LINEAR SYSTEMS
SECOND YEAR GSC/GSA/GTT/GT

LAB SESSION3 - ACADEMIC YEAR 2016-17

Sampling is the base to many modern signal processing and communication systems and consists in obtaining a discrete sequence taking equally spaced samples from a continuous-time signal. The sampling theorem guarantees that if the continuous-time signal is sufficiently smooth and the samples are taken quickly enough, the continuous-time signal can be recovered in a unique and exact manner from the discrete sequence and its samples. We can take advantage of this fact to implement a continuous-time system with its discrete equivalent system provided that the input signal has a limited bandwidth. Specifically, the objectives that we are seeking in this lab session are:

- Understanding the concept of sampling and the limitations imposed by the sampling theorem so that we can fully recover the continuous-time signal out of its samples.
- Analysing the sampling process and the reconstruction of the signal, both in time domain as well as in frequency domain, paying special attention to the limitations introduced by the simulation tool Matlab.
- Observing what happens when the sampling theorem conditions are not fulfilled and we are faced with an aliasing problem.

1. Sampling of a sinusoidal signal

The sampling theorem guarantees that a continuous-time signal, $x_c(t)$, limited in bandwidth (that is, $X_c(j\omega) = 0$ for $|\omega| > \omega_M$)can be reconstructed in a unique and exact manner from its sequence of samples, $x_d[n] = x_c(nT_s)$, whenever $\omega_s > 2\omega_M$. Otherwise spectral overlapping ("aliasing") will occur and the recovered signal will be different from the original one.

In this section we try to corroborate these results using sinusoidal signals of the type

$$x_c(t) = \cos(\omega_0 t).$$

As we learned in the previous lab session, we cannot work with continuous-time signals in Matlab. Therefore, we will simulate them with discrete sequences with a significantly high number of samples (that is, using a sampling frequency that is much higher than the minimum required by the sampling theorem). Note also that, given the existent inverse time-frequency relation, a band-limited continuous-time signal requires that its duration be infinite (that is the case for the previous continuous signal $x_c(t)$). As it is, obviously, impossible to simulate an infinitely long signal in Matlab we will have to work with a signal of the type:

$$\tilde{x}_c(t) = x_c(t)p(t),$$

where $p(t)$ is a rectangular pulse whose duration will depend on the portion of the signal which we want to simulate. As a consequence, when working in frequency domain the obtained Fourier transform will be:

$$\tilde{X}_c(j\omega) = \frac{1}{2\pi} X_c(j\omega) * P(j\omega).$$

This fact needs to be taken into account when interpreting the obtained Fourier transforms, both in the continuous and in the discrete cases.

Once we have constructed the simulated continuous-time signal, you may sample it using the continuous-time to discrete-time converter (named `conv_cd`). The function input variables are: the continuous-time vector `xc`, which contains the samples of the continuous-time signal, vector `t`, which contains the time instants corresponding to `xc`, and the scalar `Ts` with the sampling period. The function output variables are the vectors: `xd`, which contains the newly sampled sequence and `tsamp`, that contains the time samples corresponding to `xd`. Note that variable `tsamp` refers to nTs.

```
function [xd, tsamp] = conv_cd(xc, t, Ts)

T = t(end) - t(1);
Ns = floor(T/Ts) + 1;
xd = [xc(1) zeros(1, Ns-1)];
tsamp = [t(1) zeros(1, Ns-1)];
for n = 1:Ns-1
   [delta, ind] = min(abs(t-t(1)-n*Ts));
    xd(n+1) = xc(ind(1));
    tsamp(n+1) = t(ind(1));
end
```

Next you are required to simulate and analyse the sampling process, both in time and frequency domain.

Exercise 1: Simulate a continuous-time signal with frequency $f_0 = 2$ Hz and duration from $t = -1$ to $t = 1$ seconds, and generate a vector of 1001 samples. Use function `conv_cd` to make the continuous-time to discrete-time conversion and use a sampling period of $T_s = 0{,}1$ seconds. Draw in the same figure the continuous-time signal in blue colour (use the command `plot`) and its samples in red colour (use the command `stem`). Check that the sampled signal is properly sampled, that is, that the amplitudes of the sampled signal coincide with those of the continuous-time signal.

Exercise 2: Compute the Fourier transform of the continuous-time signal, `Xc`, and of its discrete sequence, `Xd`, using the instructions given in the previous lab session. To do so generate the continuous signal $x_c(t)$, now with a duration from $t = -2$ to $t = 2$ seconds and with a high number of samples (e.g. with $\Delta t = 0{,}001$), and compute $X_c(j\omega)$ numerically between $\omega = -6\pi$ and $\omega = 6\pi$ using 1001 points. Next, sample the new signal $x_c(t)$ with $T_s = 0{,}1$ seconds, obtain $x_d[n] = x_c(nT_s)$, and evaluate $X_d(e^{j\omega})$ (using the functions and scripts developed in Section 2 of Lab Session 2) between $\omega = -2\pi$ and $\omega = 2\pi$ using 1001 points. Using the command `subplot` draw both transforms in the same figure, but in different sub-figures. Do both results match the expected theoretical results? What do you think the reasons for the differences are? Compute again the

Fourier transform of the discrete sequence using $T_s = 0{,}01$. Is the new transform closer to the theoretical result?

2. Reconstruction of a sinusoidal signal from its samples

The recovery of the simulated continuous-time signal from the discrete sequence of its samples is performed simply applying the interpolation formula given in theory:

$$x_r(t) = \sum_{n=-\infty}^{\infty} x_d[n] h_r(t - nT_s).$$

The Matlab code for implementing the previous interpolation formula is given by

```
for i=1:length(t)
    xr(i)=sum(xd.* reconstr_filter(t(i)-tsamp, Ts));
end
```

where `t` is the time vector of the continuous-time signal, and `reconstr_filter` is a function containing the implementation of the reconstruction filter which is used. Note that the duration of the reconstructed signal should be the same as the original continuous-time signal.

To obtain a perfect reconstruction it would be necessary to obtain the ideal filter, whose impulse response is

$$h_r(t) = \frac{T_s \sin(\omega_s t/2)}{\pi t}.$$

The values of this impulse response for a specific set of time instant values are easily computed using the following Matlab function, which takes into account the special case when $t = 0$, in which $h_r(t) = 1$.

```
function hi = ideal_filter(t, Ts)

hi = Ts*sin(pi*t/Ts)./(pi*t);
ind = find(t==0);
hi(ind) = 1;
```

Unfortunately, the ideal filter is unfeasible in practice due to its non-causal nature and the infinite duration of its impulse response. As a consequence, in practice, other simpler reconstruction filters are used. One of them is the zero-order filter (Sample and Hold interpolation), whose impulse response is

$$h_0(t) = \begin{cases} 1, & 0 \le t < T_s; \\ 0, & t \ge T_s \end{cases}$$

and whose Matlab implementation is as follows

```
function h0 = order0_filter(t, Ts)

h0 = zeros(size(t));
epsilon = 1e-10;
ind = find((t>=0) & (t<(Ts-epsilon)));
h0(ind) = 1;
```

Another alternative is the linear interpolation filter, whose impulse response is

$$h_1(t) = \begin{cases} (T_s - |t|)/T_s, & |t| < T_s; \\ 0, & |t| \geq T_s. \end{cases}$$

Its Matlab implementation is described below

```
function h1 = linear_filter(t, Ts)

h1 = zeros(size(t));
ind = find(abs(t)<Ts);
h1(ind) = (Ts-abs(t(ind)))/Ts;
```

Generally, in these cases the reconstructed signal is not going to be exactly the same as the original one. However, the reconstruction error will be small if the sampling frequency is sufficiently large in comparison with the maximum signal frequency.

In this section we try to analyse these three alternatives for the reconstruction of a continuous–time signal when it has been properly sampled.

Exercise 3: Use the ideal reconstruction filter to recover the continuous-time signal generated in the previous exercise (i.e. the sinusoidal signal with frequency $f_0 = 2$ Hz, duration from $t = -2$ to $t = 2$ seconds, $\Delta t = 0{,}001$ seconds) from the two sampled sequences (one with $T_s = 0{,}1$ and the other one with $T_s = 0{,}01$ seconds). Please represent both in the same figure but in different sub-figures for both cases: the continuous-time signal in blue (with the `plot` command), the sampled sequence in red (with the command `stem`) and the reconstructed signal in green (with the command `plot`). Analyse the reconstruction error for both cases (`zoom` in into the figures to better appreciate the error) and draw conclusions.

Exercise 4: Repeat the previous exercise using the zero-order reconstruction filter ('sample and hold') and the linear interpolator. Compare the obtained results with the ones obtained in the previous exercise and draw conclusions about the reasons that would make the usage of these filters appropriate.

3. Effects of an incorrect sampling: "aliasing"

When the continuous-time signal is not properly sampled (that is, the signal is sampled at a frequency that is lower than the minimum given by the sampling theorem), spectral overlapping will occur (otherwise called "aliasing") and the reconstructed signal will not coincide with the original signal. In this section we will explore the effects of "aliasing" when working with sinusoidal and voice signals.

Exercise 5: Generate a sinusoidal signal like the ones in the previous exercise (now with $f_0 = 5$ Hz, duration from $t = -2$ to $t = 2$ seconds, $\Delta t = 0{,}001$ seconds) and sample it with $T_s = 0{,}25$ seconds. Then recover the signal using the ideal reconstruction filter and draw in a single figure the original continuous signal, the sampled sequence and the reconstructed signal (as it was done in exercise 3). What is the shape of the recovered signal? What is its relation with the original signal? In order to be able to better appreciate the differences, it is advisable to compute also the Fourier transform of the original signal and the reconstructed signal, depicting both in the same figure.

The effects of incorrect sampling can be clearly appreciated when working with audible signals, as it is proposed in the next exercise.

Exercise 6: Generate three new sinusoidal signals with $f_0 = 2000$ Hz, duration from $t = 0$ to $t = 3$ seconds and the following sampling frequencies: $f_s = 5000$, $f_s = 3000$ and $f_s = 2500$ Hz. Reproduce alternatively the three signals with the command `sound(xc, fs)`. What is the effect you observe on the frequency of the signals? Could you compute the frequency of the reproduced tones?

4. Extension exercise

Record a voice signal of at most two seconds (following the process described in the extension exercise of the previous lab session) using a recording frequency of $f_g = 16000$ Hz, and check that you can reproduce the signal correctly. Next, sample the signal using a sampling frequency progressively smaller and smaller (e.g. $f_s = 8000$, $f_s = 4000$, $f_s = 2000$, etc.) and reconstruct the signal using the ideal filter. In which value of f_s can you appreciate a notorious degradation of the voice signal? What conclusions can you draw regarding its bandwidth?