Lista 6

May 15, 2022

1 Implementações

Fernanda Gomes e Igor Michels

[1]:
```
using LinearAlgebra;
```

1.1 Simplex duas fases

1.1.1 Forma Padrão

[2]:
```
function SEF(obj, c, A, sig, b, x_cons)
    """
    obj    : string (Min or Max)
    c      : vector of costs
    A      : matrix of coeff
    sig    : constraints' signal (vector of strings)
    b      : constraints (vector)
    x_cons : vector of constraints for x vector ("<=", ">=" or "" in relation␣
 ↪to 0)
    """

    if obj == "Min"
        obj = "Max";
        c = -c;
    end

    for i in 1:length(x_cons)
        if x_cons[i] == "<="
            A[:, i] = -A[:, i];
            c[i] = -c[i];
        elseif x_cons[i] == ""
            A = hcat(A, -A[:, i]);
            append!(c, -c[i]);
            x_cons[i] = ">=";
            append!(x_cons, [">="]);
        end
    end
```

```
    for i in 1:length(sig)
        if sig[i] == ">="
            n_lin, n_col = size(A);
            A = hcat(A, zeros(n_lin, 1));
            A[i, n_col + 1] = -1;
            sig[i] = "=";
            append!(x_cons, [">="]);
            append!(c, [0]);
        elseif sig[i] == "<="
            n_lin, n_col = size(A);
            A = hcat(A, zeros(n_lin, 1));
            A[i, n_col + 1] = 1;
            sig[i] = "=";
            append!(x_cons, [">="]);
            append!(c, [0]);
        end
    end
    return obj, c, A, sig, b, x_cons;
end;
```

1.1.2 Iteração Simplex

[3]:
```
function iteracao_simplex(c, A, b, B, N)
    for i in 1:size(b, 1)
        if b[i] < 0
            b[i] = - b[i];
            A[i, :] = - A[i, :];
        end
    end

    x = zeros(1, size(A, 2));
    x[B] = b;
    i = findfirst(c .> 0);
    if isnothing(i)
        return B, N, x;
    end

    coeff = A[:, i];
    t = b ./ coeff;
    if t[t .> 0] == []
        return "Ilimitado";
    end

    t  = minimum(t[t .> 0]);
    j = findfirst(t .== t );
    x[i] = t ;
    for k in 1:length(B)
```

```
        x[B[k]] -= coeff[k] * t ;
    end

    B[j] = i;
    N = Array(1:size(A, 2));
    N = N[N .  Ref(B)];
    sort!(B);
    sort!(N);

    return B, N, x;
end;
```

1.1.3 Forma Canônica

[4]:
```
function CF(c, A, b, B = nothing, N = nothing, z = 0)
    if isnothing(B)
        n_lin, n_col = size(A);
        corte = n_col - n_lin + 1;
        B = Array(corte:n_col);
        N = Array(1:(corte - 1));
    end

    A_B = A[:, B];
    A_B ¹ = inv(A_B);
    y = transpose(A_B ¹) * c[B];
    z = z + transpose(y) * b;
    c = c - transpose(A) * y;
    A = A_B ¹ * A;
    b = A_B ¹ * b;
    return c, A, b, B, N, z;
end;
```

1.1.4 Teste de Infactibilidade

```
[5]: function infeasibility(A, b)
    n, m = size(A);
    A_aux = hcat(A, Matrix(I, n, n));
    c = vcat(zeros(m), ones(n));
    sig = ["=" for i in 1:n];
    x_cons = [">=" for i in 1:(m + n)];
    x, z = simplex("Min", c, A_aux, sig, b, x_cons, false, false, false);
    if z == 0
        return true;
    else
        return false;
    end
end;
```

1.1.5 Implementação do Simplex

```
[6]: function simplex(obj, c, A, sig, b, x_cons, verify = true, printing = true,␣
 ↪printing_iter = false)
    if obj == "Min"
        min = true;
    else
        min = false;
    end

    obj, c, A, sig, b, x_cons = SEF(obj, c, A, sig, b, x_cons);
    B = nothing;
    N = nothing;
    z = 0;
    feasible = true;
    if verify
        feasible = infeasibility(A, b);
    end

    if ~feasible
        println("Problema infactível");
        return "Infactível";
    end

    while true
        c, A, b, B, N, z = CF(c, A, b, B, N, z);
        sol = iteracao_simplex(c, A, b, B, N);
        if printing_iter
            println("Vetor c: $c");
            println("Vetor b: $b");
            println("Matriz A:");
            for i in 1:size(A, 1)
                println(A[i, :]);
```

```
            end

            println();
        end

        if sol == "Ilimitado"
            if printing
                println("Problema ilimitado");
            end
            return "Ilimitado", Inf;
        else
            B, N, x = sol;
        end

        if c[c .> 0] == []
            if min
                z = - z;
            end

            if printing
                println("Ótimo em $x, com valor z = $z");
            end
            return x, z;
        end
    end
end;
```

1.2 Tableau

[7]:
```
function tableau(obj, A, b, c)
    """
    obj   : string ("Max" or "Min")
    A     : matrix
    b     : vector
    c     : vector
    """
    if obj == "Min"
        c = -c;
    end

    T = hcat(1, -c', 0);
    aux = hcat(zeros(size(b)), A, b);
    T = vcat(T, aux);
    n_lin, n_col = size(T);
    cond = T[1, 2:(n_col - 1)] .< 0;
    while sum(cond) > 0
        j = findfirst(cond .== 1) + 1;
```

```
        column = T[2:n_lin, j];
        b_aux = T[2:n_lin, n_col];
        b_aux = b_aux[column .!= 0] ./ column[column .!= 0];
        if b_aux[b_aux .> 0] == Float64[]
            return "Ilimitado", Inf;
        end
        i = findfirst(b_aux .== minimum(b_aux[b_aux .> 0])) + 1;
        T[i, :] = T[i, :] ./ T[i, j];
        for k in 1:n_lin
            if k != i
                T[k, :] = T[k, :] - T[k, j] * T[i, :];
            end
        end

        cond = T[1, 2:(n_col - 1)] .< 0;
    end

    x = convert(Vector{Float64}, T[1, 2:(n_col - 1)] .== 0);
    x[x .!= 0] = T[2:n_lin, 2:(n_col - 1)][:, x .!= 0] * T[2:n_lin, n_col];
    z = T[1, n_col];
    return x, z;
end;
```

[8]:
```
A = [ 1  1  1  0  0
      2  1  0  1  0
     -1  1  0  0  1];
b = [6, 10, 4];
c = [2, 3, 0, 0, 0];

sol = tableau("Max", A, b, c)
```

[8]: ([1.0, 5.0, 0.0, 3.0, 0.0], 17.0)

2 Problemas

2.1 Questão 1

Pela implementação realizada, a base inicial sempre corresponde as últimas colunas da matriz A, assim reordenamos as equações para satisfazer essa condição. Além disso, a implementação realizada foi elaborada seguindo a regra de Bland e retorna as soluções de acordo com os certificados vistos durante o curso.

[9]:
```
A = [ 2 -2  1  0
      1  3  0  1];
b = [2, 5];
c = [3, 1, 0, 0];
sig = ["=", "="];
```

```
x_cons = [">=", ">=", ">=", ">="];

simplex("Max", c, A, sig, b, x_cons, true, true, true);
```

```
Vetor c: [3.0, 1.0, 0.0, 0.0]
Vetor b: [2.0, 5.0]
Matriz A:
[2.0, -2.0, 1.0, 0.0]
[1.0, 3.0, 0.0, 1.0]

Vetor c: [0.0, 4.0, -1.5, 0.0]
Vetor b: [1.0, 4.0]
Matriz A:
[1.0, -1.0, 0.5, 0.0]
[0.0, 4.0, -0.5, 1.0]

Vetor c: [0.0, 0.0, -1.0, -1.0]
Vetor b: [2.0, 1.0]
Matriz A:
[1.0, 0.0, 0.375, 0.25]
[0.0, 1.0, -0.125, 0.25]

Ótimo em [2.0 1.0 0.0 0.0], com valor z = 7.0
```

Desfazendo o reordenamento feito, temos que $x^* = (0, 2, 1, 0)^\top$.

2.2 Questão 2

[10]:
```
A = [ 2   5   1   0   3   1
      0   2   2  -4   2  -4
      3   5   1   2   6   3];
b = [9/4, 0, 4];
c = [2, -4, 1, 4, 8, 4];
sig = ["=", "=", "="];
x_cons = [">=", ">=", ">=", ">=", ">=", ">="];

simplex("Max", c, A, sig, b, x_cons, true, true, true);
```

```
Problema infactível
```

2.3 Questão 3

[11]:
```
A = [-2   1   1   1   0   0
      1  -1   0   0   1   0
      2  -3  -1   0   0   1];
b = [1, 2, 6];
c = [2, 1, -1, 0, 0, 0];
sig = ["=", "=", "="];
```

```
x_cons = [">=", ">=", ">=", ">=", ">=", ">="];

simplex("Max", c, A, sig, b, x_cons, true, true, true);
```

```
Vetor c: [2.0, 1.0, -1.0, 0.0, 0.0, 0.0]
Vetor b: [1.0, 2.0, 6.0]
Matriz A:
[-2.0, 1.0, 1.0, 1.0, 0.0, 0.0]
[1.0, -1.0, 0.0, 0.0, 1.0, 0.0]
[2.0, -3.0, -1.0, 0.0, 0.0, 1.0]

Vetor c: [0.0, 3.0, -1.0, 0.0, -2.0, 0.0]
Vetor b: [2.0, 5.0, 2.0]
Matriz A:
[1.0, -1.0, 0.0, 0.0, 1.0, 0.0]
[0.0, -1.0, 1.0, 1.0, 2.0, 0.0]
[0.0, -1.0, -1.0, 0.0, -2.0, 1.0]

Problema ilimitado
```

2.4 Questão 5

Modelando a questão, como feito nas primeiras listas, temos que queremos maximizar o valor do anel restringindo a quantidade de impurezas, assim, queremos maximizar $x_1 + 2x_2 + 3x_3 + 2x_4$ restringindo que $x_1 + 2x_2 + 2x_3 + x_4 \leq 6$ e $x_2 + x_3 + 2x_4 \leq 10$.

[12]:
```
A = [1 2 2 1
     0 1 1 2];
b = [6, 10];
c = [1, 2, 3, 2];
sig = ["<=", "<="];
x_cons = [">=", ">=", ">=", ">="];
obj = "Max";

obj, c, A, sig, b, x_cons = SEF(obj, c, A, sig, b, x_cons);
println("Objetivo na forma padrão: $obj");
println("Vetor c na forma padrão: $c");
println("Vetor b na forma padrão: $b");
println("Matriz A na forma padrão:");
for i in 1:size(A, 1)
    println(A[i, :]);
end

println("Relação entre Ax e b na forma padrão: $sig");
println("Restrições de x na forma padrão: $x_cons");

println();
println("Otimização:");
```

```
simplex(obj, c, A, sig, b, x_cons);
```

```
Objetivo na forma padrão: Max
Vetor c na forma padrão: [1, 2, 3, 2, 0, 0]
Vetor b na forma padrão: [6, 10]
Matriz A na forma padrão:
[1.0, 2.0, 2.0, 1.0, 1.0, 0.0]
[0.0, 1.0, 1.0, 2.0, 0.0, 1.0]
Relação entre Ax e b na forma padrão: ["=", "="]
Restrições de x na forma padrão: [">=", ">=", ">=", ">=", ">=", ">="]

Otimização:
Ótimo em [0.0 0.0 0.666666666666667 4.666666666666666 0.0 0.0], com valor z =
11.333333333333332
```

2.5 Questão 6

2.5.1 Item a

No Simplex sempre buscamos uma "coordenada" em que vamos aumentar a função objetivo. Como agora queremos minimizar, devemos buscar uma "coordenada" em que vamos diminuir a função objetivo, assim, a Regra de Bland muda para pegar a primeira coordenada negativa. De modo análogo, quando não tivermos mais coordenadas negativas paramos o algoritmo no local ótimo.

2.5.2 Item b

A ideia é análoga a do simplex, sempre buscando um caminho em que vamos diminuir a função custo. Assim, com um argumento análogo ao da optimalidade do Simplex, podemos garantir que a solução é ótima.

2.5.3 Item c

A ideia é análoga a do simplex, sempre buscando um caminho em que vamos diminuir a função custo. Assim, com um argumento análogo ao da ilimitação do Simplex, podemos garantir que o problema é ilimitado.

3 Seção 2.6

3.1 Questão 1

3.1.1 Item a

[13]:
```
obj = "Max";
A = [ 1 -2  1
     -1  3 -2];
b = [2, -3];
c = [1, 3, 2];
sig = ["=" for i in 1:size(b, 1)];
x_cons = [">=" for i in 1:size(A, 2)];
```

```
simplex(obj, c, A, sig, b, x_cons);
```

```
Problema infactível
```

Aqui, apesar de termos a solução $(1, 0, 1)^\top$), terá em seu teste de infactibilidade o problema de minimizar a soma de duas variáveis auxiliares. Nesse problema, temos que a direção $(t, t, t, 2, 3)^\top$ possui valor constante na função objetivo, logo, o problema é ilimitado e, portanto, a primeira fase falha, pois o valor objetivo não foi 0.

3.1.2 Item b

[14]:
```
obj = "Max";
A = [ 2  4  2
     -1 -3 -2];
b = [2, 0];
c = [27, 2, -6];
sig = ["=" for i in 1:size(b, 1)];
x_cons = [">=" for i in 1:size(A, 2)];

simplex(obj, c, A, sig, b, x_cons);
```

```
Problema infactível
```

3.1.3 Item c

[15]:
```
obj = "Max";
A = [ 1  1  1 -1  0
      0  1  2  0  1];
b = [1, 2];
c = [0, 1, 0, -1, 0];
sig = ["=" for i in 1:size(b, 1)];
x_cons = [">=" for i in 1:size(A, 2)];

simplex(obj, c, A, sig, b, x_cons, true);
```

```
Problema infactível
```

Aqui, temos que a solução é $(0, 1, 0, 0, 1)^\top$. Novamente, o problema está na primeira fase do simplex, onde a direção $(t+1, 0, 0, t, 2, 0, 0)^\top$ tem valor constante na função objetivo, logo temos que o retorno é ilimitado e, com isso, repete-se o argumento do item a.

3.1.4 Item d

[16]:
```
obj = "Max";
A = [-1  1  0  0 -1
      1  0 -1  1  2];
b = [1, 1];
c = [2, 0, 2, 0, 3];
```

```
sig = ["=" for i in 1:size(b, 1)];
x_cons = [">=" for i in 1:size(A, 2)];

simplex(obj, c, A, sig, b, x_cons);
```

```
Problema ilimitado
```

Para ver que realmente é ilimitado, basta tomar a direção $(t, t+1, t, 1, 0)^\top$.

3.1.5 Item e

[17]:
```
obj = "Max";
A = [ 2 -1  4 -2  1
     -1  0 -3  1 -1];
b = [2, 1];
c = [-3, -1, 1, 4, 7];
sig = ["=" for i in 1:size(b, 1)];
x_cons = [">=" for i in 1:size(A, 2)];

simplex(obj, c, A, sig, b, x_cons);
```

```
Problema infactível
```