

Lab 2: Introduction to Signal Processing –Sampling, Reconstruction, and Aliasing

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 what you see here might be useful for the project or the exam. For any questions, please contact us at sis-ta@groupepfl.ch

Information

In the following, you will find several exercises and questions.

- The notation **S** means that the question can be solved using only additional simulation.
- The notation **Q** means that the question can be answered theoretically, without any simulation.
- The notation **I** means that the problem has to be solved by implementing a piece of code and performing a simulation.
- The notation **B** means that the question is optional 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. You will be introduced to signal creation, sampling, aliasing, and reconstruction in this three-part exercise. 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 `lab02.zip` and extract it in your working directory. Now start Matlab and change your “Current directory” to be `lab02/part01/`.

Part 1: Signal Creation, Sampling and Reconstruction in Matlab

Note: You will need the signal processing toolbox. If it is not installed already, you can go to the Add-Ons menu  in the HOME tab in the top bar.

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 `part01` directory of the `lab02` folder. There are five places in this file where parameters can be modified:

- Lines 18-19 in the source code correspond to the frequencies and amplitudes contained in the signal. The signal is built through a summation of multiple sine functions. You can define the parameters of the different sine functions, where: $f(n)$ 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 (line 43), F_{max} stands for the maximal frequency displayed in the FFT plots .

- At the third location (lines 65-70), you can specify if you want to add a filter. For this lab, we will leave this parameter set to “none”.
- At the fourth place (line 134), f_s stands for the sampling frequency used for the sampling. You can specify an array of sampling frequencies if you want, for example: $f_s=[5\ 10\ 15]$. 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. Comment out the one you don't need.

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 1. Then, it filters it if the parameters have been set (not used in this lab). It then samples the signal at frequency f_s and reconstructs the signal using the selected interpolation formula.

1. **(I):** 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?
2. **(Q):** 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?
3. **(S):** Specify three sampling frequencies (f_s): 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?
4. **(I):** 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 f_s 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?
5. **(I):** 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 you saw in the lecture*).

Part 2: Aliasing

In this part, you will play with sampling frequencies to understand aliasing. Open the script `part02.m` in the folder `part02/`. You will study what happens when the maximal frequency of the signal is higher than half of the sampling frequency, therefore violating the Nyquist-Shannon theorem for an error-free reconstruction of the signal.

The code contains a single sine wave with a single frequency. You can change the frequency of the sine wave, F , as well as the sampling frequency, F_s . If you run the code, it plots the true FFT as well as the FFT of the sampled signal.

Note: Do not worry about the magnitude of the signal, it may have an unexpected value in some cases due to noise. In those cases, you can see the noise on the FFT plot.

6. **(S):** You can change the F and F_s in lines 6,7,8 (the units are Hz). Run the code for the default values and verify that the FFT is plotted as expected.

7. **(S):** Repeat by setting $F = 10$ and $F_s = 20, 18, 16, 10, 8,$ and 4 . What do you see in the FFT of the sampled signal?
Note: the frequency at which you see the peak is the alias frequency, and it is different from F .
8. **(S):** Try to play with a few frequencies and find a relation between the frequency of the signal, F , sampling frequency, F_s , and the alias frequency F_a . Recall what is meant by the aliased frequency.
9. **(S):** Recall from the lecture that aliasing can cause challenges during reconstruction because of overlapping of frequencies in the frequency domain. However, if the signal is bound, you can use aliasing to find where the actual frequencies lie.
Consider a situation where you want to find the frequency components of a signal. You know that the frequency components are in the range of 50-60Hz. You have a data acquisition system with a maximum sampling frequency, $F_{s_max} = 50\text{Hz}$. What sampling frequency will you use?
Hint: use your findings from the previous question.

Part 3: Application: Lighthouse Problem

A lighthouse is a tower that emits light to serve as a navigational aid for sailors and ship captains. Suppose that you are in charge of monitoring multiple lighthouses. To know if they are working well, you have a sensor station recording the light coming from the lighthouses. The obtained data comes in the shape of a single file with the time stamp light intensity measured by the sensor. You now want to automatize the procedure by using the Discrete Fourier Transform to determine which lighthouses are working.

To solve the following problem, you need to consider how lighthouses work. They contain a rotating light source emitting light so that seen from a distance, it appears to be a blinking light in a given direction. You can then use the FFT of the light signal to determine the frequency of the blinking and hence the speed of rotation.

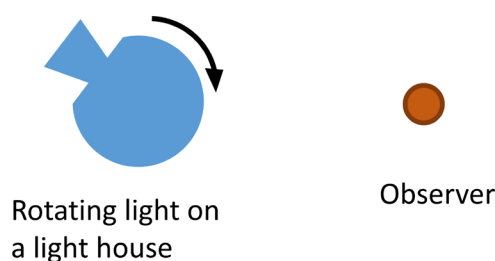


Figure: Illustration of a lighthouse

In the folder `lab02/part03a` you can find the Matlab function `analyzeLightSignal.m` and the following set of data corresponding to five different experiments: `lighthouse1.txt`, `lighthouse2.txt`, `lighthouse3.txt`, `signalA.txt` and `signalB.txt`. To analyze the data use `analyzeLightSignal('lighthouseX.txt')` where X is the corresponding number.

10. **(S):** Use the Matlab function `analyzeLightSignal` to analyze and visualize the data from `lighthouse1.txt`. Looking at the time domain signal, can you tell how many lighthouses are present in this experiment?
11. **(Q):** From the previous plot in the time domain we can see that the source of light is only one lighthouse. Can you estimate its rotation frequency just by looking at the temporal signal? Now look only at the frequency domain and check if you can answer this same question. Only the first 2 Hz are plotted.

12. **(S)**: Repeat Question 10 for files `lighthouse2.txt` and `lighthouse3.txt`. What are the angular speeds of `lighthouse2` and `lighthouse3`?
13. **(S)**: Now analyze and visualize `signalA.txt`. Can you determine how many lighthouses are present in this experiment? Can you say which lighthouses (`lighthouse1`, `lighthouse2` and/or `lighthouse3`) are present? Was it easier to get this information by looking at the time domain plot or by looking at the frequency domain? *Hint: use the marker tool to measure points in the plots.*
14. **(S)**: Repeat Question 13 for `signalB.txt`. *Hint: take into account that there might be harmonics.*

Moving forward, let us now look at a signal that is sampled and pre-recorded from a light sensor mounted on a boat. The boat is in an environment with several lighthouses, each with its light rotating with a certain frequency. The value and time of the samples were stored in `lab02/part03b/ls_values.txt`.

15. **(Q)**: 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.
16. **(S)**: Open Matlab and run the `ReconstructionSignal.m` file. As provided, this script loads the sampled signal and performs a zero-order hold interpolation. Look at the FFT of the interpolated signal. How many frequency components do you observe? Do they all correspond to the original signal?
17. **(I)** Change the reconstruction algorithm (set the method at line 29 to ‘linear’ instead of ‘zoh’) to use linear interpolation and implement the linear interpolation in the `linearSignalReconstruction` function at line 107 (implement your solution within the marked area). Compare the FFT of the interpolated signal with the one from the zero-order hold interpolation in the previous question. How many frequency components do you observe now? Which interpolation method best reconstructs the original signal?

Hint: you can leverage the following expression to implement the linear interpolation. It continuously interpolates the data points x_k and x_{k+1} at respective times t_k and t_{k+1} :

$$x_{lin}(\tau) = x_k + \frac{x_{k+1} - x_k}{t_{k+1} - t_k}(\tau - t_k), \quad \tau \in [t_k, t_{k+1}]$$

Before proceeding further, here is a brief review on the Whittaker-Shannon interpolation. When discretizing a signal, this discretization is equivalent to multiplying the signal with a train of Dirac impulses in the time domain with frequency $f_s = 1/T_s$. A multiplication in the time domain corresponds to a convolution in the frequency domain, and the chain of Dirac impulses in time domain remains a chain of Dirac impulses in the frequency domain. Therefore, the spectrum of the signal is repeated at each impulse in the frequency domain, spaced by the sampling frequency $\omega_s = 2\pi/T_s$. The best reconstruction is then given by applying an ideal low-pass filter to only preserve a single repetition of the signal spectrum with cutoff frequency $\omega_c = 2\pi f_c$. Since this filter is applied by multiplication in the frequency domain, the equivalent operation in time domain is a convolution. The time domain equivalent of the low-pass filter is a “sinc” function $\frac{\sin(\pi t)}{\pi t}$. Therefore, the reconstruction/interpolation of the signal in the time domain is done by summing a “sinc” function centered on each signal sample, scaled by the sample amplitude. See lecture slides, for a visual overview of this explanation (time domain interpretation of signal reconstruction).

18. **(I)** Now set the reconstruction method to use Whittaker-Shannon (set the method at line 29 to ‘ws’). Implement the Whittaker-Shannon interpolation and perform a similar comparison as in the previous question. We recall the WS reconstruction below:

$$x_{WS}(t) = \sum_{n=-\infty}^{\infty} x(nT_s) \operatorname{sinc}\left(\frac{t - nT_s}{T_s}\right)$$

19. **(S)**: Reduce the sampling frequency of the signal by four times (uncomment line 7) and repeat the previous reconstruction with the Whittaker-Shannon method. 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?