Problem Sheet 4

Problems in Part A will be discussed in class. Problems in Part B come with solutions and should be tried at home.

Part A

(4.1) For the following linear programming problems,

$$\begin{array}{ll} \text{maximize} & x_1 + 2x_2 \\ \text{subject to} & x_1 + x_2 \leq 2 \\ & x_1 - x_2 \leq 1 \\ & x_1 \geq -1 \end{array} \tag{LP1}$$

$$\begin{array}{ll} \text{maximize} & x_1 + x_2 \\ \text{subject to} & x_2 - x_1 \leq 2 \\ & x_1 + x_2 \leq 8 \\ & x_1 + 2x_2 \leq 10 \\ & x_1 \leq 4 \\ & x_1 \geq 0 \\ & x_2 \geq 0. \end{array} \tag{LP2}$$

(a) Sketch the polyhedron of feasible points and find the vertices;

(b) Find a solution, if it exists (you may use a computer program such as CVXPY in Python or CVX in MATLAB).

(4.2) Given a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$ such that the polyhedron $P = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}\}$ is not empty and bounded. Show that if the optimal value of

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

is finite, it is attained at a vertex $\boldsymbol{x}^*$ of P.

(4.3) Formulate the following optimization problem as a linear programming problem,

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

and describe the dual problem.

(4.4) Show that there exists a vector $\boldsymbol{x} \neq \boldsymbol{0}$ satisfying

$$\boldsymbol{x} \geq 0, \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{0} \tag{P}$$

if and only if there is no vector $\boldsymbol{y}$ such that

$$\boldsymbol{A}^\top \boldsymbol{y} > \boldsymbol{0}. \tag{D}$$

Give a geometric interpretation of this fact.

Part B

(4.5) (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.

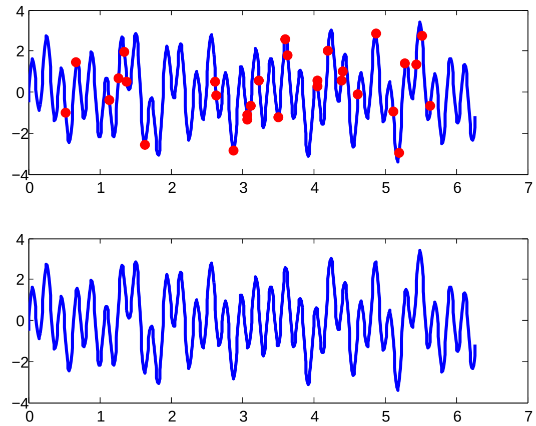

Figure 1: Signal sampled at 30 points, and reconstructed from these.

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

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

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

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

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

$$\boldsymbol{f} = \boldsymbol{D}\boldsymbol{c}, \quad (4)$$

where the matrix $\boldsymbol{D}$ has the entries $\boldsymbol{D} = (e^{i\frac{2\pi jk}{n}}/n)_{0\le j,k\le n-1}$. 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.

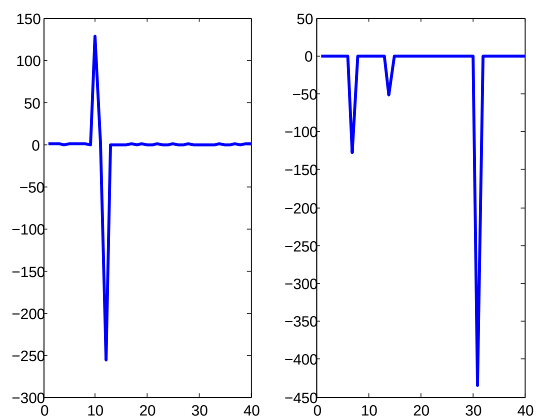

Figure 2: Sparse DFT for signal from Figure 1.

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

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

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.