Minimização com Restrições de Igualdade
Método de penalização

Lucas Magno

1 Introdução

Lorem ipsum

2 Otimização Restrita

Diferentemente da otimização irrestrita, neste trabalho consideraremos problemas de otimização nos quais a região viável não é todo o $\mathbb{R}^n$ e sim um subconjunto seu determinado por várias equações não-lineares.

Podemos formular o problema na seguinte forma:

$$\begin{aligned} &\text{minimizar} && f(x) \\ &\text{sujeita a} && c_i(x) = 0, \quad i = 1, m \end{aligned} \tag{2.1}$$

onde $x \in \mathbb{R}^n$ e [1]

$$\begin{aligned} f: &\quad \mathbb{R}^n \to \mathbb{R} \\ c_i: &\quad \mathbb{R}^n \to \mathbb{R} \quad , \quad i = 1, m \end{aligned}$$

Portanto, não podemos mais utilizar os algoritmos já desenvolvidos e precisamos definir as condições de otimalidade neste novo problema. Para isto, é preciso que se defina uma condição de qualificação, isto é, hipóteses adicionais que nos permitirão qualificar os minimizadores.

[1] Assumindo f e c_i continuamente diferenciáveis.

2.1 LICQ e Multiplicadores de Lagrange

Uma qualificação possível (de fato a mais fraca [Wachsmuth, 2013]) é a *Linear Independence Constraint Qualification* ou LICQ, definida a seguir.

Definição 2.1. *Um ponto x^* é dito satisfazer LICQ se e somente o conjunto formado pelos gradientes das restrições em x^**

$$\{\nabla c_1(x^*), \ldots, \nabla c_m(x^*)\}$$

é linearmente independente.

Assim, podemos agora enunciar a condição necessária de primeira ordem [Friedlander, 1994]:

Teorema 2.1. *Seja x^* um minimizador local do problema 2.1 que satisfaça LICQ, então existe $\lambda^* \in \mathbb{R}^m$ tal que*

$$\begin{aligned} \nabla \mathcal{L}(x^*) &= \nabla f(x^*) - \sum_{i=0}^{m} \lambda_i^* \nabla c_i(x^*) \\ &= 0 \end{aligned} \tag{2.2}$$

onde λ^* é conhecido como o vetor de multiplicadores de langrange.

Uma vez que temos uma condição de otimalidade, podemos a utilizar como critério de parada no algoritmo a ser implementado.

Vale notar também que podemos generalizar este método para restrições de *desigualdade*, da forma

$$c_i(x) \geq 0$$

e então obtemos as condições de Karush-Kuhn-Tucker (KKT). [Nocedal and Wright, 2006]

3 Método de Penalização

Mas como lidar com as restrições? A solução simples é não lidar com elas! [Friedlander, 1994] Quer dizer, podemos definir o seguinte problema irrestrito

$$\underset{x \in \mathbb{R}^n}{\text{minimizar}} \quad Q(x, \mu) = f(x) + \frac{1}{2\mu} \sum_{i=0}^{m} c_i(x)^2 \tag{3.1}$$

para dado $\mu \in \mathbb{R}$, no qual pontos fora da região viável recebem uma *penalidade* com peso $1/2\mu$ e assim, para μ suficientemente grande, espera-se que o algoritmo descarte pontos fora da região viável.

De fato, pode-se mostrar que, sendo a sequência $\{\mu_k\}_{k=0}^{\infty}$ tal que

$$\mu_k \to 0 \quad (k \to \infty) \tag{3.2}$$

então a sequência $\{x_k\}_{k=0}^{\infty}$ de soluções dos subproblemas

$$\underset{x \in \mathbb{R}^n}{\text{minimizar}} \quad Q(x, \mu_k)$$

tende a x^*, que é solução do problema 2.1.

No entanto, isso requer a solução exata do subproblema 3.1. Por outro lado, se tivermos x_k tal que

$$\|\nabla Q(x_k, \mu_k)\| \leq r_k, \quad r_k \in \mathbb{R}$$

e

$$r_k \to 0$$

também é possível mostrar que $x_k \to x^*$ e

$$\lambda_i^k = -\frac{c_i(x_k)}{\mu_k} \to \lambda_i^* \quad i = 1, m$$

o que é muito mais útil de um ponto de vista prático.

4 Algoritmo

Tendo isto, podemos montar um algoritmo a fim de resolver o problema 2.1.
Dados $\{\mu_k\}_{k=0}^{\infty}$ e $\{r_k\}_{k=0}^{\infty}$ como definidos acima e $x_0^S \in \mathbb{R}^n$

Algorithm 4.1 Método de Penalização

for $k = 0, 1, 2, \ldots$ **do**
Passo 1 Usando x_k^S como ponto inicial, encontrar x_k solução aproximada do subproblema 3.1 tal que $\|\nabla Q(x_k, \mu_k)\| \leq r_k$
Passo 2 Se x_k satisfaz o critério 2.2, parar.
Passo 3 Escolher x_{k+1}^S
Passo 4 Escolher μ_{k+1}
end for

Por motivo de simplicidade, serão usadas as fórmulas

$$r_k = \epsilon$$

$$x_{k+1}^S \leftarrow x_k$$

$$\mu_{k+1} \leftarrow \mu_k / 2$$

onde ϵ é a tolerância definida para o programa.

4.1 Mal condicionamento

Como método no primeiro passo foi escolhido o de Newton, no entanto é necessário uma observação. Para encontrar uma direção de descida d, o método resolve um sistema na forma

$$\nabla^2 Q(x, \mu) d = -\nabla Q(x, \mu)$$

Porém, como

$$\begin{aligned}\nabla^2 Q(x,\mu) &= \nabla^2 f(x) + \frac{1}{\mu}\sum_{i=0}^{m} c_i(x)\nabla^2 c_i(x) + \frac{1}{\mu}\sum_{i=0}^{m} \nabla c_i(x)\nabla c_i(x)^\top \\ &= \nabla^2 \mathcal{L}(x) + \frac{1}{\mu}\sum_{i=0}^{m} \nabla c_i(x)\nabla c_i(x)^\top\end{aligned} \tag{4.1}$$

nota-se que o último termo é uma matriz de posto m e, para μ suficientemente pequeno, vale que $\nabla^2 Q(x,\mu)$ é mal condicionada, o que pode inviabilizar a solução do sistema.

Em seu lugar, podemos utilizar um sistema equivalente introduzindo uma variável auxiliar $z \in \mathbb{R}^m$:

$$\begin{bmatrix} \nabla^2 \mathcal{L}(x) & A(x)^\top \\ A(x) & -\mu I \end{bmatrix} \begin{bmatrix} d \\ z \end{bmatrix} = \begin{bmatrix} -\nabla Q(x,\mu) \\ 0 \end{bmatrix}$$

onde

$$A(x) = \begin{bmatrix} \nabla c_1(x) \\ \vdots \\ \nabla c_m(x) \end{bmatrix}$$

Esse sistema não apresenta mal condicionamento devido a μ pequeno e portanto será utilizado na implementação do método de Newton, mas fora isso o método se mantém o mesmo do trabalho anterior.

5 O Programa

Para implementação do algoritmo foi feito um programa em `FORTRAN 2008`, usando como base o código já desenvolvido anteriormente, portanto aqui só serão mencionadas as adições referentes a este trabalho.

5.1 Módulos

Os módulos novos criados foram:

`constrained` Implementa o método de penalização bem como o de Newton modificado.

`stats2` Fornece *wrappers* e subrotinas para automatizar a contagem e a impressão de diversas etapas durante a execução dos algoritmos.

`test` Implementa vários problemas de teste para o programa.

5.2 Arquivos

Os arquivos mantiveram a hierarquia, mas foi criado um diretório `data/` para guardar os resultados da execução (`.dat`) e também os códigos para geração dos gráficos (`.gp`), através do programa `gnuplot`.

5.3 Compilação

A compilação manteve a mesma estrutura:

`make EP2` Compila o programa, criando o executável.

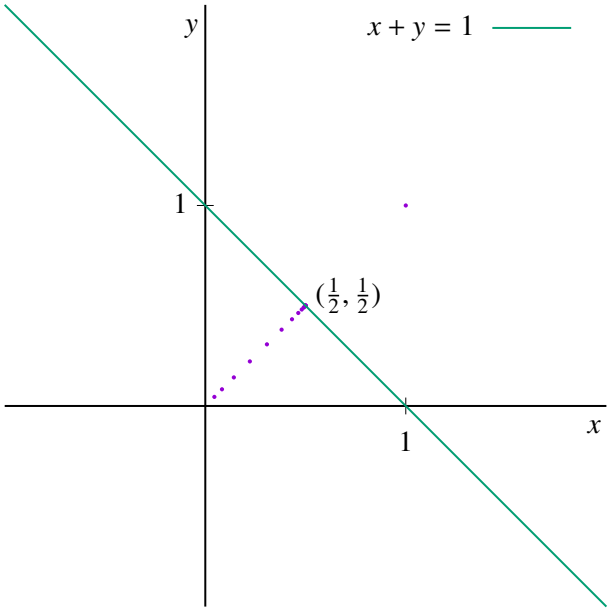
`make tex2` Gera os gráficos e compila o relatório, criando o arquivo `.pdf`.

`make clean` Realiza a limpeza, deletando os arquivos de saída.

A única observação é que, para o relatório, foi utilizado o programa `rubber`, em sua versão 1.4.

6 Execução

7 Resultados

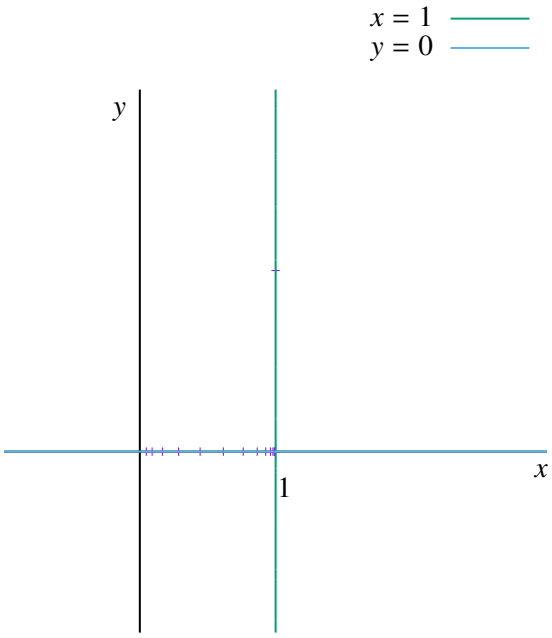


8 Conclusão

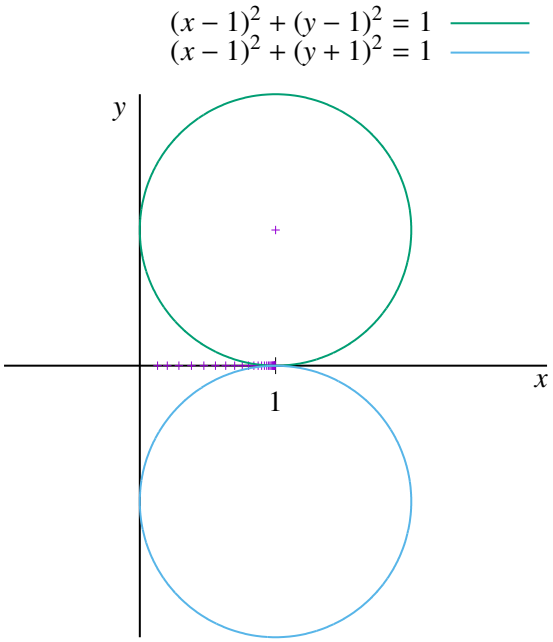
Referências

A. Friedlander. *Elementos de Programação Não-Linear*. Editora UNICAMP, 1 edition, 1994.

J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, 2nd edition, 2006.



G. Wachsmuth. On {LICQ} and the uniqueness of lagrange multipliers. *Operations Research Letters*, 41(1):78 – 80, 2013. ISSN 0167-6377. doi: http://dx.doi.org/10.1016/j.orl.2012.11.009. URL http://www.sciencedirect.com/science/article/pii/S0167637712001459.


$(x-1)^2+(y-1)^2=1$
$(x-1)^2+(y+1)^2=1$
y
x
1