EE 567

Project

Due Tuesday, December 3, 2019 at 6:40 p.m.

Work all 3 Parts.

Instructions.

Your project should be typed on one side of the paper only and stapled in the upper left hand corner. You should include a cover page and an appendix where you include your Matlab code. Do not place your project inside any kind of binder. This is to be an individual effort. You may consult any written material (hard or soft copy) but you may not solicit input from any person except that you may ask the professor or TA questions regarding your project. Your project report should be self-contained, that is, the reader should be able to understand the problems and your solutions without consulting the actual project assignment.

Part 1.

Assume we have downconverted a received signal via a mixing operation and now we wish to apply a LPF. One way to implement a LPF is simply to compute an average. In continuous time we would just integrate the signal since dividing by the integration time T to compute the average would not affect the performance since the noise would be scaled by the same amount as the signal component. In discrete time we would implement a sum instead of an integral. For this part we will assume we have a discrete time signal but we will compute an average instead of just a sum.

So let us assume the input to the LPF is a signal of the form

$$s(k) = \sqrt{E} + \text{ double frequency terms } + n(k), \quad k = 0, 1, \ldots$$

where we have assumed scaling so that $n(k)$ is a standard normal random variable for each k and n_i is independent of n_j for $i \neq j$, $i, j = 0, 1, \ldots$. We may ignore the double frequency terms and assume they are suppressed, either completely or at least sufficiently, by the LPF. The output of the LPF

is

$$y(n) = \frac{1}{N} \sum_{k=n-N+1}^{n} s(k), \quad n = 0, 1 \ldots$$

where we take $y(n) = 0$ for $n < 0$.

Even though we are computing an average we will refer to this type of filter as an *integrate and dump* or I&D filter.

Now for implementation purposes we can also construct a LPF using an IIR filter. Let

$$\tilde{y}(n) = (1 - \alpha)s(n) + \alpha\tilde{y}(n - 1), \quad n = 0, 1 \ldots$$

where we take $\tilde{y}(-1) = 0$.

a. Determine (analytically) the value of $\alpha = \alpha(N)$ so that the mean and variance of the IIR filter output matches the mean and variance of the I&D filter output as $n \to \infty$. For this task you may assume without loss of generality that you only have noise present.

b. Compute (analytically) the impulse response of the I&D filter.

c. Compute (analytically) the step response of the I&D filter.

d. Compute (analytically) the impulse response of the IIR filter using the value of α found in part (a).

e. Compute (analytically) the step response of the IIR filter using the value of α found in part (a).

f. Plot the impulse response for each filter on the same graph using $N = 8$.

g. Plot the step response for each filter on the same graph using $N = 8$.

Part 2.

Assume we have downconverted a BPSK signal such that the input to a LPF is of the form

$$s(k) = A + \text{ double frequency terms } + n(k)$$

where A is a constant and $n(k),\ k = 0, 1, \ldots$ are independent zero-mean normal random variables with variance σ^2. We may ignore the double frequency terms and assume they are suppressed, either completely or at least sufficiently, by the LPF. Suppose the LPF is an integrator of the form

$$y = \frac{1}{N} \sum_{k=0}^{N-1} s(k).$$

After applying this LPF the output y is a test statistic with bit-energy-to-noise ratio E_b/N_0. We know that for BPSK the probability that we make a wrong decision using this test statistic is

$$P_b = Q\left(\sqrt{\frac{2E_b}{N_0}}\right).$$

We can approximate this LPF operation using an IIR filter of the form

$$\tilde{y}(n) = (1-\alpha)s(n) + \alpha\tilde{y}(n-1), \quad n = 0, 1 \ldots N-1$$

where we take $\tilde{y}(-1) = 0$.

a. Using Matlab simulate the signal $s(k)$ and the LPF operation producing y above for $N = 8$ and plot your simulated bit error rate vs. E_b/N_0 on the same graph as P_b vs. E_b/N_0 to compare. Your E_b/N_0 range should be large enough so that the BER ranges from 0.5 to 10^{-6}.

b. Using Matlab simulate the signal $s(k)$ and the IIR filter operation producing $\tilde{y}(n)$ above for $N = 8$ and plot your simulated bit error rate vs. E_b/N_0 on the same graph as the simulated bit error rate for y to compare. You should use the value of α that you found in Part 1 of this project. Your E_b/N_0 range should be large enough so that the BER ranges from 0.5 to 10^{-6}.

c. The value of α that you just used does not necessarily minimize the probability of bit error for the IIR filter approach. Using Matlab and some trial and error find the value of α that does minimize the probability of bit error for $E_b/N_0 = 7$ dB. Using this new α (if it is different than that found in Part 1), plot your simulated bit error rate vs. E_b/N_0 on the same graph as the simulated bit error rate for y and the simulated bit error for the α found in Part 1 to compare. Your E_b/N_0 range should be large enough so that the BER ranges from 0.5 to 10^{-6}.

Part 3.

In this part we are going to investigate the detection of signals in noise. We will consider both integration detection and M of N logic detection.

Suppose we receive a signal of the form

$$r(t) = A\cos(2\pi f_c t + \phi) + n(t), \quad 0 \leq t \leq T$$

where A is a constant over T seconds taking on the value of $A = 1$ or $A = 0$ where in the latter case we have only noise present, $f_c = 1$ MHz, $T = 1$ msec and $n(t)$ is a Gaussian random variable at time t with mean 0 and variance σ^2.

a. Simulate the direct integration approach to signal detection using the square law detector as covered in class (this detector yielded a chi-square random variable with 2 degrees of freedom). Assume there is no post detection integration. In your sims use a threshold required for a probability of false alarm of 10^{-4}. You should find this threshold analytically. You can assume that just prior to squaring and adding the downconverted and filtered (integrated over T seconds) received waveform has the form (after scaling)

$$r_1(T) = A\cos(\phi) + n_1$$

in the upper path of the circuit and

$$r_2(T) = A\sin(\phi) + n_2$$

in the lower path of the circuit, where, n_1 and n_2 are independent Gaussian random variables with mean 0 and variance σ^2. The output of the detector is then

$$z(T) = r_1^2 + r_2^2.$$

Plot probability of detection results for SNR ranging from 0 to 15 dB. The y-axis of your plot should use a log scale and your x-axis should be in dB.

b. Repeat part (a) but now assume we also include a post detector integrator of length $N = 16$. Now our overall probability of false alarm after post detection integration is 10^{-4} so a new threshold will need to be found. You may find the new threshold to control false alarms via simulation or numerically evaluating the appropriate integrals.

c. Repeat part (b) but now assume the post detecting integration is replaced with an M of N logic detector where $N = 16$ and $M = 8$. Our overall probability of false alarm after the M of N logic is still 10^{-4}. You will need to find a new threshold to use here right before the M of N logic.