HOMEWORK 4
Elena Gómez
February 7, 2023

PROBLEM 0: Homework checklist
I worked with Alejandro Mayo, Mario Utiel and Lucía Ostolaza

PROBLEM 1: Log-barrier terms
The basis of a class of methods known as interior point methods is that we can handle non-negativity constraints such as $x \geq 0$ by solving a sequence of unconstrained problems where we add the function $b(x;\mu) = -\mu \sum_i log(x_i)$ to the objective. Thus, we convert

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && x \geq 0. \end{aligned}$$

into

$$\text{minimize} \qquad f(x) + b(x;\mu)$$

1. Explain why this idea could work. (Hint: there's a very useful picture you should probably show here!)

In order to understand why this idea could work, let's plot $b(x;\mu)$
In this case we set $\mu = 0.1$

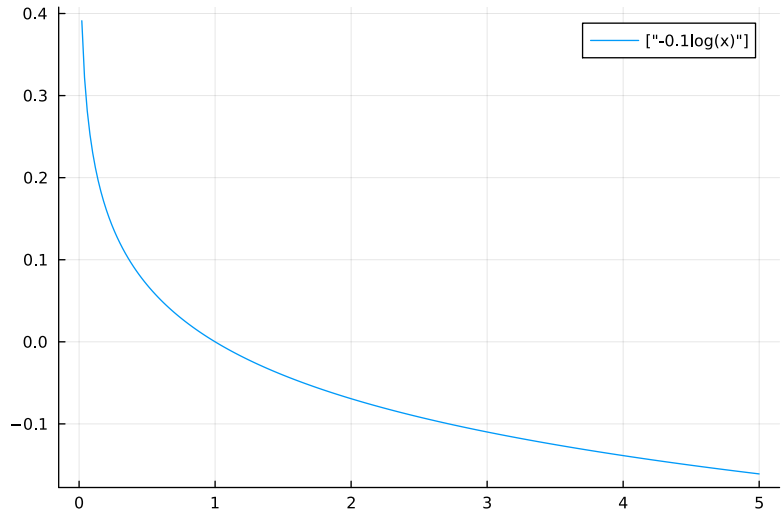
In [1]:
```
using Plots
f1(x) = -0.1*log(x)

xs = range(0, 5, length=251)

ys1 = f1.(xs)


ys1[xs .== 0.0] .= NaN
plot!(xs, ys1 ,label= ["-0.1log(x)"] )
```

Out[1]:



As the plot shows, when $x \to 0^+$, $b(x;\mu) \to +\infty$.
So, if the initial value of x is greater than 0, $x_0 > 0$, while minimizing, a negative minimizer cannot be chosen because the objective function of the new unconstrained problem becomes too big.
When $\mu \to 0^+$, $b(x;\mu) \to 0$, so $f(x) + b(x;\mu) \approx f(x)$. The solution of the new unconstrained problem tends to the solution of the initial constrained problem .\

2. Write a matrix expression for the gradient and Hessian of $f(x) + b(x;\mu)$ in terms of the gradient vector $g(x)$ and the Hessian matrix $H(x)$ of f.

Let's call the new function $t(x)$, $t(x) = f(x) + b(x;\mu)$. Let's compute $\frac{\partial t(x)}{\partial x_i}$ and $\frac{\partial^2 t(x)}{\partial x_i}$

$\frac{\partial t(x)}{\partial x_i} = \frac{\partial f(x)}{\partial x_i} + \frac{\partial b(x;\mu)}{\partial x_i} = \frac{\partial f(x)}{\partial x_i} - \mu\frac{1}{x_i}$
$\frac{\partial^2 t(x)}{\partial x_i^2} = \frac{\partial^2 f(x)}{\partial x_i^2} + \frac{\partial^2 b(x;\mu)}{\partial x_i^2} = \frac{\partial^2 f(x)}{\partial x_i^2} + \mu\frac{1}{x_i^2}$
notice that, $\forall i \neq j \frac{\partial^2 t(x)}{\partial x_i \partial x_j} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j}$

Let's write the gradient and hessian of t in terms of the gradient vector $g(x)$ and the Hessian matrix $H(x)$ of f:

$\nabla t(x) = g(x) - \mu[\frac{1}{x_i}]_i$
$\nabla^2 t(x) = H(x) + \mu \cdot diag(\frac{1}{x_i^2})_i$

3. (Open ended) Explain if there is anything special about log that makes it especially useful here. For instance, are there other functions that could work and give you the same effect? Would log be superior?

One property of the log function is that it is a **strictly increasing function**, as its arguments increases, its value increases as well. It allows a creation of a **barrier** that grows more and more steep as one approaches the boundary of the feasible region. Creating a barrier that discourages solutions from moving outside the feasible region. Another important property of the log function is that it is approaches **negative infinity** as its argument approaches **zero**, and approaches **positive infinity** as its arguments approaches **positive infinity**. For this reason, it is really useful for defininig constraints, because it ensures that the optimization problem is feasible only if certain conditions are met. There are other functions that could work as barrier functions, for example, the square root function or the inverse function. But, the log function is preferred because it is **easier to differentiate** and because of **its mathematical properties**.

PROBLEM 2:Inequality constraints
Draw a picture of the feasible region for the constraints:

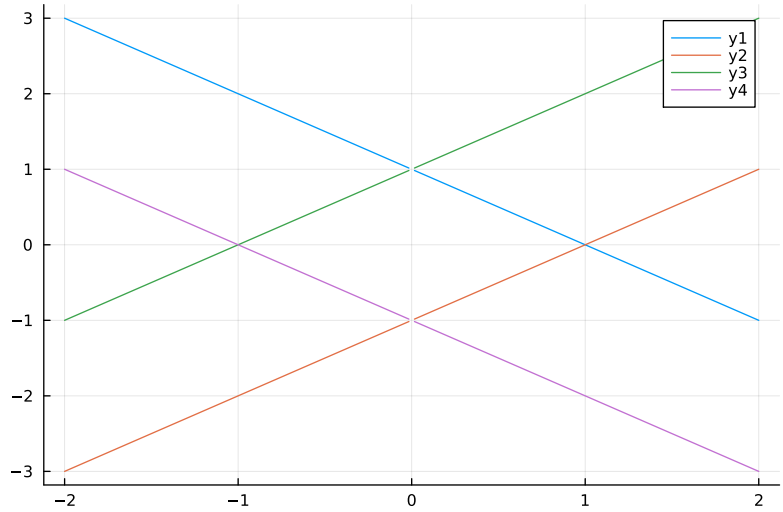
$$\begin{bmatrix} 1 - x_1 - x_2 \\ 1 - x_1 + x_2 \\ 1 + x_1 - x2 \\ 1 + x_1 + x_2 \end{bmatrix} \geq 0$$

Let's plot these constraints.

In [2]:

```
using Plots
f1(x) = 1-x
f2(x) = x-1
f3(x) = x+1
f4(x) = -x-1
xs = range(-2, 2, length=251)
ys1 = f1.(xs)
ys2 = f2.(xs)
ys3 = f3.(xs)
ys4 = f4.(xs)
ys1[xs .== 0.0] .= NaN
ys2[xs .== 0.0] .= NaN
ys3[xs .== 0.0] .= NaN
ys4[xs .== 0.0] .= NaN
plot(xs, [ys1,ys2,ys3,ys4])
```

Out[2]:



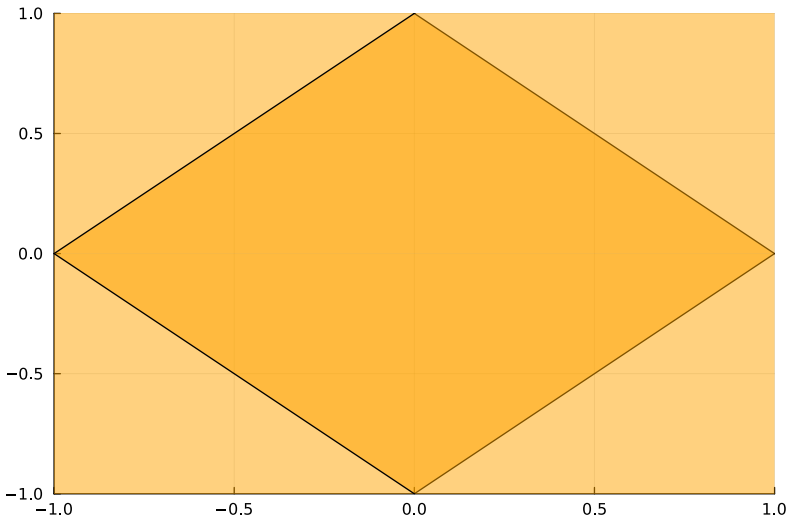
In [3]:

```
using Plots

fs1(x1) = 1 - x1
fs2(x1) = x1 - 1
fs3(x1) = x1 + 1
fs4(x1) = -x1 - 1

x1 = -1:0.01:1
p = plot(x1, [fs1.(x1), fs2.(x1), fs3.(x1), fs4.(x1)], fill=(0, .5, :orange),
         fillalpha=0.5,
         xlim=(-1, 1),
         ylim=(-1, 1),
         linecolor=:black,
         legend=false)

display(p)
```



In order to draw a picture of the feasible region for the constraints above, we plot the 4 functions in the same figure. The feasible region is the region created by the intersection of the functions. As the plot shows, we obtain a paralelogram with 4 vertex and 4 sides. The center of the region is at (0,0). We can see it augmented in the second plot, the darker region is the feasible region.

PROBLEM 3:Necessary and sufficient conditions
Let $f(x) = \frac{1}{2}x^T Q x - x^T c$.

1. Write down the necessary conditions for the problem:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \geq 0. \end{array}$$

The **necessary conditions** for $x^* \in R^n$ to be a local minimizer are the following ones:
$\exists \lambda^* \in R^m$
$x^* \geq 0$, this condition ensures that the optimization variable satisfies the constraint

$\lambda^* \geq 0$, this condition ensures that the Lagrange multiplier is non-negative.

$g(x^*) = \lambda^*$

$Z^T H(x^*) Z \geq 0$, where Z is a basis of the feasible conditions. And can change depending on active constrains. If no constraint is active, then $Z = I$. If the i^{th} constraint is active, $e_i \in Z$. And if all constrains are active, the feasible region is a point and $Z = 0$

$(\lambda^*)^T x^* = 0$

2. Write down the sufficient conditions for the same problem.

The **sufficient conditions** for $x^* \in R^n$ to be a local minimizer are the following ones:
$\exists \lambda^* \in R^m$
$x^* \geq 0$, this condition ensures that the optimization variable satisfies the constraint

$\lambda^* \geq 0$, this condition ensures that the Lagrange multiplier is non-negative.

$g(x^*) = \lambda^*$

$Z^T H(x^*) Z > 0$, where Z is a basis of the feasible conditions. And can change depending on active constrains. If no constraint is active, then $Z = I$. If the i^{th} constraint is active, $e_i \in Z$. And if all constrains are active, the feasible region is a point and $Z = 0$

$(\lambda^*)^T x^* = 0$

3. Consider the two-dimensional case with

$Q = \begin{bmatrix} 2 & 3 \\ 3 & 2 \end{bmatrix} \quad c = \begin{bmatrix} -1.5 \\ 1.5 \end{bmatrix}$

Determine the solution to this problem by any means you can, and justify your work.

$f(x) = \frac{1}{2}x^T Q x - x^T c$ and $Q = \begin{bmatrix} 2 & 3 \\ 3 & 2 \end{bmatrix} \quad c = \begin{bmatrix} -1.5 \\ 1.5 \end{bmatrix}$

So, if we substitute Q and c

$f(x) = \frac{1}{2}[x_1, x_2]^T \begin{bmatrix} 2 & 3 \\ 3 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - [x_1, x_2]^T \begin{bmatrix} -1.5 \\ 1.5 \end{bmatrix} = \frac{1}{2}[2x_1 + 3x_2, 3x_1 + 2x_2]^T \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + 1.5x_1 - 1.5x_2 = \frac{1}{2}(2x_1^2 + 3x_1x_2 + 3x_1x_2 + 2x_2^2)$

$+ 1.5x_1 - 1.5x_2 = \frac{1}{2}(2x_1^2 + 6x_1x_2 + 2x_2^2) + 1.5x_1 - 1.5x_2 = x_1^2 + 3x_1x_2 + x_2^2 + 1.5x_1 - 1.5x_2$

And

$g(x) = g(x_1, x_2) = (2x_1 + 3x_2 + 1.5)\imath + (3x_1 + 2x_2 - 1.5)\jmath$

We can compute the solution with Julia using $f(x)$ and $g(x)$:

In [4]:

```
using Plots, LinearAlgebra, Random, SparseArrays
using Optim
fun(x) = (1/2)*(2*x[1]^2 + 6*x[1]*x[2] + 2*x[2]^2) +1.5*x[1] - 1.5*x[2]

function fun_grad!(g, x)
g[1] = 2*x[1]+3*x[2]+1.5
g[2] =3x[1]+2x[2]-1.5
end

function fun_hess!(h, x)
h[1, 1] =2
h[1, 2] = 3
h[2, 1] = 2
h[2, 2] = 3
end
```

Out[4]:

```
fun_hess! (generic function with 1 method)
```

We add a lower bound in order to fulfill the constraint

As the lower bound is $(0, 0)$, we set an $x_0 = (0.0000000000000001, 0.0000000000000001)$, values close to 0 but not 0 because if not the algorithm does not work

In [5]:

```
x0 = [0.0000000000000001, 0.0000000000000001]
df = TwiceDifferentiable(fun, fun_grad!, fun_hess!, x0)

lx = [0.0, 0.0]; ux = fill(Inf, 2)
dfc = TwiceDifferentiableConstraints(lx, ux)

solution = optimize(df, dfc, x0, IPNewton())
```

Out[5]:

```
 * Status: success

 * Candidate solution
    Final objective value:     -5.625000e-01

 * Found with
    Algorithm:     Interior Point Newton

 * Convergence measures
    |x - x'|               = 3.30e-09 ≤/0.0e+00
    |x - x'|/|x'|          = 4.39e-09 ≤/0.0e+00
    |f(x) - f(x')|         = 0.00e+00 ≤ 0.0e+00
    |f(x) - f(x')|/|f(x')| = 0.00e+00 ≤ 0.0e+00
    |g(x)|                 = 3.75e+00 ≤/1.0e-08

 * Work counters
    Seconds run:   0  (vs limit Inf)
    Iterations:    83
    f(x) calls:    126
    ∇f(x) calls:   126
```

In [6]:

```
sol=solution.minimizer
```

Out[6]:

```
2-element Vector{Float64}:
 4.0975489566527573e-151
 0.7499999967040356
```

In [7]:

```
print("x_1=",sol[1])
print(" ")
print(" ")
print(" ")
print(" ")
print("x_2=",sol[2])
```

```
x_1=4.0975489566527573e-151    x_2=0.7499999967040356
```

As x_1 is a value close to 0, and x_2 is a value close to 0.75, we can say that the minimizer is $x = (0, 0.75)$

4. Produce a Julia or hand illustration of the solution showing the function contours, and gradient. What are the active constraints at the solution? What is the value of λ in $A^T\lambda = g$?

Let's plot the function contours and gradient:

In [8]:

```
using Plots
gr()


function e()
  X = range(0, stop=2, length=50)
  Y = range(0, stop=2, length=50)
  f(x, y) = (1/2)*(2*x^2 + 6*x*y + 2*y^2) +1.5*x - 1.5*y

  contour(X, Y, f)

  x = range(0, stop=2, length=15)
  y = range(0, stop=2, length=15)

  df(x, y) = [2*x+3*y+1.5; 3x+2y-1.5] /25
  quiver!(repeat(x,11), vec(repeat(y',11)), quiver=df, c=:blue)

  xlims!(0, 2)
  ylims!(0, 2)


end
```

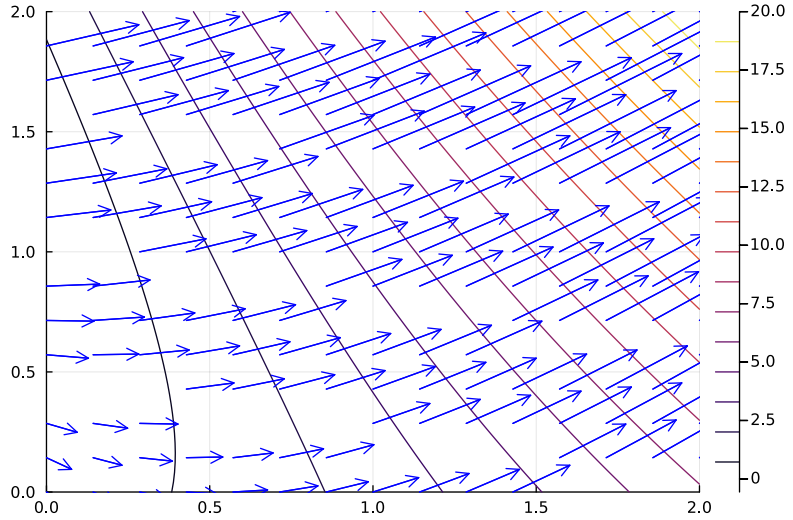
Out[8]:

```
e (generic function with 1 method)
```

In [9]:

```
e()
```

Out[9]:



All the constraints are active at the solution.
The value of λ is equal to the gradient at the solution($g(x) = \lambda$).
$g(x) = Q^T x - c$, $\lambda = Q^T x - c$. Let's compute the value of λ

$$\lambda = Q^T x - c = \begin{bmatrix} 2 & 3 \\ 3 & 2 \end{bmatrix}^T \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} -1.5 \\ 1.5 \end{bmatrix} = \begin{bmatrix} 2x_1 + 3x_2 \\ 3x_1 + 2x_2 \end{bmatrix} - \begin{bmatrix} -1.5 \\ 1.5 \end{bmatrix} = \begin{bmatrix} 2x_1 + 3x_2 + 1.5 \\ 3x_1 + 2x_2 - 1.5 \end{bmatrix}$$

As $x = [0, 0.75]$, we substitute x_1 and x_2, and we get:

$$\lambda = \begin{bmatrix} 3.75 \\ 0 \end{bmatrix}$$

5. What changes when we set $Q = \begin{bmatrix} -2 & 3 \\ 3 & -2 \end{bmatrix}$?

Let's substitute Q and compute the solution

$$f(x) = \frac{1}{2}[x_1, x_2]^T \begin{bmatrix} -2 & 3 \\ 3 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - [x_1, x_2]^T \begin{bmatrix} -1.5 \\ 1.5 \end{bmatrix} = \frac{1}{2}[-2x_1 + 3x_2, 3x_1 - 2x_2]^T \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + 1.5x_1 - 1.5x_2$$
$$= \frac{1}{2}(-2x_1^2 + 3x_1x_2 + 3x_1x_2 - 2x_2^2) + 1.5x_1 - 1.5x_2 = \frac{1}{2}(-2x_1^2 + 6x_1x_2 - 2x_2^2) + 1.5x_1 - 1.5x_2 = -x_1^2 + 3x_1x_2 - x_2^2 + 1.5x_1 - 1.5x_2$$

And
$g(x) = g(x_1, x_2) = (-2x_1 + 3x_2 + 1.5)\imath + (3x_1 - 2x_2 - 1.5)\jmath$

We can compute the solution with Julia using $f(x)$ and $g(x)$:

In [10]:

```
using Plots, LinearAlgebra, Random, SparseArrays
using Optim
fun(x) = (1/2)*(-2*x[1]^2 + 6*x[1]*x[2] - 2*x[2]^2) +1.5*x[1] - 1.5*x[2]

function fun_grad!(g, x)
g[1] = -2*x[1]+3*x[2]+1.5
g[2] =3x[1]-2x[2]-1.5
end

function fun_hess!(h, x)
h[1, 1] =-2
h[1, 2] = 3
h[2, 1] = 3
h[2, 2] = -2
end
```

Out[10]:

```
fun_hess! (generic function with 1 method)
```

In [11]:

```
x0 = [0.0000000000000001, 0.0000000000000001]
df = TwiceDifferentiable(fun, fun_grad!, fun_hess!, x0)

lx = [0.0, 0.0]; ux = fill(Inf, 2)
dfc = TwiceDifferentiableConstraints(lx, ux)

solution = optimize(df, dfc, x0, IPNewton())
```

Out[11]:

```
 * Status: success (objective increased between iterations)

 * Candidate solution
    Final objective value:     -1.123681e-16

 * Found with
    Algorithm:     Interior Point Newton

 * Convergence measures
    |x - x'|               = 0.00e+00 ≤ 0.0e+00
    |x - x'|/|x'|          = 0.00e+00 ≤ 0.0e+00
    |f(x) - f(x')|         = 0.00e+00 ≤ 0.0e+00
    |f(x) - f(x')|/|f(x')| = 0.00e+00 ≤ 0.0e+00
    |g(x)|                 = 1.50e+00 ≤/1.0e-08

 * Work counters
    Seconds run:   0  (vs limit Inf)
    Iterations:    3
    f(x) calls:    6
    ∇f(x) calls:   6
```

In [12]:

```
sol=solution.minimizer
```

Out[12]:

```
2-element Vector{Float64}:
 3.0442388555103338e-19
 7.521646713046171e-17
```

In [13]:

```
print("x_1=",sol[1])
print(" ")
print(" ")
print(" ")
print(" ")
print("x_2=",sol[2])
```

```
x_1=3.0442388555103338e-19    x_2=7.521646713046171e-17
```

When we change Q the minimizer is close to $(0, 0)$, so we can say that the minimizer $x^* = (0, 0)$

PROBLEM 4:Constraints can make a non-smooth problem smooth.
Show that
minimize $\sum_i max(a_i^T x - b_i, -0.5)$
can be reformulated as a constrained optimization problem with a continously differentiable objective function and both linear equality and inequality constraints

Let's reformulate this as a constrained optimization problem with a continously differentiable objective function both linear equality and inequality constraints. A solution of the new problem would also solve the initial problem

Notice that: $\forall i \in \{1, 2, 3, \ldots, n\}$, $a_i^T x - b_i \geq -0.5 \rightarrow max(a_i^T x - b_i, -0.5) = a_i^T x - b_i$

$$\begin{aligned} \underset{c}{\text{minimize}} \quad & e^T c \\ \text{subject to} \quad & c = a_i^T x - b_i \text{ and } c \geq -0.5. \end{aligned}$$

Where the new objective function is the sum of all components in c and it is continuously differentiable