Linear Systems

SECOND YEAR GSC/GSA/GTT/GT

LAB SESSION 2 - ACADEMIC YEAR 2017/2018

MATLAB can be very useful for the analysis of signals and systems in the time and frequency domains. In this lab session, students will work with time representations of continuous and discrete-time signals, along with their corresponding Fourier transforms. Specifically, the goals of this lab session are:

- Understanding the representation of continuous-time signals using MATLAB
- Representation of Fourier transforms of several continuous and discrete time signals, paying special attention to the complex nature of such transforms
- Numerical evaluation of the Fourier transforms of continuous and discrete-time signals

1. Representation of continuous-time signals in MATLAB

One of the first difficulties we encounter when working with continuous-time signals is that MATLAB, like any other computer program, represents data using finite arrays of numbers, generally complex. In this way, the representation of a sequence such as

$$y[n] = cos\left(\pi n/3\right)$$

is straightforward, whereas that of continuous-time signals, such as

$$y(t) = cos\left(2\pi t\right)$$

is not so trivial, due to the continuous nature of the independent variable *t*. This problem is easily overcome if we the continuous-time signal is represented using a sufficiently high enough number of samples thereof. This procedure is called sampling, and will be studied in more detail in a future lesson during the course. By way of an example, run the following piece of code

```
n_points = 100;
t = linspace(-3,3,n_points);
y = cos(2*pi*t);plot(t,y);
xlabel('t'), ylabel('y(t)')
```

we can see that the array "y" provides enough information to represent the continuous-time signal in the interval [-3,3]. However, it must be clear that the above curve has been obtained joining the pairs of points $\{t^{(k)}, y^{(k)}\}, k = 1, \ldots, 100$ of the array "y". A similar approach can be used when representing Fourier transforms of continuous and discrete-time signals, $H(j\omega)$ and $H(e^{j\omega})$, respectively, since both are functions of a continuous independent variables. The following piece of code represents the Fourier transform of a rectangular pulse:

$$x(t) = \begin{cases} 1 & |t| \leq T_1 \\ 0 & |t| > T_1 \end{cases} \quad \overset{F.T.}{\longleftrightarrow} \quad X(j\omega) = 2\frac{\sin(\omega T_1)}{\omega}$$

```
n_points = 200;
T1 = 3;
w = linspace(-10,10,n_points);
Xw = 2*sin(w*T1)./w;
plot(w,Xw);
xlabel('w'), ylabel('X(jw)')
```

Exercise 1: Plot the Fourier transform of the signal $u(t)\,e^{-at}$, with *a* being a real positive constant, and $u(t)$ the Heaviside step function. In this case, the transform is complex, so you must represent the real and imaginary parts, as well as the modulus and the phase of the transform. Study the impact that parameter *a* has on the spectrum of the signal. Recall that the real and imaginary parts (or modulus and phase) may be plot in the same window with the *subplot* command, or in the same figure with the *hold on* command. Make use of the *xlabel, ylabel, title* and *legend* commands to property describe the plots.

Exercise 2: Plot the Fourier transform, $X_{pulse}(e^{j\omega})$, of the rectangular pulse sequence given by

$$x_{pulse}[n] = \begin{cases} 1 & |n| \leq N_1 \\ 0 & |n| > N_1 \end{cases}$$

where N_1 is a positive integer number. Recall that the Fourier transform of discrete signals is periodic, so it is enough to represent one period of the above transform. In addition, due to the fact that the sequence is real and even in time domain, $X_{pulse}(e^{j\omega})$ will be a real signal and thus, plotting its modulus is enough. Change the value of N_1 within the range [1,9], and study the qualitative impact of that parameter on the frequency representation of the signal.

2. Numeric computation of Fourier transforms of sequences

It is known that the Fourier transform of a sequence x*[n]* is determined by the following expression:

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n}$$

When *x[n]* is a sequence of finite duration, the number of non-zero terms in the sum will be finite too, and thus, it is easy to numerically evaluate the Fourier transform for any specific value of the frequency ω_k. Notice that this procedure does not require the analytic derivation of a closed form expression for the previous sum.

As an example, let $X_{pulse}(e^{j\omega})$ be the signal used in Exercise 2 with N_1=5. Then, the Fourier transform can be numerically evaluated:

```
%Representation of the signal in the time domain
N1 = 5;
n = -10:10;
x_pulse = zeros(size(n));
x_pulse(abs(n)<=N1)=1;subplot(2,1,1);
stem(n,x_pulse);xlabel( 'n' );
ylabel( 'x_{pulse}[n] ' ); title( 'Time Domain' )
% Representation of the signal in the frequency domain
n_points = 200;
w = linspace(-pi,pi,n_points);
for k = 1:length(w)
X_pulse(k) = sum(x_pulse.*exp(-j*n*w(k)));
end
subplot(2,1,2)
plot(w,real(X_pulse));
xlabel('w'); ylabel( 'X_{pulse}(e^{jw})' ); title( 'Frequency Domain' )
```

A careful analysis of the values of the *X_pulse* variable reveals that some of the array entries have non-zero imaginary parts, which are produced by the finite representation of numbers in the computer. With the aim of validating the previous procedure, represent in the same plot the Fourier transforms of the discrete pulse obtained both through numeric evaluation, and analytically (Exercise 2).

Exercise 3: Fourier transform of periodic sequences

- Analyze the behavior of the following function, which extends the *x_pulse* sequence by consecutively repeating the signal with a period of 21.

```
function [n_ext,x_ext] = repeat_pulse(N1,n_times)
n = -10:10;x = zeros(size(n));
x(abs(n)<=N1)= 1;
x_ext = x;
for k = 1:n_times
x_ext = [x x_ext x];
end
max_n = (length(x_ext)-1)/2;
n_ext = -max_n:max_n;
```

- Plot the Fourier transforms of the sequences that are obtained by periodically repeating the *x_pulse* signal, as the number of repetitions increases. What is the separation among the peaks of the Fourier transform? Can you explain this behavior?

3. Numeric computation of Fourier transforms of sequences

The evaluation of the integral in the analysis equation of the Fourier Transform can also be numerically evaluated for the case of time-limited signals, through an approximation based on a Riemann integral

$$X(j\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt = \lim_{\Delta t \to 0} \sum_{n=-\infty}^{\infty} x(n\Delta t)e^{-j\omega n\Delta t}\Delta t$$

Therefore, in order to approximate the value of the Fourier Transform it is sufficient to evaluate the sum on the right-hand side for a small enough value of Δt.

As an example, consider again the signal

$$x(t) = \begin{cases} 1 & |t| \leq T_1 \\ 0 & |t| > T_1 \end{cases}$$

Run the following piece of code and compare the result obtained for different values of *delta_t* with the exact Fourier transform, representing them together.

```
% Explore different values for delta_t
delta_t =????
t = -5:delta_t:5;
x = zeros(size(t));
x(abs(t)<=T1)=1;
w = linspace(-10,10,n_points);
for k = 1:length(w)
Xw_num(k) = sum(x.*exp(-j*w(k)*t)*delta_t);
end
% Xw_num contains the numerical evaluation of the Fourier analysis integral
```

4. Extension exercise: Expansion of a sequence by inserting zeros

- Record a voice file of around 5 seconds using the Windows recording tool, or any other similar application. Set up the recording in single channel mode (mono) and 16 kHz. Record the file in .wav format. Such format can be read in MATLAB using the *wavread* or *audioread* commands, depending on the MATLAB version.
- Store the recorded voice signal in a variable named *original_voice*. You can play the sequence in MATLAB with the command

```
sound(original_voice,16000);
```

- Obtain an expanded version of the original signal by inserting zeros

```
expanded_voice = zeros(1,2*length(original_voice));
expanded_voice (1:2:end) = original_voice;
```

- Numerically compute the Fourier transforms of the *original_voice* and *expanded_voice* sequences, and plot their modulus in a single figure. What do you observe?
- Finally, play the modified signal

```
sound(expanded_voice,16000);
```

- What difference do you observe between both signals in the time domain? Can you explain it in terms of their frequency content?