Problem Sheet 5

Problems in Part A will be discussed in class. The Problem in Part B can be tried at home.

Part A

(5.1) Recall that to a linear programming system in standard primal form

$$\text{minimize} \quad \langle \boldsymbol{c}, \boldsymbol{x} \rangle \qquad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b},\ \boldsymbol{x} \geq \boldsymbol{0} \tag{P}$$

we can associate a function

$$F(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{s}) = \begin{pmatrix} \boldsymbol{A}^\top \boldsymbol{y} + \boldsymbol{s} - \boldsymbol{c} \\ \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \\ \boldsymbol{X}\boldsymbol{S}\boldsymbol{e} \end{pmatrix}$$

that arises from the optimality conditions without the inequalities. Consider the linear programming problem

$$\text{minimize} \quad x_1 \quad \text{subject to} \quad x_1 + x_2 = 1,\ x_1 \geq 0,\ x_2 \geq 0. \tag{1}$$

(a) Write down the equation $F(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{s})$ and the Jacobian $\boldsymbol{J}F$.

(b) Solve the optimization problem (1) graphically, or by any other means.

(c) Compute the solution to $F(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{s}) = \boldsymbol{0}$ (either using Newton's method or by any other means) and conclude that it is unrelated to the solution of the optimization problem.

This problem shows that one cannot just solve for the optimality conditions without taking into account the inequalities.

(5.2) Translate the following optimization problem

$$\text{minimize} \quad \|\boldsymbol{x}\|_1 \qquad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{2}$$

into an equivalent linear programming problem of the form

$$\text{maximize} \quad \langle \boldsymbol{c}, \boldsymbol{z} \rangle \qquad \text{subject to} \quad \boldsymbol{A}'\boldsymbol{z} \leq \boldsymbol{b}'$$

with $\boldsymbol{z} \in \mathbb{R}^{2n}$ and some suitable vector $\boldsymbol{A}'$,$\boldsymbol{b}'$ and $\boldsymbol{c}$. By equivalent we mean that any solution of (2) can be recovered from a solution of the linear programming problem.

The importance of this problem comes from *compressed sensing*: if one wants to find a solution of $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ with less rows than columns, then there are infinitely many solutions. The above optimization problem often finds a *sparsest* solution, which one with the least number of non-zero entries.

Part B

(5.3) Describe the dual problem to the problem in (5.3).

(5.4) (Compressive Sensing) Consider a signal $f\colon [0,2\pi] \to \mathbb{R}$ with the property that $f(0) = f(2\pi)$. In practice, one often does not see the whole signal, but only samples certain values at discrete time intervals. It turns out that optimization can help to reconstruct a signal using much fewer samples than commonly thought possible.

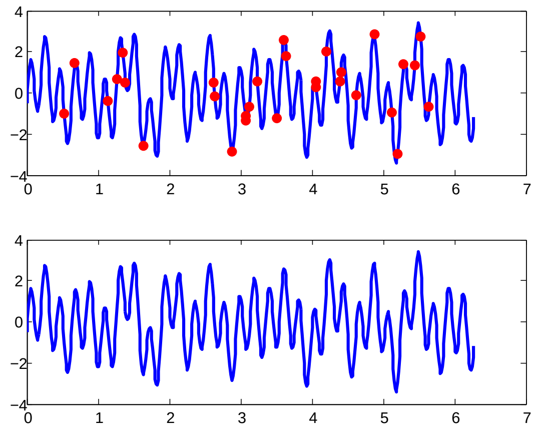

Figure 1: Signal sampled at 30 points, and reconstructed from these.

To understand how the reconstruction from few samples works, we have to look at the Fourier Transform. A periodic function can be written as a Fourier Series

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos(nx) + b_n \sin(nx).$$

Setting $c_n = (a_n + ib_n)/2$, $c_{-n} = (a_n - ib_n)/2$ for $n > 0$, and $c_0 = a_0/2$, the series can also be written in exponential form as as

$$f(x) = \sum_{n=-\infty}^{\infty} c_n e^{inx}, \tag{1}$$

where $e^{ix} = \cos(x) + i\sin(x)$ and $i = \sqrt{-1}$. While this representation involves complex numbers, the resulting function is real due to the way the imaginary parts in the summands combine. We can obtain the coefficients c_n in the series (1) by computing

$$c_n = \hat{f}(n) := \frac{1}{2\pi}\int_0^{2\pi} f(x)e^{-inx}\, dx.$$

The operation $f \mapsto \hat{f}$ is called the *Fourier Transform*. A characteristic feature of many signals is that they are *sparse* in the Fourier domain, meaning that only very few summands in the expansion (1) are necessary to describe the signal accurately (in two dimensions, this principle is the basis of the JPEG image compression standard). For the particular function shown in Figure 1, the representation is

$$f(x) = 0.5\sin(5x) + 0.5\cos(9x) - \cos(11x) + 0.2\sin(13x) + 1.7\sin(30x). \tag{2}$$

Often we are not so much interested in the analytic expression for the function f but in its values at a discrete set of points

$$x_0 = 0, \quad x_n = 2\pi, \quad x_k = \frac{2\pi k}{n}.$$

The goal of reconstructing f then becomes the reconstruction of *all* values $f_j = f(x_j)$ from the knowledge of only a few samples f_k, $k \in I \subset \{0,\dots,n-1\}$, $|I| = m < n$.

The function is now represented by a *vector* $\boldsymbol{f} = (f_0,\dots,f_{n-1})^\top$. The *discrete Fourier transform* $\mathrm{DFT}_n(f)$ is a vector $\boldsymbol{c} = (c_0,\dots,c_{n-1})^\top$ such that

$$f_j := f(x_j) = \frac{1}{n}\sum_{k=0}^{n-1} c_k e^{ik\frac{2\pi j}{n}} \tag{3}$$

for $j = 0,\dots,n-1$. Computing the DFT amounts to solving a linear system

$$\boldsymbol{f} = \boldsymbol{D}\boldsymbol{c}, \tag{4}$$

where the matrix $\boldsymbol{D}$ has the entries $\boldsymbol{D} = (e^{i\frac{2\pi jk}{n}}/n)_{0\leq j,k\leq n-1}$. The DFT can be computed using the Fast Fourier Transform in $O(n\log n)$ operations. Vectors $\boldsymbol{f}$ that come from the discretisation of signals like (2) have the property that the *Fourier coefficients* $\mathrm{DFT}_n(f) = \boldsymbol{c}$ are *sparse*, i.e., have only few non-zero entries, see Figure 2.

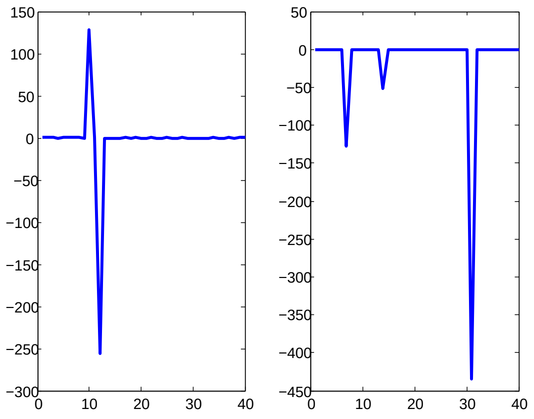

Figure 2: Sparse DFT for signal from Figure 1.

The sparsity of DFT vector is the key reason why the following approach for reconstructing $\boldsymbol{f}$ from few samples $\boldsymbol{f}_I$ works. Consider the optimization problem

$$\begin{aligned} \text{minimize} \quad & \|\boldsymbol{c}\|_1, \\ \text{subject to} \quad & \boldsymbol{D}_I\boldsymbol{c} = \boldsymbol{f}_I, \end{aligned} \tag{5}$$

with the function given as in (2), where $I \subset [n]$ and $\boldsymbol{D}_I$ is the matrix consisting of the rows indexed by I and $\boldsymbol{f}_I$ the subvector indexed by the entries of I (the red dots in Figure 1). For example,

$$\boldsymbol{D} = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 5 \end{pmatrix}, \; I = \{1,3\}, \quad \Longrightarrow \boldsymbol{D}_I = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 4 & 5 \end{pmatrix}$$

In words, we are looking for a vector $\boldsymbol{c}$ of minimal 1-norm that satisfies *a small part* of the quadratic system of equations (4). Once we have such a solution $\hat{\boldsymbol{c}}$, the hope is that $\boldsymbol{D}\hat{\boldsymbol{c}} = \boldsymbol{f}$ gives back the full vector.

(a) Formulate the conditions $\boldsymbol{D}_I\boldsymbol{c} = \boldsymbol{f}_I$ as *linear* constraints involving real numbers. Hint: split $\boldsymbol{c}$ and $\boldsymbol{D}_I$ in real and imaginary parts and reformulate the matrix-vector product accordingly.

(b) Solve the optimization problem (5) as follows.

- Set $n = 512$ and generate points $x_i = 2\pi j/n$ with corresponding values $f_j = f(x_j)$.
- Generate the matrix $\boldsymbol{D}$ (in Python, using `numpy.ifft;`) and choose a random set of 50 indices to generate a matrix $\boldsymbol{D}_I$ and $\boldsymbol{f}_I$.
- Using CVX, solve the optimization problem (5) and compare the computed vector $\hat{\boldsymbol{f}}$ with the original $\boldsymbol{f}$.

(c) Repeat the experiment in Part (b) with different values of m in order to determine how many samples are necessary to reconstruct the vector $\boldsymbol{f}$ accurately.

(d) Reformulate (5) as a linear programming problem.