ECE 3311 / B-term 2016
FINAL PROJECT

Description of the project

- Your task is to simulate a digital radio receiver, as described in the modified version of Chapter 15 (available on the course web page). You will use techniques you have developed in the homework assignments to create a Matlab simulation that will demodulate and equalize a pulse-shaped digital signal transmitted at radio frequencies. All of the receiver specifications are given in the modified Chapter 15.

- The system employs a (5,2) block code which adds redundancy to the bits as described in Chapter 14. We have not (yet) covered coding in lecture, but you should familiarize yourself with section 14.6.1 at the very least.

- Test files are available on the course web site. Each file will contain a rolloff factor, a vector of received data, the index of the user you are to recover (1, 2, or 3) and the original text messages. There will be 3 test files: `easy.mat` (noiseless, unity channel, no carrier frequency offset, no baud timing offset, etc), `medium.mat` (moderate impairments), and `hard.mat`.

- You are not required to decode the message contents for the first 5 frames. This is to allow your algorithms time to converge.

- You are permitted to discuss the project with other students, but the receiver that you design must be your own. The university code of academic integrity will be strictly enforced (https://www.wpi.edu/about/policies/academic-integrity).

Suggestions

- You should parameterize your design as much as possible (e.g. presumed carrier frequency, presumed baud rate, step sizes for each algorithm, etc.). You are encouraged to put all of the parameters in one place at the top of your code so that they can be easily changed during the debugging and tuning of your receiver.

- Almost all of the parameters your receiver will estimate are time-varying parameters. This includes channel variation, noise variation, oscillator drift, etc. Thus, your receiver will not work well if you attempt to do “block” or “batch” processing. For example, the optimal equalizer coefficients at the beginning of the transmission could be very different from the optimal equalizer coefficients at the end of the transmission.

- You are encouraged to use the transmitter (specify the parameters in `m6param_fdma.m`, which will call `BigTransmitter_fdma.m` to generate test signals) provided in class to test your receiver under a wider range of conditions. Tackle one problem at a step. First design your receiver that can decode `easy.mat`, then use the transmitter provided in the class to add impairment one by one, and add adaptive

elements to handle these impairments. Becoming familiar with the transmitter operation will help you understand the signal composition. Before you seek help from the TA or professor, you should always attempt to debug the problem by using your transmitter to drive your receiver.

- Try to break your receiver. See how much noise can be present in the received signal before accurate demodulation seems impossible (e.g. BER $>10^{-2}$). Find the fastest change in the carrier phase that your receiver can track, even with a bad initial guess. Try to determine how bad the worst channel can be through which a signal can be transmitted where your receiver correctly decodes the signal.

- You may want to implement a "debug" mode in your receiver. Perhaps you want a flag called DEBUG_FLAG, which, when set to 1, plots spectra, time-histories of adaptive parameters, eye diagrams, constellations, squared error, correlation with the marker sequence, etc.

- To increase the speed of your code, you should try to vectorize operations as much as possible. Also, you should never dynamically allocate memory – always allocate space for vectors and matrices before their use with the zeros command. Further tips for speeding up your code can be found in the Matlab documentation (type help desk and search for the string "Optimizing MATLAB Code").

Evaluation

- You should encapsulate your design in one Matlab function `Rx.m` which should have the following declaration: `[decoded text, y] = Rx(r, rolloff, desired user).` The input argument r is the received signal vector as shown in Figure 15.2 of the textbook, the input argument rolloff is the square root raised cosine filter roll-off parameter (β) which will be in the range 0.1 to 0.3, and the input argument desired user is the index of the user (1, 2, or 3) that you need to recover. The output argument decoded text is the decoded message of the desired user with all marker and training data removed, and the output argument y is the output of your equalizer (i.e. the "soft" decisions before the quantizer/slicer). When plotted, y should look something like Fig. 1.

- Your decoded text vector should contain decoded text for the desired user starting from the first preamble you are able to identify onward. However, your grade will only be based on the performance of your receiver after the 5th frame. Again, the decoded text vector should have the preamble and all undesired user data removed.

- Your receiver will be tested on a "mystery signal" created with a given rolloff factor and other parameters within the ranges specified in the modified Chapter 15. Your grade will be determined in the following manner:
  Project Grade = 16×% from easy.mat + 5×% from medium.mat + 4×% from mystery.mat
- The "mystery signal" will have signal impairments with severity somewhere between the `medium.mat` and `hard.mat` test vectors.

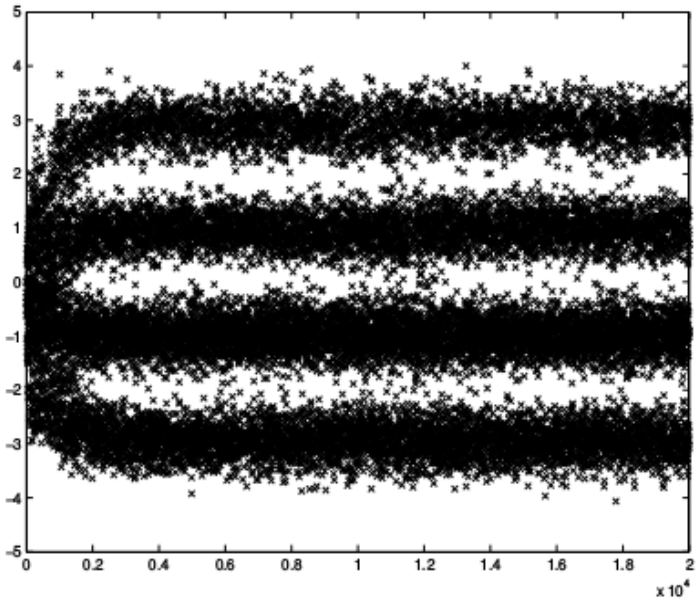

Figure 1: An example of what y might look like

- There will be several sessions on Dec 13 Tuesday and Dec 14 Wednesday for you to test your receiver on the "mystery signal". There will be a sign-up sheet outside of AK 302 for 10 minute slots during these sessions. You are highly encouraged to take advantage of this, because you will then be able to confirm that you will receive an acceptable Grade.

- The script `tester.m` that will be used to test your receiver and determine your grade is posted on the course web page. This will enable you to see how the testing procedure will work, and will allow you to make sure your code will work.

- Your receiver code should be emailed to ksgill@WPI.EDU by Dec 16 by midnight. Late projects will not be accepted. All of your code should be included in a single file `Rx.m`. If you had split your code into several files, you should put them all into `Rx.m` as sub-functions (type "help function" in Matlab for help with subfunctions). The output of your software will be tested for accuracy and a grade will be assigned accordingly.

- In addition to submitting your code, you are also required to turn in a brief report, also due on Dec 16 by midnight. The report should be in .pdf format, and included with your emailed Matlab receiver code. This report should contain a well documented listing of all of your Matlab code comprising your software radio. You should also include a prose description with equations of the operation and pertinent variables of each of the receiver modules, i.e. bandpass filter (cutoff frequencies and transfer function), low-pass filter (cutoff frequency and transfer function), matched filter (transfer function), carrier phase tracking algorithm, baud-timing recovery scheme, equalizer, and quantizer. Your report should also summarize the performance of your receiver after running `tester.m` on the three signals on the course web site.