

# EE 460 / Spring 2019

## Digital Communication Systems

### 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 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 number 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.
- This assignment is to be completed individually. 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 (<http://catalog.wvu.edu/content.php?catoid=11&navoid=2091>).

## Suggestions

- You should parameterize your design as much as possible (e.g. presumed carrier frequency, presumed baud rate, stepsizes 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 provided in class to test your receiver under a wider range of conditions. Becoming familiar with the transmitter operation will help you understand the signal composition. Before you seek help from the instructor, 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.  $\text{BER} > 10^{-2}$ ). 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 `helpdesk` 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 rolloff parameter ( $\beta$ ) which will be in the range 0.1 to 0.3, and the input argument `desired_user` is the user number (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. You will be graded on how accurately your receiver demodulates the signal. The “mystery signal” will have signal impairments with severity somewhere between the `medium.mat` and `hard.mat` test vectors.

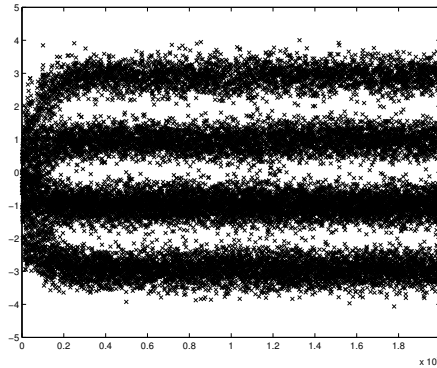


Figure 1: An example of what  $y$  might look like

- You have virtually unlimited opportunity to test your receiver on the “mystery signal” (e.g., in office hours and in lab sessions) up until the last lab session when you will have a single opportunity. You are highly encouraged to plan ahead, 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 submitted through Canvas by **Friday 07-Jun-2019 5:00p**. 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 subfunctions (type “`help function`” in MATLAB for help with subfunctions). You do not need to include any of the MATLAB files available on the course website (e.g., `plotspec`, etc). 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 Friday 07-Jun-2019 5:00p. The report should follow the usual format, should be submitted in .pdf format, and should be sent in a single email along with your MATLAB receiver code as a .m file. 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.
- The “completion multiplier” used in the lab grading rubric will be determined as follows for this lab. It will be the **larger** of the score you receive on the mystery signal, and:

- If easy/med/difficult  $\geq 100/95/70$ , then:  
50% plus,  
15% for a functioning carrier recovery (max, evaluated on mystery),  
15% for a functioning timing recovery (max, evaluated on mystery).
- If easy/med/difficult  $\geq 80/50/0$ , then:  
35% plus,  
15% for carrier (max, on mystery),  
15% for timing (max, on mystery).
- If easy/med/difficult  $\geq 70/0/0$ , then:  
20% plus,  
15% for carrier (max, on mystery),  
15% for timing (max, on mystery).

If your receiver doesn't run with tester.m, or doesn't do better than 70% on easy.mat you will earn a multiplier of 10%.

## Bonus

The student receiver demonstrating best performance on the easy, medium, and hard test vectors will receive a 10% bonus on the lab. In the event of a tie (or multiple students receiving 100% performance on all test vectors), an even more difficult test vector will be used to break the tie.