Lab 1: Introduction to Signal Processing: Fourier Series, Fourier Transform, and Convolution

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 exam 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 *lab01.tar.gz* or *lab01.zip* in your personal directory. Now, extract the lab archive (you can type: `tar xvfz lab01.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 `rect` function defined as follows:

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

7. **(S)**: First generate the symbolic variable `x`. For that, you will need to type the command `syms 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 kind of 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 `Cn` 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 `Cn` using

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

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 `Lab01Signals.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? Investigate how the function finds Fourier coefficents. *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 is 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: Continuous Convolution

In this part the continuous signals are handled internally by Matlab as discrete signals, but you will operate with the provided scripts that are inside folder `part3` as if they were continuous signals. This might have some side effects given that Matlab will work with sampled versions of the continuous signals.

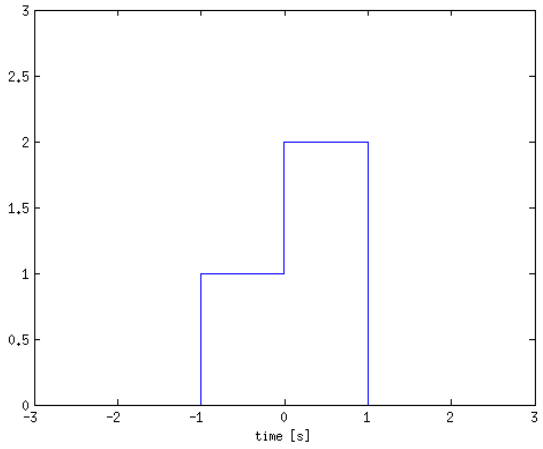

Figure 1: Plot of f(t)

28. **(Q)**: 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 Part 1.

29. **(I):** The function `rect2(a,b)` provides a rectangle function with its center at `a` and of width `b`. Use `plot_function(f)` to plot the function.

    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. *Note*: *the definition of the* `rect2` *function in this question is different from the definition used in Part1*.

30. **(Q)**: 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.

31. **(S)**: Do the convolution of question 30**(Q)** 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.

32. **(S):** Do the convolution of h with itself several times: `hconv = convolution(h,hconv),` and observe how the signal changes. Do you think the signal is converging to anything? Note that you need to convolve the original signal with the results of the convolution several times.

Part 4: Discrete 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 Part 0:

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

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

33. **(S)**: As you already learned in Part 0, 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.

34. **(Q):** 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].`

35. **(S)**: 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.

36. **(I):** Write the code for performing the convolution by yourself (i.e, without using the Matlab function `conv`). Plot and compare the result with the one you found in previous question.
Hint: You will need two for loops, one inside the other.