Matlab Practice Project

Problem 1: Sampling

The operation of signal deletion (or decimation or down sampling) is defined by the pointwise mapping from $l_2(\{0,1,2,\ldots,N-1\}; \mathbb{R})$ to $l_2(\{0,1,2,\ldots,\frac{N}{M}-1\}; \mathbb{R})$, $N$ divisible by $M$:

$$y(n) = x(nM), n \in \{0,1,2,\ldots,\frac{N}{M}-1\}$$

in which the signal $x$ is down-sampled by an integer factor $M$. For example if

$$x = \{\ldots,-2,4,3,-6,5,-1,8,\ldots\}$$

then the down-sampled sequence by a factor 2 are given by

$$y = \{\ldots,-2,3,5,8,\ldots\}$$

a) Write a MATLAB function dnsample that has the form

$$y = dnsample(x, M)$$

to implement the above operation. Use the indexing mechanism of MATLAB with careful attention to the origin of the time axis $n = 0$.

b) Generate

$$x(n) = \sin(0.2\pi n), \quad -100 \leq n \leq 100.$$ 

Decimate $x$ by a factor of 4 to generate $y$. Plot both $x(n)$ and $y(n)$ using the subplot command and comment on the results.

c) Repeat the above using

$$x(n) = \sin(0.6\pi n), \quad -100 \leq n \leq 100.$$ 

Qualitatively discuss the effect of sampling on the signals.

d) Using the FFT function in Matlab calculate and plot the Fourier transform of all the above signals. Compare the Fourier transform of the original and down-sampled signals in parts (b) and (c). In which case, can the original signal be recovered using the down-sampled version? Calculate the maximum frequency of the sinusoid such that one can recover the original signal using the down-sampled signal.

FFT is a very important algorithm to implement DTFT ($\mathcal{F}_{DD}$) in a computationally efficient fashion. The $\text{fft}$ command in Matlab generates the transform.

Problem 2: Low-Pass Filters and Noise Removal

Suppose we have signal $x \in S$ given by:

$$x(t) = 5e^{-(t-25)^2/4}$$
Suppose this signal is perturbed by some high-frequency noise signal given by:

\[ n(t) = 4 \cos(16\pi t) + 2 \sin(20\pi t) \]

Suppose we have a receiver which receives:

\[ y(t) = x(t) + n(t), \]

The goal of the receiver is to reconstruct the original signal with low distortion.

(i) Plot \( x \) and \( y \) as a function of time.

(ii) We wish to remove the noise term from the signal: Observe that the signal \( x \in S \) does not have compact support. As such, we could work with CCFT (\( F_{CC} \)) and exploit the fact that the signal’s CCFT bandwidth also decays to zero very fast: Recall that signals in \( S \) are transformed to signals in \( S \) under CCFT; and in fact, by Homework 6, Problem 3, the CCFT of a Gaussian signal is also Gaussian.

One way to extract the signal is to sample \( y \) with some period sufficiently small to obtain a discrete time representation, using the Nyquist-Shannon Sampling Theorem (by approximating the bandwidth of the original signal by some finite value). We should be able to reconstruct the continuous-time signal back from the samples if we can take the noise terms out.

Now that we have a discrete-time signal, we could apply it to a low-pass filter to take the noise terms out of the signal. Use an ideal low-pass filter for this step, and convert the signal back to the time domain.

(iii) Plot the resulting signal. Plot the resulting continuous time signal.

(v) We will now repeat steps (iii-iv) using a more practical filter. The ideal low-pass filter is not easy to implement and it requires a non-causal system. There are a number of filtering methods used in practice.

One common practical method is known as the Butterworth filter. This filter has a linear system description:

\[ \hat{h}(f) = \frac{\sum_{k=0}^{N} b_k e^{-2\pi f_k}}{\sum_{m=0}^{N} a_m e^{-2\pi f_m}}. \]

It is easier to conceptually visualize such a filter in continuous time: An \( N \)th order Butterworth filter with cut-off frequency \( f_c \) in continuous time has the following frequency response magnitude:

\[ |\hat{h}(f)| = \frac{1}{1 + \left| \frac{f}{f_c} \right|^N} \]

Observe that, the filter is such that, for large \( N \) values, the filter resembles an ideal low pass filter.

In Matlab, the command, \texttt{butter}(n,W) returns a filter of order \( n \) - an \( n \)th order linear system - with the desired characteristics. That is, this function returns a linear system \([A,B,C,D]\) with description:

\[ x(n+1) = Ax(n) + Bu(n) \]
\[ y(n) = Cx(n) + Du(n). \]

Here, \( u \) is the input signal and \( y \) is the output.

Using Matlab, pick an appropriate \( W \) and an order \( n \) such that the resulting plot resembles the noise free signal. Plot the output.