BILKENT UNIVERSITY ELECTRICAL AND ELECTRONICS ENGINEERING DEPARTMENT

2019 Summer School
EEE - 321: Signals and Systems
Lab 5 - Off-Lab Assignment
Lab Day/Time/Room: 25.07.2019 - 17:40-19:30 - EE 211
Report Submission Deadline: 26.07.2017 - 17:20

Please carefully study this off-lab assignment before coming to the laboratory. There will be a quiz at the beginning of the lab session; this may contain conceptual, analytical, and Matlab-based questions about the lab. Some of the exercises will be performed by hand and others by using Matlab. The report should be prepared as hardcopy and written by hand (except the Matlab based plots and the code segments). Include all the plots, codes, calculations, explanations etc. to your report in a readable format.

Part 1: Basic operations

Consider the following piecewise function:

$$g(t) = \begin{cases} \frac{3}{2}t - 1 & 0 \le t < 1 \\ \frac{1}{2} & 1 \le t < 2 \\ \frac{3-t}{2} & 2 \le t \le 4 \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Sketch $g(t)$ by hand and indicate the critical points. Is $g(t)$ a continuous function? Justify your answer.

Write $g(t)$ as a composite of singularity functions (i.e. ramp function, unit step function and impulse function).

Suppose that the signal $g(t)$ is sampled at rate $F_s = 1/T_s$. Based on Shannon-Nyquist theorem, is it possible to recover the original signal from the sampled one? If so, find F_s. Either case, justify your answer based on concrete mathematical calculations and graphical illustrations.

Part 2: Sampling and Interpolation Basics

Let $x(t)$ be a continuous signal. The sampling operation can be formulated as a multiplication by impulse train. Therefore, the sampled signal in the continuous domain with the sampling period Ts:

$$\tilde{x}(t) = x(t) \sum_{n=-\infty}^{\infty} \delta(t - nT_s) = \sum_{n=-\infty}^{\infty} x(nT_s)\delta(t - nT_s)$$

Since impulse train is always zero except for $t = nT_s$ for $n \in \mathbb{Z}$, $\tilde{x}(t)$ is always zero for $t \neq nT_s$. Therefore, the discrete representation of x for $n \in \mathbb{Z}$ can be defined as follows:

$$x[n] \equiv x(t)|_{t=nTs}$$

So, the sampling operation can be seen as a system, where the input is $x(t)$ and the output is $\tilde{x}(t)$. Is the sampling operation linear, time-invariant? Justify your answer mathematically in your report.

What is the relation between the Fourier transforms of $x(t)$ and $\tilde{x}(t)$? Indicate your answer in your report with a mathematical justification and graphical illustration.

An important problem of signal processing is the reconstruction of the original continuous signal $x(t)$ from its sampled version $x[n]$, which is also named discrete-to-continuous conversion, digital-to-analog conversion or interpolation. Let us denote the reconstructed signal with $x_R(t)$. A common practice is to form $x_R(t)$ by convolving $x(t)$ by an interpolating pulse $h(t)$. The reconstructed signal $x_R(t)$ can be written as follows:

$$x_R(t) = \sum_{n'=-\infty}^{\infty} x[n']h(t - n'T_s) \tag{2}$$

Ideally, we would want to have $x_R(t) = x(t)$ for all t, but only in certain cases this is exactly possible. Usually, with our selections for $h(t)$ and Ts, we try to minimize the difference between $x_R(t)$ and $x(t)$.

Also note that, above Equation does not guarantee that $x_R(nTs) = x(nTs)$, for all n. However, in many applications, even if the perfect reconstruction is not achieved, it is desired that $x_R(nTs) = x(nTs)$. By doing so, for all n we can obtain the original samples $\hat{x}[n]$ when we sample $x_R(t)$ with T_s once again. If $x_R(nTs) = x(nTs)$ for all n, then, the interpolation is said to be **consistent**. Show that we have $x_R(nTs) = \hat{x}[n]$ for all n if $h(0) = 1$ and $h(kTs) = 0$ for all nonzero integers k. Include your work to your report.

Common choices for the function $h(t)$ are:
$h_Z(t) = rect(t/T_s)$ where

$$rect(t) = \begin{cases} 1 & -1/2 \le t \le 1/2 \\ 0 & \text{otherwise.} \end{cases}$$

in which case the reconstruction process is called zero order hold interpolation, Another choice is $h_L(t) = tri(\frac{t}{2T_s})$, where

$$tri(t) = \begin{cases} 1 - \frac{|t|}{0.5} & -1/2 \le t \le 1/2 \\ 0 & \text{otherwise.} \end{cases}$$

in which case the reconstruction process is called linear interpolation.
Another choice is $h_I(t) = sinc(\frac{t}{T_s})$, where

$$sinc(t) = \begin{cases} 1 & t = 0 \\ \frac{sin(\pi t)}{\pi t} & \text{otherwise.} \end{cases}$$

in which case the reconstruction process is called ideal band-limited interpolation.

The reconstruction success of these interpolators can be different depending on the properties of the original signal that is desired to be reconstructed. In the following parts you will investigate these. Now first answer the following questions:

- What are the values of $h_Z(t)$, $h_L(t)$ and $h_I(t)$ at $t = 0$.
- What are the values of $h_Z(t)$, $h_L(t)$ and $h_I(t)$ at $t = kT_s$, where k is a nonzero integer?
- Based on your answers to items a and b, are the interpolations performed using $h_Z(t)$, $h_L(t)$ and $h_I(t)$ consistent?

Part 3: Generation of Interpolating Functions in Matlab

In this part you will write a Matlab function which generates one of the three interpolating functions given in the previous part. Your function will look like **h = generateInterp(type,Ts,dur)**, where

- **type** determines the type of the function. It can be either 0, 1 or 2. Here 0 indicates zero order interpolation function, 1 indicates linear interpolation function and 2 indicates ideal interpolation function
- **Ts** determines the sampling rate
- **dur** determines the time duration of the signal. So, the time interval of the generated pulse is $-dur/2 \leq t < dur/2$. For example, if $dur = 2$, then the interval of the interpolating pulse will be between -1 and 1.

Important: Note that in the previous part, we defined the interpolating functions as continuous functions. However, here you have to generate a discrete version of them with the aim of doing a simulation of the interpolation process. Therefore, the discrete interpolating function that you generate here should be sampled much denser than the signal to be interpolated so that it behaves like a continuous function. For example, $Ts/1000$ can be chosen for the sampling period of the interpolating function.

Note: You are NOT allowed to use built-in commands of Matlab to generate the pulses. Include your Matlab code to your report.

Now let dur equals to the last digit of your ID number (take it 7 if it is 0) and Ts equals to $dur/6$. Using generateInterp function that you wrote, compute all three pulses $h_Z(t)$, $h_L(t)$ and $h_I(t)$ and plot them versus t. Include your plots to your report.

Now let dur equals to the second last digit (one before the last digit) of your ID number (take it 4 if it is 0) and Ts equals to $dur/3$. Using generateInterp function that you wrote, compute all three pulses $h_Z(t)$, $h_L(t)$ and $h_I(t)$ and plot them versus t. Include your plots to your report.

Part 4: Matlab Implementation of Interpolation

In this part, you are going to write a program in order to simulate the interpolation process. In your program, the continuous like (now you know why the word "like" is added here) signal $x(t)$ will be interpolated from its samples $x[n] = x(nT_s)$ according to Equation 2. Note that for a general $x(t)$, the formula in Equation 2 is impossible to be exactly implemented because it contains an infinite number of samples. Usually, it is assumed that $x(t)$ has a finite duration so that it produces a finite number of nonzero samples when sampled (say N). Under this assumption, we have:

$$x_R(t) = \sum_{n'=0}^{N-1} x[n']h(t - n'T_s)$$

Your program should look like:
function xR=DtoA(type,Ts,dur,Xn)
where

- **type** denotes the type of the interpolation. It can be either 0, 1 or 2. Here 0 indicates zero order interpolation, 1 indicates linear interpolation and 2 indicates ideal interpolation is going to be performed.
- **Ts** and **dur** have the same meaning as in the previous part.
- **Xn** (of size 1 x N) contains the samples of x(t) which are assumed to be taken at 0, Ts, 2Ts,..., (N - 1) Ts, so that Xn(1)=x(0), Xn(2)=x(Ts), ..., Xn(N) = x((N - 1)Ts).
- **xR** (of size 1 x M) denotes the reconstructed signal.

While writing this function, make use of the function **generateInterp**. Within your code, you need to generate the time variable t as well. You can generate t according to the explanation made in the previous part. Also, in your code, do not use any for loop over the t array and write your function as efficient as possible. Include your code to your report.

Part 5: Matlab Simulation for Interpolation

In this part, you will compare the efficiency of the three interpolating methods on the reconstruction of the signal g(t), which is given in Part 1.

First generate sampled version of g (t) for $0 \le t \le 4$ with Ts equals to 1/(10a) seconds, where a is a random integer that you will generate using Matlab and it should be between 2 and 6. Include both your code and stem plot of $g(nT_s)$ to your report. Now, using **DtoA** function, generate $g_R(t)$ for each interpolating method separately. Include the plots of the reconstructed signals. Discuss and compare the success of the interpolating methods. Now, increase Ts gradually, generate $g(nT_s)$ and their reconstructed versions for each Ts. Examine the reconstructed signals. Does the reconstruction becomes more successful while increasing Ts. Do the same while decreasing Ts. Discuss your observations. (You do not need to put any code or plot for this question)

Part 6: Reconstruction of a Sum of Cosines

Let D denote your ID number, and let D_4 denote your ID number in modulo 4. That is

$$D \equiv D_4 \mod 4$$

Let

$$x(t) = \begin{cases} 0.7cos(3\pi t + 2\pi/5) + 0.4sin(2\pi t + 1/3) + 0.5cos(5\pi t - 0.8e) & -2 \le t \le 2 \\ 0 & otherwise \end{cases}$$

- Let $T_s = 0.003(D_4 + 2)$. Compute continuous like function x(t) and display it versus t over the interval [-2, 2] using the plot command. On the same figure, using the stem command, display the sampled function x(nTs) with a different color. Include the plot to your report. Next, using the samples Xn, compute xR(t) for all three interpolators, using the function you developed in Part 4, and then plot the reconstructed signals. Include these plots to your report as well. Examine the results. Which interpolator seems to be the most successful one? In particular, can you distinguish the reconstruction of the ideal interpolator from the original signal? If there is a great difference, why? Include your comments to your report.

- Repeat a for $T_s = 0.3 + 0.02D_4$.

- Repeat a for $T_s = 0.099$.

Run your codes with several other Ts values between $0.01 < T_s < 0.2$ and examine the results (you do not need to provide any plots). In particular, what do you recognize about the performance of the ideal bandlimited interpolator? Can you notice any difference between the original and the reconstructed signal as long as $0.01 < T_s < 0.1$. What happens afterwards ($0.1 \le T_s \le 0.2$)? Why does this happen? Include your comments to your report.