1 Introduction

This project involves estimating the bit error rate (BER) of a digital communication channel. The BER is defined as the number of bit errors per unit time. In this case, received values over this channel were simulated using MATLAB, and the bit error rate was estimated as the average number of bit errors for each trial of this random experiment. A single received value can be modelled as:

$$R = \sqrt{E_b}B + \sqrt{\frac{N_0}{2}}N$$

Where:

- R is the received value.
- E_b is energy per bit.
- B is the transmitted bit value ($+1$ or -1).
- N_0 is noise power spectral density.
- N is a standard normal random variable with a mean of 0 and variance of 1.

It has been assumed that $B = -1$, since the BER is the same if $B = \pm 1$. Therefore, a bit error occurs when $R > 0$. Furthermore, to estimate the BER, let the indicator random variable X be 1 if a bit error occurs ($R > 0$) and 0 if a bit error does not occur ($R < 0$). Therefore, the estimated BER is the mean of X, or $E[X]$. However, it is also desirable to specify a confidence interval around $E[X]$ in which the true BER could be in. This can be done using either the population variance or sample variance of X.

The true BER of the channel can be computed using the following equation:

$$\text{BER} = 0.5\, erfc\left(\sqrt{\frac{E_b}{N_0}}\right)$$

Where $erfc()$ is the complementary error function. The estimated BER, together with its confidence intervals, can then be compared with the true BER.

2 Program Behavior

When the MATLAB code in Appendix A is first run, the user will be asked in the MATLAB command window to choose between using the sample variance or the population variance to create the confidence intervals, as shown in Figure 2.1.

```
Which variance would you like to use? Enter 'sample'
(without the quotation marks) for sample variance or
enter 'population' for population variance:
```

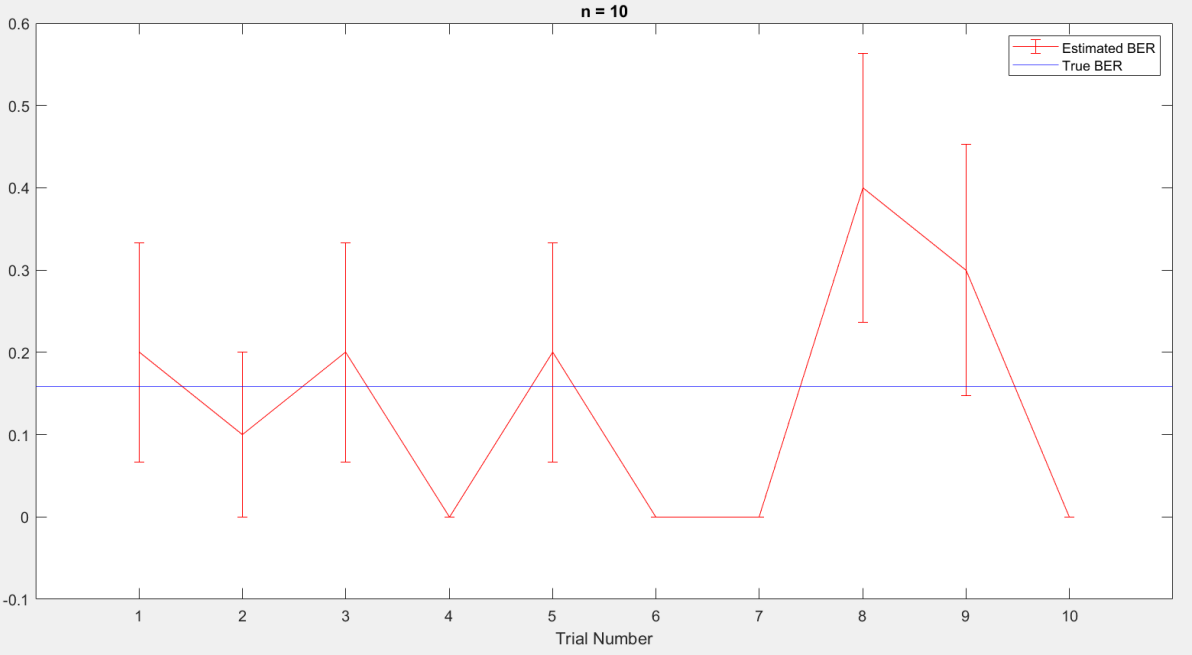
Figure 2.1 – Initial user input prompt

Once the user chooses one of the two variances to use, the percentage of $m = 100$ trials for which the true BER fell in the confidence interval will be printed in the MATLAB command window for $n = 10,100,1000$, as shown in Figure 2.2.

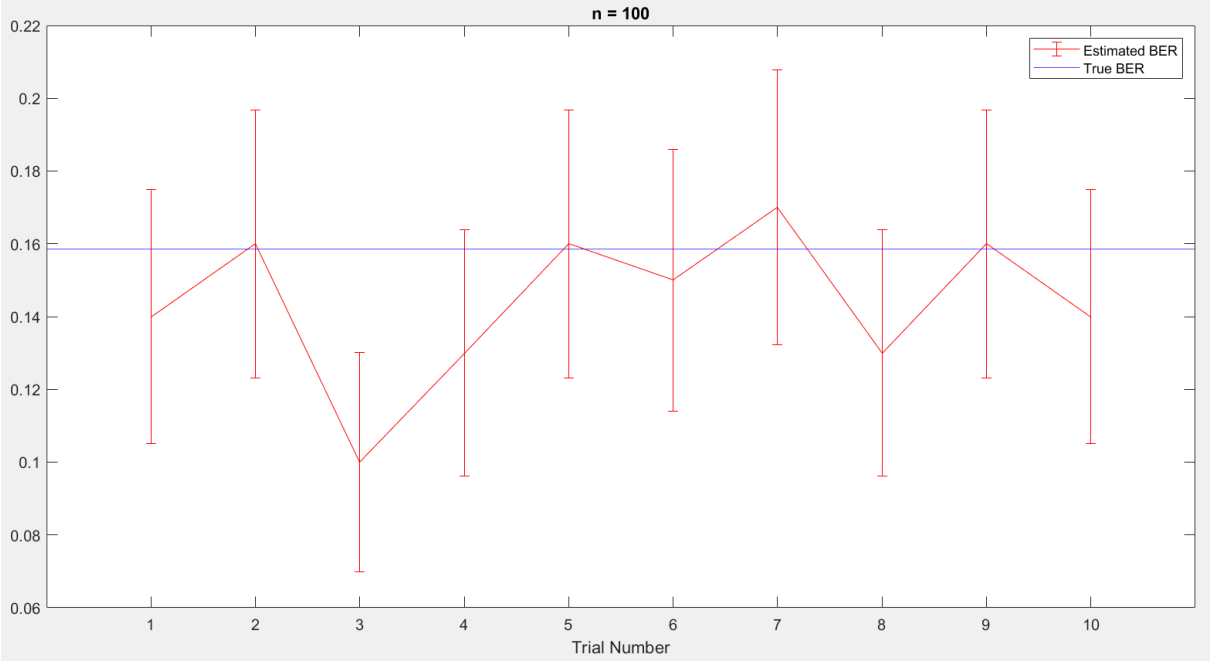
```
Which variance would you like to use? Enter 'sample'
(without the quotation marks) for sample variance or
enter 'population' for population variance:

sample

The percentage of 100 trials for which the true BER
fell in the confidence interval is 69% for n = 10.

The percentage of 100 trials for which the true BER
fell in the confidence interval is 60% for n = 100.

The percentage of 100 trials for which the true BER
fell in the confidence interval is 72% for n = 1000.
```

Figure 2.2 – Feedback based on user input

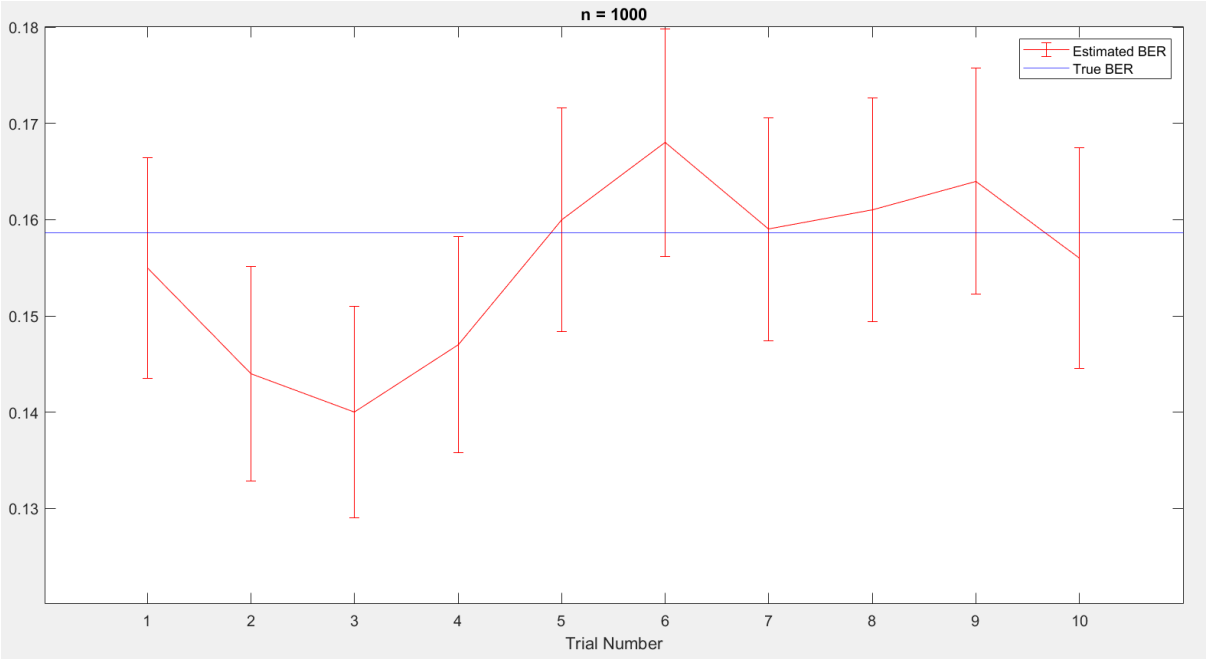
Additionally, three plots of the estimated BER and the true BER for the first 10 trials and for different values of n will be displayed in three separate figures, as shown in Figures 2.3(a), 2.3(b), and 2.3(c).



(a)



(b)



(c)

Figure 2.3 – Estimated BER vs. True BER plots for (a) n = 10, (b) n = 100, and (c) n = 1000

Note that the error bars at each point represent the confidence interval, and if the blue line intersects one of these error bars, then that means that the true BER fell in the confidence interval around the estimated BER for that trial.

3 Code Setup

In addition to the code that is explained in this section, full comments can be found in the MATLAB code in Appendix A.

```
%-------------------------------------------------------------
% initialize constants

N0 = 2;
Eb = N0 * 0.5;
```

The N_0 and E_b values are initialized such that $\frac{E_b}{N_0} = 0.5$.

```
%-------------------------------------------------------------
% define the confidence level

alpha = 0.317;

y = norminv(1-(alpha/2));
```

Since a 68.3% confidence interval around the sample mean of X is required, then $1 - \alpha = 0.683$, which means that $\alpha = 0.317$. Also, to calculate half of the confidence interval δ, the following equation can be used:

$$\delta = \frac{\sigma_X y}{\sqrt{n}}$$

Where:

$$y = \phi^{-1}\left(1 - \frac{\alpha}{2}\right)$$

And $\phi^{-1}()$ is the inverse of the standard normal cumulative distribution function. σ_X is the population or sample variance of X and n is the number of received values R.

```
%-------------------------------------------------------------
% compute the population mean of X (this is the same as the
% true BER)

mean_X_pop = 0.5 * erfc(sqrt(Eb/N0));
```

This is equivalent to calculating the true BER using the following equation:

$$\text{BER} = 0.5\, erfc\left(\sqrt{\frac{E_b}{N_0}}\right)$$

```
%-------------------------------------------------------------
% compute the population variance of X

var_X_pop = (normcdf(0,(-sqrt(Eb)),(N0/2)))*...
            (1 - normcdf(0,(-sqrt(Eb)),(N0/2)));
```

Recall that:

$$R = \sqrt{E_b}B + \sqrt{\frac{N_0}{2}}N$$

Since B is assumed to be -1, then it is not random. Therefore:

$$R = \sqrt{\frac{N_0}{2}}N - \sqrt{E_b}$$

Hence, R is an affine transformation of the random variable N. Since N is a standard normal random variable with a mean of 0 and a variance of 1, then R is a normal random variable with a mean of $-\sqrt{E_b}$ and a variance of $\frac{N_0}{2}$. Now, since X is a Bernoulli random variable, then its probability mass function is:

$$P_X(x) = \begin{cases} P(R < 0)\ for\ x = 0 \\ P(R > 0)\ for\ x = 1 \end{cases}$$

The variance of a Bernoulli random variable is [1]:

$$pq$$

Since $p = P(R > 0)$ and $q = P(R < 0)$ in this case, then the variance of X is:

$$Var(X) = P(R > 0) \times P(R < 0)$$

$P(R > 0)$ can be calculated using the following MATLAB code:

```
1 - normcdf(0,(-sqrt(Eb)),(N0/2))
```

Note also that:

$$P(R > 0) = 0.5\ erfc\left(\sqrt{\frac{E_b}{N_0}}\right)$$

And $P(R < 0)$ can be calculated using the following MATLAB code:

```
normcdf(0,(-sqrt(Eb)),(N0/2))
```

Hence, the variable `var_X_pop` stores the population (known) variance of X.

```
%-------------------------------------------------------------
% define seed used to initialize random number generator

% 'a'  +  'o'  +  'u'  +  'a'  +  'e'  +  'a'  +  'e'
%  16  +  512  +  1024 +   16  +   64  +   16  +   64 = 1712
```

```
seed = 1712;
```

The seed used to initialize the random number generator is 1712, which is based on the vowels in my full name.

```
%-------------------------------------------------------------
% number of trials

m = 100;

%-------------------------------------------------------------
% initialize the random number generator

rng(seed,'twister');
```

The number of trials for each value of n is set to 100 and the random number generator is initialized.

```
%-------------------------------------------------------------
% initialize array to store results for the first 10 trials
% of each iteration. The first column contains the sample
% mean, the second column contains the lower confidence
% boundary, and the third column contains the upper
% confidence boundary

results = zeros(10,3,3);
```

The `results` variable is used to store the results for the first 10 trials for $n = 10{,}100{,}1000$. It consists of 10 rows, 3 columns, and 3 pages, where each page corresponds to a value of n. The 10 rows store the results for the 10 trials. The first column will store the estimated BER values, the second column will store the lower confidence boundary, and the third column fill store the upper confidence boundary.

```
%-------------------------------------------------------------
% initialize array indexing variable for use inside for loop

i = 0;
```

The `i` variable is used as an iterator index in addition to iterating over n.

```
for n = [10,100,1000] % number of received values for each trial
```

A for loop is used to iterate over each value of n.

```
    %-------------------------------------------------------------
    % generate an m x n array of standard Normal random numbers

    N = randn(m,n);

    %-------------------------------------------------------------
    % generate m x n array B of transmitted -1's

    B = -ones(m,n);
```

```
%-------------------------------------------------------------
% generate m x n array R of received values

R = sqrt(Eb)*B + sqrt(N0/2)*N;

%-------------------------------------------------------------
% generate m x n array X of bit errors

X = R > 0;
```

The N array consists of standard Gaussian noise, and each row of this array represents an individual trial. The B array consists of -1's, and it is the same size as the N array. The R array is computed according to the following equation:

$$R = \sqrt{E_b}B + \sqrt{\frac{N_0}{2}}N$$

The X array consists of 0's where $R < 0$ and 1's where $R > 0$. It is the same size as the R array.

```
%-------------------------------------------------------------
% compute the sample mean of X, which is the same as the
% estimated BER, for each of the m trials and store the results
% for the first 10 trials

mean_X_samp = mean(X,2);

results(:,1,i) = mean_X_samp(1:10);
```

The estimated BER, which is the same as the sample mean of X, is calculated for each of the 100 trials by computing the mean of each row of the X array. The sample means for the first 10 trials are stored in the first 10 rows of the first column and the i^{th} page of the `results` variable.

```
%-------------------------------------------------------------
% compute the appropriate variance of X based on user input

if strcmp(chosen_variance,'sample')

    var_X = (n/(n-1))*(mean_X_samp - mean_X_samp.^2);

else

    var_X = var_X_pop;

end
```

If the user chooses to use the sample variance to compute the confidence intervals, then it is calculated using the equation:

$$S_n^2 = \frac{n}{n-1}(M_n - M_n^2)$$

Where:

- S_n^2 is the sample variance of X.
- n is the number of received values.
- M_n is the sample mean of X.

Recall that:

$$S_n^2 = \frac{1}{n-1}\left[\left(\sum_{i=1}^{n} X_i^2\right) - nM_n^2\right]$$

Since X can take the value of either 0 or 1, then $X^2 = X$. Hence:

$$S_n^2 = \frac{1}{n-1}\left[\left(\sum_{i=1}^{n} X_i\right) - nM_n^2\right]$$

Since:

$$\sum_{i=1}^{n} X_i = nM_n$$

Then:

$$S_n^2 = \frac{1}{n-1}(nM_n - nM_n^2)$$

$$= \frac{n}{n-1}(M_n - M_n^2)$$

Which is the same as the equation used above. Otherwise, the population variance is used, which was previously calculated.

```
%-------------------------------------------------------------
% find the confidence intervals for each of the m trials and
% store the results for the first 10 trials

delta = (y*sqrt(var_X))/(sqrt(n));

conf_interval = [(mean_X_samp - delta), (mean_X_samp + delta)];

results(:,2:3,i) = conf_interval(1:10,:);
```

Recall that:

$$\delta = \frac{\sigma_X y}{\sqrt{n}}$$

This is computed, and the confidence interval is determined by adding and subtracting the `delta` variable from the sample mean of X. The confidence intervals for the first 10 trials are stored in the 2nd and 3rd columns and the i^{th} page of the `results` variable.

```
%-------------------------------------------------------------
% compute the fraction of m trials for which the true BER
% is in the confidence interval

is_in_interval = mean_X_pop > conf_interval(:,1) &...
                 mean_X_pop < conf_interval(:,2);

frac_BER = (sum(is_in_interval)/size(is_in_interval,1));
```

The `is_in_interval` array contains 1's where the true BER falls within the confidence interval and 0's where it doesn't. Since there are 100 trials, then it is a 100 row by 1 column array. The `frac_BER` variable stores the fraction of trials for which the true BER falls within the confidence interval. This is calculated by summing all the elements of the `is_in_interval` array and then dividing that by the total number of rows of the array.

4 Results

n	Known Variance	Sample Variance
10	0.52	0.69
100	0.6	0.6
1000	0.72	0.72

Table 4.1 – Fraction of trials for which the true BER fell within the confidence interval

As shown in Table 4.1, the fraction of trials in which the true BER fell within the confidence interval increased gradually as n increased using the known variance. Since the true BER was computed to be only 0.1587, which is the same as $P(R > 0)$, then small n values, such as $n = 10$, will lead to little to no bit errors for each trial in the X array. This leads to a bad estimate of the sample mean of X, and therefore a bad estimate of the BER. This is because it is possible that all 10 received values are zero, which leads to a sample mean of 0. This explains why increasing n leads to a corresponding increase in the fraction of trials for which the true BER fell within the confidence interval, as there are more opportunities for a bit error to occur, and therefore a better estimate of the sample mean of X is possible.

Conversely, the fraction of trials in which the true BER fell within the confidence interval first decreased and then increased as n increased using the sample variance. Because the population (known) variance did not change between each trial, there was a initially a clear relationship between n and the fraction of trials. However, in this case, the sample variance changed every trial, and therefore the confidence interval changed as well. Recall that half the length of the confidence interval was calculated as follows:

$$\delta = \frac{\sigma_X y}{\sqrt{n}}$$

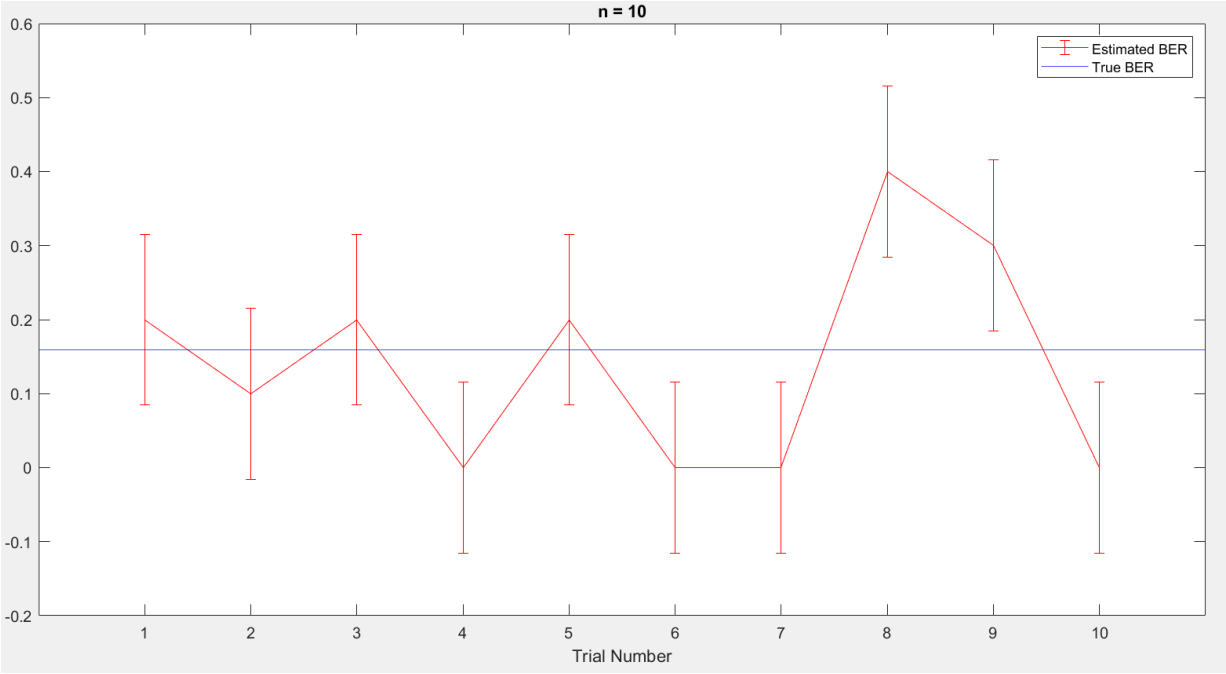
Since y is constant and n is initially small, then δ is large. Also, recall that the sample variance σ_X is calculated using the following equation:

$$S_n^2 = \frac{n}{n-1}(M_n - M_n^2)$$

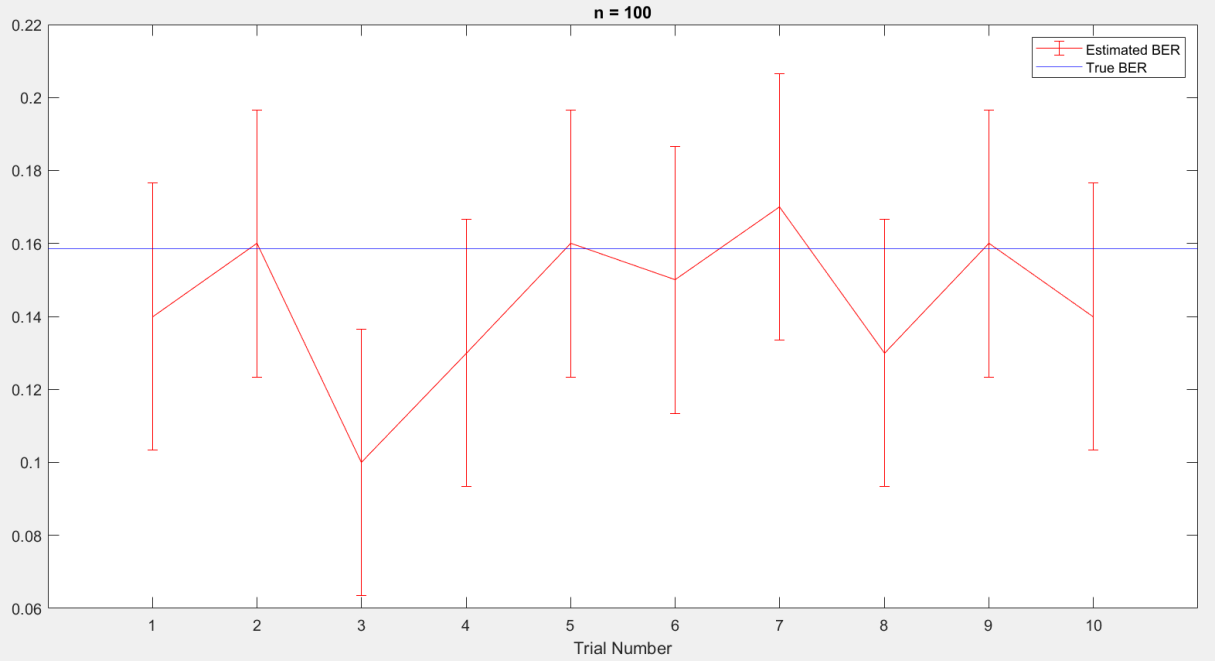
Note that because n is initially small ($n = 10$), a bad estimate of the sample mean M_n will similarly lead to a bad estimate of the sample variance S_n^2. It is possible that all 10 received values are 0, which leads to a sample variance of 0 and therefore a confidence interval of ± 0.

Hence, the size of the confidence interval rapidly changes between trials, as it is either very large or very small. This explains why the fraction of trials for which the true BER fell in the confidence interval is initially large for $n = 10$, as the confidence intervals are large enough that the true BER often falls in the confidence interval.

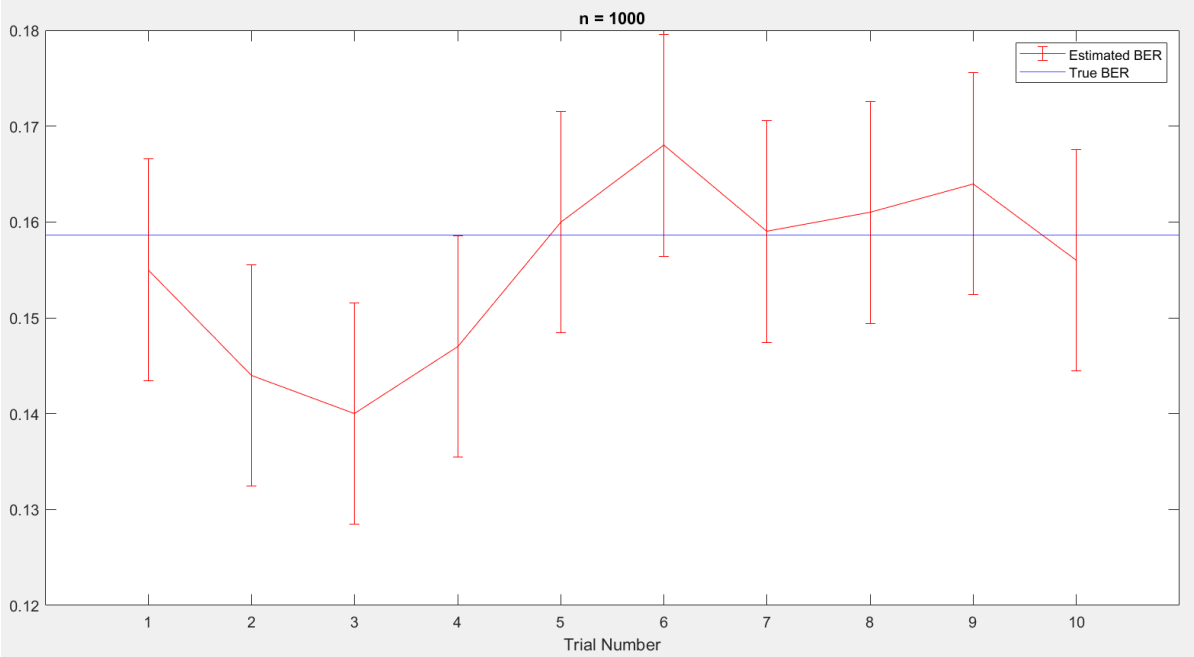
However, when $n = 100$ and $n = 1000$, the fraction of trials for which the true BER fell in the confidence interval using the sample variance starts to resemble that of the known variance, which is expected, as better estimates of the sample mean and sample variance are possible.



(a)



(b)

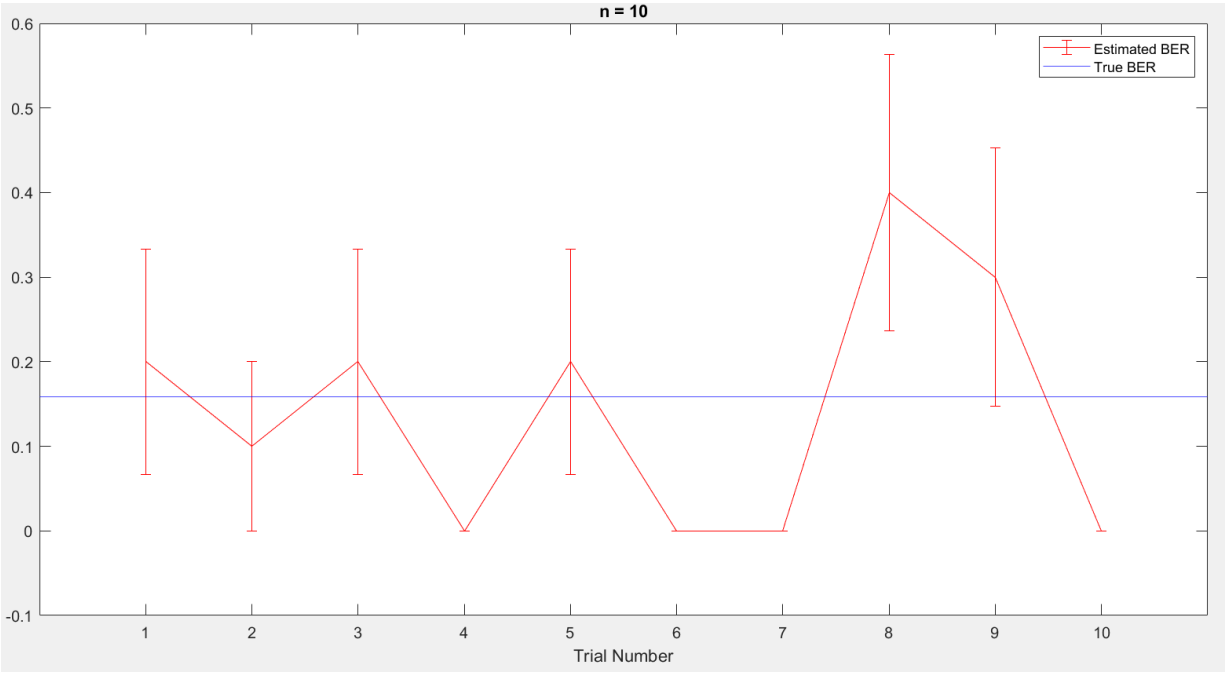


(c)

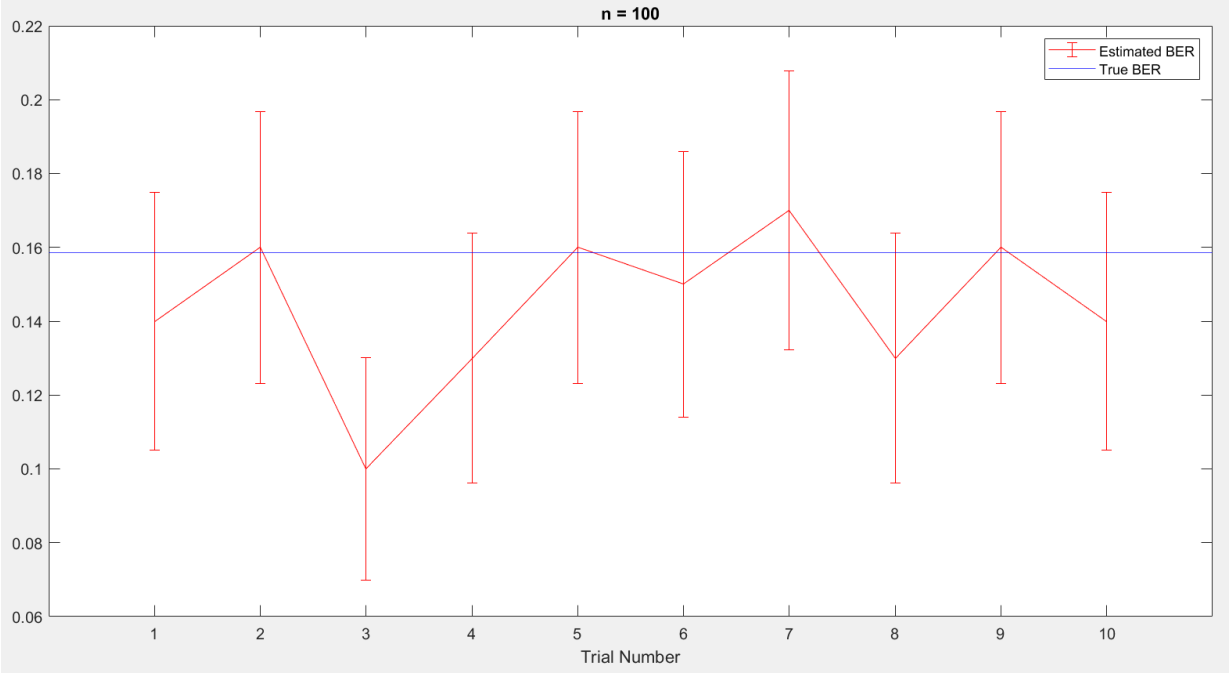
Figure 4.1 – Estimated BER vs. True BER plots for the first 10 trials using the known variance for (a) n = 10, (b) n = 100, and (c) n = 1000

Figure 4.1 shows the three plots for the estimated BER and true BER using the known variance. Note again that the error bars represent the confidence intervals, and the intersection of the blue line with one of the confidence intervals means that the true BER fell within that interval. Since

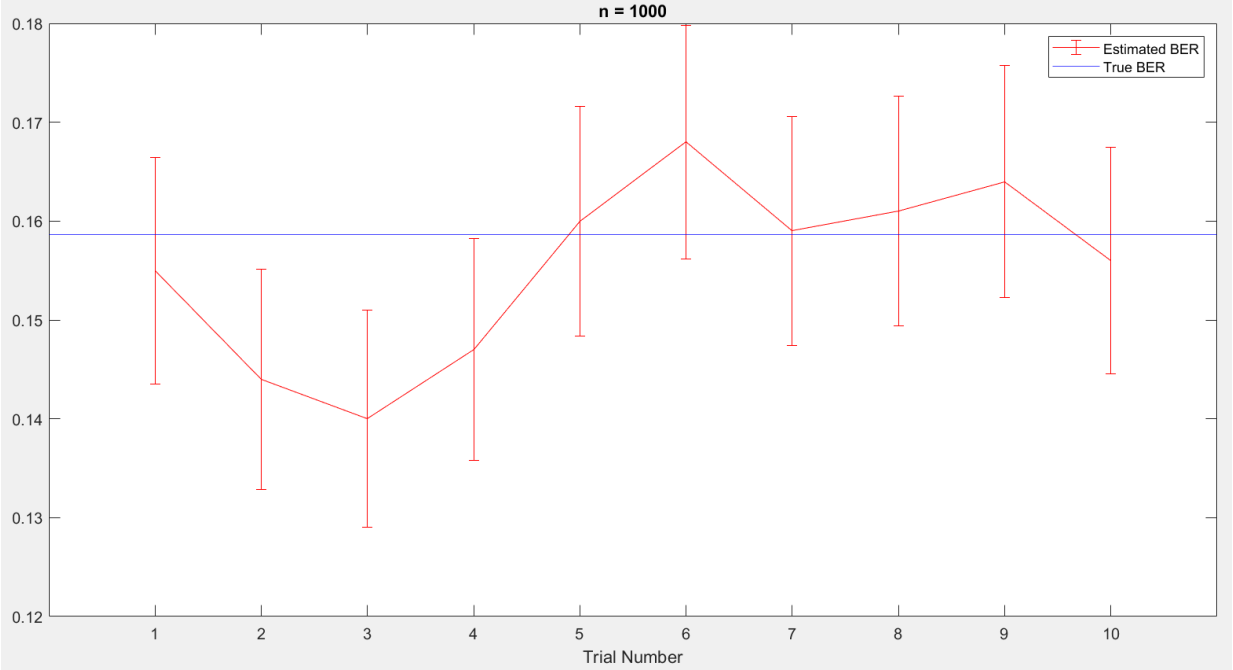
the known variance does not change between trials, the confidence intervals are the same for every sample mean.



(a)



(b)



(c)

Figure 4.2 – Estimated BER vs. True BER plots for the first 10 trials using the sample variance for (a) n = 10, (b) n = 100, and (c) n = 1000

Figure 4.2 shows the three plots for the estimated BER and true BER using the sample variance. Note again that the error bars represent the confidence intervals, and the intersection of the blue line with one of the confidence intervals means that the true BER fell within that interval. Since the sample variance changes between trials, the confidence intervals also change between trials. Note that the change in the size of the confidence intervals between trials is much smaller at $n = 1000$ than it is at $n = 10$. Note in Figure 4.2(a) that when the sample mean of X is 0, the confidence interval size is ± 0, which means no bit errors occurred, and this also demonstrates the previous point that $n = 10$ is too small to obtain good estimates of the sample mean and the sample variance.

It can also be seen in both Figures 4.1 and 4.2 that the change in the BER estimate between trials decreases as n increases. This can be seen as the "smoothing" or low-pass filtering of the line that connects consecutive BER estimates. This is consistent with the fact that the sample mean is a form of a low-pass filter and is also an unbiased estimator of the population mean (true BER), which means that as n increases, the sample mean converges to the population mean.

5 Conclusion

This project has allowed me to practice creating confidence intervals using practical data. More precisely, I learned about the limitations of using a small number of iterations per trial, as this could distort the sample mean and variance estimates, which in turn distorts the size of the confidence intervals. A minimum of $n = 100$ iterations were required in this case to be able to accurately estimate the sample mean and variance. Additionally, I have learned about a different

method that can be used to construct confidence intervals when it is difficult to derive the population mean and population variance of a probability distribution.

6 References

[1] M. Taboga, "Bernoulli distribution," [Online]. Available: https://www.statlect.com/probability-distributions/Bernoulli-distribution#hid4. [Accessed 1 November 2019].

7 Appendices

7.1 Appendix A – MATLAB Code

```
%-------------------------------------------------------------
% initialize constants

N0 = 2;
Eb = N0 * 0.5;

%-------------------------------------------------------------
% define the confidence level

alpha = 0.317;

y = norminv(1-(alpha/2));

%-------------------------------------------------------------
% compute the population mean of X (this is the same as the
% true BER)

mean_X_pop = 0.5 * erfc(sqrt(Eb/N0));

%-------------------------------------------------------------
% compute the population variance of X

var_X_pop = (normcdf(0,(-sqrt(Eb)),(N0/2)))*...
            (1 - normcdf(0,(-sqrt(Eb)),(N0/2)));

%-------------------------------------------------------------
% ask user for what kind of variance to use

txt1 = ['\nWhich variance would you like to use? Enter ''sample''\n'...
        '(without the quotation marks) for sample variance or\n'...
        'enter ''population'' for population variance:\n\n'];

chosen_variance = input(txt1,'s');

%-------------------------------------------------------------
% define seed used to initialize random number generator

% 'a'  +  'o'  +  'u'  +  'a'  +  'e'  +  'a'  +  'e'
%  16  +  512  +  1024 +   16  +   64  +   16  +   64 = 1712

seed = 1712;
```

```
%-------------------------------------------------------------
% number of trials

m = 100;

%-------------------------------------------------------------
% initialize the random number generator

rng(seed,'twister');

%-------------------------------------------------------------
% initialize string used to print the percentage of trials
% for which the true BER fell in the confidence interval

txt2 = ['\nThe percentage of %d trials for which the true BER\n'...
        'fell in the confidence interval is %d%% for n = %d.\n'];

%-------------------------------------------------------------
% initialize array to store results for the first 10 trials
% of each iteration. The first column contains the sample
% mean, the second column contains the lower confidence
% boundary, and the third column contains the upper
% confidence boundary

results = zeros(10,3,3);

%-------------------------------------------------------------
% initialize array indexing variable for use inside for loop

i = 0;

for n = [10,100,1000] % number of received values for each trial
    %---------------------------------------------------------
    % increment array indexing variable

    i = i + 1;

    %---------------------------------------------------------
    % generate an m x n array of standard Normal random numbers

    N = randn(m,n);

    %---------------------------------------------------------
    % generate m x n array B of transmitted -1's

    B = -ones(m,n);

    %---------------------------------------------------------
    % generate m x n array R of received values

    R = sqrt(Eb)*B + sqrt(N0/2)*N;

    %---------------------------------------------------------
    % generate m x n array X of bit errors

    X = R > 0;
```

```
%-------------------------------------------------------------
% compute the sample mean of X, which is the same as the
% estimated BER, for each of the m trials and store the results
% for the first 10 trials

mean_X_samp = mean(X,2);

results(:,1,i) = mean_X_samp(1:10);

%-------------------------------------------------------------
% compute the appropriate variance of X based on user input

if strcmp(chosen_variance,'sample')

    var_X = (n/(n-1))*(mean_X_samp - mean_X_samp.^2);

else

    var_X = var_X_pop;

end

%-------------------------------------------------------------
% find the confidence intervals for each of the m trials and
% store the results for the first 10 trials

delta = (y*sqrt(var_X))/(sqrt(n));

conf_interval = [(mean_X_samp - delta), (mean_X_samp + delta)];

results(:,2:3,i) = conf_interval(1:10,:);

%-------------------------------------------------------------
% compute the fraction of m trials for which the true BER
% is in the confidence interval

is_in_interval = mean_X_pop > conf_interval(:,1) &...
                 mean_X_pop < conf_interval(:,2);

frac_BER = (sum(is_in_interval)/size(is_in_interval,1));

%-------------------------------------------------------------
% show percentage of m trials for which true BER is in
% the confidence interval for each of n

fprintf(txt2,m,frac_BER*100,n);

%-------------------------------------------------------------
% plot results in three separate figures with confidence
% intervals

figure(i)
errorbar(1:10,results(:,1,i),...
        (results(:,3,i)-results(:,2,i))/2,'-r');
hold on
yline(mean_X_pop,'-b');
hold off
```

```
    xlim([0,11]);
    xticks(1:1:10);
    legend('Estimated BER','True BER');
    title(['n = ',num2str(n)]);
    xlabel('Trial Number');
end
```