Solutions to Part B of Problem Sheet 1

Solution (1.4)

(a) The unit circle with respect to the ∞-norm is the square with corners $(\pm 1, \pm 1)^\top$.

(b) The trick in transforming the unconstrained problem

$$\text{minimize} \quad \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_\infty \tag{5}$$

into a constrained linear programming problem is to characterise the ∞-norm as the solution of a minimization problem. In fact, for any set of numbers $x_1, \dots, x_n$,

$$\max_{1\leq i\leq n} |x_i| = \min_{\forall i\colon\, |x_i|\leq t} t.$$

Put simply, the *maximum* of a set of non-negative numbers is the *smallest* upper bound on these numbers. We can further replace the condition $|x_i| \leq t$ by $-t \leq x_i \leq t$, so that the problem (5) becomes

$$\begin{aligned} \underset{(\boldsymbol{x},t)}{\text{minimize}} \quad & t \\ \text{subject to} \quad & -t \leq \boldsymbol{a}_1^\top \boldsymbol{x} \leq t \\ & \cdots \\ & -t \leq \boldsymbol{a}_m^\top \boldsymbol{x} \leq t, \end{aligned} \tag{6}$$

where $\boldsymbol{a}_i^\top$ are the rows of the matrix $\boldsymbol{A}$. This problem can be brought into *standard form* by replacing each condition with the pair of conditions

$$\begin{aligned} \boldsymbol{a}_i^\top \boldsymbol{x} - t &\leq 0 \\ -\boldsymbol{a}_i^\top \boldsymbol{x} - t &\leq 0. \end{aligned}$$

The solution $\boldsymbol{x}$ of Problem (5) can be read off the solution $(\boldsymbol{x}, t)$ of Problem (6).

Solution (1.5)

(a) This function is not convex. There are various ways of deriving this, for example, Theorem 2.4(2), where one verifies that the Hessian, or second derivative, is $-1/x^2$, which is not positive semidefinite.

Alternatively, one can also prove the statement using a pedestrian approach. We have to show that there are points $y \neq x$ and $\lambda \in [0, 1]$ such that

$$\log(\lambda x + (1-\lambda)y) > \lambda \log(x) + (1-\lambda)\log(y).$$

Let's choose $y = 0$. Then what needs to be shown is that for the points $\boldsymbol{p}_1 = (1, 0)$ and $\boldsymbol{p}_2 = (x, \log(x))$, the line joining $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ lies *below* the curve $(t, \log(t))$ between 1 and x. The line is given by the equation

$$\ell(t) = \frac{\log(x)}{x-1}(t-1).$$

Evaluating this, for example, at $x = 2$ and $t = 1.5$, one sees that $\ell(t) > \log(t)$, which is enough evidence that $\log(t)$ is not convex. With a little more effort one can deduce that the function is actually concave.

(b) The function $f(x) = x^4$ is convex, as we will verify using Theorem 2.4. First, note that the derivative $4x^3$ is an increasing function with x. Given two points (x, x^4) and (y, y^4) with $y > x$, the line connecting them has slope $(y^4 - x^4)/(y - x)$. By the mean value theorem, there exists a $z \in (x, y)$ such that

$$\frac{y^4 - x^4}{y - x} = f'(z) = 4z^3 \geq 4x^3.$$

Rearranging this inequality, we get

$$f(y) - f(x) = y^4 - x^4 \geq 4x^3(y - x) = f'(x)(y - x),$$

which is precisely the criterium for convexity in Theorem 2.4(1).

(c) Using Theorem 2.4(2), we compute the Hessian as

$$\nabla^2 f(\boldsymbol{x}) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

This matrix is positive semidefinite on $\mathbb{R}^2_{++}$, since for all $\boldsymbol{x} \in \mathbb{R}^2_{++}$ we have

$$\boldsymbol{x}^\top \nabla^2 f(\boldsymbol{x}) \boldsymbol{x} = 2x_1 x_2 > 0.$$

It follows that the function $f(\boldsymbol{x}) = x_1 x_2$ is convex.

(d) The Hessian matrix of $f(\boldsymbol{x}) = x_1/x_2$ is

$$\nabla^2 f(\boldsymbol{x}) = \begin{pmatrix} 0 & -\frac{1}{x_2^2} \\ -\frac{1}{x_2^2} & 2\frac{x_1}{x_2^3} \end{pmatrix}.$$

This matrix is not positive semidefinite for all valid values of $\boldsymbol{x}$ (take for example $\boldsymbol{x} = (1, 1)^\top$, which leads to a negative eigenvalue).

(e) The function $e^x - 1$ is convex, as is easily seen using Theorem 2.4(2) by computing the second derivative.

(f) The function $f(\boldsymbol{x}) = \max_i x_i$ is convex. Here, we can't use the criteria from Theorem 2.4 since the function is not differentiable, so we have to verify convexity directly:

$$\max_i \lambda x_i + (1 - \lambda) y_i \leq \lambda \max_i x_i + (1 - \lambda) \max_i y_i.$$

Solution (1.6) We want to apply gradient descent to the function

$$f(\boldsymbol{x}) = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2.$$

The gradient is given by

$$\nabla f(\boldsymbol{x}) =$$

The Python implementation, using numpy, looks as follows.

In [1]:

```
import numpy as np
import numpy.linalg as la

def graddesc(A, b, x, tol):
    # Compute the negative gradient r = A^T(b-Ax)
    r = np.dot(A.transpose(),b-np.dot(A,x))
    # Start with an empty array
    xout = []
    while la.norm(r,2) > tol:
        # If the gradient is bigger than the tolerance
        Ar = np.dot(A,r)
        alpha = np.dot(r,r)/np.dot(Ar,Ar)
        x = x + alpha*r
        xout.append(x)
        r = r-alpha*np.dot(A.transpose(),Ar)
    return np.array(xout).transpose()

A = np.array([[1,2], [2,1], [-1,0]])
b = np.array([10, -1, 0])
tol = 1e-4
x = np.zeros(2)

traj = graddesc(A, b, x, tol)
```

We can plot the trajectory on top of a contour plot.

In [2]:

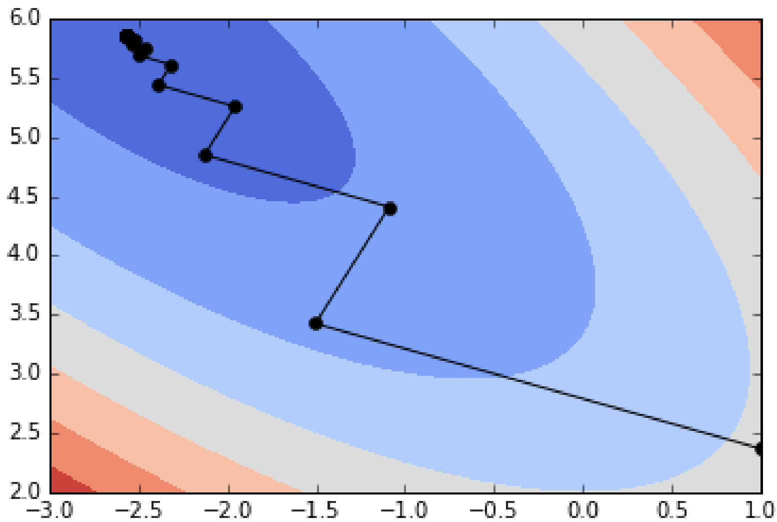
```
import matplotlib.pyplot as plt
% matplotlib inline

# Define the function we aim to minimize
def f(x):
    return np.dot(np.dot(A,x)-b,np.dot(A,x)-b)

# Create a mesh grid
xx = np.linspace(-3,1,100)
yy = np.linspace(2,6,100)
X, Y = np.meshgrid(xx, yy)
Z = np.zeros(X.shape)
for i in range(Z.shape[0]):
    for j in range(Z.shape[1]):
        Z[i,j] = f(np.array([X[i,j], Y[i,j]]))

# Get a nice monotone colormap
cmap = plt.cm.get_cmap("coolwarm")

# Plot the contours and the trajectory
plt.contourf(X, Y, Z, cmap = cmap)
plt.plot(traj[0,:], traj[1,:], 'o-k')
plt.show()
```


6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
−3.0
−2.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0