Posting Date: November 11, 2015.

Problem 1: In the lecture, we considered an analog signal $x_a(t)$ sampled at sampling intervals T by impulses according to

$$x_\delta(t) = \sum_{n=-\infty}^{\infty} x_a(nT)\delta(t-nT).$$

We showed that the Fourier transform of the impulse-sampled signal $x_\delta(t)$ is related to $X_a(f)$, the Fourier transform of $x_a(t)$, by

$$X_\delta(f) = \frac{1}{T}\sum_{n=-\infty}^{\infty} X_a\left(f - \frac{n}{T}\right).$$

We can derive a similar relationship between $X_a(f)$ and the Fourier transform of the discrete-time signal

$$x(n) = x_a(nT), \quad -\infty < n < \infty.$$

To make this easier, we will define the frequency variable $\tilde{f} = fT = f/f_s$. This variable, which has units of cycles per sample and is often called the *normalized frequency*, is used as the frequency variable for discrete-time signals. For example, a discrete-time sinusoid can be represented as $x(n) = \cos 2\pi\tilde{f}n$. For discrete-time signals, only frequencies in the range $-\frac{1}{2} \leq \tilde{f} \leq \frac{1}{2}$ are unique; all frequencies $|\tilde{f}| > \frac{1}{2}$ are aliases. The Fourier transform of a discrete-time signal $x(n)$ is defined by

$$X(\tilde{f}) = \sum_{n=-\infty}^{\infty} x(n)e^{-j2\pi\tilde{f}n}.$$

and the inverse transform is defined by

$$x(n) = \int_{-1/2}^{1/2} X(\tilde{f})e^{j2\pi\tilde{f}n}d\tilde{f}.$$

Derive the relationship between $X(\tilde{f})$ and $X_a(f)$.

Hint: This is a standard relationship whose derivation can be found in many texts that treat digital signal processing. You are free to use such a text as a guide or you may perform the derivation yourself following these steps:

(a) Express $x(n) = x_a(nT)$ in terms of $X_a(f)$.

(b) Equate this expression with the inverse transform definition given above.

(c) Make a change of variable to eliminate $\tilde{f}$ in favor of f in the expression.

(d) Divide the range of the integral that goes from $-\infty$ to ∞ into an infinite number of integrals of width f_s, then re-express the integral as a sum of integrals.

(e) Make some deductions to arrive at the desired relationship between $X(\tilde{f})$ and $X_a(f)$.

Problem 2: In the lecture, we showed that uniform (first-order) sampling and digital down-conversion can be employed to obtain in-phase $I(m)$ and quadrature $Q(m)$ samples of the baseband complex representation of a bandpass signal $x_a(t)$. If quantization effects are ignored, then with sufficient processing, $I(m)$ can be made to approach $x_c(mT_l)$ and $Q(m)$ can be made to approach $x_s(mT_l)$ with arbitrary accuracy, where $x_c(t)$ and $x_s(t)$ are the continuous-time in-phase and quadrature components of the baseband representation $x_l(t) = x_c(t) + jx_s(t)$, respectively, and T_l is the complex sampling interval.

Knowing how to go back and forth between a discrete-time baseband representation of a signal in terms of $I(m)$ and $Q(m)$ and a discrete time representation of the signal at the original carrier frequency (or at some intermediate frequency) is useful for simulation and analysis of baseband signals.

Write two MATLAB functions for converting between these signal representations. Your function for converting from $I(m)$ and $Q(m)$ samples to a bandpass representation at some arbitrary f_{IF} should be called `iq2if` and should adhere to the following interface:

```
function [xVec] = iq2if(IVec,QVec,Tl,fIF)
% IQ2IF : Convert baseband I and Q samples to intermediate frequency samples.
%
% Let xl(m*Tl) = I(m*Tl) + j*Q(m*Tl) be a discrete-time baseband
% representation of a bandpass signal.  This function converts xl(n) to a
% discrete-time bandpass signal x(n) = I(n*T)*cos(2*pi*fIF*n*T) -
% Q(n*T)*sin(2*pi*fIF*n*T) centered at the user-specified intermediate
% frequency fIF, where T = Tl/2.
%
%
% INPUTS
%
% IVec -------- N-by-1 vector of in-phase baseband samples.
%
% QVec -------- N-by-1 vector of quadrature baseband samples.
%
% Tl ---------- Sampling interval of baseband samples (complex sampling
%               interval), in seconds.
%
% fIF --------- Intermediate frequency to which the baseband samples will
%               be up-converted, in Hz.
%
%
% OUTPUTS
%
% xVec -------- 2*N-by-1 vector of intermediate frequency samples with
%               sampling interval T = Tl/2.
%
%
%+------------------------------------------------------------------------------+
% References:
%
%
%+==============================================================================+
```

Your function for converting from a bandpass representation at some arbitrary f_{IF} to $I(m)$ and $Q(m)$ samples should be called `if2iq` and should adhere to the following interface:

```
function [IVec,QVec] = if2iq(xVec,T,fIF)
% IF2IQ : Convert intermediate frequency samples to baseband I and Q samples.
%
% Let x(n) = I(n*T)*cos(2*pi*fIF*n*T) - Q(n*T)*sin(2*pi*fIF*n*T) be a
% discrete-time bandpass signal centered at the user-specified intermediate
% frequency fIF, where T is the bandpass sampling interval.  Then this
% function converts the bandpass samples to quadrature samples from a complex
% discrete-time baseband representation of the form xl(m*Tl) = I(m*Tl) +
% j*Q(m*Tl), where Tl = 2*T.
%
%
% INPUTS
%
% xVec -------- N-by-1 vector of intermediate frequency samples with
%               sampling interval T.
%
% T ----------- Sampling interval of intermediate frequency samples, in
%               seconds.
%
% fIF --------- Intermediate frequency of the bandpass signal, in Hz.
%
%
% OUTPUTS
%
% IVec -------- N/2-by-1 vector of in-phase baseband samples.
%
% QVec -------- N/2-by-1 vector of quadrature baseband samples.
%
%
%+------------------------------------------------------------------------------+
% References:
%
%
%+==============================================================================+
```

For the function `iq2if`, you may wish to use the MATLAB function `interp` from the signal processing toolbox to resample the I and Q data.

For the function `if2iq`, you could use one of the methods described in the papers

- `Quadrature sampling with high dynamic range.pdf`
- `Bandpass signal sampling and coherent detection.pdf`

found on iLearn. But you can just as well employ the straightforward discrete-time implementation of the continuous-time "quadrature approach" to bandpass sampling discussed in lecture, in which case you may find helpful the MATLAB function `decimate` from the signal processing toolbox .

The functions `if2iq` and `iq2if` should act as inverse operations; that is, aside from some high-frequency components lost in filtering, by calling `iq2if` and then `if2iq` you should recover your original data.

Problem 3: Check to see that your functions `iq2if` and `if2iq` preserve the spectral shape of GPS data. Download the binary file from the server at

`http://overmars.engr.ucr.edu/GNSS_Course/niData01head.bin`

The file contains baseband complex (I and Q) samples of the GPS L_1 C/A signal originally centered at 1575.42 MHz with a bandpass signal bandwidth of 4 MHz. Samples were taken at a complex sampling rate of $f_s = \frac{4}{0.8} = 5$ MHz. You can use the following snippet of MATLAB code to read in 0.5 seconds of data and store it in a complex vector `Y`:

```
clear; clc;
%----- Setup
Tfull = 0.5;          % Time interval of data to load
fsampIQ = 5.0e6;      % IQ sampling frequency (Hz)
N = floor(fsampIQ*Tfull);
nfft = 2^9;           % Size of FFT used in power spectrum estimation

%----- Load data
fid = fopen('C:\yourDataPath\niData01head.bin','r','l');
Y = fread(fid, [2,N], 'int16')';
Y = Y(:,1) + j*Y(:,2);
fclose(fid);
```

Use the `pwelch` function to estimate the power spectrum of the complex data just as you did in Homework Assignment 2 for the data from the Stanford "big dish." Next, use your `iq2if` function to convert the data to a bandpass signal at $f_{IF} = 2.5$ MHz. Again, use the `pwelch` function to estimate the power spectrum. Describe how this spectrum differs from that of the original baseband data. Finally, convert the bandpass data *back* to baseband with your function `if2iq` and compare the power spectrum of this data with the spectrum of the original data. What difference do you note?

Problem 4: Suppose we have a bandpass signal with $B = 4$ MHz centered at the GPS L_1 frequency. Assuming $f_H = f_{\mathrm{L}_1} + B/2$ and $f_L = f_{\mathrm{L}_1} - B/2$, calculate the maximum wedge index $k_{\max}$ and the theoretical minimum sampling frequency $f_{s,\min}$ for this signal. To avoid aliasing, assume that we instead sample at $f_s = 2W$, where $W = \frac{B}{0.8}$. This effectively introduces guard bands on each side of the signal band. Recalculate $k_{\max}$ for this case. Draw a diagram of the operating point within an exploded view of the $k_{\max}$th wedge of the plot for allowed and forbidden sampling frequency regions for bandpass signals showing the distance between the operating point and each of the forbidden walls. Your diagram should resemble Figure 6.4.4 of the Proakis reading (found on iLearn).

Problem 5: It is important to understand how to model noise in the conversion from analog to digital signals. Consider the alternative models presented in Figure 1. In both models, $n_a(t)$ is a Gaussian zero-mean white noise process with (two-sided) power spectral density $N_0/2$ W/Hz. This means that the total power in $n_a(t)$, which is given by

$$P_n = \int_{-\infty}^{\infty} S_n(f)df = \int_{-\infty}^{\infty} \frac{N_0}{2}df$$

is infinite. Of course, this cannot really be the case, but we model the noise as spectrally flat (white) nonetheless. If we model the sampling process according to the upper model of Figure 1, then the variance of the discrete time samples is

$$\sigma_n^2 = E[n_a(jT)n_a(jT)] = R_n(0) = P_n.$$

In other words, our model predicts an infinite variance of the discrete-time noise samples, which is not realistic or convenient. In the lower model in Figure 1, the sampler is preceded by an anti-aliasing filter with a (single-sided) noise-equivalent bandwidth of B Hz. For this model, calculate the variance of the noise samples $n(j)$ in terms of N_0 and B.

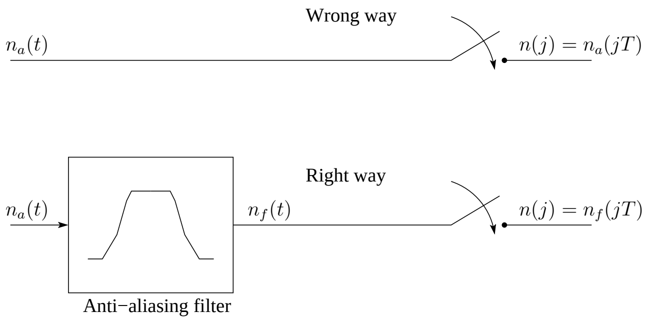

Figure 1: Two models of wideband noise sampling.

Problem 6: Perform GPS L_1 C/A signal acquisition on the complex baseband data in the binary file at the server

`http://overmars.engr.ucr.edu/GNSS_Course/niData01head.bin`

Determine the following for each signal present:

- PRN identifier
- Approximate apparent Doppler frequency (in Hz)
- Approximate C/A code start time offset assuming that the first sample corresponds to $t = 0$.
- Approximate carrier-to-noise ratio C/N_0.

Hints:

(a) Note that because the data in `niData01head.bin` are complex baseband data, you will need to either (i) shift them up to some intermediate frequency (e.g., 2.5 MHz) using your `iq2if.m` function, or (ii) operate on the data with a complex baseband (zero-intermediate-frequency) model in mind.

(b) You will have to try all PRNs. The 1-ms L_1 C/A PRN codes are Gold codes built up by combining two maximum-length LFSR sequences. For GPS L_1 C/A, the length of the LFSR is $n = 10$ and the length of the m-sequences is $N = 2^{10} - 1 = 1,023$. The characteristic polynomials for the GPS L_1 C/A codes are:

$$\begin{aligned} f_1(D) &= 1 + D^3 + D^{10} \\ f_2(D) &= 1 + D^2 + D^3 + D^6 + D^8 + D^9 + D^{10}. \end{aligned}$$

Start off with both the f_1 and f_2 LFSRs loaded with the all-ones sequence `1111111111`. Obtain the different GPS PRN Gold codes by circularly shifting the output of the f_2 LFSR before modulo-2 adding it to the output of the f_1 LFSR.

The GPS Interface Specification (IS) posted on iLearn has details on generating the GPS L_1 C/A codes. See section 3.3.2.3 and the accompanying figures. Also see Table 3-I. The easiest way to interpret this table is to take the code delay chips (column 5) and shift the f_2 sequence by that amount before modulo-2 adding it to the f_1 sequence.