Universidade Federal de São Paulo

Instituto de Ciência e Tecnologia

Projeto Computacional 4
Otimização Linear - Programação Linear

Carlos Eduardo de Moraes Ferreira

Laís da Silva

Lucas Eduardo Nogueira Gonçalves

Marila Torres de Aguiar

Professor: Luís Felipe Bueno

São José dos Campos

Julho, 2019

1. INTRODUÇÃO

Dado os problemas de programação linear abaixo, resolva esses com os diferentes métodos vistos em sala, sendo eles o primal dual e o preditor corretor. A partir do ponto inicial escolhido (x^0), calculamos ρ^0,λ^0e μ^0. O método Primal-Dual foi originalmente desenvolvido como um algoritmo teórico-polinomial para Programação Linear, a principal ideia do método é mover-se através de uma sequência estritamente viável de soluções primárias e duais que se aproximam cada vez mais da satisfação das condições complementares de negligência. Para isso, aproxima-se o sistema não linear por um sistema linear, esta maneira para determinar equações não lineares linearizando-as, é conhecida como método de Newton, em cada iteração do método Primal-Dual, realiza-se uma iteração de Newton para encontrar um ponto mais próximo da solução ótima. O método Preditor Corretor é a modificação do método Primal-Dual que reduz o número de iterações requeridas pelo método com apenas um pequeno aumento no custo para cada iteração.

Vamos então considerar o seguinte conjunto de restrições:

$$x + y \leq 0.5;$$
$$-x + y \leq 0.1;$$
$$-x + 4y \leq 0.6;$$
$$x \leq 0.2;$$
$$x, y \geq 0.$$

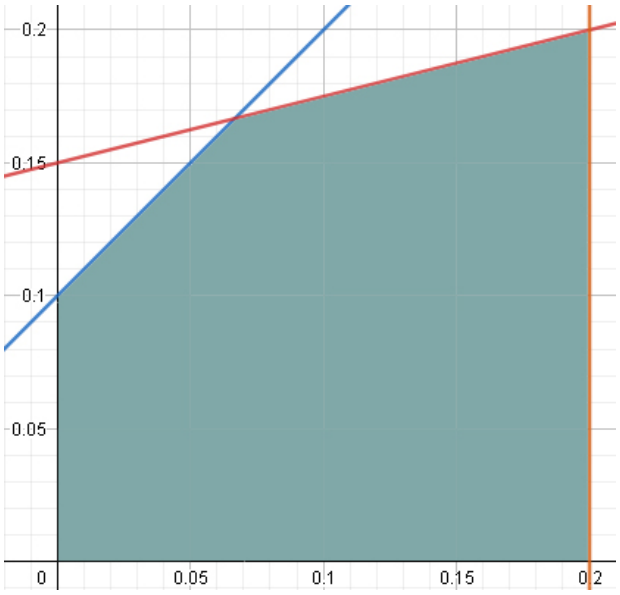

Figura 1. Região Factível

e os seguintes problemas $\min -x$, $\min x - y$, $\min x - 4y$.

2. DISCUSSÃO E RESULTADOS

Na teoria de pontos interiores, tem-s que resolver um sistema não-linear dado por certas condições e este sistema pode ser resolvido com a ajuda do já conhecido método de Newton, porém há uma dificuldade, ele garante apenas convergência local, ou seja, convergência a partir de um ponto inicial que está suficientemente próximo a solução ideal. A seguir serão apresentados as simulações de alguns pontos realizadas através dos algoritmos Primal Dual Afim Escala, referente as duas primeiras sessões a o método Preditor de Caminhos, onde seus resultados serão apresentados na terceira sessão.

2.1 PRIMEIRA SIMULAÇÃO, MIN -X

Na primeira simulação tomamos os seguintes valores:

$\lambda^0 = (-1.8, -0.7, -2.3, -3.7)$
$\mu^0 = (1.5, 11.7, 1.8, 0.7, 2.3, 3.7)$
$x^0 = (0.001, 0.001, 4.998, 1., 5.995, 1.999)$
$\alpha = 0.99$

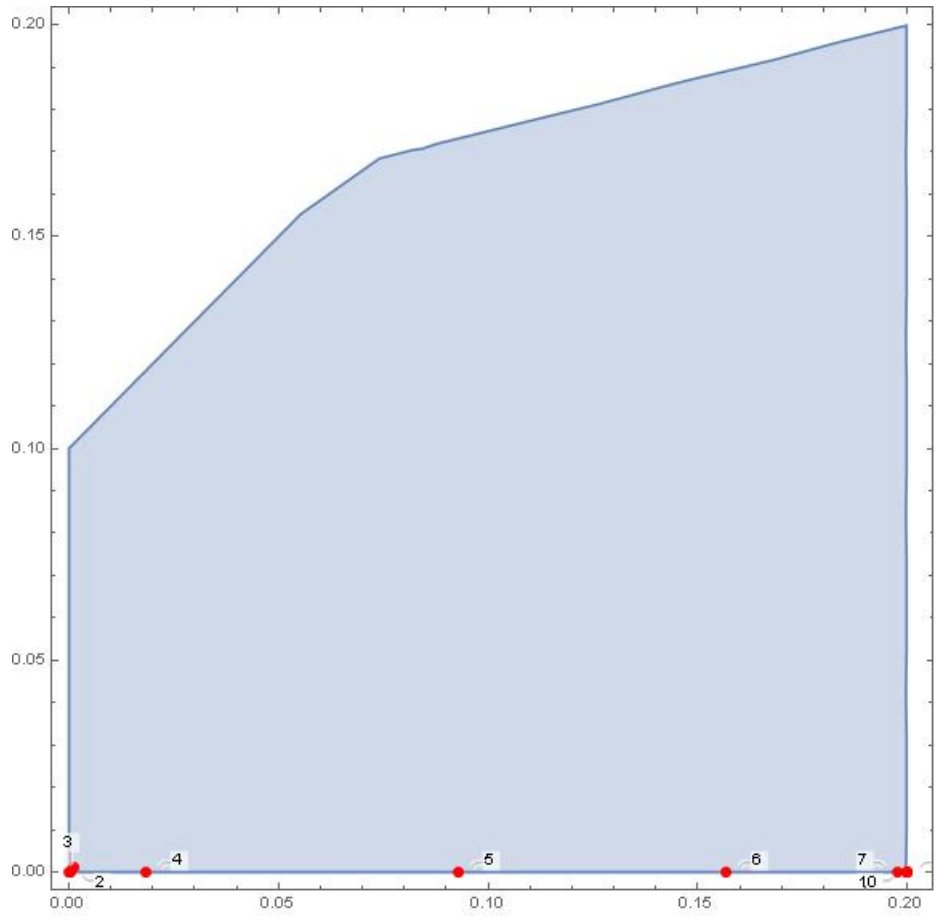

Figura 2. Resultado da simulação 1

Tomando $\alpha = 0.99$ *na primeira simulação do primal dual foi possível perceber a rápida convergência do método, sendo essa em 10 iterações, vale a pena ressaltar que este resultado independe do valor do* α*, uma vez que* x^0 *se encontra próximo à origem e durante cada iteração o ponto continua "andando" sobre o eixo das abcissas, dessa forma, diminuindo o valor de* α *só altera o número de iterações necessárias.*

2.2 SEGUNDA SIMULAÇÃO, MIN -X

Na segunda simulação tomamos os seguintes valores:

$\lambda^0 = (-5.7, -4.7, -1.1, -1.5)$
$\mu^0 = (-5.7, -4.7, -1.1, -1.5)$
$x^0 = (-5.7, -4.7, -1.1, -1.5)$
$\alpha = 0.7$

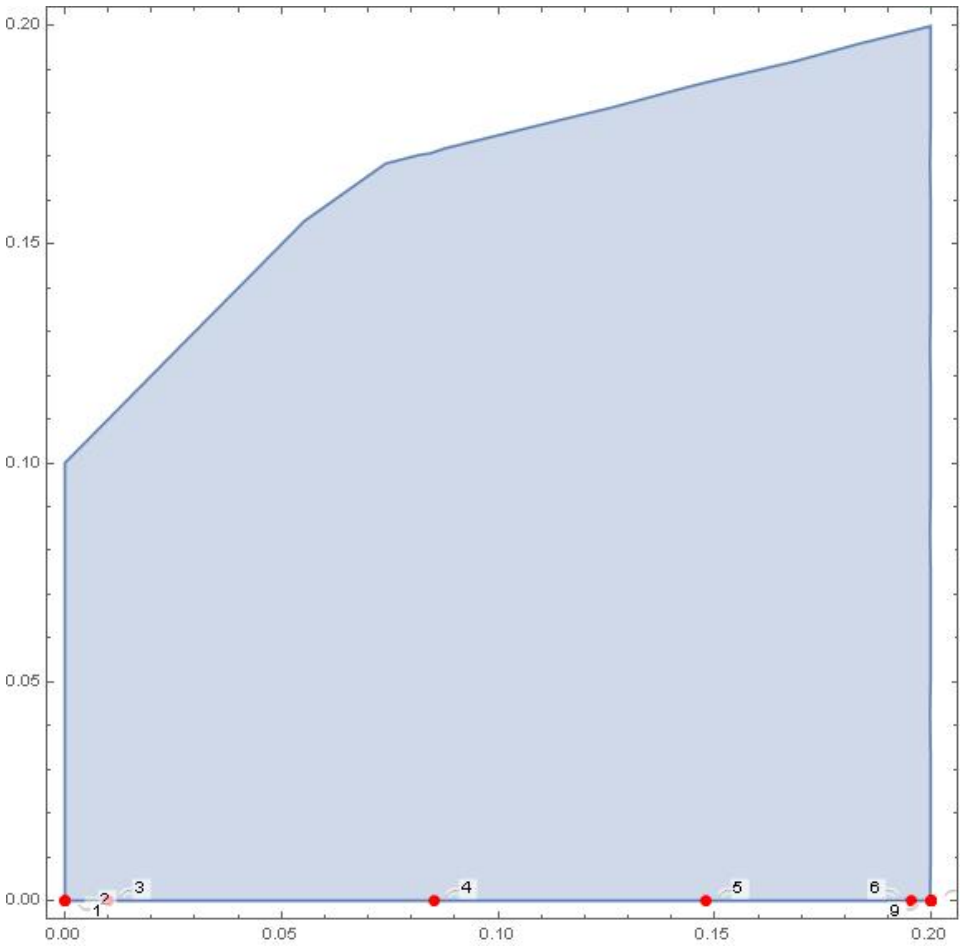

Figura 3. Resultado da simulação 2

Tomando $\alpha = 0.7$ *, temos que o método converge em 22 iterações, podendo assim notar a diferença de tempo necessário em relação à simulação anterior, onde adotamos o valor de* 0.99 *para a variável. Logo abaixo temos o resultado obtido tomando* $\alpha =$

0.2, *o "caminho" realizado se torna mais claro, por outro lado, foram necessários 93 iterações.*

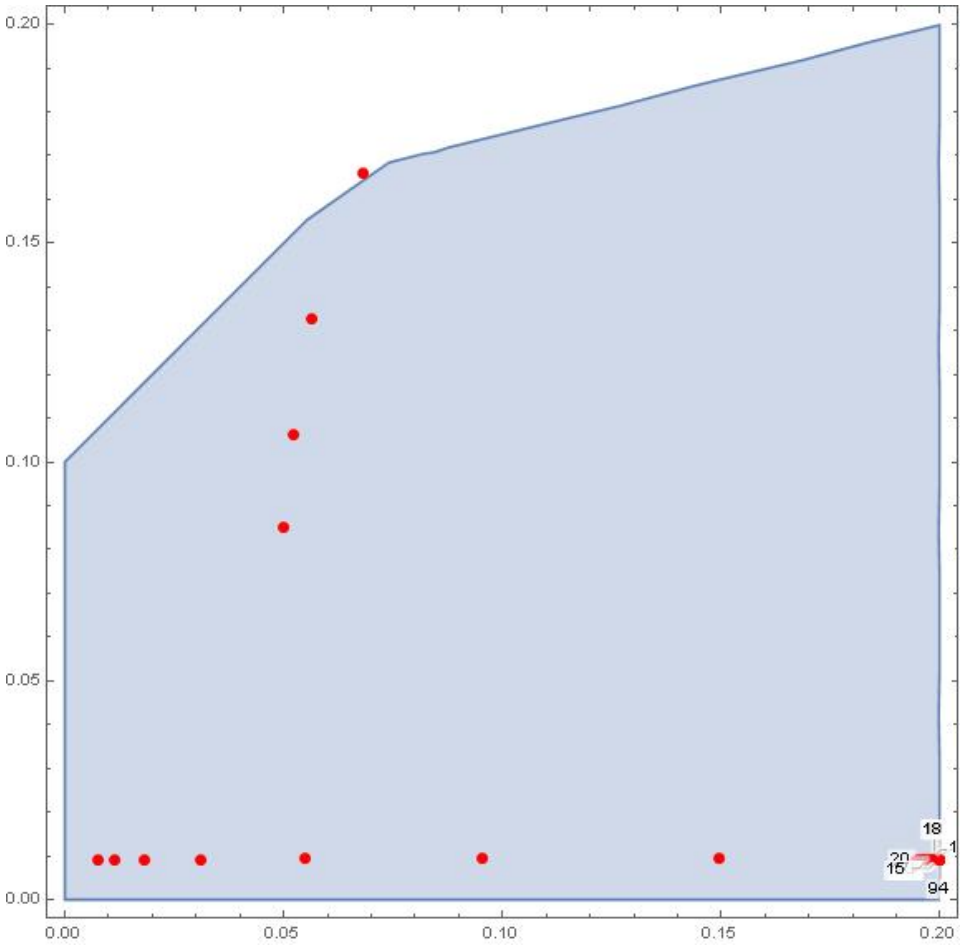

Figura 4. Resultado da simulação 2.1

2.3 TERCEIRA SIMULAÇÃO, MIN -X

Na terceira simulação tomamos os seguintes valores:

$\lambda^0 = (-2.1, -3.1, -0.4, -0.9)$
$\mu^0 = (0.5, 5.8, 2.1, 3.1, 0.4, 0.9)$
$x^0 = (0.12334, 0.05779, 4.998, 1, 5.995, 1.999)$
$\alpha = 0.2$

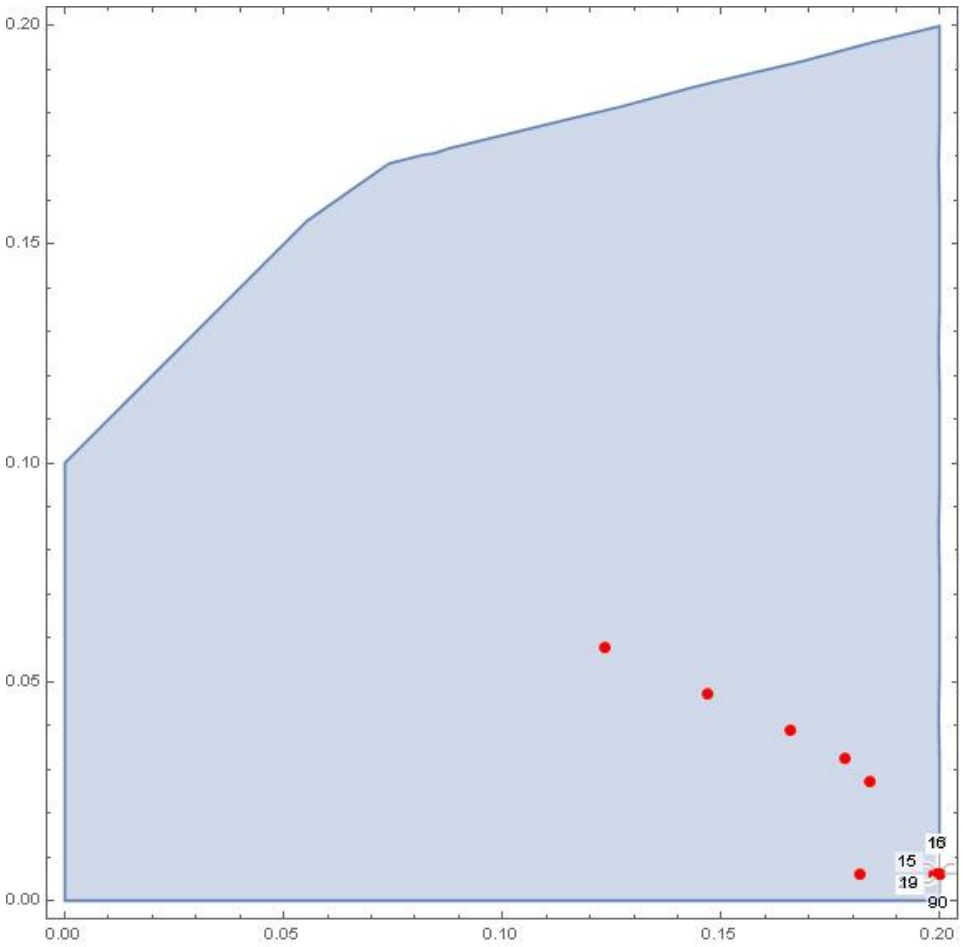

Figura 5. Resultado da simulação 3

Durante a simulação 3 o ponto x^0 *escolhido foi o que chamamos de centro analítico, primeiramente vale a pena ressaltar que para tal* x^0 *o valor de* α *não pode ser tomado tão próximo de um, caso isso ocorra, saímos da região factível. Além disso não temos um valor tão exato para a solução, mesmo depois de 89 iterações chegamos ao resultado* $x = (0.2, 0.0062063)$*, onde temos uma diferença considerável em relação ao valor obtido em todas as outras simulações* $x = (0.2, 0.000123607)$*.*

2.4 QUARTA SIMULAÇÃO, MIN -X

Na quarta simulação tomamos os seguintes valores:

$\lambda^0 = (-2.3, -0.98, -0.54, -0.76)$
$\mu^0 = (0.54, 5.44, 2.3, 0.98, 0.54, 0.76)$
$x^0 = (0.299, 0.199, 4.502, 1.1, 4.905, 1.701)$
$\alpha = 0.9$

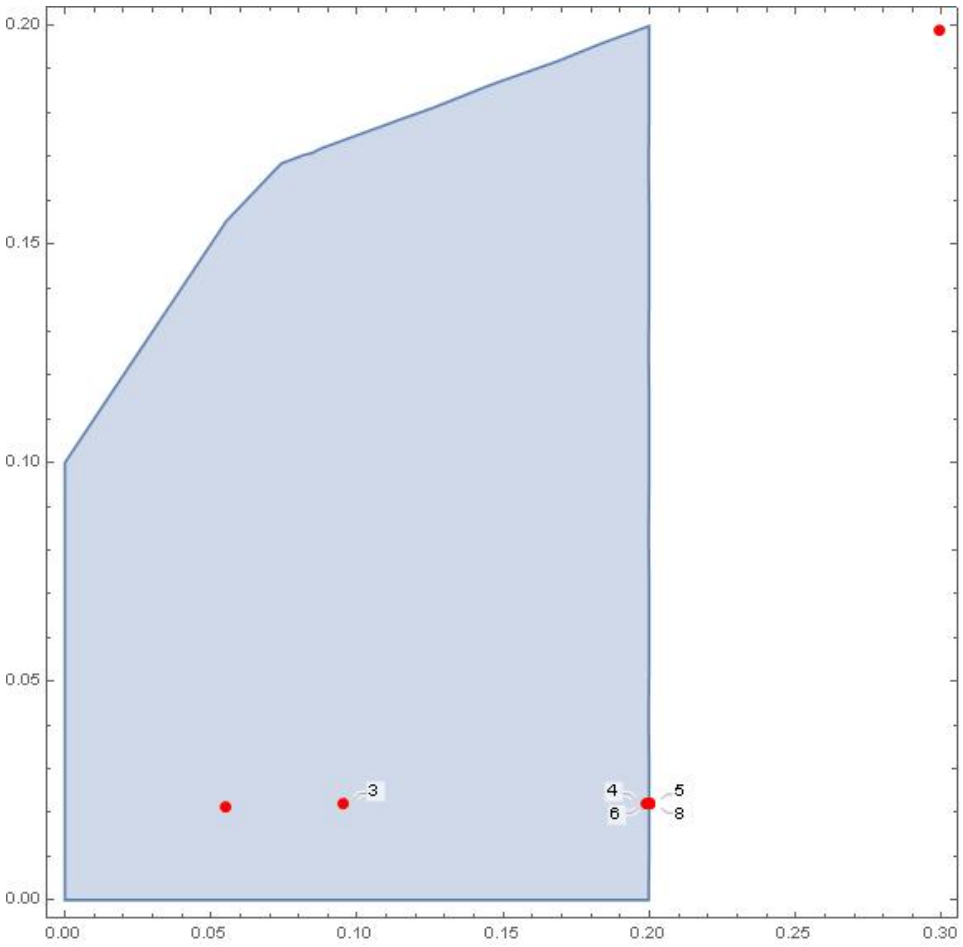

Figura 6. Resultado da simulação 4.

Tomando $\alpha = 0.9$ mesmo com o valor inicial de x^0 (tomado propositalmente) fora da região factível, temos que o método converge (relativamente) em apenas 9 iterações, é fácil ver que o ponto final obtido se encontra longe do ponto desejado, isso independente do valor tomado para α. Além disso é importante notar que a partir da segunda iteração somos deslocados novamente para dentro da região factível permanecendo na mesma até o fim do processo.

2.5 PRIMEIRA SIMULAÇÃO, MIN X-Y

Na primeira simulação tomamos os seguintes valores:

$\lambda^0 = (-5.1, -6.3, -1.7, -2.3)$
$\mu^0 = (0.4, 17.2, 5.1, 6.3, 1.7, 2.3)$
$x^0 = (0.001, 0.001, 4.998, 1., 5.995, 1.999)$
$\alpha = 0.99$

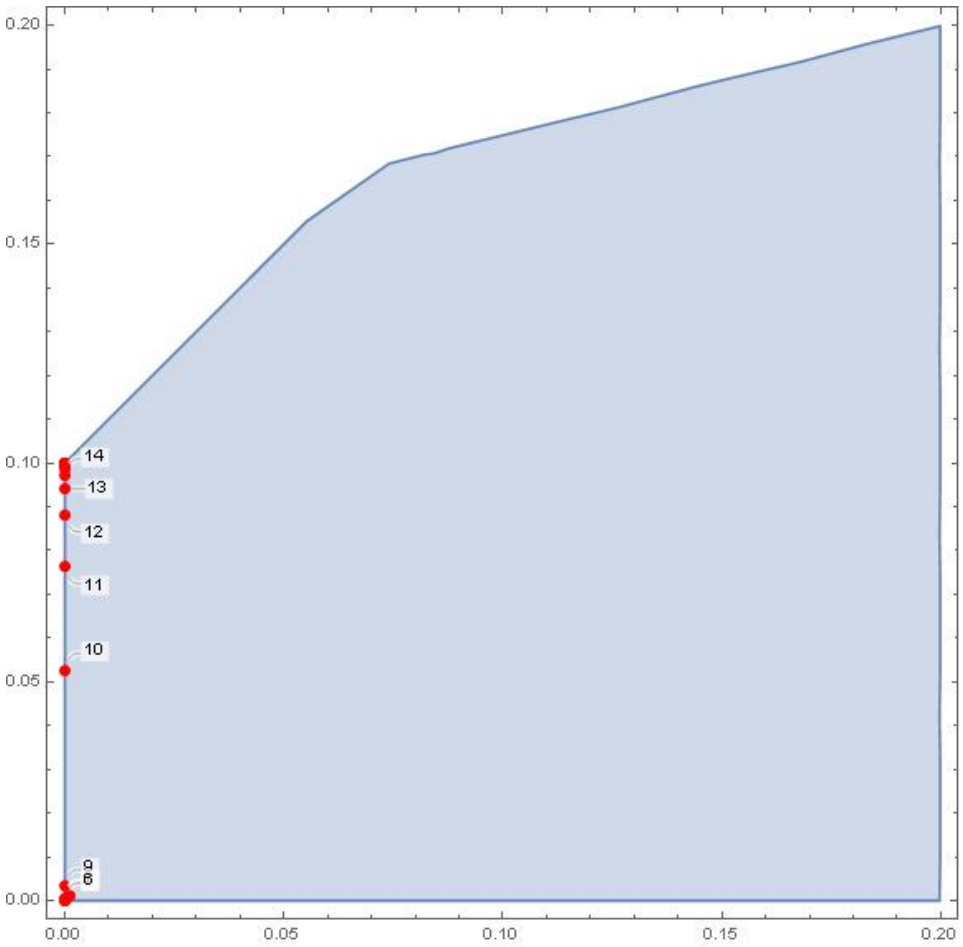

Figura 7. Resultado da simulação 1

Não tivemos nenhuma complicação ou anomalia referente a está simulação, o método convergiu em 15 iterações. O ponto inicial foi bem próximo a origem e numa distância suficientemente pequena em relação ao vértice onde se encontra o ponto ótimo do nosso problema.

2.6 SEGUNDA SIMULAÇÃO, MIN X-Y

Na segunda simulação tomamos os seguintes valores:

$\lambda^0 = (-2.1, -3.1, -0.4, -0.9)$
$\mu^0 = (0.5, 5.8, 2.1, 3.1, 0.4, 0.9)$
$x^0 = (0.068, 0.166, 4.766, 0.902, 5.268, 1.932)$
$\alpha = 0.99$

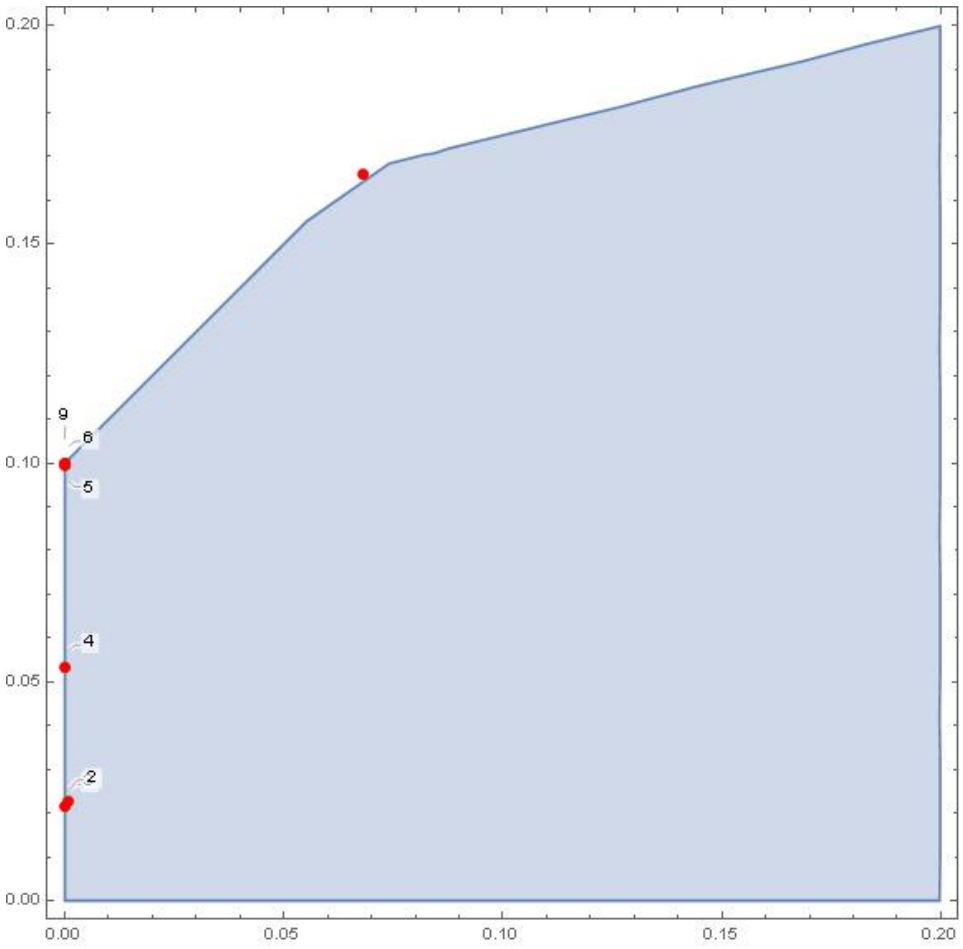

Figura 8. Resultado da simulação 2

Fixado $\alpha = 0.99$ *o método convergiu em apenas 8 iterações. Algo que deve ser pontuado é que mesmo que o ponto inicial se encontrasse próximo ao ponto solução (no caso ao vértice que o possui), após a primeira iteração* x^1 *é levado próximo a origem, onde segue se iterando em cima do eixo das ordenadas.*

2.7 TERCEIRA SIMULAÇÃO, MIN X-Y

Na terceira simulação tomamos os seguintes valores:

$\lambda^0 = (-2.1, -3.1, -0.4, -0.9)$
$\mu^0 = (0.5, 5.8, 2.1, 3.1, 0.4, 0.9)$
$x^0 = (0.12334, 0.05779, 4.998, 1, 5.995, 1.999)$
$\alpha = 0.2$

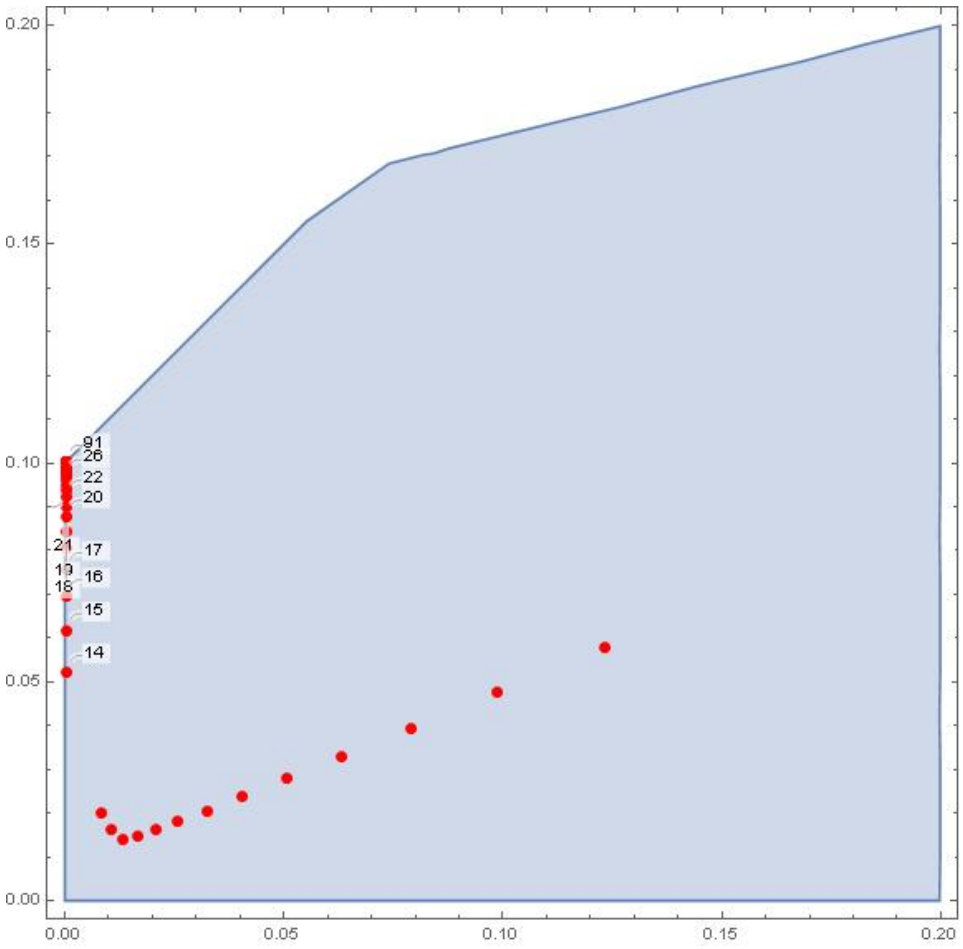

Figura 9. Resultado da simulação 3

Nesta simulação estamos utilizando novamente o centro analítico apresentado na sessão anterior, neste caso não temos o mesmo problema com α suficientemente próximo de um - relação diretamente ligada com a função objetivo - o valor tomado é de $\alpha = 0.2$ para ilustrar o caminho realizado até a convergência que acontece depois de 92 iterações.

2.8 PRIMEIRA SIMULAÇÃO, MIN X-4Y

Na primeira simulação tomamos os seguintes valores:

$\lambda^0 = (-4.3, -5.9, -1.1, -2.1)$
$\mu^0 = (0.4, 10.6, 4.3, 5.9, 1.1, 2.1)$
$x^0 = (0.001, 0.001, 4.998, 1, 5.995, 1.999)$
$\alpha = 0.75$

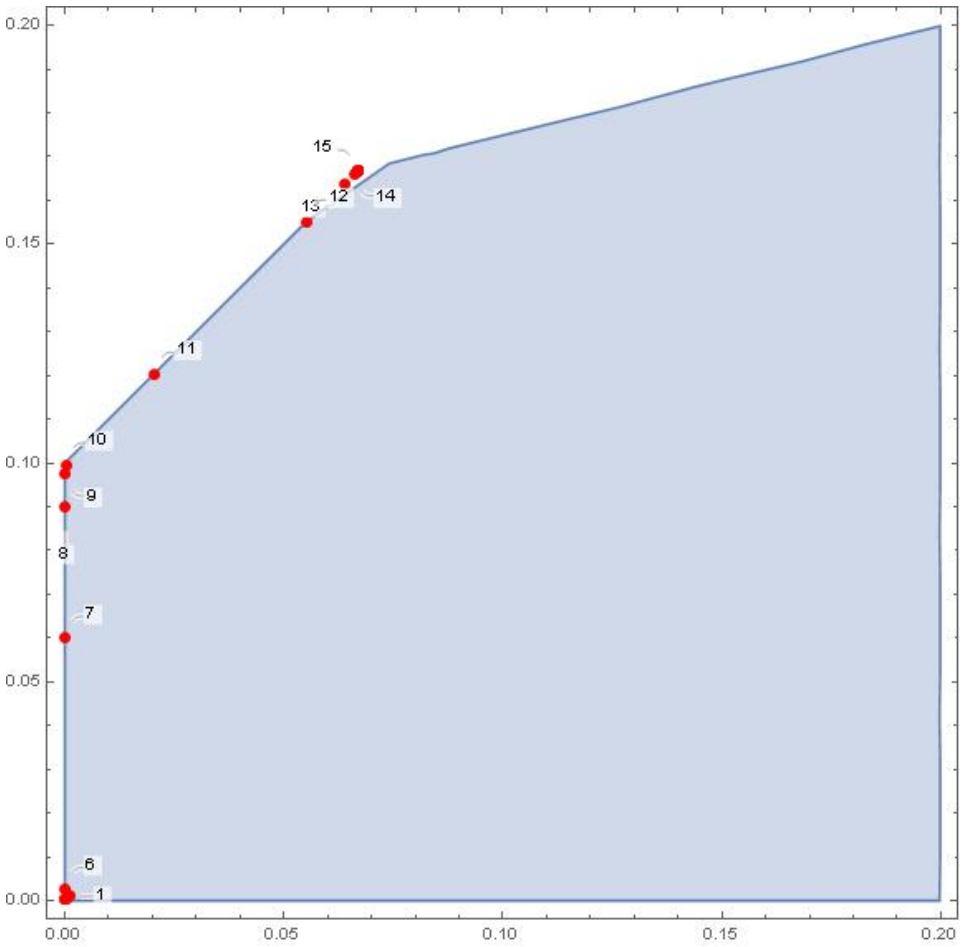

Figura 10. Resultado da simulação 1

Tomando o ponto inicial perto da origem, o método convergiu em 25 iterações, passando sempre pelos vértices.

2.9 SEGUNDA SIMULAÇÃO, MIN X-4Y

Na segunda simulação tomamos os seguintes valores:

$\lambda^0 = (-7.7, -8.7, -0.1, -5.4)$
$\mu^0 = (5.3, 12.8, 7.7, 8.7, 0.1, 5.4)$
$x^0 = (0.068, 0.166, 4.766, 0.902, 5.268, 1.932)$
$\alpha = 0.75$

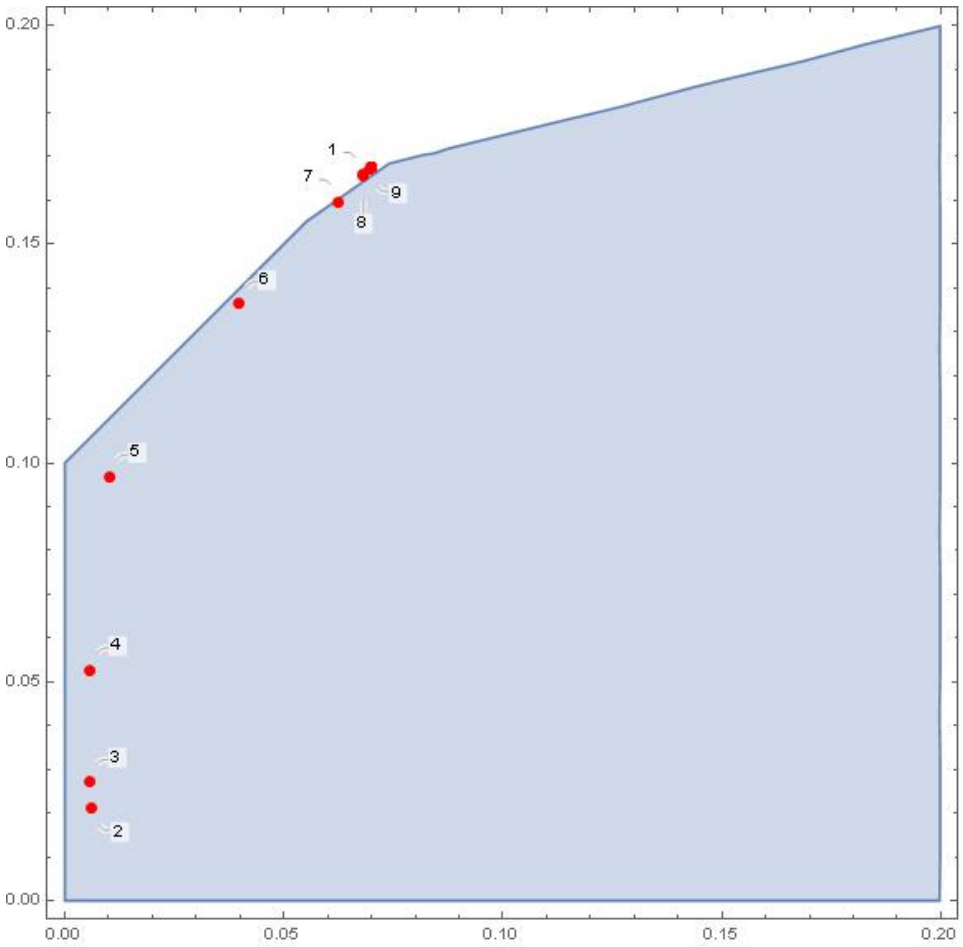

Figura 11. Resultado da simulação 2

Mesmo com o ponto inicial próximo do resultado esperado, temos que na primeira iteração o ponto sequência é jogado próximo a origem e depois convergindo.

2.10 TERCEIRA SIMULAÇÃO, MIN X-4Y

Na terceira simulação tomamos os seguintes valores:

$\lambda^0 = (-8.03, -7.54, -6.77, -11.54)$
$\mu^0 = (6.26, 38.65, 8.03, 7.54, 6.77, 11.54)$
$x^0 = (0.12334, 0.05779, 4.998, 1, 5.995, 1.999)$
$\alpha = 0.25$

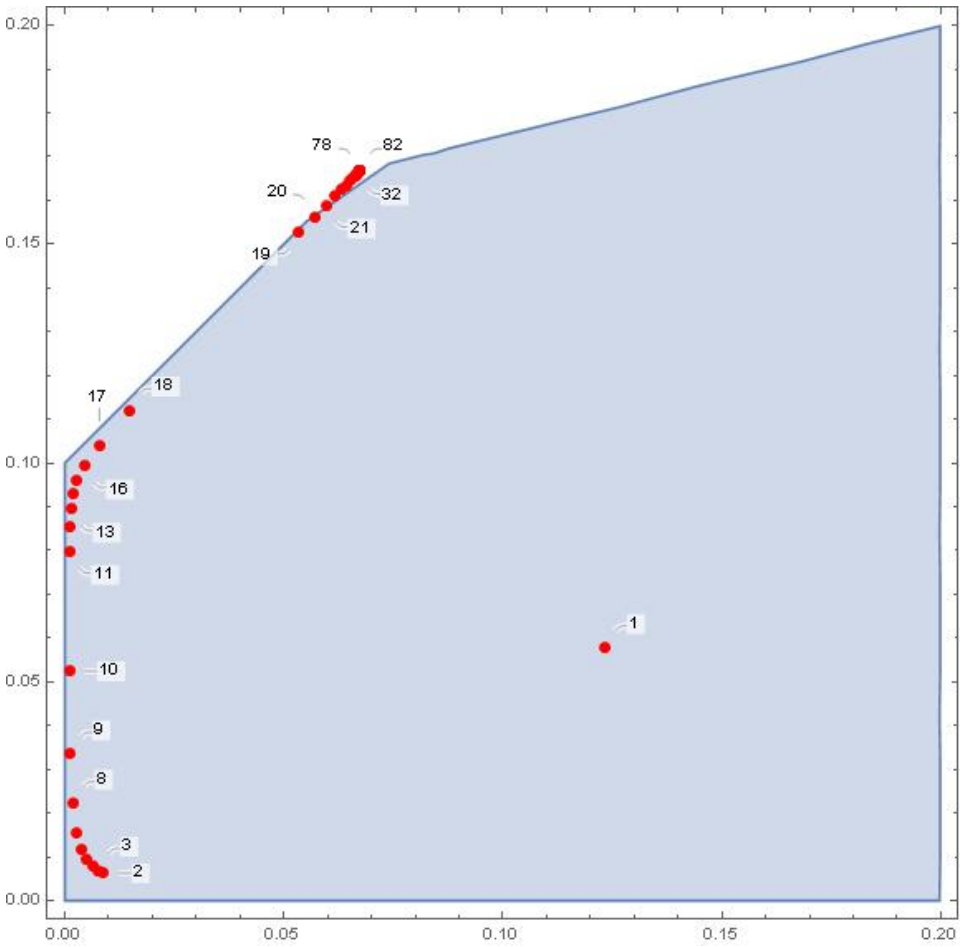

Figura 13. Resultado da simulação 3

Nessa última simulação utilizamos novamente o centro analítico e um valor pequeno de α para analisar toda sua trajetória. O método convergiu em 85 iterações para o ponto $(0.066, 0.166)$

3

CONSIDERAÇÕES FINAIS

No total foram implementados no software *Wolfram Mathematica 12* três métodos de pontos interiores, o Primal Dual Afim Escala, Preditor Corretor e o Método de Barreira, mas durante o processo de testes e obtenção de dados notamos que o único método que se encontrava estável e acaba sempre convergindo nas restrições apresentadas no começo , não necessariamente para o ponto ótimo, mas sim numa vizinhança próxima era o Primal Dual Afim Escala, sendo dele os resultados apresentados anteriormente.

Vale a pena ressaltar que este resultado está diretamente ligado com as restrições, utilizamos

$$\begin{gathered} x + y \leq 0.5; \\ -x + y \leq 0.1; \\ -x + 4y \leq 0.6; \\ x \leq 0.2; \\ x, y \geq 0. \end{gathered}$$

mas quando tratados como

$$\begin{gathered} x + y \leq 5; \\ -x + y \leq 1; \\ -x + 4y \leq 6; \\ x \leq 2; \\ x, y \geq 0. \end{gathered}$$

temos a convergência em todos os métodos. Abaixo temos um exemplo de $\min x - 4y$ com as restrições aumentadas aplicado ao algoritmo Preditor Corretor. O ponto inicial tomado foi $x^0 = (0.1, 0.55, 4.35, 0.55, 3.7, 1.9)$ com $\alpha = 0.2$, o método convergiu após 95 iterações (mesmo não sendo para o ponto ótimo) e em nenhum momento saímos da região factível. Os resultados apresentados pelo método de barreira são análogos.

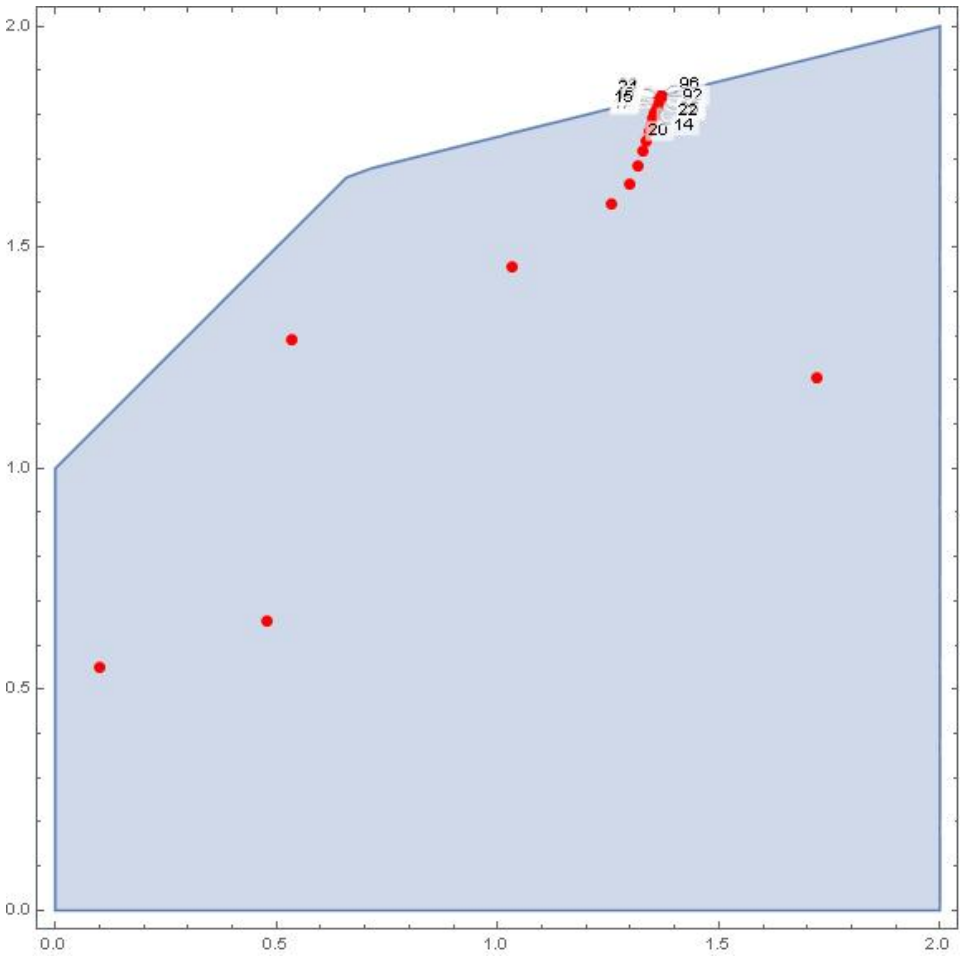

Figura 14. Resultado do Algoritmo Preditor Corretor

Agora em relação ao método Primal Dual, não tivemos complicações, mesmo tomando pontos iniciais fora da região factível a partir da iteração seguinte o mesmo era transferido para dentro e assim não alterando o resultado final.

4

CÓDIGOS

4.1 PRIMAL DUAL

```
PrimalDualBuenoDif[A_List, b_List, c_List, x_List, lamb_List, mi_List,
alpha_, erro_] := {
A1 = A;
b1 = b;
c1 = c;
lamb1 = lamb;
mi1 = mi;
x1 = x;
\[Alpha] = alpha;
Pontos = {};
Pontos = Append[Pontos, Flatten[x1[[1 ;; 2]]]];
nit = 0;
While[Abs[(Transpose[c1].x1 - Transpose[b1].lamb1)[[1, 1]]] > erro,
MA = ArrayFlatten[( {
{A1, 0, 0},
{0, Transpose[A1], IdentityMatrix[Dimensions[A1][[2]]]},
{DiagonalMatrix[ArrayFlatten[mi1, 1]], 0,
DiagonalMatrix[ArrayFlatten[x1, 1]]}
} )];
IA = ArrayFlatten[( {
{b1 - A1.x1},
{c1 - Transpose[A1].lamb1 - mi1},
{-(Table[{x1[[i, 1]]*mi1[[i, 1]]}, {i, 1,
Dimensions[A1][[2]]}])}
} )];
ans = N[LinearSolve[MA, IA]];
dirx = {};
dirlamb = {};
dirmi = {};
For[i = 1, i <= Length[ans], i++,
If[ans[[i, 1]] != 0,
If[i <= Dimensions[A1][[2]],
```

```
dirx = Append[dirx, N[{-x1[[i, 1]]/ans[[i, 1]]}]];
,
If[i <= ( Dimensions[A1][[2]] + Dimensions[A1][[1]]),
dirlamb =
Append[dirlamb,
N[{-lamb1[[(i - Dimensions[A1][[2]]), 1]]/ans[[i, 1]]}]];
,
dirmi =
Append[dirmi,
N[{-mi1[[(i - Dimensions[A1][[2]] - Dimensions[A1][[1]]),
1]]/ans[[i, 1]]}]];
]
]
];
];
x2 = {};
lamb2 = {};
mi2 = {};
For[j = 1, j <= Length[ans], j++,
If[j <= Dimensions[A1][[2]],
x2 = Append[x2, ans[[j]]];
,
If[j <= ( Dimensions[A1][[2]] + Dimensions[A1][[1]]),
lamb2 = Append[lamb2, ans[[j]]];
,
mi2 = Append[mi2, ans[[j]]];
]
]
];
If[Min[x1 + x2] <= 0,
x1 = x1 + \[Alpha]*Min[Select[Flatten[dirx], # > 0 &]]*x2;
Pontos = Append[Pontos, Flatten[x1[[1 ;; 2]]]];
,
x1 = x1 + x2;
Pontos = Append[Pontos, Flatten[x1[[1 ;; 2]]]];
];
If[Min[mi1 + mi2] <= 0,
mi1 = mi1 + \[Alpha]*Min[Select[Flatten[dirmi], # > 0 &]]*mi2;
lamb1 =
lamb1 + \[Alpha]*Min[Select[Flatten[dirmi], # > 0 &]]*lamb2;
```

```
,
mi1 = mi1 + mi2;
lamb1 = lamb1 + lamb2;
];
nit++;
];
Print["Nº de Iterações = ", nit];
Print["x = ", x1];
Print["\[Lambda] = ", lamb1];
Print["\[Mu] = ", mi1];
Print["FO = ", Abs[(Transpose[c].x1 - Transpose[b].lamb1)[[1, 1]]]];
}

Timing[PrimalDualBuenoDif[A7, b7, c7, x17, lamb17, mi17, \[Alpha]7,
erro7]][[1]]
Show[RegionPlot[
And @@ Table[(A7[[;; , 1 ;; 2]].{{x}, {y}})[[i,
1]] <= (b7[[i, 1]]), {i, 1, Length[A7[[;; , 1 ;; 2]]]}], {x, 0,
Max[b7] + 1}, {y, 0, Max[b7] + 1}, PlotRange -> All],
ListPlot[Pontos -> Table[ToString[i], {i, 1, Length[Pontos]}],
PlotStyle -> {PointSize[Large], Red}], ImageSize -> Large]
```

4.2 PREDITOR CORRETOR

```
PredCorBuenoDif[A_List, b_List, c_List, x_List, lamb_List, mi_List,
alpha_, erro_] := {
nit = 0;
A1 = A;
b1 = b;
c1 = c;
lamb1 = lamb;
mi1 = mi;
x1 = x;
lamba = lamb;
mia = mi;
xa = x;
lamb1b = lamb;
mi1b = mi;
x1b = x;
\[Alpha] = alpha;
```

```
Pontos = {};
Pontos = Append[Pontos, Flatten[x1[[1 ;; 2]]]];
While[Abs[(Transpose[c1].x1b - Transpose[b1].lamb1b)[[1, 1]]] > erro,
x1 = x1b;
mi1 = mi1b;
lamb1 = lamb1b;
MA = ArrayFlatten[( {
{A1, 0, 0},
{0, Transpose[A1], IdentityMatrix[Dimensions[A1][[2]]]},
{DiagonalMatrix[ArrayFlatten[mi1, 1]], 0,
DiagonalMatrix[ArrayFlatten[x1, 1]]}
} )];
IA = ArrayFlatten[( {
{b1 - A1.x1},
{c1 - Transpose[A1].lamb1 - mi1},
{-(Table[{x1[[i, 1]]*mi1[[i, 1]]}, {i, 1,
Dimensions[A1][[2]]}])}
} )];
ans = N[LinearSolve[MA, IA]];
dirx = {};
dirlamb = {};
dirmi = {};
For[i = 1, i <= Length[ans], i++,
If[ans[[i, 1]] != 0,
If[i <= Dimensions[A1][[2]],
dirx = Append[dirx, N[{-x1[[i, 1]]/ans[[i, 1]]}]];
,
If[i <= ( Dimensions[A1][[2]] + Dimensions[A1][[1]]),
dirlamb =
Append[dirlamb,
N[{-lamb1[[(i - Dimensions[A1][[2]]), 1]]/ans[[i, 1]]}]];
,
dirmi =
Append[dirmi,
N[{-mi1[[(i - Dimensions[A1][[2]] - Dimensions[A1][[1]]),
1]]/ans[[i, 1]]}]];
]
]
];
];
```

```
x2 = {};
lamb2 = {};
mi2 = {};
For[j = 1, j <= Length[ans], j++,
If[j <= Dimensions[A1][[2]],
x2 = Append[x2, ans[[j]]];
,
If[j <= ( Dimensions[A1][[2]] + Dimensions[A1][[1]]),
lamb2 = Append[lamb2, ans[[j]]];
,
mi2 = Append[mi2, ans[[j]]];
]
]
];
If[Min[x1 + x2] <= 0,
xa = x1 + \[Alpha]*Min[Select[Flatten[dirx], # > 0 &]]*x2;
dxa = \[Alpha]*Min[Select[Flatten[dirx], # > 0 &]]*x2;
,
xa = x1 + x2;
dxa = x2;
];
If[Min[mi1 + mi2] <= 0,
mia = mi1 + \[Alpha]*Min[Select[Flatten[dirmi], # > 0 &]]*mi2;
dmia = \[Alpha]*Min[Select[Flatten[dirmi], # > 0 &]]*mi2;
lamba =
lamb1 + \[Alpha]*Min[Select[Flatten[dirmi], # > 0 &]]*lamb2;
,
mia = mi1 + mi2;
dmia = mi2;
lamba = lamb1 + lamb2;
];
MA = ArrayFlatten[( {
{A1, 0, 0},
{0, Transpose[A1], IdentityMatrix[Dimensions[A1][[2]]]},
{DiagonalMatrix[ArrayFlatten[mi1, 1]], 0,
DiagonalMatrix[ArrayFlatten[x1, 1]]}
} )];
IA = ArrayFlatten[( {
{b1 - A1.x1},
{c1 - Transpose[A1].lamb1 - mi1},
```

```
{-(Table[{x1[[i, 1]]*mi1[[i, 1]]}, {i, 1,
Dimensions[A1][[2]]}] -
N[(((Transpose[xa].mia)[[1, 1]])/Length[x1])] +
N[(Transpose[dxa].dmia)[[1, 1]]])}
} )];
ans = N[LinearSolve[MA, IA]];
dirx = {};
dirlamb = {};
dirmi = {};
For[i = 1, i <= Length[ans], i++,
If[ans[[i, 1]] != 0,
If[i <= Dimensions[A1][[2]],
dirx = Append[dirx, N[{-x1[[i, 1]]/ans[[i, 1]]}]];
,
If[i <= ( Dimensions[A1][[2]] + Dimensions[A1][[1]]),
dirlamb =
Append[dirlamb,
N[{-lamb1[[(i - Dimensions[A1][[2]]), 1]]/ans[[i, 1]]}]];
,
dirmi =
Append[dirmi,
N[{-mi1[[(i - Dimensions[A1][[2]] - Dimensions[A1][[1]]),
1]]/ans[[i, 1]]}]];
]
]
];
];
x2 = {};
lamb2 = {};
mi2 = {};
For[j = 1, j <= Length[ans], j++,
If[j <= Dimensions[A1][[2]],
x2 = Append[x2, ans[[j]]];
,
If[j <= ( Dimensions[A1][[2]] + Dimensions[A1][[1]]),
lamb2 = Append[lamb2, ans[[j]]];
,
mi2 = Append[mi2, ans[[j]]];
]
]
```

```
];
If[Min[x1 + x2] <= 0,
x1b = x1 + \[Alpha]*Min[Select[Flatten[dirx], # > 0 &]]*x2;
Pontos = Append[Pontos, Flatten[x1b[[1 ;; 2]]]];
,
x1b = x1 + x2;
Pontos = Append[Pontos, Flatten[x1b[[1 ;; 2]]]];
];
If[Min[mi1 + mi2] <= 0,
mi1b = mi1 + \[Alpha]*Min[Select[Flatten[dirmi], # > 0 &]]*mi2;
lamb1b =
lamb1 + \[Alpha]*Min[Select[Flatten[dirmi], # > 0 &]]*lamb2;
,
mi1b = mi1 + mi2;
lamb1b = lamb1 + lamb2;
];
nit++;
];
Print["Nº de Iterações = ", nit];
Print["x = ", x1b];
Print["\[Lambda] = ", lamb1b];
Print["\[Mu] = ", mi1b];
Print["FO = ",
Abs[(Transpose[c].x1b - Transpose[b].lamb1b)[[1, 1]]]];
}
```

4.3 MÉTODO DE BARREIRA

```
BarreiraBuenoDif[A_List, b_List, c_List, x_List, lamb_List, mi_List,
rho_, alpha_, erro_] := {
nit = 0;
A1 = A;
b1 = b;
c1 = c;
lamb1 = lamb;
mi1 = mi;
x1 = x;
\[Rho] = rho;
\[Alpha] = alpha;
Pontos = {};
```

```
Pontos = Append[Pontos, Flatten[x1[[1 ;; 2]]]];
While[Abs[(Transpose[c1].x1 - Transpose[b1].lamb1)[[1,
1]] - \[Rho]] > erro,
MA = ArrayFlatten[( {
{A1, 0, 0},
{0, Transpose[A1], IdentityMatrix[Dimensions[A1][[2]]]},
{DiagonalMatrix[ArrayFlatten[mi1, 1]], 0,
DiagonalMatrix[ArrayFlatten[x1, 1]]}
} )];
IA = ArrayFlatten[( {
{b1 - A1.x1},
{c1 - Transpose[A1].lamb1 - mi1},
{-(Table[{x1[[i, 1]]*mi1[[i, 1]] - \[Rho]}, {i, 1,
Dimensions[A1][[2]]}])}
} )];
ans = N[LinearSolve[MA, IA]];
dirx = {};
dirlamb = {};
dirmi = {};
For[i = 1, i <= Length[ans], i++,
If[ans[[i, 1]] != 0,
If[i <= Dimensions[A1][[2]],
dirx = Append[dirx, N[{-x1[[i, 1]]/ans[[i, 1]]}]];
,
If[i <= ( Dimensions[A1][[2]] + Dimensions[A1][[1]]),
dirlamb =
Append[dirlamb,
N[{-lamb1[[(i - Dimensions[A1][[2]]), 1]]/ans[[i, 1]]}]];
,
dirmi =
Append[dirmi,
N[{-mi1[[(i - Dimensions[A1][[2]] - Dimensions[A1][[1]]),
1]]/ans[[i, 1]]}]];
]
]
];
];
x2 = {};
lamb2 = {};
mi2 = {};
```

```
For[j = 1, j <= Length[ans], j++,
If[j <= Dimensions[A1][[2]],
x2 = Append[x2, ans[[j]]];
,
If[j <= ( Dimensions[A1][[2]] + Dimensions[A1][[1]]),
lamb2 = Append[lamb2, ans[[j]]];
,
mi2 = Append[mi2, ans[[j]]];
]
]
];
If[Min[x1 + x2] <= 0,
x1 = x1 + \[Alpha]*Min[Select[Flatten[dirx], # > 0 &]]*x2;
Pontos = Append[Pontos, Flatten[x1[[1 ;; 2]]]];
,
x1 = x1 + x2;
Pontos = Append[Pontos, Flatten[x1[[1 ;; 2]]]];
];
If[Min[mi1 + mi2] <= 0,
mi1 = mi1 + \[Alpha]*Min[Select[Flatten[dirmi], # > 0 &]]*mi2;
lamb1 =
lamb1 + \[Alpha]*Min[Select[Flatten[dirmi], # > 0 &]]*lamb2;
,
mi1 = mi1 + mi2;
lamb1 = lamb1 + lamb2;
];
\[Rho] = (Transpose[x1].mi1)[[1, 1]]/Length[x1];
nit++;
];
Print["Nº de Iterações = ", nit];
Print["x = ", x1];
Print["\[Lambda] = ", lamb1];
Print["\[Mu] = ", mi1];
Print["\[Rho] = ", \[Rho]];
Print["FO = ",
Abs[(Transpose[c].x1 - Transpose[b].lamb1)[[1, 1]]] - \[Rho]*
Length[x1]];
}
```