Exercises Week #2: Sampling

Solutions should be submitted via moodle before April 27th, 2021 8.00 a.m. (CET)

Exercise 1 – Convolution theorem (Difficulty level: easy/medium)

Let $f(x), g(x)$ be two periodic signals and $\mathcal{F}$ denote the Fourier transform. Moreover, $\odot$ indicates an element-wise multiplication and $\otimes$ a convolution. We will use the reverse version of the convolution theorem

$$\mathcal{F}[f \odot g] = \mathcal{F}[f] \otimes \mathcal{F}[g] \tag{1}$$

to compute the Fourier coefficients of the signal

$$f(x) = \cos^2(kx) = \cos(kx) \odot \cos(kx) \tag{2}$$

where k is some integer.

Task 1A. (Difficulty level: medium) First compute the Fourier coefficients of $\cos^2(kx)$ directly by representing $\cos(kx)$ using the Euler formula[1] and squaring it. This will give you a closed-form expression for the Fourier coefficients c_n of $\cos^2(kx)$ for arbitrary k. Your calculation does not need to be submitted.

Hint: The signal $\cos^2(kx)$ has only three non-zero Fourier coefficients c_n: $c_{\pm 2k} = \frac{1}{4}, c_0 = \frac{1}{2}$.

Task 1B. (Difficulty level: very easy) In the following, we try to verify, numerically and graphically, the closed-form expression derived in 1A for the specific choice $k = 3$. To do so, create a NumPy array that stores the Fourier coefficients of $\cos^2(3x)$ assuming $N = 128$ sampling points (if you weren't able to calculate the coefficients yourself, take the expression from the hint for 1A).

Task 1C. (Difficulty level: easy) Sample $\cos^2(3x)$ using $N = 128$ sampling points in $[0, 2\pi)$ and compute its Fourier transform using NumPy's FFT. From the Fourier transform compute the *Fourier coefficients*.

Hint: Remember that the Fourier transform $\hat{f}_n$ is related to the Fourier coefficient c_n by $\hat{f}_n = Nc_n$.[2]

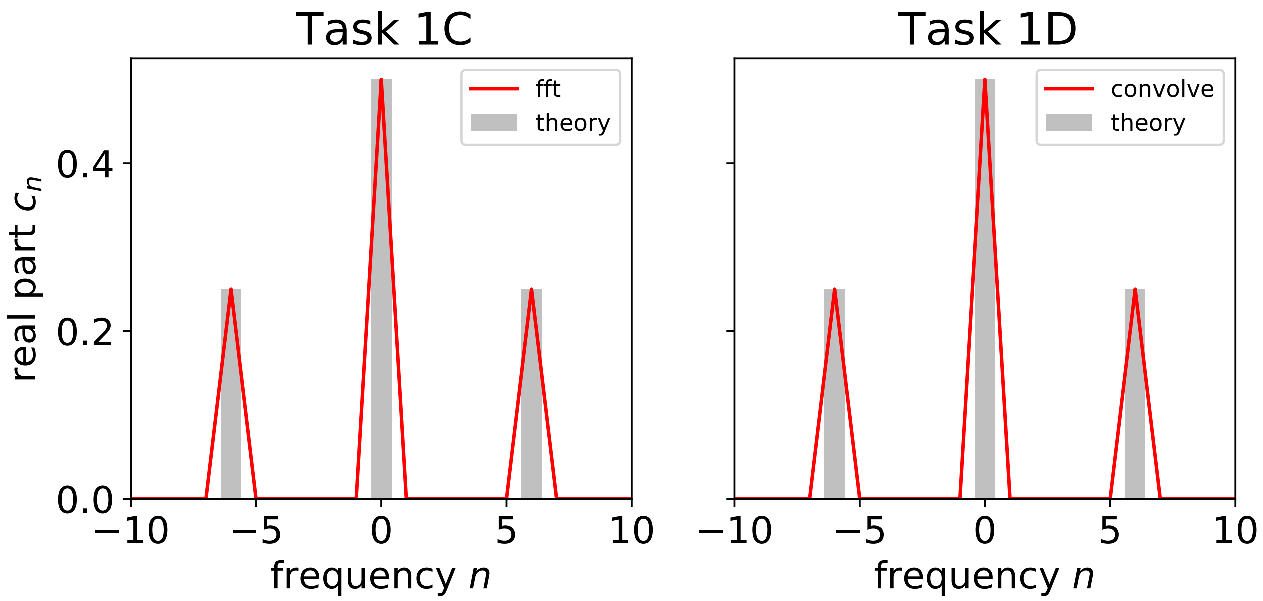
---

[1] See page 10 of the PDF version of the Jupyter notebook for Lecture 12 of the last semester.
[2] See page 13 of Lecture 12 from last semester.

Task 1D. (Difficulty level: medium) Now use theorem (1) to compute the Fourier transform of $\cos^2(3x)$. Again use $N = 128$ sampling points in $[0, 2\pi)$ to sample $\cos(3x)$ (**no square!**) and evaluate the Fourier transform of $\cos^2(3x)$ by convolving the Fourier transform of $\cos(3x)$ with itself.

Hint: The convolution is available in NumPy (`np.convolve`) and computes a *full* convolution.

Task 1E. (Difficulty level: easy) Plot the real part of the theoretical Fourier coefficients from 1B (shown as gray bars) and the values computed in 1C and 1D (red lines). Generate a plot similar to:



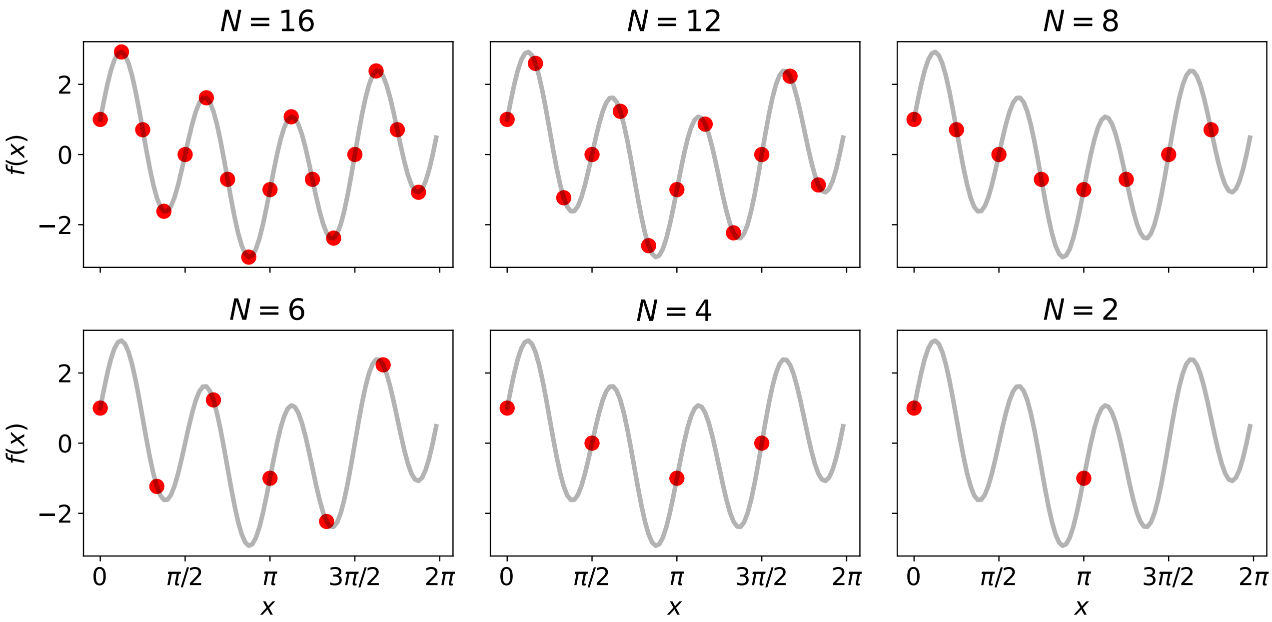
Exercise 2 – Sampling (Difficulty level: easy/medium)

This exercise studies the sampling of the signal

$$f(x) = \cos(x) + 2\sin(4x) \tag{3}$$

using a varying number of equi-distant points in the interval $[0, 2\pi)$.

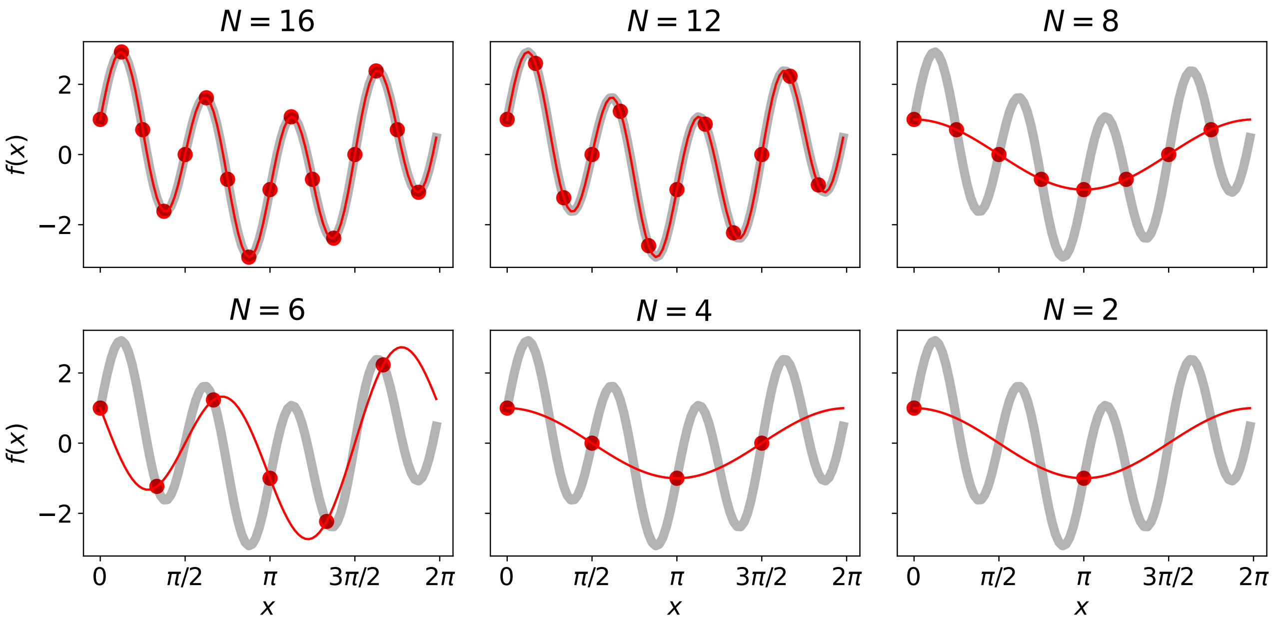
Task 2A. (Difficulty level: easy) Use $N = 96$ sampling points to represent signal (3) in a quasi-continuous fashion (we will call this the *fine* or *continuous signal* in the following). Downsample the fine signal using $N \in \{16, 12, 8, 6, 4, 2\}$ sampling points. For each choice of N, plot the continuous signal as a transparent black line and the samples using red dots as in the following figure:



Task 2B. (Difficulty level: medium) Now reconstruct the "continuous" signal (sampled with `N=96` points) from the downsampled signals. Use the Fourier transform to do the interpolation/resampling of the downsampled signals. Let `f` be the array storing the downsampled signal and proceed as follows:

1. Compute the Fourier transform of `f` and divide the amplitudes by the size of the array (remember that `fft` gives you the Fourier coefficients multiplied by the size of the array).
2. Shift the zero-frequency component to the middle of the array.
3. Pad the Fourier coefficients with zeros so as to obtain an array of the desired size `N`. Multiply the zero-padded array by `N` (in order to convert the Fourier coefficients to a Fourier transform of an array of size `N`).
4. Back-transform the zero-padded array (remember to undo the frequency shift before you back-transform).

Produce a figure similar to the one below where the original finely sampled signal is shown as thick transparent line in gray and the resampled signal is shown as a thin red line. What is the minimum number of samples that we need to obtain a correct reconstruction?



Hint: Implement the resampling procedure as a Python function `resample(f, N)` with two arguments: the downsampled signal `f` (a NumPy array) and the desired size of the resampled signal `N` (an integer larger than `len(f)`). The function should return an array of size `N` that stores the interpolated signal and can then be called for all downsampled signals studied in 2A. Code cell `[17]` in the PDF version of the Jupyter notebook accompanying the lecture about sampling implements a resampling method based on the FFT.

Exercise 3 – NumPy oneliners (Difficulty level: easy/medium)

Try to solve the following tasks in a single line of Python code using NumPy (the template provides more details). Each task gains one point, if you solve it in a single line. Every additional line reduces the number of points by 0.25.

Task 3A Let x, y be two vectors of equal size. Compute the inner product $\sum_i x_i y_i$.

Task 3B Let x, y be two vectors. Form the matrix z whose elements are $z_{ij} = x_i + y_j$.

Task 3C Let x, y be two matrices of equal shape. Compute the *matrix inner product* $\sum_{i,j} x_{ij} y_{ij}$.

Task 3D Let x be a vector, compute its FFT.

Task 3E Let x be a matrix/image, compute its FFT.

Task 3F Let x, y be two vectors, compute their *full* convolution.

Task 3G Let x, y be two vectors storing positions and frequencies, respectively. Form the DFT matrix with elements $\exp(-i\, x_i y_j)$ (note that i is the *imaginary* number, and that x_i uses an index i that has nothing to do with the imaginary number).

Task 3H Let x be an integer array. Count the number of elements in x that are greater than one.

Task 3I Let x be an integer array. Set all elements in x that are smaller than zero to zero.

Task 3J Let x be a double array. Set all elements in x that are larger than $-1.$ and smaller than $1.$ to zero.