Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Raízes de Equações Convexas em $\mathbb{R}^n$

Bianca Costa Guimarães

Belo Horizonte

Julho de 2004

Bianca Costa Guimarães

Raízes de Equações Convexas em $\mathbb{R}^n$

Dissertação apresentada ao curso de Pós-Graduação do Departamento de Ciência da Computação da Universidade Federal de Minas Gerais, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Belo Horizonte

2 de Julho de 2004

À
minha querida mãe, Maria Alice,
meu querido pai, Guimarães.

Agradecimentos

Gostaria de agradecer a todos que contribuíram de forma direta ou indireta para a realização deste trabalho:

Aos meus pais, Guimarães e Alice e ao meu irmão Bruno, pela convivência harmoniosa em família e pela educação baseada em nobres valores humanos e cristãos.

Ao André, que sempre esteve ao meu lado me proporcionando momentos especiais e me incentivando na busca dos meus ideais.

À CAPES, pelo apoio financeiro.

Às colegas, Silvana e Kíssia, minhas colaboradoras e exemplos em quem me espelhava para seguir adiante.

Aos professores e funcionários do Departamento de Ciência da Computação da UFMG.

Ao coordenador do curso, Geraldo Robson Mateus, que atráves de ótimas aulas me passou um conhecimento abrangente e seguro na área de Otimização.

Ao Prof. Frederico F. Campos, filho, por me ter dado a oportunidade de iniciar uma carreira de professora universitária em Cálculo Numérico e claro, pela indicação do caminho e força para vencer as barreiras impostas pela área de Ciência da Computação.

Ao meu orientador de mestrado, Marcos Augusto dos Santos, que me proporcionou alternativas neste trabalho e me guiou, cotidianamente, durante o seu desenvolvimento.

Uma menção especial à Deus que meu deu uma vida com saúde e sabedoria.

Resumo

Neste trabalho é apresentado um método numérico para refinar raízes de equações de problemas convexos com dimensões superiores a um, que é obtido através da generalização do método de Newton. Para tanto, o método utiliza uma função convexa escrita como a diferença de duas funções, uma côncava e outra convexa, e seus respectivos hiperplanos de suporte. Geometricamente, cada iteração pode ser interpretada como o ponto de interseção dos hiperplanos de suporte. São apresentados alguns exemplos numéricos existentes na literatura e outros propostos por nós. Comparamos o nosso método com o método de Newton-Raphson para problemas diferenciáveis multidimensionais. Expandimos os testes para a classe de problemas não necessariamente diferenciáveis, onde o método de Newton-Raphson não pode ser aplicado dada a ausência de informações de segunda ordem. Optou-se por construir esses problemas a partir da literatura de programação convexa não suave. Problemas de encontrar o zero de funções convexas não suave foram testados substituindo o conceito de gradiente pelo conceito de subgradiente.

Abstract

We present a method numerical for finding a real root of a convex function in a euclidean space of dimension above one, that is obtained from generalization of Newton method. The method uses a convex function writing as the difference of two functions, a concave one and other convex one, and your respective support hyperplanes. Geometrically, each iteration can be interpreted like a intersection point of the support hyperplanes. Some numerical tests are presented, where compare our method with a Newton-Raphson method, for differentiables problems. Expand the tests for convex nonsmooth problems, where the Newton-Raphson method can not be applied given the absence of informations of second order. Problems for finding a root of a convex function non necessarily differentiable were tested. In these cases, the gradient concept was changed to subgradient concept.

Sumário

1 Introdução **1**

2 Métodos de Newton para resolução de equações algébricas e transcendentes **3**

2.1 Método de Newton 4

2.2 Método de Newton-Raphson 5

3 Generalização do método de Newton em $\mathbb{R}^n$ - método das projeções **9**

3.1 O método de Newton em $\mathbb{R}^n$ - método das projeções 10

3.2 Exemplo do método das projeções 15

3.3 Alguns fundamentos teóricos 19

4 Experiência numérica **25**

4.1 Problemas diferenciáveis 29

4.2 Problemas não necessariamente diferenciáveis 37

5 Conclusões **46**

A Problemas diferenciáveis e não diferenciáveis **50**

A.1 Funções diferenciáveis em $\mathbb{R}^n$, onde $n = 1$. 50

A.2 Funções diferenciáveis em $\mathbb{R}^n$ 55

A.3 Funções diferenciáveis não separáveis 57

A.4 Funções convexas não suaves 59

Lista de Tabelas

3.1 Desempenho do método para a função $f(x) = 5x_1^3 - 2x_2^2 + 8x_1 - 10$. 19

4.1 Resultados numéricos para metódos de raízes de equações (TRbis) 31

4.2 Resultados numéricos para metódos de raízes de equações (Iterações) 32

4.3 Resultados para $n = 1$ 33

4.4 Resultados Numéricos para $x \in R^n$ 34

4.5 Resultados para $x \in R^n$ com funções não separáveis 34

4.6 Método das projeções aplicado ao Teste 29. 35

4.7 Método das projeções aplicado ao Teste 36 onde $x \in R^5$. 36

4.8 Desempenho do método para a quadrática $Q(x)$ 37

4.9 Desempenho dos métodos de Otimização convexa não suave nos problemas adaptados para produzir os nossos testes. 40

4.10 Resultados numéricos para o problema não diferenciável CB3. 41

4.11 Resultados numéricos para o problema não diferenciável DEM. 42

4.12 Resultados numéricos para o problema não diferenciável LQ. 42

4.13 Resultados numéricos para o problema não diferenciável Mifflin2. 43

4.14 Resultados numéricos para o problema não diferenciável Maxq. 44

4.15 Resultados numéricos para o problema não diferenciável Maxl. 45

Lista de Figuras

2.1 Interpretação geométrica do método de Newton. 5

2.2 Algoritmo de Newton [Cam01]. 6

2.3 Interpretação geométrica do método de Newton-Raphson. 7

2.4 Algoritmo de Newton-Raphson para resolver equações algébricas e transcendentes. 8

3.1 $f(x) = g(x) - h(x)$; onde $x_0 = 2$; $g(x_0) > h(x_0)$ e $x^* \approx 1.4$ 10

3.2 Método de relaxação usando $\lambda > 1$. 12

3.3 Método de relaxação usando λ conveniente. 12

3.4 Aproximações lineares de $G(w)$ e $H(w)$ no ponto w_0. 13

3.5 Algoritmo para encontrar ponto de interseção de dois hiperplanos. 13

3.6 Algoritmo para encontrar uma raiz de uma equação convexa em R^n. 14

3.7 Método de relaxação. 17

3.8 Interpretação geométrica da prova do lema 4.1 20

4.1 Algoritmo para encontrar uma raiz de uma equação convexa em R^n segundo o paradigma $x_{k+1} = x_k + t_k \, .d_k$ 27

4.2 Algoritmo de descida para encontrar uma raiz de uma equação convexa em R^n. 28

4.3 Comparação dos métodos em $\mathbb{R}^n$. 30

4.4 Valores de $f(x_k)$ sugerem convergência quadrática. 35

4.5 (a)Valores de $f(x_k)$ produziram uma seqüência monótona decrescente, (b)Gráfico ampliado. 38

5.1 Forma mista para a resolução de um sistema de equações lineares usando as funções $Q(x)$ e $D(x)$. 47

A.1 Exemplo CB3 (teste 49) de função convexa não necessariamente diferenciável. 60

A.2 Exemplo DEM (teste 50) de função convexa não necessariamente diferenciável. 60

A.3 Exemplo QL (teste 51) de função convexa não necessariamente diferenciável. 61

A.4 Exemplo LQ (teste 52) de função convexa não necessariamente diferenciável. 62

A.5 Exemplo Mifflin1 (teste 53) de função convexa não necessariamente diferenciável. 62

A.6 Exemplo Mifflin2 (teste 54) de função convexa não necessariamente diferenciável. 63

Capítulo 1

Introdução

A resolução de equações algébricas e transcendentes está no cerne de várias aplicações e problemas da área de Ciências Exatas; é a "mais básica das tarefas em matemática aplicada" [PTVF92]. Os valores especiais que satisfazem a equação

$$f(x) = 0, \tag{1.1}$$

são chamados de raízes da equação (1.1) ou zeros da função $f(x)$. Nos casos mais simples em que $x \in \mathbb{R}$, o problema de calcular uma raiz pode ser dividido em duas fases [Cam01]:

1. isolamento da raiz, que consiste em encontrar um intervalo [a,b] que contenha uma, e somente uma, raiz de $f(x) = 0$ e,

2. refinamento da raiz, que a partir de um valor inicial $x_0 \in [a, b]$, gera uma seqüência $\{x_0, x_1, x_2, ..., x_k, ...\}$ que converge para uma raiz exata ξ de $f(x) = 0$.

Para o refinamento da raiz existem diversos métodos. Dentre eles destacam-se os métodos pégaso, de Newton e de Brent. Quando a equação envolve várias variáveis, é usual modelar o problema como um problema de otimização [PTVF92], onde busca-se, por exemplo, uma solução que minimiza $(f(x))^2$.

A dissertação aborda um novo método numérico para refinar raízes de equações de problemas convexos com dimensões superiores a um. No que se segue, o método é referenciado como método das projeções. Os testes sugerem convergência quadrática para uma classe específica de problemas na qual as variáveis podem ser separadas em funções de uma única variável, isto é,

$$f(x) = \sum_{i=1}^{n} f_i(x_i) + k, \tag{1.2}$$

onde $x \in \mathbb{R}^n$, $x_i \in R$, k é uma constante real, $f_i(x_i) \in C^1$ e $f_i(x_i)$ é convexa. Nestes casos, todos os testes produziram uma seqüência monótona estritamente decrescente dos valores de $f(x)$.

O método de Newton-Raphson também resolve eficientemente esta mesma classe de problemas. Embora as comparações de desempenho tenham sido favoráveis ao método tratado neste texto, há de se ressaltar que o algoritmo de Newton-Raphson é facilmente implementado para a classe de funções (1.2), caso $f(x) \in C^2$. Entretanto, em problemas convexos não suaves, o método de Newton-Raphson não pode ser aplicado dada a ausência de informações de segunda ordem. Essa classe de problemas pode ser naturalmente abordada pelo método proposto; sua generalização para a resolução de problemas não suaves é obtida substituindo-se o conceito do gradiente (informação de primeira ordem) pelo conceito de subgradiente.

A dissertação tem os seguintes objetivos:

1. consolidar resultados obtidos com o método [dS02] para funções convexas diferenciáveis (classe C^1) em $\mathbb{R}^n$ e,

2. explorar o desempenho do método em problemas que envolvem funções convexas não necessariamente diferenciáveis. Problemas clássicos da literatura, baseados naqueles presentes em [Kiw90],[LE77],[MN92],[Sho85],[Shr89], são testados e confrontados com resultados documentados.

Este trabalho está organizado em cinco capítulos dos quais este é o primeiro. Com o objetivo meramente didático, o capítulo 2 descreve como o problema 1.1 é resolvido pelos métodos de Newton e de Newton-Raphson. A nossa principal contribuição encontra-se no capítulo 3, onde tem-se uma generalização do Método de Newton para a resolução de problemas $n-$dimensionais. A experiência numérica encontra-se no capítulo 4. O capítulo 5 apresenta as conclusões sobre este trabalho e as perspectivas para desenvolvimento de trabalhos futuros. Finalizando, tem-se as referências bibliográficas e o apêndice, onde a relação de problemas resolvidos são apresentadas.

Capítulo 2

Métodos de Newton para resolução de equações algébricas e transcendentes

Um modelo, no sentido em que é apreendido pela programação matemática, é uma abstração útil da realidade; é uma simplificação que torna possível a aplicação de uma dada metodologia.

A expansão em série de Taylor para uma função real $f(x)$ diferenciável com $x \in \mathbb{R}$ em torno de um ponto x_k;

$$f(x) = f(x_k) + (x - x_k)f'(x_k) + \frac{1}{2!}(x - x_k)^2 f''(x_k) + \frac{1}{3!}(x - x_k)^3 f'''(x_k) + \ldots;$$

permite gerar funções que modelam $f(x)$ pelo truncamento de seus termos, em geral, de ordem superior a um e dois:

$$f(x) \approx f_k^{\triangle}(x) = f(x_k) + (x - x_k)f'(x_k)$$

e

$$f(x) \approx f_k^{\triangle}(x) = f(x_k) + (x - x_k)f'(x_k) + \frac{1}{2}(x - x_k)^2 f''(x_k)\,.$$

A função modelo $f_k^{\triangle}(x)$ é uma representação aceitável para $f(x)$ que é válida em uma região próxima de x_k.

Nos casos em que $x \in \mathbb{R}^n$ e $f(x) \in C^2$, utilizamos duas classes de modelo no ponto x_k;

$$f(x) \approx f_k^{\triangle}(x) = f(x_k) + (x - x_k)^T \nabla f(x_k)\,; \tag{2.1}$$

e

$$f(x) \approx f_k^{\triangle}(x) = f(x_k) + (x - x_k)^T \nabla f(x_k) + \frac{1}{2}(x - x_k)^T H(x_k)(x - x_k)\,; \tag{2.2}$$

onde $\nabla f(x_k) = \begin{pmatrix} \frac{\partial f(x_k)}{\partial x_1} \\ \vdots \\ \frac{\partial f(x_k)}{\partial x_n} \end{pmatrix}$ é o vetor gradiente no ponto x_k e

$$H(x_k) = \begin{pmatrix} \frac{\partial f(x_k)}{\partial x_1 \partial x_1} & \frac{\partial f(x_k)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial f(x_k)}{\partial x_1 \partial x_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f(x_k)}{\partial x_n \partial x_1} & \frac{\partial f(x_k)}{\partial x_n \partial x_2} & \cdots & \frac{\partial f(x_k)}{\partial x_n \partial x_n} \end{pmatrix}$$

é a matriz hessiana de $f(x)$ no ponto x_k.

Este capítulo faz uma revisão didática do método de Newton para a resolução de equações algébricas e transcendentes e discute a aplicação do método de Newton-Raphson nesse contexto. No restante do trabalho, uma referência ao método de Newton corresponde a uma referência ao método do item 2.1, que resolve problemas de raízes de equações em $\mathbb{R}$. Problemas multidimensionais são resolvidos com o método de Newton-Raphson. O nosso trabalho consiste em estudar uma generalização do método de Newton para resolver problemas nos quais $x \in \mathbb{R}^n$.

2.1 Método de Newton

O método de Newton, conhecido também como o método das tangentes, resolve problemas para o cálculo de raízes de equações unidimensionais. Ele é equivalente a substituir um pequeno arco da curva $y = f(x)$ por uma reta tangente à curva (figura 2.1).

Seja $f : x \in \mathbb{R} \longrightarrow \mathbb{R}$ um funcional e [a,b] um intervalo onde f, f' e f'' são contínuas e $f'(x) \neq 0$. Para obter uma aproximação x_1 da raiz ξ, traça-se, a partir de $B_0\,[x_0, f(x_0)]$ uma reta tangente à curva $y = f(x)$. Esta intercepta o eixo das abscissas no ponto x_1. De $B_1\,[x_1, f(x_1)]$ traça-se outra reta tangente à curva que corta esse mesmo eixo no ponto x_2, sendo este ponto uma melhor aproximação da raiz. O processo é repetido até que seja encontrada uma aproximação da raiz ξ segundo uma tolerância pré-estabelecida.

Geometricamente, tem-se que:

$$tg\,\alpha = \frac{f(x_0)}{x_0 - x_1} = f'(x_0), \text{ então}$$

$$x_0 - x_1 = \frac{f(x_0)}{f'(x_0)} \text{ ou}$$

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}\,.$$

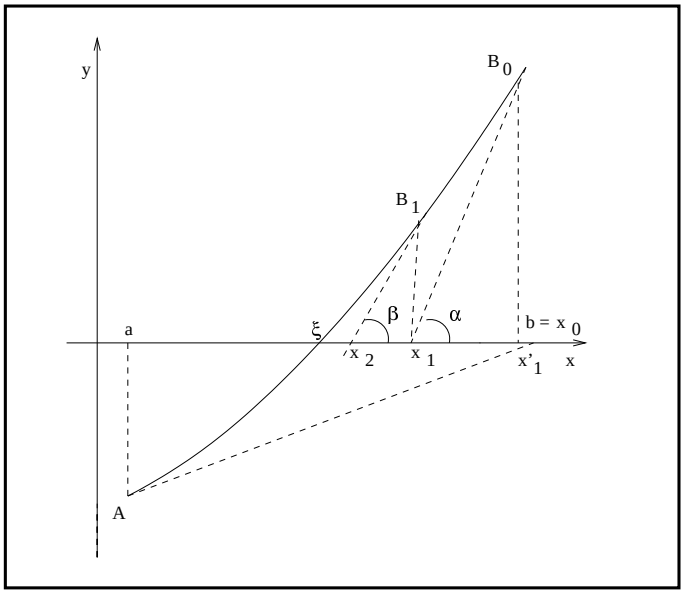


Figura 2.1: Interpretação geométrica do método de Newton.

Já $tg\,\beta$ pode ser escrita como,

$$tg\,\beta = \frac{f(x_1)}{x_1 - x_2} = f'(x_1), \text{ que produz}$$

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}\,.$$

Generalizando, tem-se que

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \quad k = 0,\ 1,\ 2, \ldots$$

Para uma descrição mais formal, ver [Cam01].

Pela figura 2.1 vê-se que traçando a tangente a partir de $A\,[x_0, f(x_0)]$ pode-se encontrar um ponto $x_1' \notin [a, b]$ e o método de Newton pode não convergir. É condição suficiente para a convergência do método de Newton que $f'(x)$ e $f''(x)$ sejam não nulas e preservem o sinal em (a, b) e x_0 seja tal que $f'(x_0).f''(x_0) > 0$.

O algoritmo é mostrado na figura 2.2.

2.2 Método de Newton-Raphson

O método de Newton-Raphson pode ser facilmente adaptado para resolver equações algébricas e transcendentes.

```
Algoritmo Newton
{ Objetivo: Calcular a raiz de uma equação pelo método de Newton }
parâmetros de entrada x0, Toler, IterMax
   { valor inicial, tolerância e número máximo de iterações }
parâmetros de saída Raiz, Iter, Erro
   { raiz, número gasto de iterações e condição de erro }
   { Avaliar a função f(x) e sua derivada f'(x) em x = x0 }
   Fx ← f(x0); DFx ← f'(x0); x ← x0; Iter ← 0
   repita
      DeltaX ← −Fx/DFx; x ← x + DeltaX
      Fx ← f(x); DFx ← f'(x); { Avaliar a função f(x) e sua derivada f'(x) }
      Iter ← Iter + 1
      escreva Iter, x, Fx, DeltaX
      se (abs(DeltaX)<Toler e abs(Fx)<Toler) ou abs(DFx)=0 ou Iter≥IterMax
         então interrompa
      fim se
   fim repita
   Raiz ← x
   se abs(Fx) < Toler então
      Erro ← 0
   senão
      Erro ← 1
   fim se
fim algoritmo
```

Figura 2.2: Algoritmo de Newton [Cam01].

Seja $f(x) : \mathbb{R}^n \longrightarrow \mathbb{R}$, uma função contínua, convexa e de classe C^2. Considere $f(x) \approx f_k^{\triangle} = Q_k(x)$ uma aproximação quadrática de $f(x)$ em x_k, segundo (2.2) (ver figura 2.3). A cada iteração k, computa-se o mínimo de $Q_k(x)$, que será o novo ponto x_{k+1}:

$$x_{k+1} = \arg\ min\ Q_k(x)$$

O processo é repetido enquanto $f(x_{k+1}) > 0$. Quando $f(x_{k+1}) \leq 0$, busca-se o valor do passo $t \geq 0$ que satisfaz $f(x_k + t.\, d_k) = 0$. A direção d_k é conhecida como direção de Newton e é obtida da seguinte forma. Como

$$Q_k(x) = f(x_k) + (x - x_k)^T\, \nabla f(x_k) + \frac{1}{2}(x - x_k)^T H(x_k)(x - x_k)\,,$$

faz-se $x - x_k = d_k$ e, após derivar e igualar a zero, obtém-se

$$\nabla f(x_k) + H(x_k).\, d_k = 0\,,$$

ou $d_k = -H^{-1}(x_k).\nabla f(x_k)$.

O algoritmo encontra-se em 2.4.

O método de Newton é generalizável para resolver os problemas em que $x \in \mathbb{R}^n$, conforme pode ser visto no capítulo 3. As comparações de desempenho que são expostas no capítulo 4, são realizadas comparando-se resultados obtidos com esta generalização com os resultados obtidos com o método de Newton-Raphson.

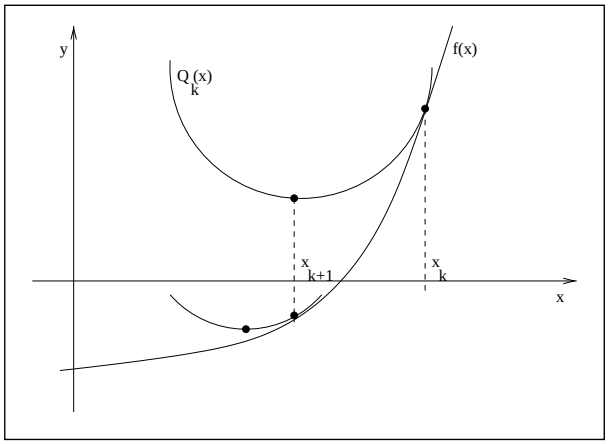


Figura 2.3: Interpretação geométrica do método de Newton-Raphson.

```
Algoritmo Newton-Raphson
{ Objetivo: Aplica o método de Newton-Raphson para determinar x̄ : f(x̄) = 0. }
parâmetros de entrada x_0, kmax, Toler
   { solução inicial, número máximo de iterações, tolerância máxima }
parâmetros de saída x { solução }
   k ← 0;
   repita
      se |f(x_k)| < Toler ou k = kmax
         então interrompa
      fim se
      Calcule H(x_k) e ▽f(x_k)
      Resolva o sistema de equações: H(x_k) .d_k = − ▽ f(x_k)
      x_{k+1} ← x_k + t.d_k
      se f(x_{k+1}) < 0
         então
            determine: t̄ : f(x_k + t̄.d_k) = 0
            x_{k+1} ← x_k + t̄.d_k
      fim se
      k ← k + 1
   fim repita
fim algoritmo
```

Figura 2.4: Algoritmo de Newton-Raphson para resolver equações algébricas e transcendentes.

Capítulo 3

Generalização do método de Newton em $\mathbb{R}^n$ - método das projeções

Fórmulas fechadas para computar zeros de funções só existem para classes muito restritas de problemas. A mais notória é a fórmula resolutiva proposta pelo matemático hindu Bhaskara (1114-1158), que se baseou nos trabalhos do matemático árabe al-Khowarizmi, para resolver equações de segundo grau. Nos casos gerais são os métodos iterativos que são utilizados. Dentre eles destaca-se o método de Newton aplicável para funções contínuas $f(x) : x \in \mathbb{R} \longrightarrow \mathbb{R}$; $f(x)$ é modelada pela expansão em série de Taylor de primeira ordem e a iteração é obtida com este modelo (ver capítulo 2). Dado um ponto inicial, suficientemente próximo da raiz, o método tem taxa de convergência quadrática para problemas com grau de multiplicidade igual a um [DM76]. O principal problema acontece na busca de raízes com grau de multiplicidade superior a um, quando então o método passa a ter convergência linear. Se é conhecido o grau de multiplicidade da raiz, o método de Schröder [Sch70] contorna este problema; a modificação de x_k é magnificada por um fator m correspondente ao grau de multiplicidade da raiz.

Neste capítulo discute-se a utilização do método de Newton em problemas que envolvem várias variáveis ($x \in \mathbb{R}^n$). Resumidamente, na iteração k substitui-se $f(x)$ pela sua expansão de primeira ordem em x_k;

$$f(x) \approx f_k^{\triangle}(x) = f(x_k) + \nabla f(x_k)^T(x - x_k);$$

e é dada uma interpretação que possibilita obter $x_{k+1} : f_k^{\triangle}(x) = 0$. Obviamente, aplicando-se o mesmo procedimento no caso particular em que $x \in \mathbb{R}$, o algoritmo funciona exatamente como o método de Newton.

3.1 O método de Newton em $\mathbb{R}^n$ - método das projeções

Geometricamente, cada iteração está relacionada com um ponto pertencente à interseção de dois hiperplanos que suportam duas funções obtidas por uma separação dos termos da função original. Seja dada $f : x \in \mathbb{R}^n \longrightarrow \mathbb{R}$, convexa de classe C^1 e $f(x) = g(x) - h(x)$.

Encontrar o zero de $f(x)$ é equivalente a determinar um ponto $x^* : g(x^*) = h(x^*)$ (ver figura 3.1). A iteração principal do algoritmo está associada aos métodos de relaxação para resolver o problema da viabilidade linear [Agm54], [Bre65], [EM67]. Dados dois hiperplanos, $G(w)$ e $H(w)$;

$$\begin{aligned} G(w) &= \{w \in R^{n+1} : a_j^T w = b_j\} \\ H(w) &= \{w \in R^{n+1} : a_i^T w = b_i\}, \end{aligned}$$

onde $a_i,\ a_j \in \mathbb{R}^{n+1}$ e $b_i, b_j \in \mathbb{R}$;
tais que $G(w) \cap H(w) \neq \emptyset$, um algoritmo para resolver o problema da viabilidade algébrica linear busca encontrar

$$\tilde{w} : \tilde{w} \in G(w) \cap H(w)\,.$$

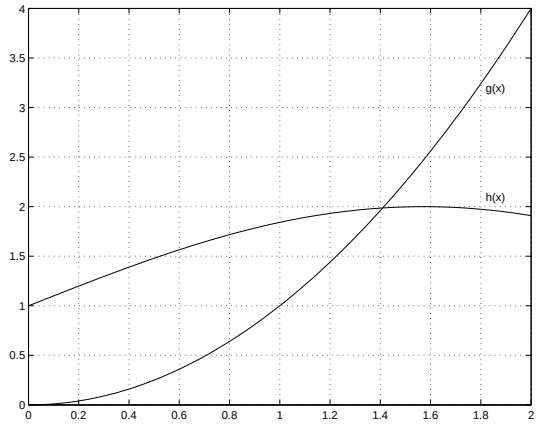


Figura 3.1: $f(x) = g(x) - h(x)$; onde $x_0 = 2$; $g(x_0) > h(x_0)$ e $x^* \approx 1.4$.

Os hiperplanos de suporte para a função convexa $g(x)$ e para a função côncava $h(x)$ em x_k são dados pela expansão de primeira ordem,

$$\begin{aligned} r &= h(x_k) + \nabla h(x_k)^T (x - x_k) \\ r &= g(x_k) + \nabla g(x_k)^T (x - x_k) \end{aligned}$$

onde $r \in R$, $\nabla h(x_k)$ e $\nabla g(x_k)$ são os vetores gradiente de $h(x)$ e $g(x)$ no ponto x_k. Rearranjando as expressões e mudando a notação, temos:

$$\begin{aligned} (a_i^k)^T w &= b_i^k \text{ e} \\ (a_j^k)^T w &= b_j^k , \end{aligned}$$

onde

$$\begin{aligned} w &= (x^T, r)^T , \\ a_j^k &= (\nabla g(x_k)^T, -1)^T , \\ a_i^k &= (\nabla h(x_k)^T, -1)^T , \\ b_j^k &= \nabla g(x_k)^T x_k - g(x_k) \text{ e} \\ b_i^k &= \nabla h(x_k)^T x_k - h(x_k) . \end{aligned}$$

Supõe-se conhecido w_0, um ponto inicial, tal que

$$w_0 \in H(w) : a_j^T w_0 > a_i^T w_0 .$$

Na iteração k, o método de relaxação aplicado ao sistema de inequações lineares

$$\begin{cases} a_i^T w_k \leq b_i \\ a_j^T w_k \leq b_j \end{cases}$$

computa w_{k+1} utilizando

$$w_{k+1} := w_k + \lambda(w_p - w_k) ,$$

onde $\lambda > 0$ é o parâmetro de relaxação e w_p é dado por

$$w_p = w_k + a_j \frac{(b_j - a_j^T w_k)}{||a_j||^2} ,$$

que é a projeção ortogonal de w_k no hiperplano associado à região inviável para w_k.

O método de relaxação pode consumir um número exponencial de iterações (veja figura 3.2), mas na instância de duas inequações e utilizando um valor conveniente para λ, é possível encontrar uma solução com uma única iteração. Na figura 3.3, a projeção do ponto

$w_k \in H(w)$ (que é w_p) é relaxada, tal que, w_{k+1} projetado em $H(w)$, produz o ponto desejado, $\tilde{w} \in G(w) \cap H(w)$. A síntese deste procedimento está no algoritmo da figura 3.5, onde foi utilizado um valor λ que está relacionado com o ângulo θ existente entre os hiperplanos $G(w)$ e $H(w)$. A dedução deste valor de λ, dado pela cossecante ao quadrado de θ, encontra-se na seção 3.3 .

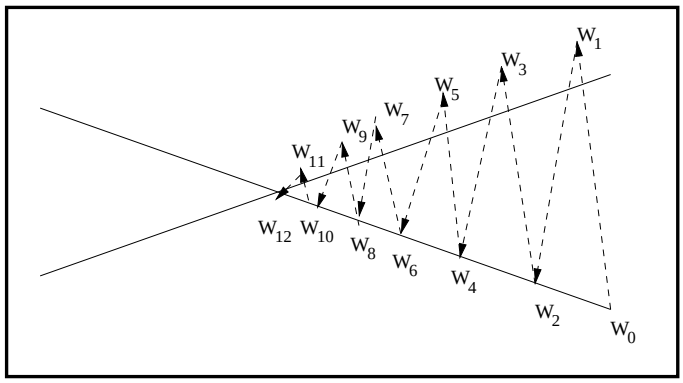


Figura 3.2: Método de relaxação usando $\lambda > 1$.

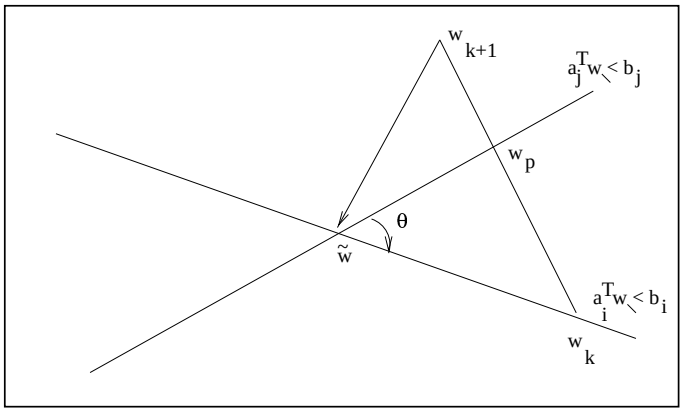


Figura 3.3: Método de relaxação usando λ conveniente.

Dado um ponto inicial $w_0 = (x_0^T, h(x_0))^T \in H_0(w)$, tal que $g(x_0) > h(x_0)$ é calculada uma aproximação da raiz, que é obtida segundo o procedimento do algoritmo da figura 3.5 (ver figura 3.4).

Na figura 3.6 é apresentado o algoritmo que explora as idéias discutidas anteriormente. O critério de parada é dado por $g(x_k) - h(x_k) < \epsilon$, onde $\epsilon > 0$ é a tolerância requerida.

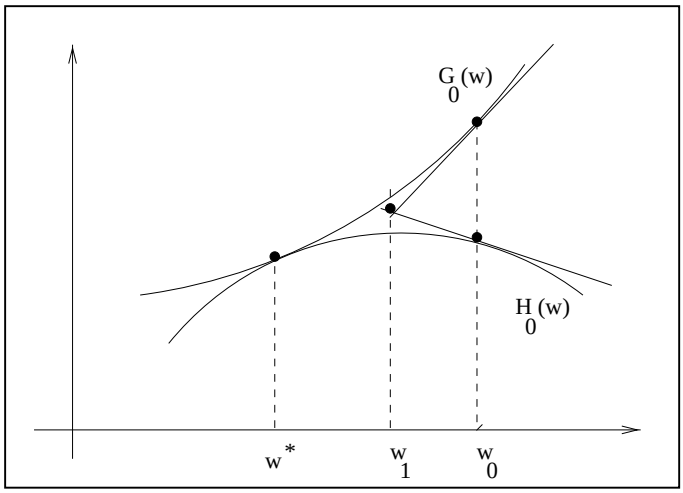


Figura 3.4: Aproximações lineares de $G(w)$ e $H(w)$ no ponto w_0.

```
Algoritmo intersecao
{ Objetivo: Aplicar o método de relaxação para encontrar o ponto }
            {pertencente à interseção de dois hiperplanos.}
parâmetros de entrada w_k, a_i, a_j, b_i, b_j
parâmetros de saída w̃ { ponto pertencente a interseção }
   cos(θ) ← (a_i)^T (a_j)/(||a_i||||a_j||)
   λ ← 1/(1 − cos^2(θ))
   w_p ← w_k + a_j(b_j − a_j^T w_k)/||a_j||^2
   w_{k+1} ← w_k + λ(w_p − w_k)
   w̃ ← w_{k+1} + a_i(b_i − a_i^T w_{k+1})/||a_i||^2
fim algoritmo
```

Figura 3.5: Algoritmo para encontrar ponto de interseção de dois hiperplanos.

```
Algoritmo das projeções
{ Objetivo: Aplicar o método das projeções para calcular uma }
                {aproximação da raiz de uma equação convexa em R^n.}
parâmetros de entrada x_0, itermax, toler
   { solução inicial, número máximo de iterações, tolerância máxima }
parâmetros de saída x̃, Erro { solução, condição de erro }
   Calcule g(x_0), h(x_0), ▽g(x_0), ▽h(x_0)
   se g(x_0) < h(x_0) então
      {Erro: g(x_0) tem que ser maior que h(x_0) }
      interrompa
   fim se
   k ← 0;
   enquanto g(x_k) − h(x_k) > toler e iter < itermax faça
      Compute a_i^k, a_j^k, b_i^k, b_j^k
      w_{k+1} ← w̃ ∈ G(w_k) ∩ H(w_k)(ver algoritmo da figura 3.5)
      x_{k+1}(i) ← w_{k+1}(i), i = 1, ..., n
      k ← k + 1
   fim enquanto
   x̃ ← x_k
   Erro ← g(x_k) − h(x_k) > toler
fim algoritmo
```

Figura 3.6: Algoritmo para encontrar uma raiz de uma equação convexa em R^n.

3.2 Exemplo do método das projeções

Para fixar as idéias apresentadas pelo método das projeções, mostra-se um exemplo numérico.

Observa-se que em vários problemas testes (ver apêndice A na página 50), relaxamos a necessidade de convexidade para a função como um todo. O importante é que a função $f(x)$ seja convexa nas proximidades da raiz. Particularmente, escolhemos um exemplo em que $f(x)$ não é convexa.

Seja a função $f(x)$, com $x \in \mathbb{R}^2$, dada por:

$f(x) = 5x_1^3 - 2x_2^2 + 8x_1 - 10$

Primeira iteração

Compute a_i^0, a_j^0, b_i^0, b_j^0 (ver algoritmo da figura 3.6)

Escolhendo $g(x)$ e $h(x)$ $\quad (f(x) = g(x) - h(x))$;

$g(x) = 5x_1^3 - 10 \quad$ e $\quad h(x) = 2x_2^2 - 8x_1$;

e dado um ponto inicial x_0 , tal que $g(x_0) > h(x_0)$, tem-se:

$$\nabla g(x) = \begin{pmatrix} 15x_1^2 \\ 0 \end{pmatrix} \quad \text{e} \quad \nabla h(x) = \begin{pmatrix} -8 \\ 4x_2 \end{pmatrix} .$$

O hiperplano de suporte associado à função $h(x)$ em x_0 é dado pela expansão de primeira ordem, como se segue. Sendo $h(x) = -8x_1 + 2x_2^2$ e $x_0 = (1,1)^T$, tem-se:
$h(x) \approx h(x_0) + \nabla h(x_0)^T (x - x_0) \implies$

$$h(x) \approx -6 + (-8\ ,\ 4) \,.\, \begin{pmatrix} x_1 - 1 \\ x_2 - 1 \end{pmatrix} \Longrightarrow$$

$h(x) \approx -6 - 8x_1 + 8 + 4x_2 - 4 \Longrightarrow$
$r = -8x_1 + 4x_2 - 2 \quad$ ou
$8x_1 - 4x_2 + r = -2$. Logo,

$$a_i^0 = \begin{pmatrix} 8 \\ -4 \\ 1 \end{pmatrix} \quad \text{e} \quad w = \begin{pmatrix} x_1 \\ x_2 \\ r \end{pmatrix} .$$

Obtém-se a equação $(a_i^0)^T\, w = b_i^0$, onde $b_i^0 = -2 \quad$ e $\quad w \in \mathbb{R}^{n+1}$:

$H(w) = \{w \in \mathbb{R}^3 : \ (a_i^0)^T \, w = -2\}$.

Repetindo o mesmo procedimento para $g(x)$, o hiperplano de suporte associado à função $g(x)$ em x_0 é dado pela expansão de primeira ordem:

$g(x) \approx g(x_0) + \nabla g(x_0)^T \, (x - x_0)$. No ponto inicial $x_0 = (1,1)^T$, tem-se:

$$g(x) \approx -5 + (15 \ , \ 0) \, . \begin{pmatrix} x_1 - 1 \\ x_2 - 1 \end{pmatrix} \implies$$

$g(x) \approx -5 + 15x_1 - 15 \implies$

$r = 15x_1 - 20$ ou

$15x_1 - r = 20$. Logo,

$$a_j^0 = \begin{pmatrix} 15 \\ 0 \\ -1 \end{pmatrix} \quad \text{e} \quad w = \begin{pmatrix} x_1 \\ x_2 \\ r \end{pmatrix} \ .$$

Obtém-se a equação $(a_j^0)^T \, w = b_j^0$, onde $b_j^0 = 20$ e $w \in \mathbb{R}^{n+1}$:

$G(w) = \{w \in \mathbb{R}^3 : \ (a_j^0)^T \, w = 20\}$

O próximo passo, " $w_{k+1} \leftarrow \tilde{w} \in G(w_k) \cap H(w_k)$ ", (ver algoritmo da figura 3.5):

$\cos(\theta) = \frac{(a_i)^T (a_j)}{||a_i|| ||a_j||}$, logo $\cos(\theta) = 0.8795$.

$\lambda = \frac{1}{1 - cos^2(\theta)}$, logo $\lambda = 4.4150$.

Dado um ponto inicial $w_0 = (x_0^T, h(x_0))^T \in H_0(w)$ (ver figura 3.4 página 13), encontra-se a projeção ortogonal de w_0 que é w_p (ver figura 3.7) :

$$w_0 = \begin{pmatrix} 1 \\ 1 \\ -6 \end{pmatrix} \ ;$$

$w_p = w_0 + a_j \frac{(b_j - a_j^T w_0)}{||a_j||^2}$;

como $||a_j^0||^2 = 226$ obtém-se $w_p = \begin{pmatrix} 0.9336 \\ 1.0000 \\ -5.9956 \end{pmatrix}$.

A seguir obtém-se w_1:

$w_1 = w_0 + \lambda(w_p - w_0)$, sendo λ o parâmetro de relaxação. Portanto,

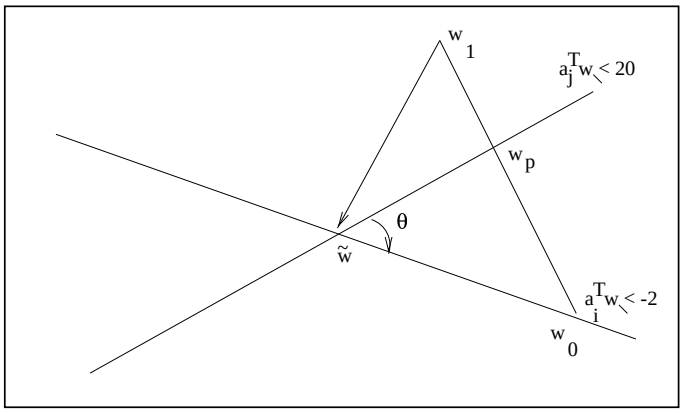


Figura 3.7: Método de relaxação.

$$w_1 = \begin{pmatrix} 0.7070 \\ 1.0000 \\ -5.9805 \end{pmatrix} .$$

Finalizando a primeira iteração computa-se $\tilde{w}$ projetando w_1 em $H(w)$, a fim de obter um $\tilde{w}$: $\tilde{w} \in \{w : \ (a_j^0)^T w = 20\} \cap \{w : \ (a_i^0)^T w = -2\}$. Assim,

$$\tilde{w} = w_1 + a_i(b_i - a_i^T w_1)/\|a_i\|^2 \implies$$

$$\tilde{w} = \begin{pmatrix} 0.9365 \\ 0.8851 \\ -5.9517 \end{pmatrix} .$$

Segunda iteração

Já que $\tilde{w} = \begin{pmatrix} \tilde{x_1} \\ \tilde{x_2} \\ \tilde{r} \end{pmatrix}$, $w_1 = \begin{pmatrix} \tilde{x_1} \\ \tilde{x_2} \\ h(\tilde{x_1}, \tilde{x_2}) \end{pmatrix}$, tem-se:

$$w_1 = \begin{pmatrix} 0.9365 \\ 0.8851 \\ -5.9252 \end{pmatrix} .$$

Computando $a_i^1,\ a_j^1,\ b_i^1,\ b_j^1$ (ver algoritmo da figura 3.6) e calculando a nova aproximação linear para $h(x)$, temos:

$$h(x) \approx -5.9252 + (-8\ ,\ 3.5404) \cdot \begin{pmatrix} x_1 - 0.9365 \\ x_2 - 0.8851 \end{pmatrix} \implies$$

$$h(x) \approx -5.9252 - 8(x_1 - 0.9365) + 3.5404(x_2 - 0.8851) \implies$$

$r = -5.9252 - 8x_1 + 7.4920 + 3.5404x_2 - 3.1336 \implies$
$r = -8x_1 + 3.5404x_2 - 1.5668 \implies$
$8x_1 - 3.5404x_2 + r = -1.5668$. Finalmente, temos:

$$a_i^1 = \begin{pmatrix} 8 \\ -3.5404 \\ 1 \end{pmatrix} \quad \text{e} \quad b_i^1 = -1.5668 \quad \text{sendo} \quad \|a_i^1\|_2 = 8.8054.$$

Fazendo o mesmo para a função $g(x)$.
Como $g(x) = 5x_1^3 - 10$, tem-se:

$$g(x) \approx -5.8933 + (13.1555\ ,\ 0)\ .\ \begin{pmatrix} x_1 - 0.9365 \\ x_2 - 0.8851 \end{pmatrix} \implies$$

$g(x) \approx -5.8933 + 13.1555x_1 - 12.3201 \implies$
$r = 13.5555x_1 - 18.2134 \implies$
$-13.1555x_1 + r = -18.2134$. Logo,

$$a_j^1 = \begin{pmatrix} -13.1555 \\ 0 \\ 1 \end{pmatrix} \quad \text{e} \quad b_j^1 = -18.2134 \quad \text{sendo} \quad \|a_j^1\|_2 = 13.1935.$$

Calculando $cos(\theta)$, λ, w_p, w_2 e $\tilde{w}$ (ver algoritmo da figura 3.5):

$cos(\theta) = -0.8973$ e $\lambda = 5.1328$.

$w_p = w_1 + a_j \frac{(b_j - a_j^T w_1)}{\|a_j\|^2}$, tem-se $w_p = \begin{pmatrix} 0.9341 \\ 0.8851 \\ -5.9250 \end{pmatrix}$.

$w_2 = w_1 + \lambda(w_p - w_1)$, logo $w_2 = \begin{pmatrix} 0.9241 \\ 0.8851 \\ -5.9243 \end{pmatrix}$.

$\tilde{w} = w_2 + a_i(b_i - a_i^T w_2)/\|a_i\|^2$,

$\tilde{w} = \begin{pmatrix} 0.9342 \\ 0.8806 \\ -5.9240 \end{pmatrix}$.

Lembre-se que o critério de parada é dado por $g(x_k) - h(x_k) < \epsilon$, onde $\epsilon > 0$ é a tolerância requerida.

O exemplo acima encontra-se na tabela 3.1 onde é resolvido o problema de encontrar o zero da função $f(x)$ em $\mathbb{R}^n$ através do algoritmo da figura 3.6 na seção 3.1. A raiz da equação é obtida com apenas quatro iterações.

$$\text{Raiz} = \begin{bmatrix} 0.9342 & 0.8806 \end{bmatrix}^T$$

k	$x_k(1)$	$x_k(2)$	$f(x_k) = G(x_k) - H(x_k)$	$\|\|x_{k+1} - x_k\|\|_2$
0	1.00000	1.00000	1.00000e+000	
1	0.93655	0.88516	1.00000e+000	1.31200e-001
2	0.93423	0.88057	3.27361e-002	5.14508e-003
3	0.93423	0.88056	3.30866e-005	5.23607e-006
4	0.93423	0.88056	3.36069e-011	5.31804e-012

Tabela 3.1: Desempenho do método para a função $f(x) = 5x_1^3 - 2x_2^2 + 8x_1 - 10$.

3.3 Alguns fundamentos teóricos

A seguir, serão apresentados alguns lemas envolvidos na fundamentação teórica do método. O lema 3.1 mostra como calcular o parâmetro λ que permite ao método de relaxação determinar a interseção de dois hiperplanos em uma única iteração. O lema 3.2 simplifica o procedimento sugerido no lema 3.1. Os lemas 3.3, 3.4 e 3.5 são utilizados para estabelecer a convergência do método.

Lema 3.1. *Considere dois hiperplanos $G(x)$ e $H(x)$. Seja θ o ângulo entre eles e, w_p a projeção ortogonal de $w_k \in H(w_k)$ em $G(w)$. Então para*

$$w_{k+1} := w_k + \lambda(w_p - w_k),$$

o valor de λ, tal que w_{k+1} projetado em $H(w_k)$, resulta $\tilde{w} \in H(w) \cap G(w)$ é $\csc^2(\theta)$.

Demonstração

Considere o triângulo retângulo $(w_k, w_{k+1}, \tilde{w})$ e (w_k, w_p, z) apresentado na figura 3.8 (e veja figura 3.3). Observe que:

$$\frac{a + \tilde{a}}{l} = \frac{a}{c}$$

ou seja,

$$\tilde{a} = (l - c)\frac{a}{c}. \tag{3.1}$$

Também,

$$\text{sen}(\theta) = \cos(90 - \theta) = \frac{c}{a}, \quad \text{e} \quad l = \frac{a}{\text{sen}(\theta)}$$

Com isto, tem-se:

$$l = \frac{a^2}{c}. \tag{3.2}$$

Substituindo (3.2) em (3.1), segue que:

$$\tilde{a} = \left(\frac{a^2}{c} - c\right)\frac{a}{c} = \frac{a}{c}\frac{a^2 - c^2}{c} = a\frac{h^2}{c^2}.$$

Mas,

$$\tilde{a} = a.\tan^2(\theta - 90) = a.\cot^2(\theta) \quad \text{e} \quad (a + \tilde{a}) = \lambda a\,.$$

Logo, tem-se:

$$\begin{aligned}\lambda &= \frac{(a + \tilde{a})}{a} \\ &= \frac{(a + a.\cot^2(\theta))}{a} \\ &= 1 + \cot^2(\theta) \\ &= \csc^2(\theta)\,.\end{aligned}$$

□

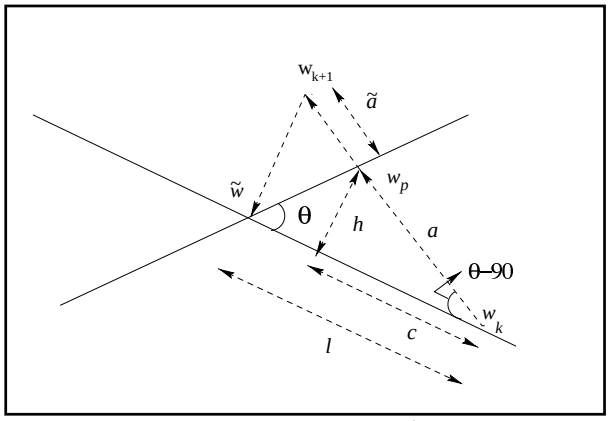


Figura 3.8: Interpretação geométrica da prova do lema 4.1

Lema 3.2. *Considere θ, w_k and w_{k+1} como no lema 3.1. Então w_{k+1} pode ser calculado por*

$$w_{k+1} := w_k + td,$$

em que,

$$\begin{aligned} \lambda &= \csc^2(\theta)\,, \\ t &= -\lambda(b_j - a_j^T w_0)\,, \\ d &= -\frac{1}{||a_j||^2}a_j + \frac{a_i^T a_j}{||a_i||^2||a_j||^2}a_i\,. \end{aligned}$$

Demonstração: Sejam,

$$w_p = w_k + a_j\frac{(b_j - a_j^T w_k)}{||a_j||^2} \tag{3.3}$$

$$w_{k+1} = w_k + \lambda(w_p - w_k) \tag{3.4}$$

$$\tilde{w} = w_{k+1} + a_i\frac{b_i - a_i^T w_{k+1}}{||a_i||^2}\,. \tag{3.5}$$

Multiplicando (3.4) por a_i^T,

$$\begin{aligned} a_i^T w_{k+1} &= a_i^T(w_k + \lambda(w_p - w_k)) \\ &= a_i^T w_k + \lambda a_i^T w_p - \lambda a_i^T w_k\,. \end{aligned} \tag{3.6}$$

Para obter $a_i^T w_p$, multiplica-se (3.3) por a_i^T:

$$\begin{aligned} a_i^T w_p &= a_i^T w_k + a_i^T a_j\frac{(b_j - a_j^T w_k)}{||a_j||^2} \\ &= a_i^T w_k + \frac{a_i^T a_j b_j}{||a_j||^2} - \frac{(a_i^T a_j)(a_j^T w_k)}{||a_j||^2}\,. \end{aligned} \tag{3.7}$$

Substituindo (3.7) em (3.6),

$$\begin{aligned} a_i^T w_{k+1} &= a_i^T w_k + \lambda\left(a_i^T w_k + \frac{a_i^T a_j b_j}{||a_j||^2} - \frac{(a_i^T a_j)(a_j^T w_k)}{||a_j||^2}\right) - \lambda a_i^T w_k \\ &= a_i^T w_k + \lambda\frac{a_i^T a_j b_j}{||a_j||^2} - \lambda\frac{(a_i^T a_j)(a_j^T w_k)}{||a_j||^2}\,. \end{aligned} \tag{3.8}$$

Substitutindo (3.3) na expressão (3.4), obtêm-se uma nova fórmula para w_{k+1}:

$$\begin{aligned} w_{k+1} &= w_k + \lambda\left(w_k + a_j\frac{(b_j - a_j^T w_k)}{||a_j||^2} - w_k\right) \\ &= w_k + \lambda\left(a_j\frac{(b_j - a_j^T w_k)}{||a_j||^2}\right) \\ &= w_k + \lambda\frac{b_j}{||a_j||^2}a_j - \lambda\frac{a_j^T w_k}{||a_j||^2}a_j\,. \end{aligned} \tag{3.9}$$

Substituindo a equação (3.9) na equação (3.5), tem-se

$$\begin{aligned} \tilde{w} &= w_k + \lambda\frac{b_j}{||a_j||^2}a_j - \lambda\frac{a_j^T w_k}{||a_j||^2}a_j + b_i a_i\frac{1}{||a_i||^2} - a_i\frac{1}{||a_i||^2}(a_i^T w_k) \\ &\quad -\frac{1}{||a_i||^2}a_i\lambda\frac{(a_i^T a_j)b_j}{||a_j||^2} + \frac{1}{||a_i||^2}a_i\lambda\frac{(a_i^T a_j)(a_j^T w_k)}{||a_j||^2}\,. \end{aligned}$$

Como $w_k \in H(w)$, então $b_i - a_i^T w_k = 0$. Daí,

$$\tilde{w} = w_k + \lambda\frac{b_j}{||a_j||^2}a_j - \lambda\frac{a_j^T w_k}{||a_j||^2}a_j - \frac{1}{||a_i||^2}a_i\lambda\frac{(a_i^T a_j)b_j}{||a_j||^2} + \frac{1}{||a_i||^2}a_i\lambda\frac{(a_i^T a_j)(a_j^T w_k)}{||a_j||^2}\,.$$

Agrupando os termos que contém λb_j e $-\lambda a_j^T w_k$ tem-se que:

$$\tilde{w} = w_k + \lambda b_j\left(\frac{1}{||a_j||^2}a_j - \frac{1}{||a_i||^2}a_i\frac{(a_i^T a_j)}{||a_j||^2}\right) - \lambda a_j^T w_k\left(\frac{1}{||a_j||^2}a_j - \frac{1}{||a_i||^2}a_i\frac{(a_i^T a_j)}{||a_j||^2}\right)$$

Obtendo o resultado desejado:

$$\tilde{w} = w_k + \lambda(b_j - a_j^T w_k)\left(\frac{1}{||a_j||^2}a_j - \frac{1}{||a_i||^2||a_j||^2}(a_i^T a_j)a_i\right),$$

ou seja

$$\tilde{w} = w_k + td$$

em que,

$$t = -\lambda(b_j - a_j^T w_k) \;\text{ e }\; d = -\frac{1}{||a_j||^2}a_j + \frac{1}{||a_i||^2||a_j||^2}(a_i^T a_j)a_i.$$

□

Nos lemas a seguir sendo dado $f(x) = g(x) - h(x)$, uma função convexa em $\mathbb{R}^n$, $f(x) \in C^1$, considere os cones:

$$\begin{aligned} K &:= \{d \in R^n : \nabla g(x)^T d \leq 0, \nabla h(x)^T d \geq 0\}\ e \\ D &:= \{d \in R^n : \nabla f(x)^T d \leq 0\}. \end{aligned}$$

A direção d obtida no lema 3.2, pode ser escrita como:

$$d(x) = -\nabla g(x)\frac{1}{||\nabla g(x)||^2} + \nabla h(x)\frac{\nabla h(x)^T \nabla g(x)}{||\nabla g(x)||^2||\nabla h(x)||^2}\ .$$

Lema 3.3. $K \subset D$

Demonstração: Por contradição, suponha que existe um vetor $d \in K$: $d \notin D$. Se $d \in K$ então $\nabla g(x)^T d \leq 0$ e $-\nabla h(x)^T d \leq 0$. Para $t > 0$, suficientemente pequeno: $f(x + td) = g(x + td) - h(x + td) < f(x) \Rightarrow d$ é uma direção de descida para $f(x)$. Mas isso é uma absurdo pois $d \notin D$. Logo, todo $d \in K$ pertence a $D \Rightarrow K \subset D$.

□

Lema 3.4. $d \in K$

Demonstração:

$$\begin{aligned} \nabla g(x)^T d &= \nabla g(x)^T \left(-\nabla g(x)\frac{1}{||\nabla g(x)||^2} + \nabla h(x)\frac{\nabla h(x)^T \nabla g(x)}{||\nabla g(x)||^2||\nabla h(x)||^2} \right) \Rightarrow \\ \nabla g(x)^T d &= \cos^2(\theta) - 1 \\ &= -\text{sen}^2(\theta) \Rightarrow \\ \nabla g(x)^T d &\leq 0; \end{aligned} \tag{3.10}$$

$$\begin{aligned} \nabla h(x)^T d &= \nabla h(x)^T \left(-\nabla g(x)\frac{1}{||\nabla g(x)||^2} + \nabla h(x)\frac{\nabla h(x)^T \nabla g(x)}{||\nabla g(x)||^2||\nabla h(x)||^2} \right) \Rightarrow \\ \nabla h(x)^T d &= 0 \end{aligned} \tag{3.11}$$

Das equações (3.10) e (3.11), tem-se que $d \in K$.

□

Lema 3.5. *d é uma direção de descida para $f(x)$.*

Demonstração: Imediato, pois $d \in K$ (lema 3.4).

□

Capítulo 4

Experiência numérica

Alguns cuidados devem ser tomados na escolha dos problemas para testar metodologias oriundas da Análise Numérica. Segundo vários autores, as seguintes condições devem ser satisfeitas:

- os problemas testes devem ser padronizados e universais; eles devem ser selecionados empiricamente, baseado em como o método é usado;
- modelagem de dificuldades padronizadas e universais para uma dada classe de problemas;
- a solução para os problemas teste deve ser conhecida;
- os problemas devem ser suficientemente breves; (não utilizar arquivos grandes ou regras complexas para computar as funções) e;
- não utilizar problemas com características específicas, favoráveis a alguns em detrimento de outros.

Neste trabalho, no qual é testado o método descrito no capítulo 3, foram utilizados problemas acadêmicos; optou-se por testes existentes na literatura ao lado de alguns propostos neste capítulo. Inicialmente, para facilitar a depuração da implementação e de se averiguar como o método trabalha, foram escolhidos problemas unidimensionais. Posteriormente, estes problemas combinados constituiram-se em uma fonte interessante de problemas multimensionais; vários problemas teste foram obtidos dessa forma.

A escolha dos testes dos problemas convexos não suaves foi mais complexa. Optou-se por construir os problemas a partir da fértil literatura de programação convexa não suave.

Problemas de otimização dessa área foram convertidos em problemas de achar zeros de funções da seguinte maneira. Conhecida a solução do problema

$$\begin{aligned} &\min\ f(x) \\ &x \in \mathbb{R}^n\ , \end{aligned} \tag{4.1}$$

onde $f(x)$ é um funcional convexo não suave, obteve-se o problema de encontrar $\bar{x} : F(\bar{x}) = 0$ fazendo

$$F(x) = f(x) - f(x^*)\,, \tag{4.2}$$

onde $x^* = \text{argmin}\, f(x)$.

Problemas que envolvem zeros de funções em que $x \in \mathbb{R}^n$ são usualmente resolvidos com o emprego de técnicas de otimização. Assim, naturalmente, é de se esperar que seja feita alguma comparação de resultados nesse contexto. O método foi reescrito para moldar-se ao paradigma da melhoria da solução que é muito utilizado nos métodos de otimização. Nestes, a nova solução x_{k+1} na iteração $k+1$ é obtida por

$$x_{k+1} = x_k + t_k\,.d_k\,, \tag{4.3}$$

onde $d_k \in \mathbb{R}^n$. No caso de problemas convexos diferenciáveis, d_k é uma direção de descida e $t_k \in \mathbb{R}$ é um passo dado nesta direção. Conforme pode ser visto na seção 3.3 do capítulo 3, o algoritmo da figura 3.6 do capítulo 3, correspondente à generalização do método de Newton para resolver problemas em $\mathbb{R}^n$, pode ser interpretado segundo 4.3, fazendo

$$\lambda = \text{csc}^2(\theta),$$

$$t_k = -\lambda(b_j^k - a_j^{k^T} w_k) \qquad e$$

$$d_k = -\frac{1}{||a_j||^2} a_j^k + \left(\frac{1}{||a_i||^2||a_j||^2} a_i^{k^T} a_j^k\right) a_i^k\ .$$

A figura 4.1 apresenta o método segundo este paradigma. A direção d_k é uma direção de descida (ver seção 3.3) e portanto, poderia ser feita a busca linear como mostrado no algoritmo da figura 4.2. Este algoritmo não foi explorado neste trabalho pois envolveria uma busca linear para o cálculo do passo t_k. Aparentemente, a complexidade da busca linear (em termos do número de operações aritméticas) é semelhante à complexidade do algoritmo das projeções (figura 4.1).

Algoritmo das projeções segundo o paradigma 4.3

{ **Objetivo:** Aplicar o método das projeções (ver capítulo 3) para calcular }
{uma aproximação da raiz de uma equação convexa em R^n.}

parâmetros de entrada x_0, itermax, toler
{ solução inicial, número máximo de iterações, tolerância máxima }

parâmetros de saída $\tilde{x}$, Erro { solução, condição de erro }

Calcule $g(x_0)$, $h(x_0)$, $\bigtriangledown g(x_0)$, $\bigtriangledown h(x_0)$

se $g(x_0) < h(x_0)$ **então**

{Erro: $g(x_0)$ tem que ser maior que $h(x_0)$ }

interrompa

fim se

$k \leftarrow 0$;

enquanto $g(x_k) - h(x_k) >$ toler **e** iter $<$ itermax **faça**

Calcule a_i^k, a_j^k, b_i^k, b_j^k

$w_k \leftarrow (x_k^T, h(x_k))^T$

$\cos(\theta) \leftarrow (a_i^{kT} a_j^k)/(||a_i^k|| ||a_j^k||)$

$\lambda \leftarrow 1/(1 - \cos^2(\theta))$

$t_k \leftarrow -\lambda(b_j^k - a_j^{kT} w_k)$

$d_k \leftarrow -\frac{1}{||a_j||^2} a_j^k + \left(\frac{1}{||a_i||^2 ||a_j||^2} a_i^{kT} a_j^k\right) a_i^k$

$w_{k+1} \leftarrow w_k + t_k d_k$

$x_{k+1(i)} \leftarrow w_{k+1(i)}$, $i = 1, \ldots, n$

$k \leftarrow k + 1$

fim enquanto

$\tilde{x} \leftarrow x_k$

Erro $\leftarrow g(x_k) - h(x_k) >$ toler

fim algoritmo

Figura 4.1: Algoritmo para encontrar uma raiz de uma equação convexa em R^n segundo o paradigma $x_{k+1} = x_k + t_k \, .d_k$.

Algoritmo para encontrar $\tilde{x}: f(\tilde{x}) = 0$
{ **Objetivo:** Encontrar uma raiz de uma equação convexa em R^n }
parâmetros de entrada x_0, itermax, toler
{ solução inicial, número máximo de iterações, tolerância máxima }
parâmetros de saída $\tilde{x}$, Erro { solução, condição de erro }
Calcule $g(x_0)$, $h(x_0)$, $\bigtriangledown g(x_0)$, $\bigtriangledown h(x_0)$
se $g(x_0) < h(x_0)$ **então**
{Erro:método não converge com este valor inicial}
interrompa
fim se
$k \leftarrow 0$;
enquanto $g(x_k) - h(x_k) >$ toler **e** iter $<$ itermax **faça**
Calcule a_i^k, a_j^k, b_i^k, b_j^k
$w_k \leftarrow (x_k^T, h(x_k))^T$
$t_k \leftarrow \text{Min}_{t>0} f(x_k + td_k)$
$d_k \leftarrow -\frac{1}{||a_j||^2} a_j^k + \left(\frac{1}{||a_i||^2||a_j||^2} a_i^{kT} a_j^k\right) a_i^k$
$w_{k+1} \leftarrow w_k + t_k d_k$
$x_{k+1(i)} \leftarrow w_{k+1(i)}$, $i = 1, \ldots, n$
$k \leftarrow k + 1$
fim enquanto
$\tilde{x} \leftarrow x_k$
Erro $\leftarrow g(x_k) - h(x_k) >$ toler
fim algoritmo

Figura 4.2: Algoritmo de descida para encontrar uma raiz de uma equação convexa em R^n.

Para comparar o desempenho do método apresentado no capítulo 3, optou-se pelo método de Newton-Raphson (ver capítulo 2) que converge mesmo considerando um passo unitário em cada iteração;

$$x_{k+1} = x_k + d_k \ ;$$

o que de certa forma, mimetiza os procedimentos para refinar intervalos em problemas unidimensionais. A possibilidade de adotar o passo unitário e a eficiência foram os fatores que mais pesaram na opção pelo Newton-Raphson.

Finalmente, o ambiente escolhido para exercícios numéricos foi o Matlab 5.2 versão para estudantes, em um microcomputador sem nenhum recurso especial. A seguir tem-se a experiência numérica com os problemas descritos no apêndice A.

4.1 Problemas diferenciáveis

Para depurar o código, foram selecionados 25 problemas unidimensionais (testes 1-25 do apêndice A na página 50). Para ilustração, esses problemas foram resolvidos pelos métodos pégaso, de Muller, de Brent e de Newton (detalhes de implementação encontram-se em [Cam01]). A tabela 4.1 apresenta o desempenho de cada um destes métodos em relação ao método da bisseção e a tabela 4.2 mostra o número de iterações gastas para a convergência. O desempenho do método, apresentado no capítulo 3, nesta mesma série de problemas encontra-se na tabela 4.3. Percebe-se que o método das projeções, no caso particular em que $x \in \mathbb{R}$, funciona exatamente como o método de Newton (ver tabelas 4.2 e 4.3).

A maioria dos problemas multidimensionais (teste 26 em diante na seção A.2 página 55) foi obtida da combinação de alguns desses 25 problemas. A linha "funções 2+11" no teste 30 do apêndice A (página 55) indica que ele foi inspirado nos problemas 2 e 11; a variável x no problema 2 foi renomeada para x_1 e, de forma semelhante, a variável x do problema 11 passou a ser a variável x_2 e uma nova função foi composta combinando estas duas. Os valores iniciais para x_1 e x_2 são os mesmos considerados nos problemas originais 2 e 11. Nem todos os problemas listados são separáveis (ver problema 33 na página 56 e problemas da seção A.3 página 57).

A tabela 4.4 apresenta o desempenho do método das projeções em comparação com o método de Newton-Raphson em problemas nos quais $x \in \mathbb{R}^n$. Nesta tabela, n é a dimensão do problema, *Niter* é o número de iterações e $f(\bar{x})$ corresponde ao valor da função na raiz encontrada. Nas colunas referentes ao método de Newton-Raphson, o número de iterações

aparece com a composição de dois números. O primeiro valor corresponde ao número de iterações para encontrar $x_{k+1} : f(x_{k+1}) \leq 0$ e o segundo valor é o número de iterações gastos para achar o valor do passo $\bar{t}$ que proporciona $f(x_k + \bar{t}.\ d_k) = 0$ (ver detalhes no capítulo 2).

Verifica-se que o método de Newton-Raphson gastou mais tempo que o método das projeções para encontrar o valor do zero da função em $\mathbb{R}^n$ (ver tabela 4.4).

O gráfico referente à tabela 4.4 (ver figura 4.3) mostra claramente a efetividade do nosso método das projeções nos problemas testados; o tempo de execução dos testes da seção A.2 (página 55) realizados com o método das projeções é menor quando comparado ao método de Newton-Raphson.

A tabela 4.5 apresenta outros exemplos numéricos de equações diferenciáveis bem acopladas (não separáveis) que foram testados com o nosso método, o método das projeções.

Cumpre observar que nem todos os problemas testados são convexos; no caso, o método pode ser aplicado em uma vizinhança suficientemente próxima da raiz.

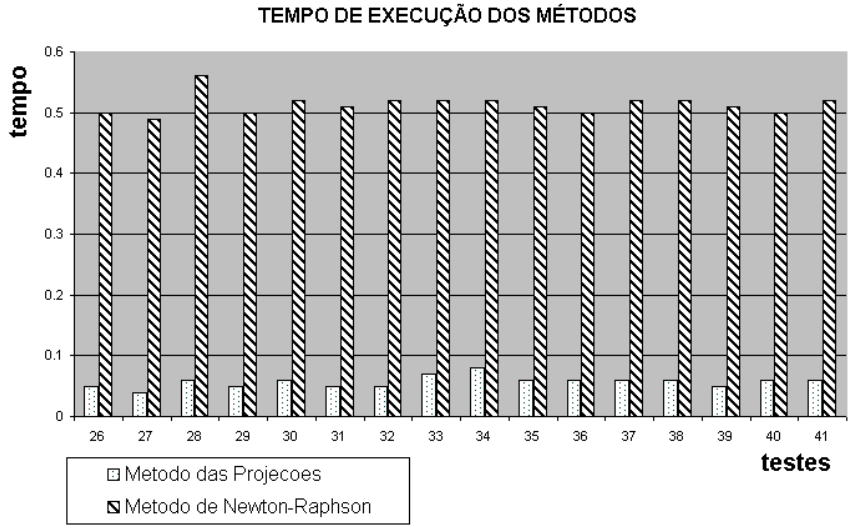


Figura 4.3: Comparação dos métodos em $\mathbb{R}^n$.

Tempo em relação ao método da bisseção (TRbis)				
Métodos				
Teste	Pégaso	Muller	WdBrent	Newton (Projeções)
1	0.31	0.28	0.55	0.31
2	0.30	0.30	0.52	0.33
3	0.29	0.25	0.56	0.31
4	0.22	0.29	0.46	0.27
5	0.26	0.27	0.55	0.29
6	0.26	0.32	0.47	0.30
7	0.25	0.31	0.53	0.25
8	0.26	0.35	0.37	0.49
9	0.30	0.30	0.48	0.30
10	0.28	0.30	0.52	0.35
11	0.29	0.32	0.53	0.35
12	0.26	0.27	0.35	0.38
13	0.29	0.29	0.41	0.41
14	0.26	0.22	0.45	0.38
15	0.28	0.26	0.46	0.30
16	0.28	0.10	0.43	0.31
17	0.34	0.25	0.52	0.20
18	0.28	0.27	0.53	0.35
19	0.31	0.31	0.53	0.39
20	0.28	0.29	0.53	0.21
21	0.28	0.28	0.57	0.29
22	0.29	0.28	0.63	0.31
23	0.24	0.15	0.37	0.24
24	0.20	0.26	0.39	0.20
25	0.07	0.32	0.06	0.35

Tabela 4.1: Resultados numéricos para metódos de raízes de equações (TRbis)

Número de iterações dos métodos				
Teste	Pégaso	Muller	WdBrent	Newton (Projeções)
1	9	7	8	6
2	10	7	9	7
3	8	4	8	6
4	6	5	6	5
5	8	6	9	6
6	8	7	8	6
7	7	5	7	5
8	8	8	6	11
9	9	6	7	6
10	8	6	8	7
11	9	7	9	7
12	8	6	7	6
13	8	6	7	7
14	7	4	8	6
15	8	5	7	6
16	9	1	8	7
17	11	5	9	4
18	8	5	8	7
19	9	6	8	8
20	8	5	8	4
21	8	5	8	5
22	8	5	9	6
23	6	2	6	5
24	5	4	5	4
25	11	8	10	8

Tabela 4.2: Resultados numéricos para metódos de raízes de equações (Iterações)

Método da Projeção							
Teste	Niter	TRbis	$\tilde{x}$	Teste	Niter	TRbis	$\tilde{x}$
1	6	0.31	1.07912	14	6	0.38	0.59934
2	7	0.33	1.05413	15	6	0.30	3.15553
3	6	0.31	1.40962	16	7	0.31	-4.00000
4	5	0.27	1.10996	17	4	0.20	1.49288
5	6	0.29	1.72313	18	7	0.35	0.94537
6	6	0.30	2.89280	19	8	0.3	0.95361
7	5	0.25	4.32324	20	4	0.22	2.93396
8	11	0.49	-0.92956	21	5	0.30	1.00000
9	6	0.30	1.28226	22	6	0.29	1.24957
10	7	0.35	1.87978	23	5	0.25	2.30940
11	7	0.35	-1.31330	24	4	0.22	9.07028
12	6	0.38	0.63288	25	8	0.38	-3.00000
13	7	0.41	1.21669				

Tabela 4.3: Resultados para $n = 1$

Métodos							
Projeção					Newton-Raphson		
Test	n	Niter	Tempo	$f(\tilde{x})$	Niter	Tempo	$f(\tilde{x})$
26	2	6	0.05	1.73433e-12	9+10	0.50	1.77636e-15
27	2	7	0.04	1.77636e-15	21+10	0.49	-1.77636e-15
28	2	9	0.06	2.49800e-13	32+7	0.56	-3.42837e-12
29	2	7	0.05	1.07470e-13	9+9	0.50	-2.05236e-10
30	2	7	0.06	5.68434e-14	22+10	0.52	5.32907e-15
31	3	7	0.05	3.55271e-15	10+10	0.51	8.88178e-15
32	3	7	0.05	1.93623e-13	22+10	0.52	-7.10543e-15
33	3	7	0.07	4.44089e-16	22+9	0.52	-8.37108e-14
34	4	8	0.08	1.77636e-15	10+14	0.52	3.55271e-15
35	4	7	0.06	2.50289e-12	11+10	0.51	-1.77636e-15
36	5	8	0.06	0.00000e+00	10+13	0.50	-1.42109e-14
37	6	7	0.06	1.42109e-13	10+11	0.52	7.10543e-15
38	7	7	0.06	1.63425e-13	10+10	0.52	-2.03784e-11
39	8	7	0.05	0.00000e+00	10+10	0.51	-3.50298e-12
40	9	7	0.06	8.98837e-13	10+11	0.50	2.84217e-14
41	10	7	0.06	5.18696e-13	10+11	0.52	-7.10543e-14

Tabela 4.4: Resultados Numéricos para $x \in R^n$

Método da Projeção			
Teste	n	Niter	$f(\tilde{x})$
42	3	8	9.37510e-10
43	3	100	5.36821e+02
44	2	19	2.55795e-12
45	3	18	6.25278e-13
46	4	10	8.75389e-12
47	5	7	1.50067e-11

Tabela 4.5: Resultados para $x \in R^n$ com funções não separáveis

As tabelas 4.6 e 4.7 exemplificam o comportamento do método em problemas separáveis. Nos casos relatados, as seqüências $\{(G(x_k) - H(x_k))\}$ e $\{||x_{k+1} - x_k||_2\}$ são estritamente decrescentes e seus valores sugerem convergência quadrática (ver figura 4.4).

A tabela 4.6 mostra as iterações do método das projeções para resolver o problema 29 (página 55), onde

$$f(x)^* = x_1^2 - \sin(x_1) - 1 + 3x_2^{x_2} - \log_{10}(2 + \sqrt{x_2}) + 3\tan(x_2) - 4$$

$$\text{Raiz} = \begin{bmatrix} -0.1549 & 0.7591 \end{bmatrix}^T$$

k	$x_k(1)$	$x_k(2)$	$f(x_k) = G(x_k) - H(x_k)$	$\|\|x_{k+1} - x_k\|\|_2$
0	2.00000	1.00000	5.28580e+00	
1	0.77301	1.01005	5.28580e+00	1.22703e+00
2	-0.05874	0.89689	2.22907e+00	8.39413e-01
3	-0.16094	0.77906	1.07026e+00	1.55978e-01
4	-0.15504	0.75949	1.58314e-01	2.04454e-02
5	-0.15493	0.75909	3.09995e-03	4.15090e-04
6	-0.15493	0.75909	1.24831e-06	1.67279e-07
7	-0.15493	0.75909	1.07470e-13	1.43604e-14

Tabela 4.6: Método das projeções aplicado ao Teste 29.

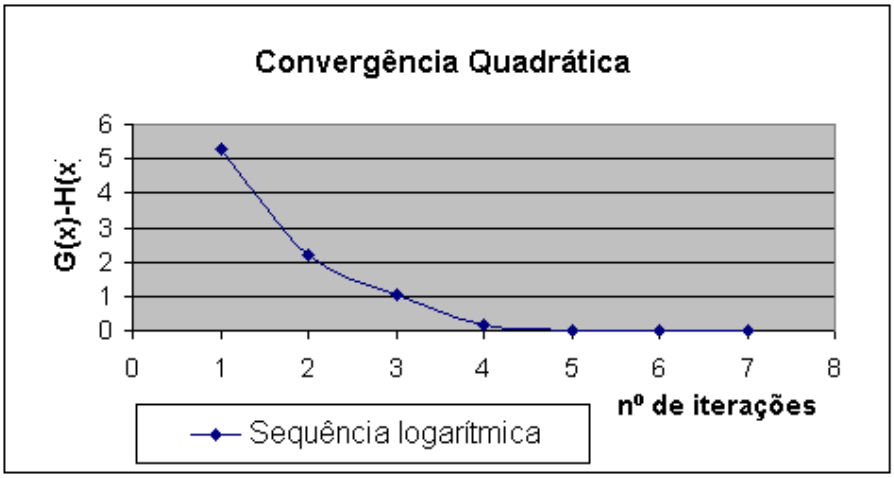


Figura 4.4: Valores de $f(x_k)$ sugerem convergência quadrática.

Outro exemplo encontra-se na tabela 4.7 onde é resolvido o problema de encontrar a raiz da função

$$\begin{aligned} f(x)^* &= e_1^x - \sin(x_1) - 2 + x_2^2 - \sin(x_2) - 1 + \\ & e^{-cos(x_3)} + 2x_3^4 - x_3^2 - 5 + e^{-x_4} + x_4^2 - 10 + \\ & e^{2x_5} - 2x_5^3 - 5 \end{aligned}$$

referente ao teste 36 (página 56), onde $x_0 = \begin{bmatrix} 2.0 & 2.0 & -2.0 & 6.0 & 2.0 \end{bmatrix}^T$ e a Raiz $= \begin{bmatrix} 1.3784 & 1.5724 & -0.7371 & 3.7948 & 0.7421 \end{bmatrix}^T$.

k	$f(x_k) = G(x_k) - H(x_k)$	$\|\|x_{k+1} - x_k\|\|_2$
0	9.06872e+01	
1	9.06872e+01	1.27730e+00
2	3.49839e+01	1.06034e+00
3	1.93650e+00	1.67275e-01
5	1.08662e-01	1.08271e-02
6	4.02748e-04	4.05172e-05
7	5.59402e-09	5.62790e-10
8	0.00000e+00	9.99201e-16

Tabela 4.7: Método das projeções aplicado ao Teste 36 onde $x \in R^5$.

O exemplo 43 (ver seção A.3 na página 57) corresponde à função quadrática $Q(x) = \frac{1}{2}(Ax-b)^T(Ax-b)$, onde $A \in \mathbb{R}^{n\times n}$, $b, x \in \mathbb{R}^n$. Caso A seja de posto completo, $Q(x) = 0$ corresponde à solução do sistema $Ax = b$. A função $Q(x)$ não é separável e suas raízes têm grau de multiplicidade 2. Neste caso, o método apresenta fraco desempenho, como pode ser visto na tabela 4.8. É bom lembrar que o método de Newton (para zeros de funções; ver 2.1) aplicado a problemas unidimensionais nessa circunstância também tem um fraco desempenho.

k	$x_k(1)$	$x_k(2)$	$x_k(3)$	$f(x_k) = G(x_k) - H(x_k)$	$\|\|x_{k+1} - x_k\|\|_2$
0	0.00000	0.00000	1.00000	4.27460e+004	
1	0.00530	0.18121	0.51981	4.27460e+004	5.13268e-001
10	-0.01884	0.25828	0.32118	4.77809e+002	3.45796e-001
20	0.13754	0.36326	0.01041	6.96481e+002	1.13912e-001
30	0.08878	0.35871	0.03279	7.73770e+002	1.05745e-001
40	0.08098	0.34318	0.07550	7.05865e+002	1.08321e-001
50	-0.06932	1.98165	-4.22415	3.43195e+006	4.57376e+000
60	0.23400	0.34291	0.04382	1.15712e+003	1.19136e-001
70	0.04910	0.33364	0.10762	6.01974e+002	1.22483e-001
80	-0.38280	0.13582	0.72237	4.52330e+002	8.67293e-001
90	0.08047	0.37643	-0.01218	1.89291e+003	1.22885e-001
100	0.08078	0.32061	0.13534	5.36821e+002	1.39693e-001

Tabela 4.8: Desempenho do método para a quadrática $Q(x)$

4.2 Problemas não necessariamente diferenciáveis

Seja f um funcional convexo não suave em $\mathbb{R}^n$. Como é usual, supõe-se que para um dado $\overline{x}$ existe um oráculo que retorna $f(\overline{x})$ e algum subgradiente s de f em $\overline{x}$. O subdiferencial de f em $\overline{x}$, definido como [Roc70]

$$\partial f(\overline{x}) := \{s \in \mathbb{R}^n : \ f(x) \geq f(\overline{x}) + (x - \overline{x})^T s\} \ ,$$

fornece substitutos para o gradiente em várias aplicações; por exemplo em alguns problemas de otimização, a função objetivo f é substituída por uma função modelo seccionalmente afim dada por

$$f(x) \approx f_k^{\Delta}(x) = \ \max\{f(x_i) + (x - x_i)^T s_i, \ i = 1, 2, \ldots k\} \ ;$$

onde $s_i \in \partial f(x_i)$.

Nesta seção aplicamos o algoritmo descrito no capítulo 3 (ver figura 3.6 página 14) substituindo o conceito de gradiente pelo conceito de subgradiente.

Em resumo, dado um ponto x_k, computa-se $g(x_k)$, $h(x_k)$, $s_k \in \partial g(x_k)$ e $v_k \in \partial h(x_k)$ com os quais é possível construir as seguintes aproximações lineares de $g(x)$ e $h(x)$:

$$g(x_k) + (x - x_k)^T s_k \qquad e$$

$$h(x_k) + (x - x_k)^T v_k \ .$$

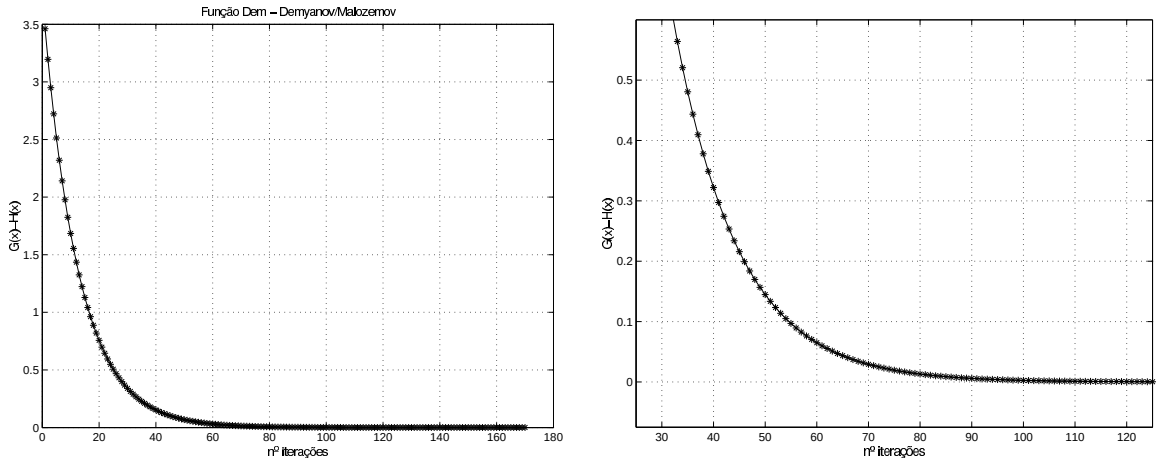
A próxima iteração, como era anteriormente feito, é dada por

$$x_{k+1} := x \ : g(x_k) + (x - x_k)^T s_k = h(x_k) + (x - x_k)^T v_k \ .$$

Por exemplo, seja o problema obtido adaptando-se o problema de Demyanov e Malozemov [DM86];

determinar $\overline{x}$: $\max\{5x_1 + x_2, -5x_1 + x_2, \ x_1^2 + x_2^2 + 4x_2\} + 3 \ = 0.$
Fazendo-se
$g(x) = Max \ \left\{\frac{1}{2}(5x_1 + x_2), \frac{1}{2}(-5x_1 + x_2), \frac{1}{2}(x_1^2 + x_2^2 + 4x_2)\right\} + 3;$
$h(x) = Min \ \left\{-\frac{1}{2}(5x_1 + x_2), -\frac{1}{2}(-5x_1 + x_2), -\frac{1}{2}(x_1^2 + x_2^2 + 4x_2)\right\};$
e tomando-se como $x_0 = (1, 1)^T$ chega-se à solução $x^* = (0, -3)$.

Neste caso, obteve-se o desempenho relatado por meio do gráfico da figura 4.5.



(a) (b)

Figura 4.5: (a)Valores de $f(x_k)$ produziram uma seqüência monótona decrescente, (b)Gráfico ampliado.

Os problemas escolhidos, conforme foi adiantado, são oriundos da programação convexa não suave. A tabela 4.9 foi retirada do artigo do Oliveira/Santos [OdS00]. Nesta tabela está o desempenho de 5 métodos na busca do ponto de mínimo de $f(x)$. A coluna *it* é o número de iterações, *nf* é o número de chamadas ao oráculo que retorna um subgradiente e uma avaliação da função $f(x)$. Os métodos GHV e CA-1 são baseados em métodos de pontos interiores que utilizam, pelo menos, um passo de Newton-Raphson por iteração.

Critérios para classificar o grau de dificuldade encontrado na solução de problemas de otimização não suave são sugeridos em [Gof77]. Seja f^* o valor ótimo de uma função convexa f e M o seu conjunto de argumentos ótimos:

$$M = \{x \in \mathbb{R}^n \ : \ f(x) = f^*\}.$$

A distância entre $x \notin M$ e o conjunto M é dada por $d(x) = ||P(x) - x||_2$, onde $P(x)$ é a projeção de x no conjunto M. O índice de condicionamento χ de uma função convexa f é, por definição, o cosseno do maior ângulo existente entre $-s \in \partial f(x)$ e $P(x) - x$, em um ponto $x \notin M$. Mais precisamente,

$$\chi := \inf_{\{x \notin M\}} \inf_{\{s \in \partial f(x)\}} \left\{ \frac{-s^T(P(x) - x)}{||s||_2 \, ||P(x) - x||_2} \right\}.$$

No nosso caso, a coluna $\bar{\chi}_K$ é o cosseno do ângulo existente entre $d_k^* = x^* - x_k$ e $-s_k \in \partial f(x_k)$. Já o cosseno do ângulo existente entre d_k proposta por nós e $-s_k$ encontra-se na coluna $cos(\omega_k)$.

Todos os problemas apresentados são mal condicionados. Dos treze testes executados (ver seção A.4 página 59), o método encontrou a solução para três problemas (CB3, LQ e Maxl) relatados nas tabelas 4.10, 4.12 e 4.15. O fracasso na maioria dos problemas pode ser explicado, em parte, pelo fato da direção d_k proposta fazer um ângulo favorável com menos o subgradiente (ver tabelas 4.11, 4.13, 4.14); qualquer método que utilize direções semelhantes a $-s_k$ terá dificuldades em alcançar a solução nesses problemas.

Fora isso, não foi possível estabelecer nenhum padrão que fornecesse a criação de um algoritmo mais eficiente.

Método		M1FC1		BT		PB		GHV		CA-1
Testes	$f*$	it	nf	it	nf	it	nf	it	nf	it
CB2	1.952225	11	31	13	16	15	16			10
CB3	2.000000	12	44	13	21	15	16			11
DEM	-3.000000	10	33	9	13	7	8			11
QL	7.200000	12	30	12	17	17	18			10
LQ	-1.414214	16	52	10	11	14	15			8
Mifflin1	-1.000000	143	281	49	74	22	23			10
Mifflin2	-1.000000	30	71	6	13	16	17			14
Rosen	-44.000000	22	61	22	32	40	41			19
Shor	22.600160	21	69	29	30	26	27	29	29	39
Maxquad1	-0.841408	29	69	45	56	41	42	87	87	79
Maxquad2	-0.841408	20	54	45	49	33	34			106
Maxq	0.000000	144	207	125	128	158	159			170
Maxl	0.000000	72	194	50	53	51	52			1

Tabela 4.9: Desempenho dos métodos de Otimização convexa não suave nos problemas adaptados para produzir os nossos testes.

Problema não diferenciável CB3 (ver teste 49 página 59)				
Iter	$G(x_k) - H(x_k)$	$\|\|x_{k+1} - x_k\|\|_2$	$cos(\omega_k)$	$\bar{\chi}_k$
1	6.10166e+000	5.58156e-001	6.18984e-002	5.40758e-001
2	2.43437e+000	4.80481e-001	1.55574e-001	2.89996e-001
3	2.82450e+000	3.88186e-001	3.03843e-001	3.04633e-001
4	1.10357e+000	3.28253e-001	2.26397e-001	2.44085e-001
5	1.13132e+000	2.51434e-001	4.14653e-001	2.32598e-001
6	3.94597e-001	1.79254e-001	3.02088e-001	1.81773e-001
7	3.52692e-001	1.16522e-001	5.08523e-001	1.30630e-001
8	1.00885e-001	6.91333e-002	3.64986e-001	1.03845e-001
9	7.26383e-002	3.39556e-002	5.58419e-001	4.89910e-002
10	1.69153e-002	1.57598e-002	3.98065e-001	3.31912e-002
11	9.59675e-003	5.93029e-003	5.74109e-001	1.10347e-002
12	1.99034e-003	2.13706e-003	4.06846e-001	5.31204e-003
13	1.01619e-003	7.02994e-004	5.76967e-001	1.45952e-003
14	2.04231e-004	2.27128e-004	4.08099e-001	5.88754e-004
15	1.02354e-004	7.21992e-005	5.77311e-001	1.52792e-004
16	2.04812e-005	2.28860e-005	4.08233e-001	5.96638e-005
17	1.02430e-005	7.24112e-006	5.77346e-001	1.53570e-005
18	2.04871e-006	2.29040e-006	4.08247e-001	5.97471e-006
19	1.02438e-006	7.24327e-007	5.77350e-001	1.53661e-006
20	2.04877e-007	2.29058e-007	4.08248e-001	5.96930e-007

Tabela 4.10: Resultados numéricos para o problema não diferenciável CB3.

Problema não diferenciável DEM (ver teste 50 página 59)				
Iter	$G(x_k) - H(x_k)$	$\|\|x_{k+1} - x_k\|\|_2$	$cos(\omega_k)$	$\bar{\chi}_k$
1	7.30769e+00	1.76505e+00	3.65148e-01	4.67462e-17
10	3.55567e+00	7.55434e-01	3.65148e-01	0.00000e+00
20	1.59698e+00	3.39294e-01	3.65148e-01	5.34770e-17
30	7.17264e-01	1.52389e-01	3.65148e-01	5.95330e-17
40	3.22150e-01	6.84437e-02	3.65148e-01	-2.65099e-16
50	1.44690e-01	3.07406e-02	3.65148e-01	-1.47560e-16
60	6.49854e-02	1.38067e-02	3.65148e-01	8.21355e-17
70	2.91874e-02	6.20112e-03	3.65148e-01	1.37156e-16
80	1.31091e-02	2.78515e-03	3.65148e-01	-2.85017e-15
90	5.88780e-03	1.25092e-03	3.65148e-01	-1.05387e-14
100	2.64443e-03	5.61833e-04	3.65148e-01	2.14774e-14
110	1.18771e-03	2.52340e-04	3.65148e-01	2.55598e-14
120	5.33445e-04	1.13335e-04	3.65148e-01	1.30898e-13
130	2.39590e-04	5.09031e-05	3.65148e-01	-2.02113e-13
140	1.07609e-04	2.28625e-05	3.65148e-01	-7.55220e-13
150	4.83311e-05	1.02684e-05	3.65148e-01	8.54712e-14
160	2.17073e-05	4.61191e-06	3.65148e-01	-2.23060e-12
170	9.74954e-06	2.07138e-06	3.65148e-01	1.32266e-11

Tabela 4.11: Resultados numéricos para o problema não diferenciável DEM.

Problema não diferenciável LQ (ver teste 52 página 61)				
Iter	$G(x_k) - H(x_k)$	$\|\|x_{k+1} - x_k\|\|_2$	$cos(\omega_k)$	$\bar{\chi}_k$
0	2.41421e+000			
1	2.22045e-016	1.70711e+000	8.16497e-001	-1.00000e+000

Tabela 4.12: Resultados numéricos para o problema não diferenciável LQ.

Problema não diferenciável Mifflin2 (ver teste 54 página 61)				
Iter	$G(x_k) - H(x_k)$	$\|\|x_{k+1} - x_k\|\|_2$	$cos(\omega_k)$	$\bar{\chi}_k$
1	1.57600e+000	5.07243e-001	1.73749e-001	9.44836e-001
10	2.33307e-001	3.71413e-002	9.67950e-001	9.58310e-002
20	1.32978e-001	2.08709e-002	9.68929e-001	7.08367e-002
30	9.27222e-002	1.44668e-002	9.69306e-001	5.86561e-002
40	7.11066e-002	1.10584e-002	9.69505e-001	5.11361e-002
50	5.76383e-002	8.94570e-003	9.69628e-001	4.59112e-002
60	4.84480e-002	7.50889e-003	9.69711e-001	4.20123e-002
70	4.17794e-002	6.46879e-003	9.69771e-001	3.89604e-002
80	3.67211e-002	5.68123e-003	9.69817e-001	3.64878e-002
90	3.27533e-002	5.06430e-003	9.69852e-001	3.44320e-002
100	2.95580e-002	4.56802e-003	9.69881e-001	3.26879e-002
110	2.69298e-002	4.16018e-003	9.69904e-001	3.11840e-002
120	2.47302e-002	3.81910e-003	9.69924e-001	2.98699e-002
130	2.28623e-002	3.52964e-003	9.69940e-001	2.87087e-002
140	2.12565e-002	3.28091e-003	9.69955e-001	2.76730e-002
150	1.98612e-002	3.06489e-003	9.69967e-001	2.67417e-002
160	1.86376e-002	2.87553e-003	9.69978e-001	2.58983e-002
170	1.75558e-002	2.70818e-003	9.69988e-001	2.51299e-002

Tabela 4.13: Resultados numéricos para o problema não diferenciável Mifflin2.

Problema não diferenciável Maxq (ver teste 58 página 64)			
Iter	$G(x_k) - H(x_k)$	$\|\|x_{k+1} - x_k\|\|_2$	$cos(\omega_k)$
1	3.61000e+002	1.00000e+001	4.99376e-002
10	1.00000e+002	5.50000e+000	9.05357e-002
20	4.90000e+001	3.50000e+000	1.41421e-001
30	2.02500e+001	2.25000e+000	2.16930e-001
40	9.00000e+000	1.50000e+000	3.16228e-001
50	4.00000e+000	1.06250e+000	4.25797e-001
60	2.25000e+000	7.50000e-001	5.54700e-001
70	1.00000e+000	5.00000e-001	7.07107e-001
80	5.62500e-001	3.75000e-001	8.00000e-001
90	2.50000e-001	2.50000e-001	8.94427e-001
100	1.40625e-001	1.87500e-001	9.36329e-001
110	6.25000e-002	1.25000e-001	9.70143e-001
120	3.51562e-002	9.37500e-002	9.82872e-001
130	1.56250e-002	6.25000e-002	9.92278e-001
140	8.78906e-003	4.68750e-002	9.95634e-001
150	3.90625e-003	3.12500e-002	9.98053e-001
160	2.19727e-003	2.34375e-002	9.98903e-001
170	9.76563e-004	1.56250e-002	9.99512e-001

Tabela 4.14: Resultados numéricos para o problema não diferenciável Maxq.

Problema não diferenciável Maxl (ver teste 59 página 64)			
Iter	$G(x_k) - H(x_k)$	$\|\|x_{k+1} - x_k\|\|_2$	$cos(\omega_k)$
1	1.90000e+001	2.00000e+001	8.94427e-001
2	1.80000e+001	1.90000e+001	8.94427e-001
3	1.70000e+001	1.80000e+001	8.94427e-001
4	1.60000e+001	1.70000e+001	8.94427e-001
5	1.50000e+001	1.60000e+001	8.94427e-001
6	1.40000e+001	1.50000e+001	8.94427e-001
7	1.30000e+001	1.40000e+001	8.94427e-001
8	1.20000e+001	1.30000e+001	8.94427e-001
9	1.10000e+001	1.20000e+001	8.94427e-001
10	1.00000e+001	1.10000e+001	8.94427e-001
11	9.00000e+000	1.00000e+001	8.94427e-001
12	8.00000e+000	9.00000e+000	8.94427e-001
13	7.00000e+000	8.00000e+000	8.94427e-001
14	6.00000e+000	7.00000e+000	8.94427e-001
15	5.00000e+000	6.00000e+000	8.94427e-001
16	4.00000e+000	5.00000e+000	8.94427e-001
17	3.00000e+000	4.00000e+000	8.94427e-001
18	2.00000e+000	3.00000e+000	8.94427e-001
19	1.00000e+000	2.00000e+000	8.94427e-001
20	0.00000e+000	1.00000e+000	8.94427e-001

Tabela 4.15: Resultados numéricos para o problema não diferenciável Maxl.

Capítulo 5

Conclusões

Foi apresentada uma generalização do método de Newton para resolver equações convexas em $\mathbb{R}^n$. O método é inspirado nos métodos de relaxação atribuídos a Agmon [Agm54], Bregman [Bre65] e Eremin [EM67]. Posteriormente, o algoritmo foi posto segundo o paradigma do decréscimo de $f(x)$, usual nos métodos de otimização, onde a iteração x_{k+1} é computada por $x_{k+1} = x_k + t_k\, d_k$. Nos problemas diferenciáveis, d_k é uma direção de descida e $t_k \in \mathbb{R}$, $t_k > 0$ é um passo nesta direção.

Os testes em problemas diferenciáveis sugerem que para pelo menos uma classe de problemas (problemas separáveis), o método tem convergência quadrática.

É sabido que o método de Newton aplicado a problemas unidimensionais tem convergência lenta na presença de uma raiz com grau de multiplicidade superior a um. Tal fato também repetiu-se aqui, notadamente em problemas quadráticos. Sabe-se que o método de Schröder [Cam01] modifica a fórmula de recorrência do método de Newton para melhorar o desempenho nos problemas que possuem pelo menos uma raiz com multiplicidade maior que um. Seria interessante procurar interpretar o método de Schröder segundo a nossa metodologia o que eventualmente poderia produzir um método mais eficiente para essas classes de problemas.

Para os problemas convexos não suaves, não existe uma técnica geral, excluindo a otimização, para resolvê-los. Assim, escolhemos um conjunto de problemas oriundos dessa área para testar a metodologia. Os resultados foram, em geral, insatisfatórios; é bom lembrar que esses problemas foram construídos de forma que a direção sugerida por menos o subgradiente fosse inadequada.

Não foi possível extrair um padrão que permitisse construir algoritmos eficientes para

essa classe de problemas.

Uma outra abordagem para trabalhos futuros, consiste em obter problemas em que a função $f(x)$ é construída mesclando uma função diferenciável convexa com uma outra não diferenciável. Por exemplo, a solução de um sistema de equações lineares $Ax = b$, onde $A \in \mathbb{R}^{n \times n}$, x, $b \in \mathbb{R}^n$ corresponde a encontrar x tal que

$$Q(x) = \frac{1}{2}\, x^T A^T A\, x - (A^T b)^T x + \frac{1}{2}\, b^T b = 0\ . \tag{5.1}$$

O mesmo problema pode ser posto em função de um problema de achar a raiz de uma função convexa não suave; isto é, o ponto $\bar{x}$ tal que $A\bar{x} = b$ é o mesmo que resolve o problema

$$D(x) = Max\ \{\ |a_i^T x - b_i|, i = 1, 2, ...m\} = 0\ . \tag{5.2}$$

Juntando 5.1 com 5.2 obtém-se:

$$\begin{aligned} f(x) \quad = \quad & \left\{ \frac{1}{2}\, x^T A^T A\, x - (A^T b)^T x + \frac{1}{2}\, b^T b \right\} + \\ & Max\ \{\ |a_i^T x - b_i|, i = 1, 2, ...m\}\ . \end{aligned} \tag{5.3}$$

Ao se fazer $f(x) = g(x) + h(x)$, onde $g(x)$ é a função 5.2 e $-h(x)$ é a função 5.1 (ver figura 5.1) pode-se aplicar a metodologia proposta no nosso trabalho.

Outra fonte de problemas para os quais a metodologia descrita no capítulo 3 pode ser útil é oriunda das áreas de Engenharia e Estatística, notadamente quando a matriz Hessiana ($H(x)$) é difícil de ser obtida. Alguns problemas complexos envolvendo raízes de equações em $\mathbb{R}^n$ (por exemplo Santos,Aloizio [Sil01]) podem ser resolvidos facilmente com o auxílio dos instrumentos desenvolvidos neste trabalho.

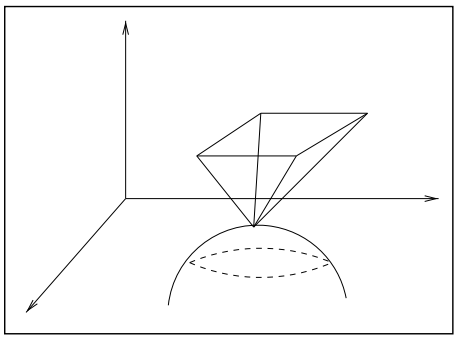


Figura 5.1: Forma mista para a resolução de um sistema de equações lineares usando as funções $Q(x)$ e $D(x)$.

Referências Bibliográficas

[Agm54] S. Agmon. The relaxation method for linear inequalities. *Canadian Journal of Mathematics*, 6:382–392, 1954.

[Bre65] L. M. Bregman. The method of sucessive projection for finding a commom point of convex sets. *Soviet Physics DokLady*, 162:688–692, 1965.

[Cam01] F. F. Campos, filho. *Algoritmos Numéricos*. Editora LTC, Rio de Janeiro, 2001.

[CC78] C. Charalambous and A. R. Conn. An efficient method to solve the minimax problem directly. *SIAM Journal on Numerical Analysis*, 15:162–187, 1978.

[DM76] B. P. Demidovich and I. A. Maron. *Computational Mathematics*. Editora Mir, Moscou, 1976.

[DM86] V. F. Demyanov and V. N. Malozemov. *Introdution to minimax*. Wiley, New York, 1986.

[dS02] M. A. dos Santos. Método das projeções sucessivas. Notas de aula, DCC-ICEx-UFMG, 2002.

[EM67] I. J. Eremin and Vl. D. Mazurov. Iteration method for solving problems of convex programming. *Soviet Physics DockLady*, 11:757–759, 1967.

[Gof77] J. L. Goffin. On convergence rates of subgradient methods. *Math Programming*, 13:329–347, 1977.

[Kiw90] K. C. Kiwiel. Proximity control in bundle methods for convex nondifferentiable minimization. *Mathematical Programing*, 46:105–122, 1990.

[LE77] C. Lemaréchal and R. Mifflin (Eds). *Nonsmooth Optimization*. Proceedings of a IIASA Workshop, Pergamanon Press, Oxford, 1977.

[MN92] M. M. Makela and P. Neittaanmaki. *Nonsmooth Optimization, Analysis and algorithms with applications to optimal control*. World Scientific Publishing Co., Singapore, 1992.

[OdS00] P. R. Oliveira and M. A. dos Santos. Cutting planes and analytic center methods for non smooth convex programming. *Lecture Notes in Mathematics*, 481:339–356, 2000.

[PTVF92] W. H. Press, S. A. Tenkolsky, W. T. Veherling, and B. P. Flannery. *Numerical Recipes in Fortran*. Cambridge University Press, Cambridge, 1992.

[Roc70] R. T. Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, 1970.

[Sch70] E. Schröder. Über unendlich viele Algorithmen zur Auflösung der Gleichungen. *Math. Ann.*, 2:317–365, 1870.

[Sho85] N. Z. Shor. *Minimization Methods for non-differentiable functions*. Springer-Verlag, Berlin, 1985.

[Shr89] H. Shramm. *Eine Kombination von bundle-und Trust-region-verfahren zur lösung nichtdifferenziebarer optimierungsprobleme*. Bayreuther Mathematidre schriften, Heft 30, Bayreuth, 1989.

[Sil01] A. P. Silva. Análise de desempenho de serviço de dados em redes de telecomunicações movéis de terceira geração. Projeto orientado, Universidade Federal de Minas Gerais, Março 2001.

Apêndice A

Problemas diferenciáveis e não diferenciáveis

A.1 Funções diferenciáveis em $\mathbb{R}^n$, onde $n = 1$.

Apresentaremos, abaixo, uma série de testes numéricos para funções convexas em R^n, onde n=1.

Teste 1 [Cam01]
Função: $f(x) = 2x^3 - \cos(x+1) - 3$
Ponto inicial: $x_0 = 1.50$
$G(x) = 2x^3$
$H(x) = \cos(x+1) + 3$

Teste 2 [Cam01]
Função: $f(x) = e^x - \sin(x) - 2$
Ponto inicial: $x_0 = 2.00$
$G(x) = e^x$
$H(x) = \sin(x) + 2$

Teste 3
Função: $f(x) = x^2 - \sin(x) - 1$
Ponto inicial: $x_0 = 2.00$
$G(x) = x^2$

$H(x) = \sin(x) + 1$

Teste 4
Função: $f(x) = -\sin(x) - \log_e(x) + 1$
Ponto inicial: $x_0 = 1.00$
$G(x) = -\sin(x)$
$H(x) = \log_e(x) - 1$

Teste 5 [Cam01]
Função: $f(x) = x^5 + x^3 + x^2 + x - 25$
Ponto inicial: $x_0 = 2.00$
$G(x) = x^5 + x^3$
$H(x) = -x^2 - x + 25$

Teste 6 [Cam01]
Função: $f(x) = 0.05x^3 - 0.4x^2 + 3\sin(x)x$
Ponto inicial: $x_0 = 2.50$
$G(x) = 0.05x^3 - 0.4x^2$
$H(x) = -3\sin(x)x$

Teste 7 [Cam01]
Função: $f(x) = \sin(x)x + 4$
Ponto inicial: $x_0 = 4.00$
$G(x) = \sin(x)x$
$H(x) = -4$

Teste 8
Função: $f(x) = 5x^3 + x^2 - e^{1-2x} + \cos(x) + 20$
Ponto inicial: $x_0 = 5.00$
$G(x) = 5x^3 + x^2 - e^{1-2x}$
$H(x) = -\cos(x) - 20$

Teste 9 [Cam01]
Função: $f(x) = 4x^3 + x + \cos(x) - 10$

Ponto inicial: $x_0 = 2.00$
$G(x) = 4x^3 + x$
$H(x) = -\cos(x) + 10$

Teste 10 [Cam01]
Função: $f(x) = 2x^3 + 5x^2 - \sin(x) - 30$
Ponto inicial: $x_0 = 4.00$
$G(x) = 2x^3 + 5x^2$
$H(x) = \sin(x) + 30$

Teste 11 [Cam01]
Função: $f(x) = e^{-cos(x)} + 2x^4 - x^2 - 5$
Ponto inicial: $x_0 = -2.00$
$G(x) = e^{-cos(x)} + 2x^4$
$H(x) = x^2 + 5$

Teste 12 [Cam01]
Função: $f(x) = 3x^x - \log_{10}(2 + \sqrt{x}) + 3\tan(x) - 4$
Ponto inicial: $x_0 = 1.00$
$G(x) = 3x^x - \log_{10}(2 + \sqrt{x})$
$H(x) = -3\tan(x) + 4$

Teste 13 [Cam01]
Função: $f(x) = 5\log_{10}(x) + 3x^4 - 7$
Ponto inicial: $x_0 = 2.00$
$G(x) = 5\log_{10}(x)$
$H(x) = -3x^4 + 7$

Teste 14 [Cam01]
Função: $f(x) = 5x^2 + \log_{10}(x + 1) - 2$
Ponto inicial: $x_0 = 1.00$
$G(x) = 5x^2$
$H(x) = -\log_{10}(x + 1) + 2$

Teste 15
Função: $f(x) = e^{-x} + x^2 - 10$
Ponto inicial: $x_0 = 6.00$
$G(x) = e^{-x}$
$H(x) = -x^2 + 10$

Teste 16 [Cam01]
Função: $f(x) = x^4 + 2x^3 - 13x^2 - 14x + 24$
Ponto inicial: $x_0 = -5.00$
$G(x) = x^4 + 2x^3 - 13x^2$
$H(x) = 14x - 24$

Teste 17 [Cam01]
Função: $f(x) = 2x^4 + 4x^3 + 3x^2 - 10x - 15$
Ponto inicial: $x_0 = 1.50$
$G(x) = 2x^4 + 4x^3$
$H(x) = -3x^2 + 10x + 15$

Teste 18
Função: $f(x) = 5x^3 - 2x^2 + 8x - 10$
Ponto inicial: $x_0 = 2.00$
$G(x) = 5x^3 - 2x^2$
$H(x) = -8x + 10$

Teste 19 [Cam01]
Função: $f(x) = e^{2x} - 2x^3 - 5$
Ponto inicial: $x_0 = 2.00$
$G(x) = e^{2x}$
$H(x) = 2x^3 + 5$

Teste 20
Função: $f(x) = e^x - 3x - 10$
Ponto inicial: $x_0 = 3.00$
$G(x) = e^x$

$H(x) = 3x + 10$

Teste 21
Função: $f(x) = 2^x + x^2 - 3$
Ponto inicial: $x_0 = 1.50$
$G(x) = 2^x$
$H(x) = -x^2 + 3$

Teste 22
Função: $f(x) = 3x^2 + \cos(x) - 5$
Ponto inicial: $x_0 = 2.50$
$G(x) = 3x^2$
$H(x) = -\cos(x) + 5$

Teste 23
Função: $f(x) = (\sin(x))^2 + (\cos(x))^2 + 3x^2 - 17$
Ponto inicial: $x_0 = 3.00$
$G(x) = (\sin(x))^2 + (\cos(x))^2$
$H(x) = -3x^2 + 17$

Teste 24
Função: $f(x) = x \log_e(x) - 20$
Ponto inicial: $x_0 = 9.50$
$G(x) = x \log_e(x)$
$H(x) = 20$

Teste 25
Função: $f(x) = x^2(x-2)^4 + 3x(x-2)^4$
Ponto inicial: $x_0 = -4.50$
$G(x) = x^2(x-2)^4$
$H(x) = -3x(x-2)^4$

A.2 Funções diferenciáveis em $\mathbb{R}^n$

Testes numéricos para funções convexas $x \in R^n$

Teste 26
Dimensão $= 2$, Funções: 2; 3
Função: $f(x)^* = e^{x_1} - \sin(x_1) - 2 + x_2^2 - \sin(x_2) - 1$

Teste 27
Dimensão $= 2$, Funções: 11; 12
Função: $f(x)^* = e^{-cos(x_1)} + 2x_1^4 - x_1^2 - 5 + 3x_2^{x_2} - \log_{10}(2 + \sqrt{x_2}) + 3\tan(x_2) - 4$

Teste 28
Dimensão $= 2$, Funções: 15; 19
Função: $f(x)^* = e^{-x_1} + x_1^2 - 10 + e^{2x_2} - 2x_2^3 - 5$

Teste 29
Dimensão $= 2$, Funções: 3; 12
Função: $f(x)^* = x_1^2 - \sin(x_1) - 1 + 3x_2^{x_2} - \log_{10}(2 + \sqrt{x_2}) + 3\tan(x_2) - 4$

Teste 30
Dimensão $= 2$, Funções: 2; 11
Função: $f(x)^* = e_1^x - \sin(x_1) - 2 + e^{-cos(x_2)} + 2x_2^4 - x_2^2 - 5$

Teste 31
Dimensão $= 3$, Funções: 2; 3, 11
Função: $f(x)^* = e_1^x - \sin(x_1) - 2 + x_2^2 - \sin(x_2) - 1 + e^{-cos(x_3)} + 2x_3^4 - x_3^2 - 5$

Teste 32
Dimensão $= 3$, Funções: 2; 11; 12
Função:
$$f(x)^* = e_1^x - \sin(x_1) - 2 + e^{-cos(x_2)} + 2x_2^4 - x_2^2 - 5 + 3x_3^{x_3} - \log_{10}(2 + \sqrt{x_3}) + 3\tan(x_3) - 4$$

Teste 33
Dimensão = 3,
Função não separável: $f(x)^* = e^{6x_1 - x_2 + 4x_3 - 2} - 3 + e^{4x_1 + x_2 + 2x_3 - 1.2} + e^{5x_1 + x_2 + 2x_3 - 3}$

Teste 34
Dimensão = 4, Funções: 2; 3; 15; 19
Função: $f(x)^* = e_1^x - \sin(x_1) - 2 + x_2^2 - \sin(x_2) - 1 + e^{-x_3} + x_3^2 - 10 + e^{2x_4} - 2x_4^3 - 5$

Teste 35
Dimensão = 4, Funções: 2; 3; 11; 12
Função: $f(x)^* = e_1^x - \sin(x_1) - 2 + x_2^2 - \sin(x_2) - 1 + e^{-cos(x_3)} + 2x_3^4 - x_3^2 - 5 + 3x_4^{x_4} - \log_{10}(2 + \sqrt{x_4}) + 3\tan(x_4) - 4$

Teste 36
Dimensão = 5, Funções: 2; 3; 11; 15; 19
Função: $f(x)^* = e_1^x - \sin(x_1) - 2 + x_2^2 - \sin(x_2) - 1 + e^{-cos(x_3)} + 2x_3^4 - x_3^2 - 5 + e^{-x_4} + x_4^2 - 10 + e^{2x_5} - 2x_5^3 - 5$

Teste 37
Dimensão = 6, Funções: 2; 3; 11; 15; 19; 20
Função: $f(x)^* = e_1^x - \sin(x_1) - 2 + x_2^2 - \sin(x_2) - 1 + e^{-cos(x_3)} + 2x_3^4 - x_3^2 - 5 + e^{-x_4} + x_4^2 - 10 + e^{2x_5} - 2x_5^3 - 5 + e^{x_6} - 3x_6 - 10$

Teste 38
Dimensão = 7, Funções: 2; 3; 11; 15; 19; 20; 21
Função: $f(x)^* = e_1^x - \sin(x_1) - 2 + x_2^2 - \sin(x_2) - 1 + e^{-cos(x_3)} + 2x_3^4 - x_3^2 - 5 + e^{-x_4} + x_4^2 - 10 + e^{2x_5} - 2x_5^3 - 5 + e^{x_6} - 3x_6 - 10 + 2^{x_7} + x_7^2 - 3$

Teste 39
Dimensão = 8, Funções: 2; 3; 11; 15; 19; 20; 21; 22
Função: $f(x)^* = e_1^x - \sin(x_1) - 2 + x_2^2 - \sin(x_2) - 1 + e^{-cos(x_3)} + 2x_3^4 - x_3^2 - 5 + e^{-x_4} + x_4^2 - 10 + e^{2x_5} - 2x_5^3 - 5 + e^{x_6} - 3x_6 - 10 + 2^{x_7} + x_7^2 - 3 + 3x_8^2 + \cos(x_8) - 5$

Teste 40
Dimensão = 9, Funções: 2; 3; 11; 15; 19; 20; 21; 22; 23
Função: $f(x)^* = e_1^x - \sin(x_1) - 2 + x_2^2 - \sin(x_2) - 1 + e^{-cos(x_3)} + 2x_3^4 - x_3^2 - 5 + e^{-x_4} + x_4^2 - 10 + e^{2x_5} - 2x_5^3 - 5 + e^{x_6} - 3x_6 - 10 + 2^{x_7} + x_7^2 - 3 + 3x_8^2 + \cos(x_8) - 5 + (\sin(x_9))^2 + (\cos(x_9))^2 + 3x_9^2 - 17$

Teste 41
Dimensão = 10, Funções: 2; 3; 11; 15; 19; 20; 21; 22; 23; 24
Função: $f(x)^* = e_1^x - \sin(x_1) - 2 + x_2^2 - \sin(x_2) - 1 + e^{-cos(x_3)} + 2x_3^4 - x_3^2 - 5 + e^{-x_4} + x_4^2 - 10 + e^{2x_5} - 2x_5^3 - 5 + e^{x_6} - 3x_6 - 10 + 2^{x_7} + x_7^2 - 3 + 3x_8^2 + \cos(x_8) - 5 + (\sin(x_9))^2 + (\cos(x_9))^2 + 3x_9^2 - 17 + x_{10}\log_e(x_{10}) - 20$

A.3 Funções diferenciáveis não separáveis

Testes em $\mathbb{R}^n$ com equações diferenciáveis bem acopladas.

Teste 42
Dimensão = 3,
Função não separável: $f(x)^* = -0.5\log_e(x^t A^t A x) + 13 - 0.5\log_e(x^t A^t A x)$

Teste 43
Dimensão = 3

Ponto inicial: $x_0 = [\ 0 \quad 0 \quad 1\ ]^T$
Função não separável: $f(x)^* = \frac{1}{2}\, x^T A^T A\, x - (A^T b)^T x + \frac{1}{2}\, b^T b$, sendo:

$$A = \begin{bmatrix} 100 & 1 & 2 \\ 3 & 200 & 4 \\ 5 & 6 & 300 \end{bmatrix} \quad \text{e} \quad b = [\ 12 \quad 100 \quad 24\ ]^T .$$

Teste 44
Dimensão = 2
Ponto inicial: $x_0 = [\ 3 \quad 2\ ]^T$
Função: $f(x)^* = e^{(x_1 - 2x_2 - 14)} + e^{(5x_1 + 4x_2 - 9)} - \frac{1}{2}\, x^T A^T A\, x + (A^T b)^T x - \frac{1}{2}\, b^T b$

$G(x) = e^{(x_1 - 2x_2 - 14)} + e^{(5x_1 + 4x_2 - 9)}$
$H(x) = \frac{1}{2}\, x^T A^T A\, x - (A^T b)^T x + \frac{1}{2}\, b^T b$
Número de Iterações = 19
$f(\tilde{x}) = 2.55795\text{e-}12$

Teste 45
Dimensão = 3
Ponto inicial: $x_0 = [\ 1 \quad 1 \quad 1\ ]^T$
Função: $f(x)^* = e^{(3x_1 - 4x_2 + x_3 - 16)} + e^{(5x_1 + 5x_2 + 12x_3 - 4)} + e^{(x_1 + 2x_2 - 4x_3 - 4)} - \frac{1}{2}\, x^T A^T A\, x + (A^T b)^T x - \frac{1}{2}\, b^T b$

Número de Iterações = 18
$f(\tilde{x}) = 6.25278\text{e-}13$

Teste 46
Dimensão = 4
Ponto inicial: $x_0 = [\ 1 \quad 1 \quad 1 \quad 1\ ]^T$
Função: $f(x)^* = e^{(3x_1 - 4x_2 + x_3 - 2x_4 - 16)} + e^{(5x_1 + 5x_2 + 12x_3 - 8x_4 - 4)} + e^{(x_1 + 2x_2 - 4x_3 + 3x_4 - 4)} + e^{(4x_1 - 8x_2 + 7x_3 - x_4 - 6)} - \frac{1}{2}\, x^T A^T A\, x + (A^T b)^T x - \frac{1}{2}\, b^T b$

Número de Iterações = 10
$f(\tilde{x}) = 8.75389\text{e-}12$

Teste 47
Dimensão = 5
Ponto inicial: $x_0 = [\ 1 \quad 1 \quad 1 \quad 1 \quad 1\ ]^T$
Função: $f(x)^* = e^{(3x_1-4x_2+x_3-2x_4+3x_5-16)} + e^{(5x_1+5x_2+12x_3-8x_4-3x_5-4)} + e^{(x_1+2x_2-4x_3+3x_4-5x_5-4)} + e^{(4x_1-8x_2+7x_3-x_4+2x_5-6)} + e^{(9x_1+x_2-7x_3+3x_4-2x_5-12)} - \frac{1}{2}\, x^T A^T A\, x + (A^T b)^T x - \frac{1}{2}\, b^T b$

Número de Iterações = 7
$f(\tilde{x})$ = 1.50067e-11

A.4 Funções convexas não suaves

Exemplos de funções convexas não necessariamente diferenciáveis:

Teste 48 - CB2 [CC78]
Dimensão = 2,
Função:
$$f(x) = max\{x_1^2 + x_2^4, (2 - x_1)^2 + (2 - x_2)^2, 2e^{-x_1+x_2}\} - 1.9522;$$

Solução: $x^* = (1.1393, 0.8994)$,
Ponto inicial: $x^0 = (-1.5, 2)$.

Teste 49 - CB3 (figura A.1) [CC78]
Dimensão = 2,
Função:
$$f(x) = max\{x_1^4 + x_2^2, (2 - x_1)^2 + (2 - x_2)^2, 2e^{-x_1+x_2}\} - 2;$$

Solução: $x^* = (1, 1)$,
Ponto inicial: $x^0 = (2, 2)$.

Teste 50 - DEM (figura A.2) [DM86]
Dimensão = 2,
Função:
$$f(x) = max\{5x_1 + x_2, -5x_1 + x_2,\ x_1^2 + x_2^2 + 4x_2\} + 3;$$

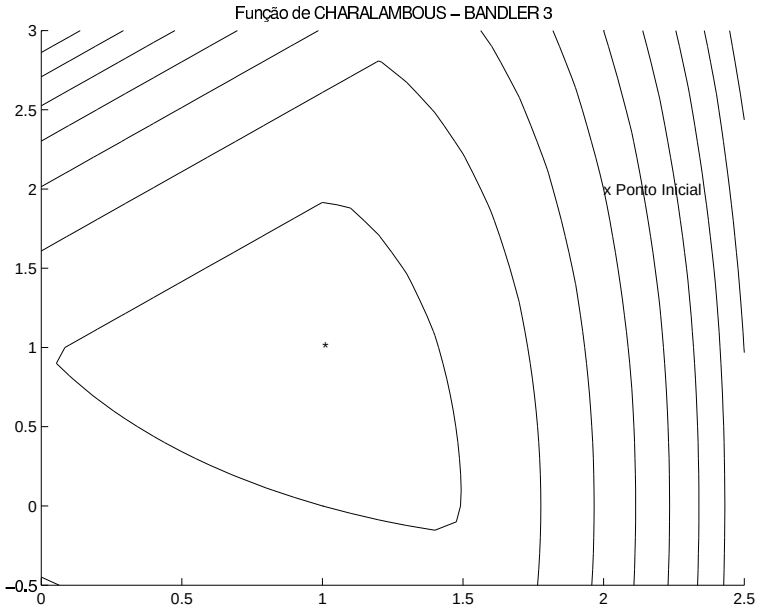


Figura A.1: Exemplo CB3 (teste 49) de função convexa não necessariamente diferenciável.

Solução: $x^* = (0, -3)$,
Ponto inicial: $x^0 = (1, 1)$.

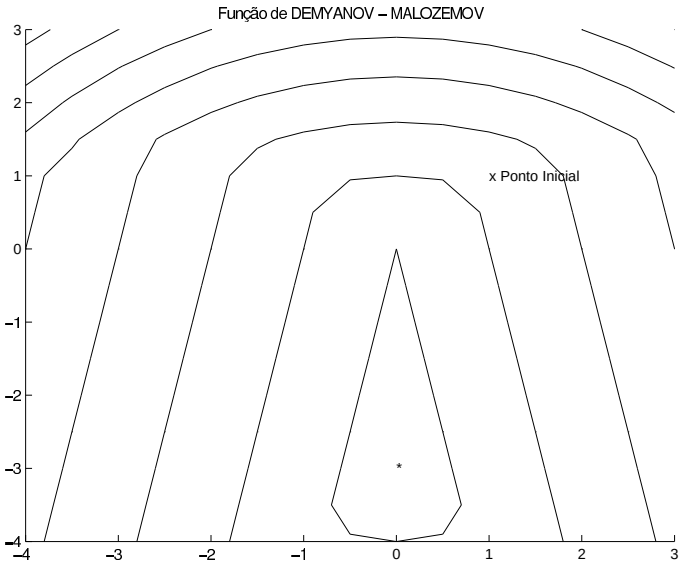


Figura A.2: Exemplo DEM (teste 50) de função convexa não necessariamente diferenciável.

Teste 51 - QL (figura A.3) [MN92]
Dimensão = 2,
Função: $f(x) = max\{x_1^2 + x_2^2,$
$x_1^2 + x_2^2 + 10(-4x_1 - x_2 + 4),$
$x_1^2 + x_2^2 + 10(-x_1 - 2x_2 + 6)\} - 7.2,$

Solução: $x^* = (1.2, 2.4)$,
Ponto inicial: $x^0 = (-1, 5)$.

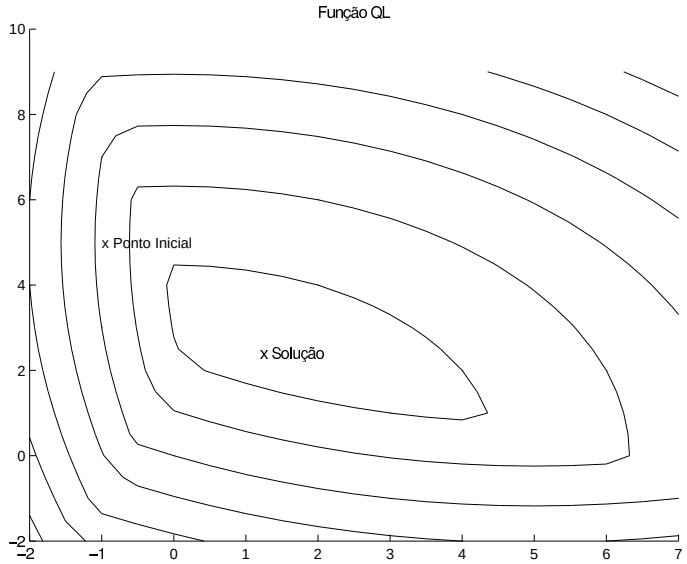


Figura A.3: Exemplo QL (teste 51) de função convexa não necessariamente diferenciável.

Teste 52 - LQ (figura A.4) [MN92]
Dimensão = 2,
Função:

$$f(x) = max\{-x_1 - x_2, -x_1 - x_2 + (x_1^2 + x_2^2 - 1)\} + \sqrt{2};$$

Solução: $x^* = \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)$,
Ponto inicial: $x^0 = (-0.5, -0.5)$.

Teste 53 - Mifflin1 (figura A.5) [LE77]
Dimensão = 2,
Função:

$$f(x) = -x_1 + 20\max\{x_1^2 + x_2^2 - 1, 0\} + 1;$$

Solução: $x^* = (1, 0)$,
Ponto inicial: $x^0 = (0.8, 0.6)$.

Teste 54 - Mifflin2 (figura A.6) [LE77]
Dimensão = 2,
Função:

$$f(x) = -x_1 + 2(x_1^2 + x_2^2 - 1) + 1.75|x_1^2 + x_2^2 - 1| + 1;$$

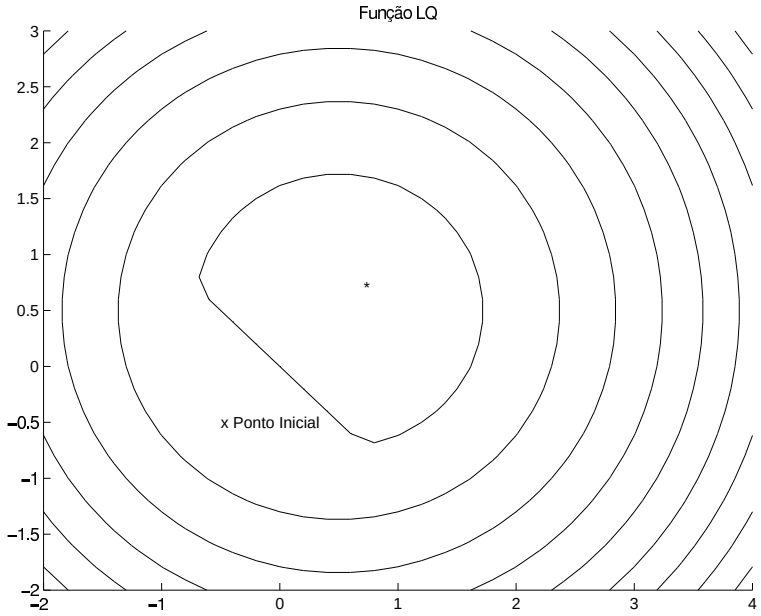


Figura A.4: Exemplo LQ (teste 52) de função convexa não necessariamente diferenciável.

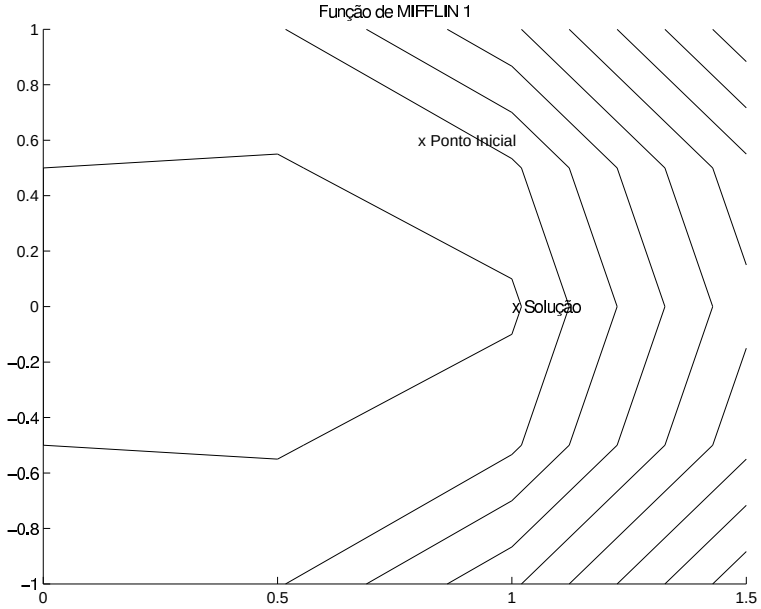


Figura A.5: Exemplo Mifflin1 (teste 53) de função convexa não necessariamente diferenciável.

Solução: $x^* = (1, 0)$,
Ponto inicial: $x^0 = (-1, -1)$.

Teste 55 - Rosen [MN92]
Dimensão = 4,
Função:

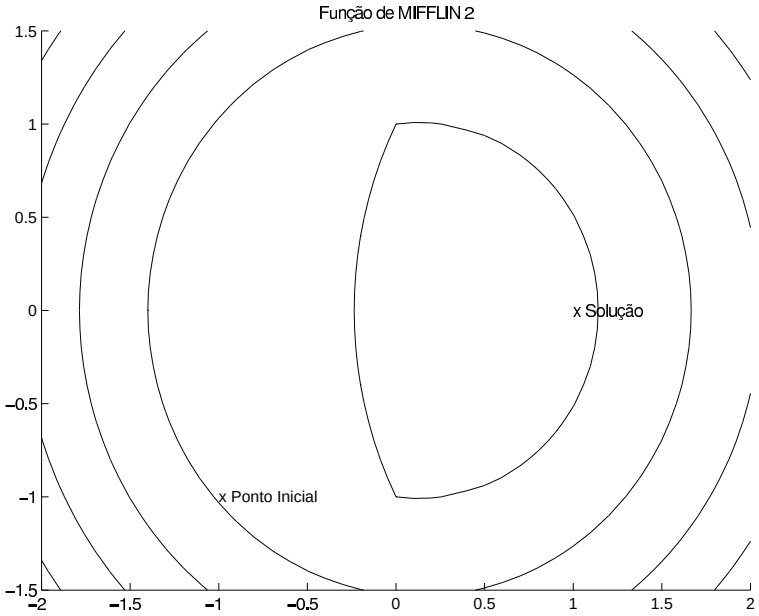


Figura A.6: Exemplo Mifflin2 (teste 54) de função convexa não necessariamente diferenciável.

$$
\begin{aligned}
f(x) = \quad & max\{f_1(x), \\
& \quad f_1(x) + 10f_2(x), \\
& \quad f_1(x) + 10f_3(x), \\
& \quad f_1(x) + 10f_4(x)\} + 44, \\
& f_1(x) = x_1^2 + x_2^2 + 2x_3^2 + x_4^2 - 5x_1 - 5x_2 - 21x_3 + 7x_4, \\
& f_2(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_1 - x_2 + x_3 - x_4 - 8, \\
& f_3(x) = x_1^2 + 2x_2^2 + x_3^2 + 2x_4^2 - x_1 - x_4 - 10, \\
& f_4(x) = x_1^2 + x_2^2 + x_3^2 + 2x_1 - x_2 - x_4 - 5\} + 44
\end{aligned}
$$

Solução: $x^* = (0, 1, 2, -1)$,
Ponto inicial: $x^0 = (0, 0, 0, 1)$.

Teste 56 - Maxquad [LE77]
Dimensão = 10,
Função:

$$f(x) = \quad \max_{1 \le i \le 5} \{x^T A_i x - b_i^T x\} + 0.8414084;$$

Ponto inicial 1: $x^0 = (1, ..., 1)$,
Ponto inicial 2: $x^0 = (0, ..., 0)$.

Teste 57 - Shor [Sho85]

Dimensão $= 5$,
Função:

$$f(x) = \max_{1 \le i \le 10} \{b_i \sum_{j=1}^{5} (x_j - a_{i,j})^2\} - 22.600016;$$

Ponto inicial: $x^0 = (0, 0, 0, 0, 1)$.

Teste 58 - Maxq [MN92]
Dimensão $= 20$,
Função:

$$f(x) = \max_{1 \le i \le 20} \{x_i^2\};$$

Ponto inicial: $x_i^0 = i,\ i = 1, ..., 10$,
Ponto inicial: $x_i^0 = -i,\ i = 11, ..., 20$.

Teste 59 - Maxl [MN92]
Dimensão $= 20$,
Função:

$$f(x) = \max_{1 \le i \le 20} \{|x_i|\};$$

Ponto inicial: $x_i^0 = i,\ i = 1, ..., 10$,
Ponto inicial: $x_i^0 = -i,\ i = 11, ..., 20$.