Otimização quadrática restrita e o método de programação quadrática sequencial

Alex Augusto Nunes Machado
Simone Aparecida Miloca - Unioeste

(*Recebido em 21/10/2022. Aceito em 29/11/2022. Publicado em 22/12/2022*)

Resumo: Este trabalho teve por intuito apresentar o algoritmo PQS para então mostrar, matematicamente, sua eficiência em resolver problemas de programação quadrática com restrições de igualdade em apenas uma iteração. Para tanto, foram enunciados alguns conceitos básicos da otimização restrita, bem como os conceitos mínimos associados ao método sequencial. Por fim, o algoritmo PQS foi implementado em linguagem Python, e três exemplos numéricos que mostram a capacidade de resolver problemas envolvendo restrições de igualdade com baixo custo temporal foram apresentados.

Palavras-chave: Otimização Quadrática Restrita, PQS, Python.

1 Introdução

De modo geral, o campo da Otimização consiste em

$$\begin{array}{ll} \text{Otimizar} & f(x) \\ \text{sujeito a} & x \in \Omega \subset \mathbb{R}^n \end{array} \tag{1}$$

sendo que f é denominada *função objetivo* e Ω representa, no caso da otimização restrita, um conjunto de restrições (outras funções) que podem ser de igualdade ou desigualdade, ou ainda, simplesmente representam uma restrição na qual o tipo de variável envolvida é explicitada, isto é, se as variáveis são números reais, não-negativos, inteiros, etc.

Em diversos campos de pesquisa nas quais os tomadores de decisões buscam obter os melhores resultados possíveis, dentro de certas condições podem surgir situações nas quais os problemas são modelados como um problema de otimização. Na construção desses modelos podem surgir funções com leis de formação das mais variadas e, com isso, conforme destacam Ribeiro e Karas (2013) e Goldbarg e Luna (2005), é possível agrupar tais problemas de otimização em algumas categorias, cada qual possuindo técnicas específicas para sua resolução.

Dentre as diversas categorias possíveis, há os Problemas de Otimização Quadrática Restrita. Tal classe de problemas aparecem em vários campos do conhecimento humano, a exemplo da Engenharia Aeronáutica, Logística, Equações Diferenciais, Engenharia Estrutural, entre vários outros [1].

[1] A título de exemplo, ver ver Júnior (2007), Ojima e Yamakami (2006), Façanha, Carneiro e Filho (2013) e Barros et al. (2017).

Problemas de Otimização Quadrática Restrita ou, equivalentemente, de *Problemas de Programação Quadrática Restrita*, são caracterizados formalmente, e de modo matricial, por

$$\begin{array}{cc} \text{Minimizar} & f(x) = \frac{1}{2}x^T Sx + v^T x + c \\ \text{sujeito a} & Ax \leq b \end{array} \tag{2}$$

com $x, v \in \mathbb{R}^n$, $b \in \mathbb{R}^l$, $c \in \mathbb{R}$, $A \in M_{l\times n}(\mathbb{R})$ e $S \in M_{n\times n}(\mathbb{R})$ sendo que S é simétrica definida positiva [2].

Diante disso, notemos que

$$\begin{aligned} f(x) &= \frac{1}{2}x^T Sx + v^T x + c \\ &= \frac{1}{2}(x_{1j})(s_{ij})(x_{j1}) + (v_{1j})(x_{j1}) + c \\ &= \frac{1}{2}(x_{1j})\left(\sum_{k=1}^{n} s_{ik}x_{k1}\right) + \left(\sum_{j=1}^{n} v_{1j}x_{j1}\right) + c \\ &= \frac{1}{2}\left(\sum_{p=1}^{n} x_{1p}\left(\sum_{k=1}^{n} s_{ik}x_{k1}\right)\right) + \left(\sum_{i=1}^{n} v_{1i}x_{i1}\right) + c \\ &= \frac{1}{2}\sum_{p=1}^{n}\sum_{k=1}^{n} x_{1p}s_{ik}x_{k1} + \left(\sum_{k=1}^{n} v_{1k}x_{k1}\right) + c \\ &= \frac{1}{2}\sum_{p=1}^{n}\sum_{k=1}^{n} x_p s_{pk}x_k + \sum_{k=1}^{n} v_k x_k + c \\ &= \frac{1}{2}\sum_{p=1}^{n}\sum_{k=1}^{n} s_{pk}x_p x_k + \sum_{k=1}^{n} v_k x_k + c \end{aligned}$$

Deste modo, podemos reformular (2) como

$$\begin{array}{cc} \text{Minimizar} & f(x) = \frac{1}{2}\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n} s_{ij}x_i x_j + \sum\limits_{i=1}^{n} v_i x_i + c \\ \text{sujeito a} & Ax \leq b \end{array} \tag{3}$$

que consiste em outra forma de caracterizar os Problemas de Otimização Quadrática Restrita.

Diante disso, observemos que essa categoria se enquadra na classe de problemas não-lineares pois embora suas restrições sejam lineares, a função objetivo a ser otimizada possui termos quadráticos e/ou produto de duas ou mais variáveis.

Isto posto, visando o estudo de métodos de resolução de problemas nos moldes de (2), ou de forma equivalente (3), as próximas seções serão destinadas a apresentação de alguns conceitos e resultados de otimização restrita de modo geral.

[2] Tendo em mente os propósitos deste trabalho, vale comentar que a Definição 2 é a que aparece na literatura. Todavia, deve-se notar que do ponto de vista técnico a soma $\frac{1}{2}x^T Sx + v^T x$ gerá uma matriz de ordem um, e como c é uma constante real, o que seria somar uma matriz com um número? Um matemático resolve esse problema recordando que existe um isomorfismo entre o espaço das matrizes de ordem um e o conjunto dos números reais, deste modo, pode desconsiderar esse abuso de linguem uma vez observado este detalhe. Todavia, embora esse detalhe possa parecer pequeno para um matemático, isso pode ser fonte de algumas mensagens de erro durante o processo de implementação computacional dependendo da linguagem utilizada (não é o caso para `Python`).

2 Alguns conceitos elementares sobre otimização restrita

Dada uma função $f\colon \mathbb{R}^n \longrightarrow \mathbb{R}$, problemas de otimização restrita são geralmente caracterizados por

$$\begin{array}{ll} \text{Otimizar} & f(x) \\ \text{sujeito a} & \Omega = \{x \in \mathbb{R}^n;\ h(x) = 0 \text{ e } g(x) \leq 0\} \end{array} \tag{4}$$

em que $h\colon \mathbb{R}^n \longrightarrow \mathbb{R}^l$ e $g\colon \mathbb{R}^n \longrightarrow \mathbb{R}^m$. O conjunto Ω recebe o nome de ***conjunto viável*** do problema de otimização.

Vale destacar que é possível encontrar situações nas quais as restrições envolvidas são apenas de igualdade, nestes casos, (4) se resume a

$$\begin{array}{ll} \text{Otimizar} & f(x) \\ \text{sujeito a} & \Omega = \{x \in \mathbb{R}^n;\ h(x) = 0\} \end{array} \tag{5}$$

com $h\colon \mathbb{R}^n \longrightarrow \mathbb{R}^l$.

2.1 Condições de Lagrange

Consideremos a função diferenciável $f\colon A \subset \mathbb{R}^n \longrightarrow \mathbb{R}$ tal que $x \mapsto f(x) = z$ esteja sujeita à restrição $h\colon \mathbb{R}^n \longrightarrow \mathbb{R}$ de forma que $x \mapsto h(x) = 0$, $x^{*T} = [x_1^*, \ldots, x_n^*]$ seja minimizador local de f, $P^T = [x_1^*, \ldots, x_n^*, f(x^*)] \in R^{n+1}$ e seja S a superfície associada a f, isto é, $S = \{(x, f(x)) \in \mathbb{R}^{n+1};\ x \in A\}$. Seja ainda a função vetorial $\gamma\colon X \subset \mathbb{R} \longrightarrow A$ tal que $\gamma(t)^T = [x_1(t), \ldots, x_n(t)]$ de modo que $\gamma(t) \in A$, $x^* \in gr(\gamma)$ e γ seja diferenciável, sendo $gr(\gamma)$ o gráfico de γ.

Com isso, tomemos t_0 como sendo o argumento de γ correspondente ao ponto x^*, assim $\gamma(t_0)^T = [x_1^*, \ldots, x_n^*]$. Isto posto, a composta $f \circ \gamma\colon X \subset \mathbb{R} \longrightarrow \mathbb{R}$ consiste nos valores de f restritos a curva γ, isto é, $(f \circ \gamma)(t) = f(\gamma(t)) = f(x)$ com $x \in A$.

Uma vez que x^* é minimizador local de f, segue que

$$f(x_1^*, \ldots, x_n^*)^T = f(x^*) \leq f(x) = f(x_1, \ldots, x_n)$$

para todo $x \in A$, assim segue da definição de γ que $x_1^* = x_1(t_0), \ldots, x_n^* = x_n(t_0)$ e $x_1 = x_1(t), \ldots, x_n = x_n(t)$ e assim

$$\begin{aligned} f(\gamma(t_0)) = f(x_1(t_0), \ldots, x_n(t_0)) &\leq f(x_1(t), \ldots, x_n(t)) = f(\gamma(t)) \\ \iff (f \circ \gamma)(t_0) &\leq (f \circ \gamma)(t) \end{aligned}$$

para todo $t \in X$. Portanto, segue que t_0 é minimizador local de $f \circ \gamma$, logo

$$\begin{aligned} 0 = \nabla(f \circ \gamma)(t_0) &= \nabla f(\gamma(t_0)) \\ &= \left(\frac{\partial}{\partial x_1} f(x^*) \cdot x_1'(t_0), \ldots, \frac{\partial}{\partial x_n} f(x^*) \cdot x_n'(t_0) \right) && \text{(regra da cadeia)} \\ &= \left(\frac{\partial}{\partial x_1} f(x^*), \ldots, \frac{\partial}{\partial x_n} f(x^*) \right) \cdot (x_1'(t_0), \ldots, x_n'(t_0)) = \nabla f(x^*) \cdot \gamma'(t_0). \end{aligned}$$

Ou seja,

$$\nabla f(x^*) \cdot \gamma'(t_0) = 0.$$

Assim, segue por definição que $\nabla f(x^*)$ é perpendicular ao vetor da tangente $\gamma'(t_0)$ para todas as curvas C. Por outro lado, supondo[3] $\nabla h(x^*)$ ortogonal a $\gamma'(t_0)$ para todas as curvas, isto é,

$$\nabla h(x^*) \cdot \gamma'(t_0) = 0,$$

obtemos que o $\nabla f(x^*)$ deve ser paralelo a $\nabla h(x^*)$, e consequentemente paralelo a $-\nabla h(x^*)$, ou seja, caso $\nabla h(x^*) \neq 0$ existe[4] algum $\lambda \in \mathbb{R}$ de forma que

$$\nabla f(x^*) = -\nabla h(x^*)\lambda \qquad \equiv \qquad \nabla f(x^*) + \nabla h(x^*)\lambda = 0. \tag{6}$$

Neste caso, (6) é conhecido como condição de Lagrange e λ é denominado como ***multiplicador de Lagrange***.

Com (6) Joseph-Louis Lagrange desenvolveu o método de busca de máximos e mínimos de funções sujeitas a uma restrição $h(x) = 0$. Esse método afirma que caso f possua máximo e mínimo, e ainda, $\nabla h(x) \neq 0$ sobre a superfície $h(x) = 0$, então

Passo 1 Determine todos os valores $x \in A \subset \mathbb{R}^n$ tais que (6) é satisfeita;

Passo 2 Calcule f em cada um dos pontos obtidos no *Passo 1*, o maior dos valores obtidos será o máximo de f e o menor será o mínimo de f.

Isto posto, conforme destacam Izmailov e Solodov (2005), é comum reescrever[5] (6) em termos da função Lagrangena, como estabelece a Definição 1.

Definição 1. Consideremos o problema de otimização restrita (5). A função $\mathcal{L}\colon \mathbb{R}^n \times \mathbb{R}^l \longrightarrow \mathbb{R}$ tal que $x \longmapsto \mathcal{L}(x, \lambda) = f(x) + h(x)\lambda$, com $h(x) = (h_1(x), \ldots, h_l(x))^T$, é denominada ***Lagrangeana*** de (5).

Assim, quando $\nabla h(x) \neq 0$, a existência de um minimizador local de um problema de otimização restrita nos moldes de (5) está condicionada a existência e unicidade do multiplicador de Lagrange λ. Tal fato fica garantido pelo Teorema 3 conhecido como *Condição de otimalidade de Lagrange*. Diante disto, para enunciar tal teorema precisamos definir o que vem a ser uma condição de regularidade.

Definição 2. Seja o problema de otimização restrita (5). As condições $\{h_i'(x^*); i = 1, \ldots, l\}$ ser um conjunto linearmente independente ou h ser uma função afim, isto é, $h(x) = Ax - a$ com $A \in M_{l\times n}$ e $a \in \mathbb{R}^l$, são denominadas ***condições de regularidade das restrições*** ou ainda condições de qualificação das restrições.

[3]Para mais detalhes, ver Stewart (2013, p. 845).

[4]Ver Guidorizzi (2000, p. 330) para mais detalhes.

[5]O leitor deve atentar-se ao fato de que a argumentação realizada anteriormente ao Teorema 3 considerou $h\colon B \subset \mathbb{R}^n \longrightarrow \mathbb{R}$, porém, o problema de otimização estabelecido em (5) define $h\colon \mathbb{R}^n \longrightarrow \mathbb{R}^l$. Dito isto, também é possível mostrar a existência de $\lambda \in \mathbb{R}^l$ para o caso em que a imagem de h ($Im\ (h)$) pertence a $\mathbb{R}^l$, contudo sua demonstração foge ao escopo deste trabalho. Ao leitor interessado, recomendamos a leitura do Capítulo 2 de Izmailov e Solodov (2005).

Teorema 3 (Condição de otimalidade de Lagrange)**.** *Consideremos o problema de otimização restrita (5). Se $x^* \in \mathbb{R}^n$ é minimizador local de (5), f é diferenciável em x^*, h é diferenciável numa vizinhança de x^*, com derivada contínua em x^*, e uma das condições de regularidade das restrições (Definição 2) válidas, então existe $\lambda^* \in \mathbb{R}^l$ tal que*

$$\mathcal{L}'_x\left(x^*, \lambda^*\right) = 0.$$

Caso a primeira condição seja satisfeita, então $\lambda^* \in \mathbb{R}^l$ é único.

Prova. Ver Izmailov e Solodov (2005, p. 46). □

Com isso, observemos que, para algum $\lambda \in \mathbb{R}^l$,

$$\mathcal{L}'_x(x, \lambda) = \nabla f(x) + (\nabla h(x))^T \lambda$$

enquanto que

$$\mathcal{L}'_\lambda(x, \lambda) = h(x),$$

deste modo, temos que

$$\nabla \mathcal{L}(x, \lambda) = \begin{bmatrix} \nabla f(x) + (\nabla h(x))^T \lambda \\ h(x) \end{bmatrix}.$$

Ou seja, quando $\nabla \mathcal{L}(x, \lambda) = 0$ temos que

$$\nabla \mathcal{L}(x, \lambda) = \begin{bmatrix} \nabla f(x) + (\nabla h(x))^T \lambda \\ h(x) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \qquad (\textit{Sistema de Lagrange}) \tag{7}$$

Conforme destacado por Izmailov e Solodov (2005) caso esta condição de Lagrange seja satisfeita significa que o gradiente da função objetivo ($f'(x^*)$) pode ser descrito como uma combinação linear das derivadas parciais das restrições, ou seja, combinação linear de $\{h'_i;\ i = 1, \ldots, l\}$.

Isto posto, assim como mostrado para problemas de otimização restrita nos termos de (5), também é possível mostrar um resultado semelhante para problemas mais gerais, nos moldes de (4), tal resultado é conhecido como "*Condições de Karush-Kuhn-Tucker*", ou mais economicamente, denominado por "*Condições de KKT*". Para mais detalhes a respeito, recomendamos a leitura de Izmailov e Solodov (2005), Izmailov e Solodov (2007) e Ribeiro e Karas (2013).

3 Programação quadrática sequencial

Conforme apresentado até o momento existem resultados matemáticos que nos permitem assegurar a existência de solução para problemas de otimização, desde que estes satisfaçam certas condições, além de apresentar meios para sua determinação. Todavia, nem sempre a busca de tais soluções é viável (ou mesmo possível) analiticamente, haja vista a não linearidade das funções envolvidas Luenberger e Ye (2016, p. 06).

Neste sentido, ao longo dos anos foram desenvolvidos numerosos algoritmos para viabilizar a obtenção de solução para problemas de otimização, mesmo que de modo aproximado. Cabe salientar que no contexto de Otimização, a terminologia corrente utilizada como sinônimo de Algoritmo é ***Método***.

Mas o que vem a ser um Algoritmo? Conforme definido em Burden, Faires e Burden (2015) um Algoritmo pode ser caracterizado como um "procedimento que descreve, sem ambiguidades, uma sequência finita de passos a serem feitos em uma ordem específica" tendo como objetivo "implementar um procedimento para resolver um problema ou aproximar uma solução do problema".

Para ilustrar essas ideias, destacando a diferença entre algoritmos sequenciais e não-sequenciais, consideremos a seguinte situação. Imaginemos que estamos cansados de resolver equações de segundo grau na mão e desejamos automatizar isso por meio de um programa de computador, um algoritmo básico para essa finalidade é descrito no Algoritmo 1 a seguir.

Entrada $a \leftarrow$ Valor do coeficiente a;
$b \leftarrow$ Valor do coeficiente b;
$c \leftarrow$ Valor do coeficiente c;
if $b^2 - 4ac \neq 0$ **then**
  $x_1 \leftarrow \dfrac{-b + \sqrt{b^2 - 4ac}}{2a}$;
  $x_2 \leftarrow \dfrac{-b - \sqrt{b^2 - 4ac}}{2a}$;
else
  $x_1 \leftarrow \dfrac{-b}{2a}$;
  $x_2 \leftarrow x_1$;
end
Output: As raízes são x_1 e x_2

Algoritmo 1: Algoritmo básico para a resolução de equações segundo grau

Notemos que o Algoritmo 1 fornece uma receita, por assim dizer, de como obter as raízes de uma equação do segundo grau, independentemente de serem reais ou complexas [6]. De toda forma, tal algoritmo ainda poderia ser melhorado no sentido de oferecer ao usuário a possibilidade de resolver mais de uma equação, conforme estabelece o Algoritmo 2. Assim, ao acrescentarmos o laço de repetição *while*, durante uma mesma execução do algoritmo teremos a possibilidade resolver varias equações de segundo grau, e mais que isso, cada resolução independe dos valores obtidos na execução do laço anterior, portanto, temos um algoritmo não-sequencial.

[6] Vale comentar que por definição um algoritmo consiste em uma sequência de passos que, ao serem implementados, resolvem um problema ou, ao menos, aproximam uma solução. Sendo assim, não há exigência de uma linguagem de programação em si, embora a utilização de uma seja necessária posteriormente.

De todo modo, a linguagem Python, que foi utilizada na implementação de PQS, tem suporte nativo aos números complexos, sendo assim o Algoritmo 1 tal como apresentado é passível de ser implementado sem qualquer problema quando $b^2 - 4ac < 0$.

$B \leftarrow 1$;
while $B = 1$ **do**
  $a \leftarrow$ Valor do coeficiente a;
  $b \leftarrow$ Valor do coeficiente b;
  $c \leftarrow$ Valor do coeficiente c;
  if $b^2 - 4ac \neq 0$ **then**
    $x_1 \leftarrow \dfrac{-b + \sqrt{b^2 - 4ac}}{2a}$;
    $x_2 \leftarrow \dfrac{-b - \sqrt{b^2 - 4ac}}{2a}$;
  else
    $x_1 \leftarrow \dfrac{-b}{2a}$;
    $x_2 \leftarrow x_1$;
  end
  Output: As raízes são x_1 e x_2
  Output: Deseja resolver outra equação? 1 para SIM. 0 para NÃO.
  $B \leftarrow$ Entrada do usuário ;
end

Algoritmo 2: Algoritmo básico para a resolução de equações de segundo grau

Todavia, existem algoritmos que para cada nova repetição do laço (*while, for,* ...) utilizam-se dos resultados obtidos anteriormente. O método da bissecção descrito no Algoritmo 3 é um exemplo deste tipo de algoritmo.

omar um intervalo $[a, b]$;
Escolher a condição inicial como sendo $x_0 = \dfrac{b+a}{2}$;
$k \leftarrow 0$;
while $f'(x_k) \neq 0$ **do**
  if $f'(x_k) < 0$ **then**
    $a \leftarrow x_k$;
  else
    $b \leftarrow x_k$;
  end
  $x_k = \dfrac{b+a}{2}$;
end

Algoritmo 3: Exemplo genérico de implementação do método da bissecção

Observemos que a cada execução do laço *while* o método da bissecção reduz o intervalo de busca da solução do problema, que só é possível pois a informação do valor da derivada em x_k é levada em consideração para atualizar o valor de a, ou de b. Ou seja, enquanto o critério de parada não for satisfeito, enquanto $f'(x_k) = 0$ não ocorrer, cada execução do laço *while* atualiza o intervalo de busca para uma nova execução considerando a informação obtida na execução atual.

É intuitivo notar que o Algoritmo 3 gera portanto uma sequência de pontos de $\mathbb{R}$, deste modo, temos que tal algoritmo é caracterizado como um *Método Sequencial*. De modo geral, conforme formalizam Izmailov e Solodov (2007), temos um ***Método Sequencial*** "quando cada ponto se define de acordo com a informação obtida nos pontos anteriores".

Definição 4. Dado um problema de otimização nos moldes de (4) as imagens $x^1, \ldots, x^k, \ldots \in \mathbb{R}^n$ da sequência de pontos geradas por um método sequencial são denominados ***aproximações*** à solução do problema, ou ainda, *iterandos* do método [7].

Convém mencionar que a sequência $(x^k)_{k\in\mathbb{N}}$ gerada por algum método sequencial também é conhecida como *trajetória*.

Definição 5. Seja $x\colon \mathbb{N} \to \mathbb{R}^n$ uma sequência gerada por algum método sequencial. A geração de uma nova aproximação x^{x+1} a partir de x^k é denominada ***iteração*** do método.

Isto posto, vale destacar que vários algoritmos foram desenvolvidos para a resolução computacional de problemas de Otimização Quadrática, dentre eles tem-se o Método de Pontos Interiores e o Simplex Modificado. Todavia, neste trabalho temos interesse em estudar o algoritmo conhecido como Programação Quadrática Sequencial (PQS) e vamos mostrar que no caso de problemas nos moldes de 2 (com restrições de igualdade) tal algoritmo os resolve em apenas uma iteração.

3.1 O algoritmo PQS

Conforme apresentado por Izmailov e Solodov (2007) e Ribeiro e Karas (2013) o método de Programação Quadrática Sequencial (PQS) consiste, em essência, "na resolução de uma sequência de problemas de otimização quadrática" em que

> a ideia consiste em substituir, a cada iteração, a função objetivo por um modelo quadrático do Lagrangiano e as restrições por equações ou inequações lineares, aproximações de Taylor de primeira ordem em torno no ponto corrente (RIBEIRO; KARAS, 2013).

Convém destacar que esse modelo quadrático do Lagrangiano consiste na aproximação de Taylor de segunda ordem para a função objetivo.

Com isso, esta abordagem permite reduzir um problema inicialmente complexo, a resolução de uma sequência de problemas relativamente mais "simples", "que muda a cada iteração de acordo com as informações disponíveis no ponto corrente" (RIBEIRO; KARAS, 2013). E conforme pontua Izmailov e Solodov (2007) quando um problema de programação quadrática é adequadamente construído pode ser uma boa aproximação do problema original, ao mesmo tempo em que é de resolução relativamente fácil.

[7] Notemos que a notação x_k, usada no contexto de $\mathbb{R}$, é substituída por x^k facilitando, desta forma, a associação com o $\mathbb{R}^n$.

Diante disso, considerando o problema (5), temos que (8) formaliza matematicamente estas ideias.

$$\begin{array}{ll} \text{Otimizar} & \mathcal{L}(x^k,\lambda^k)+\nabla_x(x^k)(x-x^k)+\frac{1}{2}\nabla^2_{xx}\mathcal{L}(x^k,\lambda^k)(x-x^k) \\ \text{sujeito a} & x\in\Omega=\{x\in\mathbb{R}^n;\ h(x^k)+h'(x-x^k)=0\} \end{array} \tag{8}$$

Isto posto, o Algoritmo 4 apresenta o método PQS [8].

escolher o palpite inicial (x^0,λ^0);
$k\leftarrow 0$;
while $\nabla\mathcal{L}(x^k,\lambda^k)\neq 0$ **do**
Passo 1: Resolva o problema $\nabla\mathcal{L}(x^*,\lambda)=0$ obtendo um ponto estacionário do problema e o defina como x^{k+1}, bem como o multiplicador de Lagrange associado e o defina como λ^{k+1}.
Passo 2: $k\leftarrow k+1$
end

Algoritmo 4: O Método PQS

Embora o Algoritmo 4 seja relativamente simples, e extremamente compacto, é natural o questionamento "*Como executamos o Passo 1?*". Para isso, consideremos o problema de minimização conforme definido em (5). Com isso, de acordo com o apresentado na Subseção 2.1, em particular na igualdade (7), segue que para x^* ser minimizador local de f deverá ocorrer (para algum $\lambda\in\mathbb{R}^l$) que

$$\nabla\mathcal{L}(x^*,\lambda)=0\iff\begin{bmatrix}\nabla f(x^*)+(\nabla h(x^*))^T\lambda\\ h(x^*)\end{bmatrix}=\begin{bmatrix}0\\0\end{bmatrix}. \tag{9}$$

À luz disso, notemos que podemos fazer uso do método de Newton-Raphson[9] ao considerarmos

$$\begin{array}{rcl}\varphi\colon\mathbb{R}^n & \longrightarrow & \mathbb{R}^{n+l}\\ x & \longmapsto & \varphi(x)=\nabla\mathcal{L}(x,\lambda).\end{array}$$

Assim, reescrevendo (9) vem que

$$\varphi(x^*)=0. \tag{10}$$

[8] Relembremos o conceito de ponto estacionário.

Definição 6. Sejam $f\colon\Omega\subset\mathbb{R}^m\longrightarrow\mathbb{R}$ diferenciável em Ω e $x^*\in\Omega$. Definimos x^* como sendo ***ponto crítico*** ou (*estacionário*) de f se, e só se, $\nabla f(x^*)=0$.

[9] Para detalhes a respeito deste método, ver Izmailov e Solodov (2007), Ribeiro e Karas (2013) ou https://www.ufrgs.br/reamat/CalculoNumerico/livro-py/livro-py.pdf.

Tomando $x^k \in \mathbb{R}^n$ como sendo uma aproximação para a solução x^* e aplicando o teorema de Taylor de primeira ordem em torno de x^k (com k fixo), obtemos que

$$\varphi(x) = \varphi(x^k) + \nabla\varphi(x^k)(x - x^k) + r(x)$$

para $\lim\limits_{x \to x^k} \dfrac{r(x)}{|x - x^k|} = 0$ e $x \in Dom(\varphi)$, isto é, para x^k suficientemente próximo à x^*, temos que

$$\varphi(x) = \varphi(x^k) + \nabla\varphi(x^k)(x - x^k), \tag{11}$$

assim, de (10) e (11) temos que

$$\varphi(x^k) + \nabla\varphi(x^k)(x^* - x^k) = 0,$$

dessa forma, a execução do Passo 1 no Algoritmo 4 depende da não-singularidade da Hessiana de $\mathcal{L}$ em (x^k, λ^k), além é claro que as somas e multiplicações matriciais sejam possíveis. Caso tais condições sejam satisfeitas, temos que

$$\begin{aligned}
&\varphi(x^k) + \nabla\varphi(x^k)(x^* - x^k) = 0 \\
&\quad \Longrightarrow \nabla\mathcal{L}\left(x^k, \lambda^k\right) + \nabla\left(\nabla\mathcal{L}\left(x^k, \lambda^k\right)\right)\left(\begin{bmatrix} x^* \\ \lambda \end{bmatrix} - \begin{bmatrix} x^k \\ \lambda^k \end{bmatrix}\right) = 0 \\
&\quad \Longrightarrow \nabla^2\mathcal{L}\left(x^k, \lambda^k\right)\left(\begin{bmatrix} x^* \\ \lambda \end{bmatrix} - \begin{bmatrix} x^k \\ \lambda^k \end{bmatrix}\right) = -\nabla\mathcal{L}\left(x^k, \lambda^k\right)
\end{aligned}$$

e para $\left(\nabla^2\mathcal{L}\left(x^k, \lambda^k\right)\right)^{-1}$ não singular, temos que

$$\begin{aligned}
&\Longrightarrow \begin{bmatrix} x^* - x^k \\ \lambda - \lambda^k \end{bmatrix} = -\left(\nabla^2\mathcal{L}\left(x^k, \lambda^k\right)\right)^{-1}\nabla\mathcal{L}\left(x^k, \lambda^k\right) \\
&\Longrightarrow \begin{bmatrix} x^* \\ \lambda \end{bmatrix} = \begin{bmatrix} x^k \\ \lambda^k \end{bmatrix} - \left(\nabla^2\mathcal{L}\left(x^k, \lambda^k\right)\right)^{-1}\nabla\mathcal{L}\left(x^k, \lambda^k\right).
\end{aligned} \tag{12}$$

Portanto, a próxima aproximação x^{k+1} para x^* é tal que

$$\begin{bmatrix} x^{k+1} \\ \lambda^{k+1} \end{bmatrix} = \begin{bmatrix} x^k \\ \lambda^k \end{bmatrix} - \left(\nabla^2\mathcal{L}\left(x^k, \lambda^k\right)\right)^{-1}\nabla\mathcal{L}\left(x^k, \lambda^k\right). \tag{13}$$

Esta abordagem é conhecida como *Método de Newton para o Sistema de Lagrange*.

Vale destacar que, em geral, como o custo computacional para inverter matrizes é maior que o custo da resolução de sistemas lineares[10] e, visando um melhor desempenho computacional, (12) foi reescrita em termos do sistema linear a seguir

$$\nabla^2\mathcal{L}\left(x^k, \lambda^k\right)\begin{bmatrix} x \\ \lambda \end{bmatrix} = \nabla^2\mathcal{L}\left(x^k, \lambda^k\right)\begin{bmatrix} x^k \\ \lambda^k \end{bmatrix} - \nabla\mathcal{L}\left(x^k, \lambda^k\right) \tag{14}$$

deste modo, a próxima aproximação x^{k+1} para x^* consiste em nada mais que a solução do sistema linear (14), isto é, $x^{k+1} = x$ e $\lambda^{k+1} = \lambda$.

[10]Para mais detalhes ver https://www.ufrgs.br/reamat/CalculoNumerico/livro-py/livro-py.pdf.

Um caso particular

Isto posto, uma questão razoável que emerge é: "E caso a função objetivo do problema que estamos trabalhando já seja uma quadrática? Ou ainda, e se além da função objetivo ser quadrática, as restrições envolvidas nesse problema já fossem lineares? Como ficaria a aplicação deste método?". Pois bem, visando responder a tais questionamentos, consideremos o problema quadrático similar ao definido em (2), porém de forma que as restrições envolvidas sejam de igualdade, isto é,

$$\begin{array}{cc} \text{Minimizar} & f(x) = \frac{1}{2}x^T Sx + v^T x + c \\ \text{sujeito a} & Ax - b = 0, \end{array} \tag{15}$$

com $x, v, b \in \mathbb{R}^n$, $c \in \mathbb{R}$, $S \in M_{n\times n}(\mathbb{R})$ e $A \in M_{l\times n}(\mathbb{R})$ sendo que S é simétrica definida positiva.

Neste caso uma vez que

$$\nabla^2\mathcal{L}\left(x^k, \lambda^k\right)_{(n+l)\times(n+l)}, \quad \begin{bmatrix} x \\ \lambda \end{bmatrix}_{(n+l)\times 1}, \quad \begin{bmatrix} x^k \\ \lambda^k \end{bmatrix}_{(n+l)\times 1} \text{ e } \nabla\mathcal{L}\left(x^k, \lambda^k\right)_{(n+l)\times 1}$$

ou seja, os vetores e matrizes envolvidos cumprem os critérios exigidos pelas definições de soma e multiplicação matricial, temos que

$$\nabla\mathcal{L}\left(x^k, \lambda^k\right) = \begin{bmatrix} \nabla f(x^*) + (\nabla h(x))^T\lambda^k \\ h(x^*) \end{bmatrix} = \begin{bmatrix} Sx^* + v + A^T\lambda^k \\ Ax^* - b \end{bmatrix}, \tag{16}$$

bem como

$$\nabla^2\mathcal{L}\left(x^k, \lambda^k\right) = \nabla\left(\nabla\mathcal{L}\left(x^k, \lambda^k\right)\right) = \nabla\left(\begin{bmatrix} Sx^* + v + A^T\lambda^k \\ Ax^* - b \end{bmatrix}\right) = \begin{bmatrix} S & A^T \\ A & 0 \end{bmatrix}. \tag{17}$$

Desta forma, levando (16) e (17) em (14) obtemos

$$\begin{bmatrix} S & A^T \\ A & 0 \end{bmatrix}\begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} S & A^T \\ A & 0 \end{bmatrix}\begin{bmatrix} x^k \\ \lambda^k \end{bmatrix} - \begin{bmatrix} Sx^* + v + A^T\lambda^k \\ Ax^* - b \end{bmatrix}, \tag{18}$$

e com isso, a determinação de (x, λ) se resume a resolução do sistema Sistema Linear de Lagrange. Com isso, ao aplicar o PQS em (15), a solução é obtida em apenas uma iteração conforme (18) justifica, desde que, é claro, a Hessiana da Lagrangena em (x^k, λ^k) seja não-singular.

Vale frisar que o número de restrições é irrelevante para (14) do ponto de vista teórico, isto é, o número de linhas na matriz de restrições A pode ser qualquer número natural, não se limitando a ter a mesma ordem que S.

Além disso, convém pontuar que por mais que teoricamente o problema seja resolvido em apenas uma iteração, esta *uma iteração* pode demandar um alto custo temporal dependendo do número de variáveis, do número de restrições e do *hardware* envolvido.

Diante disso, a seção seguinte ilustra a resolução de problemas quadráticos com o PQS conforme apresentado nas linhas anteriores.

4 Exemplos

Com o objetivo de apresentar alguns exemplos numéricos da teoria exposta nas seções anteriores, bem como ter uma noção do desempenho computacional, o Algoritmo 4 foi traduzido para a linguagem Python considerando o Método de Newton para o Sistema de Lagrange nos termos de (14).

Com isso, um banco de problemas de Programação Quadrática Restrita foi criado por intermédio do algoritmo "`generating-input-data.py`" disponível em https://github.com/AugustoAlex/Mathematica-V2-PQS, sendo que a matriz S simétrica definida positiva foi criada com entradas pseudo-aleatórias [11], por meio do módulo *random* [12] da linguagem Python, conforme pode ser verificado no código-fonte supracitado.

Além disso, é pertinente mencionar as características físicas do *hardware* no qual os exemplos foram resolvidos, a saber, consistiu em um *notebook* pessoal com processador Intel(R) Core(TM) i7-5500U munido de dois núcleos e quatro *thread(s)* possuindo velocidade mínima de 500 MHz e máxima de 3000 MHz. Além de contar com dois chips de memória RAM SODIMM DDR3 de 4096MB cada, operando a 1600 MHz, bem como um SSD GIGABYTE GP-GSTFS31240GNTD.

Como características de software, utilizou-se o Python em sua versão 3.10.8 com seu interpretador padrão (`CPython`). Tal software foi executado dentro de um sistema operacional GNU/Linux, mais precisamente, `Arch Linux`.

Diante disso, seguem os exemplos.

Exemplo 1. Consideremos o problema de otimização

$$\begin{aligned} \text{Otimizar} \quad & f(x) = x^T \begin{pmatrix} 142,71099 & -123,3046 \\ -123,30464 & 108,28907 \end{pmatrix} x + \begin{bmatrix} 47 & 59 \end{bmatrix} x + 4 \\ \text{sujeito a} \quad & \begin{bmatrix} -63 & -77 \\ 2 & -48 \\ -60 & 12 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} 64 \\ 32 \\ 14 \end{bmatrix} = 0 \end{aligned}$$

Condição inicial x^1	$f(x^1) \approx$	Solução aproximada obtida (x^2)	$f(x^2) \approx$
$\begin{pmatrix} 59 & 36 \end{pmatrix}$	$61.561,63686$	$\begin{pmatrix} -0,07745 & -0,60868 \end{pmatrix}$	$-20,87660$

[11] O algoritmo para obtenção dessa matriz S foi implementado segundo sugere Ribeiro e Karas (2013), enquanto que todos os demais elementos do problema (2) também foram gerados com entradas pseudo-aleatórias.

Ainda, convém também pontuar que quando falamos de programação computacional não existe aleatoriedade no sentido estrito, mas sim obtenção de números que *parecem* aleatórios, em suma na pseudo-aleatoriedade a saída do gerador *parece* aleatória para um observador externo, conforme conceituado em Gutterman, Pinkas e Reinman (2006).

[12] Para mais detalhes a respeito deste módulo recomenda-se a leitura de https://docs.python.org/3/library/random.html?highlight=random#module-random.

Ainda, convém destacar que a obtenção da solução para este exemplo (com apenas 3 variáveis) e três restrições levou cerca de $0,00345$ segundos.

A Figura 1 ilustra o gráfico associado a este exemplo. Nele, o ponto em vermelho identifica o minimizador deste problema.

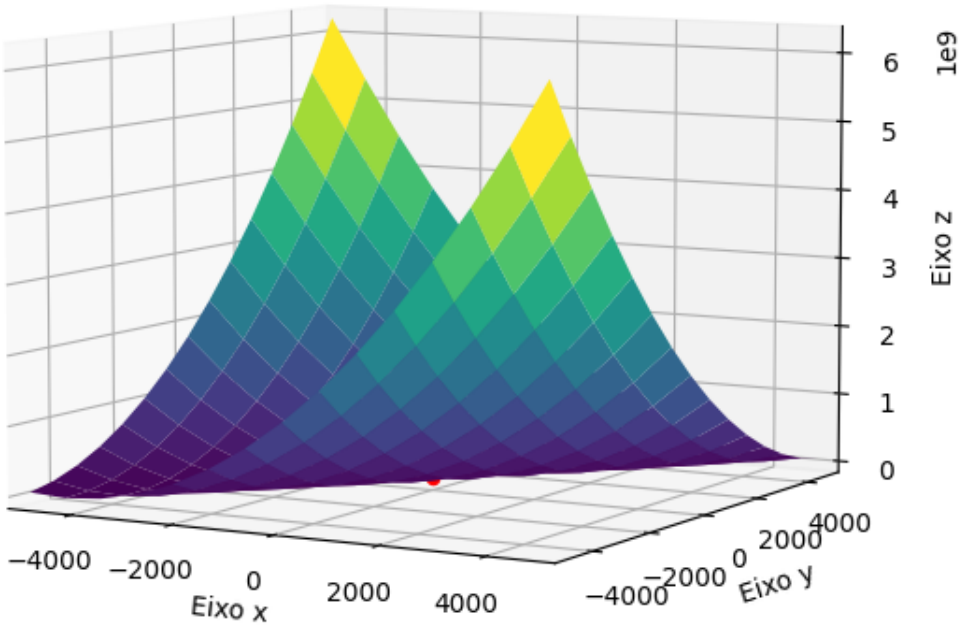

Figura 1: Gráfico associado ao problema de otimização do Exemplo 1

Na sequência, são apresentados os valores funcionais iniciais e finais, bem como o tempo de execução, de dois outros exemplos, um com 1000 variáveis e 1000 restrições. E, outro com 5000 variáveis e 5000 restrições.

Exemplo 2. Este exemplo trata-se de um problema contendo 1000 variáveis e 1000 restrições que, conforme descrito anteriormente, foi gerado pelo algoritmo "`generating-input-data.py`". Os resultados obtidos são apresentados a seguir.

$f(x^1)$ aproximado	$f(x^2)$ aproximado	Tempo (s)
$802.407.623,98335$	$1.798.430,50698$	$0,33481$

Exemplo 3. Similarmente ao exemplo anterior, este consiste de um problema contendo 5000 variáveis e 5000 restrições que foi construído pelo algoritmo "`generating-input-data.py`". Os resultados obtidos são dispostos a seguir.

$f(x^1)$ aproximado	$f(x^2)$ aproximado	Tempo (s)
$442.862.516,44263$	$98.902,74119$	$18,48927$

Conclusões

Neste trabalho escrevemos sobre o Método de Programação Quadrática Restrita, mostramos que no caso da função objetivo ser quadrática, e possuir restrições lineares (de igualdade), o problema é resolvido em apenas uma iteração. Por fim, visando exemplificar a teoria bem como ter noção do desempenho computacional, apresentamos três exemplos numéricos.

Referências

BARROS, L. N. et al. Técnica de programação semidefinida por arco viável e aplicação à maximização da frequência natural de estruturas mecânicas. Universidade Federal da Paraíba, 2017.

BUENO, H. P. *Álgebra linear*: um segundo curso. Rio de Janeiro: SBM, 2006. v. 1.

BURDEN, R. L.; FAIRES, J. D.; BURDEN, A. M. *Análise numérica*. Tradução All Tasks. Revisão técnica Helena Maria Ávila de Castro. São Paulo: Cengage Learning, 2015. v. 3.

FAÇANHA, T. S.; CARNEIRO, A. L.; FILHO, J. T. Filtro de kalman via programação quadrática. 2013.

FARIAS, C. A. Modelos de otimização de portfólios: análise comparativa e aplicações ao mercado acionário brasileiro. Universidade Federal de Viçosa, 2003.

GUIDORIZZI, H. L. *Um curso de cálculo*, vol. 2. [S.l.]: Grupo Gen-LTC, 2000.

GUTTERMAN, Z.; PINKAS, B.; REINMAN, T. *Analysis of the Linux Random Number Generator*. 2006. Cryptology ePrint Archive, Paper 2006/086. Disponível em: https://eprint.iacr.org/2006/086.

IZMAILOV, A.; SOLODOV, M. *Otimização - volume 1*: Condições de otimalidade, elementos de análise convexa e de dualidade. Rio de Janeiro: Impa, 2005.

IZMAILOV, A.; SOLODOV, M. *Otimização - volume 2*: Métodos computacionais. Rio de Janeiro: Impa, 2007.

JÚNIOR, F. de Souza. Simulação numérica de otimização de projeto de compressores axiais utilizando o método da programação sequencial quadrática. 2007.

LUENBERGER, D. G.; YE, Y. *Linear and nonlinear programming*. [S.l.]: Springer, 2015. v. 4.

OJIMA, A. L.; YAMAKAMI, A. Modelo de programação quadrática para análise da movimentação logística e comercialização da soja brasileira. *Engenharia Agrícola*, SciELO Brasil, v. 26, p. 552–560, 2006.

RIBEIRO, A. A.; KARAS, E. W. *Otimização contínua*: aspectos teóricos e computacionais. São Paulo: Cengaje Learning, 2013.

STEWART, J. *Cálculo*: volume 2. EZ2 Translate. 7. ed. São Paulo: Cengage Learning, 2013.

VANDERBEI, R. J. *Linear Programming*: Foundations and extensions. 4. ed. New York: Springer, 2014.

Alex Augusto Nunes Machado
augusttoalex@gmail.com

Simone Aparecida Miloca
smiloca@gmail.com