Technische Universität Hamburg
Mathematical Image Processing (Winter Term 2022/2023)
Prof. Dr. Marko Lindner, Kristof Albrecht

Exercise Sheet 5

Deadline: 21.11.22, 12:00pm

Exercise 1 (Continuous Fourier Transform)**.**
In the lecture, we defined the Fourier transform $\mathcal{F}f = \hat{f}$ of a function $f \in L^1(\mathbb{R}^d)$ via the formula

$$\hat{f}(\omega) := (2\pi)^{-d/2} \cdot \int_{\mathbb{R}^d} f(x) \cdot e^{-i\cdot\langle x,\omega\rangle_2}\, dx \quad \text{for } \omega \in \mathbb{R}^d.$$

Recall that the inverse Fourier transform $\mathcal{F}^{-1}f = \check{f}$ of a function $f \in L^1(\mathbb{R}^d)$ is then given by

$$\check{f}(x) := (2\pi)^{-d/2} \cdot \int_{\mathbb{R}^d} f(\omega) \cdot e^{i\cdot\langle \omega,x\rangle_2}\, d\omega \quad \text{for } x \in \mathbb{R}^d.$$

a) Compute the Fourier transform of the one-dimensional Gaussian function

$$f(x) := e^{-x^2/2} \quad \text{for } x \in \mathbb{R}.$$

Hint: You can use that f is the unique solution to the differential equation

$$f'(x) = -x \cdot f(x) \quad \text{for all } x \in \mathbb{R}, \quad f(0) = 1.$$

b) Use part a) to compute the Fourier transform of the multivariate scaled Gaussian function

$$f_a(x) := (4\pi a)^{-d/2} \cdot e^{-\|x\|_2^2/4a} \quad \text{for } x \in \mathbb{R}^d$$

and fixed $a > 0$.

c) Show that the function family $(f_a)_{a\in\mathbb{R}_+}$ from part b) satisfies the property

$$f_a * f_b = f_{a+b} \qquad \text{for all } a, b \in \mathbb{R}_+.$$

Exercise 2 (Discrete Fourier Transform)**.**
Let $f = (f_1, ..., f_N) \in \mathbb{C}^N$ be a finite signal. In this exercise, we want to derive the discrete version of the Fourier Transform.

a) For $j \in \mathbb{Z}$, we set $\omega_N^{(j)} = 2\pi \cdot \frac{j}{N}$. Consider the space $\mathbb{C}^N$ with the standard inner product

$$\langle x, y\rangle = \sum_{k=1}^{N} x_k \cdot \overline{y_k} \qquad \text{for all } x, y \in \mathbb{C}^N.$$

Show that the vectors

$$b_j := \left(N^{-1/2} \cdot e^{i\cdot\omega_N^{(j-1)}\cdot(k-1)}\right)_{1\le k\le N} \in \mathbb{C}^N \qquad \text{for } j \in \{1, ..., N\}$$

form an orthonormal basis of $\left(\mathbb{C}^N, \langle\cdot,\cdot\rangle\right)$. Find coefficients $\hat{f} = \left(\hat{f}_1, ..., \hat{f}_N\right)$ such that

$$f = \sum_{j=1}^{N} \hat{f}_j \cdot b_j.$$

The mapping $F_N : \mathbb{C}^N \to \mathbb{C}^N,\ f \mapsto \hat{f}$ is called *Discrete Fourier Transform (DFT).*

b) Find a matrix $M_{F_N} \in \mathbb{C}^{N\times N}$ such that $\hat{f} = M_{F_N} \cdot f$. Invert this matrix to determine the *Inverse Discrete Fourier Transform (IDFT)* F_N^{-1}.

c) We can define $F(f)_j$ on the whole integer domain by

$$F(f)_j = N^{-1/2} \cdot \sum_{k=1}^{N} f_k \cdot e^{-i \cdot \omega_N^{(j-1)} \cdot (k-1)} \qquad \text{for all } j \in \mathbb{Z}.$$

Show that this is the periodic continuation of the finite DFT, i.e.

$$F(f)_{j+l\cdot N} = F(f)_j \qquad \text{for all } j \in \{1, ..., N\}, l \in \mathbb{Z}.$$

Remark: For $l = -1$, this implies

$$(F(f)_1, ..., F(f)_N) = \big(F(f)_1, ..., F(f)_{\lceil N/2 \rceil}, F(f)_{\lceil N/2 \rceil + 1 - N}, ..., F(f)_0\big)$$

The Octave / Matlab function `fftshift` performs a shift

$$\big(F(f)_1, ..., F(f)_{\lceil N/2 \rceil}, F(f)_{\lceil N/2 \rceil + 1 - N}, ..., F(f)_0\big) \rightsquigarrow \big(F(f)_{\lceil N/2 \rceil + 1 - N}, ..., F(f)_0, F(f)_1, ..., F(f)_{\lceil N/2 \rceil}\big).$$

Note that this ordering of the Fourier coefficients is better suited for certain applications like low-pass filters (see exercise 3), as the coefficients that correspond to high frequencies now lie on the outside of the array.

Exercise 3 (Low-Pass Filter)**.**
The DFT can be transferred to the two-dimensional case. For an image $u : \{1, ..., M\} \times \{1, ..., N\} \to \mathbb{R}$, the discrete Fourier transform $\hat{u}$ of u is given by

$$\hat{u}_{j,l} = (M \cdot N)^{-1/2} \cdot \sum_{k=1}^{M} \sum_{n=1}^{N} u_{k,n} \cdot e^{-i \cdot \omega_M^{(j-1)} \cdot (k-1)} \cdot e^{-i \cdot \omega_N^{(l-1)} \cdot (n-1)}$$

for $(j, l) \in \{1, ..., M\} \times \{1, ..., N\} = \Omega$. Its inverse is defined analogously to the one-dimensional setting. Using the two-dimensional DFT, we want to implement a simple *low-pass filter* for image denoising. Given the image u and an array $d = (d_1, d_2) \in \mathbb{N}^2$, the algorithm should perform the following steps, where you can use the respective built-in functions from Octave / Matlab:

1. Compute the Fourier transform $\hat{u}$.
2. Shift the matrix of Fourier coefficients via `fftshift`, see exercise 2c).
3. Let

$$S_d = [(M+1)/2 - d_1, (M+1)/2 + d_1] \times [(N+1)/2 - d_2, (N+1)/2 + d_2] \subset \mathbb{R}^2.$$

   Truncate the Fourier coefficients corresponding to the higher frequencies via

$$\hat{u}^{LP}_{j,l} = \hat{u}_{j,l} \cdot \chi_{S_d}(j, l) \qquad \text{for all } (j, l) \in \Omega.$$

4. Shift back the remaining Fourier coefficent via `ifftshift`.
5. Compute the inverse Fourier tranform of $\hat{u}^{LP}$ and take the real part of this array as the denoised version of u.

Implement this low-pass filter as a function

```
function [B] = LPFilter(A, d),
```

where A is an 8-bit gray scale image and $d = [d_1, d_2]$ is an array of length two containing positive integers. Make sure to convert the image A to double before computing its Fourier transform and to convert the end result back to uint8, such that B is an 8-bit grayscale image again.

($\star$) **Exercise 4** (Convolution and Fast Fourier Transform)**.**
In the lecture, we learned about the Convolution Theorem regarding the continuous Fourier Transform. We want to peove a similar result for the DFT in this exercise. Moreover, we will take a look at the recursion formula which leads to the *Fast Fourier Transform*, an algorithm that computes the discrete Fourier transform of an signal in a very efficient way.

a) Let $f, g \in \mathbb{C}^N$ be finite signals of same length. Prove the discrete convolution property

$$F_N\left(f \otimes g\right)_j = N^{1/2} \cdot F_N(f)_j \cdot F_N(g)_j \qquad \text{for } j \in \{1, ..., N\},$$

where $\otimes$ denotes the circular convolution from exercise sheet 4.

b) We assume that $N = 2M$ with $M \in \mathbb{N}$ and set

$$u := (f_{2k-1})_{1 \leq k \leq M} \in \mathbb{C}^M \qquad \text{and} \qquad v := (f_{2k})_{1 \leq k \leq M} \in \mathbb{C}^M.$$

Prove the recursion formula

$$F_N(f)_j = \begin{cases} 2^{-1/2} \cdot \left(F_M(u)_j + e^{-i \cdot \omega_N^{(j-1)}} \cdot F_M(v)_j\right) & \text{if } j \in \{1, ..., M\} \\ 2^{-1/2} \cdot \left(F_M(u)_{j-M} + e^{-i \cdot \omega_N^{(j-1)}} \cdot F_M(v)_{j-M}\right) & \text{if } j \in \{M+1, ..., N\}. \end{cases}$$

Remark: Part a) shows that we can compute the circular convolution of f, g by multiplying their respective Fourier transforms componentwise and then apply the IDFT. This method is more efficient than computing the circular convolution via the standard formula if there is an efficient way to compute discrete Foruier transforms.

Part b) shows that the Fourier transform of a signal with even length is decomposed of the two Fourier transforms $F_M(u)$ and $F_M(v)$, where the vecotr u contains the values belonging to odd indices and v contains the values belonging to even indices. Thus, the computation of the Fourier tranform is split into smaller subproblems. If $N = 2^m$, where $m \in \mathbb{N}$, we can recursively split our array m times until we arrive at arrays of length 1, whose discrete Fourier transform is trivial. This algorithm is called *Fast Fourier Transform*, which is an example of a so-called *divide and conquer alogorithm.*