Lab 4: Introduction to Signal Processing: Fourier Transform

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 the following, text 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.

Getting Started (Short reminder)

To start with this lab, you will need to download the material available on Moodle. Download *lab04.tar.gz* in your personal directory. Now, extract the lab archive (you can type: `tar xvfz lab04.tar.gz`)

Part 0: Basic signal operations

Matlab is a very powerful tool when analyzing and processing discrete time signals. In this section, you will practice some basic operations like plotting and adding signals. Let's first consider this simple discrete-time signal

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

1. **(Q)**: What shape is it?

2. **(Q)**: Draw this signal on a piece of paper.

3. **(S)**: In Matlab, a discrete-time signal is the same as a vector. Thus, the above signal can be represented as `f=[0 0 1 1 0]` for `n=[-2:2]`. Using Matlab's function `plot(x,y)` and `stem(x,y)`, plot this signal. What difference can you see between those two functions? Which one would you choose to plot discrete time signals? Why?

4. **(Q)**: Now consider the discrete-time signal

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

   What is the relationship between `f[n]` and `g[n]`? Write `g[n]` as a function of `f[n]` (2 possibilities).

5. **(Q)**: What would be the resulting signal `h[n]=f[n]+g[n]`?

6. **(S)**: Implement this addition in Matlab. Plot the signal for `n=[-3:3]`.

Part 1: Fourier Transform

In class we introduced the concept of the *Fourier Transform:*

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(t)e^{-i2\pi\xi t}dt$$

and its inverse:

$$f(t) = \int_{-\infty}^{\infty} F(\xi)e^{i2\pi t\xi}d\xi$$

This part will be focused on the relation between time and frequency spaces. The scripts you will use can be found in folder `part1`. This part uses symbolic variables, allowing Matlab to compute analytical solutions. To generate a symbolic variable type `syms variable_name` in the command prompt. To get the Fourier Transform of a function use the Matlab function `F = fourier(f)`. To plot the magnitude of a Fourier-transformed function, use the provided function `plot_fourier(f)`.

You will start working with the continuous-time rectangle function or rectangular function or `rect` function defined as follows:

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

7. **(S)**: First generate the symbolic variable `x`. For that, you will need to type the command `sym x`. Symbolic variables allow Matlab to do the Fourier Transform analytically. Using the function `rect(x)` provided with the lab, generate a rectangle function centered in 0 and of width 1. Use `plot_function(f)` to plot the function. To be able to use the functions above, don't forget to set your "current folder" in Matlab to the `part1` folder.

8. **(S)**: Use both `fourier` function and `plot_fourier` function to plot the magnitude of the Fourier Transform of `rect(x)`. What function is it? *Hint: The magnitude is always positive.*

9. **(S)**: Do the same for a rectangle centered around 0.2. What difference do you notice in the Fourier Transform plot? *Hint: Solve this and the following questions without modifying rect.m.*

10. **(S)**: Do the same for a rectangle with width 2, centered in 0. What happens to the Fourier Transform? Try with other sizes.

11. **(S)**: Do the same for a rectangle of height 2 and width 1, centered on 0. What happened?

12. **(S)**: Plot the Fourier Transform of a sine using the Matlab function `sin(a*x)` with pulsation of `a = 6*pi rad/s`. Describe the obtained function.

13. **(S)**: Plot the Fourier Transform of the sum of two sines at pulsations of f_1 `= 6*pi rad/s` and f_2 `= 8*pi rad/s`. What is the result?

14. **(S)**: Plot the Fourier Transform of the product of two sines at pulsations f_1 `= 6*pi rad/s` and f_2 `= 8*pi rad/s`. How does the plot compare to the one in previous question?

15. **(S)**: Plot the Fourier Transform of `rect(x)*sin(5.5*pi*x)`. Do you recognize the obtained magnitude plot in previous questions?

Part 2: Fourier Series, Discrete Fourier Transform and FFT

In class, we introduced the concept of the *Fourier series*. Fourier states that any periodic function can be decomposed into a (possibly infinite) sum of sines and cosines.

For a signal `x(t)`, the complex *Fourier* coefficients C_n can be calculated using

$$C_k = \frac{1}{T}\int_0^T x(t)e^{-ik2\pi\xi t}dt$$

And the signal `x(t)` can be synthesized from the coefficients C_n using

$$x(t) = \sum_{-\infty}^{+\infty} C_k e^{ik2\pi\xi t}$$

The above equations can only be used for continuous functions (in time and amplitude). When processing digital signals in digital computers (and thus Matlab) we need the equivalent function for the discrete case (in time and amplitude). These are as follows.

For a signal `x[n]` of length `N`, its *Discrete Fourier Transform* `X[k]` can be calculated using:

$$X[k+1] = \sum_{n=0}^{N-1} x[n+1]e^{\frac{-i2\pi}{N}kn}, 0 \le k \le N-1$$

As analogy with the Fourier series, `X[k]` can be also labeled *Fourier coefficients*. Note that the index `[k+1]` instead of `[k]` as listed on the lecture slides is only due to the array indexing convention in Matlab (starting with 1 instead of 0).

Via an *Inverse Discrete Fourier Transform*, the signal `x[n]` of length `N` can be synthesized from the discrete frequency spectrum (or Fourier coefficients) `X[k]` using:

$$x[n+1] = \frac{1}{N}\sum_{n=0}^{N-1} X[k+1]e^{\frac{i2\pi}{N}kn}, 0 \le k \le N-1$$

16. **(S)**: Go to the folder `part2` and load the 3 signals `f2[n]`, `g2[n]` and `l[n]` using load `Lab04Signals.mat`. Plot `f2[n]`, `g2[n]`, using `stem()`.
17. **(S)**: The function `fourier_compute.m` implements the Fourier analysis. Compute the Fourier coefficients `X[k]` for `f2[n]` using `fourier_compute.m` and plot their magnitude using `stem()`. Do you see symmetry? *Hint: Remember that the Fourier coefficients are complex. You can plot the magnitude using the function* `abs()`.
18. **(S)**: Now synthesize the signal `f2[n]` by applying `fourier_revert()` to `X[k]` and plot it. Do you get the original signal `f2[n]` back? *Hint: a good way to quickly check how similar two signals are is to simply plot their difference.*
19. **(S)**: Now set all Fourier coefficients `X[k]` to 0 except for `k = 4,5,6,124,125,126.` Resynthesize `f2[n]` and plot it. Is the signal still the same as the one you got in 16(**S**)?
20. **(Q)**: You just set almost all Fourier coefficients `X[k]` to 0 yet the resynthesized signal still looks very much the same as the one resynthesized from only non-zero coefficients. Why?
21. **(S)**: Now also set the coefficients for `k=4` and `k=126` to 0, resynthesize and plot. Does the signal still look the same?
22. **(S)**: In the next step, in addition of `k=4` and `k=126`, set also the coefficients for `k=5` and `k=125` to 0, resynthesize and plot and observe how the signal changes.

23. **(Q)**: Using just 3 Fourier coefficients `k=4,5,6` (and their symmetric values `k=126,125,124`) of the original signal are apparently enough to synthesize the signal `f2[n]`. What does that tell us about the original signal `f2[n]`?

24. **(S)**: Determine the Fourier coefficients `X[k]` for `g2[n]`. Eliminate all `X[k]` for which `|X[k]|<3.` Resynthesize `g2[n]` and plot it. Is it the same as the original signal?

25. **(Q)**: Using just 3 Fourier coefficients `k=4,5,6` (and their symmetric values `k=126,125,124`) was enough to synthesize the signal `f2[n]`. Why do we need many more coefficients (all?) to properly resynthesize `g2[n]`?

The computation time for the `fourier_compute.m` increases quadratically with the number of values in the signal. A much more time-efficient way to compute the Fourier coefficients is the *Fast Fourier Transform* algorithm by Cooley and Tukey (originally by Gauss). Without going into the details of the algorithm, the following two problems just demonstrate the computational efficiency of the FFT over the straight implementation of the formula. The FFT is implemented in Matlab as `fft()` and its inverse is `ifft()`, which is the efficient implementation of `fourier_revert.m`.

26. **(S)**: `time_fourier_compute.m` computes the Fourier coefficients of `l[n]`, a random signal with 10000 values, using `fourier_compute.m` and `time_fft.m` does the same, but uses Matlab's built-in `fft()`. Both programs indicate the time it took them to complete. Execute both and compare the computation times.

27. **(S)**: Using `fft()`, revisit 18(**S**). Compute the Fourier coefficients for `f2[n]` using `fft()` and compare them to the ones computed using `fourier_compute.m`.

Part 3: Using Discrete Fourier Transform (implemented as FFT) to detect Lighthouses

A lighthouse is a tower which emits light to serve as a navigational aid for sailors and ship captains. Suppose that you are in charge of monitoring multiple lighthouses. For knowing 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 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 FFT of the light signal to determine the frequency of the blinking and hence the speed of rotation.

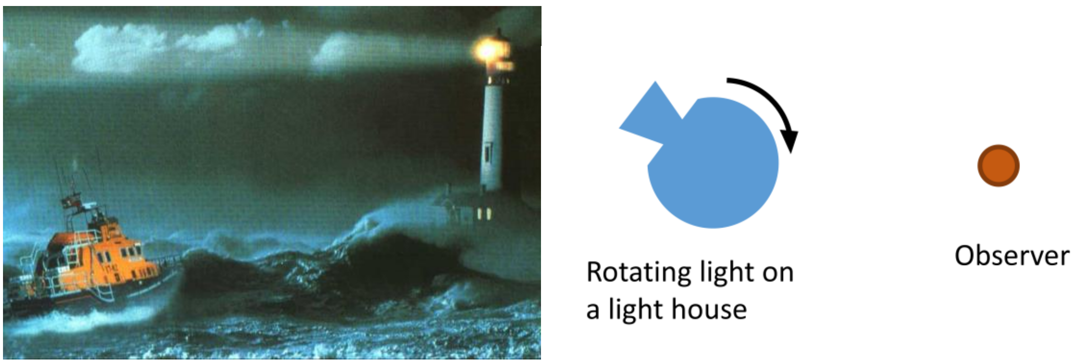

Illustration of a lighthouse

In the folder `part3` 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 date use `analyzeLightSignal(lighthouseX.txt)` where X is the corresponding number.

28. **(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?

29. **(Q)**: From the previous plot in 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.

30. **(S)**: Repeat question 28(**S**) for files `lighthouse2.txt` and `lighthouse3.txt`. What are the angular speeds of lighthouse2 and lighthouse3?

31. **(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 to the time domain plot or by looking at the frequency domain? *Hint: Use the marker tool to measure points in the plots.*

32. **(S)**: Repeat 31(**S**) for `signalB.txt`. *Hint: take into account that there might be harmonics.*