Mariana Ventureli da Veiga

Métodos para a Resolução do Subproblema do Algoritmo de Região de Confiança

Florianópolis
2017

MARIANA VENTURELI DA VEIGA

MÉTODOS DE RESOLUÇÃO DO SUBPROBLEMA DO ALGORITMO DE REGIÃO DE CONFIANÇA

Trabalho de Conclusão de Curso submetido à Universidade Federal de Santa Catarina, como requisito necessário para obtenção do grau de Bacharel em Matemática e Computação Científica

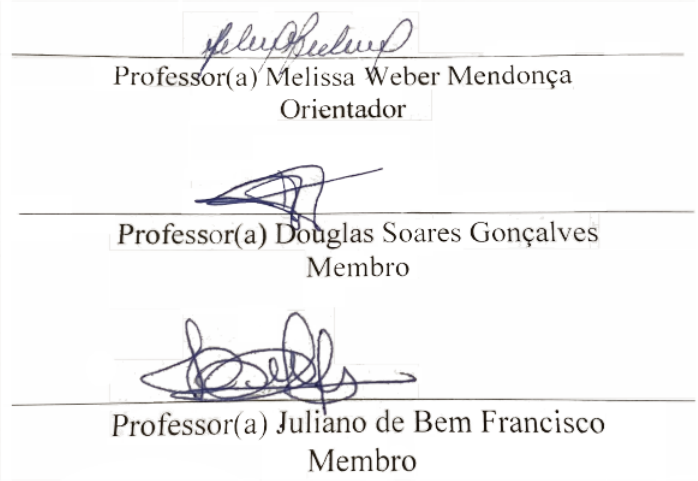
Orientador: Dra. Melissa Weber Mendonça

Florianópolis, 17 novembro de 2017

UNIVERSIDADE FEDERAL DE SANTA CATARINA

MARIANA VENTURELI DA VEIGA

Esta Monografia foi julgada adequada para a obtenção do título de Bacharel em Matemática e Computação Científica, sendo aprovada em sua forma final pela banca examinadora:



Florianópolis, 17 de novembro de 2017

Agradecimentos

Agradeço primeiramente aos meus pais, José Carlos e Marisa, ao meu irmão Carlos Eduardo e à minha cunhada Franciele Luane por todo apoio e suporte nesses anos de graduação, vocês tornaram possível a realização desse sonho.

Agradeço também ao meu namorado Vinícius por acreditar na minha capacidade, motivar quando o cansaço era grande, consolar quando as coisas não davam certo, ouvir explicações de coisas que não faziam o mínimo sentido e todas as outras pequenas ajudas, conselhos e coisas maravilhosas que tu fez.

Agradeço a minha orientadora, professora Melissa, por me apresentar o maravilhoso mundo da otimização, por aceitar ser minha orientadora e pelas dicas e infindáveis ajudas durante este trabalho e em todas as disciplinas que eu tive a felicidade de ser sua aluna.

Agradeço de coração aos "bagunceiros" Francieli, Josiane, Rafaela, Jéssica, Marduck, Elemar e Everton por todo o apoio, as risadas, as ajudas e os cafés das 15h. Vocês são amigos maravilhosos.

Agradeço ao meu gato Muffin por tentar me ajudar na digitação deste e outros trabalhos subindo (e deitando) em cima do teclado e por todos os outros momentos de relaxamento e descontração.

Resumo

Problemas de otimização estão presentes em diversas áreas de pesquisa e até no cotidiano das pessoas. Existem diversos métodos para resolver esses problemas, dentre eles o método de região de confiança. Em sua implementação é necessário resolver um subproblema, que também é um problema de otimização. Neste trabalho, apresentaremos o método de região de confiança e dois métodos para a resolução do subproblema. Por último, serão apresentados resultados numéricos da aplicação desses métodos.

Palavras-chave: otimização, região de confiança, subproblema.

Abstract

Optimization problems are present in several research areas and even in people's daily lives. There are several methods to solve these problems, among them the trust region method. In its implementation it is necessary to solve a subproblem, which is also an optimization problem. In this work, we will present the trust region method and two methods for solving the subproblem. Finally, numerical results of the application of these methods will be presented.

Keywords: optimization, trust region, subproblem.

Lista de ilustrações

Lista de tabelas

Lista de símbolos

$\mathbb{R}$	Conjunto dos números reais
$\mathbb{R}^n$	Espaço Euclidiano dos números reais de dimensão n
$\mathcal{A}(x)$	Conjunto das restrições ativas em x
$\mathcal{V}$	Vizinhança
$\mathcal{B}_k$	Região de Confiança na iteração k
$f(\cdot)$	Função objetivo
$\nabla f(\cdot)$	Gradiente da função objetivo
$\nabla^2 f(\cdot)$	Hessiana da função objetivo
x_k	k-ésima iteração do ponto x
H_k	Hessiana da função objetivo em x_k
$\mathcal{I}$	Conjunto de índices da restrição de desigualdade
$\mathcal{K}(H, g_0, k)$	Espaço de Krylov
$m_k(\cdot)$	Modelo quadrático definido na iteração k
$q(\cdot)$	Função quadrática
s_k	Passo na iteração k
V	Conjunto das iterações bem sucedidas
S	Conjunto das iterações muito bem sucedidas
x_*	Minimizador local de f
x_k^{AC}	Ponto de Cauchy aproximado
x_k^C	Ponto de Cauchy
$x_k^C(\cdot)$	Arco de Cauchy
x_k^M	Solução exata do subproblema
γ_1	Constante do algoritmo BTR

γ_2	Constante do algoritmo BTR
η_1	Constante do algoritmo BTR
η_2	Constante do algoritmo BTR
Δ_k	Raio da região de confiança
ε	Precisão do algoritmo BTR
ε_{MS}	Precisão do algoritmo MS
ε_{GCT}	Precisão do algoritmo GCT
κ_{lbf}	Limitante inferior da função objetivo
κ_{ufh}	Limitante superior da hessiana da função objetivo
κ_{umh}	Limitante superior da hessiana do modelo
Λ	Matriz dos autovalores
λ^U	Limitante superior para o multiplicador de Lagrange
λ^L	Limitante inferior para o multiplicador de Lagrange
$\phi(\cdot)$	Equação secular
ρ_k	Razão entre redução no modelo e redução na função objetivo
$\|\| \cdot \|\|_k$	Norma vetorial

Lista de abreviaturas e siglas

BTR	*Basic Trust Region* (Região de Confiança Básico)
MS	Moré-Sorensen
GCT	Gradiente Conjugado Truncado
KKT	Karush-Kuhn-Tucker

Sumário

Introdução

Na Matemática a área de Otimização é relativamente nova se compararmos com as outras áreas, apesar que frequentemente problemas de otimização surgem no cotidiano de muitas pessoas. Por exemplo um empresário gostaria de maximizar os lucros da sua empresa ou um fabricante gostaria de minimizar o custo na produção de um determinado produto.

Muitos problemas modelados nas ciências aplicadas, engenharia e economia podem ser reformulados como problemas de otimização. Com o avanço teórico e tecnológico dessas áreas os problemas ficaram maiores e mais complexos, logo são necessários métodos que resolvam esses problemas utilizando de forma inteligente as informações dadas. Felizmente a Matemática, e em particular a Otimização, avançaram bastante na teoria e graças ao desenvolvimento tecnológico das últimas décadas, é possível resolver os problemas que surgem nas ciências aplicadas e na vida em geral.

Neste trabalho estudaremos o método de região de confiança e sua teoria de convergência para a resolução de um problema de otimização irrestrita. Além disso, estudaremos dois métodos que solucionam o subproblema presente no algoritmo de região de confiança, sendo que ele também é um problema de otimização, mas agora restrito. Finalmente, são apresentados resultados numéricos.

Para o bom acompanhamento da teoria que será apresentada aqui é necessário que o leitor tenha conhecimentos prévios de Cálculo e Álgebra Linear, assim recomendamos [1] e [2] como literatura complementar.

1 O Método de Região de Confiança

O problema de otimização irrestrito está presente em diversas áreas da engenharia, física e outras ciências aplicadas. Ele também pode ser um subproblema a ser resolvido para obtermos a solução principal de um problema.

Estamos interessados em determinar $x_* \in \mathbb{R}^n$ que é solução do problema

$$\min_{x \in \mathbb{R}^n} f(x) \tag{1.1}$$

onde $f : \mathbb{R}^n \to \mathbb{R}$ é uma função duas vezes continuamente diferenciável, ou seja, f e ∇f são contínuas e diferenciáveis e a hessiana de f, $\nabla^2 f$, é contínua. Queremos funções com essa característica pois usaremos as informações de segunda ordem, já que ela tem papel importante na teoria que desenvolveremos adiante. Chamamos (1.1) de um problema de minimização irrestrito, pois não há restrições sobre a variável x.

Considere o problema $\max f(x)$, então queremos determinar x_* tal que $f(x_*) \geq f(x)$. Multiplicando a desigualdade anterior por -1 temos $-f(x_*) \leq -f(x)$, logo x_* é solução do problema $\min -f(x)$. Assim,

$$\max f(x) = -\min(-f(x))$$

e daqui em diante trataremos apenas problemas de minimização.

Problemas de otimização irrestrita pequenos podem ser resolvidos de maneira analítica, isto é, utilizando ferramentas como soma, subtração e regras de derivação, por exemplo. Porém por vezes estamos interessados em obter a solução para problemas de grande porte com milhares de variáveis e fazer isso manualmente se torna inviável. Assim, estudaremos métodos computacionais que possam calcular a solução aproximada de algum problema de otimização mais rapidamente e com uma boa precisão.

Existem diversos métodos para resolver o problema (1.1), dentre eles estamos interessados no método de região de confiança, que utiliza o Teorema de Taylor para derivar o modelo que faz parte do método e que usaremos ao longo deste trabalho. A seguir enunciaremos este resultado.

Teorema 1.1. *(Teorema de Taylor) Sejam $f : \mathbb{R}^n \to \mathbb{R}$ função continuamente diferenciável e $s \in \mathbb{R}^n$. Então existe algum $t \in (0, 1)$ tal que*

$$f(x + s) = f(x) + \langle \nabla f(x + ts), s \rangle. \tag{1.2}$$

Mais ainda, se f é duas vezes continuamente diferenciável, temos que

$$\nabla f(x + s) = \nabla f(x) + \int_0^1 \nabla^2 f(x + ts)s \; dt. \tag{1.3}$$

e que

$$f(x+s) = f(x) + \langle \nabla f(x), s\rangle + \frac{1}{2}\langle s, \nabla^2 f(x+ts)s\rangle, \tag{1.4}$$

para algum $t \in (0,1)$.

Note que os $t \in (0,1)$ que aparece em (1.2), (1.3) e (1.4) não são necessariamente iguais.

O Teorema de Taylor, nos diz que localmente podemos aproximar uma função qualquer por uma quadrática, para tanto a função deve ser duas vezes continuamente diferenciável. Além disso, o Teorema 1.1 também é muito importante pois justifica as condições de otimalidade que caracterizarão a solução do problema (1.1). Assim sendo, primeiro caracterizaremos a solução do problema para então estudar o método de região de confiança.

1.1 A solução do problema irrestrito

Na otimização irrestrita, minimizar uma função $f : \mathbb{R}^n \to \mathbb{R}$ em $\mathbb{R}^n$, significa encontrar o ponto $x_* \in \mathbb{R}^n$ para o qual a função atinge seu menor valor em todo o seu domínio, ou seja,

$$f(x) \geq f(x_*), \qquad \text{para todo } x \in \mathbb{R}^n.$$

Neste caso, dizemos que x_* é um *minimizador global* da função. Em geral, temos apenas informações locais da função, então encontrar o minimizador global pode ser muito difícil. Entretanto, podemos utilizar essas informações de f para encontrar localmente o ponto em que a função atinge seu menor valor. Formalmente, dizemos que x_* será um *minimizador local* de f se em uma vizinhança aberta $\mathcal{V}$ de x_* temos que

$$f(x_*) \leq f(x), \qquad \text{para todo } x \in \mathcal{V}.$$

Se a desigualdade acima for estrita, então dizemos que x_* é um *minimizador local estrito* de f.

Temos a definição de um minimizador local, porém ainda é difícil saber se um determinado ponto é de fato mínimo naquela vizinhança, pois podem haver infinitos pontos nela. Verificar cada um desses pontos pode ser computacionalmente caro e não é desejável esperar demasiadamente por um resultado. Assim, apresentaremos propriedades, chamadas de condições de otimalidade, que facilitarão no reconhecimento de possíveis mínimos locais. As condições apresentadas a seguir são necessárias, ou seja, o provável minimizador da função objetivo deve satisfazer essas condições.

Teorema 1.2. *(Condição Necessária de Primeira Ordem) Se* x_* *é um minimizador local de* $f : \mathbb{R}^n \to \mathbb{R}$*, onde* f *é continuamente diferenciável numa vinhança aberta* $\mathcal{V}$ *de* x_**,*

então

$$\nabla f(x_*) = 0. \tag{1.5}$$

Como não temos hipóteses sobre a convexidade de f, não podemos afirmar se o ponto é de máximo, mínimo ou cela. Assim se f for duas vezes continuamente diferenciável em uma vizinhança $\mathcal{V}$ de x_*, então temos uma segunda condição, enunciada a seguir.

Teorema 1.3. *(Condição Necessária de Segunda Ordem) Se x_* é um minimizador local de f e f é duas vezes continuamente diferenciável numa vizinhança aberta $\mathcal{V}$ de x_*, então*

1. $\nabla f(x_*) = 0$;
2. $\nabla^2 f(x_*)$ *é semi definida positiva, isto é,* $\langle x, H(x^*)x\rangle \geq 0$, *para todo* x.

O ponto que satisfizer (1.5) é chamado de *ponto estacionário*. Além disso, note que um minimizador é sempre um ponto estacionário mas a recíproca não é verdadeira. Felizmente, o próximo resultado garante que se um ponto estacionário x_* satisfazer certas condições, então ele é um minimizador local.

Teorema 1.4. *(Condições Suficientes de Segunda Ordem) Se f é uma função duas vezes continuamente diferenciável em uma vizinhança $\mathcal{V}$ de x_* e se x_* satisfazer*

1. $\nabla f(x_*) = 0$ *e*
2. $\nabla^2 f(x_*)$ *é definida positiva, isto é,* $\langle x, \nabla^2 f(x_*)x\rangle > 0$, *para todo* x,

então x_* é minimizador local estrito de f.

A diferença entre as condições dadas pelos Teoremas 1.2 e 1.3 e o Teorema 1.4 é que os dois primeiros resultados não garantem que o x_* encontrado será de fato um minimizador, porém o terceiro resultado garante que se as condições são satisfeitas, então o ponto encontrado será um minimizador local.

Agora que caracterizamos a solução podemos seguir para o estudo do método de região de confiança, descrevendo o funcionamento dele e sua teoria de convergência.

1.2 O Algoritmo de Região de Confiança

Existem diversos tipos de métodos, por exemplo os métodos diretos e os iterativos. O primeiro tipo resolve o problema de forma exata, já o segundo gera uma sequência cujos

pontos de acumulação são candidatos a pontos críticos. O método de região de confiança é um exemplo de método iterativo e o descreveremos a seguir.

Um algoritmo de região de confiança básico funciona da seguinte forma: dado um ponto inicial x_0, a cada iteração k, definimos a "região de confiança" ao redor do ponto x_k, que é o conjunto

$$\mathcal{B}_k = \{x \in \mathbb{R}^n | \ ||x - x_k||_k \leq \Delta_k\} \tag{1.6}$$

onde Δ_k é chamado o raio da região de confiança e $||\cdot||_k$ é uma norma vetorial que pode ser definida a cada iteração. Como usaremos apenas a norma Euclidiana durante todo o trabalho, deste ponto em diante denotaremos ela por $||\cdot||$.

Em seguida definimos um modelo que deve representar bem a função na região de confiança. Como o Teorema 1.1 garante que podemos aproximar qualquer função por uma quadrática localmente, definimos o modelo quadrático $m_k(s_k)$ que aproximará a função objetivo na região de confiança como

$$m_k(s_k) = f(x_k) + \langle \nabla f(x_k), s_k \rangle + \frac{1}{2}\langle s_k, H_k s_k \rangle \tag{1.7}$$

onde $\nabla f(x_k)$ é o gradiente de f no ponto x_k e H_k é a matriz hessiana de f em x_k ($\nabla^2 f(x_k)$). Queremos que o formato da região de confiança reflita a região onde acreditamos que o modelo aproxime bem a função objetivo, ou seja, onde a razão

$$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{m_k(0) - m_k(s_k)} \tag{1.8}$$

esteja perto de 1. Na figura 1.2 é possível ver a similaridade da função e do modelo na região de confiança após algumas iterações.

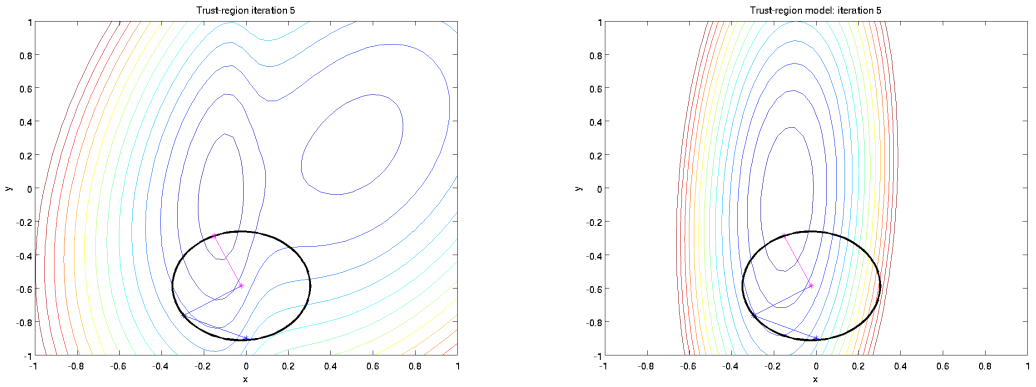


Figura 1 – Comparativo entre a função e o modelo dentro da região de confiança

Precisamos encontrar uma direção s_k que reduza o modelo dentro da região de confiança, ou seja, determinar a solução do subproblema

$$\min_{x_k + s \in \mathcal{B}_k} m_k(s_k). \tag{1.9}$$

Essa solução nos dará um passo s_k que reduz o modelo na região de confiança. Então definimos o ponto teste $x_k + s_k$ e veremos se a redução predita pelo modelo se realiza quando aplicamos o ponto teste na função objetivo. Fazemos isso analisando o resultado obtido em (1.8). Se ρ_k estiver perto de 1 ou $\rho_k \geq \eta_1$, para $\eta_1 \in (0, 1)$, temos que a redução predita pelo modelo confirmou-se, então o ponto teste é dito bem-sucedido e é aceito para a nova iteração, $x_{k+1} = x_k + s_k$. Se $\rho_k < \eta_1$, a redução predita pelo modelo não se concretizou na função objetivo, então descartamos o ponto teste e reduzimos o raio da região de confiança para que possamos resolver o problema (1.9) novamente e eventualmente o raio ficará pequeno suficiente para que o Teorema de Taylor aproxime bem a função,assim acharemos uma direção em que $x_k + s_k$ seja bem-sucedido.

Algoritmo 1.5. (**Algoritmo Básico de Região de Confiança (BTR)**)
Passo 0: Inicialização

Um ponto inicial x_0 e um raio de região de confiança inicial Δ_0 são dados. As constantes η_1, η_2, γ_1 e γ_2 também são dadas e satisfazem

$$0 < \eta_1 \leq \eta_2 < 1 \text{ e } 0 < \gamma_1 \leq \gamma_2 < 1 \tag{1.10}$$

Calcule $f(x_0)$ e defina $k = 0$.

Passo 1: Definição do modelo

Defina um modelo m_k em $\mathcal{B}_k$.

Passo 2: Cálculo do passo

Calcule um passo s_k que reduz o modelo "suficientemente" e tal que $x_k + s_k \in \mathcal{B}_k$, ou seja, resolva (1.9).

Passo 3: Aceitação do passo experimental

Calcule $f(x_k + s_k)$ e defina

$$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)}$$

Se $\rho_k \geq \eta_1$, defina $x_{k+1} = x_k + s_k$. Caso contrário, $x_{k+1} = x_k$.

Passo 4: Atualização do raio da região de confiança

Faça

$$\Delta_{k+1} \in \begin{cases} [\Delta_k, \infty) \text{ se } \rho_k \geq \eta_2 \\ [\gamma_2 \Delta_k, \Delta_k] \text{ se } \rho_k \in [\eta_1, \eta_2) \\ [\gamma_1 \Delta_k, \gamma_2 \Delta_k] \text{ se } \rho_k < \eta_1 \end{cases} \tag{1.11}$$

Faça $k = k + 1$.

Iterações que tem $\rho_k \geq \eta_1$ são chamadas iterações bem sucedidas e denotamos o conjunto dos seus índices pelo símbolo S. Assim

$$\mathsf{S} = \{k \geq 0 \mid \rho_k \geq \eta_1\}$$

Similarmente definimos

$$\mathsf{V} = \{k \geq 0 \mid \rho_k \geq \eta_2\}$$

o conjunto das iterações que são muito bem sucedidas. Iterações que não pertencem a S são ditas mal sucedidas.

O Algoritmo 1.5 deixa muitos detalhes não especificados, como os valores para os parâmetros em (1.10), a atualização do raio da região de confiança em (1.11) e a norma usada no Passo 1.

A seguir veremos a teoria de convergência para o Algoritmo 1.5, onde mostraremos o que é necessário para que a sequência de iterações gerada por esse algoritmo convirja para um ponto crítico da função.

1.3 Teoria de Convergência

Nesta seção trataremos dos resultados que garantem a convergência do método de região de confiança como foi apresentado aqui. Para tanto, seguimos o desenvolvimento feito em [6].

1.3.1 Hipóteses

Agruparemos por tipo todas as hipóteses necessárias no decorrer desta seção. São elas:

Hipótese 1. ***(Sobre a função objetivo)*** *A função objetivo f é tal que*

1. *$f : \mathbb{R}^n \to \mathbb{R}$ é duas vezes continuamente diferenciável em $\mathbb{R}^n$.*
2. *$f(x)$ é limitada inferiormente em $\mathbb{R}^n$, ou seja, existe uma constante κ_{lbf} tal que, $\forall x \in \mathbb{R}^n$, $f(x) \geq \kappa_{lbf}$.*
3. *A Hessiana de f é uniformemente limitada, ou seja, existe uma constante positiva κ_{ufh} tal que, $\forall x \in \mathbb{R}^n$, $||\nabla^2 f(x)|| \leq \kappa_{ufh}$.*

Note que o item 3 da Hipótese 1 geralmente é muito forte, visto que temos apenas informações locais da função. Assim, normalmente precisamos que $||\nabla^2 f(x)||$ seja limitada para valores de x que estão entre duas iterações sucessivas do algoritmo. Esse requerimento

mais fraco é, então, satisfeito se as iterações permanecem num subconjunto limitado de $\mathbb{R}^n$, como o conjunto $\{x \in \mathbb{R}^n | f(x) \leq f(x_0)\}$ para algum x_0. Além disso, observe que podemos supor sem perda de generalidade que

$$\kappa_{ufh} \geq 1.$$

Hipótese 2. ***(Sobre o modelo)*** *O modelo m_k é tal que*

1. *Para todo k, o modelo m_k é duas vezes diferenciável em $\mathcal{B}_k$.*
2. *Para todo k, os valores da função objetivo e do modelo coincidem na iteração atual, ou seja,*
$$m_k(x_k) = f(x_k). \tag{1.12}$$
3. *Para todo k, o gradiente do modelo em x_k é igual ao gradiente da função objetivo, isto é,*
$$\nabla m_k(x_k) = \nabla f(x_k). \tag{1.13}$$
4. *A Hessiana do modelo permanece limitada na região de confiança, ou seja,*
$$||\nabla^2 m_k(x)|| \leq \kappa_{umh} - 1, \quad x \in \mathcal{B}_k \tag{1.14}$$
para todo k e onde $\kappa_{umh} \geq 1$ é uma constante que independe de k.

A principal razão para desenvolver a teoria para o caso mais geral é esclarecer a relação entre a função objetivo e o seu modelo. Além disso, a aplicabilidade mais ampla da teoria mais geral justifica a complexidade adicional.

Seguimos agora para outra parte importante da teoria de convergência, a caracterização da solução do subproblema e como ela reduzirá o modelo.

1.3.2 O ponto de Cauchy e a Redução do Modelo

Um ponto crucial do algoritmo 1.5 é a determinação de s_k que reduz "suficientemente" o modelo na região de confiança. Veremos a caracterização deste passo.

Como o modelo que utilizamos é uma função quadrática, podemos verificar o que acontece na direção de $-\nabla m_k(x_k)$, pois por um resultado do Cálculo sabemos que essa direção é a de máximo declive. Então tentaremos achar ao longo desta direção um ponto que esteja em $\mathcal{B}_k$ e que reduza o modelo satisfatoriamente. Em outras palavras, gostaríamos de calcular um mínimo de (1.7) ao longo do arco de Cauchy, definido por

$$x_k^C(t) = \{x | x = x_k - t\nabla m_k(x_k),\ t \geq 0 \text{ e } x \in \mathcal{B}_k\}.$$

Note que $x_k^C(t) = x_k$ para todo $t \geq 0$ sempre que $\nabla m_k(x_k) = 0$.

Lembrando que não temos hipóteses de $\nabla^2 f$ ser positiva definida, logo não temos garantia da convexidade do modelo, precisamos considerar a curvatura dele a cada iteração do método, então definimos

$$\beta_k = 1 + \max_{x \in \mathcal{B}_k} ||\nabla^2 m_k(x)|| \tag{1.15}$$

como um limitante superior da curvatura.

Como usamos um modelo quadrático, então podemos obter a solução exata que minimiza ele ao longo do arco de Cauchy, que é o nosso objetivo aqui. O ponto resultante, que denotaremos por $x_k^C \stackrel{\text{def}}{=} x_k - t_k^C \nabla f(x_k)$ é solução do problema

$$\min_{\substack{t \geq 0 \\ x_k - t\nabla f(x_k) \in \mathcal{B}_k}} \quad m_k(x_k - t\nabla f(x_k)) \tag{1.16}$$

onde t_k^C é o tamanho do passo que precisamos dar para chegar na solução. Chamamos x_k^C de ponto de Cauchy. A seguir veremos o resultado que nos mostra a redução que deve ocorrer no modelo avaliado no ponto de Cauchy, independente da curvatura do modelo.

Teorema 1.6. *Se o modelo é dado por (1.7) e se definimos o ponto de Cauchy por (1.16), temos que*

$$m_k(x_k) - m_k(x_k^C) \geq \frac{1}{2} ||\nabla m_k(x_k)|| \min \left\{ \frac{||\nabla m_k(x_k)||}{\beta_k}, \Delta_k \right\} \tag{1.17}$$

onde β_k é definido por (1.15).

O Teorema 1.6 nos diz que a redução obtida no ponto de Cauchy deve ser maior que uma fração da norma do gradiente no ponto atual.

Em alguns casos, determinar um minimizador exato pode ser muito caro computacionalmente. Como alternativa poderíamos utilizar algum método iterativo, por exemplo backtraking (veja [3]), para aproximar o ponto que reduz o modelo dentro da região de confiança. O ponto resultante é chamado de ponto de Cauchy aproximado e o denotamos por x_k^{AC}.

A seguir apresentamos o resultado que garante a redução do modelo no ponto de Cauchy aproximado.

Teorema 1.7. *Suponha que o modelo seja duas vezes continuamente diferenciável. Então é possível obter um ponto de Cauchy aproximado x_k^{AC} que satisfaz*

$$m_k(x_k) - m_k(x_k^{AC}) \geq \kappa_{dcp} ||\nabla m_k(x_k)|| \min \left\{ \frac{||\nabla m_k(x_k)||}{\beta_k}, \Delta_k \right\} \tag{1.18}$$

onde $\kappa_{dcp} \in (0, 1)$ é uma constante independente de k e β_k é definido por (1.15).

Observamos nos Teoremas 1.6 e 1.7 que as reduções do modelo obtidas nos pontos de Cauchy exato e aproximado têm a mesma forma. Devido a essa similaridade e pelo fato

de ambos os pontos terem o mesmo papel na teoria, deixamos de lado a distinção entre x^C e x^{AC}.

Com a possibilidade do modelo ser reduzido pelo menos tanto quanto no ponto de Cauchy, como dado em (1.17) e (1.18), é aceitável exigir do método que resolverá o subproblema uma redução do modelo da seguinte forma:

Para todo k,

$$m_k(x_k) - m_k(x_k + s_k) \geq \kappa_{mdc} ||\nabla m_k(x_k)|| \min \left\{ \frac{||\nabla m_k(x_k)||}{\beta_k}, \Delta_k \right\} \tag{1.19}$$

para alguma constante $\kappa_{mdc} \in (0, 1)$. Algumas vezes, a desigualdade (1.19) é referida como condição de Cauchy. Ela implica que a diminuição total do modelo é ao menos uma fração da obtida no ponto de Cauchy. Esta suposição tem a seguinte consequência útil.

Teorema 1.8. *Suponha que o terceiro item da Hipótese 2 e (1.19) sejam verdadeiros e que* $\nabla m_k(x_k) \neq 0$. *Então* $m_k(x_k + s_k) < m_k(x_k)$ *e* $s_k \neq 0$.

Dessa forma vemos que o valor de ρ_k, como definido em (1.8), está bem definido, desde que x_k não seja um ponto crítico de primeira ordem, ou seja,

$$\nabla f(x_k) = 0.$$

Note que (1.19) é um limitante inferior para a redução do modelo, então se decidirmos aceitar o custo dos cálculos adicionais, podemos querer resolver o subproblema (1.9) exatamente. Porém precisamos apenas achar uma aproximação para o minimizador do modelo para garantir (1.19), como veremos no resultado a seguir.

Teorema 1.9. *Seja* x_k^M *a solução de (1.9). Suponha que, para todo* k, *o passo* s_k *garante que*

$$m_k(x_k) - m_k(x_k + s_k) \geq \kappa_{amm}[m_k(x_k) - m_k(x_k^M)] \tag{1.20}$$

onde $\kappa_{amm} \in (0, 1]$ *é uma constante. Então (1.19) é verdadeira para* κ_{mdc} *definido adequadamente.*

Uma discussão mais detalhada sobre ponto de Cauchy, assim como as demonstrações dos resultados apresentados nesta subseção podem ser encontrados na Seção 6.3 de [6].

1.3.3 Convergência para Pontos Críticos de Primeira Ordem

Queremos mostrar que o algoritmo BTR é globalmente convergente para pontos críticos de primeira ordem, isto é, independente do ponto inicial escolhido e da escolha do raio da região de confiança, se as Hipóteses 1 e 2 são satisfeitas, então todos os pontos de

acumulação x_* de $\{x_k\}$ gerados pelo algoritmo são pontos críticos de primeira ordem para o problema (1.1), ou seja, eles satisfazem

$$\nabla f(x_*) = 0.$$

O primeiro passo da análise da convergência é examinar o tamanho do erro entre a função objetivo e seu modelo no novo iterado $x_k + s_k \in \mathcal{B}_k$. Para isso, utilizamos o seguinte teorema.

Teorema 1.10. *Suponha que as Hipóteses 1 e 2 sejam satisfeitas. Então temos, para todo k,*

$$|f(x_k + s_k) - m_k(x_k + s_k)| \leq \max\{\kappa_{ufh}, \kappa_{umh}\}\Delta_k^2 \tag{1.21}$$

onde $x_k + s_k \in \mathcal{B}_k$.

Observamos então que o erro entre a função objetivo e o modelo diminui quadraticamente com o raio da região de confiança, ou seja, quanto menor for o raio, melhor o modelo aproxima a função objetivo, o que intuitivamente garante que minimizar o modelo dentro de uma região de confiança suficientemente pequena também diminuirá a função objetivo, como desejado. Mostraremos através do Teorema a seguir que esta intuição é justificada.

Teorema 1.11. *Suponha que as Hipóteses 1 e 2 sejam satisfeitas e também que (1.19) seja verificada. Suponha ainda que $\nabla f(x_k) \neq 0$ e que*

$$\Delta_k \leq \frac{\kappa_{mdc}||\nabla f(x_k)||(1 - \eta_2)}{\kappa_{ubh}} \tag{1.22}$$

Então a iteração k é muito bem sucedida e

$$\Delta_{k+1} \geq \Delta_k. \tag{1.23}$$

Como uma consequência desta propriedade, podemos agora provar que o raio não pode ficar muito pequeno enquanto um ponto crítico de primeira ordem não for atingido. Esta propriedade é crucial pois assegura que o progresso do algoritmo é sempre possível, exceto em pontos críticos de primeira ordem. Ela também indica o quão pequeno o raio da região de confiança tem que ser, relativo a $||\nabla f(x_k)||$, para garantir o sucesso da iteração.

Teorema 1.12. *Suponha que as Hipóteses 1 e 2 sejam satisfeitas e também que (1.19) seja verificada. Suponha ainda que existe uma constante $\kappa_{lbg} > 0$ tal que $||\nabla f(x_k)|| \geq \kappa_{lbg}$, para todo k. Então existe uma constante $\kappa_{lbd} > 0$ tal que*

$$\Delta_k \geq \kappa_{lbd} \tag{1.24}$$

para todo k.

Os Teoremas 1.11 e 1.12 são suficientes para estabelecer a criticidade do único ponto de acumulação da sequência de iterações quando existem somente finitas iterações bem-sucedidas.

Teorema 1.13. *Suponha que as Hipóteses 1 e 2 sejam satisfeitas e que (1.19) seja verificada. Suponha ainda que há apenas um número finito de iterações bem-sucedidas. Então $x_k = x_*$ para todo k suficientemente grande e x_* é crítico de primeira ordem.*

O resultado acima (Teorema 1.13) mostra a convergência desejada para o caso onde o conjunto das iterações bem sucedidas S é finito. Agora voltamos nossa atenção ao caso em que S é infinito. Começamos por provar que pelo menos um ponto de acumulação da sequência de iterações (quando a sequência é infinita) deve ser crítico de primeira ordem. A intuição por trás deste resultado é que, se Δ_k é suficientemente pequeno, o modelo deveria aproximar bem a função, pelo Teorema 1.10, mas ao mesmo tempo Δ_k não pode ser muito pequeno se $||\nabla f(x_k)||$ for limitada por 0, como nos garante o Teorema 1.12. Assim, a convergência para um ponto crítico de primeira ordem deveria ocorrer.

Teorema 1.14. *Suponha que as Hipóteses 1 e 2 sejam satisfeitas e (1.19) seja verificada. Então temos que*

$$\lim_{k\to\infty} \inf ||\nabla f(x_k)|| = 0. \tag{1.25}$$

O Teorema 1.14 é o primeiro passo na análise de convergência de um algoritmo de região de confiança. Ele implica que se a sequência de iterações tem pontos de acumulação, então pelo menos um deles satisfaz a condição necessária de primeira ordem, ou seja,

$$\nabla f(x_*) = 0.$$

Sob as mesmas hipóteses do Teorema 1.14 é possível mostrar que

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

ou seja, todos os pontos de acumulação da sequência de iterações geradas pelo algoritmo de região de confiança são pontos críticos de primeira ordem.

O estudo da convergência do algoritmo para pontos críticos de segunda ordem não foi abordado neste trabalho, mas o leitor interessado pode seguir para as Seções 6.5 e 6.6 de [6].

Agora que vimos a teoria de convergência para o método de região de confiança, observamos que a convergência dele depende muito do método que escolhemos para resolver o subproblema. Então no capítulo a seguir estudaremos dois métodos para resolução de tal problema.

2 O Subproblema da Região de Confiança

Vimos que um ponto central do Algoritmo 1.5 é determinar a solução do subproblema de região de confiança, dado por (1.9), pois o passo definido a partir desta solução deve satisfazer algumas condições para garantir a convergência do método, como foi visto no Capítulo 1.

No algoritmo, para resolver de fato o subproblema precisamos fazer duas escolhas importantes. A primeira, é escolher a norma que definirá a região de confiança. A segunda é escolher o método que usaremos para determinar a solução do problema de minimização eficientemente e que satisfaça (1.19).

Neste capítulo apresentaremos dois métodos para a resolução do subproblema de região de confiança. Além disso, faremos a análise para problemas irrestritos e métodos que usam norma Euclidiana. Considerando essa norma, definimos a região de confiança como

$$\mathcal{B}_k = \{x \in \mathbb{R}^n | \ ||s|| \leq \Delta_k\}$$

e lembrando que o modelo considerado é quadrático podemos reescrever o subproblema como

$$\min_{||s|| \leq \Delta_k} m_k(x_k + s) = \min_{||s|| \leq \Delta_k} \left\{ f(x_k) + \langle g_k, s \rangle + \frac{1}{2}\langle s, H_k s \rangle \right\} \tag{2.1}$$

onde f é a função objetivo, g_k é o gradiente de f em x_k e H_k é a hessiana de f em x_k. A partir deste momento omitiremos k relativo a iteração do método de região de confiança para que não haver confusão com a iteração interna dos métodos para a resolução do subproblema e a matriz hessiana será denotada exclusivamente por H para não sobrecarregar a notação.

No estudo da teoria de convergência do método de região de confiança, vimos que um passo que seja solução de (2.1) deve satisfazer a condição de Cauchy tanto se for a solução exata quanto se for a solução aproximada, como mostra o Teorema 1.9. Assim, serão estudados os métodos que nos dão as soluções exata e aproximada. Porém antes de falarmos de tais métodos, faremos uma caracterização da solução do subproblema de região de confiança dado por (2.9).

2.1 A solução do subproblema

O subproblema (2.1) consiste em minimizar uma quadrática sujeita a uma restrição, a região de confiança. Assim, precisamos primeiro caracterizar um minimizador local

restrito de uma função.

Suponha que tenhamos um problema restrito do tipo

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x) \\ \text{s a} \quad & g(x) \leq 0 \end{aligned}$$

onde $g : \mathbb{R}^n \to \mathbb{R}^m$ tal que $g = (g_1, \ldots, g_m)$. Denotamos o conjunto de índices das restrições de desigualdade por $\mathcal{I}$. Dizemos que um ponto $x \in \mathbb{R}^n$ é um ponto viável se x satisfaz as restrições imposta ao problema.

Se x é um ponto viável, dizemos que a restrição de desigualdade g_i, para algum $i \in \mathcal{I}$, está ativa se $g_i(x) = 0$. Denotamos por $\mathcal{A}(x)$ o conjunto das restrições de desigualdade que estão ativas em x, ou seja,

$$\mathcal{A}(x) = \{x \mid g_i(x) = 0, i \in \mathcal{I}\}. \tag{2.2}$$

O Lagrangiano para uma função f é dado por

$$\mathcal{L}(x, \lambda) = f(x) + \sum_{i\in\mathcal{I}} \lambda_i g_i(x)$$

onde os escalares λ_i, $i \in \mathcal{I}$, são chamados de multiplicadores de Lagrange.

Em um problema com restrições, uma solução não é apenas caracterizada pelas condições impostas à função objetivo; também temos condições sobre as restrições. Essas condições são chamadas de *qualificações de restrições* e elas ajudam a garantir que excluamos casos patológicos nas restrições. Existem várias qualificações, mas usaremos apenas a qualificação a seguir.

Proposição 2.1. *(Qualificação de Independência Linear das Restrições) Sejam um ponto x_* e seu conjunto $\mathcal{A}(x_*)$ correspondente, a qualificação de restrições de independência linear é verificada se os gradientes ativos*

$$\{\nabla g_i(x_*) \mid i \in \mathcal{A}(x_*)\}$$

são linearmente independentes.

Isto é equivalente a pedir que o Jacobiano das restrições ativas em x_* tenha posto completo. Agora podemos enunciar a condição necessária de primeira ordem para problemas restritos.

Teorema 2.2. *Assuma que x_* seja uma solução local de um problema de otimização com restrições e que a condição (2.1) seja satisfeita no ponto. Então existe um vetor de*

multiplicadores de Lagrange λ^**, com componentes* λ_i^**,* $i \in \mathcal{I}$*, tal que as seguintes condições são satisfeitas:*

$$\nabla f(x_*) + J_g(x_*)^T \lambda = 0, \tag{2.3a}$$

$$g_i(x_*) \leq 0, \ \textit{para todo } i \in \mathcal{I}, \tag{2.3b}$$

$$\lambda^* \geq 0 \tag{2.3c}$$

$$\lambda_i^* g_i(x_*) = 0, \ \textit{para todo } i \in \mathcal{I}. \tag{2.3d}$$

As condições (2.3) são conhecidas também como condições Karush-Kuhn-Tucker, KKT. Note que (2.3a) pode ser reescrito como $\nabla_x \mathcal{L}(x, \lambda) = 0$.

O resultado acima foi descrito para um problema de minimização com restrições não lineares de desigualdade quaisquer. Então voltando ao nosso problema,

$$\begin{aligned} \min_{s \in \mathbb{R}^n} \quad & \left\{ m(s) = \frac{1}{2}\langle s, Hs \rangle + \langle g, s \rangle \right\} \\ \text{s a} \quad & ||s|| \leq \Delta \end{aligned}$$

Observe que temos apenas uma restrição, a região de confiança, e podemos reescrevê-la como $||s|| - \Delta \leq 0$, ou ainda, $s^T s - \Delta^2 \leq 0$. Além disso, o Lagrangiano para este problema é dado por

$$\mathcal{L}(s, \lambda) = \frac{1}{2}\langle s, Hs \rangle + \langle g, s \rangle + \lambda(s^T s - \Delta^2)$$

e derivando $\mathcal{L}(x, \lambda)$ com relação a x temos que

$$Hs + g + \lambda s = 0,$$

ou ainda, $(H + \lambda I)s = -g$ onde I é a matriz identidade. Então as condições KKT aplicadas ao nosso problema podem ser escritas como

$$(H + \lambda I)s = -g, \tag{2.4a}$$

$$||s|| - \Delta \leq 0, \tag{2.4b}$$

$$\lambda \geq 0, \tag{2.4c}$$

$$\lambda(||s|| - \Delta) = 0. \tag{2.4d}$$

Assim, a solução do subproblema de região de confiança deve satisfazer as condições (2.4). O resultado a seguir mostra exatamente isso.

Teorema 2.3. *Um ponto s^M tal que $||s^M|| \leq \Delta$ é um minimizador global do problema (2.1) se, e somente se,*

$$H(\lambda^M)s^M = -g \tag{2.5}$$

onde $H(\lambda^M) = H + \lambda^M I$ é semi definida positiva, $\lambda^M \geq 0$ e $\lambda^M(||s^M|| - \Delta) = 0$. Se $H(\lambda^M)$ é positiva definida então s^M é único.

A demonstração deste resultado pode ser vista na seção 7.2 de [6].

O Teorema 2.3 pode ser interpretado da seguinte forma, o multiplicador de Lagrange λ^M pode ser entendido como um regularizador da curvatura do modelo para que a solução do subproblema satisfaça as condições KKT. Assim, se a solução do problema estiver no interior da região de confiança temos $\lambda^M = 0$ e se ela estiver na fronteira teremos $\lambda^M \geq 0$.

Com a interpretação anterior do Teorema 2.3, temos a base para o método de Moré-Sorensen, que nos dá uma forma de obter solução exata do subproblema (2.1).

2.2 Solução exata: Método de Moré-Sorensen

O Teorema 2.3 mostra que a solução do problema pode estar no interior da região de confiança, e neste caso $\lambda^M = 0$, ou ela está na fronteira, e então $\lambda^M > 0$.

Supondo que a matriz H tenha uma decomposição em autovalores da forma $H = U^T \Lambda U$ com $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$, o Teorema 2.3 nos diz que o λ que procuramos deve satisfazer $\lambda^M \geq -\lambda_1$ (pois só então $H(\lambda)$ é semidefinida positiva) e se $\lambda^M > -\lambda_1$, então o minimizador é único (pois garante que $H_k(\lambda)$ é definida positiva).

Considerando que a solução s^M de (2.5) depende de λ^M, ou seja, λ é um parâmetro neste problema, podemos reescrever o problema da seguinte forma: determinar λ tal que

$$||s(\lambda)|| = \Delta \tag{2.6}$$

onde $s(\lambda)$ é a solução de (2.1). De (2.5) e usando a decomposição de em autovalores de H, obtemos que

$$s(\lambda) = -H(\lambda)^{-1}\nabla f(x) = -U^T(\Lambda + \lambda I)^{-1}U\nabla f(x)$$

e se aplicarmos essa equação em (2.6) e elevarmos a norma de $s(\lambda)$ ao quadrado teremos que

$$||s(\lambda)||^2 = \sum_{i=1}^{n} \frac{\omega_i}{(\lambda_i + \lambda)^2},$$

onde ω_i é a i-ésima componente do produto $U\nabla f(x)$. Assim, se λ ficar muito próximo de $-\lambda_1$ esta equação vai para o infinito, logo pode ser difícil obter uma solução nessa caso. Porém existe uma expressão alternativa chamada *equação secular* definida por

$$\phi(\lambda) = \frac{1}{||s(\lambda)||} - \frac{1}{\Delta} \tag{2.7}$$

e agora quando λ tende $-\lambda_1$ temos que a função tende a 0. E, assim, é muito melhor se quisermos aplicar um método para determinar raízes para calcular λ^M.

Um método para determinarmos a solução de (2.7) é usar o método de Newton, aproveitando que é fácil determinar a derivada de $\phi(\lambda)$. Então, em cada iteração fazemos

$$\lambda^{\text{novo}} = \lambda^{velho} - \frac{\phi(\lambda^{velho})}{\phi'(\lambda^{velho})}$$

Felizmente, $\phi(\lambda)$ e sua derivada podem ser determinadas a partir da solução de sistemas lineares envolvendo $H(\lambda)$. Como na região de interesse essa matriz é definida positiva podemos utilizar a fatoração de Cholesky $H(\lambda) = L(\lambda)L(\lambda)^T$ para obter

$$\lambda^{novo} = \lambda + \left(\frac{||s|| - \Delta}{\Delta}\right)\left(\frac{||s||^2}{||w||^2}\right) \tag{2.8}$$

onde w é solução de $L(\lambda)w = s(\lambda)$.

Além desta utilização, a fatoração de Cholesky tem outro papel importante no método de Moré-Sorensen, pois ela só existe no caso em que $\lambda > -\lambda_1$ e se $\lambda = -\lambda_1$, então a fatoração pode ou não existir, mas certamente uma ou mais entradas da diagonal de $L(\lambda)$ seriam nulas.

No entanto, o método de Newton não tem convergência garantida para qualquer ponto inicial, então precisamos ter uma salvaguarda para que o método não seja divergente. É possível mostrar que se conseguirmos achar um iterado entre $-\lambda_1$ e λ^M, então a convergência do método de Newton é garantida (veja o Lema 7.3.2 de [6]). Como o propósito do trabalho é apresentar uma versão simplificada do método, existem detalhes técnicos e alguns casos problemáticos que não estão sendo considerados neste trabalho, porém o leitor interessado pode se dirigir à Seção 7.3 de [6] para um estudo mais detalhado. Assim, construímos o *intervalo de incerteza* $\left[\lambda^L, \lambda^U\right]$, no qual é garantido que a solução irá ocorrer e que encolherá a cada iteração até que tenhamos a solução. O intervalo de incerteza inicial deve ser definido de modo que λ^M esteja dentro dele. Para tanto podemos usar o quociente de Rayleigh, dado por x^TAx/x^Tx onde $A \in \mathbb{R}^{n\times n}$ é simétrica, para $H(\lambda)^T H(\lambda)$, pois $H(\lambda)$ é simétrica, e como requeremos que $||s(\lambda)||_2 \leq \Delta$ e $H(\lambda)s(\lambda) = -g$, podemos escrever

$$\frac{||g||}{\Delta} - \lambda_n \leq \lambda \leq \frac{||g||}{\Delta} - \lambda_1$$

onde λ_1 e λ_n são o menor e o maior autovalor de H, respectivamente. Agora podemos trocar esses dois autovalores por qualquer estimativa facilmente calculável, por exemplo os círculos de Gershgorin ou as estimativas das normas matriciais $||A||_F$ e $||A||_\infty$. Para mais informações sobre o assunto recomendamos [7].

Podemos interpretar λ como a convexidade que devemos adicionar a H_k de modo que a solução seja trazida para dentro da região de confiança. Sabemos também que se $\lambda = 0$ não é solução, então s^M deve estar na fronteira da região de confiança. No entanto,

como em qualquer outro método numérico, testar se $||s(\lambda)|| = \Delta$ pode ser difícil, então testamos se $||s(\lambda)|| \in ((1-\varepsilon^{\Delta})\Delta, (1+\varepsilon^{\Delta})\Delta)$, para algum $\varepsilon^{\Delta} > 0$ definido previamente. O algoritmo é descrito a seguir.

Algoritmo 2.4. (Método de Moré-Sorensen (MS))

Passo 1: Se a fatoração de $H(0)$ for bem sucedida, resolva $H(0)s(0) = -g$ para obter $s(0)$. Se $||s(0)|| < \Delta(1+\varepsilon^{\Delta})$, então $s = s(0)$ e pare.

Passo 2: Determine um intervalo inicial $[\lambda^L, \lambda^U]$ e um λ inicial nesse intervalo.

Passo 3: Tente fazer a fatoração de Cholesky de $H(\lambda) = LL^T$. Se for bem sucedida resolva $LL^T s = -g$. Se $(1-\varepsilon^{\Delta})\Delta \leq ||s||_2 \leq (1+\varepsilon^{\Delta})\Delta)$, pare. Se não, calcule (2.8).

Passo 4: Atualize o intervalo de incerteza fazendo:

- se $||s||_2 > \Delta(1+\varepsilon^{\Delta})$ ou se a fatoração falhou, redefina $\lambda^L = \lambda$;
- se $||s||_2 < \Delta(1-\varepsilon^{\Delta})$, redefina $\lambda^U = \lambda$.

Passo 5: Escolha λ suficientemente dentro de $[\lambda^L, \lambda^U]$ e tão próximo de λ^{novo} quanto possível, se ele foi calculado. Vá para o **Passo 3**.

Na Figura 2.2 podemos ver o funcionamento do Algoritmo 2.4. No primeiro gráfico a esquerda podemos ver o modelo e que o minimizador dele está fora da região de confiança, logo o ponto procurado pelo algoritmo deve estar na fronteira da região de confiança. No gráfico central notamos que o modelo teve sua forma modificada pelo algoritmo e agora a solução se encontra no interior da região de confiança, porém sabemos que a solução procurada está na fronteira. No gráfico a direita, observamos que o método localizou a solução na fronteira, logo o algoritmo deve parar.

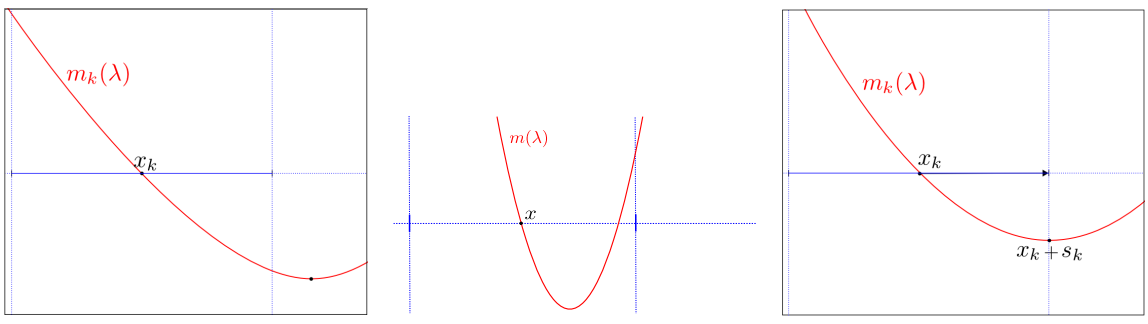


Figura 2 – Funcionamento do algoritmo de Moré-Sorensen

Há muitos detalhes que precisam ser definidos, como o valor inicial de λ e uma regra de terminação aceitável. Para um melhor detalhamento da teoria e do algoritmo recomendamos ao leitor seguir as seções 7.3.4 a 7.3.11 de [6].

2.3 Solução aproximada: Método de Gradiente Conjugado Truncado (Steihaug-Toint)

Na seção anterior estudamos o método que nos dá a solução exata para o subproblema (2.1), porém nele precisamos fazer a fatoração de Cholesky de H e sabemos que esse cálculo nem sempre é barato computacionalmente.

Lembrando o Teorema 1.9, vimos que a solução aproximada de (2.1) pode ser uma boa alternativa. Então veremos a seguir um dos métodos mais utilizados para obter a solução aproximada: o método de gradiente conjugado truncado. Primeiramente veremos sua base teórica e depois como ele se encaixa na resolução de (2.1).

2.3.1 Método de Gradiente Conjugado

Para que possamos entender o funcionamento do método que é o foco desta seção, antes precisamos estudar a sua teoria, para tanto vamos focar em um problema diferente nessa subseção.

Estamos interessados em resolver um problema particular de otimização: minimizar uma função quadrática (estritamente) convexa, ou seja,

$$\min_{x\in\mathbb{R}^n} q(x) = \min_{x\in\mathbb{R}^n} \frac{1}{2}\langle x, Hx\rangle + \langle c, x\rangle \tag{2.9}$$

onde $H \in \mathbb{R}^{n\times n}$ é uma matriz simétrica semi definida positiva e $c \in \mathbb{R}^n$.

Sabemos que se H for definida positiva, então das condições de otimalidade vistas na Seção 1.1, o problema (2.9) é equivalente a determinar a solução de $\nabla q(x_*) = 0$, ou seja,

$$Hx_* = -c$$

A solução exata deste sistema pode ser obtida diretamente fazendo-se a fatoração de H, porém estamos interessados em problemas grandes, logo a fatoração pode se tornar cara. Deste modo, buscamos métodos iterativos que aproximem satisfatoriamente a solução.

O Método de Gradiente Conjugado se baseia no fato de que a função q pode ser minimizada em n passos se a minimizarmos sucessivamente num certo conjunto de direções, chamadas de direções conjugadas. Um conjunto $\{p_0, \ldots, p_m\}$ é dito ser conjugado com respeito a uma matriz H simétrica e definida positiva se

$$p_i^T H p_j = 0, \qquad \text{para todo} \qquad i \neq j$$

Direções conjugadas são úteis como uma maneira de gerar aproximações do minimizador de q. Definimos então o método de gradiente conjugado.

Algoritmo 2.5. (Método do Gradiente Conjugado)

Dado x_0, defina $g_0 = Hx_0 + c$ e $p_0 = -g_0$. Para $k = 0, 1, \ldots$ até a convergência

$$\begin{aligned} \alpha_k &= \frac{||g_k||_2^2}{\langle p_k, Hp_k \rangle} \\ x_{k+1} &= x_k + \alpha_k p_k \\ g_{k+1} &= g_k + \alpha_k H p_k \\ \beta_k &= \frac{||g_{k+1}||_2^2}{||g_k||_2^2} \\ p_{k+1} &= -g_{k+1} + \beta_k p_k \end{aligned}$$

O método de Gradiente Conjugado é equivalente a minimizar q sucessivamente no espaço de Krylov de H definido por

$$\mathcal{K}(H, g_0, j) = \text{span}\{g_0, \ldots, g_j\} = \text{span}\{g_0, Hg_0, \ldots, H^j g_0\} \tag{2.10}$$

Como dito anteriormente, o método de gradiente conjugado converge para a solução em até n iterações. Mas em certos casos a convergência pode ocorrer antes, por exemplo quando a matriz H tem aglomerados de autovalores. O resultado a seguir nos mostra o que esperar do erro cometido a cada iteração.

Teorema 2.6. *O erro $\varepsilon_k = x_k - x_*$ das iterações geradas pelo método do gradiente conjugado satisfaz a desigualdade*

$$||\varepsilon_k||_H \leq 2 \left(\frac{\sqrt{\kappa(H)} - 1}{\sqrt{\kappa(H)} + 1} \right)^k ||\varepsilon_0||_H \tag{2.11}$$

onde $\kappa(H)$ é o número de condição espectral de H.

Vimos no Teorema 2.6 que o comportamento da convergência do método do gradiente conjugado depende fortemente do condicionamento de H, logo quanto maior o número de condição mais lenta ela será. Uma maneira de aumentar a velocidade de convergência do método é melhorar o condicionamento da matriz H. Este é o objetivo do pré-condicionamento.

Seja $R \in \mathbb{R}^{n \times n}$ uma matriz não singular. Considere o problema (2.9)

$$\min_{\bar{x} \in \mathbb{R}^n} \frac{1}{2} \langle \bar{x}, \bar{H}\bar{x} \rangle + \langle \bar{c}, \bar{x} \rangle \tag{2.12}$$

onde $\bar{x} = Rx$, $\bar{H} = R^{-T} H R^{-1}$ e $\bar{c} = R^{-T} c$.

Agora teremos que resolver o seguinte problema:

$$(R^{-T} H R^{-1}) Rx = -R^{-T} c \tag{2.13}$$

e a solução é dada por

$$x_* = -(R^{-T}HR^{-1}R)^{-1}G^{-T}c = -H^{-1}c$$

ou seja, a solução para o problema pré-condicionado é a mesma que a solução original.

Se conseguirmos encontrar uma matriz R tal que o número de condição de $\bar{H}$ seja menor que o número de condição de H, então podemos aplicar o método de gradiente conjugado para o sistema (2.13) e termos uma convergência mais rápida. No entanto, não queremos a forma explícita de $\bar{H}$ e $\bar{c}$, pois a matriz transformada pode ser densa, enquanto H é esparsa. O algoritmo apresentado a seguir não usa explicitamente R mas $M = R^T R$, que é simétrica e definida positiva.

Algoritmo 2.7. (Método do Gradiente Conjugado Pré-Condicionado)

Dado x_0, defina $g_0 = Hx_0 + c$ e seja $v_0 = M^{-1}g_0$ e $p_0 = -v_0$. Para $k = 0, 1, \ldots$ até a convergência, faça

$$\alpha_k = \frac{\langle g_k, v_k \rangle}{\langle p_k, Hp_k \rangle}$$

$$x_{k+1} = x_k + \alpha_k p_k$$

$$g_{k+1} = g_k + \alpha_k H p_k$$

$$v_{k+1} = M^{-1} g_{k+1}$$

$$\beta_k = \frac{\langle g_{k+1}, v_{k+1} \rangle}{\langle g_k, v_k \rangle}$$

$$p_{k+1} = -v_{k+1} + \beta_k p_k$$

É importante notar que os autovalores de $M^{-1}H$ são os mesmos de $\bar{H} = R^{-T}HR^{-1}$ e, então, qualquer tentativa de influenciar os autovalores de $\bar{H}$ pode ser obtido diretamente usando M.

Comparando os Algoritmos 2.5 e 2.7, notamos que o segundo é mais caro computacionalmente que o primeiro, porém se escolhermos um bom pré-condicionador poderemos reduzir o número de iterações necessário para obter o nível de precisão desejado na solução. Logo, o custo do método pré-condicionado pode ser menor, porém dependemos justamente de escolher um pré-condicionador que melhore as propriedades espectrais da matriz original e que não seja caro computacionalmente.

Estudamos também o método de Lanczos para a resolução de (2.9), mas como pode ser visto na Seção 5.2 em [6], esse método e o de gradiente conjugado tem uma relação próxima, então estudamos apenas o método de gradiente conjugado.

Veremos na subseção a seguir como podemos aplicar o método de gradiente conjugado para o subproblema que estamos interessados em resolver.

2.3.2 Método de Gradiente Conjugado Truncado (Steihaug-Toint)

Vimos que o método de gradiente conjugado é utilizado para minimizar funções quadráticas (estritamente) convexas. Entretanto, apesar de trabalharmos com funções quadráticas no algoritmo de região de confiança, não temos mais a garantia de convexidade.

Como estamos no caso em que a região de confiança é definida pela norma Euclidiana, é possível obter uma generalização do método de gradiente conjugado. Além disso, na subseção anterior vimos que a versão pré-condicionada do método é mais geral, então apresentaremos a versão pré-condicionada deste método.

Assumindo que a matriz $M \in \mathbb{R}^{n\times n}$, simétrica e definida positiva, seja o pré-condicionador escolhido para resolver o problema, se este não for o caso basta considerar M como sendo a matriz identidade. Para o caso geral, o problema pré-condicionado é equivalente ao subproblema de região de confiança definido pela norma M, ou seja,

$$\min_{||s||_M \leq \Delta} m(s) = \min_{||s||_M \leq \Delta} \frac{1}{2}\langle s, Hs\rangle + \langle g, s\rangle.$$

Aplicando o método de gradiente conjugado a esse problema, três casos podem ocorrer:

1. $\langle p_k, Hp_k\rangle > 0$, para toda iteração k;
2. $\langle p_k, Hp_k\rangle \leq 0$, para alguma iteração k;
3. A iteração s_k não está na região de confiança, para alguma iteração k.

Se o primeiro caso ocorre, então o modelo m é convexo e todo s_k permanece no interior da região de confiança, logo é suficiente encontrar o minimizador irrestrito de $m(s)$. Se o segundo caso acontece, temos que m não é limitado inferiormente na direção αp_k, então nosso objetivo é minimizar ao longo de $s_k + \alpha p_k$ o modelo tanto quanto possível e permanecer na região de confiança. Fazemos isso trocando o α obtido via Algoritmo 2.7 pela raiz positiva de

$$||s_k + \alpha p_k||_M = \Delta. \tag{2.14}$$

Se ocorrer o terceiro caso, não sabemos ao certo como proceder, pois as iterações sucessoras poderiam voltar para dentro da região de confiança e, assim, interferir com a iteração atual, podendo "estragar" a possível convergência do método. O resultado a seguir mostra que esse não é o caso.

Teorema 2.8. *Suponha que o Algoritmo 2.7 é aplicado para minimizar $m(s)$ partindo do ponto inicial $s_0 = 0$ e que $\langle p_i, Hp_i\rangle > 0$ para $0 \geq i \geq k$. Então a iteração s_j satisfaz a desigualdade*

$$||s_j||_M < ||s_{j+1}||_M$$

para $0 \geq j \geq k-1$.

O resultado nos mostra que se alguma iteração do método de gradiente conjugado pré-condicionado sair da região de confiança então não vale a pena seguir nessa direção. Assim, podemos determinar um minimizador restrito local de $m(s)$ procurando a interseção entre a linha definida por $s_k + \alpha p_k$ e a fronteira da região de confiança, ou seja, resolver (2.14) como fizemos para o caso em que a curvatura é negativa.

A seguir apresentamos a base para o método de gradiente conjugado truncado de Steihaug-Toint.

Algoritmo 2.9. (Método de Gradiente Conjugado Truncado (Steihaug-Toint))

Defina $s_0 = 0$, $g_0 = g$, $v_0 = M^{-1}g_0$ e $p_0 = -v_0$. Para $k = 0, 1, \ldots$ até a convergência, faça:

Passo 1: Defina $\kappa_k = \langle p_k, Hp_k \rangle$.

Passo 2: Se $\kappa_k \leq 0$, calcule σ_k como a raiz positiva de $||s_k + \sigma p_k||_M = \Delta$, defina $s_{k+1} = s_k + \sigma_k p_k$ e pare.

Passo 3: Defina $\alpha_k = \dfrac{\langle g_k, v_k \rangle}{\kappa_k}$.

Passo 4: Se $||s_k + \alpha_k p_k||_M \geq \Delta$, calcule σ_k como a raiz positiva de $||s_k + \sigma p_k||_M = \Delta$, defina $s_{k+1} = s_k + \sigma_k p_k$ e pare.

Passo 5: Defina:

$$\begin{aligned} s_{k+1} &= s_k + \alpha_k p_k, \\ g_{k+1} &= g_k + \alpha_k H p_k, \\ v_{k+1} &= M^{-1} g_{k+1}, \\ \beta_k &= \frac{\langle g_{k+1}, v_{k+1} \rangle}{\langle g_k, v_k \rangle}, \\ p_{k+1} &= -v_{k+1} + \beta_k p_k. \end{aligned}$$

Note que se a curvatura do modelo é negativa ou se a iteração está fora da região de confiança, paramos o algoritmo imediatamente e tomamos a iteração atual como solução. Porém não temos um critério de parada caso a solução de (2.1) esteja no interior da região de confiança. Uma opção de parada do algoritmo é testar se a norma do gradiente na iteração k foi reduzida a uma fração da norma do gradiente inicial.

Tendo estudado dois métodos para determinar as soluções exata e aproximada do subproblema de região de confiança, veremos a seguir os resultados numéricos obtidos para cada um deles.

3 Resultados

Apresentaremos agora os resultados numéricos obtidos aplicando o método de região de confiança, descrito pelo Algoritmo 1.5, para minimizar um conjunto de problemas de otimização irrestrita, onde utilizamos os dois métodos estudados nas seções 2.2 e 2.3 para a resolução do subproblema da região de confiança. O primeiro método utilizado foi o de Moré-Sorensen (MS) que é descrito no Algoritmo 2.4 e determina a solução exata de (2.1). O segundo método testado foi o de gradiente conjugado truncado (GCT) descrito no Algoritmo 2.9 e que aproxima a solução do subproblema de região de confiança.

Sabemos que no método MS é necessário fazer uma ou mais fatorações de Cholesky, o que pode ser computacionalmente caro para problemas de grande porte. Enquanto no método CGT o maior custo é o produto matriz-vetor. Assim, para problemas bem condicionados é esperado que o segundo método apresente resultados melhores quanto a número de iterações e tempo computacional.

As implementações dos algoritmos de região de confiança, Moré-Sorensen e gradiente conjugado truncado foram feitas no software Matlab R2015a em um computador com processador Intel Core i5 quad-core com 1,7 GHz e 6 Gbytes de memória RAM. Além disso, para os testes do algoritmo básico de região de confiança foram utilizados os seguintes parâmetros

$$\varepsilon = 10^{-8}, \quad t_{\text{máx}} = 1800 \text{ s}, \quad N_{\text{máx}} = 2000, \quad \eta_1 = 0.1, \quad \eta_2 = 0.9 \quad \text{e} \quad \Delta_0 = 1,$$

onde ε é a precisão, $t_{\text{máx}}$ é o tempo máximo de execução do algoritmo, $N_{\text{máx}}$ é o número máximo de iterações, η_1 e η_2 são os parâmetros que definem se o novo ponto obtido através da resolução do subproblema será bem sucedido ou muito bem sucedido e Δ_0 é o raio inicial da região de confiança.

Na implementação do método de Moré-Sorensen é necessário fazer a fatoração de Cholesky, para tanto foi utilizado a função interna **chol** do próprio Matlab. Apenas um parâmetro foi utilizado na implementação, a precisão da solução, dada por

$$\varepsilon_{MS} = 10^{-12}$$

Na implementação do método de gradiente conjugado truncado utilizamos como pré-condicionador a matriz identidade, pois o objetivo era testar com um grande número de problemas. O único parâmetro utilizado foi a precisão da solução, dada por

$$\varepsilon_{GCT} = 10^{-9}.$$

Os algoritmos foram testados usando a coleção de problemas de otimização CUTEst, considerando apenas os irrestritos (veja [8] para maiores detalhes). Ao todo foram 173 problemas testados.

Foi elaborada uma tabela, apresentada no Apêndice A (Tabela 1), na qual apresentamos os números de iterações e os tempos de execução do algoritmo de região de confiança com a resolução do subproblema pelos métodos de Moré-Sorensen e de gradiente conjugado truncado, respectivamente. Considerando os resultados presentes nessa tabela com relação aos tempos de execução, podemos fazer a comparação das duas implementações do algoritmo BTR, uma obtendo a solução por MS e outra por GCT. Para tanto vamos fazer o perfil de desempenho para esses dois métodos usando como base a teoria vista em [9].

Na Figura 3 podemos notar que os dois métodos comparados tiveram um desempenho bastante parecido, com uma baixa probabilidade de sucesso na resolução de problemas. O final do gráfico nos mostra que BTR-MS resolveu o maior número de problemas.

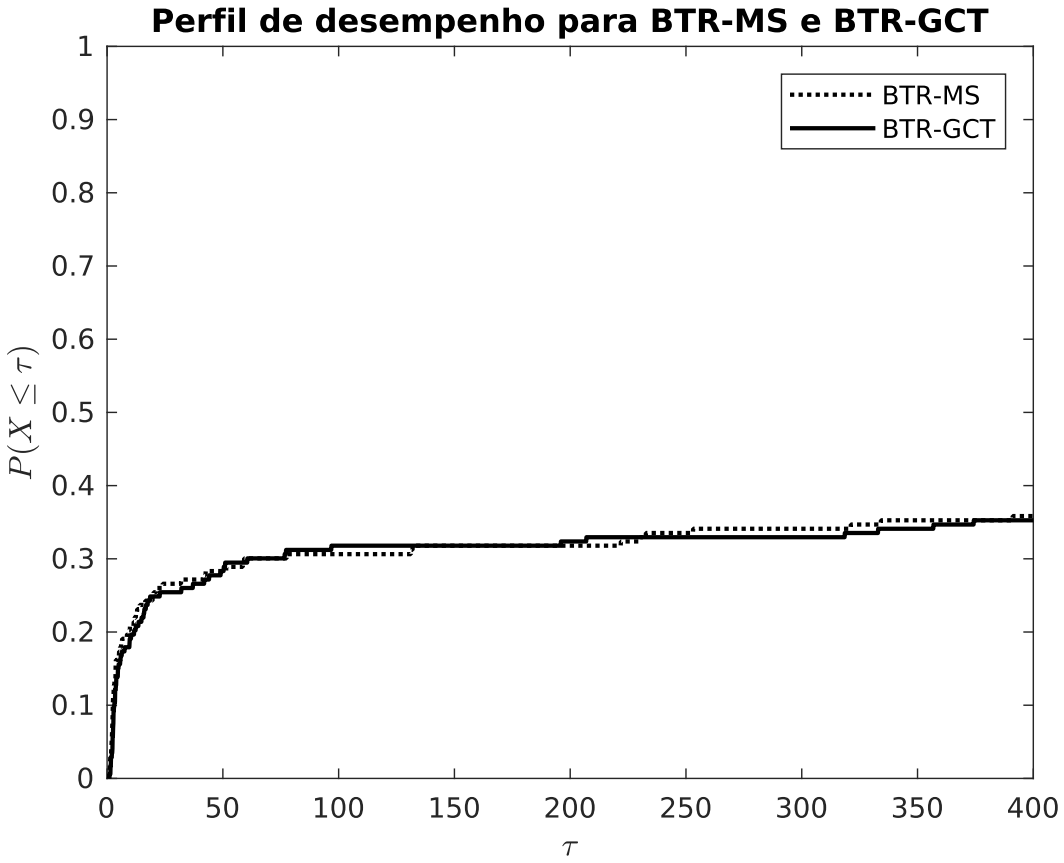


Figura 3 – Perfil de desempenho

A Figura 4 representa uma visão ampliada do trecho inicial do gráfico de perfil de desempenho mostrado na Figura 3. Observamos que o método BTR-MS foi mais rápido que o BTR-GCT para resolver a maioria dos problemas, mostrando que problemas mal

condicionados afetam o desempenho do método GCT.

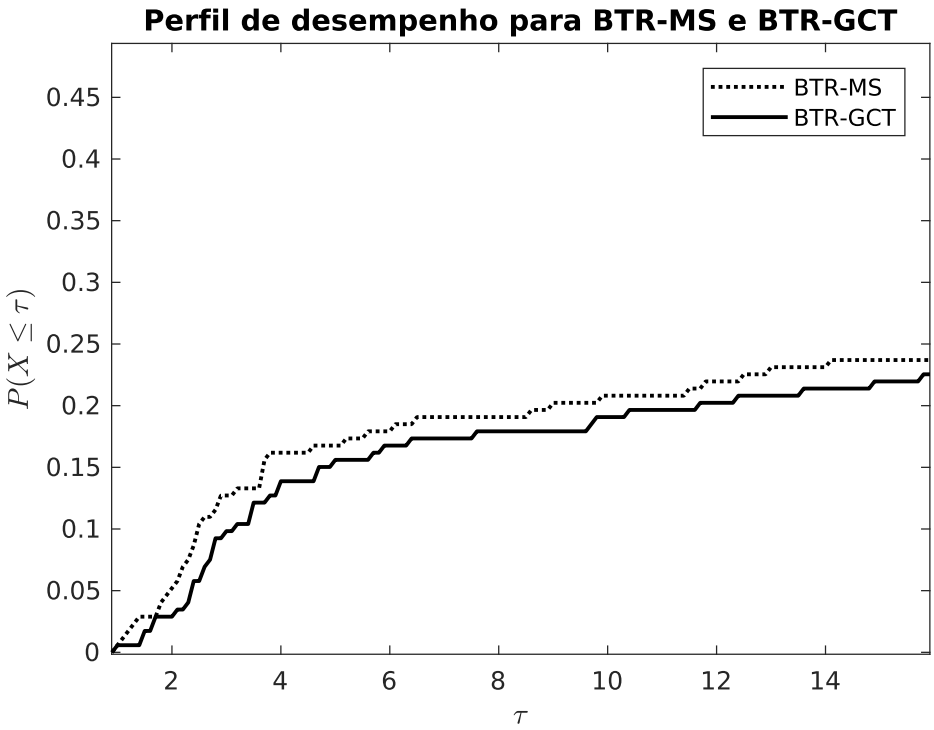


Figura 4 – Visão ampliada da parte inicial do gráfico de perfil de desempenho

Na Tabela 2, apresentada no Apêndice A, mostramos o número de fatorações de Cholesky necessárias até que a solução fosse encontrada, o número máximo de iterações fosse atingido ou o tempo limite alcançado. Com os dados dessa tabela e sabendo que o custo de cada fatoração de Cholesky é da ordem de $n^3/6$ temos uma ideia do custo computacional para resolver cada problema. Se tivermos problemas com muitas variáveis ou a matriz hessiana do problema for densa, é necessário refletir se compensa arcarmos com o custo computacional do método.

Na Tabela 3, localizada no Apêndice A, expomos os dados referentes a quantidade de iterações do método de gradiente conjugado truncado até que a solução fosse obtida, o número máximo de iterações atingido ou tempo esgotado. O custo máximo do método GCT é o de um produto matriz-vetor, que é da ordem de n^2. Para problemas pequenos ou em que a matriz é esparsa ou possui aglomerados de autovalores este custo não é significativo. Por outro lado, para problemas com muitas variáveis ou com características não favoráveis, o custo pode ser maior do que estamos dispostos a aceitar.

Neste capítulo discutimos a implementação dos métodos estudados nos Capítulos 1 e 2, apresentamos os parâmetros utilizados nos testes e por fim expomos e analisamos os resultados numéricos obtidos. Primeiro comparamos o algoritmo BTR associado aos métodos de Moré-Sorensen e gradiente conjugado truncado com relação ao tempo, para tanto analisamos o gráfico do perfil de desempenho. Depois, apresentamos na Tabela 2 os

resultados relacionados ao método de Moré-Sorensen, como o número de fatorações de Cholesky realizadas para resolução do problema. Finalmente, na Tabela 3 apresentamos os resultados relacionados ao método de gradiente conjugado truncado, falando brevemente sobre o custo computacional.

Os algoritmos tiveram uma implementação bastante simples para que o funcionamento deles fosse bem compreendido, assim para o futuro buscaremos refinar a implementação.

4 Conclusão

Neste trabalho apresentamos o método de região de confiança para a resolução de problemas de otimização irrestrita. Estudamos sua teoria de convergência e dois métodos para a resolução do subproblema interno do algoritmo de região de confiança.

No Capítulo 1, primeiro apresentamos o problema de otimização irrestrito e analisamos quais propriedades a sua solução deve ter. Depois apresentamos o método de região de confiança e estudamos a sua teoria de convergência, discutindo as hipóteses necessárias, quais condições o passo deve satisfazer para que haja um decréscimo na função e, por fim, estudamos os teoremas de convergência.

No capítulo 2, primeiramente apresentemos o subproblema de região de confiança e fizemos uma caracterização da sua solução, observando quais condições ela deve satisfazer. Depois apresentamos os métodos de Moré-Sorensen e de gradiente conjugado truncado para a resolução do subproblema desenvolvendo a teoria e depois apresentando o algoritmo.

No capítulo 3, apresentemos os resultados numéricos provenientes das implementações dos métodos estudados. Fizemos também uma comparação com relação ao tempo do algoritmo de regição de confiança resolvendo o subproblema por Moré-Sorensen e gradiente conjugado truncado.

Para o futuro pretendemos seguir estudando o método de região de confiança e os métodos que resolvam o subproblema, além disso procuramos melhorar a implementação dos métodos já feitos.

Referências

1 STEWART, J. *Calculo*. 7. ed. [S.l.]: Cengage, 2013. v. 1. Citado na página 21.

2 STRANG, G. *Linear algebra and its applications*. 4ed.. ed. [S.l.]: Brooks, 2005. Citado na página 21.

3 NOCEDAL, J.; WRIGHT, S. *Numerical Optimization*. 2nd ed. ed. [S.l.]: Springer, 2006. (Springer series in operations research). Citado na página 30.

4 SHEWCHUK, J. R. *An introduction to the conjugate gradient method without the agonizing pain*. [S.l.]: Carnegie-Mellon University. Department of Computer Science, 1994. Nenhuma citação no texto.

5 MENDONCA, M. W. *Multilevel Optimization: convergence theory, algorithms and application to derivative-free optimization*. Tese (Doutorado) — Phd thesis, Facultés Universitaires Notre-Dame de la Paix, Namur, Belgium, 2009. Nenhuma citação no texto.

6 CONN, A. R.; GOULD, N. I. M.; TOINT, P. L. *Trust-region methods*. [S.l.]: SIAM, 1987. (MPS-SIAM series on optimization). Citado 7 vezes nas páginas 28, 31, 33, 38, 39, 40 e 43.

7 GOLUB, G. H.; VAN LOAN, C. F. *Matrix computations*. [S.l.]: JHU Press, 2012. v. 3. Citado na página 39.

8 GOULD, N. I.; ORBAN, D.; TOINT, P. L. Cutest: a constrained and unconstrained testing environment with safe threads for mathematical optimization. *Computational Optimization and Applications*, Springer, v. 60, n. 3, p. 545–557, 2015. Citado na página 48.

9 DOLAN, E. D.; MORÉ, J. J. Benchmarking optimization software with performance profiles. *Mathematical programming*, Springer, v. 91, n. 2, p. 201–213, 2002. Citado na página 48.

Apêndices

APÊNDICE A – Tabelas de desempenho

Neste apêndice são apresentadas algumas tabelas mostrando o desempenho dos métodos discutidos no Capítulo 2. A Tabela 1 apresenta o número de iterações necessárias e os tempos de execução do algoritmo de região de confiança com a resolução do subproblema pelos métodos MS e GCT. Já a Tabela 2 apresenta o número de fatorações de Cholesky feito pelo método de Moré-Sorensen e a Tabela 3, por sua vez, apresenta o número de iterações feito pelo método GCT.

Tabela 1 – Desempenho BTR-MS e BTR-GCT

Problema	Nº de Variáveis	Iterações BTR-MS	CPU BTR-MS	Iterações BTR-GCT	CPU BTR-GCT
AKIVA	2	6	0.164310	6	0.128555
ALLINITU	4	13	0.011746	6	0.001903
ARGLINA	200	4	0.783860	4	0.738763
ARGLINB	200	2000	294.652332	2000	324.361689
ARGLINC	200	2000	282.657507	2000	297.149626
ARWHEAD	5000	6	5.913859	6	1.588243
BARD	3	8	0.001849	12	0.002429
BDQRTIC	5000	1094	1800.502519	2000	213.529782
BEALE	2	9	0.002640	8	0.001534
BIGGS6	6	1086	0.158426	2000	0.418909
BOX3	3	8	0.001742	8	0.001688
BOXPOWER	10000	20	146.359269	22	43.663266
BOX	10000	11	90.792924	9	12.701320
BRKMCC	2	3	0.014324	3	0.393806
BROWNAL	200	10	0.867870	31	3.219427
BROWNBS	2	22	0.003276	33	1.824684
BROWNDEN	4	11	0.002001	2000	0.423201
BROYDN7D	5000	451	1800.485385	401	181.059317
BRYBND	5000	33	247.101903	31	75.210916
CHAINWOO	4000	184	167.299024	2000	1421.954578

Continua na próxima página

Tabela 1 – continuação da página anterior

Problema	Nº de Variáveis	Iterações BTR-MS	CPU BTR-MS	Iterações BTR-GCT	CPU BTR-GCT
CHNROSNB	50	82	0.017088	75	0.033427
CHNRSNBM	50	107	0.022819	80	0.033055
CLIFF	2	27	0.004325	27	0.004167
COSINE	10000	310	1804.686048	22	31.184252
CRAGGLVY	5000	20	17.267174	18	68.661237
CUBE	2	46	0.006116	44	0.006304
CURLY10	10000	23	265.930229	15	2174.685874
CURLY20	10000	298	1801.766591	13	2140.551196
CURLY30	10000	279	1802.612625	15	2461.061213
DECONVU	63	858	0.488732	59	0.050311
DENSCHNA	2	6	0.001218	6	0.001489
DENSCHNB	2	6	0.001310	6	0.001558
DENSCHNC	2	10	0.001674	10	0.003034
DENSCHND	3	51	0.007015	44	0.006389
DENSCHNE	3	43	0.006409	10	0.002282
DENSCHNF	2	6	0.001225	6	0.001650
DIXMAANA	3000	10	2.530375	9	0.549936
DIXMAANB	3000	11	5.293496	12	2.969891
DIXMAANC	3000	12	5.528627	21	2.338900
DIXMAAND	3000	13	5.806432	2000	52.962720
DIXMAANE	3000	20	10.897646	13	25.260478
DIXMAANF	3000	23	21.133335	36	26.821715
DIXMAANG	3000	36	35.035053	45	36.170013
DIXMAANH	3000	40	46.627775	37	20.030588
DIXMAANI	3000	24	7.121404	32	764.863465
DIXMAANJ	3000	43	57.175233	82	491.855225
DIXMAANK	3000	43	46.547539	150	628.538945
DIXMAANL	3000	39	59.144870	114	392.098212
DIXMAANM	3000	10	2.800250	21	697.043322
DIXMAANN	3000	54	91.975947	43	464.465867
DIXMAANO	3000	63	108.698930	68	600.011979
DIXMAANP	3000	61	56.400976	76	485.044622
DIXON3DQ	10000	5	72.618397	6	2156.875490

Continua na próxima página

Tabela 1 – continuação da página anterior

Problema	Nº de Variáveis	Iterações BTR-MS	CPU BTR-MS	Iterações BTR-GCT	CPU BTR-GCT
DJTL	2	2000	0.298584	2000	0.435714
DQDRTIC	5000	5	4.510475	5	0.734384
DQRTIC	5000	51	123.403294	22	1875.710211
EDENSCH	2000	18	2.330752	28	2.757637
EG2	1000	2000	35.118381	3	0.021035
EIGENALS	2550	131	625.753455	89	264.913240
EIGENBLS	2550	1950	1800.300967	243	1800.438119
EIGENCLS	2652	1610	1800.140827	674	1816.884891
ENGVAL1	5000	13	11.355214	13	10.115130
ENGVAL2	3	17	0.002629	17	0.003020
ERRINROS	50	475	0.093711	57	0.024302
ERRINRSM	50	826	0.180658	2000	0.362604
EXPFIT	2	11	0.001979	12	0.002062
EXTROSNB	1000	2000	63.927356	2000	89.131528
FLETBV3M	5000	99	164.025879	54	7.235339
FLETCBV2	5000	3	5.838961	12	1723.477300
FLETCBV3	5000	1328	1800.539715	2000	208.857179
FLETCHBV	5000	1301	1800.389639	2000	215.905259
FLETCHCR	1000	2000	41.335856	2000	196.745599
FMINSRF2	5625	102	378.563130	1015	297.405316
FMINSURF	5625	101	455.073852	1542	1527.509157
FREUROTH	5000	2000	1751.640346	2000	158.217047
GENHUMPS	5000	1203	1800.232560	2000	197.492801
GENROSE	500	548	3.566104	506	2.129982
GROWTHLS	3	2000	0.312706	2000	0.327427
GULF	3	228	0.094200	34	0.028889
HAIRY	2	54	0.008888	70	0.010331
HATFLDD	3	24	0.003943	21	0.003714
HATFLDE	3	19	0.003668	14	0.002617
HATFLDFL	3	2000	0.327941	351	0.050705
HEART6LS	6	2000	0.288732	707	0.135763
HEART8LS	8	196	0.030409	166	0.032157
HELIX	3	12	0.002236	22	0.003845

Continua na próxima página

Tabela 1 – continuação da página anterior

Problema	Nº de Variáveis	Iterações BTR-MS	CPU BTR-MS	Iterações BTR-GCT	CPU BTR-GCT
HIELOW	3	8	0.055370	9	0.063536
HILBERTA	2	2	0.000719	2	0.000965
HILBERTB	10	3	0.000926	3	0.000960
HIMMELBB	2	43	0.008155	20	0.003269
HIMMELBF	4	242	0.042204	2000	0.262755
HIMMELBG	2	7	0.001654	8	0.001542
HIMMELBH	2	2	0.000714	5	0.001072
HUMPS	2	2000	0.312225	2000	0.277205
HYDC20LS	99	2000	1.881117	2000	52.734634
INDEFM	10000	22	472.374403	2000	721.531750
INDEF	5000	437	1800.206381	2000	232.711334
JENSMP	2	2000	0.238555	2000	0.208994
JIMACK	3549	39	59.316596	28	1852.919899
KOWOSB	4	15	0.002624	12	0.002600
LIARWHD	5000	16	15.425547	16	4.008916
LOGHAIRY	2	2000	0.301543	2000	0.278674
MANCINO	100	2000	43.011019	2000	43.359610
MARATOSB	2	2000	0.292226	2000	0.234099
MEXHAT	2	2000	0.229277	61	0.008087
MEYER3	3	2000	0.337802	2000	0.308739
MODBEALE	20000	4	1953.124693	8	2129.411984
MOREBV	5000	1	4.645096	9	1801.106194
MSQRTALS	1024	389	58.596210	42	84.749063
MSQRTBLS	1024	760	89.421295	37	39.108532
NCB20B	5000	1943	1800.446942	2000	1142.371594
NCB20	5010	153	364.348102	2000	441.088257
NONCVXU2	5000	705	1801.503708	2000	313.526440
NONCVXUN	5000	940	1800.721421	2000	313.595911
NONDIA	5000	6	22.748585	6	1.566522
NONDQUAR	5000	24	24.203663	14	1914.327616
NONMSQRT	4900	959	1802.016844	59	1802.132138
OSBORNEA	5	160	0.166144	2000	0.564592
OSBORNEB	11	21	0.008379	19	0.008920

Continua na próxima página

Tabela 1 – continuação da página anterior

Problema	Nº de Variáveis	Iterações BTR-MS	CPU BTR-MS	Iterações BTR-GCT	CPU BTR-GCT
OSCIGRAD	10000	9	2244.731098	2000	906.039668
OSCIPATH	10	2000	0.279227	2000	0.437281
PALMER1C	8	92	0.014004	2000	0.336104
PALMER1D	7	5	0.001732	6	0.001819
PALMER2C	8	4	0.001561	58	0.010516
PALMER3C	8	4	0.001624	5	0.001814
PALMER4C	8	5	0.001716	5	0.001809
PALMER5C	6	4	0.000991	4	0.001093
PALMER6C	8	5	0.001434	5	0.001730
PALMER7C	8	8	0.002019	2000	0.368632
PALMER8C	8	5	0.001538	6	0.004959
PARKCH	15	2000	1502.004598	19	14.247888
PENALTY1	1000	108	3.854332	107	2.788489
PENALTY2	200	2000	3.252780	2000	3.942219
PENALTY3	200	2000	79.311693	2000	82.414636
POWELLSG	5000	27	24.047322	27	10.404595
POWER	10000	46	395.232079	11	2011.878427
QUARTC	5000	51	123.980660	22	1952.682760
ROSENBR	2	33	0.004571	34	0.006803
S308	2	14	0.002653	12	0.002250
SBRYBND	5000	1902	1800.418766	1274	1800.049665
SCHMVETT	5000	2000	1787.075427	1232	1801.140045
SCOSINE	5000	292	1804.750658	2000	898.037354
SCURLY10	10000	19	1827.802744	126	1839.571083
SCURLY20	10000	33	1807.712078	96	1822.531121
sCURLY30	10000	22	1812.041114	96	1833.563358
SENSORS	100	22	0.310453	18	0.245522
SINEVAL	2	78	0.010019	71	0.009721
SINQUAD	5000	2000	1785.186121	79	17.511303
SISSER	2	18	0.002607	18	0.002272
SNAIL	2	118	0.015180	105	0.014951
SPARSINE	5000	41	469.912824	52	1965.778652
SPARSQUR	10000	30	171.784362	30	458.775025

Continua na próxima página

Tabela 1 – continuação da página anterior

Problema	Nº de Variáveis	Iterações BTR-MS	CPU BTR-MS	Iterações BTR-GCT	CPU BTR-GCT
SPMSRTLS	4999	35	65.403381	25	81.208630
SROSENBR	5000	8	7.207270	8	1.342417
SSBRYBND	5000	35	393.115376	1441	1800.877567
SSCOSINE	5000	1024	1800.755518	2000	380.987853
STRATEC	10	78	24.771838	2000	638.587999
TESTQUAD	5000	4	3.496648	5	204.788641
TOINTGOR	50	2000	0.352600	11	0.007619
TOINTGSS	5000	8	8.047696	8	5.202585
TOINTPSP	50	2000	0.330785	55	0.012143
TOINTQOR	50	3	0.001361	3	0.001351
TQUARTIC	5000	28	45.883132	10	2.182503
TRIDIA	5000	4	5.105649	4	84.955434
VARDIM	200	30	0.041872	134	0.218422
VAREIGVL	50	17	1.241897	24	0.011516
VIBRBEAM	8	102	0.029701	131	0.039714
WATSON	12	27	0.009269	59	0.027493
WOODS	4000	73	54.742976	68	16.203550
YATP1LS	2600	28	15.199042	40	3.383031
YATP2LS	2600	64	35.194680	1114	1832.056259
YFITU	3	244	0.036715	65	0.011013
ZANGWIL2	2	1	0.000813	2	0.000656

Tabela 2 – Desempenho Moré-Sorensen

Problema	Número de variáveis	Fatorações de Cholesky
AKIVA	2	6
ALLINITU	4	24
ARGLINA	200	4
ARGLINB	200	7344
ARGLINC	200	4710
ARWHEAD	5000	6
BARD	3	14
Continua na próxima página		

Tabela 2 – continuação da página anterior

Problema	Número de variáveis	Fatorações de Cholesky
BDQRTIC	5000	2138
BEALE	2	14
BIGGS6	6	1638
BOX3	3	10
BOXPOWER	10000	25
BOX	10000	16
BRKMCC	2	3
BROWNAL	200	99
BROWNBS	2	41
BROWNDEN	4	12
BROYDN7D	5000	1233
BRYBND	5000	307
CHAINWOO	4000	539
CHNROSNB	50	133
CHNRSNBM	50	190
CLIFF	2	29
COSINE	10000	318
CRAGGLVY	5000	20
CUBE	2	57
CURLY10	10000	73
CURLY20	10000	337
CURLY30	10000	341
DECONVU	63	3541
DENSCHNA	2	6
DENSCHNB	2	8
DENSCHNC	2	10
DENSCHND	3	97
DENSCHNE	3	155
DENSCHNF	2	6
DIXMAANA	3000	11
DIXMAANB	3000	14
DIXMAANC	3000	18
DIXMAAND	3000	22
DIXMAANE	3000	78
DIXMAANF	3000	82

Continua na próxima página

Tabela 2 – continuação da página anterior

Problema	Número de variáveis	Fatorações de Cholesky
DIXMAANG	3000	147
DIXMAANH	3000	208
DIXMAANI	3000	45
DIXMAANJ	3000	272
DIXMAANK	3000	208
DIXMAANL	3000	226
DIXMAANM	3000	15
DIXMAANN	3000	306
DIXMAANO	3000	361
DIXMAANP	3000	353
DIXON3DQ	10000	13
DJTL	2	3837
DQDRTIC	5000	5
DQRTIC	5000	186
EDENSCH	2000	27
EG2	1000	2101
EIGENALS	2550	4348
EIGENBLS	2550	5882
EIGENCLS	2652	6737
ENGVAL1	5000	13
ENGVAL2	3	33
ERRINROS	50	771
ERRINRSM	50	1745
EXPFIT	2	22
EXTROSNB	1000	4177
FLETBV3M	5000	711
FLETCBV2	5000	7
FLETCBV3	5000	9253
FLETCHBV	5000	9030
FLETCHCR	1000	2458
FMINSRF2	5625	327
FMINSURF	5625	324
FREUROTH	5000	2002
GENHUMPS	5000	9766
GENROSE	500	1358

Continua na próxima página

Tabela 2 – continuação da página anterior

Problema	Número de variáveis	Fatorações de Cholesky
GROWTHLS	3	5198
GULF	3	347
HAIRY	2	208
HATFLDD	3	33
HATFLDE	3	26
HATFLDFL	3	6351
HEART6LS	6	5442
HEART8LS	8	550
HELIX	3	35
HIELOW	3	15
HILBERTA	2	2
HILBERTB	10	3
HIMMELBB	2	286
HIMMELBF	4	1369
HIMMELBG	2	17
HIMMELBH	2	2
HUMPS	2	13253
HYDC20LS	99	9719
INDEFM	10000	110
INDEF	5000	2432
JENSMP	2	2000
JIMACK	3549	190
KOWOSB	4	29
LIARWHD	5000	18
LOGHAIRY	2	10085
MANCINO	100	2018
MARATOSB	2	5106
MEXHAT	2	2031
MEYER3	3	8232
MODBEALE	20000	4
MOREBV	5000	4
MSQRTALS	1024	626
MSQRTBLS	1024	1085
NCB20B	5000	2001
NCB20	5010	783
Continua na próxima página		

Tabela 2 – continuação da página anterior

Problema	Número de variáveis	Fatorações de Cholesky
NONCVXU2	5000	3934
NONCVXUN	5000	4690
NONDIA	5000	27
NONDQUAR	5000	28
NONMSQRT	4900	1981
OSBORNEA	5	326
OSBORNEB	11	41
OSCIGRAD	10000	412
OSCIPATH	10	3578
PALMER1C	8	137
PALMER1D	7	9
PALMER2C	8	6
PALMER3C	8	7
PALMER4C	8	10
PALMER5C	6	4
PALMER6C	8	9
PALMER7C	8	14
PALMER8C	8	9
PARKCH	15	2064
PENALTY1	1000	111
PENALTY2	200	3964
PENALTY3	200	2089
POWELLSG	5000	28
POWER	10000	46
QUARTC	5000	186
ROSENBR	2	39
S308	2	15
SBRYBND	5000	2074
SCHMVETT	5000	2003
SCOSINE	5000	2443
SCURLY10	10000	352
SCURLY20	10000	362
SCURLY30	10000	350
SENSORS	100	100
SINEVAL	2	99

Continua na próxima página

Tabela 2 – continuação da página anterior

Problema	Número de variáveis	Fatorações de Cholesky
SINQUAD	5000	2057
SISSER	2	18
SNAIL	2	155
SPARSINE	5000	577
SPARSQUR	10000	30
SPMSRTLS	4999	100
SROSENBR	5000	8
SSBRYBND	5000	488
SSCOSINE	5000	2201
STRATEC	10	123
TESTQUAD	5000	4
TOINTGOR	50	2002
TOINTGSS	5000	9
TOINTPSP	50	2013
TOINTQOR	50	4
TQUARTIC	5000	55
TRIDIA	5000	6
VARDIM	200	30
VAREIGVL	50	72
VIBRBEAM	8	434
WATSON	12	187
WOODS	4000	143
YATP1LS	2600	95
YATP2LS	2600	309
YFITU	3	426
ZANGWIL2	2	1

Tabela 3 – Desempenho GCT

Problema	No. Variáveis	Iterações GCT
AKIVA	2	12
ALLINITU	4	13
ARGLINA	200	1
ARGLINB	200	938
Continua na próxima página		

Tabela 3 – continuação da página anterior

Problema	Número de Variáveis	Iterações GCT
ARGLINC	200	544
ARWHEAD	5000	12
BARD	3	33
BDQRTIC	5000	798
BEALE	2	15
BIGGS6	6	10221
BOX3	3	23
BOXPOWER	10000	89
BOX	10000	15
BRKMCC	2	6
BROWNAL	200	81
BROWNBS	2	38
BROWNDEN	4	2154
BROYDN7D	5000	1741
BRYBND	5000	836
CHAINWOO	4000	22568
CHNROSNB	50	1721
CHNRSNBM	50	1602
CLIFF	2	86
COSINE	10000	72
CRAGGLVY	5000	773
CUBE	2	66
CURLY10	10000	5983
CURLY20	10000	5907
CURLY30	10000	6982
DECONVU	63	1913
DENSCHNA	2	12
DENSCHNB	2	11
DENSCHNC	2	20
DENSCHND	3	115
DENSCHNE	3	15
DENSCHNF	2	12
DIXMAANA	3000	10
DIXMAANB	3000	81
DIXMAANC	3000	60

Continua na próxima página

Tabela 3 – continuação da página anterior

Problema	Número de Variáveis	Iterações GCT
DIXMAAND	3000	158
DIXMAANE	3000	803
DIXMAANF	3000	791
DIXMAANG	3000	1111
DIXMAANH	3000	604
DIXMAANI	3000	24447
DIXMAANJ	3000	15539
DIXMAANK	3000	19965
DIXMAANL	3000	12471
DIXMAANM	3000	22112
DIXMAANN	3000	14969
DIXMAANO	3000	18762
DIXMAANP	3000	15656
DIXON3DQ	10000	6215
DJTL	2	2498
DQDRTIC	5000	5
DQRTIC	5000	22000
EDENSCH	2000	140
EG2	1000	3
EIGENALS	2550	9668
EIGENBLS	2550	74458
EIGENCLS	2652	58490
ENGVAL1	5000	111
ENGVAL2	3	45
ERRINROS	50	1229
ERRINRSM	50	4189
EXPFIT	2	12
EXTROSNB	1000	29042
FLETBV3M	5000	41
FLETCBV2	5000	20239
FLETCBV3	5000	897
FLETCHBV	5000	981
FLETCHCR	1000	65199
FMINSRF2	5625	1982
FMINSURF	5625	2570

Continua na próxima página

Tabela 3 – continuação da página anterior

Problema	Número de Variáveis	Iterações GCT
FREUROTH	5000	353
GENHUMPS	5000	747
GENROSE	500	3076
GROWTHLS	3	503
GULF	3	71
HAIRY	2	62
HATFLDD	3	45
HATFLDE	3	27
HATFLDFL	3	1101
HEART6LS	6	3092
HEART8LS	8	790
HELIX	3	46
HIELOW	3	28
HILBERTA	2	2
HILBERTB	10	5
HIMMELBB	2	22
HIMMELBF	4	5998
HIMMELBG	2	10
HIMMELBH	2	5
HUMPS	2	998
HYDC20LS	99	1425785
INDEFM	10000	579
INDEF	5000	1169
JENSMP	2	4000
JIMACK	3549	42639
KOWOSB	4	34
LIARWHD	5000	33
LOGHAIRY	2	1462
MANCINO	100	188
MARATOSB	2	3128
MEXHAT	2	163
MEYER3	3	3766
MODBEALE	20000	12
MOREBV	5000	18000
MSQRTALS	1024	17431

Continua na próxima página

Tabela 3 – continuação da página anterior

Problema	Número de Variáveis	Iterações GCT
MSQRTBLS	1024	11226
NCB20B	5000	10846
NCB20	5010	2868
NONCVXU2	5000	1993
NONCVXUN	5000	1999
NONDIA	5000	13
NONDQUAR	5000	21822
NONMSQRT	4900	21181
OSBORNEA	5	15933
OSBORNEB	11	171
OSCIGRAD	10000	1136
OSCIPATH	10	19210
PALMER1C	8	988
PALMER1D	7	39
PALMER2C	8	365
PALMER3C	8	54
PALMER4C	8	47
PALMER5C	6	6
PALMER6C	8	48
PALMER7C	8	156
PALMER8C	8	54
PARKCH	15	235
PENALTY1	1000	201
PENALTY2	200	26613
PENALTY3	200	2085
POWELLSG	5000	96
POWER	10000	5579
QUARTC	5000	22000
ROSENBR	2	60
S308	2	22
SBRYBND	5000	19377
SCHMVETT	5000	19732
SCOSINE	5000	8818
SCURLY10	10000	5119
SCURLY20	10000	5031

Continua na próxima página

Tabela 3 – continuação da página anterior

Problema	Número de Variáveis	Iterações GCT
sCURLY30	10000	5062
SENSORS	100	64
SINEVAL	2	107
SINQUAD	5000	145
SISSER	2	34
SNAIL	2	156
SPARSINE	5000	22199
SPARSQUR	10000	1282
SPMSRTLS	4999	905
SROSENBR	5000	10
SSBRYBND	5000	19212
SSCOSINE	5000	2801
STRATEC	10	950
TESTQUAD	5000	2304
TOINTGOR	50	383
TOINTGSS	5000	53
TOINTPSP	50	185
TOINTQOR	50	36
TQUARTIC	5000	17
TRIDIA	5000	926
VARDIM	200	227
VAREIGVL	50	359
VIBRBEAM	8	654
WATSON	12	1188
WOODS	4000	238
YATP1LS	2600	88
YATP2LS	2600	75057
YFITU	3	259
ZANGWIL2	2	1