

Programação Linear

Relatório EP3 – Método Simplex

Daniel Augusto Cortez – 2960291
Lucas Rodrigues Colucci – 6920251

18 de junho de 2012

1 Introdução

Apresentamos neste relatório a implementação do método simplex (full tableau), utilizando a linguagem de programação Octave. O algoritmo é desenvolvido em detalhes em [1]. Considera-se o problema de programação linear em seu formato padrão

$$\begin{aligned} &\text{minimizar} && \mathbf{c}^T \mathbf{x} \\ &\text{sujeito a} && \mathbf{Ax} = \mathbf{b} \\ &&& \mathbf{x} \geq \mathbf{0}, \end{aligned} \tag{1.1}$$

onde $\mathbf{x}, \mathbf{c} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$ e $\mathbf{A} \in \mathbb{R}^{m \times n}$. O algoritmo pode ser descrito pelos seguintes passos:

Fase I.

1. Multiplique algumas das restrições por -1 , modificando o problema de tal forma que $\mathbf{b} \geq 0$.
2. Introduza variáveis artificiais y_1, \dots, y_m e aplique o método simplex ao problema auxiliar com custo $\sum_{i=1}^m y_i$.
3. Se o custo ótimo do problema auxiliar é positivo, o problema original é inviável e o algoritmo termina.
4. Se o custo ótimo do problema auxiliar é nulo, uma solução viável do problema original é obtida. Se nenhuma variável artificial está na base

final, as variáveis artificiais e respectivas colunas são eliminadas, e uma base viável para o problema original está disponível.

5. Se a ℓ -ésima variável básica é artificial, examine a ℓ -ésima entrada das colunas $\mathbf{B}^{-1}\mathbf{A}_j$, $j = 1, \dots, n$. Se todas as entradas são nulas, então a ℓ -ésima linha representa uma restrição redundante e é eliminada. De outra forma, se a ℓ -ésima entrada da j -ésima coluna é diferente de zero, aplique uma mudança de base: a ℓ -ésima variável básica sai e x_j entra na base. Repita esse procedimento até que todas as variáveis artificiais sejam levadas para fora da base.

Fase II.

1. Faça a base e o tableau final obtidos na fase I serem a base e o tableau inicial para a fase II.
2. Calcule o custo reduzido de todas as variáveis para essa base inicial, usando o vetor de custos do problema original.
3. Aplique o método simplex ao problema original.

Uma iteração da implementação através do tableau.

1. Uma iteração típica começa com um tableau associado com uma matriz básica \mathbf{B} e a solução básica correspondente \mathbf{x} .
2. Examine todos os custos reduzidos na linha zero do tableau. Se todos forem não-negativos, a solução viável básica atual é ótima, e o algoritmo termina. Caso contrário, escolha algum j tal que $\bar{c}_j < 0$.
3. Considere o vetor $\mathbf{u} = \mathbf{B}^{-1}\mathbf{A}_j$, que é a j -ésima coluna (a coluna pivô) do tableau. Se nenhuma componente de \mathbf{u} é positiva, o custo ótimo é $-\infty$ e o algoritmo termina.
4. Para cada i tal que $u_i > 0$, calcule a razão $x_{B(i)}/u_i$. Seja ℓ o índice da linha que corresponde à menor razão. A coluna $\mathbf{A}_{B(\ell)}$ saí da base e a coluna \mathbf{A}_j entra.
5. Adicione a cada linha do tableau um múltiplo constante da ℓ -ésima linha tal que o u_ℓ (elemento pivô) se torne um e todas as outras entradas da coluna pivô se tornem zero.

A implementação do método simplex em Octave seguiu exatamente os passos acima, conforme descrevemos a seguir.

2 Funções Implementadas

No arquivo `simplex.m`, escrevemos a função

```
[ind x] = simplex(A, b, c, m, n, print)
```

que pode ser chamada externamente para resolver qualquer PL na forma (1.1). Os parâmetros de entrada da função são:

- **A**: matriz de dimensão $m \times n$ de coeficientes das restrições;
- **b**: vetor de dimensão m do lado direito das restrições;
- **c**: vetor de dimensão n de custos;
- **m**: número de restrições;
- **n**: número de variáveis;
- **print**: booleano indicando se o tableau deve ser impresso a cada iteração.

Os parâmetros de saída são:

- **ind**: valor 0 se o problema tiver solução ótima finita, -1 se tiver solução ilimitada e $+1$ se o problema for inviável;
- **x**: solução ótima do problema, quando existir.

A função `simplex` começa resolvendo a fase I do método simplex. Em seguida, testa se o custo encontrado na fase I é positivo. Nesse caso, o problema é inviável. Caso contrário, prossegue com a fase II.

A fase I do método é resolvida pela função

```
[T B m] = phase_1(A, B, m, n, print)
```

Os parâmetros de entrada da função são:

- **A**: matriz de dimensão $m \times n$ de coeficientes das restrições;
- **b**: vetor de dimensão m do lado direito das restrições;
- **m**: número de restrições;

- n : número de variáveis;
- `print`: booleano indicando se o tableau deve ser impresso a cada iteração.

Os parâmetros de saída são:

- T : matriz de dimensão $(m + 1) \times (n + 1)$ que representa o tableau final da fase 1;
- B : vetor de índices das variáveis básicas;
- m : número de linhas do tableau final, excluindo a primeira linha.

A função `phase_1` começa colocando o problema (1.1) no formato $\mathbf{b} \geq 0$, depois adiciona as m variáveis artificiais à matriz \mathbf{A} (colocando uma matriz identidade $m \times m$ depois da última coluna de \mathbf{A}). Em seguida, monta o tableau inicial em T (implementado na função `phase_1_tableau`). O vetor de índices das variáveis básicas B é inicialmente definido com os índices das variáveis artificiais. O tableau T é então iterado, pivotando de acordo com o método simplex, através da função `tableau_solve`. O tableau resultante das iterações é testado para verificar a viabilidade do problema original (custo nulo). Se o problema for viável, prossegue removendo as colunas das variáveis artificiais do tableau, depois removendo possíveis restrições redundantes (função `remove_redundants`), e finalmente remove as variáveis artificiais da solução básica encontrada, se estiverem presentes (função `exit_artificiais`). Essas duas últimas funções implementam o passo 5 da fase I do método simplex.

A fase II do método é resolvida pela função

```
[ind x] = phase_2(T, B, c, m, n, print)
```

Os parâmetros de entrada da função são:

- T : matriz de dimensão $(m + 1) \times (n + 1)$ que representa o tableau final da fase 1;
- B : vetor de índices das variáveis básicas;
- \mathbf{c} : vetor de dimensão n de custos;
- m : número de restrições;

- n : número de variáveis;
- `print`: booleano indicando se o tableau deve ser impresso a cada iteração.

Os parâmetros de saída são:

- `ind`: valor 0 se o problema tiver solução ótima finita, -1 se tiver solução ilimitada e $+1$ se o problema for inviável;
- `x`: solução ótima do problema, quando existir.

A fase II começa calculando o custo e os custos reduzidos do problema original em função do tableau final retornado pela fase I. Esses custos são utilizados para atualizar a primeira linha do tableau, que é iterado em seguida pela função `tableau_solve`.

Tanto a fase I quanto a fase II utilizam essencialmente a função

```
[T B iter ind x] = tableau_solve(T, B, m, n, print, iter)
```

cujos parâmetros de entrada são

- T : matriz de dimensão $(m + 1) \times (n + 1)$ que representa o tableau da iteração `iter-1`;
- B : vetor de índices das variáveis básicas da iteração `iter-1`;
- m : número de variáveis básicas;
- n : número de variáveis;
- `print`: booleano indicando se o tableau deve ser impresso a cada iteração.
- `iter`: número da iteração atual.

Os parâmetros de saída são:

- T : matriz de dimensão $(m + 1) \times (n + 1)$ que representa o tableau da iteração `iter`;
- B : vetor de índices das variáveis básicas da iteração `iter`;

- `iter`: número da iteração atual.
- `ind`: valor 0 se o problema tiver solução ótima finita, -1 se tiver solução ilimitada e $+1$ se o problema for inviável;
- `x`: solução ótima do problema, quando existir.

A função acima implementa os passos descritos no algoritmo “Uma iteração da implementação através do tableau” (página 2). A mesma utiliza a regra Bland [1] na determinação da variável básica a sair da base e da não-básica a entrar na mesma. Com isso, evita-se a possibilidade do algoritmo ciclar. A operação básica em `tableau_solve` é a pivotação do tableau, que é feita pela função `pivot`.

A impressão do tableau durante a execução do programa é feita pela função

```
print_tableau(T, B, m, n, i_pivot, j_pivot, iter, msg)
```

Os parâmetros de entrada dessa função são

- T : matriz de dimensão $(m + 1) \times (n + 1)$ que representa o tableau da iteração `iter`;
- B : vetor de índices das variáveis básicas da iteração `iter`;
- m : número de variáveis básicas;
- n : número de variáveis;
- `i_pivot`: linha de pivotamento do tableau (zero caso não aplicável);
- `j_pivot`: coluna de pivotamento do tableau (zero caso não aplicável)
- `iter`: número da iteração atual.
- `msg`: observação a ser impressa ao lado do número da iteração.

A função basicamente itera sobre todas as entradas da matriz T , imprimindo o seu conteúdo com a formatação adequada. Caso o elemento iterado corresponda à posição `(i_pivot, j_pivot)`, um símbolo `*` é impresso para marcar o pivô.

As demais funções implementadas são utilizadas para modularizar o código e são de propósito específico:

```

[A b] = make_b_positive(A, b, m, n)
[T B m] = remove_redundants(T, B, m, n, print, iter)
[T B] = exit_artificials(T, B, m, n, print, iter)
bool = is_zero_vector(v)
x = get_solution(T, B, m, n)
T = pivot(T, i_pivot, j_pivot, m, n)

```

Não entraremos em detalhes sobre a implementação das mesmas.

3 Alguns Exemplos

Os exemplos abaixo mostram a saída gerada ao se rodar o script `ep3.m` com o Octave. (`$ octave -qf ep3.m`). A saída também está disponível no arquivo `out.txt`. O script apresenta a formulação de 7 problemas no formato (1.1) que são resolvidos com a função `simplex` implementada em `simplex.m` e descrita anteriormente. Os exemplos mostram problemas com uma solução ótima finita (1, 2 e 3), problemas inviáveis (4 e 5) e problemas ilimitados (6 e 7).

```

-----
Exemplo 1 - Solução Ótima - Linhas LI
-----
m = 3
n = 6
A =

    1    2    2    1    0    0
    2    1    2    0    1    0
    2    2    1    0    0    1

b =

    20
    20
    20

c =

   -10
   -12
   -12
    0
    0
    0

Simplex: Fase 1

Iteração 1
-----
          | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 | x9 |
-60.000 | -5.000 | -5.000 | -5.000 | -1.000 | -1.000 | -1.000 | 0.000 | 0.000 | 0.000 |
-----
x7  20.000 | 1.000 | 2.000 | 2.000 | 1.000 | 0.000 | 0.000 | 1.000 | 0.000 | 0.000 |
x8  20.000 | 2.000*| 1.000 | 2.000 | 0.000 | 1.000 | 0.000 | 0.000 | 1.000 | 0.000 |
x9  20.000 | 2.000 | 2.000 | 1.000 | 0.000 | 0.000 | 1.000 | 0.000 | 0.000 | 1.000 |
-----

Iteração 2
-----
          | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 | x9 |
-10.000 | 0.000 | -2.500 | 0.000 | -1.000 | 1.500 | -1.000 | 0.000 | 2.500 | 0.000 |
-----
x7  10.000 | 0.000 | 1.500 | 1.000 | 1.000 | -0.500 | 0.000 | 1.000 | -0.500 | 0.000 |
x1  10.000 | 1.000 | 0.500 | 1.000 | 0.000 | 0.500 | 0.000 | 0.000 | 0.500 | 0.000 |
-----

```

x9	0.000		0.000		1.000*		-1.000		0.000		-1.000		1.000		0.000		-1.000		1.000	
----	-------	--	-------	--	--------	--	--------	--	-------	--	--------	--	-------	--	-------	--	--------	--	-------	--

Iteração 3

	-10.000		x1		x2		x3		x4		x5		x6		x7		x8		x9	
	0.000		0.000		0.000		-2.500		-1.000		-1.000		1.500		0.000		0.000		2.500	
x7	10.000		0.000		0.000		2.500*		1.000		1.000		-1.500		1.000		1.000		-1.500	
x1	10.000		1.000		0.000		1.500		0.000		1.000		-0.500		0.000		1.000		-0.500	
x2	0.000		0.000		1.000		-1.000		0.000		-1.000		1.000		0.000		-1.000		1.000	

Iteração 4

	0.000		x1		x2		x3		x4		x5		x6		x7		x8		x9	
	0.000		0.000		0.000		0.000		0.000		0.000		0.000		1.000		1.000		1.000	
x3	4.000		0.000		0.000		1.000		0.400		0.400		-0.600		0.400		0.400		-0.600	
x1	4.000		1.000		0.000		0.000		-0.600		0.400		0.400		-0.600		0.400		0.400	
x2	4.000		0.000		1.000		0.000		0.400		-0.600		0.400		0.400		-0.600		0.400	

Iteração 5 (tableau final da fase 1)

	0.000		x1		x2		x3		x4		x5		x6	
	0.000		0.000		0.000		0.000		0.000		0.000		0.000	
x3	4.000		0.000		0.000		1.000		0.400		0.400		-0.600	
x1	4.000		1.000		0.000		0.000		-0.600		0.400		0.400	
x2	4.000		0.000		1.000		0.000		0.400		-0.600		0.400	

Simplex: Fase 2

Iteração 1

	136.000		x1		x2		x3		x4		x5		x6	
	0.000		0.000		0.000		0.000		3.600		1.600		1.600	
x3	4.000		0.000		0.000		1.000		0.400		0.400		-0.600	
x1	4.000		1.000		0.000		0.000		-0.600		0.400		0.400	
x2	4.000		0.000		1.000		0.000		0.400		-0.600		0.400	

Solução ótima encontrada com custo -136.00000:

x =

4
4
4
0
0
0

Exemplo 2 - Solução Ótima - Linhas LD

m = 4

n = 4

A =

1	2	3	0
-1	2	6	0
0	4	9	0
0	0	3	1

b =

3
2
5
1

c =

1
1
1
0

Simplex: Fase 1

Iteração 1

	-11.000		x1		x2		x3		x4		x5		x6		x7		x8	
	0.000		-0.000		-8.000		-21.000		-1.000		0.000		0.000		0.000		0.000	
x5	3.000		1.000		2.000		3.000		0.000		1.000		0.000		0.000		0.000	
x6	2.000		-1.000		2.000*		6.000		0.000		0.000		1.000		0.000		0.000	
x7	5.000		0.000		4.000		9.000		0.000		0.000		0.000		1.000		0.000	
x8	1.000		0.000		0.000		3.000		1.000		0.000		0.000		0.000		1.000	

Iteração 2										
	x1	x2	x3	x4	x5	x6	x7	x8		
	-3.000	-4.000	0.000	3.000	-1.000	0.000	4.000	0.000	0.000	

x5	1.000	2.000*	0.000	-3.000	0.000	1.000	-1.000	0.000	0.000	
x2	1.000	-0.500	1.000	3.000	0.000	0.000	0.500	0.000	0.000	
x7	1.000	2.000	0.000	-3.000	0.000	0.000	-2.000	1.000	0.000	
x8	1.000	0.000	0.000	3.000	1.000	0.000	0.000	0.000	1.000	

Iteração 3										
	x1	x2	x3	x4	x5	x6	x7	x8		
	-1.000	0.000	0.000	-3.000	-1.000	2.000	2.000	0.000	0.000	

x1	0.500	1.000	0.000	-1.500	0.000	0.500	-0.500	0.000	0.000	
x2	1.250	0.000	1.000	2.250	0.000	0.250	0.250	0.000	0.000	
x7	0.000	0.000	0.000	0.000	0.000	-1.000	-1.000	1.000	0.000	
x8	1.000	0.000	0.000	3.000*	1.000	0.000	0.000	0.000	1.000	

Iteração 4										
	x1	x2	x3	x4	x5	x6	x7	x8		
	0.000	0.000	0.000	0.000	0.000	2.000	2.000	0.000	1.000	

x1	1.000	1.000	0.000	0.000	0.500	0.500	-0.500	0.000	0.500	
x2	0.500	0.000	1.000	0.000	-0.750	0.250	0.250	0.000	-0.750	
x7	0.000	0.000	0.000	0.000	0.000	-1.000	-1.000	1.000	0.000	
x3	0.333	0.000	0.000	1.000	0.333	0.000	0.000	0.000	0.333	

Iteração 5 (removendo variáveis redundantes)										
	x1	x2	x3	x4						
	0.000	0.000	0.000	0.000	0.000					

x1	1.000	1.000	0.000	0.000	0.500					
x2	0.500	0.000	1.000	0.000	-0.750					
x3	0.333	0.000	0.000	1.000	0.333					

Iteração 5 (tableau final da fase 1)										
	x1	x2	x3	x4						
	0.000	0.000	0.000	0.000	0.000					

x1	1.000	1.000	0.000	0.000	0.500					
x2	0.500	0.000	1.000	0.000	-0.750					
x3	0.333	0.000	0.000	1.000	0.333					

Simplex: Fase 2

Iteração 1										
	x1	x2	x3	x4						
	-1.833	0.000	0.000	0.000	-0.083					

x1	1.000	1.000	0.000	0.000	0.500					
x2	0.500	0.000	1.000	0.000	-0.750					
x3	0.333	0.000	0.000	1.000	0.333*					

Iteração 2										
	x1	x2	x3	x4						
	-1.750	0.000	0.000	0.250	0.000					

x1	0.500	1.000	0.000	-1.500	0.000					
x2	1.250	0.000	1.000	2.250	0.000					
x4	1.000	0.000	0.000	3.000	1.000					

Solução ótima encontrada com custo 1.75000:

x =

0.50000
1.25000
0.00000
1.00000

Exemplo 3 - Solução Ótima - Removendo Variáveis Artificiais na Fase 1

m = 2
n = 2
A =

1 -2
2 -8

b =

```

2
4

c =

-5
-1

Simplex: Fase 1

Iteração 1
      | x1 | x2 | x3 | x4 |
-----|---|
-6.000 | -3.000 | 10.000 | 0.000 | 0.000 |
-----|---|
x3  2.000 | 1.000*| -2.000 | 1.000 | 0.000 |
x4  4.000 | 2.000 | -8.000 | 0.000 | 1.000 |

Iteração 2
      | x1 | x2 | x3 | x4 |
-----|---|
0.000 | 0.000 | 4.000 | 3.000 | 0.000 |
-----|---|
x1  2.000 | 1.000 | -2.000 | 1.000 | 0.000 |
x4  0.000 | 0.000 | -4.000 | -2.000 | 1.000 |

Iteração 3 (removendo variável artificial)
      | x1 | x2 |
-----|---|
0.000 | 0.000 | 4.000 |
-----|---|
x1  2.000 | 1.000 | -2.000 |
x4  0.000 | 0.000 | -4.000*|

Iteração 3 (tableau final da fase 1)
      | x1 | x2 |
-----|---|
0.000 | 0.000 | 0.000 |
-----|---|
x1  2.000 | 1.000 | 0.000 |
x2 -0.000 | -0.000 | 1.000 |

Simplex: Fase 2

Iteração 1
      | x1 | x2 |
-----|---|
10.000 | 0.000 | 0.000 |
-----|---|
x1  2.000 | 1.000 | 0.000 |
x2 -0.000 | -0.000 | 1.000 |

Solução ótima encontrada com custo -10.00000:
x =

2
-0

```

Exemplo 4 - Problema Inviável - Simplex

```

-----
m = 2
n = 2
A =

1 2
2 4

b =

1
3

c =

1
1

Simplex: Fase 1

Iteração 1
      | x1 | x2 | x3 | x4 |
-----|---|
-4.000 | -3.000 | -6.000 | 0.000 | 0.000 |
-----|---|
x3  1.000 | 1.000*| 2.000 | 1.000 | 0.000 |
x4  3.000 | 2.000 | 4.000 | 0.000 | 1.000 |

```

```

Iteração 2
      | x1 | x2 | x3 | x4 |
-----|---|
-1.000 | 0.000 | 0.000 | 3.000 | 0.000 |
-----|---|
x1  1.000 | 1.000 | 2.000 | 1.000 | 0.000 |
x4  1.000 | 0.000 | 0.000 | -2.000 | 1.000 |

```

Problema inviável.

Exemplo 5 - Problema Inviável

```

m = 3
n = 5
A =

-2  -3  5  3  -6
 1 -13  4  1  -7
 0  6  2  3  1

```

b =

```

-9
-12
 1

```

c =

```

-1
-2
 1
 2
 3

```

Simplex: Fase 1

```

Iteração 1
      | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 |
-----|---|
-22.000 | -1.000 | -22.000 | 7.000 | 1.000 | -14.000 | 0.000 | 0.000 |
-----|---|
x6  9.000 | 2.000* | 3.000 | -5.000 | -3.000 | 6.000 | 1.000 | 0.000 |
x7 12.000 | -1.000 | 13.000 | -4.000 | -1.000 | 7.000 | 0.000 | 1.000 |
x8  1.000 | 0.000 | 6.000 | 2.000 | 3.000 | 1.000 | 0.000 | 1.000 |

```

```

Iteração 2
      | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 |
-----|---|
-17.500 | 0.000 | -20.500 | 4.500 | -0.500 | -11.000 | 0.500 | 0.000 |
-----|---|
x1  4.500 | 1.000 | 1.500 | -2.500 | -1.500 | 3.000 | 0.500 | 0.000 |
x7 16.500 | 0.000 | 14.500 | -6.500 | -2.500 | 10.000 | 0.500 | 1.000 |
x8  1.000 | 0.000 | 6.000* | 2.000 | 3.000 | 1.000 | 0.000 | 1.000 |

```

```

Iteração 3
      | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 |
-----|---|
-14.083 | 0.000 | 0.000 | 11.333 | 9.750 | -7.583 | 0.500 | 3.417 |
-----|---|
x1  4.250 | 1.000 | 0.000 | -3.000 | -2.250 | 2.750 | 0.500 | -0.250 |
x7 14.083 | 0.000 | 0.000 | -11.333 | -9.750 | 7.583 | 0.500 | -2.417 |
x2  0.167 | 0.000 | 1.000 | 0.333 | 0.500 | 0.167* | 0.000 | 0.167 |

```

```

Iteração 4
      | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 |
-----|---|
-6.500 | 0.000 | 45.500 | 26.500 | 32.500 | 0.000 | 0.500 | 11.000 |
-----|---|
x1  1.500 | 1.000 | -16.500 | -8.500 | -10.500 | 0.000 | 0.500 | -3.000 |
x7  6.500 | 0.000 | -45.500 | -26.500 | -32.500 | 0.000 | 0.500 | -10.000 |
x5  1.000 | 0.000 | 6.000 | 2.000 | 3.000 | 1.000 | 0.000 | 1.000 |

```

Problema inviável.

Exemplo 6 - Problema Ilimitado - Simplex

```

m = 1
n = 3
A =

 0  1  1

```

b = 1

c =

```

-1

```

```

0
0

Simplex: Fase 1

Iteração 1
      | x1 | x2 | x3 | x4 |
-----|---|
-1.000 | -0.000 | -1.000 | -1.000 | 0.000 |
-----|---|
x4  1.000 | 0.000 | 1.000*| 1.000 | 1.000 |

Iteração 2
      | x1 | x2 | x3 | x4 |
-----|---|
0.000 | 0.000 | 0.000 | 0.000 | 1.000 |
-----|---|
x2  1.000 | 0.000 | 1.000 | 1.000 | 1.000 |

Iteração 3 (tableau final da fase 1)
      | x1 | x2 | x3 |
-----|---|
0.000 | 0.000 | 0.000 | 0.000 |
-----|---|
x2  1.000 | 0.000 | 1.000 | 1.000 |

Simplex: Fase 2

Iteração 1
      | x1 | x2 | x3 |
-----|---|
-0.000 | -1.000 | 0.000 | 0.000 |
-----|---|
x2  1.000 | 0.000 | 1.000 | 1.000 |

Problema ilimitado.

-----
Exemplo 7 - Problema Ilimitado
-----

m = 3
n = 5
A =

-5  1  1  0  0
-2  1  0 -1  0
-1  1  0  0 -1

b =

1
-1
-2

c =

1
-1
1
1
-1

Simplex: Fase 1

Iteração 1
      | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 |
-----|---|
-4.000 | 2.000 | 1.000 | -1.000 | -1.000 | -1.000 | 0.000 | 0.000 | 0.000 |
-----|---|
x6  1.000 | -5.000 | 1.000 | 1.000*| 0.000 | 0.000 | 1.000 | 0.000 | 0.000 |
x7  1.000 | 2.000 | -1.000 | -0.000 | 1.000 | -0.000 | 0.000 | 1.000 | 0.000 |
x8  2.000 | 1.000 | -1.000 | -0.000 | -0.000 | 1.000 | 0.000 | 0.000 | 1.000 |

Iteração 2
      | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 |
-----|---|
-3.000 | -3.000 | 2.000 | 0.000 | -1.000 | -1.000 | 1.000 | 0.000 | 0.000 |
-----|---|
x3  1.000 | -5.000 | 1.000 | 1.000 | 0.000 | 0.000 | 1.000 | 0.000 | 0.000 |
x7  1.000 | 2.000*| -1.000 | 0.000 | 1.000 | 0.000 | 0.000 | 1.000 | 0.000 |
x8  2.000 | 1.000 | -1.000 | 0.000 | 0.000 | 1.000 | 0.000 | 0.000 | 1.000 |

Iteração 3
      | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 |
-----|---|
-1.500 | 0.000 | 0.500 | 0.000 | 0.500 | -1.000 | 1.000 | 1.500 | 0.000 |
-----|---|
x3  3.500 | 0.000 | -1.500 | 1.000 | 2.500 | 0.000 | 1.000 | 2.500 | 0.000 |
x1  0.500 | 1.000 | -0.500 | 0.000 | 0.500 | 0.000 | 0.000 | 0.500 | 0.000 |
x8  1.500 | 0.000 | -0.500 | 0.000 | -0.500 | 1.000*| 0.000 | -0.500 | 1.000 |

```

Iteração 4									
	x1	x2	x3	x4	x5	x6	x7	x8	
	0.000	0.000	0.000	0.000	0.000	0.000	1.000	1.000	1.000

x3	3.500	0.000	-1.500	1.000	2.500	0.000	1.000	2.500	0.000
x1	0.500	1.000	-0.500	0.000	0.500	0.000	0.000	0.500	0.000
x5	1.500	0.000	-0.500	0.000	-0.500	1.000	0.000	-0.500	1.000

Iteração 5 (tableau final da fase 1)					
	x1	x2	x3	x4	x5
	0.000	0.000	0.000	0.000	0.000

x3	3.500	0.000	-1.500	1.000	2.500
x1	0.500	1.000	-0.500	0.000	0.500
x5	1.500	0.000	-0.500	0.000	-0.500

Simplex: Fase 2

Iteração 1					
	x1	x2	x3	x4	x5
	-2.500	0.000	0.500	0.000	-2.500

x3	3.500	0.000	-1.500	1.000	2.500
x1	0.500	1.000	-0.500	0.000	0.500*
x5	1.500	0.000	-0.500	0.000	-0.500

Iteração 2					
	x1	x2	x3	x4	x5
	0.000	5.000	-2.000	0.000	0.000

x3	1.000	-5.000	1.000*	1.000	0.000
x4	1.000	2.000	-1.000	0.000	1.000
x5	2.000	1.000	-1.000	0.000	0.000

Iteração 3					
	x1	x2	x3	x4	x5
	2.000	-5.000	0.000	2.000	0.000

x2	1.000	-5.000	1.000	1.000	0.000
x4	2.000	-3.000	0.000	1.000	1.000
x5	3.000	-4.000	0.000	1.000	0.000

Problema ilimitado.

Referências

- [1] D. Bertsimas & J. N. Tsitsiklis, *Introduction to Linear Optimization*, 1997, Athena Scientific.