MAT 423: Homework 9 (11/14)

Ex 1. [3pts]

Let: $J(x_1, x_2) = x_1 + 5x_2$ with the constraint Ω:

$$x_1 \geq 0,\ x_2 \geq 0,$$
$$x_1 + x_2 \geq 2, \quad x_1 - x_2 \geq 0.$$

a) The domain Ω is plotted in figure 1. 1.5pt

b) There are two corners $\mathbf{x}_c^1 = (1,1)$, $\mathbf{x}_c^2 = (2,0)$ with $J(\mathbf{x}_c^1) = 6$ and $J(\mathbf{x}_c^2) = 2$. Since J is lower-bounded on Ω ($J(\mathbf{x}) \geq 0$ on Ω), there exists a minimum of J over Ω and the minimum is reached at a corner. Therefore, the minimum is at $\mathbf{x}_c^2 s$. 1.5pt

Ex 2. [3pts]

Let $J(x_1, x_2) = 2x_1 + x_2$ with the constraint Ω:

$$x_1 \geq 0,\ x_2 \geq 0,$$
$$x_1 \leq 2, \quad x_1 + x_2 \leq 2, \quad -x_1 + x_2 \leq 1.$$

a) The domain Ω is given in figure 2. 1.5pt

b) The domain Ω has 4 corners. We can easily rule out as maximum $(0,0)$ and $(0,1)$ since the gradient is pointing inside the domain Ω at these corners. Moreover, the gradient c indicates an increase of J going from corner $(\frac{1}{2}, \frac{3}{2})$ to corner $(2,0)$. Thus, the maximum of J is reached at $(2,0)$. 1.5pt

Ex 3. [4pts]

Let $J(x_1, x_2, x_3) = x_1 - x_2 + x_3$ with constraints Ω:

$$\mathbf{x} = (x_1, x_2, x_3) \geq 0$$
$$x_1 + x_2 + x_3 \geq 1, \quad -x_1 - x_2 - x_3 \geq -2.$$

a) Denote $\omega_1 = x_1 + x_2 + x_3 - 1$, $\omega_2 = -x_1 - x_2 - x_3 + 2$ and let $\tilde{\mathbf{x}} = (x_1, x_2, x_3, \omega_1, \omega_2)$. It must satisfy:

$$\tilde{\mathbf{x}} \geq 0 \quad , \qquad \tilde{A}\tilde{\mathbf{x}} = \mathbf{b} \ \text{ with } \tilde{A} = \begin{bmatrix} 1 & 1 & 1 & -1 & 0 \\ -1 & -1 & -1 & 0 & -1 \end{bmatrix} \text{ and } \mathbf{b} = (1, -2)^T.$$ 1pt

b) We try to find a corner of the form $\tilde{\mathbf{x}}_c = (x_1, 0, 0, \omega_1, 0)$. Plugin in the constraint $\tilde{A}\tilde{\mathbf{x}} = \mathbf{b}$ gives:

$$\begin{bmatrix} 1 & -1 \\ -1 & 0 \end{bmatrix} \begin{pmatrix} x_1 \\ \omega_1 \end{pmatrix} = \mathbf{b},$$

solving this system gives: $x_1 = 2$, $\omega_1 = 1$. Since the coefficients are positive, the corner $\tilde{\mathbf{x}}_c = (2, 0, 0, 1, 0)$ is feasible. 1pt

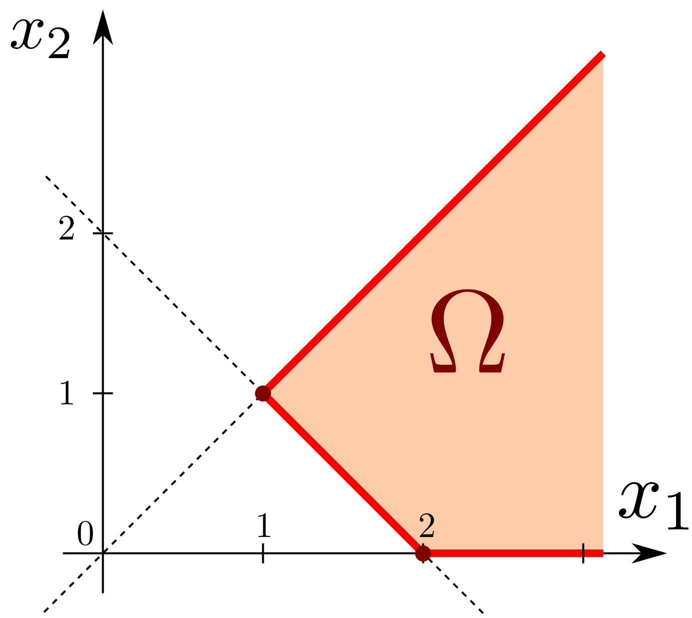

Figure 1: The domain Ω for **Ex 1**. There are two corners $(1, 1)$, $(2, 0)$.

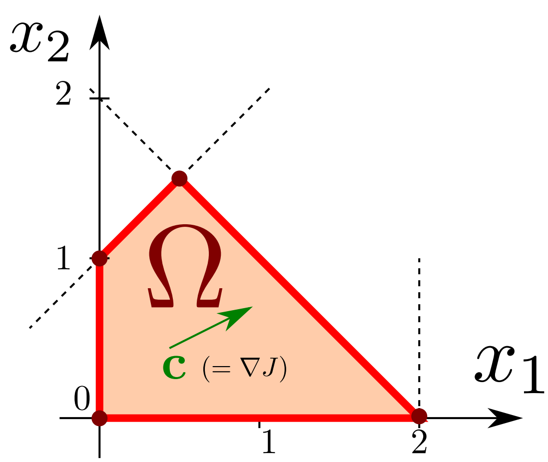

Figure 2: The domain Ω for **Ex 2**. Following the gradient of J (the vector c) leads to the corner $(2, 0)$.

c) We introduce the tableau:

$$T = \left[\begin{array}{ccccc|c} 1 & 1 & 1 & -1 & 0 & 1 \\ -1 & -1 & -1 & 0 & -1 & -2 \\ \hline 1 & -1 & 1 & 0 & 0 & 0 \end{array}\right]$$ .5pt

Reducing to obtain the corner $\tilde{\mathbf{x}}_c = (2, 0, 0, 1, 0)$ gives (making the first and fourth column 'hot-vectors'):

$$\left[\begin{array}{ccccc|c} 1 & 1 & 1 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ \hline 0 & -2 & 0 & 0 & -1 & -2 \end{array}\right].$$ .5pt

In particular, we recover that $J(\tilde{\mathbf{x}}_c) = 2$. We use x_2 has an new entry variable making the second column a 'hot-vector':

$$\left[\begin{array}{ccccc|c} 1 & 1 & 1 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ \hline 2 & 0 & 2 & 0 & 1 & 2 \end{array}\right]$$ .5pt

Since all the entries on the last line are positive, this corner is optimal. Thus, the minimum is reached at $\tilde{\mathbf{x}}_* = (0, 2, 0, 1, 0)$ where $J(\tilde{\mathbf{x}}_*) = -2$. .5pt

Ex 4.

a-b) The programs are written below.

Extra Rather than giving the next corner, we simply give the indices for the next corner. The coordinates of the corner could then be found using the previous function. 1pt

```julia
using LinearAlgebra
using IterTools

#------------------------------------------------------------------------#
#--------                    test a corner                --------------#
#------------------------------------------------------------------------#
function test_corner(A,b,index)
    """ Test if a corner with indices 'index' is feasible: A x >= b and x >=0
    """
    # init
    m,n = size(A)
    T = [A diagm(0=>-ones(m))]  # tableau
    B = T[:,index]              # select the columns with indices
    isFeasible = false
    x_c = zeros(n+m)
    # find the corner
    if (det(B)==0)
        #println("----- Singular matrix ------\n")
        return false,x_c
    else
        x_b = inv(B)*b
        x_c[index] = x_b
        # test if the corner is feasible
        if all(x_b .>= 0)
            isFeasible = true
        else
        end
        return isFeasible,x_c
    end
end
# example
A = [1 1 1;
     -1 -1 -1]
b = [1;-2]
index = [1,4]
isFeasible,x_c = test_corner(A,b,index)

#------------------------------------------------------------------------#
#--------                    test a corner                --------------#
#------------------------------------------------------------------------#
function find_corner(A,b)
    """ Try to find a corner satisfying
            A x >= b and x >=0
    """
```

```
    # init
    m,n = size(A)
    for index in subsets(1:(n+m),m)
        isFeasible,x_c = test_corner(A,b,index)
        if (isFeasible)
            println(index)
            println(x_c)
            break
        end
    end
end
# example
A = [1 1 1;
     -1 -1 -1]
b = [1;-2]
find_corner(A,b)

#------------------------------------------------------------------------#
#--------                  next step                   --------------#
#------------------------------------------------------------------------#
function next_corner(A,b,c,index)
    """
       Move to the next corner using simplex method
    """
    # init
    m,n = size(A)
    T = [A diagm(0=>-ones(m))]
    c_aug = [c;zeros(m)]
    # tableau reduced
    B = T[:,index]              # select the columns with indices
    invB = inv(B)
    T_reduced = invB*T
    # the corner
    x_b = invB*b
    # optimal condition
    r = c_aug - T_reduced'*c_aug[index]
    # check if the corner is optimal
    valueMin,indexMin = findmin(r)
    if (valueMin>=0)
        println("optimal corner")
        return index
    else
        # there is a better corner
        index_new = copy(index)
```

```
        ratio = x_b./T_reduced[:,indexMin]
        pivot,indexPivot = findmin(ratio + (ratio.<0)*10^6)
        index_new[indexPivot] = indexMin
        return index_new
    end
end
# example
A = [1 1 1;
     -1 -1 -1]
b = [1;-2]
find_corner(A,b)
c = [1;2;3]
index = [1;4]
next_corner(A,b,c,index)
```