11 October 2022

Lab 6 – CTFT, Sampling and Quantization

Objectives: In this lab, we will

- numerically compute Fourier transform (FT) of some common continuous-time signals we have seen in class and plot them, verify some properties of FT;
- process periodic signals (using FS coefficients) with LTI systems acting as filters (given their frequency response), plot and compare input & output signals.
- use discrete-time samples of continuous-time signals and perform empirical reconstruction using various interpolation methods.
- study audio signals, their *bit rates* and *sampling frequency*
- use Matlab to quantize *continuous-valued* signals, compute SQNR • study effect of quantization on the quality of reconstructed signal.

6.1 Continuous-time Fourier transform

(a) Write a matlab function `X = continuousFT(t,xt,a,b,ω)` to numerically compute continuous-time FT of the given signal $x(t)$ which has finite support in [a, b] and is zero outside. The inputs to this function are

- t – symbolic variable
- xt – signal whose FT is to be computed (function of symbolic variable t)
- a,b – the signal is equal to xt in the interval [a, b] and zero outside
- ω – the vector ω contains the values of frequency where FT is to be computed.

>> The function should return a vector X which contains the FT of $x(t)$ for each of the frequencies in the input vector ω.

(b) Write a matlab script that calls the function `continuousFT` for a rectangular pulse of unit amplitude in [-T, T] where T = 2 and ω = -5:0.1:5. In a single figure, using `subplot()` commands to get a 2x2 grid of subplots, plot the real part, imaginary part, absolute value and phase of the computed FT as function of ω.

>> For phase use the command `angle()` in matlab. Can you explain each of the observed subplots?

(c) Repeat part (b) for T = 1 and T = 4. Use ω = -5:0.1:5. What FT property supports your observations when T is changed?

(d) Repeat part (b) for $x(t) = e^{jt}$, and $x(t) = \cos(t)$. Limit signals to the interval [-T, T] where T = π and ω = -5:0.1:5. What is the expected FT? What are the shapes you are observing?

(e) Repeat part (b) for a triangle pulse of height 1 and base/support [-1,1]. How would you express xt for this case?
What is the expected FT? Hint: express the triangle pulse as convolution of two signals.

(f) Optional: play with some more signals $x(t)$ to test your function and verify whether standard properties of FT are satisfied as expected.

6.2 A signal and its samples

Consider the continuous-time signal $x(t) = \cos(5\pi t) + \sin(10\pi t)$ for analysis. Since continuous-time signals cannot be exactly represented in Matlab, we will use a very fine time-grid to approximate continuous nature of time. Let `t_fine = 0:0.001:2` be the time-grid for representing continuous time-signals (note that in this session `t_fine` is a proxy for continuous-time). Write matlab script for following:

>> Plot this signal as a function of time using the `plot()` command. The time axis should be from 0 to 2s. In this lab we will restrict to the time interval [0, 2] and the time vector `t_fine` will be used in all the tasks below. You should use `plot(t,x)` instead of just `plot(x)` to get appropriate markings on the time axis, else matlab will default to positive integer markings (vector index) which is not informative. Same applies for the `stem()` plots below.

>> Let this signal be sampled with sampling interval Ts = 0.1s and denote the discrete-time signal as $x[n] = x(nT_s)$. In the same figure above, plot the samples $x[n]$ using the `stem()` command in the time interval [0, 2]. Plotting would be easier if you generate the time vector corresponding to the location of the samples: `t_samples = 0:Ts:2`. Use appropriate `t_samples` in the tasks below as well.

6.3 Fast Fourier Transform (Radix-2)

Write a matlab function `radix2fft` which takes as input an N-length vector x and returns an N-length vector X. To compute X, implement the decimation-in-time radix-2 FFT algorithm for an input vector whose length is a power of 2 i.e. assume that $N = 2^m$. Note that this function will have a recursive structure. Specifically, `radix2fft` is called within itself until it reaches the stopping criteria of N = 2. What is the DFT when N = 2? Verify that the output of your function matches with that of `fft` within numerical precision.

6.4 Filtering of periodic signals with LTI systems

A continuous-time periodic signal $x(t)$ has Fourier series (FS) coefficients a_k. If $x(t)$ is input to the LTI system with frequency response $H(\omega)$, what are Fourier series coefficients of the output signal? What about the periodicity of the output signal?

Ideal low pass filter (LPF): let $\omega_c > 0$ be its cut-off frequency. We wish to find FS coefficients b_k, $k = -N: N$, of the output signal when the input signal with coefficients a_k, $k = -N: N$, is passed through an Ideal LPF.

(a) Write a code for the matlab function `B = myLPF(A,w0_FS,wc)` which takes input signal FS coefficents A, frequency of the input periodic signal `w0_FS`, cut-off frequency wc and returns the output signal FS coefficients in the vector B. Note that your code should be written for general N.

Note: the coefficient a_k corresponds frequency $k\omega_0$, use this information while implementing your LPF.

(b) We will now write a matlab script file to visualize input and output signals of the filter. For this purpose we will use function '`x = partialfouriersum(A,T,t)`' from last week's lab session.

>> Initialize $\omega_0 = 1$, and FS coefficients A to obtain the signal $x(t) = \cos(t)$

>> Call function `myLPF` with input A and $\omega_c = 2$

>> Use the function `partialfouriersum` to obtain the time domain signals corresponding to A and B and plot them in the same figure. Use the inputs as T = 2π, and t = -2T:0.01:2T
>> What happens when we change cut-off to $\omega_c = 0.5$?

(c) Ideal high pass filter (HPF): let $\omega_c > 0$ be its cut-off frequency.

>> Repeat (a) for an ideal HPF and write matlab function `B = myHPF(A,w0_FS,wc)`.
>> Repeat (b) with the ideal LPF replaced by ideal HPF (continue in the same script file).

(d) Non-ideal filter: let the frequency response be $H(\omega) = \frac{G}{a+\omega}$, where G and a are positive real constants. What is the nature of this filter?

>> Write a code to implement this non-ideal filter on FS coefficients A as the matlab function `B = NonIdeal(A,w0_FS,G,a)`.
>> Repeat (b) with ideal LPF replaced by the non-ideal filter. Use G = 1, a = 1.
>> How is the complex-valued nature of the LTI system frequency response manifested in the output signal?

(e) Repeat the script with the input signal as $x(t) = \sin(2t) + \cos(3t)$. Note that you must appropriately modify A, ω_0, and T for this example. For this input set ideal LPF and ideal HPF filter cut-offs to be $\omega_c = 2.5$.

Optional: repeat when A corresponds to FS coefficients of the periodic square wave.

6.5 Reconstruction methods

We will make use of the `interp1()` matlab function for reconstruction of continuous-time signal from samples. Read the documentation and examples before using this function. For this matlab script, repeat part 3.1 above and plot it in the top-left panel of a figure with 2x2 subplots. In each of the following three parts, perform reconstruction as indicated and plot the original samples ($x[n]$) and reconstructed signal in the remaining panels. Use the values of `t_fine, t_samples` and Ts from part 6.1.

a) From the samples $x[n]$, perform *zero-order hold* reconstruction of $x(t)$. Use `interp1()` command to get this signal with appropriate selection of the 'method'.

Note that the reconstruction signal should be computed over the time-grid `t_fine`.

b) From the samples $x[n]$, perform *linear interpolation* based signal reconstruction of $x(t)$. Use the `interp1()` command.

c) Recall that the ideal reconstruction using sinc function is given by the formula

$$x_r(t) = \sum_{n=-\infty}^{\infty} T_s\, x(nT_s) \frac{\sin\big(\omega_c(t - nT_s)\big)}{\pi(t - nT_s)} \quad \rightarrow (1)$$

Though this is an infinite sum and cannot be exactly implemented in a computer, we will approximately implement it by restricting to the time interval [0, 2] and using only the samples $x[n]$ we have from that interval.

>> Write a matlab function `sinc_recon()` with inputs and outputs as follows:

```
function xr = sinc_recon(n,xn,Ts,t_fine)
% n - the integer locations of the samples x[n]
% xn - the sampled signal x[n] = x(n*Ts)
% Ts - the sampling interval
% t_fine - the time-grid for reconstruction of xr
% xr - the reconstructed signal over the time-grid t_fine
```

From the samples $x[n]$, find the approximate *sinc interpolated* signal as given by the above formula. Always use a cut-off frequency of $\omega_c = \omega_s/2$, where $\omega_s = 2\pi/T_s$.

While manually writing the sinc expression take precaution to avoid divide-by-zero issue (matlab will not flag this but your code will have errors). Alternately, you can use the inbuilt `sinc()` command but with appropriate time scaling to get required cutoff frequency. Read up the documentation for this function before using.

Restrict each of the sinc in the summation to the interval [0, 2].

>> Compare the quality of the three interpolations visually. How does the quality of sinc reconstruction vary within the interval [0, 2]? Give explanation for your observations.
>> In your script, for each of the three interpolation methods above, compute the maximum absolute error (MAE) between the original signal and the reconstructed signal in the interval [0.25, 1.75].
>> (Optional): try some of the other interpolation 'method' available in the command `interp1()` and check how the quality of reconstruction and the MAE changes.

6.6 Sampling non-band-limited signal

We know that sampling theorem can be applied only for band-limited signals. All the above tasks had band-limited signals. We now consider a non-band-limited signal and investigate its reconstruction as sampling interval Ts is changed. Write a matlab script for following.

>> Consider the continuous-time triangular pulse signal of height 1, base in the interval [-1,1], and zero otherwise. Because this is a time-limited signal, only finite number of terms appear in the reconstruction formula (1) above (though the sinc shape is still infinite in time extent).
>> For a sampling interval of Ts, what is the corresponding `t_samples` vector so that we only consider samples at the base of the triangle (assume there is a sample starting at -1)? Generate the corresponding samples $x[n]$ and the discrete-time indices n. Use these as inputs below.
>> Perform sinc interpolation for this signal using samples generated for the four intervals i) Ts = 0.5s, ii) Ts = 0.2s, iii) Ts = 0.1s and iv) Ts = 0.05s. For reconstruction, use a time-grid of `t_fine = -10:0.001:10`.

>> Create a figure with 2x2 subplots, one panel for each Ts. In each panel plot the samples and the reconstructed signal corresponding to the four sampling intervals. What are your observations as sampling interval is changed?

6.7 Audio signals

Download the 4 audio files given here. Save them in your working folder so that the audio files and code are in same folder. Write a matlab script for the following tasks.

>> Look up the audio file properties and note down its *bit rate*.
>> Look up documentation of the inbuilt matlab function `audioread()`. Use it to load the audio file in to your matlab workspace. What is the *sampling frequency* f_s of these audio signals?
>> From the length of the loaded signal and the *sampling frequency,* compute the duration of each of the audio signals in seconds.
>> From the observed *bit rate* and *sampling frequency*, compute how many bits the ADC must have used while quantizing/storing these signals. How many levels of quantization this ADC can perform?

>> Look up documentation of the inbuilt matlab function `sound()`. This function in combination with appropriate hardware in your computer performs DAC. Use it to listen to the audio files you have loaded in your workspace (ignore the 'nBits' input to this function).
>> For one of the files, listen to the sound using lower sampling frequency than the true f_s (for example you can use $0.9\, f_s$, $0.8\, f_s$, $0.7\, f_s$, etc. while using `sound()` command). What do you notice?
>> Repeat the above using higher sampling frequency than the true f_s (for example you can use $1.2\, f_s$, $1.\,4\, f_s$, $1.6\, f_s$, etc. while using `sound()` command). What do you notice?
>> What property of Fourier transform can you use to explain your observations above.
>> Write down your answers as comments at the end of the code.

6.8 Aliasing

Aliasing occurs when the sampling frequency is less than the Nyquist rate required by the sampling theorem. In this matlab script we will look at the effect of aliasing for the signal $x(t) = \cos\,(5\pi t)$.

>> What is the Nyquist rate for this $x(t)$?
>> Consider samples of $x(t)$ for the following sampling intervals i) Ts = 0.1s, ii) Ts = 0.2s, iii) Ts = 0.3s, and iv) Ts = 0.4s. For each of these cases perform sinc interpolation from samples over the interval [0, 2].
>> Create a figure with 2x2 subplots, one panel for each Ts. In each panel plot the samples and the reconstructed signal corresponding to the four sampling intervals. What are your observations as sampling interval is changed?