 3 ou maior implicam custos computacionais muito grandes que invalidariam o ganho que se tem com a maior precisão das aproximações (RIBEIRO; KARAS,2013). Então a série de Taylor para aproximar uma função multivariável $f(\mathbf{x})$ em torno de um ponto $\mathbf{x}^k$é dada pela seguinte expressão:

$$f(\mathbf{x}^{k+1}) = \boldsymbol{f}(\mathbf{x}^k) + \nabla f(\mathbf{x}^k)^T(\mathbf{x}^{k+1} - \mathbf{x}^k) + \frac{1}{2}(\mathbf{x}^{k+1} - \mathbf{x}^k)^T\nabla^2 f(\mathbf{x}^k)(\mathbf{x}^{k+1} - \mathbf{x}^k) + \cdots \qquad (2.10)$$

Fazendo a adaptação $\Delta\mathbf{x}^k = (\mathbf{x}^{k+1} - \mathbf{x}^k)$ e $f^k = f(\mathbf{x}^k)$ podemos reescrever a fórmula (2.10) como:

$$f^{k+1} = f^k + (\nabla f^k)^T\Delta\mathbf{x}^k + \frac{1}{2}(\Delta\mathbf{x}^k)\nabla^2 f^k\Delta\mathbf{x}^k \qquad (2.11)$$

2.8. Exemplo de uma aproximação utilizando o polinômio de Taylor de 2° ordem.

Dada a função $f(\mathbf{x}) = 2\sqrt{x_1} + 3\ln(x_2)$ considerado em um ponto $\mathbf{x}^k = [5 \quad 4]$.

O vetor gradiente é $\nabla f^T = \left[(x_1)^{-\frac{1}{2}} \quad \frac{3}{x_2}\right]$ que avaliado no ponto $\mathbf{x}^k$ fica

$$(\nabla f^k)^T = [0{,}447 \quad 0{,}750].$$

As derivadas parciais de 2° ordem são as seguintes:

$$\frac{\partial^2 f}{\partial x_1^2} = -\frac{1}{2}(x_1)^{-\frac{3}{2}} \qquad \frac{\partial^2 f}{\partial x_1 \partial x_2} = 0 \qquad \frac{\partial^2 f}{\partial x_2 \partial x_1} = 0 \qquad \frac{\partial^2 f}{\partial x_2^2} = -\frac{3}{x_2^2}$$

Que formam a matriz hessiana $\mathbf{H}(\mathbf{x}) = \begin{bmatrix} -\frac{1}{2}(x_1)^{-\frac{3}{2}} & 0 \\ 0 & -\frac{3}{x_2^2} \end{bmatrix}$ que avaliada no ponto $\mathbf{x}^k = [5 \quad 4]$ fica $\mathbf{H}(\mathbf{x}) = \begin{bmatrix} -0{,}045 & 0{,}0 \\ 0{,}0 & -0{,}188 \end{bmatrix}$.

Utilizando a relação (2.11) e avaliando cada um de seus termos em $\mathbf{x}^k = [5 \quad 4]$ nossa aproximação de Taylor de 2° ordem fica:

$$f(\mathbf{x}) \approx 8{,}631 + [0{,}477 \quad 0{,}750]\begin{bmatrix} x_1 - 5 \\ x_2 - 4 \end{bmatrix} + \frac{1}{2}[x_1 - 5 \quad x_2 - 4]\begin{bmatrix} -0{,}045 & 0{,}0 \\ 0{,}0 & -0{,}188 \end{bmatrix}\begin{bmatrix} x_1 - 5 \\ x_2 - 4 \end{bmatrix}$$

que agrupando e simplificando termos semelhantes fica:

$$f(\mathbf{x}) \approx 1{,}300 + 0672x_1 + 1{,}502x_2 - 0{,}023x_1^2 - 0{,}094x_2^2$$

Para entender a eficiência da aproximação observe a tabela 2.2 com um comparativo da função original e da aproximação pela série de Taylor de 2° ordem par alguns pontos, incluindo o ponto $\mathbf{x}^k = [5 \quad 4]$, que foi em torno do qual se construiu a aproximação.

Tabela 2.2 - Comparativo de função e sua aproximação de Taylor de 2° ordem – adaptado de (PARKINSON;BALLING; HEDENGREN, 2018)

$\mathbf{x}^T$	Aprox. de Taylor de 2° ordem	$f(\mathbf{x}) = 2\sqrt{x_1} + 3\ln(x_2)$	Erro absoluto
$[5 \quad 4]$	8,63	8,63	0,00
$[5 \quad 5]$	9,28	9,3	0,02
$[6 \quad 4]$	9,05	9,06	0,01
$[7 \quad 6]$	10,55	10,67	0,12
$[2 \quad 1]$	3,98	2,83	1,15
$[9 \quad 2]$	8,19	8,08	0,11

2.9. Funções quadráticas

Boa parte da teoria da otimização continua é baseada em funções quadráticas, pois estas aproximam com um erro aceitável qualquer função em uma vizinhança $V_\epsilon(\mathbf{x}^k)$ de um ponto $\mathbf{x^k}$. Essa aproximação por uma função quadrática é feita por meio da série de Taylor vista anteriormente, considerado seu desenvolvimento até o terceiro termo.

2.10. Aproximação de Taylor de 2° ordem para a função

$$f(\mathbf{x}) = 12 - 5x_1 + 3x_2 + 4x_1^2 + 6x_1x_2 + x_2^2$$

Primeiro consideramos a forma de representação de uma equação quadrática da seguinte forma:

$f(\mathbf{x}) = a + \mathbf{b}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{C}\mathbf{x}$, onde $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

Então nossa função pode ser representada como se segue:

$$f(\mathbf{x}) = 6 + [-5 \quad 3]\mathbf{x} + \frac{1}{2}\mathbf{x}^T \begin{bmatrix} 8 & 5 \\ 7 & 2 \end{bmatrix} \mathbf{x}$$

2.11. Propriedades da função quadrática

Algumas propriedades relacionadas a função quadrática segundo (PARKINSON;BALLING; HEDENGREN, 2018) que são uteis no entendimento dos algoritmos de otimização serão tratados adiante. Como foi dito na seção sobre o polinômio de Taylor, enfatizando que serão utilizados apenas aproximação de 1° e 2° ordens, fica evidente a necessidade de se entender melhor a função quadrática.

- As equações que compõe o vetor gradiente ∇f da função quadrática são equações lineares. Isso possibilita uma fácil resolução do quando se toma $\nabla f = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$.
- A matriz hessiana da função quadrática é uma matriz de constantes, isso significa que a curvatura da função é sempre a mesma. O fato de a hessiana ser sempre composta de constantes permite que se utilize sem ambiguidade $\mathbf{H}$ ou $\nabla^2 f$ no lugar de $\mathbf{H}(\mathbf{x})$ ou $\nabla^2 f(\mathbf{x})$.
- Funções quadráticas tem apenas um ponto estacionário, ou seja, apenas um ponto onde o gradiente $\nabla f(\mathbf{x})$ é zero. Exceto quando a matriz hessiana é semi-definida.
- Dados o vetor gradiente $\nabla f(\mathbf{x}^k)$ e a matriz hessiana $\mathbf{H}(\mathbf{x}^k)$ em um ponto $\mathbf{x}^k$, o gradiente em outro ponto é dado por: $\nabla f^{k+1} = \nabla^k + \mathbf{H}(\mathbf{x}^{k+1} - \mathbf{x}^k)$.
- Dado o gradiente $\nabla f(\mathbf{x}^k)$ em um ponto $\mathbf{x}^k$, a matriz hessiana $\mathbf{H}$, e uma direção de busca $\mathbf{s}$, o tamanho ótimo do passo α^*, ao longo da direção de busca $\mathbf{s}$ é dada pela expressão:

$$\alpha^* = \frac{(\nabla f^k)^T \mathbf{s}}{\mathbf{s}^T \mathbf{H} \mathbf{s}} \qquad (2.12)$$

2.12. Condições para otimização irrestrita

Nesta seção iremos discutir a teoria que torna possível a otimização irrestrita de uma função, tendo como condição de partida que esta seja derivável em torno de uma vizinhança de um ponto $\mathbf{x}^*$, ou seja, em $V_\epsilon(\mathbf{x}^*)$.

Condição necessária: As condições necessárias para $\mathbf{x}^*$seja ponto de ótimo irrestrito de uma função f são:

- $\nabla f(\mathbf{x}^*) = 0$ e ,
- $f(\mathbf{x})$ seja derivável em $\mathbf{x}^*$

Todavia essa condição é necessária, mas não suficiente, pois como se vê no estudo de diferenciabilidade de funções de várias variáveis, $\mathbf{x}^*$ pode ser tanto um ponto de máximo, de mínimo ou ponto de cela. É necessária outra condição que garanta que este ponto seja de máximo ou de mínimo inequivocamente. Como veremos a seguir, o instrumento que nos permite fazer essa garantia é a matriz hessiana $\mathbf{H}(\mathbf{x})$. Vale ressaltar que as condições suficientes logicamente incluem as condições necessárias.

Condições suficientes para que $\mathbf{x}^*$ seja um ponto mínimo: As condições suficientes para um mínimo em $\mathbf{x}^*$são:

i. $\nabla f(\mathbf{x}^*) = 0$ e ,
ii. $f(\mathbf{x})$ seja derivável em $\mathbf{x}^*$
iii. A matriz hessiana $\mathbf{H}(\mathbf{x}^*)$ avaliada no suposto ponto de mínimo tem de ser positiva definida.

Condições suficientes para que $\mathbf{x}^*$ seja ponto um máximo: As condições suficientes para um máximo em $\mathbf{x}^*$são:

i. $\nabla f(\mathbf{x}^*) = 0$ e ,
ii. $f(\mathbf{x})$ seja derivável em $\mathbf{x}^*$
iii. A matriz hessiana $\mathbf{H}(\mathbf{x}^*)$ avaliada no suposto ponto de máximo tem de ser negativa definida.

A seguir apresentamos um modelo geral de otimização continua, que explica alguns aspectos fundamentais. Este modelo foi adaptado de Ribeiro e Karas (2013, p.ix-x).

Trataremos aqui sob a perspectiva de minimização de uma função objetivo(F.O) f, sendo que o problema de maximização requer poucas adaptações na F.O. Relembremos brevemente o modelo geral de otimização irrestrita, que pode ser definido como se segue (RIBEIRO; KARAS, 2013):

Formalmente, pode-se dizer que otimização é o processo de procurar e encontrar pontos $\mathbf{x}$ de mínimo ou de máximo de uma função real f, de modo que estes pontos pertençam a um conjunto $\Omega \subset \mathbb{R}^n$. Pode-se formular esse problema como segue:

$$minimizar\ f(\mathbf{x})$$
$$sujeito\ a\ \mathbf{x} \subset \Omega$$

Para problemas de otimização restrita, o conjunto Ω é definido por restrições de igualdade e/ou desigualdade, sendo assim limitado por propriedades como segue,

$$\Omega = \{\mathbf{x} \in \mathbb{R}^n | c_{\mathcal{E}}(\mathbf{x}) = 0\ , c_{\mathcal{J}}(\mathbf{x}) \leq 0\ \}$$

onde $c_{\mathcal{E}}: \mathbb{R}^n \to \mathbb{R}^m$ e $c_{\mathcal{J}}: \mathbb{R}^n \to \mathbb{R}^p$ são funções gerais. Interpretando dessa forma o domínio de busca da variável de controle $\mathbf{x}$, pode-se reformular o problema geral de otimização como:

$$minimizar\ f(\mathbf{x})$$
$$sujeito\ a\ \ c_{\mathcal{E}}(\mathbf{x}) = 0$$
$$c_{\mathcal{J}}(\mathbf{x}) \leq 0$$

Quando restringimos o problema de otimização ao caso de otimização sem restrições podemos representa-lo como

$$minimizar\ f(\mathbf{x})$$
$$sujeito\ a\ \mathbf{x} \subset \mathbb{R}^{\mathrm{n}}$$

Colocado o modelo de otimização acima e suas diversas particularidades quanto ao domínio da variável de controle $\mathbf{x}$, podemos dizer resumidamente que os métodos de otimização propostos adiante, trabalham em cima dessa variável $\mathbf{x}$ com o objetivo de encontrar valores que consigam determinar o valor máximo ou mínimo da função f.

3 MÉTODOS DE ORDEM UM - POLAK-RIBIÈRE

3.1. Breve histórico dos métodos de direção conjugada

Os métodos de gradiente conjugado são largamente utilizados para resolução de sistemas de equações que possuem uma quantidade muito grande de variáveis, sendo muito úteis também em processos de otimização. Existem dois tipos desse algoritmo, o gradiente conjugado linear e o gradiente conjugado não-linear. O gradiente conjugado linear foi proposto por Hestenes e Stiefel nos anos 50, para resolver sistemas lineares cuja matriz de coeficientes é positiva definida(NOCEDAL; WRIGTH, 2006). O gradiente conjugado não-linear foi introduzido por Fletcher e Reeves nos anos 1960, sendo está uma das primeiras técnicas conhecidas para resolver problemas de otimização de larga escala. Algoritmos baseados no tipo linear são mais indicados para otimizar funções quadráticas fortemente convexas. No caso de algoritmos baseados no tipo gradiente conjugado não linear são muito eficientes na otimização de funções convexas gerais, não apenas quadráticas.

O algoritmo que discutiremos adiante se baseia no tipo gradiente conjugado não-linear, o método de Polak-Ribière, uma variação de Fletcher-Reeves.

3.2. Direções conjugadas

Considere $\mathbf{A} \in \mathbb{R}^{n \times n}$ sendo uma matriz positiva definida. Diz-se que os vetores $\mathbf{d}_0, \mathbf{d}_1, \dots, \mathbf{d}_k \in \mathbb{R}^n - \{0\}$ são **A**-conjugados, ou conjugados em relação a matriz **A** se$(\mathbf{d}_i)^T \mathbf{A}\, \mathbf{d}_j = \mathbf{0}, \forall i, j = 0,1, \dots, k, \text{com } i \neq j$

3.3. Método Polak-Ribière

O método de gradiente conjugado não linear Polak-Ribière(PR-CG) é uma variação do método Fletcher-Reeves(FR-CG) que é largamente utilizado, diferindo

deste apenas por uma alteração no parâmetro β_k, onde este agora é determinado como abaixo:

$$\beta_{k+1}^{PR} = \frac{\nabla f_{k+1}^T(\nabla f_{k+1}^T - \nabla f_k)}{\|\nabla f_k\|^2} \tag{3.1}$$

Como fim de comparação, vejamos também o parâmetro β_k definido para o algoritmo Fletcher-Reeves,

$$\beta_{k+1}^{FR} = \frac{\nabla f_{k+1}^T \nabla f_{k+1}}{\nabla f_k^T \nabla f_k} \tag{3.2}$$

Uma observação pertinente a se fazer é que quando a função objetivo f que se deseja minimizar é quadrática e fortemente convexa e o parâmetro α_k é exato (o que acarreta que os vetores gradientes são ortogonais um em relação ao outro) é que $\beta_{k+1}^{FR} = \beta_{k+1}^{PR}$. Porém, se f é uma função não linear geral, não podendo-se garantir que seja quadrática e fortemente convexa, com a busca de linha α_k inexata, os parâmetros β_k dos algoritmos FR-CG e PR-CG são notadamente diferentes.

Tanto o método FR-CG quanto PR-CG têm o problema de que quando a condição $\nabla f_k^T \mathbf{s}_k > 0$ se verifica, temos $\mathbf{s}_k$ como uma direção de acréscimo, ou direção de subida

Abaixo apresentamos uma versão do método Polak-Ribière adaptado de Nocedal(2006, p.121), onde partimos do algoritmo Fletcher-Reeves exposto no livro e incluímos algumas adaptações recomendadas pelo mesmo, de modo a obter o algoritmo abaixo:

Algoritmo 2.1 (PR-CG)
Calcule $f_0 = f(\mathbf{x}_0), \nabla f_0 = \nabla f(\mathbf{x}_0)$;
Inicie $\mathbf{s}_0 = -\nabla f_0, k \leftarrow 0$;
Enquanto $\nabla f_k \neq 0$
Defina $\nabla f_k^T \mathbf{s}_k = -\|\nabla f_k\|^2 + \beta_k^{\mathrm{PR}} \mathbf{s}_{k-1}$
Se $\nabla f_k^T > 0$, então imponha as condições fortes de Wolfe (3.3) e (3.4) sobre α_k:
Defina $0 < c_1 < c_2 < \frac{1}{2}$
$f(\mathbf{x}_k + \alpha_k \mathbf{s}_k) \leq f(\mathbf{x}_k) + c_1 \alpha_k \nabla f_k^T \mathbf{s}_k,$ (3.3)
$\lvert\nabla f(\mathbf{x}_k + \alpha_k \mathbf{s}_k)^T\rvert \leq c_2 \lvert\nabla f_k^T \mathbf{s}_k\rvert$ (3.4)

Determine α_k e defina $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{s}_k$

Encontre α_k que minimize $f(\mathbf{x}_{k+1})$ ao longo da direção $\mathbf{s}_k$

Calcule ∇f_{k+1};

$\beta_{k+1}^{\mathrm{PR}} = \frac{\nabla f_{k+1}^T(\nabla f_{k+1} - \nabla f_k)}{\|\nabla f_k\|^2}$

$\mathbf{s}_{k+1} = -\nabla \mathrm{f}_{k+1} + \beta_{k+1}^{\mathrm{PR}} \mathbf{s}_k$

$k \leftarrow k + 1$

Uma observação que tem de ser feita é que as condições fortes de Wolfe não garantem que a direção $\mathbf{s}_k$ no método Polak-Ribière sempre será uma direção de descida(NOCEDAL; WRIGHT, 2006, p.122). Isso implica que o algoritmo pode necessitar de mais passos do precisaria para atingir o valor ótimo de f, caso as direções fossem sempre de descida. Para que a direção $\mathbf{s}_k$ seja sempre de descida é preciso que se aplique a seguinte condição em cada repetição do método:

$$\beta_{k+1}^{PR} = \beta_{k+1}^{+} = \max\{\beta_{k+1}^{PR}, 0\} \tag{3.5}$$

que garantem que a propriedade de descida se mantenha.

Essa adaptação dá origem a uma nova versão do algoritmo PR-CG que pode ser referida como PR^+. Porém, ambas as versões são largamente utilizadas, pois mesmo que no caso do algoritmo PR-CG sejam necessários passos adicionais, sua performance continua robusta para problemas de alta dimensão.

Abaixo apresentamos as etapas de funcionamento do método PR-CG como um algoritmo descritivo.

Método Polak-Ribière – algoritmo descritivo

1. Inicie com um ponto arbitrário $\mathbf{x}_0$
2. Definia a primeira direção de busca $\mathbf{s}_0 = -\nabla f(\mathbf{x}_0) = -\nabla f_0$
3. Encontre o ponto $\mathbf{x}_1$a partir da relação $\mathbf{x}_1 = \mathbf{x}_0 + \alpha_0 \mathbf{s}_0$, onde α_0^* é o tamanho ótimo de passo na 1º iteração ao longo da direção $\mathbf{s}_0$.
4. Atualize $i = 1$ e vá para o próximo passo.

5. Encontre $\nabla f_i = \nabla f(\mathbf{x}_i)$, e execute a seguinte atribuição:

$$\mathbf{s}_i = -\nabla f_i + \frac{\nabla f_{k+1}^T(\nabla f_{k+1} - \nabla f_k)}{\|\nabla f_{i-1}\|^2}\mathbf{s}_{i-1}$$

6. Determine o tamanho ótimo de passo α_i^*ao longo da direção $\mathbf{s}_i$, e encontre o novo ponto de acordo com a relação:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i^*\mathbf{s}_i$$

7. Verifique se o ponto $\mathbf{x}_{i+1}$ é ponto de ótimo da função f, ou seja, verifique se $\nabla f(\mathbf{x}_{i+1}) = 0$. Caso contrário, vá para o passo 4.

Vale ainda comentar alguns resultados e observações sobre convergência e funcionamento dos métodos de gradiente conjugado não lineares.

Problemas de otimização de grandes dimensões, requerem incontáveis avaliações e cálculos do parâmetro β_k, o que leva este parâmetro muitas vezes a acumular erros de precisão, devido aos limites de ponto flutuante dos computadores. Ao se aferir um parâmetro por meio dos cálculos, componentes do processador responsáveis pelas operações aritméticas básicas como a unidade lógica aritmética(ALU, do inglês *Arithmetic Logic Unit*), entre outras, são limitadas em sua capacidade de cálculo e de precisão (STALLINGS; 2010). Essa imprecisão inerente é controlada por parâmetros que limitam a precisão dos cálculos e, ao longo das diversas repetições que um método de otimização realiza, é certo que diversas imprecisões são acumuladas. Reiniciar periodicamente o parâmetro β_k é muito benéfico para a precisão do algoritmo(NOCEDAL; WRIGHT, 2006).

Na tabela 3.1, adaptada de Nocedal e Wright (2006, p.124), apresenta-se uma comparação entre o algoritmo Polak-Ribière denotado por PR-CG e a versão melhorada PR^+, onde se incluiu a condição da fórmula (3.5) que garante que a direção $\mathbf{s}_k$ sempre seja uma direção de descida.

Tabela 3.1 - Comparação dos algoritmos PR-CG e PR^+ - Adaptada de Nocedal e Wright (2006, p.124)

Problema	n – dimensão do problema	Algoritmo PR-CG lt/f-g	Algoritmo PR^+ lt/f-g	mod
CALCVAR3	200	2631/5263	2631/5263	0
GENROS	500	1068/2151	1067/2149	1
XPOWSING	1000	212/473	97/229	3
TRIDIA1	1000	262/527	262/527	0
MSQRT1	1000	113/231	113/231	0
XPOWELL	1000	212/473	97/229	3
TRIGON	1000	40/92	40/92	0

A última coluna na tabela 3.1, denotada por “mod” indica quantas iterações das totais indicados na 4° coluna foram necessárias para se garantir a condição dada pela fórmula (3.1) para o algoritmo PR^+. Nota-se que nos problemas CALCVAR3, GENROS, TRIDIA1, MSQRT1 e TRIGON as diferenças de iterações e avaliações da função gradiente entre os algoritmos são estatisticamente desprezíveis. No entanto, os problemas XPOWSING e XPOWELL tem diferenças significativas, como podemos ver na tabela 3.2.

Tabela 3.2 - Comparativo entre os métodos PR-CG e PR^+.

Problema	n – dimensão do problema	
CALCVAR3	200	0% / 0%
GENROS	500	0,09% / 0,09%
XPOWSING	1000	54,24% / 51,58%
TRIDIA1	1000	0% / 0%
MSQRT1	1000	0% / 0%
XPOWELL	1000	54,24% / 51,58%

TRIGON	1000	0% / 0%

Considerando o algoritmo para o método PR-CG exposto anteriormente, desenvolveremos um exemplo utilizando esse algoritmo. Aqui se considerará uma função quadrática fortemente convexa, com o intuito de tornar o exemplo mais didático. Essa escolha nos permite simplificar o algoritmo, fazendo com que a definição do parâmetro β_k seja igual a do algoritmo Fletcher-Reeves, ou seja, $\beta_{k+1}^{FR} = \beta_{k+1}^{PR}$. Desse modo pode-se reescrever o algoritmo PR-CG como:

Algoritmo 3.2 (PR-CG)

Calcule $f_0 = f(\mathbf{x}_0), \nabla f_0 = \nabla f(\mathbf{x}_0)$;

Inicie $\mathbf{s}_0 = -\nabla f_0, k \leftarrow 0$;

Enquanto $\nabla f_k \neq 0$

Determine α_k e defina $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{s}_k$

Encontre α_k que minimize $f(\mathbf{x}_{k+1})$ ao longo da direção $\mathbf{s}_k$

Calcule ∇f_{k+1};

$\beta_{k+1}^{\mathrm{PR}} = \frac{\nabla f_{k+1}^T(\nabla f_{k+1} - \nabla f_k)}{\|\nabla f_k\|^2}$

$\mathbf{s}_{k+1} = -\nabla \mathrm{f}_{k+1} + \beta_{k+1}^{\mathrm{PR}} \mathbf{s}_k$

$k \leftarrow k + 1$

3.4. Exemplo

Minimizar $f(x_1, x_2) = x_1 - x_2 + 2x_1^2 + 2x_1x_2 + x_2^2$, começando pelo ponto $\mathbf{x}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$.

Inicialmente, determinemos o vetor gradiente $\nabla f(\mathbf{x})$ e a matriz hessiana $\mathbf{H}$:

$$\nabla f(\mathbf{x}) = \begin{bmatrix} 1 + 4x_1 + 2x_2 \\ -1 + 2x_1 + 2x_2 \end{bmatrix}$$

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{bmatrix} = \begin{bmatrix} 4 & 2 \\ 2 & 2 \end{bmatrix}$$

- **Iteração 1**:

<u>Passo 1</u>:

avaliar ∇f em $\mathbf{x}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$. Então temos $\nabla f_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$

determinar direção de busca $\mathbf{s}_1 = -\nabla f_1$. Então temos $\mathbf{s}_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$.

<u>Passo 2</u>:

Nesse passo, deveríamos primeiramente determinar a expressão para o ponto $\mathbf{x}_2$, escrevendo-a em função do parâmetro de busca linear α_1, donde teríamos a $x_2 = x_1 + \alpha_1 \mathbf{s}_1$. Com essa expressão, faríamos $f(x_1 + \alpha_1 \mathbf{s}_1)$, o que tornaria a função f de duas variáveis (x_1 e x_2) em uma função de apenas uma variável, agora escrita em termo de α_1. Então, nesse ponto, poder-se-ia utilizar algum método de otimização unidimensional, como método da *secção áurea*, método de *Fibonacci.* Com isso, obteríamos o comprimento ótimo de passo α_k que minimiza a função f em $\mathbf{x}_1$. No entanto, é possível determinar o comprimento ótimo para α_1 utilizando a seguinte expressão:

$$\alpha_k = \frac{\nabla f_k^T \nabla f_k}{\mathbf{s}_k^T \mathbf{H} \mathbf{s}_k}$$

Como o exemplo que estamos desenvolvendo tem apenas duas variáveis, a utilização dessa relação tem caráter didático, pois quando o número de variáveis se torna muito grande não é vantajoso seu uso. O custo computacional do cálculo da matriz hessiana $\mathbf{H}$ é muito mais elevado que o processo de otimização unidimensional utilizado para calcular o valor ótimo de α_k.

Todavia, temos o seguinte:

$$\alpha_1 = \frac{\nabla f_1^T \nabla f_1}{\mathbf{s}_1^T \mathbf{H} \mathbf{s}_1} = \frac{\begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix}}{\begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} 4 & 2 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix}} = \frac{2}{2} \rightarrow \alpha_1 = 1$$

Portanto o novo ponto é:

$$\mathbf{x}_2 = \mathbf{x}_1 + \alpha_1 \mathbf{s}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + 1 \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

$$\mathbf{x}_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

<u>Passo 3</u>:

Aqui checamos se atingimos o mínimo da função f no ponto $\mathbf{x}_2$. Para isso avaliamos o gradiente da função, $\nabla f(\mathbf{x})$, no ponto $\mathbf{x}_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$, e verificamos se a seguinte relação é verdadeira:

$$\nabla f(\mathbf{x}_2) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\nabla f(\mathbf{x}_2) = \ \nabla f_2 = \begin{bmatrix} 1 + 4(-1) + 2(1) \\ -1 + 2(-1) + 2(1) \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

Vemos então que $\nabla f_2 = \begin{bmatrix} -1 \\ -1 \end{bmatrix} \neq \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, o que indica que $\mathbf{x}_2$ não é ponto de mínimo. Assim, partimos para próxima iteração.

- **Iteração 2**:

<u>Passo 1</u>:

avaliar ∇f em $\mathbf{x}_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$. Então temos $\nabla f_2 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$

determinar direção de busca $\mathbf{s}_2 = -\nabla f_1 + \beta_2^{\mathrm{PR}} \mathbf{s}_1$. Percebemos que a expressão aqui utilizada para determinar $\mathbf{s}_2$ é diferente da utilizada na iteração 1. Isso se deve ao fato de que na primeira iteração a direção de busca $\mathbf{s}_k$ é definida como o vetor gradiente negativado, ou seja, $\mathbf{s}_1 = \ -\nabla f_1$.

Temos a seguinte relação para determinar o parâmetro β_k^{PR}:

$$\beta_{k+1}^{\mathrm{PR}} = \frac{\nabla f_{k+1}^T (\nabla f_{k+1} - \nabla f_k)}{\|\nabla f_k\|^2}$$

$$\beta_2^{\mathrm{PR}} = \frac{\nabla f_2^T(\nabla f_2 - \nabla f_1)}{\|\nabla f_1\|^2} = \frac{[-1 \quad -1]\left(\begin{bmatrix}-1\\-1\end{bmatrix} - \begin{bmatrix}-1\\1\end{bmatrix}\right)}{\left\| \sqrt{(-1)^2+(1)^2} \right\|^2} = \frac{[-1 \quad -1]\left(\begin{bmatrix}0\\-2\end{bmatrix}\right)}{2} = \frac{2}{2} \;\rightarrow\; \beta_2^{\mathrm{PR}} = 1$$

Então, temos a seguinte direção de busca:

$\mathbf{s}_2 = -\nabla f_2 + \beta_2^{\mathrm{PR}}\mathbf{s}_1$

$\mathbf{s}_2 = \begin{bmatrix}1\\1\end{bmatrix} + 1\begin{bmatrix}-1\\1\end{bmatrix} = \begin{bmatrix}0\\2\end{bmatrix} \;\rightarrow\; \mathbf{s}_2 = \begin{bmatrix}0\\2\end{bmatrix}$

<u>Passo 2</u>:

Temos o seguinte:

$$\alpha_2 = \frac{\nabla f_2^T \nabla f_2}{\mathbf{s}_2^T \mathbf{H} \mathbf{s}_2} = \frac{[-1 \quad -1]\begin{bmatrix}-1\\-1\end{bmatrix}}{[0 \quad 2]\begin{bmatrix}4 & 2\\2 & 2\end{bmatrix}\begin{bmatrix}0\\2\end{bmatrix}} = \frac{2}{8} \;\rightarrow \alpha_1 = \frac{1}{4}$$

Portanto o novo ponto é:

$\mathbf{x}_3 = \mathbf{x}_2 + \alpha_2\mathbf{s}_2 = \begin{bmatrix}-1\\1\end{bmatrix} + \frac{1}{4}\begin{bmatrix}0\\2\end{bmatrix}$

$\mathbf{x}_2 = \begin{bmatrix}-1\\1{,}5\end{bmatrix}$

<u>Passo 3</u>:

Aqui checamos se atingimos o mínimo da função f no ponto $\mathbf{x}_3$. Para isso avaliamos o gradiente da função, $\nabla f(\mathbf{x})$, no ponto $\mathbf{x}_3 = \begin{bmatrix}-1\\1{,}5\end{bmatrix}$, e verificamos se a seguinte relação é verdadeira:

$$\nabla f(\mathbf{x}_3) = \begin{bmatrix}0\\0\end{bmatrix}$$

$$\nabla f(\mathbf{x}_3) = \nabla f_3 = \begin{bmatrix}1 + 4(-1) + 2(1{,}5)\\-1 + 2(-1) + 2(1{,}5)\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix}$$

Vemos então que $\nabla f_3 = \begin{bmatrix}0\\0\end{bmatrix}$, o que indica que $\mathbf{x}_3$ é ponto de mínimo. Assim, encerramos o algoritmo.

4 MÉTODO DE ORDEM DOIS: NEWTON TRUNCADO

Neste capítulo trataremos de métodos, ditos, de ordem dois, pois utilizam informações de derivadas parciais de segunda ordem, ou mais especificamente, da matriz hessiana da função objetivo a ser otimizada. Como visto no capítulo 2, a hessiana nos permite fazer algumas considerações sobre a curvatura da função objetivo. Por sua vez, são as informações de curvatura da função que permitem aos métodos de otimização tomar decisões de como iterar até atingir o ponto ótimo ou um ponto tão próximo deste ótimo, quanto se queira. Aqui voltaremos nossa atenção para o método de Newton-CG, que é uma variação do método de newton puro, e é também conhecido como método de Newton truncado. Todavia, faz-se necessário introduzir o método de Newton puro antes de abordar Newton-CG, pois as modificações são apenas pontuais, mais especificamente, alterações na direção de pesquisa $\mathbf{p}_k^N$

O método de Newton é muito prático para se otimizar problemas de pequenas ou grandes dimensões. Em cada iteração do algoritmo busca-se um ponto $\mathbf{x}^*$que minimize a função f. A iteração é do tipo:

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \mathbf{p}_k \tag{4.1}$$

onde :

α_k → tamanho do passo, em inglês *step lenght*

$\mathbf{p}_k$ → direção de Newton-pura $\mathbf{p}_k^N$ou uma aproximação

Existem algumas formas de se obter α_k, fazendo com que este satisfaça as condições de Wolfe ou as condições de Goldstein. Vejamos abaixo estas três condições.

4.1. Condição de Wolfe

Uma condição de busca inexata linear muito comum impõe que α_k deve, primeiramente, proporcionar decrescimento suficiente na função objetivo f, de modo que se verifique a desigualdade abaixo, conhecida como desigualdade ou condição de Wolfe:

$$f(\mathbf{x}^k + \alpha \mathbf{p}_k) \leq f(\mathbf{x}^k) + c_1 \alpha \nabla f_k^T \mathbf{p}_k \tag{4.2}$$

Onde c_1 é uma constante escolhida, tal que $c_1 \in (0,\ 1)$. Colocando de uma forma mais verbal que matemática, é necessário que a redução proporcionada na função objetivo f seja proporcional ao comprimento do passo α_k e a derivada direcional $\nabla f_k^T \mathbf{p}_k$.

4.2. Condição de Goldstein

Semelhantemente a condição de Wolfe, as condições de Goldstein também garantem que o tamanho do passo α proporcione um decrescimento suficiente para a função objetivo f, ao mesmo tempo que evita que α seja demasiadamente pequeno de modo que esta não proporcione efeito sobre f (NOCEDAL; WRIGHT, 2006). As condições de Goldstein podem ser declaradas como um par de desigualdades, como segue:

$$f(\mathbf{x}^k) + (1-c)\alpha_k \nabla f_k^T \mathbf{p}_k \leq f(\mathbf{x}^k + \alpha_k \mathbf{p}_k) \leq f(\mathbf{x}^k) + c\alpha_k \nabla f_k^T \mathbf{p}_k$$
(4.3)

onde $0 < c < \frac{1}{2}$.

Recordando as noções de modelagem e otimização introduzidas no capítulo 2, o nosso objetivo aqui é encontrar um ponto $\mathbf{x}^* \in \boldsymbol{\Omega}$ de modo que, para qualquer outro $\bar{\mathbf{x}} \in \boldsymbol{\Omega}$ tenha-se $f(\mathbf{x}^*) \leq f(\bar{\mathbf{x}})$. Abaixo apresentamos um algoritmo para o método de Newton puro e a partir dele, faremos considerações sobre seus aspectos práticos e de convergência, que na maior parte, se estenderão ao método de Newton-CG. Vejamos então:

Algoritmo de Newton-puro adaptado de (RIBEIRO; KARAS, 2013, P.98)
Dado $\mathbf{x^1} \in \mathbf{R}^{\mathrm{n}}$, um ponto inicial $k = 0$

Repita enquanto $\nabla f(\mathbf{x}^k) \neq 0$

Defina $\mathbf{p}_k^N = -\left(\nabla^2 f(\mathbf{x}^k)\right)^{-1} \nabla f(\mathbf{x}^k)$

(4.4)

Determine o tamanho do passo α_k

Faça $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \mathbf{p}_k^N$

$k = k + 1$

Um aspecto que é de importância prática destacar, pois facilita o processo de implementação do método em qualquer linguagem de programação, é que a direção d_k é determinada resolvendo-se o sistema abaixo:

$$\nabla^2 f(\mathbf{x}^k)\mathbf{p}_k^N = -\nabla f(\mathbf{x}^k) \tag{4.5}$$

A abordagem de resolução do sistema em (4.5) é encontrar uma solução exata. Todavia, essa abordagem, sob uma perspectiva de custo computacional, é menor do que a inversão da matriz hessiana em (4.4). Outro aspecto a respeito da direção $\mathbf{p}_k^N$ que vale ressaltar, é que, caso a matriz hessiana $\mathbf{H}(\mathbf{x}^k) = \nabla^2 f(\mathbf{x}^k)$ não seja positiva definida, a direção de busca pode não estar bem definida, ou seja, não é possível garantir que esta seja sempre uma direção de descida (RIBEIRO; KARAS, 2013). Nesse sentido o que se pode fazer é, testar uma abordagem diferente, utilizando o polinômio de Newton de segunda ordem para aproximar a função objetivo f. Essa abordagem tem a vantagem de lidar com uma função quadrática, para a qual a matriz hessiana é sempre, no caso da minimização, positiva definida. Desse modo, ter-se-ia a seguinte aproximação:

$$p_2(\mathbf{x}) = f(\mathbf{x}^k) + \nabla f(\mathbf{x}^k)^T(\mathbf{x} - \mathbf{x}^k) + \frac{1}{2}(\mathbf{x} - \mathbf{x}^k)^T \nabla^2 f(\mathbf{x}^k)(\mathbf{x} - \mathbf{x}^k) \tag{4.6}$$

Partindo do polinômio de aproximação obtido, faríamos

$$\nabla f(\mathbf{x}^k) + \nabla^2 f(\mathbf{x}^k)(\mathbf{x} - \mathbf{x}^k) = \nabla p(\mathbf{x}) = 0$$

e lembre-se que a condição $\nabla p(\mathbf{x}) = 0$ indica que temos um ponto de mínimo, ou seja, quando o vetor gradiente em um ponto $\mathbf{x}$ é igual a zero, isso indica que atingimos o menor valor possível para a função f. Fazendo-se algumas operações chega-se na seguinte relação:

$$\nabla^2 f(\mathbf{x}^k)(\mathbf{x} - \mathbf{x}^k) = -\nabla f(\mathbf{x}^k) \tag{4.7}$$

$$(\mathbf{x} - \mathbf{x}^k) = -\left(\nabla^2 f(\mathbf{x}^k)\right)^{-1} \nabla f(\mathbf{x}^k) \tag{4.8}$$

e como $\mathbf{p}_k^N = (\mathbf{x} - \mathbf{x}^k)$, tem-se $d_k = -\left(\nabla^2 f(\mathbf{x}^k)\right)^{-1} \nabla f(\mathbf{x}^k)$ que é a mesma expressão que a dada por (4)(cinco). Desse modo, garantindo que a matriz $\nabla^2 f(\mathbf{x}^k)$, seja sempre positiva definida.

A seguir enunciaremos 3 teoremas, para os quais não forneceremos provas, pois estas são de complexidade elevada que fogem ao objetivo deste texto, que é de caráter introdutório. No entanto, indicaremos a referência onde pode-se obter as demonstrações completas destes teoremas. Porém, antes é necessário introduzir a seguinte definição:

Definição (RIBEIRO; KARAS, 2013, p.82, está como Definição 4.11): Um algoritmo é dito globalmente convergente quando para qualquer que seja a sequência de pontos $(\mathbf{x}^k)$ gerada pelo algoritmo e qualquer ponto de acumulação $\bar{\mathbf{x}}$ da sequência $(\mathbf{x}^k)$, tem-se que $\mathbf{x}$ é estacionário.

Teorema 1 (RIBEIRO; KARAS, 2013, p.82, está como Teorema 4.12): O algoritmo de descida, também conhecido como algoritmo de descida mais íngreme ou steepest descent, cujo tamanho de passo é calculado pela busca exata, é globalmente convergente.

Teorema 2 (RIBEIRO; KARAS, 2013, p.83, está como Teorema 4.13): O algoritmo de descida, também conhecido como algoritmo de descida mais íngreme ou steepest descent, cujo tamanho de passo é calculado pela condição de Armijo, é globalmente convergente.

Teorema 3 (RIBEIRO; KARAS, 2013, p.100 - 101, está como Teorema 5.6): Suponha que a matriz hessiana $H(\mathbf{x}) = \nabla^2 f(\mathbf{x})$ é positiva definida, para todo $\mathbf{x} \in \mathbb{R}^n$. Então o algoritmo de Newton-puro, com tamanho de passo α_k calculado pela busca exata, é globalmente convergente, segundo a Definição 1. O mesmo resultado é válido caso se utilize a busca de Armijo para calcular α_k.

Demonstração: Note que o método de Newton-puro pode ser considerado situação particular do algoritmo de descida mais íngreme, sendo a matriz $\mathbf{H}(\mathbf{x}^k) = \left(\nabla^2 f(\mathbf{x}^k)\right)^{-1}$, para todo $k \in \mathbb{N}$. Desse modo, as afirmações do teorema 3 seguem diretamente dos Teoremas 1 e 2.

4.3. Relação entre método de Newton-puro e Newton-CG (ou Método de Newton Truncado)

A direção de pesquisa de Newton-Puro $\mathbf{p}_k^N$ é obtido resolvendo-se o sistema linear simétrico $n\ x\ n$:

$$\nabla^2 f(\mathbf{x}^k)\mathbf{p}_k^N = -\nabla f(\mathbf{x}^k) \tag{4.9}$$

Como descrito no início deste capítulo, para se obter convergência global é preciso que a direção de pesquisa $\mathbf{p}_k^N$ seja uma direção de descida e, para que esta direção seja de descida é preciso que a matriz hessiana $\nabla^2 f(\mathbf{x}^k)$ seja positiva definida. Tendo isso em vista, a ideia do método de Newton Truncado é utilizar uma técnica que garanta que a direção $\mathbf{p}_k^N$ seja sempre de descida, mas que não tome o caminho de computar a solução exata, pois isso recairia no método de Newton-Puro, com um custo computacional que pretendemos evitar. Além de garantir essa direção de descida para o parâmetro $\mathbf{p}_k^N$ é preciso ter em mente uma preocupação corrente na literatura dos métodos de Newton, que é, garantir que o custo computacional seja tão pequeno quanto possível. No caso do método de Newton-CG, o qual trabalharemos adiante, uma forma de garantir que o custo seja reduzido é, na etapa de cálculo da direção de Newton-puro, $\mathbf{p}_k^N$, se contentar com uma boa aproximação.

Uma das principais barreiras para se utilizar o método de Newton-puro em problemas de larga escala é o custo computacional elevado em se resolver, de forma exata o sistema (4.9). As técnicas utilizadas para computar a matriz hessiana $\nabla^2 f(\mathbf{x}^k)$ do sistema (4.9), que são baseadas em eliminação gaussiana ou algum tipo de fatoração da matriz de coeficientes, geralmente tem custo de computação elevado quando as dimensões(número de variáveis da função objetivo) do problema são

elevadas. Para contornar esse alto custo, uma opção é se utilizar de métodos iterativos de resolução de sistemas de equações de modo aproximado. E, como a solução para o sistema (4.5) é de caráter aproximado, é necessário estabelecer critérios de parada, regras matemáticas que nos digam quando a solução inexata é suficientemente boa para o problema em questão. Muitas regras de terminação deste método de resolução de sistemas se baseiam, segundo Nocedal e Wright (2006, p.136), na seguinte expressão residual:

$$r_k = \nabla^2 f(\mathbf{x}^k)\mathbf{p}_k + \nabla f(\mathbf{x}^k) \tag{4.10}$$

onde $\mathbf{p}_k$ é a direção de Newton aproximada. Como o tamanho de r_k varia conforme a função objetivo f é multiplicada por uma constante, então para se determinar o tamanho adequado do resíduo r_k o comparamos com o vetor gradiente $\nabla f(\mathbf{x}^k)$ no sistema (4.5). Dito isso, o processo iterativo de aproximação da solução do sistema (4.5) é terminado quando se verifica a seguinte condição:

$$\|r_k\| \leq \eta_k \|\nabla f(x^k)\| \tag{4.11}$$

onde (η_k), com $(0 < \eta_k < 1, \forall k)$ é uma sequência numérica chamada sequência de *forcing*, ou sequência de forçamento. Um ponto interessante a se destacar é, como a taxa de convergência de métodos de Newton inexatos baseados nas condições impostas por (4.9) -- (4.11) é afetada pela escolha da sequência *forcing* (η_k).

Um resultado interessante dado pelo Teorema 4, abaixo, diz que a convergência local é atingida garantindo simplesmente que a sequência (η_k) seja limitada superiormente por 1. Vejamos então o Teorema.

Teorema 4 (NOCEDAL;WRIGHT, 2006, pag.137, está como Teorema 6.1): Suponha que $\nabla f(\mathbf{x})$ seja contínua e derivável na vizinhança $V_\delta(\mathbf{x}^*)$ do minimizador $\mathbf{x}^*$ e, assuma também que a matriz hessiana $\mathbf{H}(\mathbf{x}^*) = \nabla^2 f(\mathbf{x}^*)$ é positiva definida. Considere também a fórmula de iteração $\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{p}_k$, onde $\mathbf{p}_k$ satisfaz a condição (11) e, assuma que $\eta_k \leq \eta$ para alguma constante $\eta \in [0,\ 1)$. Então, se o ponto inicial $\mathbf{x}^1$ está suficientemente próximo de $\mathbf{x}^*$, a sequencia $(\mathbf{x}^k)$ converge linearmente para o ponto minimizador $\mathbf{x}^k$. Portanto, $\forall k$ suficientemente grande, temos que:

$$\|\mathbf{x}^k - \mathbf{x}^*\| \leq c\|\mathbf{x}^k - \mathbf{x}^*\| \tag{4.12}$$

para alguma constante c, tal que $0 < c < 1$.

Não daremos a demonstração do Teorema 4 pelo mesmo motivo colocado para as Teoremas 1 e 2, mas consultando Nocedal e Wright (2006, p.137) é possível encontrar uma demonstração informal.

Teorema 5 (NOCEDAL;WRIGHT, 2006, pag.137, está como Teorema 6.2): Suponha que as condições do Teorema 4 sejam válidas e assuma que a sequência de pontos iterados $(\mathbf{x}^k)$ gerado pelo método de Newton inexato convirja para o ponto ótimo $\mathbf{x}^*$. Então a taxa de convergência é superlinear se $\eta_k \to 0$ e quadrática se $\eta_k = O(\|\nabla f(\mathbf{x}^k)\|)$.

As provas detalhadas dos teoremas 4 e 5, assim como resultados mais aprofundados sobre a convergência dos métodos de Newton inexatos, lembrando que o termo inexato se refere ao processo de determinação de uma solução aproximada para a direção de busca de Newton $\mathbf{p}_k$, podem ser encontrados em DEMBO et. Al (1982).

A seguir discutiremos efetivamente o método de Newton-CG ou Newton Truncado, que é classificado como um método de busca linear por utilizar na busca da direção $\mathbf{p}_k$ um método de gradiente conjugado linear(CG-linear). Nesta modificação do método de Newton-puro, a determinação da direção de pesquisa $\mathbf{p}_k$ é feita utilizando-se o método do gradiente conjugado linear para aproximação da solução de (4.9), respeitando-se a condição de parada estabelecida por (4.11). É preciso notar que no âmbito de resolução de sistemas de equações, o método CG-linear é adequado para resolve-los quando a matriz de coeficientes do sistema é positiva definida, conceito este discutido no capitulo 2.

Apesar da solução do sistema requerer que a matriz de coeficientes seja positiva definida em uma vizinhança do ponto de solução, pode-se ter sem prejuízo fora dessa vizinhança uma matriz hessiana com autovalores negativos, configurando assim a mesma como matriz negativa definida. Sobre essa ideia, deve-se destacar também que o gráfico da função objetivo tem superfície com curvatura negativa quando a matriz é negativa definida, que nesse caso implica que os autovalores são negativos. O método CG-linear utilizará dessas informações sobre a função objetivo e de sua matriz hessiana para encerar sua iteração em busca de uma aproximação da solução do sistema (4.9). Quando o método CG-linear gera uma direção de

pesquisa que indica que a superfície tem curvatura negativa, ele encerra a iteração, pois isso indica que estamos nos afastando do ponto de mínimo.

O método CG-linear garante que a direção de busca $\mathbf{p}_k$, gerada em cada iteração, é uma direção de descida e, que a taxa de convergência do método de Newton que é alta em comparação a métodos de ordem 1 seja preservada.

Parece ambíguo estarmos nos detendo tanto em discutir o método CG-linear quando o objetivo do capitulo é o método de Newton truncado, todavia essa digressão é necessária, pois o CG-linear está no cerne da superioridade do método de Newton-CG em relação ao método de Newton-puro.

A seguir discutiremos algumas condições que o método do gradiente conjugado linear segue na etapa de solucionar o sistema em (4.9), segundo proposto em Nocedal e Wright(2006). Antes, entretanto estabeleçamos as seguintes notações auxiliares:

$i)\ A = \nabla^2 f_k$

$ii)\ \mathbf{p}^i = \mathbf{p}_k$

As condições colocadas para que o método de CG-linear resolva o sistema (4.9) são:

1ª) O ponto inicial para iteração do método CG é $\mathbf{x}^0 = 0$

2 ª) Teste de curvatura negativa: Se a direção de pesquisa $\mathbf{p}^i$ gerada pela iteração do método CG satisfaz a condição

$$(\mathbf{p}^i)A\mathbf{p}^i \leq 0 \tag{4.13}$$

Então checamos se esta é a primeira iteração do método CG sobre (4.9). Se for a 1° iteração, completa-se ela e então computa-se o novo ponto $\mathbf{x}^1$ e termina-se a iteração atual. Se a condição em (4.10) é válida e $i > 0$, então interrompe-se a iteração do método CG-linear e retorna-se o ponto solução $\mathbf{x}^i$ mais recente.

3ª) O passo de Newton $\mathbf{p}_k$ é então definido como sendo o ponto $\mathbf{x}^i$ na condição anterior.

A seguir e, por fim, daremos uma descrição geral do método Newton-CG proposta em Nocedal e Wright(2006, p.140). Escolhemos a sequência *forcing* como

$\eta_k = min\left(0{,}5\,, \sqrt{\|\nabla^2 f_k\|}\ \right)$, pois assim obteremos uma taxa de convergência superlinear.

Método de Newton-CG linear(NOCEDAL; WRIGHT, 2006, p.140)

Tome o ponto inicial $\mathbf{x}^0$

Para $k = 0,1,2, \ldots$

Compute a direção de pesquisa $\mathbf{p}_k$ aplicando o método CG-linear para resolver $\nabla^2 f(\mathbf{x}^k)\mathbf{p} = -\nabla f(\mathbf{x}^k)$, começando pelo ponto $\mathbf{x}^0 = 0$. Termine quando $\|r_k \leq \min\left(0{,}5\ ,\ \sqrt{\|\nabla f_k\|}\ \|\nabla f(\mathbf{x}^k)\|\right.$, ou se uma curvatura negativa é encontrada, como descrito na condição (b).

Tome $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \mathbf{p}_k$, onde α_k satisfaz as condições de Wolfe (4.2) ou as condições de Goldstein (4.3).

Fim_para

5 SIMULAÇÕES DE MINIMIZAÇÃO DE ALGUMAS FUNÇÕES COM OS MÉTODOS POLAK-RIBIÈRE E NEWTON TRUNCADO

Neste capitulo pretendemos fazer algumas simulações de minimização com os dois métodos discutidos no capitulo 3 e capítulo 4. Para tanto, utilizamos como referência as funções de benchmark propostas e descritas no trabalho de Jamil e Yang (2013) . A metodologia adotada nos testes foi a seguinte:

- Os dois métodos minimizaram as mesmas funções, sob as mesmas condições, para que houvesse parâmetros adequados de comparação e se evitasse enviesamento de resultados.
- Os métodos foram executados no sistema de computação em nuvem Google Colab, que oferece uma plataforma de programação em Python com todas as bibliotecas atualizadas em sua última versão
- Utilizou-se a linguagem Python e as bibliotecas *Scipy*, *Numpy* e *Autograd*. Nenhuma alteração foi feita na biblioteca, utilizou-se a configuração padrão de precisão pré-definida pela biblioteca Scipy para cada um dos dois métodos.
- Os métodos utilizados da biblioteca Scipy foram o *Conjugate Gradiente*(referido na documentação do Python como CG e é baseado na versão de Polak-Ribière, uma modificação do original algoritmo proposto por Hestenes e Stiefel) e, o método de Newton Truncado(referido na documentação do Python com *Newton-CG*).

Algumas características que as funções testes possuem são: i) É n-dimensional, ou seja, a quantidade de variáveis é definida de acordo com a conveniência; ii) é contínua, ou seja, não apresenta saltos ou quebras em nenhuma parte do intervalo onde está definida; ii) é derivável, caraterística essa, fundamental para a iteração e convergência dos métodos estudados neste trabalho; iv) Não separável; v) é unimodal, ou seja, possui única inflexão em sua superfície, o que caracteriza a existência de um único mínimo e; vi) não-convexa;

A tabela 5.1 apresenta um resumo das funções utilizadas para simulação dos testes de minimização e de suas características.

Tabela 5.1 - Características das funções de teste

Função	Características

Rosenbrock function $f(\mathrm{x}) = f(x_1, x_2, \dots, x_n) = \sum_{i=1}^{n-1}[b(\mathrm{x}_{i+1} - \mathrm{x}_i^2)^2 + (a - \mathrm{x}_i)^2]$ $a = 1$ e $b = 100$	n-dimensional, Contínua, Derivável, Não-separável, Multimodal e Não-convexa
Ackley 2 Function $f(\mathrm{x}) = f(x_1, x_2)$ $= -200 \exp\left(-0,2\sqrt{x_1^2 + x_2^2}\right)$	Bi-dimensional, Contínua, Derivável, Unimodal e Não-escalável
Wayburn Seader 3 Function $f(\mathrm{x}) = 2\frac{x_1^3}{3} - 8x_1^2 + 33x_1 - x_1x_2 + 5 + [(x_1 - 4)^2 + (x_2 - 5)^2 - 4]^2$	Bi-dimensional Não-separável Unimodal Escalável Derivável
Beale Function $f(\mathrm{x}) = (1,5 - x_1 + x_1x_2)^2 + (2,25 - x_1 + x_1x_2)^2 + (2,625 - x_1 + x_1x_2^3)^2$	Bi-dimensional Contínua Derivável Não-separável Não-escalável Unimodal
Booth Function $f(\mathrm{x}) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$	Bi-dimensional Contínua, Derivável, Não separável, Unimodal e Não-escalável

A seguir apresentamos os gráficos de nível de cada uma das cinco funções:

Rosenbrook Function

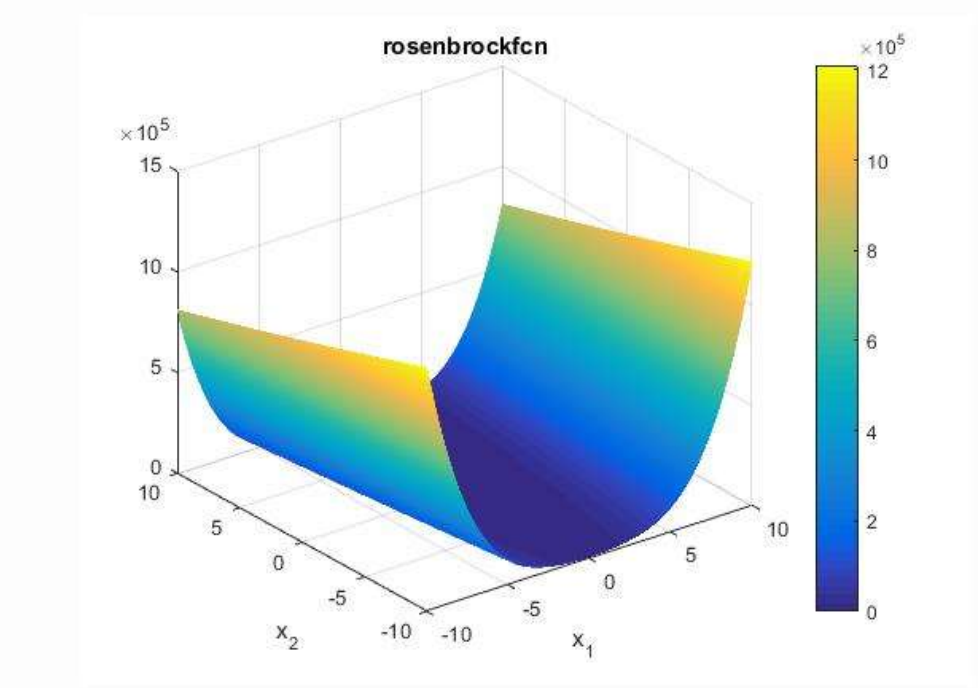

Figura 1 - Gráfico Rosenbrook Function

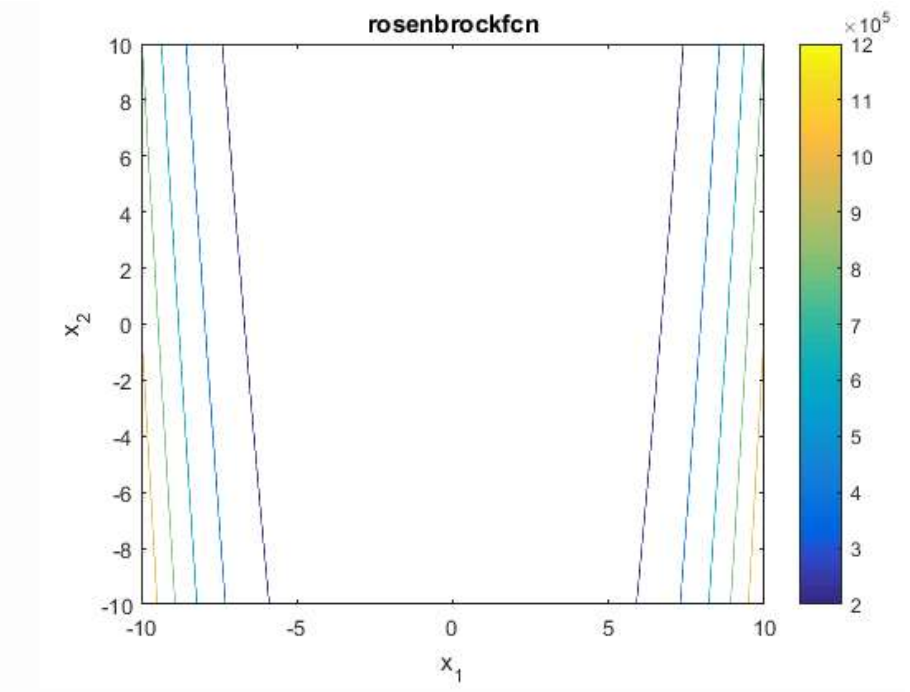

Figura 2 - Curvas de nível da Rosenbrook Function

Ackley 2 Function

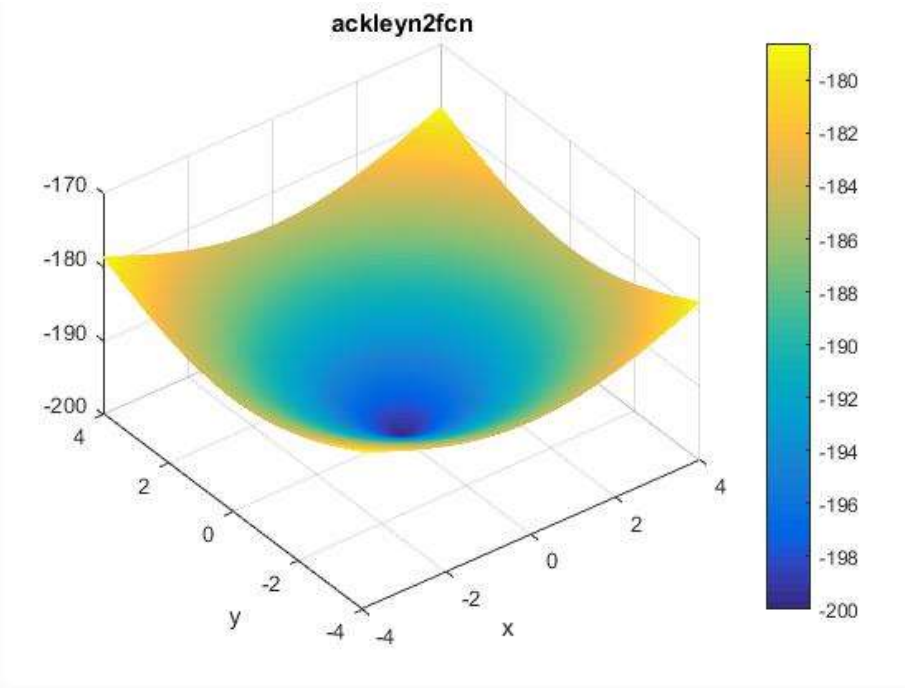

Figura 3 - Gráfico da Ackley 2 Function

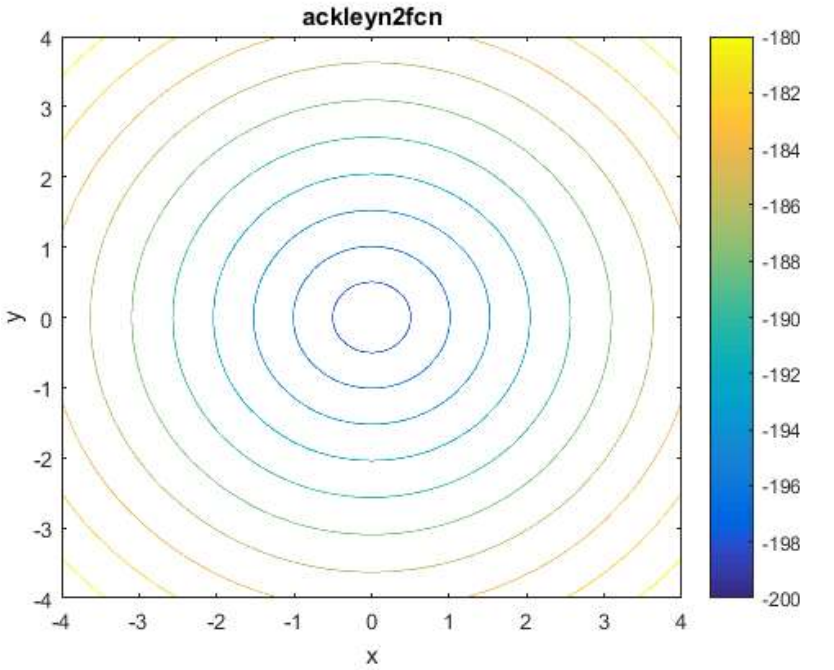

Figura 4 - Curvas de nível da Ackley 2 Function

Beale function

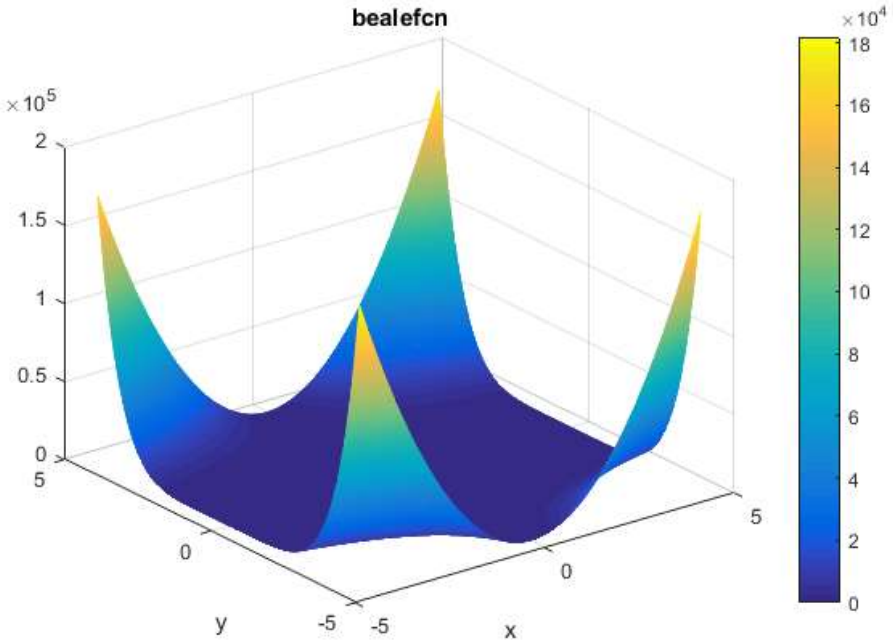

Figura 5 - Gráfico da Beale Function

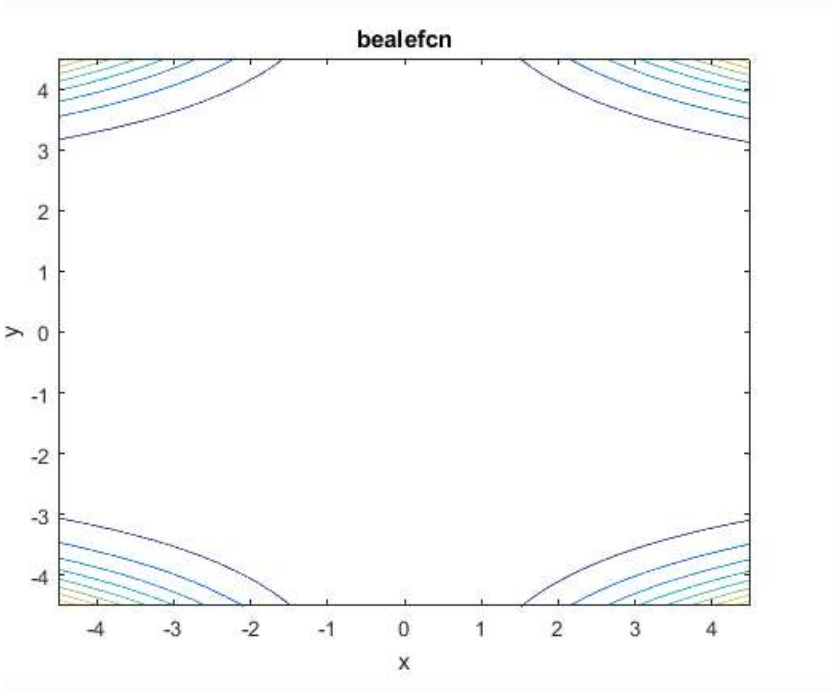

Figura 6 - Curvas de nível da Beale Function

Booth Function

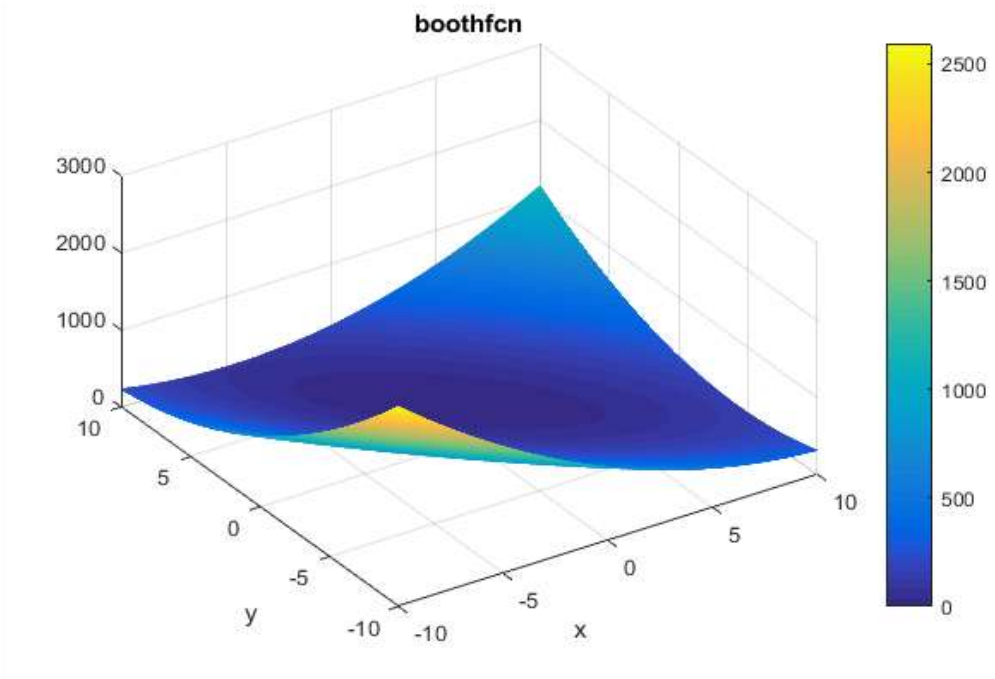

Figura 7 - Gráfico da Booth Funcition

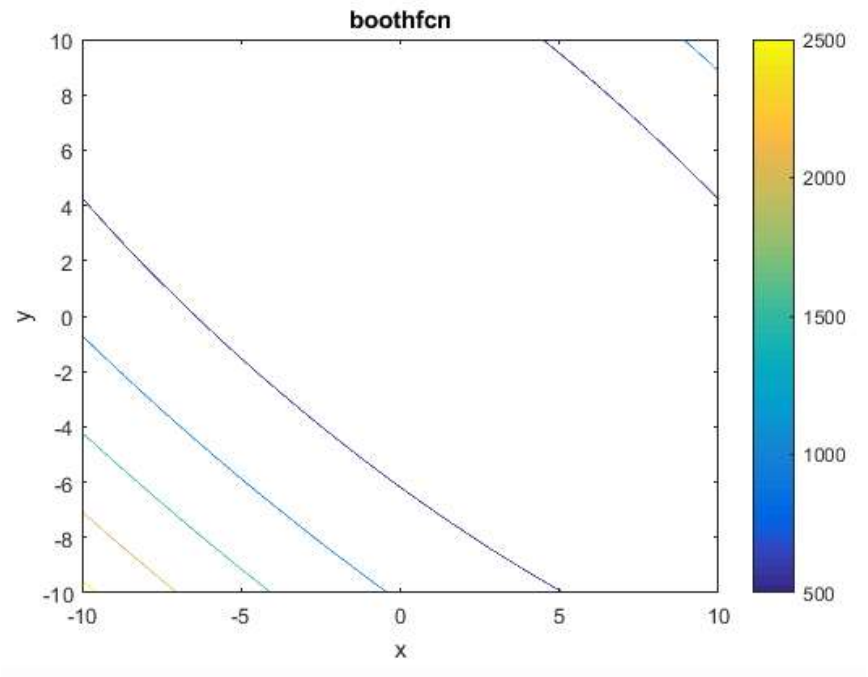

Figura 8 - Curvas de nível da Booth Funcition

Na tabela 5.2 apresentamos a relação dos pontos de mínimos e dos valores mínimos das funções, segundo descrito no trabalho de Jamil e Yang (2013) que traz uma relação bem detalhada de uma quantidade considerável de funções recorrentemente utilizadas na literatura para testes de otimização irrestrita.

Tabela 5.2 - Pontos de mínimo e valores mínimos da função

Função $f(\mathbf{x})$	**Ponto de mínimo** $(\mathbf{x}^*)$	**Valor mínimo da função** $f(\mathbf{x}^*)$
Rosenbrock Function	$\mathbf{x}^* = (1; 1)$	$f(\mathbf{x}^*) = 0$
Ackley 2 Function	$\mathbf{x}^* = (0; 0)$	$f(\mathbf{x}^*) = -200$
Wayburn Seader 3 Function	$\mathbf{x}^* = (5{,}611\,; 6{,}187)$	$f(\mathbf{x}^*) = 21{,}35$
Beale Function	$\mathbf{x}^* = (3; 0{,}5)$	$f(\mathbf{x}^*) = 0$
Both Function	$\mathbf{x}^* = (1; 3)$	$f(\mathbf{x}^*) = 0$

Os resultados retornados pela minimização feita pelos métodos da biblioteca *Scipy* do Python foram tratados e estão apresentados na tabela 5.3. No entanto, os resultados brutos, assim como o algoritmo que implementou todas as funções e métodos utilizados neste trabalho, estão disponíveis no final deste documento em anexo no Apêndice 1.

Tabela 5.3 - Comparação dos métodos na minimização das funções. M_func: Valor Mínimo DA FUNÇÃO; N_it: Número de iterações do método; N_tj: Número de testes do Jacobiano; E_M: Erro entre o valor mínimo obtido pelo método e o valor mínimo na literatura

Rosenbrook Function				
Método ↓	*M_func*	*N_tf*	*N_it*	*N_tj*
Polak-Ribière	$1{,}6251 * 10^{-11}$	180	13	52
Newton Truncado	$1{,}5601 * 10^{-15}$	11	10	52
Ackley 2 Function				
Método ↓	*M_func*	*N_tf*	*N_it*	*N_tj*
Polak-Ribière	$-199{,}9999$	492	1	120
Newton Truncado	$-199{,}1075$	2	1	2

		Wayburn Seader 3 Function		
Método↓	*M_func*	*N_tf*	*N_it*	*N_tj*
Polak-Ribière	19,1058	96	9	24
Newton Truncado	19,1058	31	30	162
		Beale Function		
Método↓	*M_func*	*N_tf*	*N_it*	*N_tj*
Polak-Ribière	$4{,}5209 * 10^{-11}$	92	11	23
Newton Truncado	$2{,}3428 * 10^{-15}$	14	13	84
		Booth Function		
Método↓	*M_func*	*N_tf*	*N_it*	*N_tj*
Polak-Ribière	$4{,}3121 * 10^{-13}$	20	2	5
Newton Truncado	$1{,}1607 * 10^{-17}$	4	3	14

Os resultados da tabela 5.3 nos permitem comparar o desempenho de cada um dos métodos. Utilizando como parâmetro de comparação principal o número de iterações de cada um dos dois métodos (*N_it*), vemos que o método *Conjugate Gradient* (CG) obteve melhores resultados para as funções *Wayburn Seader* 3, *Beale* e *Booth.* No caso da função de *Rosenbrook* vemos que o método de Newton truncado obteve melhores resultados. Para a função de Ackley 2, os números de iterações foram os mesmos e, nesse caso, olhando para os outros indicadores como número de testes do jacobiano e número de testes da função, percebe-se que o método de Newton Truncado obteve melhores resultados.

Outro resultado muito importante é quão bom é aproximação que o método consegue fazer do ponto de mínimo $\mathbf{x}^*$ e, por conseguinte, o quão próximo o método consegue estimar o valor de mínimo da função. A tabela 5.4 nos fornece esse resultado.

Tabela 5.4 - Comparativo dos erros entre valor estimado (M_func) e valor teórico (M_teórico)

		Rosenbrook Function	
Método↓	*M_func*	*M_teórico*	*Erro relativo(%)*
Polak-Ribière	$1{,}6251 * 10^{-11}$	0	0
Newton Truncado	$1{,}5601 * 10^{-15}$	0	0
		Ackley 2 Function	
Método↓	*M_func*	*M_teórico*	*Erro relativo(%)*
Polak-Ribière	$-199{,}9999$	-200	$\rightarrow 0$
Newton Truncado	$-199{,}1075$	-200	$\rightarrow 0$
		Wayburn Seader 3 Function	
Método↓	*M_func*	*M_teórico*	*Erro relativo(%)*
Polak-Ribière	$19{,}1058$	$21{,}35$	10
Newton Truncado	$19{,}1058$	$21{,}35$	10
		Beale Function	
Método↓	*M_func*	*M_teórico*	*Erro relativo(%)*
Polak-Ribière	$4{,}5209 * 10^{-11}$	0	$\rightarrow 0$
Newton Truncado	$2{,}3428 * 10^{-15}$	0	$\rightarrow 0$
		Booth Function	
Método↓		*M_teórico*	*Erro relativo(%)*
Polak-Ribière	$4{,}3121 * 10^{-13}$	0	$\rightarrow 0$
Newton Truncado	$1{,}1607 * 10^{-17}$	0	$\rightarrow 0$

6 CONCLUSÃO

Cremos que o objetivo proposto para este trabalho foi alcançado, pois apresentamos, como estabelecido, uma introdução suscinta mas precisa à otimização contínua, expondo os fundamentos matemáticos necessários, assim como o estudo dos métodos de Polak-Ribière e de Newton truncado. Nesse sentido, destacamos que a contribuição deste trabalho reside na apresentação de um meio simples de se aplicar métodos robustos de otimização(minimização) com o uso da linguagem de programação Python, que é um recurso gratuitamente disponível na internet e de fácil aprendizado. Todavia, os resultados das simulações mostram um detalhe passível de investigação em novos trabalhos, a saber, o fato de o método de Newton Truncado ter apresentado resultados piores que o método de Polak-Ribière para a minimização da maioria das funções utilizadas, contrariando assim, a suposta superioridade de um método de ordem dois sobre um de ordem um.

REFERÊNCIAS

DEMBO, Ron S.; EISENSTAT, Stanley C.; STEIHAUG, Trond. Inexact newton methods. **SIAM Journal on Numerical analysis**, v. 19, n. 2, p. 400-408, 1982.

IZMAILOV, Alexey.; SOLODOV, Mikhail. Otimização volume 2: Métodos Computacionais. 3. Ed, Rio de Janeiro: IMPA, 2018.

JAMIL, Momin; YANG, Xin-She. **A literature survey of benchmark functions for global optimisation problems**. International Journal of Mathematical Modelling and Numerical Optimisation, v. 4, n. 2, p. 150-194, 2013.

LIMA, Elon Lages. **Análise Real, vol. 1**: **Funções de uma variável**. 12. Ed. Rio de Janeiro: IMPA, 2018.

NOCEDAL, Jorge; WRIGHT, Stephen. **Numerical optimization**. Springer Science & Business Media, 2006.

PARKINSON, A. R.; BALLING, R.; HEDENGREN, J. D. Optimization Methods for Engineering Design, Second Edition, Brigham Young University, 2018.

RIBEIRO, Ademir Alves.; KARAS, Elizabeth Wegner. Otimização Continua: Aspectos teóricos e computacionais. **São Paulo: Cengage Learning**, 2013.

APENDICE 1 – CÓDIGO EM PYTHON DOS TESTES DE MINIMIZAÇÃO

```
import warnings
warnings.filterwarnings('ignore')
"""
Created on Sun May 23 16:47:17 2021

@author: joasr
"""
from scipy import optimize
import autograd.numpy as np
import math

import autograd.numpy as np
from autograd import grad, jacobian


# Rosenbrook Function
def Rosenbrook(x):
    return .5*(1-x[0])**2 + (x[1] - x[0]**2)**2

def jac_Rosenbrook(x):
    return np.array((-
2*.5*(1 - x[0]) - 4*x[0]*(x[1] - x[0]**2), 2*(x[1] - x[0]**2)))


# Ackley 2 Function
def Ackley_2(x):
  if isinstance(x, np.numpy_boxes.ArrayBox): #teste se a variável x é u
m ArrayBox, caso seja, acessa seu atributo value e o retorna
    return -200*math.exp(-
0.002*math.sqrt(x[0]._value**2 + x[1]._value**2))
  else:
    return -200*math.exp(-0.002*math.sqrt(x[0]**2 + x[1]**2))


# Wayburn Seader 3 Function
def Wayburn_S3(x):
    return 2*(x[0]**3/3) - 8*x[0]**2 + 33*x[0] -
x[0]*x[1] + 5 + ((x[0] - 4)**2 + (x[1] - 5)**2 - 4 )**2

# Beale Function
def Beale(x):
  return (1.5 - x[0] + x[0]*x[1])**2 + (2.25 - x[0] + x[0]*(x[1]**2))**
2 + (2.625 - x[0] + x[0]*(x[1]**3))**2

# Booth Function
def Booth(x):
```

```
  return (x[0] + 2*x[1] -7)**2 + (2*x[0] + x[1] - 5)**2

r1 = optimize.minimize(Rosenbrook, [2, -1], method="CG")
r2 = optimize.minimize(Rosenbrook, [2, -1], method="Newton-
CG", jac=jacobian(Rosenbrook))

r3 = optimize.minimize(Ackley_2, [2, -1], method="CG")
r4 = optimize.minimize(Ackley_2, [2, -1], method="Newton-
CG", jac=jacobian(Ackley_2))

r5 = optimize.minimize(Wayburn_S3, [2, -1], method="CG")
r6 = optimize.minimize(Wayburn_S3, [2, -1], method="Newton-
CG", jac=jacobian(Wayburn_S3))

r7 = optimize.minimize(Beale, [2, -1], method="CG")
r8 = optimize.minimize(Beale, [2, -1], method="Newton-
CG", jac=jacobian(Beale))

r9 = optimize.minimize(Booth, [2, -1], method="CG")
r10 = optimize.minimize(Booth, [2, -1], method="Newton-
CG", jac=jacobian(Booth))


print(float(r1.fun)**11)
print("Método CG - Rosenbrook \n",r1)
print("\n\n\nMétodo Newton-CG - Rosenbrook \n",r2)


print("\n\n\n Método CG - Ackley_2 \n",r3)
print("\n\n\n Método Newton-CG - Ackley_2 \n",r4)

print("\n\n\n Método CG - Wayburn_S3 \n",r5)
print("\n\n\n Método Newton-CG - Wayburn_S3 \n",r6)

print("\n\n\n Método CG - Beale \n",r7)
print("\n\n\n Método Newton-CG - Beale \n",r8)

print("\n\n\n Método CG - Booth \n",r9)
print("\n\n\n Método Newton-CG - Booth \n",r10)
```

APENDICE 2 – RESULTADOS BRUTOS DA EXECUÇÃO DO CÓDIGO

```
Método CG - Rosenbrook
      fun: 1.650917861123459e-11
     jac: array([-6.15448419e-06,  2.53833953e-07])
 message: 'Optimization terminated successfully.'
    nfev: 108
     nit: 13
    njev: 27
  status: 0
 success: True
       x: array([0.99999426, 0.99998863])



Método Newton-CG - Rosenbrook
      fun: 1.5601357400786612e-15
     jac: array([ 1.05753092e-07, -7.48325277e-08])
 message: 'Optimization terminated successfully.'
    nfev: 11
    nhev: 0
     nit: 10
    njev: 52
  status: 0
 success: True
       x: array([0.99999995, 0.99999988])



 Método CG - Ackley_2
      fun: -199.99999999956327
     jac: array([0.34470749, 0.3846302 ])
 message: 'Desired error not necessarily achieved due to precision
loss.'
    nfev: 492
     nit: 1
    njev: 120
  status: 2
 success: False
       x: array([-9.76534853e-10,  4.88267426e-10])



 Método Newton-CG - Ackley_2
      fun: -199.10756983090647
     jac: array([0., 0.])
 message: 'Optimization terminated successfully.'
    nfev: 2
    nhev: 0
     nit: 1
    njev: 2
  status: 0
 success: True
       x: array([ 2., -1.])
```

```
Método CG - Wayburn_S3
     fun: 19.105879794568004
    jac: array([ 1.19209290e-06, -2.38418579e-07])
message: 'Optimization terminated successfully.'
   nfev: 96
    nit: 9
   njev: 24
 status: 0
success: True
      x: array([5.14689673, 6.83958974])
```

```
Método Newton-CG - Wayburn_S3
     fun: 19.105879794568025
    jac: array([5.08273568e-06, 8.04771483e-06])
message: 'Optimization terminated successfully.'
   nfev: 31
   nhev: 0
    nit: 30
   njev: 162
 status: 0
success: True
      x: array([5.14689674, 6.83958975])
```

```
Método CG - Beale
     fun: 4.5209698652403765e-11
    jac: array([-4.75759725e-06, -2.05721620e-06])
message: 'Optimization terminated successfully.'
   nfev: 92
    nit: 11
   njev: 23
 status: 0
success: True
      x: array([2.99998321, 0.49999578])
```

```
Método Newton-CG - Beale
     fun: 3.753911232411541e-15
    jac: array([-5.20798790e-07,  8.57500417e-07])
message: 'Optimization terminated successfully.'
   nfev: 14
   nhev: 0
    nit: 13
   njev: 74
 status: 0
success: True
      x: array([2.99999997, 0.5       ])
```

```
Método CG - Booth
```

```
     fun: 4.3121006620681157e-13
    jac: array([2.80883339e-06, 2.90473955e-06])
message: 'Optimization terminated successfully.'
   nfev: 20
    nit: 2
   njev: 5
 status: 0
success: True
      x: array([1.00000013, 3.00000018])
```

Método Newton-CG - Booth

```
     fun: 1.1607378107326936e-17
    jac: array([ 1.20051062e-08, -7.97238542e-09])
message: 'Optimization terminated successfully.'
   nfev: 4
   nhev: 0
    nit: 3
   njev: 14
 status: 0
success: True
      x: array([1., 3.])
```