Dennis Gavrilov, Miraj Patel, Yacine Manseur
4/6/15
Professor Keene
ECE 302

Estimation Exercises 1

Scenario 1:

Consider the system $\boldsymbol{X} = h\boldsymbol{\theta} + \boldsymbol{V}$, where $\boldsymbol{V}$ is Gaussian with zero-mean and variance σ^2, and h is a known parameter. For finding the Bayes MMSE estimate of $\boldsymbol{\theta}$, we assume $\boldsymbol{\theta}$ is a Gaussian random variable with mean μ and variance σ_0^2. By solving for the mean of **X**, we find the following:

$$\mu_X = h \times \mu_\theta$$
$$\sigma_X{}^2 = h^2\sigma_\theta{}^2 + \sigma^2$$

We can easily arrange these two functions to get them in terms of the mean and variance of $\boldsymbol{\theta}$. This means if we estimate $\boldsymbol{X}$, we can easily alter the estimation to be an estimator of $\boldsymbol{\theta}$. We know that the MMSE is generally defined as $E[\mu_{\boldsymbol{X}}, \sigma_{\boldsymbol{X}}{}^2|\boldsymbol{X}]$. This means we need to find the pmf of $\boldsymbol{\mu_X}$ and $\boldsymbol{\sigma_X}^{\mathbf{2}}$ given $\boldsymbol{X}$, also known as the posterior. We need to take note that the mean of **X** updates every time a new observation of $\boldsymbol{X}$ is obtained. Using Baye's Rule, we can define the posterior pmf as the following:

$$p(\mu, \sigma^2|\boldsymbol{X}) \propto p(\boldsymbol{X}|\mu, \sigma^2)p(\mu, \sigma^2)$$

For simplicity, we rewrite the variance as the precision, defining that $\tau = \frac{1}{\sigma^2}$. Since $\boldsymbol{\theta}$ is a Gaussian random variable with mean μ and variance σ_0^2, we redefine the Likelihood function as follows:

$$p(\boldsymbol{X}|\mu,\tau) = \prod_{i=1}^{N}\left[\sqrt{\frac{\tau}{2\pi}}\, exp\left[-\frac{\tau}{2}(x_i-\mu)^2\right]\right] = \left(\sqrt{\frac{\tau}{2\pi}}\right)^N\ exp\left[-\frac{\tau}{2}\sum_{i=1}^{N}(x_i-\mu)^2\right]$$

We can simplify this function further using the following formula for the sum of differences from the mean where $\bar{x}$ is the sample mean:

$$\sum_{i=1}^{N}(x_i-\mu)^2 = \sum_{i=1}^{N}(x_i-\bar{x})^2 + N(\bar{x}-\mu)^2$$

The simplified likelihood function is now the following:

$$p(\boldsymbol{X}|\mu,\tau) = \left(\sqrt{\frac{\tau}{2\pi}}\right)^N\ exp\left[-\frac{\tau}{2}\sum_{i=1}^{N}(x_i-\bar{x})^2 + N(\bar{x}-\mu)^2\right]$$

Since the variance is known, we are able to define the pmf of μ as the following:

$$p(\mu) = \sqrt{\frac{\tau_0}{2\pi}}\ exp\left[-\frac{\tau_0}{2}(\mu-\mu_0)^2\right]$$

Note that μ_0 and τ_0 relate to the actual mean and precision of $\boldsymbol{\theta}$. Following the relation of the posterior pmf using Baye's rule, we solve for the relation of the posterior pmf as follows:

$$p(\boldsymbol{X}|\mu)p(\mu,\sigma^2) = \left(\sqrt{\frac{\tau}{2\pi}}\right)^N exp\left[-\frac{\tau}{2}\sum_{i=1}^{N}(x_i-\bar{x})^2 + N(\bar{x}-\mu)^2\right]\sqrt{\frac{\tau_0}{2\pi}}\ exp\left[-\frac{\tau_0}{2}(\mu-\mu_0)^2\right]$$

Since we are dealing with a relation, we are able to omit constants and simplify what we have into hopefully an exponent resembling a Gaussian distribution. Through simplification, extensive factoring, and using a formula for the sum of two quadratics, we are able to eliminate all constant factors that don't involve μ and get the following:

$$p(\mu|\boldsymbol{X}) \propto exp\left[-\frac{1}{2}(N\tau+\tau_0)\left(\mu-\frac{N\tau\bar{x}+\tau_0\mu_0}{N\tau+\tau_0}\right)^2\right]$$

Since this pmf is in the form of a Gaussian distribution, we now have the relation that

$$p(\mu|\boldsymbol{X})\sim N\left(\frac{N\tau\bar{x}+\tau_0\mu_0}{N\tau+\tau_0},\frac{1}{N\tau+\tau_0}\right)$$

This means that the Bayes MMSE estimate of $\boldsymbol{\theta}$ is the following:

$$\hat{\theta} = \frac{1}{h}\times E[\mu_X|\boldsymbol{X}] = \frac{1}{h}\times \widehat{\mu_{\mu|X}} = \frac{1}{h}\times\frac{N\tau\bar{x}+\tau_0\mu_0}{N\tau+\tau_0}$$

For finding the Maximum Likelihood Estimate of $\boldsymbol{\theta}$, we assume $\boldsymbol{\theta}$ to be deterministic in this case. Since $\boldsymbol{\theta}$ is deterministic, its mean is $\boldsymbol{\theta}$ and its variance is 0. By solving for the mean of $\boldsymbol{X}$ with $\boldsymbol{\theta}$ being deterministic, we find the following:

$$\mu_X = h \times \theta$$

This means that the Maximum Likelihood Estimate of $\boldsymbol{\theta}$ is the Maximum Likelihood estimate of the mean of $\boldsymbol{X}$ divided by the factor h. To find the Maximum Likelihood Estimate of **X** we maximize the log-likelihood function. The likelihood function is found as follows:

$$L(\boldsymbol{X}|\mu,\sigma^2) = \prod_{i=1}^{N} p_X(x_i|\mu,\sigma^2) = \left(\frac{1}{2\pi\sigma^2}\right)^{\frac{N}{2}} exp\left[\frac{-1}{2\sigma^2}\sum_{i=1}^{N}(x_i-\mu)^2\right]$$

The log of the likelihood function is as follows:

$$\ln[L(\boldsymbol{X}|\mu,\sigma^2)] = -\frac{N}{2}\ln(2\pi) - \frac{N}{2}\ln(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(x_i-\mu)^2$$

We then maximize the log-likelihood function by taking a partial derivative with respect to μ and σ^2 and setting the equations to zero:

$$\frac{\partial}{\partial\mu}(\ln[L(\boldsymbol{X}|\mu,\sigma^2)]) = -\frac{1}{\sigma^2}\sum_{i=1}^{N}(x_i-\mu) = -\frac{1}{\sigma^2}\sum_{i=1}^{N}(x_i) - \frac{N}{\sigma^2}\hat{\mu} = 0$$

$$\frac{\partial}{\partial\sigma^2}(\ln[L(\boldsymbol{X}|\mu,\sigma^2)]) = -\frac{N}{2\widehat{\sigma^2}} + \frac{1}{2(\widehat{\sigma^2})^2}\sum_{i=1}^{N}(x_i-\mu)^2 = 0$$

Solving for $\hat{\mu}$ and $\widehat{\sigma^2}$ gives us the following:

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N}(x_i)$$

$$\widehat{\sigma^2} = \frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^2$$

This means that the Maximum Likelihood estimate for $\boldsymbol{\theta}$ is as follows:

$$\hat{\theta} = \frac{1}{N \times h}\sum_{i=1}^{N}(x_i)$$

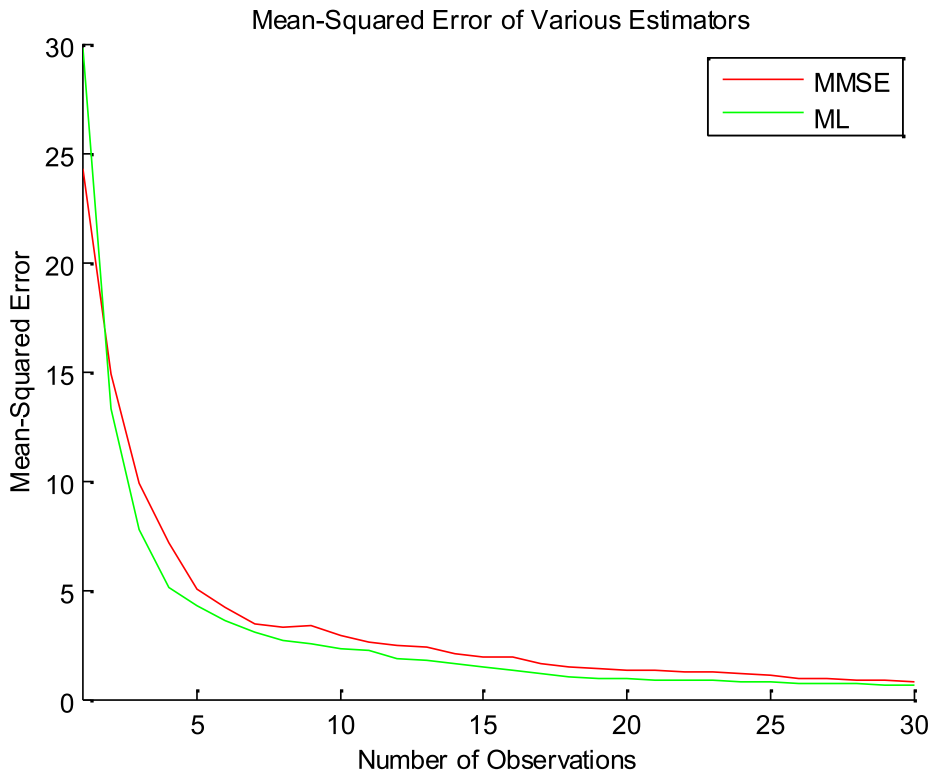
The Mean-Squared error of an estimator is defined as the mean of the difference between the estimator and what is estimated squared. For the Maximum Likelihood Estimator, the MSE is as follows:

$$E\left[\left(\theta - \hat{\theta}\right)^2\right] = E\left[\left(\theta - \frac{1}{N \times h}\sum_{i=1}^{N}(x_i)\right)^2\right]$$

For the Bayes MMSE Estimator, the MSE is as follows:

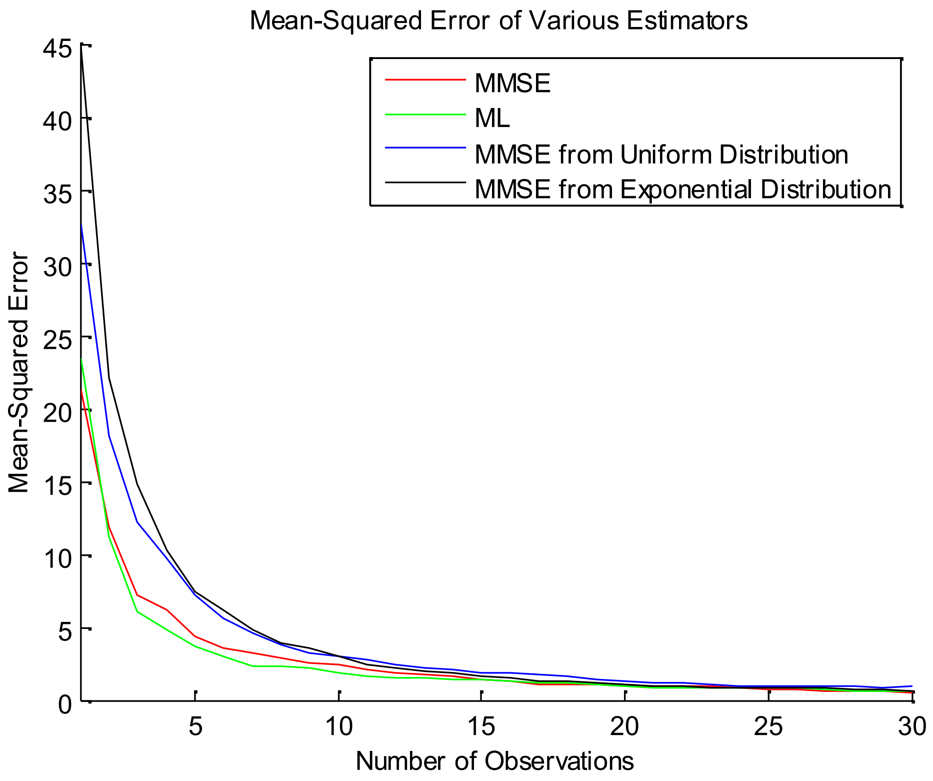
$$E\left[\left(\theta - \hat{\theta}\right)^2\right] = E\left[\left(\theta - \frac{1}{h} \times \frac{N\tau\bar{x} + \tau_0\mu_0}{N\tau + \tau_0}\right)^2\right]$$

A simulation was run in MATLAB to implement these Estimators and compare the Mean-Squared error of each estimator as the number of measurements increased. The result of this simulation is shown below:



From the graph shown above, we see that the MMSE estimator out performs the ML estimator at a low number of observations. This is expected because the MMSE is based off of the prior knowledge of $\boldsymbol{\theta}$. When playing around with initial values, we are able to gain an understanding of how the estimators work and what cases work best. When the mean of $\boldsymbol{\theta}$ is below seven, the MMSE estimator is able to estimate $\boldsymbol{\theta}$ better than the ML estimator. When the mean of $\boldsymbol{\theta}$ increases to above twenty, we see that the ML estimator outperforms the MMSE by a large factor. This makes sense because the larger the mean of $\boldsymbol{\theta}$, the less it is affected by the added Gaussian white noise. Since the Gaussian white noise only has a minimal influence on the calculated $\boldsymbol{X}$, the sample mean is a much better representation of $\boldsymbol{\theta}$ than what the MMSE estimator returns. When the variance of the noise is larger, the MMSE estimator works as a better estimate for $\boldsymbol{\theta}$. The lower the variance of the Gaussian noise is, the better the ML estimator performs. This is also understandable since the sample mean is the best way to estimate a Gaussian random variable.

We also used the MMSE Estimator for two more cases where the posterior probability was taken from a different distribution. For one case, we found $\boldsymbol{\theta}$ from a uniform distribution centered at its initial mean. For the second case, we found $\boldsymbol{\theta}$ using an exponential distribution with λ equal to the initial mean of $\boldsymbol{\theta}$. We compare the MSE of all four estimators shown below:



When comparing these new MMSE estimators to the MMSE estimator using the correct posterior, we see that the incorrect estimators perform horribly at a low number of observations. This is expected since the Bayes MMSE estimator we found assumed that $\boldsymbol{\theta}$ was Gaussian. An interesting finding is that even though the prior is from a different distribution, the Mean-Squared error still converges to that of the ML estimate for all cases. This is because as the number of observations increases, we find the MMSE estimate of $\boldsymbol{\theta}$ to become:

$$\lim_{N\to\infty} \frac{1}{h} \times \frac{N\tau\bar{x} + \tau_0\mu_0}{N\tau + \tau_0} = \frac{\bar{x}}{h}$$

As the number of observations approaches infinity, we see that the MMSE estimator is the same as the ML estimator, and it is independent of the type of distribution that $\boldsymbol{\theta}$ is drawn from.

Scenario 2: Dealing With Interference

This scenario introduces the problem of determining the time interval during which two signals overlap each other in a channel with additive white Gaussian noise (AWGN). The desired and interfering signals in this scenario are both BPSK modulated, in phase, and of equal power, but the interfering signal only lasts from time t_1 to t_2. To determine t_1 and t_2, given only that $t_2 > t_1$, a Maximum Likelihood estimator must be applied to the received data set.

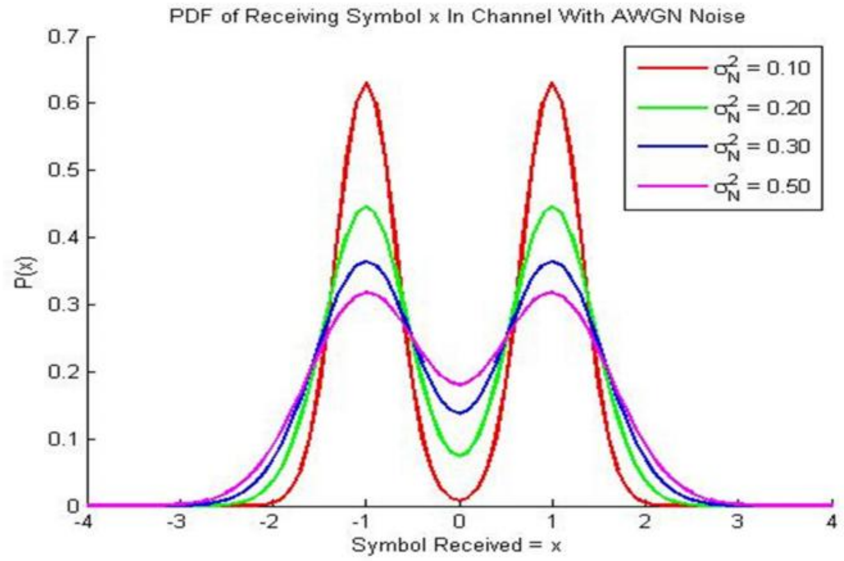
Mathematically Modeling The System

In order to perform statistical testing on this system, the distribution of the received signal, Rx, must be thoroughly defined. However, the received signal of this system is actually a time dependent composition of mixed Gaussian random signals due to the nature of the underlying zero-mean Gaussian noise that affects each transmitted BPSK symbol.
The first region of operation is defined as the time interval for $t < t_1$, in the beginning of the transmission when there is no interference, but just noise. Without noise, BPSK modulated symbols, either transmitted as "-1" or "+1" occur with 50% probability, as each are assumed equally likely. However, the presences of additive white Gaussian noise adds a whole spectrum of values to each "-1" and "+1" (dependent on the variance of the noise). Thus, the effective probability density function of the received signal is a weighted sum of two Gaussian functions centered on each different BPSK symbol:

$$P(x) = \frac{1}{2} * N(+1, \sigma^2) + \frac{1}{2} * N(-1, \sigma^2)$$

$$P(x) = \frac{1}{2} * \left(\frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-1)^2}{2\sigma^2}}\right) + \frac{1}{2} * \left(\frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x+1)^2}{2\sigma^2}}\right)$$

This region of operation is denoted as Region A, and the noise-variance dependent pdf functions are shown below:

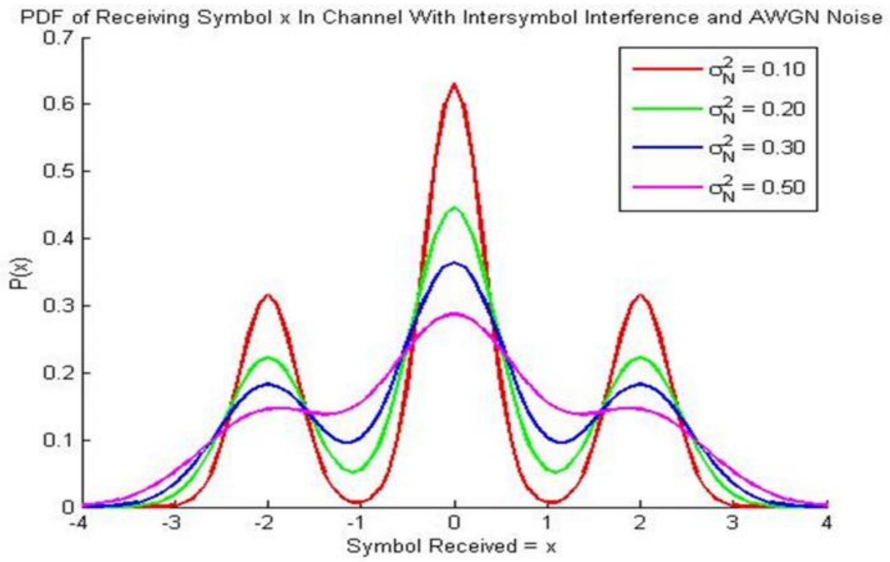


Once the interfering BPSK signal begins its transmission from time t_1 to t_2, the distribution of possible received symbols significantly changes. During this time interval, the transmitted symbol "-1" and "+1" in a sense constructively or destructively interfere, resulting in a new sample space of possible symbols: {-2, 0, +2}. The "-2" and "+2" both occur with probability of 0.25, and the "0" occurs with probability 0.50, and each servers as a weight to an underlying Gaussian curve due to the additive noise.

$$P(x) = \frac{1}{4} * N(-2, \sigma^2) + \frac{1}{4} * N(+2, \sigma^2) + \frac{1}{2} * N(0, \sigma^2)$$

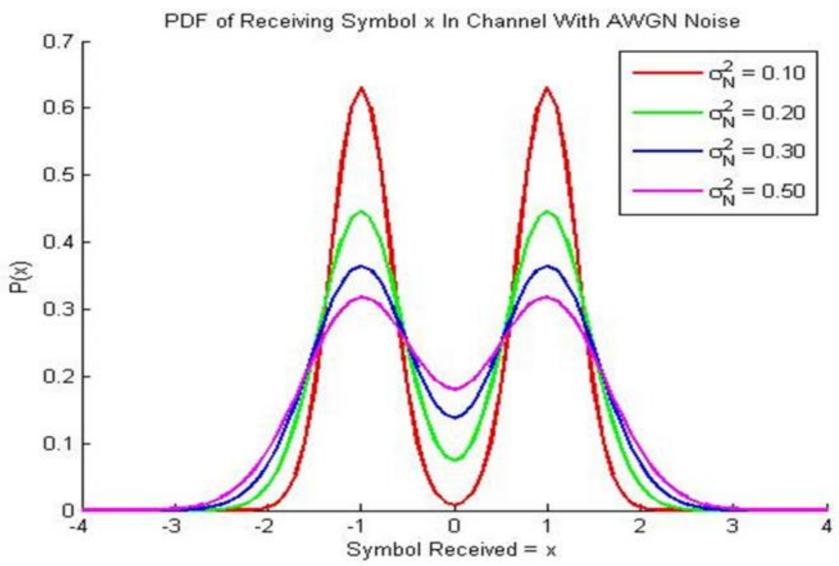
$$P(x) = \frac{1}{4} * \left(\frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x+2)^2}{2\sigma^2}}\right) + \frac{1}{4} * \left(\frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-2)^2}{2\sigma^2}}\right) + \frac{1}{2} * \left(\frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-0)^2}{2\sigma^2}}\right)$$

This time interval, or region of operation, is denoted as Region B, and the figure below shows four sample noise-variance dependent pdfs of the resulting received signal:



When the interfering BPSK signal ceases to transmit in the same channel as the desired BPSK signal, the final region of operation (Region C) experiences the same behavior as derived in Region A – where either BPSK signal, {-1, +1}, experiences the additive behavior of the Gaussian noise. The pdf function and sample variance-dependent distribution curves as shown again for convenience:

$$P(x) = \frac{1}{2} * \left(\frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-1)^2}{2\sigma^2}}\right) + \frac{1}{2} * \left(\frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x+1)^2}{2\sigma^2}}\right)$$



Obtaining The Maximum Likelihood Estimators for t1 and t2

An analytic method to solve for the maximum likelihood estimator for t_1 and t_2 given a received signal data set, if possible, would be too difficult to prove. Instead, a brute-force exhaustive search is performed by computing the likelihood of receiving the given data set for all values of t_1 and t_2 (for $t_2 > t_1$). The pair of t_1 and t_2 values that results in the highest probability of producing the received signal is chosen as the experimental solution and compared to the predefined t_1 and t_2 when created the desired and interfering signals. However, after initial simulation, computing the probability of sample at time *t* and multiplying it forward with the probability of the next sample is too computationally expensive. Thus, the log-likelihood was computed which turned the accumulated product into an accumulated sum of the log of the probability, which is much less computationally intensive. Refer to Appendix B for the MATLAB code that simulates and computes the log probability of obtaining Rx(t) from each possible piece wise mixed Gaussian distribution dependent on t_1' and t_2'.

Results

Isolating the max of all the log likelihood probabilities lets us backtrack to specify which (t_1', t_2') pair resulted to the maximum probability; this (t_1', t_2') pair is then compared to the t_1 and t_2 and a conclusion on the accuracy of the results is drawn.

Number of Iterations	Noise Variance	Actual *t1*	Actual *t2*	Avg. *t1'*	Var(*t1'*)	Avg. *t2'*	Var(*t2'*)
100	0.10	15	75	15.01	0.475	74.99	0.515
100	0.05	15	75	15.00	0.040	75.00	0.040

The two summarized simulations above show that the Maximum Likelihood estimator was able to find the actual values of *t1* and *t2* with high accuracy. The slight variation in the values concluded by each iteration is due to the variance of the noise that affects each symbol. Decreasing the variance of the noise and running the simulation again shows that the variance of the noise, which affects the Gaussian each symbol rides on, significantly helped reduce the variation in the results of the Maximum Likelihood test, most likely due to the fact that the dampened effect of the noise reduced its impact on the symbol received, which in turn lessened the probability of being interpreted as a different signal entirely corrupting the desired signal. Nonetheless, the Maximum Likelihood was quite effective in concluding the *t1* and *t2* parameters of the simulation in situations with higher noise variance, up to a point.

Appendix A: MATLAB Simulation Code for Scenario 1

```
%%  Scenario 1
%   Dennis Gavrilov, Miraj Patel, and Yacine Manseur
%   x = h*theta + v

close all;
clear all;
clc;

%% Initialization...

observations = 30;
iterations = 100;

h = 0.5;

meanTheta = 7;
varTheta = 4;
stdDevTheta = sqrt(varTheta);

meanNoise = 0;
varNoise = 6;
stdDevNoise = sqrt(varNoise);

sampleMeanX = 0;

% Arbitrary initial mean and precision of Theta
mu0 = meanTheta;
T0 = 1/varTheta;
T = 1/varNoise;

% Initiallize x
x = zeros(1,observations);

% Initialize matrices for MSE
MSEML = zeros(iterations,observations);
MSEBayesMMSE = zeros(iterations,observations);
MSEBayesMMSEIncorrectPriorUniform = zeros(iterations,observations);
MSEBayesMMSEIncorrectPriorExponential = zeros(iterations,observations);

%% Creating the Estimators and finding the error associated with each one

% This loop creates the ML Estimator where Theta is deterministic
for i = 1:iterations
    theta = meanTheta;
    for n = 1:observations
        v = normrnd(meanNoise,stdDevNoise); % gaussian noise
        x(n) = h*theta + v;
        sampleMeanX = (1/n)*sum(x);
        MSEML(i,n) = (theta - (1/h * sampleMeanX))^2;
    end
    % reset the x vector
    x = zeros(1,observations);
```

```
end

% This loop creates the Bayes MMSE Estimator with Theta being Gaussian
for i = 1:iterations
    theta = normrnd(meanTheta,stdDevTheta);
    for n = 1:observations
        v = normrnd(meanNoise,stdDevNoise); % gaussian noise
        x(n) = h*theta + v;
        sampleMeanX = (1/n)*sum(x);
        MSEBayesMMSE(i,n) = (theta-
1/h*((n*T*sampleMeanX+T0*mu0)/(n*T+T0)))^2;
    end
    % reset the x vector
    x = zeros(1,observations);
end

% This loop creates the MMSE Estimator with Theta from a Uniform
% Distribution
for i = 1:iterations
    % Create an incorrect theta
    theta = 2*meanTheta*rand(1);
    for n = 1:observations
        v = normrnd(meanNoise,stdDevNoise); % gaussian noise
        x(n) = h*theta + v;
        sampleMeanX = (1/n)*sum(x);
        MSEBayesMMSEIncorrectPriorUniform(i,n) = (theta-
1/h*((n*T*sampleMeanX+T0*mu0)/(n*T+T0)))^2;
    end
    % reset the x vector
    x = zeros(1,observations);
end

% This loop creates the MMSE Estimator with Theta from an Exponential
% Distribution
for i = 1:iterations
    % Create an incorrect theta
    theta = exprnd(meanTheta);
    for n = 1:observations
        v = normrnd(meanNoise,stdDevNoise); % gaussian noise
        x(n) = h*theta + v;
        sampleMeanX = (1/n)*sum(x);
        MSEBayesMMSEIncorrectPriorExponential(i,n) = (theta-
1/h*((n*T*sampleMeanX+T0*mu0)/(n*T+T0)))^2;
    end
    % reset the x vector
    x = zeros(1,observations);
end

%% Plotting the results

figure('name', 'Mean-Squared Error of Various Estimators');

hold on;
plot(mean(MSEBayesMMSE), 'r');
plot(mean(MSEML), 'g');
plot(mean(MSEBayesMMSEIncorrectPriorUniform), 'b');
plot(mean(MSEBayesMMSEIncorrectPriorExponential), 'k');
xlim([1 observations]);
title('Mean-Squared Error of Various Estimators');
xlabel('Number of Observations');
ylabel('Mean-Squared Error');
legend('MMSE', 'ML', 'MMSE from Uniform Distribution', 'MMSE from Exponential
Distribution');
hold off;
```

Appendix B: MATLAB Simulation Code for Scenario 2

```
% BPSK signals represented by either -1 or 1
% Simulation Parameters defined here
N = 100;    % Number of Symbols
Niter = 10; % Number of trials to see how accurately the ML estimator finds
t1 and t2
t1 = 15;    % right when interference starts
t2 = 75;    % right after when interference stops
noiseVariance = 0.10;   % Variance of the zero-mean AWGN

% Using t1 to represent when intersymbol interference occurs, and t2 for
% when it stops, 3 time intervals for the transmission are defined:
% Region A is defined as time: t < t1
% Region B is defined as time: t1 <= t < t2
% Region C is defined as time: t >= t2

% The Regions A and C represent time intervals where there is no
% intersymbol interference due to the 2nd signal, so the pdf is:
figure;
varNoise = 0.10; % The variance of the AWGN
AandCregion = 0.5*normpdf([-4:.1:4], 1, sqrt(varNoise)) + 0.5*normpdf([-
4:.1:4], -1, sqrt(varNoise));
varNoise = 0.20;
AandCregion2 = 0.5*normpdf([-4:.1:4], 1, sqrt(varNoise)) + 0.5*normpdf([-
4:.1:4], -1, sqrt(varNoise));
varNoise = 0.30;
AandCregion3 = 0.5*normpdf([-4:.1:4], 1, sqrt(varNoise)) + 0.5*normpdf([-
4:.1:4], -1, sqrt(varNoise));
varNoise = 0.50;
AandCregion4 = 0.5*normpdf([-4:.1:4], 1, sqrt(varNoise)) + 0.5*normpdf([-
4:.1:4], -1, sqrt(varNoise));

hold on;
plot([-4:.1:4], AandCregion, 'r', 'LineWidth', 2);
plot([-4:.1:4], AandCregion2, 'g', 'LineWidth', 2);
plot([-4:.1:4], AandCregion3, 'b', 'LineWidth', 2);
plot([-4:.1:4], AandCregion4, 'm', 'LineWidth', 2);
hold off;

title('PDF of Receiving Symbol x In Channel With AWGN Noise');
xlabel('Symbol Received = x');
ylabel('P(x)');
legend('\sigma^{2}_{N} = 0.10', '\sigma^{2}_{N} = 0.20', '\sigma^{2}_{N} =
0.30', '\sigma^{2}_{N} = 0.50');



%%
% Region B's pdf:
figure;
varNoise = 0.10;
Bregion = 0.25*normpdf([-4:.1:4], 2, sqrt(varNoise)) + 0.25*normpdf([-
4:.1:4], -2, sqrt(varNoise)) + 0.5*normpdf([-4:.1:4], 0, sqrt(varNoise));
varNoise = 0.20;
```

```
Bregion2 = 0.25*normpdf([-4:.1:4], 2, sqrt(varNoise)) + 0.25*normpdf([-
4:.1:4], -2, sqrt(varNoise)) + 0.5*normpdf([-4:.1:4], 0, sqrt(varNoise));
varNoise = 0.30;
Bregion3 = 0.25*normpdf([-4:.1:4], 2, sqrt(varNoise)) + 0.25*normpdf([-
4:.1:4], -2, sqrt(varNoise)) + 0.5*normpdf([-4:.1:4], 0, sqrt(varNoise));
varNoise = 0.50;
Bregion4 = 0.25*normpdf([-4:.1:4], 2, sqrt(varNoise)) + 0.25*normpdf([-
4:.1:4], -2, sqrt(varNoise)) + 0.5*normpdf([-4:.1:4], 0, sqrt(varNoise));

hold on;
plot([-4:.1:4], Bregion, 'r', 'LineWidth', 2);
plot([-4:.1:4], Bregion2, 'g', 'LineWidth', 2);
plot([-4:.1:4], Bregion3, 'b', 'LineWidth', 2);
plot([-4:.1:4], Bregion4, 'm', 'LineWidth', 2);
hold off;

title('PDF of Receiving Symbol x In Channel With Intersymbol Interference and
AWGN Noise');
xlabel('Symbol Received = x');
ylabel('P(x)');
legend('\sigma^{2}_{N} = 0.10', '\sigma^{2}_{N} = 0.20', '\sigma^{2}_{N} =
0.30', '\sigma^{2}_{N} = 0.50');

%%

% PDF's of the mixed Gaussian distributions in each region
syms pdfAandC(x) pdfB(x);
pdfAandC(x) = 0.5*(1/(sqrt(2*pi*varNoise)))*exp((-(x-1)^2)/(2*varNoise)) +
0.5*(1/(sqrt(2*pi*varNoise)))*exp((-(x+1)^2)/(2*varNoise));
pdfB(x) = 0.25*(1/(sqrt(2*pi*varNoise)))*exp((-(x-2)^2)/(2*varNoise)) +
0.25*(1/(sqrt(2*pi*varNoise)))*exp((-(x+2)^2)/(2*varNoise)) +
0.5*(1/(sqrt(2*pi*varNoise)))*exp((-(x-0)^2)/(2*varNoise));
% Will just use normpdf to evaluate the probability

%%
% Simulating the system and its received signal
varNoise = noiseVariance;
count = 0;
logMLest = zeros(N);
t1vals = zeros(Niter, 1); % Store results of each iteration
t2vals = zeros(Niter, 1);

for iter = 1:Niter

    Rx = zeros(N,1);
    for t = 1:N
        % Generate the the BPSK signal
        Tx = 2*round(rand()) - 1;

        if (t < t1) || (t >= t2)
            Rx(t) = Tx; % Region A or Region C
        elseif (t >= t1) && (t < t2)
            interSym = 2*round(rand()) - 1;
            Rx(t) = Tx + interSym; % Region B
        end
        addednoise = normrnd(0,sqrt(varNoise));
        % Add AWGN noise with mean 0 and a predefined variance
        Rx(t) = Rx(t) + addednoise;
    end

    for  t1prime = 1:N    %t1
        for t2prime = 1:N %t2
            if t1prime<t2prime
                 logMLest(t1prime,t2prime) =
sum(log(0.5*normpdf(Rx(1:t1prime-
1),1,sqrt(varNoise))+0.5*normpdf(Rx(1:t1prime-1),-1,sqrt(varNoise))))...
                 +sum(log(0.5*normpdf(Rx(t1prime:t2prime-
1),0,sqrt(varNoise))+0.25*normpdf(Rx(t1prime:t2prime-
1),2,sqrt(varNoise))+0.25*normpdf(Rx(t1prime:t2prime-1),-
2,sqrt(varNoise))))...

+sum(log(0.5*normpdf(Rx(t2prime:N),1,sqrt(varNoise))+0.5*normpdf(Rx(t2prime:N
),-1,sqrt(varNoise))));
            else
                 logMLest(t1prime,t2prime) = -inf; % Define as negative
infinity so the invalid indices are disregarded
            end
        end
    end

    [value, location] = max(logMLest(:));
    [t1found, t2found] = ind2sub(size(logMLest), location);

    if (t1 == t1found) && (t2 == t2found)
        count = count + 1;
    end
end

t1var = var(t1vals)
t2var = var(t2vals)
t1avg = mean(t1vals)
t2avg = mean(t2vals)
fprintf('Percentage t1 and t2 exactly found = %d%%\n', (count/Niter)*100);
```