UCLA — Electrical and Computer Engineering Dept.
ECE132A: Introduction to Communication Systems
Project
Due: Friday June 3, 2022

---

1. Please form groups of one to three people. Make sure that at least one person in the group is reasonably familiar with MATLAB or PYTHON.

2. Submit MATLAB or PYTHON code and all the figures. Please, comment your code! (Remember that in MATLAB anything that follows a % sign is considered a comment; in PYTHON the comment character is a #.)

3. If you do generate data (symbol of bit vectors), then we'll find a way for you to share it with me. Stay tuned.

4. Each group should generate one set of charts (one per group) describing the work you've done, the results (plots, block diagrams, if any, etc.), and any comments you may have. Preferably, use Powerpoint or Keynote. Please indicate clearly the names of the team members in the front page of your charts.

5. Please submit your projects to Gradescope.

Project Description

You are given two files containing I and Q samples of a digitally modulated signal. Start with the vector named `yrx`. Your task is to analyze the samples and, at a minimum determine the modulation scheme. Try to also provide the symbols and/or the bits. The processing required for you to determine the modulation includes estimating and removing a frequency offset, match-filtering, finding the correct sampling times, and possibly estimating and removing a phase offset. Do only what you can. Anything you do (if well documented and explained) will be considered extra credit. Remember: the project is entirely optional.

1 What is known

The following is known:

1. The symbol rate is normalized to 1 ($T = 1$, $R_\mathrm{s} = 1/T = 1$).

2. The sampling rate is 16 samples per symbol ($F_\mathrm{s} = 16$).

3. The pulse shape is root raised square cosine with rolloff equal to 0.5.

4. In addition, we also know that the signal contains *frames* of 1000 symbols, each preceded by a preamble, which is a sequence of 10 binary symbols of alternating sign, i.e., $-1, 1, -1, 1, -1, 1, -1, 1, -1, 1$.

2 Assignment

1. Start by looking at the power spectrum of the signal, using a command of the type
   `[px,pf] = pwelch(yrx(1:N),[],[],1024,Fs,'centered');`
   for some value of `N`, in the order of, say 2^{20}. You can experiment with this value and other parameters of `pwelch`. You can display the spectrum by typing:
   `semilogy(pf,px)`.

2. The frequency offset can be estimated by looking at the FFT of signal raised to a power as in:

   ```
   semilogy(frqs/p,fftshift(abs(fft(yrx(1:N).^p))))
   ```

   where `p` is 2, or 4, or 8. Experiment with this. `N` does not need to be the same as before. Try different values. I suggest something like 2^{15}. If you choose too small a value, you may not have a sufficiently fine resolution. The frequency vector is defined as:
   `frqs = (-N/2:N/2-1)'*Fs/N;`
   Again, feel free to experiment. Once you estimate the frequency offset, derotate (downconvert) the signal by that frequency:

   ```
   yderot = yrx .* exp(-2i*pi*frEst*(0:length(yrx)-1)'/Fs);
   ```

3. Generate a root raised cosine pulse with the correct rolloff parameter. You can use MATLAB's filter (somewhat hard to figure out, admittedly), or use the simple code `rrc.m` provided. In this case, the filter is obtained by typing
   `hrrc = rrc((-6:1/samplesPerSymbol:6)',rolloff,1)/samplesPerSymbol;`
   This corresponds to a pulse with a span of 12 symbols.

4. Match-filter the derotated signal. Sample the matched filter output at one sample per symbol and try to determine the correct sampling time (by finding the sampling point that gives you the "tightest" cluster plot: you can determine that visually, or, if so inclined, you can write a program that does it). You can use something like the following:

   ```
   for k = 1:sampsPerSym
       plot(yrxmf(k:sampsPerSym:N),'*');
       title(['k = ',num2str(k)])
       axis('square');
       pause(0.5);
   end
   ```

   Again, experiment with `N`. If you feel inspired, you could resample at a higher number of samples per symbol in order to have a finer resolution. Look at MATLAB's `interp1` function.

5. The constellation you find should be one of the DVB-S2 or DVB-S2X constellations (see the standard documents, provided). Which one is it?

6. Is there a phase offset? Can you estimate it? You could use the preamble to determine both timing and phase offsets with better accuracy (only if you want to and have the time). Cross-correlate the derotated signal with the modulated preamble:

```
preambleSig = conv(upsample([-1;1;-1;1;-1;1;-1;1;-1;1],sampsPerSym),hrrc,"full");
preambleSig = preambleSig(fltDelay*sampsPerSym+1:end-fltDelay*sampsPerSym);
plot(real(xcorr(yderot(1:N),preambleSig)))
```

7. The second file, `yrx2`, has been sampled at a rate which is slightly higher than 16 samples per symbol. Look at this file only if you have additional time and interest. The symbol rate could be determined looking at a spectral line obtained by passing the signal through a nonlinearity, as in the following:

```
semilogy(frqs,fftshift(abs(fft(abs(yrx2(1:N))))))
```

Neglect the line at DC. Can you determine what the symbol rate is (relative to the sampling rate)? You could resample the signal so as to have an integer number of samples per symbol (again, look at `interp1`). Once you have resampled the file, you could process it the same way as `yrx`.