Lab 5: Introduction to Signal Processing – Convolution, Sampling, and Reconstruction

This laboratory requires the following equipment:

- Matlab

The laboratory duration is approximately 3 hours. Although this laboratory is not graded, we encourage you to take your own personal notes as the lab verification test might leverage results acquired during this laboratory session. For any questions, please contact us at **sis-ta@groupes.epfl.ch**

1.1 Information

In this assignment, you will find several exercises and questions.

- The notation $\mathbf{Q_x}$ means that the question can be answered theoretically or with simple commands in the Linux operating system, without implementing or running any code.
- The notation $\mathbf{S_x}$ means that the question can be solved only by compiling and running a piece of code or an additional simulation.
- The notation $\mathbf{I_x}$ means that the problem has to be solved by implementing, possibly compiling, and running a piece of code.
- The notation $\mathbf{B_x}$ means that the question is optional (bonus) and should be answered if you have enough time at your disposal.

Outline

This lab is intended to continue the introduction to the topic of signal processing. Parts 1 and 2 introduce the operation of convolution. Parts 3 and 4 discuss sampling, the Nyquist-Shannon theorem, and reconstruction methods. You will work both with discrete and continuous signals.

Getting Started (Short reminder)

To start with this lab, you will need to download the material available on Moodle. Download `lab05.tar.gz` and extract it in your working directory (you can type `tar xvfz lab05.tar.gz`). Now start Matlab and change your "Current directory" to be `lab05/part01/`.

Part 1: Continuous Function Manipulation and Convolution

In this part the continuous signals are handled internally by Matlab as discrete signals, but you will operate with the provided scripts as if they were continuous signals. This might have some side effects given that Matlab will work with sampled versions of the continuous signals.

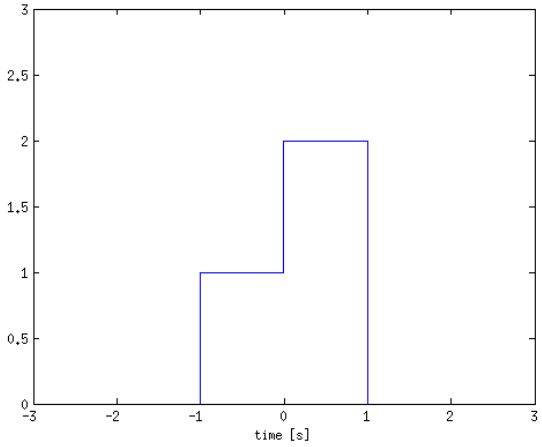

Figure 1: Plot of f(t)

Q1: In the picture above, a plot of function f(t) is shown. Find a mathematical expression for f(t) using only a combination of the rectangle shape functions that you have already used in Lab 4. The definition of the rectangle function is repeated below for your convenience:

$$\mathrm{rect}(t) = \Pi(t) = \begin{cases} 0 & \text{if } |t| > \frac{1}{2} \\ \frac{1}{2} & \text{if } |t| = \frac{1}{2} \\ 1 & \text{if } |t| < \frac{1}{2}. \end{cases}$$

I2: Using the function `rect2(a, b)`provided with the lab, generate function f(t) in Matlab. It is possible to add or subtract rectangle functions together. The function `rect2(a, b)` provides a rectangle function with its middle at `a` and of width `b`. Use `plot_function(f)` to plot the function. *Note*: *the definition of the* `rect` *function in this question is different from the definition used in* ***Q1***.

Q3: Convolution (French: “convolution”, German: “Faltung”) is an operation which is extensively used in signal processing, particularly in system analysis and filtering. Write an analytic expression for f(t) as a convolution of one rectangle function with a sum of Dirac delta functions.

S4: Do the convolution of **Q3** in Matlab using the function `h = convolution(f,g)` where f and g are the functions you want to convolve. The Dirac function is provided in Matlab with `ddirac(t)` where t is the position of the Dirac.

S5: Do the convolution of h with itself several times: `h = convolution(h,h),` and observe how the signal changes. Do you think the signal is converging to anything?

Part 2: Discrete Function Manipulation and Convolution

Since computers are not able to work with continuous signals (besides analytic equation solving), signal processing on computers is done in discrete time using discrete quantities. In this part, you will learn how to work with discrete time signals. First, let us start with discrete function manipulations. Below is the definition of the discrete functions w[n] and v[n] you already used in Lab 4:

$$w[n] = \begin{cases} 1 & if\ n = 0, 1 \\ 0 & otherwise \end{cases}$$

$$v[n] = \begin{cases} 1 & if\ n = -1, 0 \\ 0 & otherwise \end{cases}$$

S6: As you already learned in Lab 4, in Matlab, a discrete time signal is the same as a vector. Thus, the signal w[n] can be represented as `w = [0 0 0 1 1 0 0]` for `n=[-3:3].` Define signal v[n] in a similar way. Using `stem(n,w)`and `stem(n,v)` plot these signals to double check you defined them properly.

Q7: As you have seen in the course, the continuous time convolution is:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau$$

The discrete time convolution is obtained by replacing the integral operation by a summation (if you recall, an integral is defined by a sum with infinitely small pieces of the function). Thus, the discrete convolution is:

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$

On a piece of paper, compute `l[n]=(w*v)[n].` *Hint: consider the formula for* `n = [-2, 2].`

S8: Use the Matlab function `l = conv(w, v)` that does a discrete convolution between w and v. Does it show the same result? Try also the function with the following parameter: `l = conv(w, v, 'same').` Use the Matlab help to understand the differences.

Part 3: Signal Creation, Sampling and Reconstruction in Matlab

For some applications it is interesting to recover the signal between sampling periods (interpolation). In this part, the sampling of a continuous signal is simulated by down-sampling a high-resolution discrete signal. You will then try to reconstruct the signal in the non-sampled points. You will use the `SamplingReconstruction.m` file located in the `part03` directory of the `lab05` folder. There are five places in this file where parameters can be modified:

- The first location corresponds to the signal creation (lines 18-19 in the source code). The signal is built through a summation of multiple sine functions. You can define the parameters of the different sine functions: *f(n)* stands for the

frequency of the n^{th} sine and *a(n)* stands for its amplitude. The signal can be built with an unlimited number of sine functions.

- At the second location, *Fmax* stands for the maximal frequency displayed in the FFT plots (line 43).

- At the third location, you can specify if you want to add a filter (lines 65-70). For this lab, we will leave this parameter set to "none".

- At the fourth place (line 144), *fs* stands for the sampling frequency used for the sampling (line 134). You can specify an array of sampling frequencies if you want, for example: *fs=[5 10 15]*. In this case, you will have results for three resampling frequencies: 5, 10 and 15 Hz.

- At the last place you can select the interpolation function used in reconstructing the original signal from the collected samples (lines 158-159). It can be either *wsSignalReconstruction* or *linearSignalReconstruction* for Whittaker-Shannon or linear interpolation, respectively.

The Matlab script first generates the high resolution signal and computes the Fast Fourier Transform (FFT), an efficient algorithm for carrying out the Discrete Fourier Transform to observe the frequencies contained in the signal, which you saw in Lab 4. Then, it filters it if the parameters have been set (not used in this lab). It then samples the signal at frequency *fs* and reconstructs the signal using the selected interpolation formula.

I_9: Create a signal consisting of 3 sinusoids at 0.5, 2, and 4 Hz with respective amplitudes of 1, 0.4, and 0.2, and run the script. Look at the FFT of the original signal. What are the amplitudes of the FFT coefficients? Why is there a difference between the amplitudes of the FFT coefficients and the amplitudes of the original signal?

Q_{10}: Compare the FFT of the original signal with the FFT of the signal sampled at 10 Hz and the signal sampled at 20 Hz. Are there any differences? Why?

S_{11}: Specify three sampling frequencies (*fs*): 2, 4, and 20 Hz. Run the script and compare the FFT of the original signal with the FFT of the sampled signals. Are there any differences? Why?

I_{12}: Create a new signal made of two sine functions, with respective frequencies of 0.5 and 7 Hz, and amplitude of 1 for both. Sample the signal at a frequency *fs* of 10 Hz. Run the script and look at the FFT of the sampled signal. What frequency components from the original signal are present in the sampled signal? Which ones are lost? Are there any new frequency components in the sampled signal? Why?

I_{13}: Create a signal made of a single sine at 1 Hz with amplitude 1. Sample it at 3 Hz. Then, modify the code to use linear interpolation instead of Whittaker-Shannon interpolation. Compare the FFT of the original signal with the FFT of the reconstructed signal. Are there any differences? Why? (*Hint: recall the frequency components for the triangular signal we saw in the lecture*).

Part 4: Signal Reconstruction of a Sampled Signal

In the fourth part of this lab, we will use a signal sampled from a light sensor, inserted in an environment where several lighting sources were pulsating with a certain frequency. The value and time of the samples were stored in `lab05/part04/ls_values.txt`.

Q14: Take a look at the sample file. What is its format? Based on the information in this file, calculate the sampling frequency that the sensor is using.

S15: Open Matlab and run the `ReconstructionSignal.m` file. As provided, this script loads the sampled signal and performs a linear interpolation. Take a look at the FFT of the interpolated signal. How many frequency components do you observe? Do they all correspond to the original signal? (*Hint: recall the results from the linear interpolation in* **I13**).

I16: Modify the `ReconstructionSignal.m` script so that it performs Whittaker-Shannon interpolation and run it again. Compare the FFT of the interpolated signal with the one from the linear interpolation in the previous question. How many frequency components do you observe now? Which interpolation method best reconstructs the original signal?

S17: Reduce the sampling frequency of the signal by four times and repeat the previous reconstruction. Have the plots of the two sampled signals the same shape? How does this translate to the results obtained for the FFT and the reconstructed signal? What would be the maximum sampling period of the sensor that would still allow this particular signal to be correctly reconstructed?