Collection of EE521 Term Project Written Communications

Summary

E-Mail, February 17, 2006

For your term projects, I am contemplating a series of functions:

a) An input, consisting of a specified test signal and a specified noise level.

b) A sampling operation

c) A quantization to a specified number of bits, followed by translation from bits to symbols.

d) Adding the noise and encoding of a specified type.

e) Decoding to recover the symbols and detection of bit errors.

f) Symbol to bit conversion to recover the sampled PAM signal.

g) Filtering to recover the input signal.

h) Operations to measure the bit error rate of the encoding/decoding operations.

I will give you your assignment in stages and ask you to bring your project to class each week so that I can look at it over the weekends and critique them for you. Each of you will be given separate test signals to use, and other differences, but the base communications link you will be building and simulating will be a simple voice link.

I will begin at the next class by asking that you simplify what you are doing with Problem 2.4 to become the first baseline for your term project. Also, in class, I will explain to you how we will build the term project step-by-step in a simple, structured way that lets you concentrate on one or two tokens at a time, not the whole project.

E-mail, March 8, 2006:

Tonight we will check attendance, and if we may have class on the term projects instead of my prepared lecture on linear block codes. If so, we will have the introduction to linear block codes next week.

Handout, March 22, 2006

This assignment is part of your Quiz 2 and will be 25% of that grade.
Generate a test signal over the speech band, add some noise to it, sample it to 8 bits and convert it to a serial bit stream, then convert it back to PAM and scale to the original scale. Before you begin, block out your system and decide what the SystemView sample rate must be, and set it. Increase the number of samples using the power-of-two spin button until the end time is 1 second or more.

- Use a frequency sweep signal, from 30 Hz to 3000 Hz..
- Add noise, initially at 17 dB SNR.
- Digitize to a serial bit stream with these tokens:
    - Quantize to 8 bits, using the quantizer from the function library. Make sure that the input voltage range accommodates the voltage of the input, plus noise.
    - Use the sampling operator to sample at the engineering Nyquist rate for the signal chosen.
    - Use a custom algebraic function to turn the sampled, quantized output into a Symbol. This will require that the signal be shifted in level and scaled to become an unsigned integer.
    - Use the symbol-to-bit converter from the Comms library to convert the symbol to a bit stream.
- Convert the bit stream back to a symbol using the bit-to-symbol converter from the comms library.
- Use a custom function to re-scale the symbol to the original scale of the signal plus noise.
- Display the output in a system sink.

You will send me your files by e-mail or have them ready for me to load on my laptop when class starts. I will have a USB flash drive and a CD-ROM drive to accomplish this. I don't have a floppy drive, so if you can't bring your laptop with the file on it, or you can't burn a CD-ROM, send the file by e-mail.
You will write a one or two sentence description of each token. This description will tell me

- What the token is -- which token is used from which library.
- What the token does -- an algebraic equation or one or two words..

- What the data format is on the input.
- What the data format is on the output.

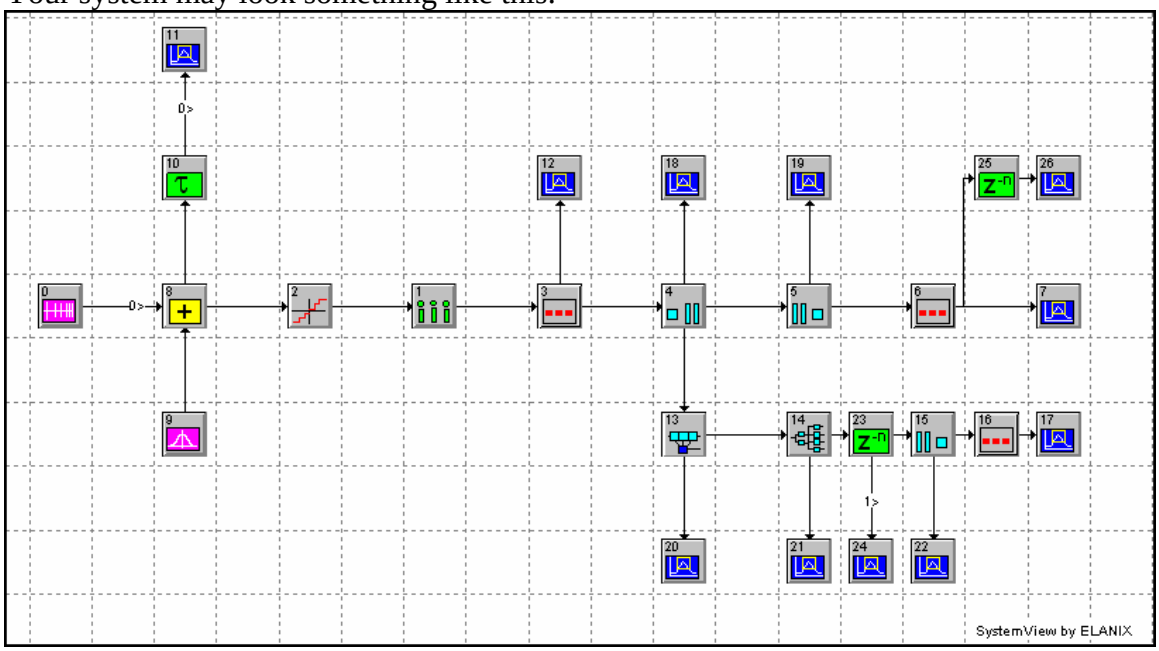
Data format parameters for token inputs and outputs will include

- Signal type -- analog, digital 0-1, digital (-1)-(+1), symbol unsigned integer, etc.
- Voltage range (-1v to +1v, 0 to 255 v, etc.) .
- Sample rate (SystemView sample rate, 6600 Hz, etc.) .

I'll entertain questions by e-mail. You can send me your file and ask what's wrong just as always.

Handout, April 5, 2006

Add a simple convolutional code-decode to your Term Project baseline. Use the default convolutional code in the Communications Encode/Decode collection. The convolutional encoder is labeled "Cnv Coder" and the decoder is labeled "Cnv dCode." Your system may look something like this:



Note that there is a delay, Token 23, that is required for the bits-to-symbol converter to work. The function of this delay is to align the 8-bit sequences in the bit stream out of the decoder with what is required by the bit-to-symbol decoder, which simply counts from the first symbol.

We have added a second output from the original data, token 26, and added a delay, token 25, so that this output is time-aligned with the output from the convolutional encoder-decoder path, token 17. You will need this to determine when bit errors occur in the next step.

When this is complete, we will add noise to the input to the convolutional decoder, token 14, and measure the bit error rate (BER). For information on BER, see the SystemView online documentation of the BER token in the Communications library, Processors tab. I

will provide input on how you may approach BER computation. You will be required to find one BER for a specific single SNR in your demo. If you choose to run a BER curve, and the specified BER is within the limits of your curve, that is sufficient, but not required. Some examples, rice_tst.svu, blk_tst.svu, golay.svu, and cnvl.svu are available in your SystemView Program Documents folder. Also, as is a more elaborate example, BER_9pde3_srx32_RRC_ud_conv.svu is also there. There is an excellent but complex application note, AN-107, on computing BER curves using SystemView.

Handout, April 12, 2006

Add noise to your Term Project baseline.

E-Mail, April 16, 2006

The test signal in the term project is a frequency sweep sine wave with added noise. This added noise is part of the test signal and has absolutely nothing to do with BER at the receiver.

The next step is the digitization of the signal. The input signal is scaled, sampled, quantized, and converted to a bit stream.

The next step is convolutional encoding. This year we all will use the rate 1/2, K=3 convolutional code, MSB first -- the default for the SystemView convolutional encoder and decoder tokens.

For BER measurement, we need to add noise at this point. The amount of noise is determined by you to simulate an Eb/N0 of 5 dB. To prevent errors in the system at this point that cause an erroneous bit stream to be present hereafter, the use must make sure that the noise that is added here is at the same sample rate as the data out of the convolutional encoder.

The next step is convolutional decoding. Use the default parameters in the SystemView convolutional decoder block -- rate 1/2, K=3, MSB first.

The convolutional decoder then provides signals for the conversion back to analog, and waveform matching to the input. This is what you did for the first quiz.

The inputs to the BER decoder are digital, and the threshold setup parameter has a default of 1/2, which means that input binary streams of zeros and ones (as opposed to -1's and +1's) are for use with this threshold. The two inputs are the output of the convolutional decoder and the input to the convolutional encoder, delayed by a number of samples that synchronizes the delayed bitstream to synchronize it with the decoded bit stream. Part of the problem for your term project is to synchronize these bit streams (i.e. to determine the delay required). Also, how to use the outputs to find BER is part of the term project. The documentation and examples that were given in class and appear on your handouts for the last two weeks provide this information. Before you go farther, you should make sure that you are getting an output and can synchronize it with the input, and that issues like

clipping the input signal are addressed.

Additional things you should investigate are the setup delay in the BER counter (the number of samples to wait before input data is available from both inputs) and the delay setups in the encoder and decoder tokens. Look at the documentation and examples for these tokens. The online help names the example files that are on your hard disk.

Additional things that you may want to do, but which I don't expect of all of you for next class, are to use a stop sink and multiple system loops to compute a BER curve, and use of the delay setup parameters on the outputs of the convolutional encoder and decoder.

Handout, April 19, 2006

Add noise to your Term Project baseline. Term Projects are due next time. Bring your file on a memory stick, or e-mail it to Dr. Beard by 3:00 PM. A report is also due. Be prepared to run your file, walk the class through its operation, discuss its inputs and outputs, and answer questions from the class and from the Instructor.