Homework Set # 7

Problem 1 (Aliasing)

Let $x(t) = \text{sinc}(t) = \frac{\sin(\pi t)}{\pi t}$.

(a) Write down $X_c(j\Omega)$, the continuous time Fourier transform (CTFT) of $x(t)$, and plot it.

(b) Now, assume that $x(t)$ is sampled at a sampling frequency $f_s = \frac{1}{2}$. Find $\tilde{X}_c(j\Omega)$, which is defined as

$$\tilde{X}_c(j\Omega) = \sum_{k=-\infty}^{\infty} X_c(j\Omega - jk\Omega_s).$$

Plot $\tilde{X}_c(j\Omega)$.

(c) Let $x[n]$ be the sequence of samples. Find $X(e^{j\omega})$, the DTFT of $x[n]$.

(d) Now, assume that we do a sinc-interpolation of the signal, as specified in Section 10.6.3 of the course notes. Let $\hat{X}(j\Omega)$ be the CTFT of the interpolation of $x[n]$. Find $\hat{X}(j\Omega)$ and plot it.

(e) Find $\hat{x}(t)$, which is the interpolation of $x[n]$. Is it equal to $x(t)$? Explain why or why not.

Problem 2

Consider the Fourier transform of the signal $x_c(t)$, given in Fig. 1.

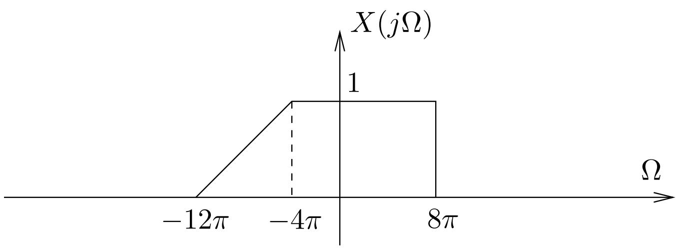

Figure 1: $X(j\omega)$

(a) What is the bandwith of the signal, *i.e.,* the minimum Ω_N such that $X(j\Omega) = 0$ for $|\Omega| > \Omega_N$? What is the Nyquist sampling frequency for this signal?

(b) We sample $x_c(t)$ with sampling period $T_s = \frac{1}{12}$ sec. Draw the sampled spectrum of the signal, $X_s(j\Omega)$. Specify the value of the important points on both the axes.

(c) Draw the spectrum of $X(e^{j\omega})$, the DTFT of the discrete signal $x[n] = x_c(\frac{n}{12})$.

(d) Let we want to recover the signal from its sampled version $X_s(j\Omega)$. Find the corresponding filter we can use for that. Is it possible to do the exact reconstruction?

(e) Repeat parts (b) and (d) for $T_s = \frac{1}{8}$ sec.

(f) Define a new function in terms of $x_c(t)$ as $y_c(t) = e^{j2\pi t}x_c(t)$. Find $Y(j\Omega)$, the Fourier transform of $y_c(t)$, and draw it.

(g) Let us sample from the new signal with sampling period $T_s = \frac{1}{10}$ sec. Draw the corresponding sampled spectrum, $Y_s(j\Omega)$.

(h) Is there any aliasing effect in $Y_s(j\Omega)$? Is it possible to recover the original signal $x_c(t)$ from $Y_s(j\Omega)$? If yes, explain the required steps and write down the explicit formula, otherwise, prove your answer.

(i) Recall the Nyquist sampling frequency found in (a). Is it in contradiction with the result of part (h)? Why?

Problem 3

We are considering a continuous-time signal $x_c(t)$ with corresponding continuous-time Fourier transform $X_c(j\Omega)$ given in Figure 2.

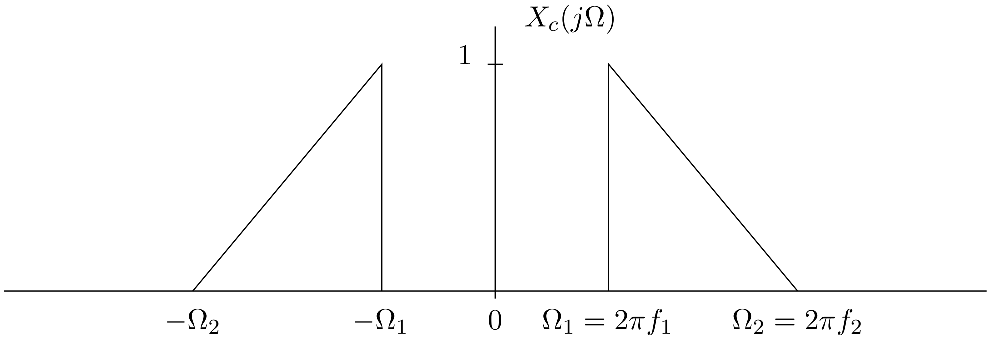

Figure 2: Continuous-time Fourier transform of $x_c(t)$ in Problem 3.

(a) Suppose that we are sampling $x_c(t)$ with period T_s, i.e. we create

$$x[n] = x_c(nT_s), \qquad n \in \mathbb{Z}.$$

According to the sampling theorem, what is the maximum sampling period T_s for which $x_c(t)$ is recoverable from $\{x[n]\}$?

(b) Suppose $\Omega_1 = 2\pi \cdot 150$, $\Omega_2 = 2\pi \cdot 200$. Let us sample at rate $f_s' = 100$ Hz. Does this satisfy the condition given in part (a)?

(c) Let $v[n] = x_c(nT_s')$, $n \in \mathbb{Z}$, with $f_s' = \frac{1}{T_s'} = 100$ Hz. Sketch the spectrum of $v[n]$.

(d) Let $x_s(t) = \sum_n v[n]\delta(t - nT_s')$ be a continuous-time signal. Sketch the continuous-time Fourier transform $X_s(j\Omega)$ of $x_s(t)$.

(e) Can we recover $x_c(t)$ from $v[n]$? If so, clearly and explicitly demonstrate the method. If not, explain.
Hint: Can $x_c(t)$ be reconstructed from $x_s(t)$ found in part (d) of problem?

Problem 4 (Interpolation)

For this exercise you need to use the MATLAB files which are in the file `hw7_matlab.zip`, available on the course webpage.

Zero-Order and First-Order Hold, Sinc Interpolation

The file `interpol.m` defines the function

```
f = interpol(x, t, I, Ts)
```

that implements the interpolation formula seen in class,

$$x(t) = \sum_{n=-\infty}^{\infty} x[n] I\left(\frac{t - nT_s}{T_s}\right).$$

The function `interpol` has the following arguments:

`x` A vector containing the finite-length signal $x[n]$.

`t` A vector of time instances on which the interpolated signal $x(t)$ will be evaluated. The return value `f` will be a vector of the same size as `t`. Of course `t` can also be just a scalar.

`I` A handle to the interpolation function $I(t)$ (look again at Homework 6 if you don't remember what a function handle is). `I` must be a handle to a function of the form `f = I(t)` where `t` is a vector containing the time instants on which the function $I(t)$ is to be evaluated.

`Ts` The sampling period T_s.

(a) Write two functions `Izero` and `Ifirst` that implement, respectively, a zero-order hold and a first-order hold interpolator. Create the signal $x[n]$, $n = 0, \ldots, 9$ by sampling the continuous time-signal $x_c(t) = \sin(2\pi f t)$ for $f = 440$ Hz and $T_s = 1$ ms:

```
>> Ts = 1/1000;
>> f = 440;
>> n = 0:9;
>> x = sin(2*pi*f*n*Ts);
```

Use `interpol` along with `Izero` and `Ifirst` to create a stem-plot of $x[n]$, superimposed with the zero-order hold and first-order hold interpolation. For your plot, use a timescale of `t = 0:Ts/100:9*Ts` i.e., the plot will show 100 interpolated points for each sample. To create the interpolated signals, write

```
>> xzero = interpol(x, t, @Izero, Ts);
>> xfirst = interpol(x, t, @Ifirst, Ts);
```

Hint: The easiest way to implement a function of the form

$$f(t) = \begin{cases} a & \text{if } t > c \\ b & \text{if } t \leq c \end{cases}$$

is using the following code:

```
f = zeros(size(t));
f(t > c) = a;
f(t <= c) = b;
```

This should make it easy to implement the interpolators.

(b) On the same figure, plot the interpolation using $I(t) = \text{sinc}(t)$.

Lagrange Interpolation

The file `interpol_lag.m` defines the function

```
f = interpol_lag(x, t, Ts)
```

that implements Lagrange interpolation (see Equation 10.33 of the course notes). The parameters `x`, `t`, and `Ts` have the same meaning as in `interpol`, except that `x` must be a vector of length $2N + 1$, representing $x[n]$ with $n = -N, \ldots, N$.

(c) Write the function

```
f = lagrange(t, k, N, Ts),
```

used by `interpol_lag`, which implements the Lagrange polynomial formula seen in class,

$$L_n^{(N)}(t) = \prod_{\substack{k=-N \\ k\neq n}}^{N} \frac{t/T_s - k}{n - k}, \quad n = -N, \ldots, N.$$

(d) Create the signal `x` as in (a), but with $n = -4, \ldots, 4$. Plot on the same figure a stem plot of $x[n]$ and its interpolation using Langrange polynomials. To compute the interpolation, write

```
>> xlagrange = interpol_lag(x, t, Ts)
```

(e) Plot the superposition of $\text{sinc}(t/T_s)$ and $L_0^{(N)}$ for $T_s = 10$ ms and for $N = 1$, 5, and 10 to verify that $L_0^{(N)}(t)$ indeed approaches $\text{sinc}(t/T_s)$ as N becomes large.

For this exercise you need to hand in a printout of all your plots and of all your MATLAB code.