Foundations of Graphics

Universität Bonn
Institut für Informatik II
November 26, 2019

Winter term 2019
Prof. Dr. Matthias Hullin

Sheet 7 - Fourier Analysis

Please upload your solutions to
`https://uni-bonn.sciebo.de/s/XfRkWhpWDEeR8Y7`
by **Thursday, 28.11.2019**.
Make sure to **list all group members** on all pages / source files.
Theoretical solutions must be texed or scanned, **photos will not be accepted**.

Theoretical Part

Assignment 1) Orthogonal functions *(3+1Pts)*

Two functions f and g are called *orthogonal* if their inner product $\langle f, g\rangle$ is zero (i.e., the definition is pretty much the same as for vectors). For functions of one variable that are defined on the interval $[a, b]$, a common definition of the inner product is

$$\langle f, g\rangle = \int_a^b f^*(x)g(x)dx, \tag{1}$$

where f^* denotes the complex conjugate; i.e., for real functions $f^* = f$.

Consider the following functions on the interval $[0, 2\pi]$:

$$f(x) = \operatorname{sgn}(\cos(x))$$
$$g(x) = \begin{cases} \frac{2}{\pi}x & \text{if } 0 < x \leq \frac{\pi}{2} \\ -\frac{2}{\pi}x + 2 & \text{if } \frac{\pi}{2} < x \leq \frac{3}{2}\pi \\ \frac{2}{\pi}x - 4 & \text{if } \frac{3}{2}\pi < x \leq 2\pi \end{cases}$$

Hint: The **sgn**(x) function returns 1 or -1 depending on the sign of x.

Show that these functions are orthogonal. In particular:

a) Compute the integral analytically and show that its value is 0.
b) The integral is a bit tedious to compute, but the functions are actually rather simple. Prove this by plotting both of them using `matplotlib` and explain in two sentences why one can see from the plots that the integral of the product must be 0.

Assignment 2) Rayleigh's identity *(0+4Pts)*

The Pythagorean theorem states that the squared length of a vector can be found by adding the squared components with respect to an orthonormal basis.

The Fourier transformation can be thought of as a transformation between infinite-dimensional vector spaces (the temporal and the frequency domain), so the same property holds (although in this case it is called Rayleigh's identity and looks a bit different):

$$\int_0^1 |f(t)|^2 \, \mathrm{d}t = \sum_{n=-\infty}^{\infty} \left|\hat{f}(n)\right|^2$$

where f is a 1-periodic function in the time domain and $\hat{f}$ are its complex Fourier coefficients computed by

$$\hat{f}(n) = \int_0^1 e^{-2\pi i n t} \cdot f(t) \, \mathrm{d}t.$$

a) Think about what you just read until you are confident that you understood it in detail. (**Hint:** Remember the analogy between the scalar product and integrals of multiplied functions as well as the interpretation of the scalar product projection one vector onto another).

b) Prove Rayleigh's identity.
Hints: The Fourier transformation of $f(t)$ can be written as:

$$f(t) = \sum_{n=-\infty}^{\infty} \hat{f}(n) \cdot e^{2\pi i n t} = \sum_{n=-\infty}^{\infty} \langle f, e_n \rangle \cdot e_n$$

Also remember:

- The algebraic properties of the complex inner product, and
- the fact that the $e_n(t) = e^{2\pi i n t}$ are orthonormal with respect to this inner product, and
- how to re-arrange and manipulate sums.

Practical Part

Assignment 3) 2-dimensional convolution in Python *(4+4Pts)*

The *convolution* of 1-dimensional functions f and g is defined as the integral

$$(f * g)(x) = \int_{-\infty}^{\infty} f(x') \cdot g(x - x') dx' \tag{2}$$

Similarly, the discrete convolution of vectors $\mathbf{f} = (f_1, ..., f_n)$ and $\mathbf{g} = (g_1, ..., g_m)$ can be defined as

$$(\mathbf{f} * \mathbf{g})_i = \sum_{j=1}^{m} f_{i-j} \cdot g_j \tag{3}$$

This can be interpreted as the sum of sliding copies of $\mathbf{f}$, weighted by the values of $\mathbf{g}$. Note the need for boundary conditions f_{i-j} may not be defined. A common choice are cyclic boundary conditions, which we obtain just by wrapping the indices to the range $1...n$ when evaluating Eq. 3. The above definition easily generalizes to higher-dimensional matrices by summing over multiple dimensions, for instance in the 2D case:

$$(\mathbf{F} * \mathbf{G})_{i,j} = \sum_{k=1}^{\dim(\mathbf{G},1)} \sum_{l=1}^{\dim(\mathbf{G},2)} F_{i-k,j-l} \cdot G_{i,j} \tag{4}$$

a) **Compute the convolution of two images**:

Convolution is often used to model blur operators. Using the PNG files provided in the framework, compute the 2-dimensional convolution of an image with a blur kernel.
The naive implementation would require a lot of loops and therefore runs very slowly when implemented in Python. Luckily, by now you have learned so much about Python that you can write an efficient implementation, e.g., by using proper vectorization. Do this in a way that the total run time does not exceed 10 seconds on a regular PC. Obviously, you are not allowed to use built-in convolution functions, but you can use all the basic numpy operations which readily vectorize. You don't need to get rid of *every* loop, as long as your code runs fast enough.
Attention: This exercise is about mastering Python and no points will be awarded for too slow solutions!

b) **Convolution in Fourier domain**:

According to the convolution theorem, a convolution in primal domain (spatial or temporal domain, depending on your application) corresponds to a multiplication in Fourier (frequency) domain:

$$h = f * g \quad \Leftrightarrow \quad \mathcal{F}(h) = \mathcal{F}(f) \cdot \mathcal{F}(g) \tag{5}$$

Using Python's built-in functions `fft2` (2D Fourier transform $\mathcal{F}$) and `ifft2` (inverse 2D Fourier transform $\mathcal{F}^{-1}$), perform the convolution of f and g through pointwise multiplication in the frequency domain.

Good luck!