Universidade Estadual de Campinas

Projeto Supervisionado

Um método iterativo para encontrar projeção em um politopo

Letícia de Faria Correia
Orientadora: Profª Drª Sandra Augusta Santos

Instituto de Matemática, Estatística e Computação Científica
Campinas - SP
15 de junho de 2018

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

Em algumas situações do cotidiano nos deparamos com muitos objetos com formatos diferentes. Em outras, simplesmente visualizamos o mesmo de maneiras diversas, proporcionando-nos a impressão de que se tratam de objetos distintos. Esse é o caso ilustrado na Figura 1.

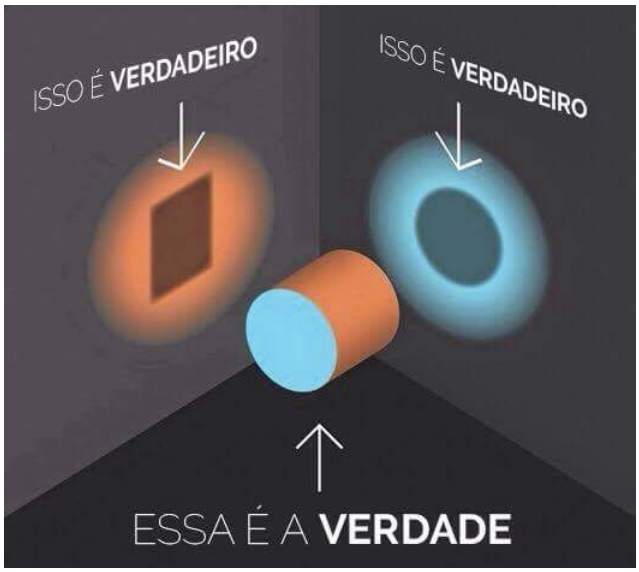


Figura 1: Diferentes pontos de vista de um mesmo objeto.(**Fonte:**https://meucerebro.com/certo-ou-errado-tudo-e-uma-questao-de-ponto-de-vista/)

Nesta perspectiva, este trabalho tem a intenção de investigar como é a Projeção em Politopos na Matemática através da análise de alguns aspectos do Algoritmo de Projeção em Politopos proposto no artigo [5].

2 O algoritmo de Projeção em Politopos

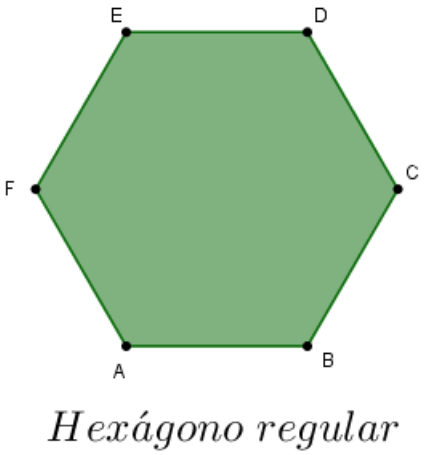
2.1 Teoria

O algoritmo investigado neste trabalho foi proposto por *Llanas e Moreno* em [5] e considera os seguintes conceitos e notações:

Politopo. Seja K um politopo delimitado pelo conjunto $\{\pi_i, i = 1, \cdots, s\}$ de hiperplanos. Denotando por $\{K_i, i = 1, \cdots, s\}$ o semiplano associado, o politopo é

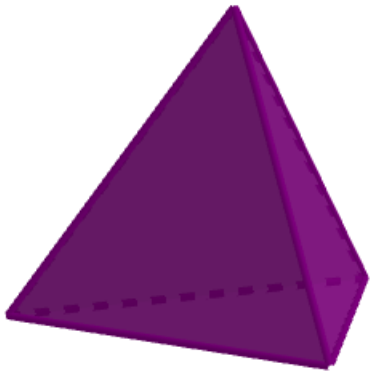
descrito por:

$$K = \bigcap_{i=1}^{s} K_i.$$



Hexágono regular

Figura 2: Politopo no R^2.



Tetraedro

Figura 3: Politopo no R^3.

Cone Normal. Para todo $x \in K$, o cone normal a K em x é definido como:

$$N_K(x) = \{z \,:\, z \cdot (y - x) \leq 0, \;\; \forall y \in K\}$$

onde $\cdot$ é o produto interno em R^n.

Projeção no politopo K. A projeção de z em K é dada por $x = P_K(z)$ se, e somente se,

$$(z - x) \cdot (y - x) \leq 0, \;\; \forall y \in K.$$

Considerando a definição de cone normal, temos que valem os seguintes resultados, cujas provas podem ser vistas em [4]:

$$x = P_K(z) \;\; \Leftrightarrow \;\; z - x \in N_K(x). \tag{1}$$

Lema 1. Se para todo $i = 1, \cdots, s$, K_i é um conjunto convexo fechado em R^n, e o interior de $K = \bigcap_{i=1}^{s} K_i$ é não vazio, então

$$N_K(x) = \sum_{i=1}^{s} N_{K_i}(x), \;\; \forall x \in K.$$

Lema 2. Seja K um conjunto convexo fechado em R^n. Então $\forall x \in K$, temos que

$$p \in N_K(x) \;\; \Leftrightarrow \;\; x = P_K(x + \lambda p), \;\; \forall \lambda > 0. \tag{2}$$

Reescrevendo o resultado 1, temos:

$$z - x = p, \;\; p \in N_K(x). \tag{3}$$

Deste modo, considerando o **Lema 1**, temos:

$$z - x = p, \quad p \in \sum_{i=1}^{s} N_{K_i}(x).$$

De acordo com os resultados anteriores, podemos escrever p como:

$$p = \sum_{i=1}^{s} p_i,$$

onde $p_i \in N_{K_i}(x)$. Pelo **Lema 2**, temos que a relação 1 pode ser escrita como:

$$z - x = \sum_{i=1}^{s} p_i. \tag{4}$$

Temos, portanto,

$$z - \sum_{i=1}^{s} p_i = x = P_K(z),$$

em que $P_K(z)$ denota a projeção de z no politopo K.

Partindo da relação 2, temos

$$x = P_{K_i}(x + \lambda p_i), i = 1, \cdots, s,$$

em que λ é um escalar positivo.

Após alguma manipulação algébrica, chegamos na seguinte relação do tipo ponto fixo, e na qual se baseará o algoritmo:

$$p_i = \frac{I - P_{K_i}}{\lambda}(x + \lambda p_i), \;\; i = 1, \cdots, s. \tag{5}$$

2.2 O algoritmo

Embasados nos resultados acima, temos:

Algoritmo. Dados o vetor a ser projetado $z \in R^n$, o escalar $\lambda \in R_+$ e $p_1^0, \cdots, p_s^0$, vetores arbitrários em R^n, tome $m = 0$ e calcule $P_K(z)$ fazendo iterativamente os seguintes passos:

$$z - x^m = \sum_{i=1}^{s} p_i^m, \tag{6}$$

$$p_i^{m+1} = \frac{I - P_{K_i}}{\lambda}(x^m + \lambda p_i^m), \;\; i = 1, \cdots, s, \tag{7}$$

$$m = m + 1.$$

Para cada iteração do algoritmo, temos que encontrar s projeções diretas p_i em cada um dos K_i hiperplanos que compõem K para obter, iterativamente, a projeção $x = P_K(z)$.

2.3 Convergência

A convergência do algoritmo é garantida de acordo com o seguinte teorema:

Teorema 1. Se $\lambda > \frac{s}{2}$, então $x^m \;\; \rightarrow \;\; x = P_K(z)$.

A prova do resultado pode ser encontrada em [5].

3 Análise do algoritmo

Polígonos são os exemplos mais simples que temos de politopos. Considerando este fato, os primeiros objetos estudados foram um triângulo equilátero e um quadrilátero. O processo de investigação do algoritmo a partir deles consistiu em três etapas:

1. Estudo da teoria envolvida;
2. Programação do algoritmo;
3. Testes para investigar o comportamento da solução em cada um dos dois politopos.

Na primeira analisamos os resultados expostos na **seção 2** e observamos um pouco mais de perto como atuam os operadores P_{K_i} e $G_\lambda = \dfrac{(I - P_{K_i})}{\lambda}$ em função de cada politopo K. Isso porque para a implementação do algoritmo fez-se necessário compreendê-los corretamente.

3.1 A Projeção P_{K_i}

Um politopo $K \in R^n$, limitado por s hiperplanos, é regido por um conjunto de desigualdades lineares, e pode ser expresso como:

$$K = \{x \in R^n \mid Ax \leq b\}, \tag{8}$$

em que $A \in R^{s \times n}$ e $b \in R^{s \times 1}$.

Consequentemente, cada semi-espaço K_i de um politopo K é tal que

$$a_i x \leq b_i, \tag{9}$$

onde a_i é a i-ésima linha da matriz A e b_i é a i-ésima componente do vetor b.

Sendo assim, dado um ponto z tal que $z \notin K_i$, para calcularmos $x = P_{K_i}(z) = x$ e , temos que considerar as seguintes relações:

$$a_i^T x = b_i \tag{10}$$

e

$$\exists \alpha \ \text{ tal que } \ x = a_i \alpha + z, \ \ \alpha \in R, \ \ z \in R^n, \ \ a_i \in R^n \tag{11}$$

De 10 e 11 temos:

$$a_i^T(ai\alpha + z) = b_i \Rightarrow$$

$$\Rightarrow (a_i^T a_i)\alpha + a_i^T z = b_i \Rightarrow$$

$$\Rightarrow (a_i^T a_i)\alpha = b_i - a_i^T z \Rightarrow$$

$$\alpha = \frac{(b_i - a_i^T)}{(a_i^T a_i)} z \tag{12}$$

Portanto,

$$x = a_i\alpha + z = \frac{(b_i - a_i^T z)}{(a_i^T a_i)} a_i + z,$$

ou seja,

$$P_{K_i}(z) = \frac{(b_i - a_i^T z)}{(a_i^T a_i)} a_i + z, \ \forall z \notin K_i \tag{13}$$

Para $z \in K_i, P_{K_i} = z$.

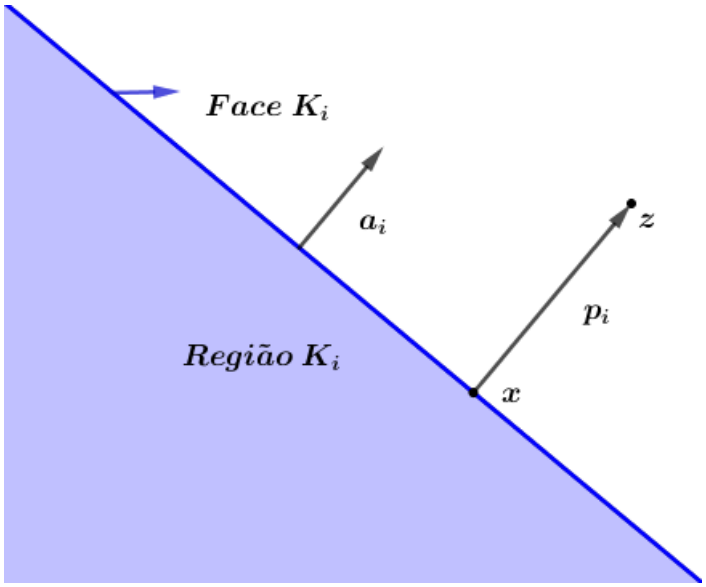


Figura 4: Projeção do ponto $z \notin K_i$ na face K_i.

3.2 O operador $G_\lambda = \dfrac{I - P_{K_i}}{\lambda}$

Para encontrar a expressão de $G_\lambda = \dfrac{I - P_{K_i}}{\lambda}$, consideremos o valor de P_{K_i} em 13 e vejamos como o operador atua em um vetor $y \in R^n$:

$$G_\lambda(y) = \frac{I - P_{K_i}}{\lambda}(y) = \frac{1}{\lambda}\left[y - \frac{(b_i - a_i^T y)}{a_i^T a_i} a_i - y\right]$$

$$G_\lambda(y) = \frac{I - P_{K_i}}{\lambda}(y) = \frac{(a_i^T y - b_i)}{(\lambda a_i^T a_i)} a_i. \tag{14}$$

3.3 O código implementado

O código para a realização dos testes foi implementado no software *GNU Octave* [2]. Para a visualização dos objetos utilizamos o *GeoGebra* [1] e o *Wolfram Mathematica* [3].

A estrutura foi montada para os testes serem realizados para qualquer politopo $K \subset R^n$ e com qualquer número s de faces, bastando apenas que o valor de s e n sejam dadas inicialmente.

Contudo, para cada teste é necessário estabelecer as desigualdades $Ax \leq b$, que definem o politopo está sujeito, ou seja, determinar a matriz $A \in R^{s \times n}$ e o vetor $b \in R^{s \times 1}$.

É importante também definir o ponto a ser projetado z, bem como P (matriz cuja i-ésima linha é o vetor p_i^0) e l (λ do algoritmo), sendo esses últimos parâmetros livres, ou seja, que podem ser determinados de maneira arbitrária.

Para a observação disso, seguem abaixo os códigos para um dos testes de cada um dos dois politopos estudados.

```
%Projecoes em poligonos (RN)
s=3;%qtde de hiperplanos do politopo
n=2;%dimensao do politopo

%Exemplo 1: Triangulo
z=[2; 0];%z dado
P=[1 0; 1/2 1/2; 0 1]; %P matriz onde cada linha i corresponde e ao
   valor de pi^0 dado
A=[-sqrt(3) 1; sqrt(3) 1; 0 -1]; %A precisa ser determinada
b=[0; sqrt(3); 0]; %b precisa ser determinado

x=zeros(n,1); %x e a projecao de z no politopo K
xant=z; %x na iteracao anterior
aux1=zeros(n,1); %vetor auxiliar no Passo 1
aux3=x-xant; % diferenca entre x e xant.
auxproj=zeros(n,1); %vetor auxiliar no Passo 2
aux2=zeros(n,1);%vetor auxiliar no Passo 2
l=2; %valor de lambda
tol=0.00000001; %tolerancia
itmax=100; %numero maximo de iteracoes caso ainda nao tenha
   encontrado

c=zeros(itmax,1);%Vetor auxiliar para plotar |x(m+1) - x(m)|

P %Imprime P na janela de comandos

j = 1;
while ((norm(aux3,inf)>=tol) && (j < itmax))

  %Passo 1: Encontrar xn
        aux1=zeros(n,1);
        for i=1:s
            aux1= aux1+P(i,:)';
        end
        xant=x;
        x=z-aux1;
        aux3 = x-xant;
```

```
36
37          if(j>2)
38          c(j) = norm(aux3,inf);
39          endif
40
41          for(v=1:n)
42              printf("Na iteracao %i, temos x(%i)= %f\n", j, v, x(v));
43          end
44
45      P'
46
47      %Passo 2: Encontrar os Pi na iteracao n+1
48          for k=1:s
49          %Calculo do vetor x+l*pi;
50              auxproj=x+l*P(k,:)';
51          %Verificando se o vetor calculado esta na regiao que satisfaz
                as desigualdades
52              aux2=A(k,:)*auxproj;
53
54              if(aux2<=b(k))
55                 P(k,:) =zeros(n,1);
56              else
57                 P(k,:)=((A(k,:)*auxproj-b(k))/(l*A(k,:)*A(k,:)'))*A(k
                       ,:);
58              end
59          end
60
61          j++
62  endwhile
63
64  %norma de x-xant
65  semilogy(c(3:j-1));
```

```
%Projecoes em poligonos (RN)
s=4;%qtde de hiperplanos do politopo
n=2;%dimensao do politopo

%Exemplo 2: Quadrilatero
z=[11;0];
P =[2 1; 4 11; 7 -1; 1 1];%P matriz onde cada linha i corresponde ao
    valor de pi^0 dado
A=[7 9; 1 -6; -1 -1; -5 7]; %A precisa ser determinada
b=[50; 29; 6; 18]; %b precisa ser determinado

x=zeros(n,1); %x e a projecao de z no politopo K
xant=z; %x na iteracao anterior
aux1=zeros(n,1); %vetor auxiliar no Passo 1
aux3=x-xant; % diferenca entre x e xant.
auxproj=zeros(n,1); %vetor auxiliar no Passo 2
aux2=zeros(n,1);%vetor auxiliar no Passo 2
l=2.1; %valor de lambda
tol=0.00000001; %tolerancia
itmax=100; %numero maximo de iteracoes caso ainda nao tenha
    encontrado

c=zeros(itmax,1);%Vetor auxiliar para plotar |x(m+1) - x(m)|

P %Imprime P na janela de comandos

j = 1;
while ((norm(aux3,inf)>=tol) && (j < itmax))

  %Passo 1: Encontrar xn
        aux1=zeros(n,1);
        for i=1:s
            aux1= aux1+P(i,:)';
        end
        xant=x;
        x=z-aux1;
        aux3 = x-xant;
```

```
36
37          if(j>2)
38          c(j) = norm(aux3,inf);
39          endif
40
41          for(v=1:n)
42              printf("Na iteracao %i, temos x(%i)= %f\n", j, v, x(v));
43          end
44
45      P'
46
47      %Passo 2: Encontrar os Pi na iteracao n+1
48          for k=1:s
49          %Calculo do vetor x+l*pi;
50              auxproj=x+l*P(k,:)';
51          %Verificando se o vetor calculado esta na regiao que satisfaz
                as desigualdades
52              aux2=A(k,:)*auxproj;
53
54              if(aux2<=b(k))
55                 P(k,:) =zeros(n,1);
56              else
57                 P(k,:)=((A(k,:)*auxproj-b(k))/(l*A(k,:)*A(k,:)'))*A(k
                       ,:);
58              end
59          end
60
61          j++
62  endwhile
63
64  %norma de x-xant
65  semilogy(c(3:j-1));
```

3.4 Triângulo Equilátero

Seja o politopo T um triângulo equilátero dado pela intersecção dos seguintes hiperplanos, ou retas do plano:

Hiperplano 1: $-\sqrt{3}x_1 + x_2 \leq 0.$
Hiperplano 2: $\sqrt{3}x_1 + x_2 \leq \sqrt{3}.$
Hiperplano 3: $-x_2 \leq 0.$

Temos que T é dado por todos os pontos $x = (x_1, x_2)$ que satisfazem a seguinte desigualdade matricial:

$$Ax \leq b,$$

em que

$$A = \begin{bmatrix} -\sqrt{3} & 1 \\ \sqrt{3} & 1 \\ 0 & -1 \end{bmatrix} \text{ e } b = \begin{bmatrix} 0 \\ \sqrt{3} \\ 0 \end{bmatrix}$$

A Figura 5 ilustra o politopo T.

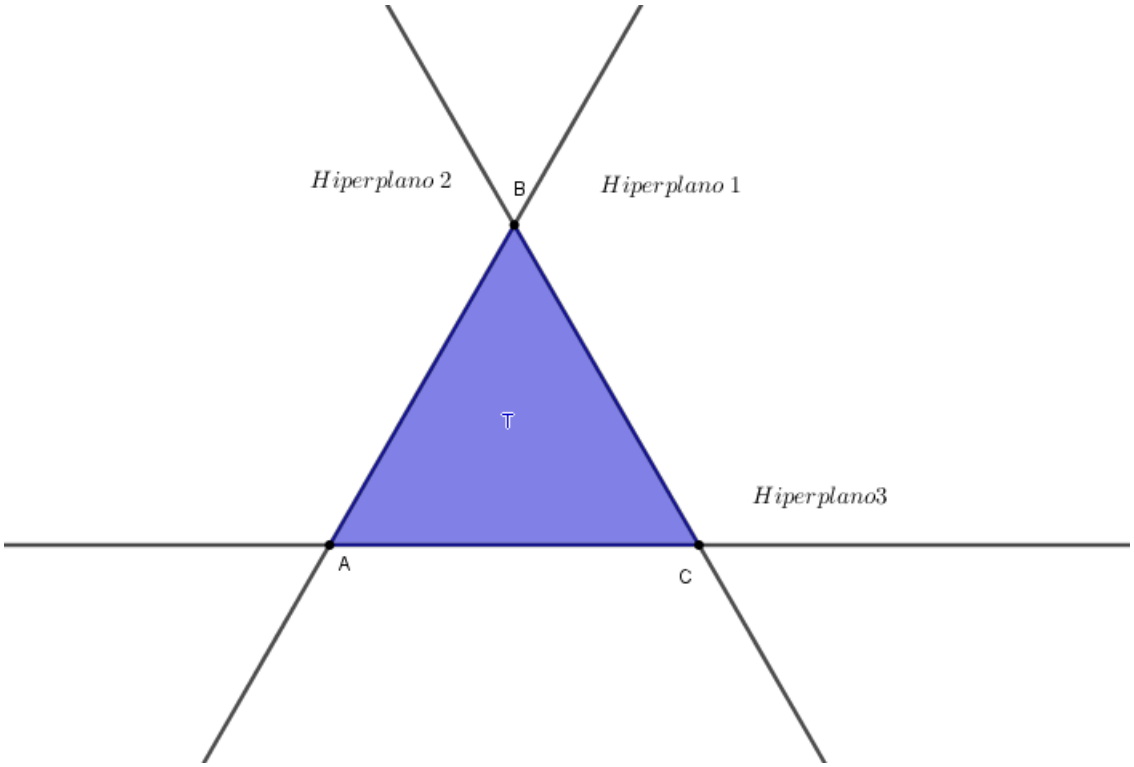


Figura 5: Politopo T : Triângulo Equilátero.

3.4.1 Teste 1: Comportamento da solução ao variar o parâmetro λ

Seja o problema de encontrar a projeção $x = P_T(z)$ no politopo T, com uma tolerância de 10^{-8} (medida por $|x^{m+1} - x^m| < 10^{-8}$), sendo $z = (2,0)^T$, $p_1^0 = (1,0)^T$, $p_2^0 = (0.5,0.5)^T$ e $p_3^0 = (0,1)^T$.

Os resultados obtidos foram:

λ	nº de iterações	$(x^*)^T$	$(p_1^*)^T$	$(p_2^*)^T$	$(p_3^*)^T$
0.5	–	diverge	-	-	-
1.0	31	$(1,0)$	$(0,0)$	$(1.00000, 0.57735$	$(0,-0.57735)$
1.5	46	$(1,0)$	$(0,0)$	$(1.00000, 0.57735$	$(0,-0.57735)$
1.6	49	$(1,0)$	$(0,0)$	$(1.00000, 0.57735$	$(0,-0.57735)$
1.7	52	$(1,0)$	$(0,0)$	$(1.00000, 0.57735$	$(0,-0.57735)$
1.8	55	$(1,0)$	$(0,0)$	$(1.00000, 0.57735$	$(0,-0.57735)$
1.9	58	$(1,0)$	$(0,0)$	$(1.00000, 0.57735$	$(0,-0.57735)$
2.0	62	$(1,0)$	$(0,0)$	$(1.00000, 0.57735$	$(0,-0.57735)$
10	291	$(1,0)$	$(0,0)$	$(1.00000, 0.57735$	$(0,-0.57735)$
100	2466	$(1,0)$	$(0,0)$	$(1.00000, 0.57735$	$(0,-0.57735)$

Tabela 1: Solução $(x^*)^T$ e os vetores finais obtidos $(p_i^*)^T, i = 1,2,3$ durante o Teste 1 no politopo T.

3.4.2 Teste 2: Comportamento da solução ao variar os vetores p_i^0, $i = 1,2,3$

Seja o problema de encontrar a projeção $x = P_T(z)$ no politopo T, com uma tolerância de 10^{-8}, sendo $z = (2,0)^T$ e $\lambda = 2.0$.

- Para os valores $p_1^0 = (1,0)^T, p_2^0 = (0.5,0.5)^T$ e $p_3^0 = (0,1)^T$, os resultados obtidos foram:

$(x^*)^T$	$(p_1^*)^T$	$(p_2^*)^T$	$(p_3^*)^T$	nº de iterações
$(1,0)$	$(0,0)$	$(1.00000, 0.57735$	$(0,-0.57735)$	62

Tabela 2: Comportamento da solução $(x^*)^T$ para os vetores iniciais $p_1^0 = (1,0)^T, p_2^0 = (0.5,0.5)^T$ e $p_3^0 = (0,1)^T$ no politopo T durante o Teste 2.

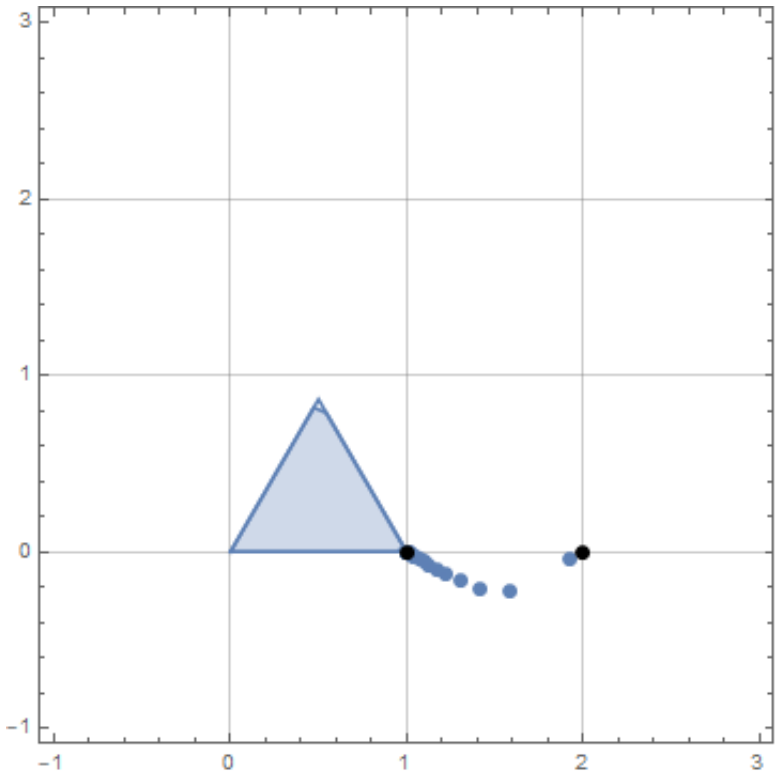


Figura 6: Visualização dos iterados x^m para a inicialização $p_1^0 = (1,0)^T, p_2^0 = (0.5,0.5)^T$ e $p_3^0 = (0,1)^T$ ao longo das 62 iterações durante o Teste 2 no politopo T.

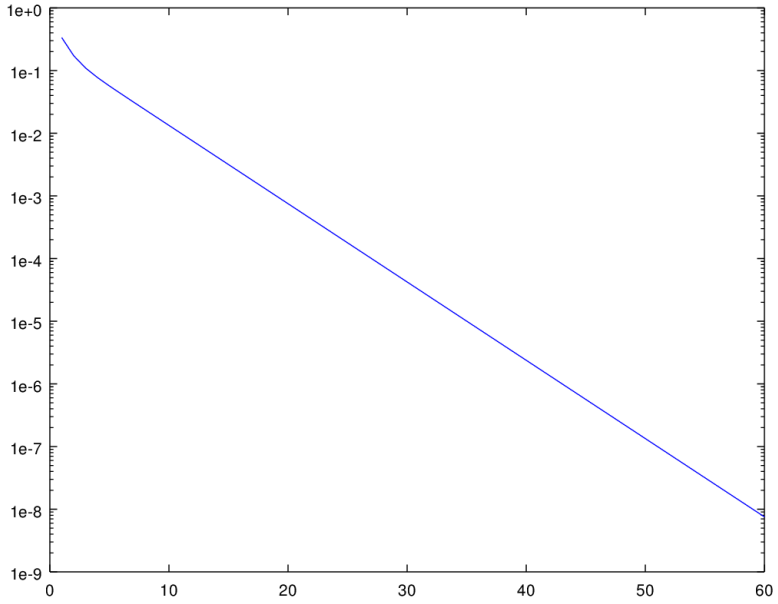


Figura 7: $|x^{m+1} - x^m|$ na escala logaritmica durante o Teste 2 no politopo T para a inicialização $p_1^0 = (1,0)^T, p_2^0 = (0.5,0.5)^T$ e $p_3^0 = (0,1)^T$ ao longo das 62 iterações.

- Para os valores $p_1^0 = p_2^0 = p_3^0 = (0,0)^T$, os resultados obtidos foram:

$(x^*)^T$	$(p_1^*)^T$	$(p_2^*)^T$	$(p_3^*)^T$	nº de iterações
$(1,0)$	$(0,0)$	$(1.00000, 0.57735$	$(0,-0.57735)$	61

Tabela 3: Comportamento da solução $(x^*)^T$ para os vetores iniciais $p_1^0 = p_2^0 = p_3^0 = (0,0)^T$ no politopo T durante o Teste 2.

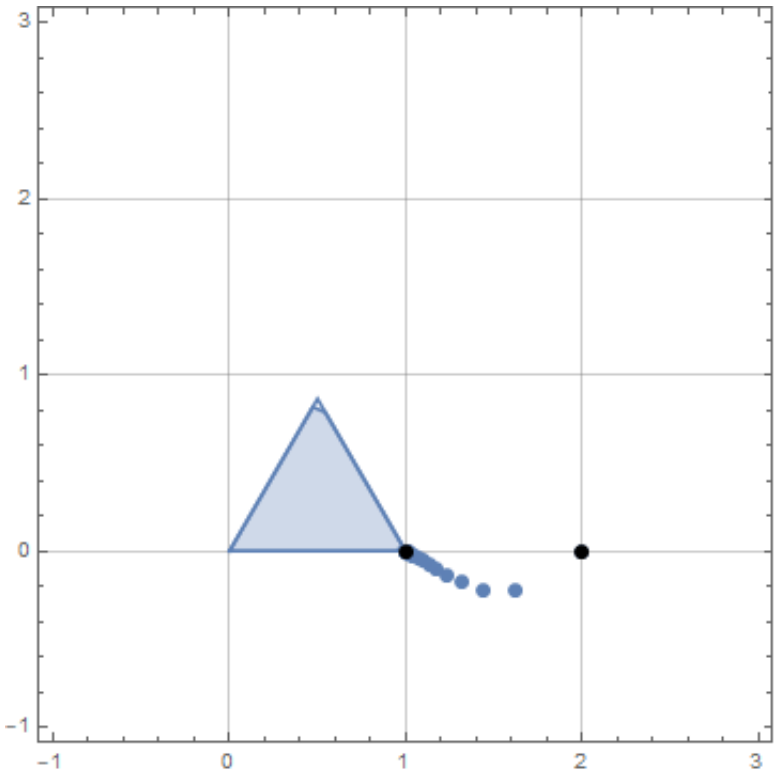


Figura 8: Visualização dos iterados x^m para a inicialização $p_1^0 = p_2^0 = p_3^0 = (0,0)^T$ ao longo das 61 iterações durante o Teste 2 no politopo T.

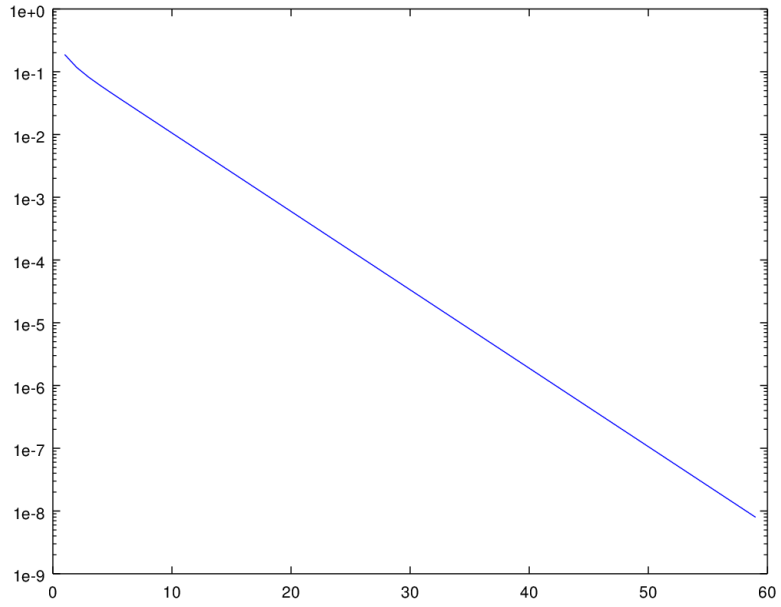


Figura 9: $|x^{m+1} - x^m|$ na escala logaritmica durante o Teste 2 no politopo T para a inicialização $p_1^0 = p_2^0 = p_3^0 = (0,0)^T$ ao longo das 61 iterações.

- Para os valores $p_1^0 = (2,1)^T, p_2^0 = (4,11)^T$ e $p_3^0 = (7,-1)^T$, os resultados obtidos foram:

$(x^*)^T$	$(p_1^*)^T$	$(p_2^*)^T$	$(p_3^*)^T$	nº de iterações
$(1,0)$	$(0,0)$	$(1.00000, 0.57735$	$(0,-0.57735)$	73

Tabela 4: Comportamento da solução $(x^*)^T$ para os vetores iniciais $p_1^0 = (2,1)^T, p_2^0 = (4,11)^T$ e $p_3^0 = (7,-1)^T$ no politopo T durante o Teste 2.

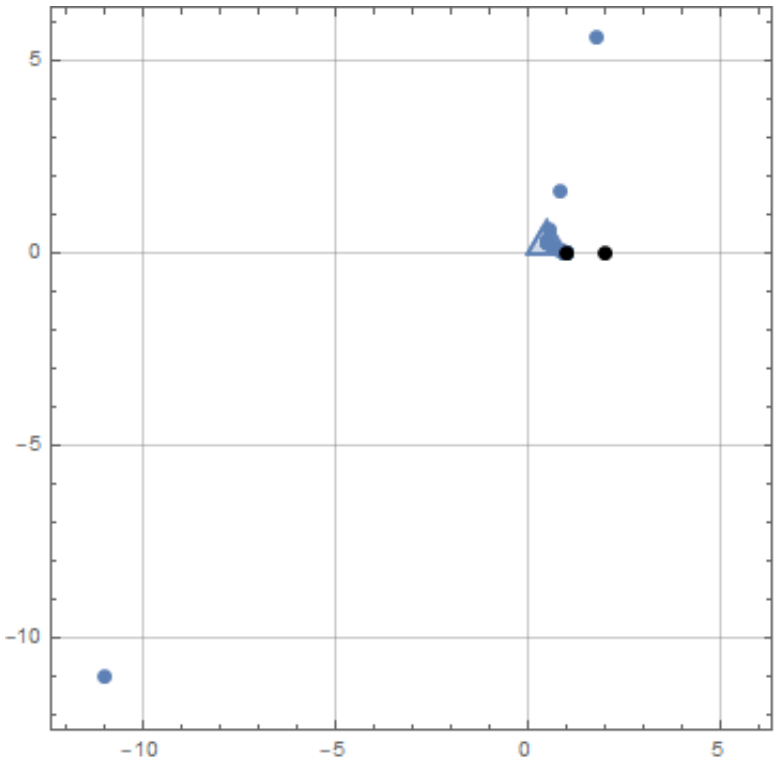


Figura 10: Visualização dos iterados x^m para a inicialização $p_1^0 = (2,1)^T, p_2^0 = (4,11)^T$ e $p_3^0 = (7,-1)^T$ ao longo das 73 iterações durante o Teste 2 no politopo T.

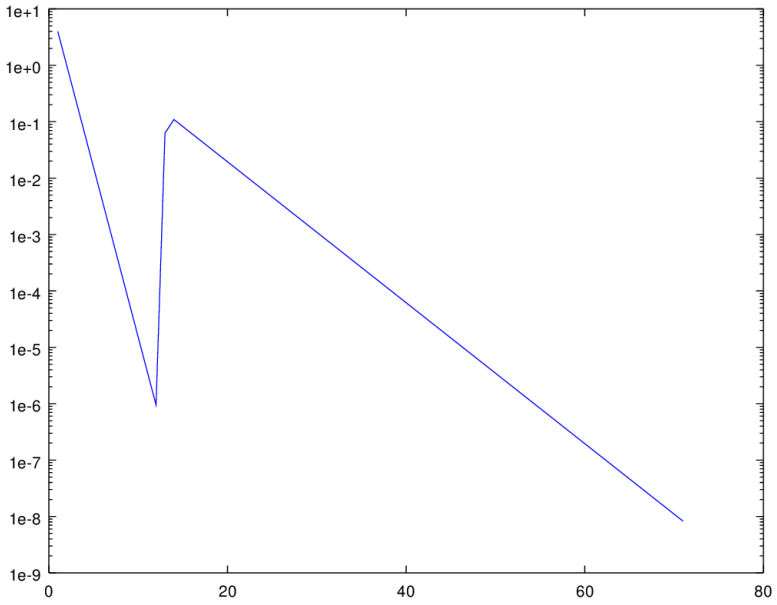


Figura 11: $|x_{m+1} - x_m|$ na escala logaritmica durante o Teste 2 no politopo T para a inicialização $p_1^0 = (2,1)^T, p_2^0 = (4,11)^T$ e $p_3^0 = (7,-1)^T$ ao longo das 73 iterações.

- Para os valores $p_1^0 = (12,0)^T, p_2^0 = (-1,-3)^T$ e $p_3^0 = (0,7)^T$, os resultados obtidos foram:

$(x^*)^T$	$(p_1^*)^T$	$(p_2^*)^T$	$(p_3^*)^T$	nº de iterações
$(1,0)$	$(0,0)$	$(1.00000, 0.57735$	$(0,-0.57735)$	62

Tabela 5: Comportamento da solução $(x^*)^T$ para os vetores iniciais $p_1^0 = (12,0)^T, p_2^0 = (-1,-3)^T$ e $p_3^0 = (0,7)^T$ no politopo T durante o Teste 2.

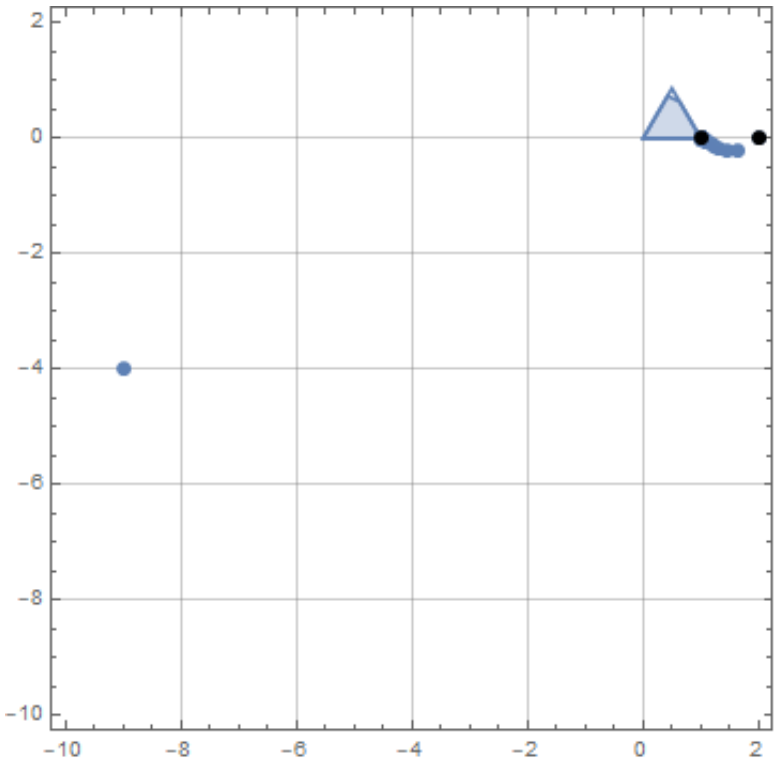


Figura 12: Visualização dos iterados x^m para a inicialização $p_1^0 = (12,0)^T, p_2^0 = (-1,-3)^T$ e $p_3^0 = (0,7)^T$ ao longo das 62 iterações durante o Teste 2 no politopo T.

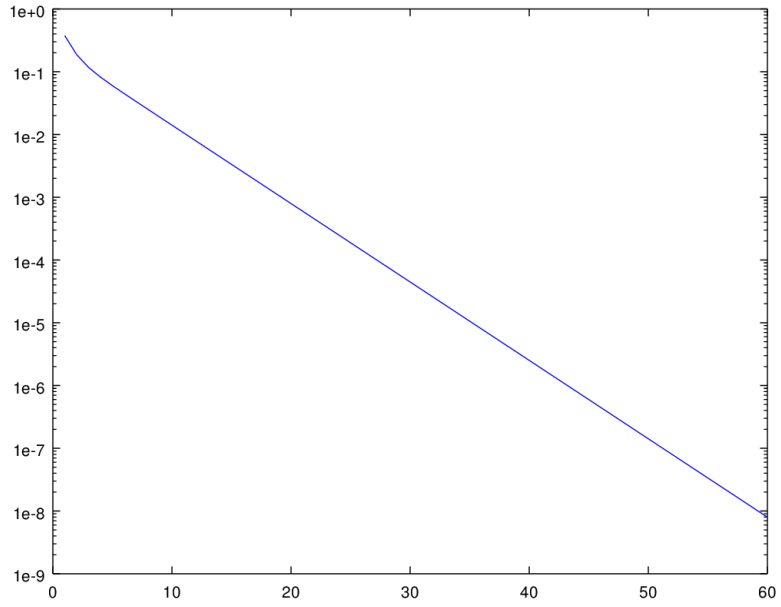


Figura 13: $|x_{m+1} - x_m|$ na escala logaritmica durante o Teste 2 no politopo T para a inicialização $p_1^0 = (12, 0)^T, p_2^0 = (-1, -3)^T$ e $p_3^0 = (0, 7)^T$ ao longo das 62 iterações.

3.5 Quadrilátero

Agora o politopo Q é o quadrilátero dado pela intersecção dos seguintes hiperplanos:

Hiperplano 1: $7x_1 + 9x_2 \leq 50$.
Hiperplano 2: $x_1 - 6x_2 \leq 29$.
Hiperplano 3: $-x_1 - x_2 \leq 6$.
Hiperplano 4: $-5x_1 + 7x_2 \leq 18$.

Temos que Q é dado por todos os pontos $x = (x_1, x_2)$ que satisfazem à seguinte desigualdade matricial:

$$Ax \leq b,$$

em que

$$A = \begin{bmatrix} 7 & 9 \\ 1 & -6 \\ -1 & -1 \\ -5 & 7 \end{bmatrix} \text{ e } b = \begin{bmatrix} 50 \\ 29 \\ 6 \\ 18 \end{bmatrix}$$

A Figura 14 ilustra o politopo Q.

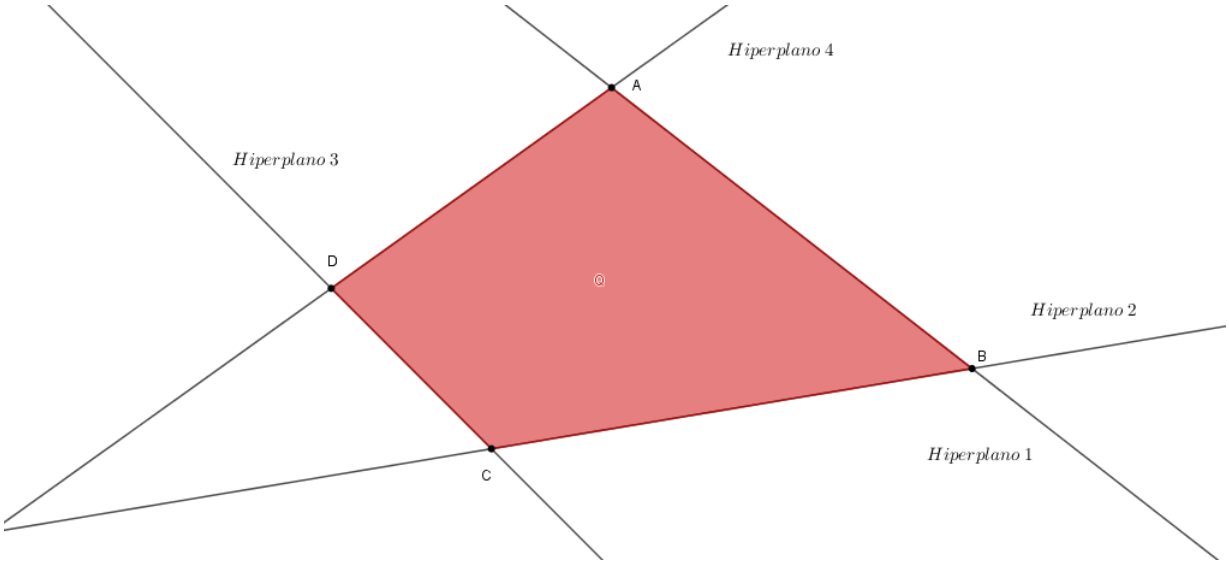


Figura 14: Politopo Q : Quadrilátero.

3.5.1 Teste 1: Comportamento da solução ao variar o parâmetro λ

Seja o problema de encontrar a projeção $x = P_Q(z)$ no politopo Q, com uma tolerância de 10^{-8} (medida por $|x^{m+1} - x^m| < 10^{-8}$), sendo $z = (11, 0)^T$, $p_1^0 = (2, 1)^T$, $p_2^0 = (4, 11)^T$, $p_3^0 = (7, -1)^T$ e $p_4^0 = (1, 1)^T$. Temos o seguinte comportamento:

λ	nº de iterações	$(x^*)^T$	$(p_1^*)^T$	$(p_2^*)^T$	$(p_3^*)^T$	$(p_4^*)^T$
0.5	–	$diverge$	–	–	–	–
1.0	7	(9.5462, −1.8692)	(1.45385, 1.86923)	(0, 0)	(0, 0)	(0, 0)
1.5	20	(9.5462, −1.8692)	(1.45385, 1.86923)	(0, 0)	(0, 0)	(0, 0)
2.0	30	(9.5462, −1.8692)	(1.45385, 1.86923)	(0, 0)	(0, 0)	(0, 0)
2.1	32	(9.5462, −1.8692)	(1.45385, 1.86923)	(0, 0)	(0, 0)	(0, 0)
2.2	34	(9.5462, −1.8692)	(1.45385, 1.86923)	(0, 0)	(0, 0)	(0, 0)
2.3	35	(9.5462, −1.8692)	(1.45385, 1.86923)	(0, 0)	(0, 0)	(0, 0)
2.4	37	(9.5462, −1.8692)	(1.45385, 1.86923)	(0, 0)	(0, 0)	(0, 0)
2.5	39	(9.5462, −1.8692)	(1.45385, 1.86923)	(0, 0)	(0, 0)	(0, 0)
10	160	(9.5462, −1.8692)	(1.45385, 1.86923)	(0, 0)	(0, 0)	(0, 0)
100	1287	(9.5462, −1.8692)	(1.45385, 1.86923)	(0, 0)	(0, 0)	(0, 0)

Tabela 6: Solução $(x^*)^T$ e os vetores finais obtidos $(p_i^*)^T, i = 1, 2, 3, 4$ durante o Teste 1 no politopo Q.

3.5.2 Teste 2: Comportamento da solução ao variar os vetores $p_i^0, i = 1, 2, 3, 4$

Seja o problema de encontrar a projeção $x = P_Q(z)$ no politopo Q, com uma tolerância de 10^{-8}, sendo $z = (11, 0)^T$ e $\lambda = 3.0$.

- Para os valores $p_1^0 = (2, 1)^T$, $p_2^0 = (4, 11)^T$, $p_3^0 = (7, -1)^T$ e $p_4^0 = (1, 1)^T$, os resultados obtidos foram:

$(x^*)^T$	$(p_1^*)^T$	$(p_2^*)^T$	$(p_3^*)^T$	$(p_4^*)^T$	nº de iterações
(9.5462, −1.8692)	(1.45385, 1.86923)	(0, 0)	(0, 0)	(0, 0)	48

Tabela 7: Comportamento da solução $(x^*)^T$ para a inicialização $p_1^0 = (2, 1)^T$, $p_2^0 = (4, 11)^T$, $p_3^0 = (7, -1)^T$ e $p_4^0 = (1, 1)^T$ no politopo Q durante o Teste 2.

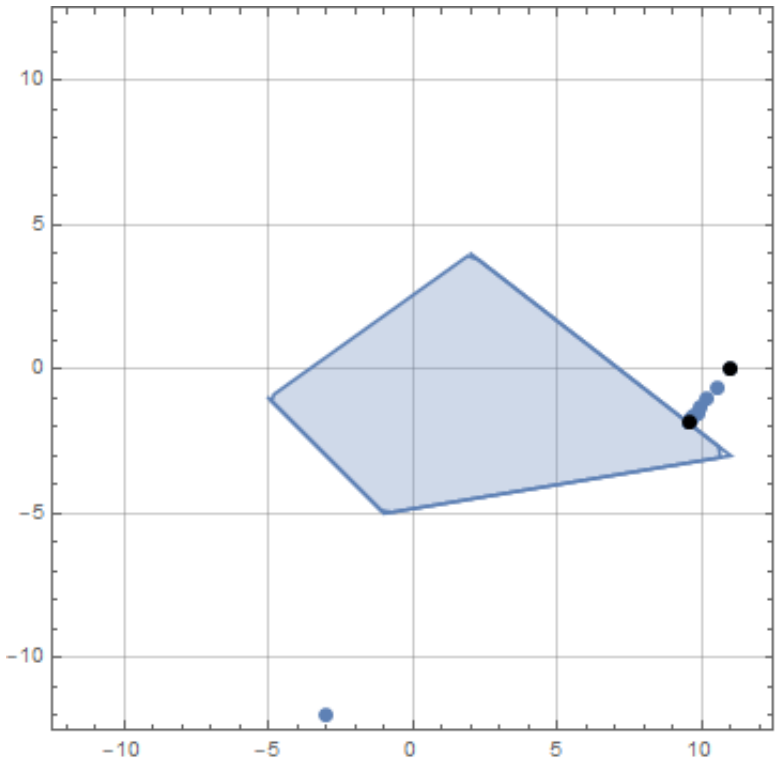


Figura 15: Visualização dos iterados x^m para a inicialização $p_1^0 = (2,1)^T, p_2^0 = (4,11)^T, p_3^0 = (7,-1)^T$ e $p_4^0 = (1,1)^T$ ao longo das 48 iterações durante o Teste 2 no politopo Q.

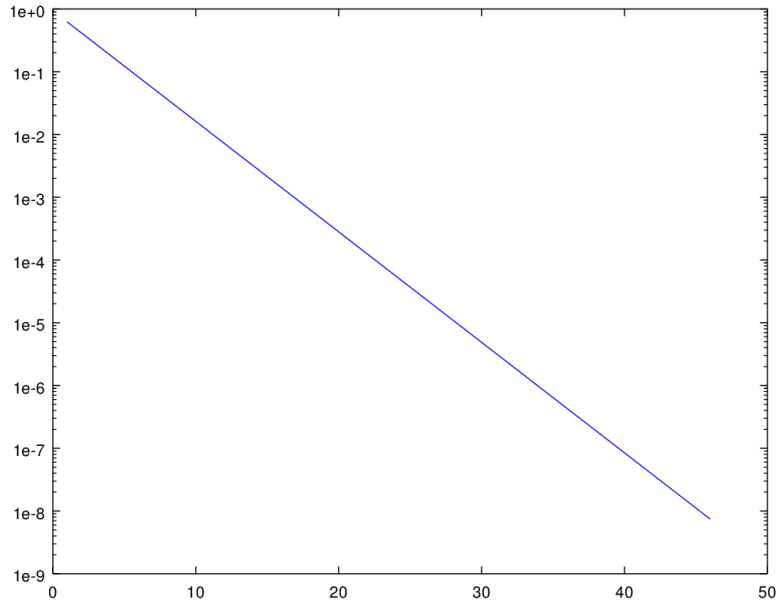


Figura 16: $|x_{m+1} - x_m|$ na escala logaritmica durante o Teste 2 no politopo Q para a inicialização $p_1^0 = (2,1)^T, p_2^0 = (4,11)^T, p_3^0 = (7,-1)^T$ e $p_4^0 = (1,1)^T$ ao longo das 48 iterações.

- Para os valores $p_1^0 = (12,0)^T, p_2^0 = (-1,-3)^T, p_3^0 = (0,7)^T$ e $p_4^0 = (2,5)^T$, os resultados obtidos foram:

$(x^*)^T$	$(p_1^*)^T$	$(p_2^*)^T$	$(p_3^*)^T$	$(p_4^*)^T$	nº de iterações
$(9.5462, -1.8692)$	$(1.45385, 1.86923)$	$(0,0)$	$(0,0)$	$(0,0)$	50

Tabela 8: Comportamento da solução $(x^*)^T$ para a inicialização $p_1^0 = (12,0)^T, p_2^0 = (-1,-3)^T, p_3^0 = (0,7)^T$ e $p_4^0 = (2,5)^T$ no politopo Q durante o Teste 2.

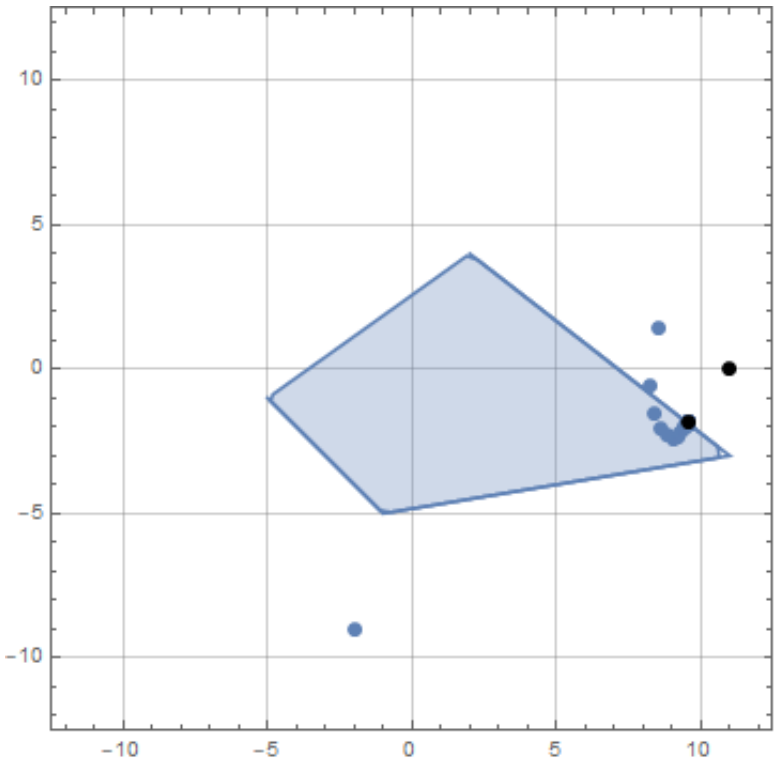


Figura 17: Visualização dos iterados x^m para a inicialização $p_1^0 = (12,0)^T, p_2^0 = (-1,-3)^T, p_3^0 = (0,7)^T$ e $p_4^0 = (2,5)^T$ ao longo das 50 iterações durante o Teste 2 no politopo Q.

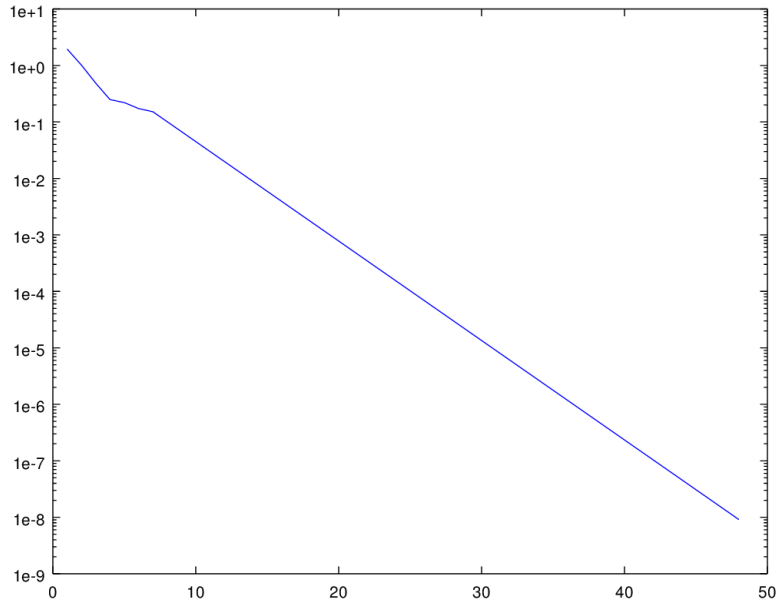


Figura 18: $|x_{m+1} - x_m|$ na escala logaritmica durante o Teste 2 no politopo Q para a inicialização $p_1^0 = (12,0)^T, p_2^0 = (-1,-3)^T, p_3^0 = (0,7)^T$ e $p_4^0 = (2,5)^T$ ao longo das 50 iterações.

4 Resultados obtidos e considerações finais

4.1 A variação de λ

Os testes realizados permitiram verificar que, de fato, o algoritmo converge para valores de $\lambda > \frac{s}{2}$ mas não somente para esses valores, como é o que ocorre com o valor $\lambda = 1$ no Teste 1 tanto no cálculo da projeção no politopo T como no politopo Q.

Os vetores finais obtidos $(x^*)^T$ e $(p_i^*)^T, i = 1, \cdots, s$ são os mesmos para todos os distintos valores de λ que fazem o algoritmo convergir.

Ainda foi possível perceber que quanto maior o valor do parâmetro, menor é a velocidade de convergência do algoritmo, ou seja, são necessárias mais iterações para obter o mesmo resultado com a tolerância dada.

4.2 A variação dos vetores $p_i^0,\ i = 1, \cdots, s$

Do mesmo modo que os distintos valores de λ fazem o algoritmo convergir para as mesmas soluções $(x^*)^T$ e $(p_i^*)^T, i = 1, \cdots, s$, a variação dos vetores iniciais $p_i^0,\ i = 1, \cdots, s$ tambem.

Alem disso, os testes permitiram constatar que o modo como isso ocorre é distinto: em alguns casos a sequência x^m entra na região interna do politopo (Figura 10 e Figura 17), e em outros não (Figura 6, Figura 8, Figura 12 e Figura 15).

Com o auxílio dos gráficos de $|x_{m+1} - x_m|$ na escala logaritmica tambem podemos perceber que, para as situações em que a sequência x^m entra na região interna do politopo, os primeiros sofrem uma pertubação e depois passam a ser lineares como nos outros casos (Figura 11 e Figura 18).

Perceber que a sequência x^m pode entrar na região interna causou uma certa estranheza a princípio pelo fato de ser um algoritmo de projeção e o ponto a ser projetado estar fora da mesma região. No entanto, essa diferença ocorre pois o algoritmo foi construído de maneira a satisfazer 2 e 3.

4.3 Planos futuros

O estudo desenvolvido nesse projeto foi muito interessante e enriquecedor. Isso, pois, apesar das dificuldades ou por causa delas, foi necessário um maior aprofundamento em conceitos considerados básicos de *Geometria Analítica* e de *Algebra Linear* e em alguns recursos computacionais como o *GNU Octave*, o *GeoGebra* e o *Wolfram Mathematica*.

No entanto, o curto período de tempo possibilitou apenas algumas análises das muitas que foram desejadas. Ficam ainda os planos futuros:

- Fazer um estudo mais amplo da convergência do algoritmo;
- Propor testes mais elaborados;
- Automatizar a geração de politopos 2D e 3D;
- Explorar a geração aleatória de pontos a serem projetados e discutir regiões de convergência.

Referências

[1] Geogebra. http://ogeogebra.com.br/site/. último acesso em: 13/06/2018.

[2] Gnu octave. https://www.gnu.org/software/octave/. último acesso em: 13/06/2018.

[3] Wolfram mathematica. https://www.wolfram.com/mathematica/. último acesso em: 13/06/2018.

[4] J.B. Hiriart-Urruty and C. Lemarechal. *Convex Analysis and Minimization Algorithms I*. Springer Berlin Heidelberg, 1993.

[5] C. Llanas, B. e Moreno. Finding the projection on a polytope: An iterative method. *Computers and Mathematics with Applications*, 32(8):33 – 39, 1996.