1 Basic signal processing for communications

The objective of this practice is to refresh the fundamental signal processing techniques that are used extensively in the analysis and design of communication systems. We will use the MATLAB environment to illustrate these techniques and their underlying concepts.

1.1 Sampling of analog signals

Every digital communication system must have an analog part, since the signal that propagates through the transmission medium (either a coaxial cable, a twisted pair, optical fiber or free space) is analog in nature. Nevertheless both the transmitter and the receiver have increasingly important digital parts and, therefore, it is necessary to convert the analog (continuous time) signals into digital (discrete time) sequences and vice versa.

The D/A conversion is carried out by *sampling*. Sampling an analog signal $x(t)$ can be represented as the multiplication of $x(t)$ by a train of impulses of amplitude 1 and period T_s. The resulting signal is

$$x_s(t) = x(t) \cdot \sum_{n=-\infty}^{\infty} \delta(t - nT_s) = \sum_{n=-\infty}^{\infty} x(nT_s)\delta(t - nT_s). \tag{1.1}$$

where T_s is the **sampling period** and its inverse, $f_s = 1/T_s$, is the **sampling rate.** In (1.1) we observe that impulses at time $t = nT_s$ are scaled by the value of $x(t)$ at those instants. The digital representation $x[n]$ is simply obtained by storing these values:

$$x[n] = x(nT_s) = x(t)|_{t=nT_s}. \tag{1.2}$$

Then, a fundamental question is whether there is any information loss when going from $x(t)$ to $x[n]$ (or from $x(t)$ to $x_s(t)$). In other words, is the sampling procedure reversible? In order to answer these questions, let us analyse the effect of the sampling operation (1.1) in the frequency domain.

Let $X(f)$ be the Fourier Transform (FT) of $x(t)$, so that

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi ft}dt, \qquad x(t) = \int_{-\infty}^{\infty} X(f)e^{j2\pi ft}df. \tag{1.3}$$

$X(f)$ provides a description of the frequencies present in $x(t)$, since $x(t)$ can be written as a sum (integral, to be precise) of complex exponentials of different frequencies, f, weighted by $X(f)$. The signal $x(t)$ is said to be **bandlimited** to B Hz if $X(f) = 0$ for $|f| > B$.

If we sample $x(t)$ at a rate $f_s = 1/T_s$ to get the signal $x_s(t)$ given by (1.1), then the corresponding spectrum (Fourier Transform of $x_s(t)$) $X_s(f)$ is

$$X_s(f) = f_s \sum_{k=-\infty}^{\infty} X(f - kf_s). \tag{1.4}$$

□ Question 1

- From expression (1.1), prove the relationship (1.4). In order to do so, keep in mind that the FT of an impulse train (spaced T_s seconds) is another impulse train (with certain separation between their individual impulses), and that a product in the time domain corresponds to a convolution in the frequency domain.

■

Thus, we see that for every integer k, $X_s(f)$ contains a copy (or replica) of $X(f)$ centered at kf_s. Then, if $x(t)$ is bandlimited to B Hz and it is sampled at a rate $f_s > 2B$, the different individual replicas will not overlap, and it will be possible to recover the replica centred at 0 Hz by means of an analog low-pass filter. If so, $x(t)$ can be recovered from $x_s(t)$. Therefore, we have proved the **sampling theorem** and the required minimum sampling rate, $2B$, also known as the **Nyquist rate**. This case is illustrated in the upper part of Figure 1.1.

However, if the sampling rate is below the Nyquist rate, then the replicas in $X_s(f)$ will overlap, and it will be impossible to recover the original spectrum $X(f)$ without distortion. This overlapping and its resulting distortion in the reconstructed signal is known as **aliasing**. In the presence of aliasing, the representation of the original analog signal is ambiguous. We must always be careful and try to avoid aliasing when sampling. Otherwise, the sampled signal will suffer an irreversible information loss. This scenario is illustrated at the bottom of Figure 1.1.

□ Question 2

- The human hearing system is capable of appreciating frequencies up to about 20 kHz. Which is the minimum sampling rate needed to digitize a musical concert? Compare your response to the rate of 44.1 kHz used in audio CDs.
- Some animals (e.g. dolphins and bats) produce sounds with a frequency content up to 50 kHz. What consequences does this fact have for a zoologist interested in recording the sounds of these animals on CD?

■

When using a computer tool such as MATLAB to simulate communication systems, it is impossible to capture all the subtleties of A/D conversion since the signals present in the simulation environment are already discrete. In other words, MATLAB cannot work directly with analog

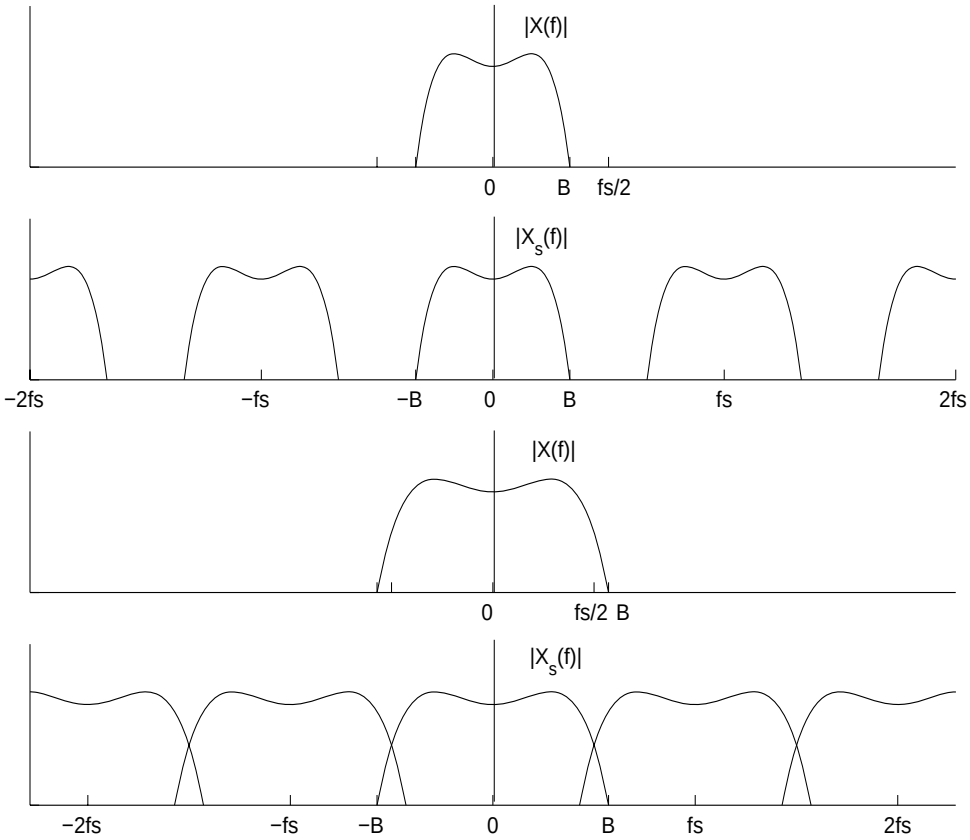

Figure 1.1: The spectrum of a sampled signal is periodic with period f_s. If the spectrum $X(f)$ is bandlimited to a frequency B less than $f_s/2$, the spectral replicas will not overlap (two upper plots). Otherwise, aliasing will take place (bottom plots).

signals. However, if we represent analog waveforms by sequences obtained with a high sampling rate, the sampling process can be simulated as a reduction of that sampling rate.

For instance, to work with a sinusoid of frequency 800 Hz, it is generally enough to sample it at its Nyquist rate, which is 1600 samples/s. But for plotting the waveform, it is often desirable a much higher sampling rate. The following MATLAB code calculates and plots the 800 Hz sinusoidal signal from $t = 0$ to $t = 0.03$ s with a sampling rate of $f_s = 80,000$ samples/s.

sine800hz.m

```
f=800;                          % frequency of the wave, in Hz
time=0.03;                      % total time in seconds
Tu=1/80000;                     % sampling interval in seconds
t=Tu:Tu:time;                   % define a time vector
w=sin(2*pi*f*t+0.3);            % define the sine wave
plot(t,w,'.');                  % plot the sinusoid vs time
xlabel('seconds');              % label the x axis
ylabel('amplitude');            % label the y axis
```

Run `sine800hz.m` and you will observe several periods of the sinusoid. Verify the duration of the period and determine the number of samples that it contains.

□ **Question 3**

- What should the sampling rate be, if each period of the sinusoid is to contain 60 samples? Check the result by modifying and then running the previous code.
- Reduce the sampling period to `Tu=1/20000` and observe the representation of the sinusoid. What does it look like? Do you think that aliasing is taking place?
- Repeat the previous point with `Tu=1/760`.

■

When $f_s \gg 2B$, the sampled representation *visually resembles* an analog signal, although actually it still is a discrete-time sequence. In this case the signal is said to be **oversampled**. Next, we will simulate the sampling process by **downsampling** the oversampled signal (reducing its sampling rate).

sine800hzsamp.m

```
f=800; time=0.007; Tu=1/80000; t=Tu:Tu:time; % freq and time vectors
w=sin(2*pi*f*t+0.3);                         % create sine wave w(t)
M=10;                                        % take 1 in M samples
wk=w(1:M:end);                               % the 'sampled' sequence
ws=zeros(size(w)); ws(1:M:end)=wk;           % sampled waveform ws(t)
subplot(211), plot(t,w);                     % plot the waveform
hold on; plot(t,ws, 'r'); hold off;          % plot 'sampled' wave
xlabel('seconds'); ylabel('amplitude');      % label the axes
subplot(212), plot(t(1:M:end),wk, 'ro');     % plot samples
xlabel('seconds'); ylabel('amplitude');      % label the axes
```

Run `sine800hzsamp.m` to get a graphical representation of downsampling the 'continuous' sinusoidal signal `w` by a factor `M=10`, which means that only one of out `M` samples is retained. In that MATLAB script,

- the vector `w` represents the analog signal $w(t)$;
- the vector `ws` represents the downsampled signal $w_s(t)$ given by

$$w_s(t) = w(t) \cdot \sum_{n=-\infty}^{\infty} \delta(t - nT_s) = \sum_{n=-\infty}^{\infty} w(nT_s)\delta(t - nT_s), \tag{1.5}$$

  where $T_s = M \cdot T_u$;
- the vector `wk` contains the samples $w(nT_s)$.

□ **Question 4**

- Consider the signal $x(t) = \mathrm{sinc}(t/T_0)$, with $T_0 = 5\times10^{-3}$ s, in the interval $-0.05 \leq t \leq 0.05$ s. Modify the code in `sine800hzsamp.m` to create an oversampled version of $x(t)$ (use $T_u = 10^{-4}$ s) and then downsample it by a factor $M = 10$.
- Repeat for $M = 30$ and $M = 50$, and comment on the results. (It is useful to determine the sampling period of the downsampled signal and to consider the spectrum of $x(t)$).

■

1.2 Spectral analysis in MATLAB

It is extremely common that a given signal lacks an analytical expression (e.g., when the signal is the result of a measurement), which prevents the computation of its spectrum by direct application of the Fourier transform (1.3). Nevertheless, even in this case, it is possible to *estimate* the signal spectrum using the **Discrete Fourier Transform (DFT)**.

The MATLAB command `fft` computes the DFT of a signal by means of an efficient algorithm called *Fast Fourier Transform*, whose details will not be discussed here. Of course, MATLAB can only work with discrete-time signals; this is not a problem, since, in practice, any measurement of an analog signal will provide a sequence of values (samples) of that signal.

A very useful MATLAB function (which internally employs `fft`) is `plotspec.m`. By running `plotspec(x,Ts)` we obtain a graphical representation of the signal whose samples are contained in `x`, together with the magnitude of its spectrum. The argument `Ts` corresponds to the sampling period, in seconds.

For a correct use of `plotspec.m` it is important to note that the number of FFT points that this function uses when calculating the spectrum is equal to the length of the signal. Therefore, if the number of avaliable samples is small, the spectral resolution will be low. In other words, the estimate of the spectrum obtained in this way will not be very accurate.

□ Question 5

- Generate samples of a sinusoid of amplitude 1 and frequency 20 Hz, sampled at $1/T_s = 10^3$ samples/s, between $t = 0$ and $t = 2$ s. Use `plotspec.m` to visualize its spectrum. Does it correspond to what you would expect?
- Vary the frequency of the sinusoid to 100, 200 and 600 Hz and comment on the results.
- For a frequency of 20 Hz, vary the sampling rate between 100 and 10^4 samples/s. How does the spectrum change in each case? What is the observed frequency range?

■

□ Question 6

- Write a short routine to generate samples of a square wave with amplitude between -1 and 1, with fundamental frequency 20 Hz and sampled at $1/T_s = 10^3$ samples/s, between $t = 0$ and $t = 2$ s (this can be very easily done by using MATLAB's `cos` and `sign` functions). Use `plotspec.m` to visualize its spectrum.
- Analytically, estimate the spectrum of a square wave by means of its Fourier series. Compare this spectrum with the estimate you just obtained in MATLAB.
- Do you think this signal can be regarded as bandlimited?

■

1.3 Design of digital filters

Linear time-invariant (LTI) filters are fundamental tools in any communication system. Their main objective is to shape the spectrum of their output signals. Although in future practices we will delve deeper into filter design, in this section we will review some important aspects.

As we already know, the operation of a (causal) digital filter can be expressed in the time domain by means of the following difference equation:

$$y[n] = \sum_{i=0}^{N} b_i x[n-i] - \sum_{j=1}^{M} a_i y[n-j], \tag{1.6}$$

where $x[n]$ and $y[n]$ are the input and output signals, respectively. Given the filter coefficients, the MATLAB function `filter` implements the filtering procedure. More concretely, the command `y=filter(b,a,x)` returns the output signal vector `y`. The vector `x` contains the samples of the input signal, while `b` and `a` contain the filter coefficients $\{b_0, b_1, \ldots, b_N\}$ and $\{1, a_1, a_2, \ldots, a_M\}$, respectively.

The effect of a filter can be easily visualized in the frequency domain: the filter attenuates or amplifies certain frequency components of the signal spectrum. The particular amount of amplification or attenuation, and the corresponding frequency regions, are determined by the *transfer function* of the filter. Therefore, even when the filtering operation is performed in the time domain by means of the difference equation (1.6), it is more natural to specify and design the filter in the frequency domain.

For instance, consider a case in which the signal of interest occupies a certain frequency band, whereas the noise or interfering signals reside in a different frequency band. In such case we would like to apply a filter to greatly attenuate the 'noisy' band while leaving the frequency components of the desired signal untouched. This intended filter response is directly translated to specifications of the transfer function of the filter.

Fortunately, MATLAB has several filter design routines that can be used in a straightforward manner. However, although these routines are quite flexible, they are not 100% accurate. Therefore, it is always advisable to verify that the resulting filter complies with the given specifications.

There are several types of filters: low-pass, high-pass, band-pass, band-stop, notch, etc. Their goal is to let certain frequency components pass without distortion (although some scaling and delay are acceptable), while frequency components outside the passband (i.e., in the *stopband*) are significantly attenuated.

- In the *passband,* the magnitude of the filter transfer function must be as flat as possible. It is also important that the phase is linear in this band, in order to have a constant delay for all of its frequency components. If this is not the case, each frequency component in the passband would suffer a different delay, thus distorting the output signal.

- In the *stopband,* the magnitude of the transfer function should be as small as possible, while the phase is irrelevant.

- The *transition band* comprises the frequencies between the passband and the stopband. Ideally, it should be as narrow as possible, but a practical filter can never have a null-width transition band. Moreover, there are tradeoffs among the width of the transition band, the flatness in the passband, and the attenuation in the stopband.

The function we will generally use to design FIR filters[1] in MATLAB is `firpm`. This function provides an impulse response, with real and symmetric coefficients, which results in the best approximation to the specified frequency response ('best' in the sense of maintaining a constant ripple in the passband). The command

```
b = firpm(order, edges, amps)
```

returns the filter impulse response in `b`. The input arguments are as follows:

[1] Recall that for FIR filters, the coefficients a_i in the difference equation (1.6) are all zero.

- `order` is the order of the filter (number of filter coefficients minus one). Note that by increasing the order we will improve the quality of the filter (it will be closer to meeting the specifications), but the resulting filter will require more resources (i.e., multiplications) in its implementation. In addition, the delay (in samples) introduced by this type of filters is half its order, so the higher the order, the greater the delay.
- `edges` is a vector specifying the band edges, in ascending order between 0 and 1, with 1 corresponding to half the sampling rate.
- `amps` is a vector specifying the required amplitudes at each band edge. Thus, it must have the same length as `edges`.

For instance, to design a 33-coefficient (thus order 32) band-pass filter that eliminates frequency components outside the interval $[f_s/8,\ 3f_s/8]$ we may use:

```
b = firpm(32, [0 .24 .26 .74 .76 1], [0 0 1 1 0 0]);
```

☐ **Question 7**

- Execute `plot([0 .24 .26 .74 .76 1],[0 0 1 1 0 0])` to visualize the 'mask', i.e., the specified frequency response of the pass-band filter, in the positive frequency range. If the sampling rate is 5,000 Hz, which frequency components will this filter let through?

■

To validate the design, we can use the function `freqz` or, even better, the tool `fvtool`.

```
freqz(b); fvtool(b);
```

☐ **Question 8**

- What is the minimum attenuation (in dB) introduced by the filter in the stopband?
- What is the peak-to-peak value (in dB) of the ripple in the passband?
- What is the delay introduced by this filter to a signal whose spectrum is confined to the passband?
- Make the transition bands narrower, e.g., by changing the band edges to `[0 .249 .251 .749 .751 1]`, and answer again the previous three questions.
- Repeat the original design, but using now an FIR filter with 129 coefficients. What can you say about the passband flatness, minimum stopband attenuation, and delay?

■

□ **Question 9**

- Assuming that our simulation works at a rate of 3.2×10^6 samples/s, specify and design a filter with two passbands, one in the interval [160, 240] kHz and the other in [400, 560] kHz.
- What is the delay, in seconds, introduced by your filter?

■

Important note: As you may have noticed, we have mentioned three different commands (`plotspec`, `freqz` and `fvtool`) for displaying data in the frequency domain. It is advisable to use `plotspec` to visualize the spectrum of *signals*, while `freqz` and `fvtool` are more suitable for displaying *filter* transfer functions.

1.4 Random variables

There are two basic commands in MATLAB for generating (pseudo)random numbers: `rand` and `randn`. The former generates numbers uniformly distributed between 0 and 1, whereas the latter generates samples from a standard normal (Gaussian) distribution, i.e. with zero mean and unit variance. From a set of samples of a random variable, we can estimate its probability density function by means of the histogram function (`histogram` in MATLAB).

□ **Question 10**

- Using `rand`, generate 10,000 samples uniformly distributed between -1 and 1. Verify the resulting distribution using the command `histogram`.
- Repeat the previous point for a normal distribution of mean 0 and variance 3, and then for mean 5 and variance 1. Change the number of histogram bins (10 by default) to 20 and 50.
- Repeat the process for a random variable that takes the values -1 and 1 with equal probability.

■

1.5 Stochastic processes

We can view a stochastic process, $X(t)$, as a 'collection' of random variables, one for each value of t, where each of these r.v.'s has its own probability density function $f_X(x;t)$.

Alternatively, recall that a random variable can be seen as a rule that assigns a number to every possible outcome of an 'experiment'; then, analogously, a random process can be seen as a rule that assigns a *time function* $x(t)$ to each outcome of an experiment. We sometimes refer to these time functions as the *realizations* of the stochastic process.

Recall the definition of the mean and autocorrelation of a random process:

$$\mu_X(t) = E\{X(t)\} = \int_{-\infty}^{\infty} x f_X(x;t)dx, \qquad R_X(t_1,t_2) = E\{X(t_1)X(t_2)\}, \tag{1.7}$$

which are *deterministic* functions of t and of (t_1, t_2) respectively.

A process is *stationary* if its statistical properties do not change over time. In particular, a random process is *wide sense stationary* if $\mu_X(t) = \mu_X$ (the mean of the process is the same at every time instant, i.e., it is constant with time) and $R_X(t_1,t_2) = R_X(t_1 - t_2)$, that is, its autocorrelation function only depends on the difference $t_2 - t_1$ (this means that the correlation, or similarity, between the random variables $X(t_1)$ and $X(t_2)$ only depends on their temporal distance, but not on the specific time instants).

□ Question 11

- Generate a realization `x = pa3(1,1000)` and calculate its temporal autocorrelation convolving $x(t)$ with $x(-t)$ and normalizing by the signal length. Since we are actually working in discrete time you will have to convolve $x[n]$ with $x[-n]$ and divide by the sequence length. Plot the result. Can you observe any symmetry? What maximum value does the autocorrelation take? Compare it with the signal power, estimated by `var`.

- Since we cannot work with realizations of infinite length, the estimated autocorrelation is just an approximation of the true function. To improve the quality of this estimation, we can average the autocorrelations of M realizations:

  ```
  x = pa3(M,N);
  for j=1:M, rx(j,:)  = conv ( x(j,:), fliplr(x(j,:))  )/N; end;
  r = mean(rx);
  ```

  Execute the previous code with $M = 50$ realizations of length $N = 500$. Assuming a sampling rate of 5 kHz, use `plotacf.m` (very similar to `plotspec.m`) to visualize the resulting autocorrelation and its Fourier Transform. What is the physical meaning of the latter? What could you say about the 'colour' of this random process?

- Now analyze the effect that filtering has on the autocorrelation of a random process. Considering the filter designed in Question 8, use the method described in the previous point to estimate the autocorrelation of the filter output, having as input the random process generated by `pa3.m`. Plot the result using `plotacf.m` and describe what you observe.

■