CHANNEL CO-EFFICIENT ESTIMATION USING MONTE-CARLO SIMULATION

Mahathi Lanka | 200425925 | Date: 10.18.2021

**Background:**

In digital communication systems, A channel is a medium through which the transmitter output is sent. In general scenarios, channels may introduce distortion to the source signal, and the characteristics of the channel/distortion need to be estimated or identified at the receiver end, to reduce or eliminate the distortion and recover the original signal. This is called channel estimation or identification.

In this project, confidence intervals are used to quantify the accuracy of channel estimation in wired communication. The system sends known (+1) and unknown (+1 or -1) symbols, after appropriate sampling and filtering, the real valued received sample for the *i*th symbol period is modelled as

where,

C = Channel co-efficient to be estimated

Ni = Sequence of i.i.d Gaussian noise random variables with zero mean and variance σN2

For every *m* received samples available for channel estimation, it is assumed that there are *n* pilot symbols.

Given that the true channel co-efficient is 10 which is going to be constant throughout the estimation. From equation (1), it can be inferred that the real-valued symbol is the sum of the true channel coefficient and the gaussian noise. We utilize the sample mean as an estimator on the received samples (*Xi*) to estimate channel co-efficient. SNR (signal to noise ratio) is used to quantize the received signal.

Given that,

SNR = -6dB, Hence the resultant variance of Noise variable can be calculated and is equal to \_\_\_.

**Objective:**

The goal of the project is to simulate the confidence intervals for varied sample sizes of the received symbols and compute the confidence intervals using 3 different approaches for estimating the Channel co-efficient.

**Experimental Setup:**

To perform the mentioned simulation, I am using MATLAB® (ML) as my platform to work. To estimate the channel coefficients, I am performing 1000 trails and doing the simulation for three different values of the received signal (n) which are n = 10, n = 100, n = 1000 and I am following the below steps:

1. Taking user input for number of samples, number of trials and trials to be used for the plot.
2. Initializing sample mean & standard deviation
3. Using a vowelVal.m function to pass a seed value to the random number generator function.
4. Calculating the sample mean and sample variance for m trails.
5. Estimating confidence interval using known variance. (Case1)
6. Estimating confidence interval using estimated sample variance. (Case2)
7. Estimating confidence interval using unknown variance when samples are assumed to be gaussian. (Case3)
8. Calculating the count of the number of times the true channel coefficient lies within the upper and lower bounds of the estimated channel coefficient.
9. Plotting graphs between estimated channel coefficients and the confidence intervals (for all the three cases).

**Calculation of the seed for random number generator**:

Rng (seed, ‘twister’) function is used to initialize the random number generator. The seed to the rng function is passed through vowelVal.m script where, seed is set to an integer determined by certain vowels in your first and last name (A=16, E=64, I=256, O = 512, U = 1024) by summing the numbers corresponding to the vowels present.

“Mahathi Lanka” would give the seed 16 (a) + 16(a) + 256 (i) + 16 (a) + 16(a) = 320

Therefore, 320 is being passed as the seed to the random number generator settings.

**Estimation of C:**

We know that the estimated value of C is given by,

Therefore, we estimate the channel coefficient by taking a mean of the received symbol X i.e.,

**Estimating Confidence interval using known variance:**

Using the given equation for the quality of the received signal,

From (2),

We calculate σN = 20

The confidence interval can be calculated using the below process,

**Table for for case 1:**

|  |  |
| --- | --- |
| **Number of samples** | **Value for** |
| N = 10 | 6.34 |
| N = 100 | 2 |

**Estimation of confidence intervals for estimated variance:**

While calculating the confidence interval using the estimated variance, we can use the same method as above and just replace actual variance with estimated variance given below,

This is the unbiased estimated variance.

**Estimation of confidence intervals with estimated variance & gaussian samples**:

Here Y/2 is first calculated using t-student distribution since this is assumed as a gaussian distribution.

Since, the variance is unknown, square root of sample variance is assumed to be

/

**Y :**

With reference to [1], From the textbook we understand that for cases with known variance and estimated variance, Y doesn’t depend on the number of samples. Therefore. For both n=10 & 100, the value for Yremains constant and is equal to 1.645.

But, for case 3 since the samples are all gaussian, they are bound to depend on the number of samples as we can see from equation (5). Therefore, with increase in number of samples, the values of Y decreases as observed in the below mentioned table.

|  |  |  |
| --- | --- | --- |
| **Number of samples** | **Case 1 & Case 2** | **Case 3** |
| N =10 | 1.645 | 1.833 |
| N= 100 | 1.645 | 1.660 |

**Results:**

|  |  |  |  |
| --- | --- | --- | --- |
| Number of Samples | Fraction within confidence interval (Known variance) | Fraction within confidence interval (Unknown variance) | Fraction within confidence interval (Unknown variance with Gaussian) |
| 10 | 0.9150 | 0.8810 | 0.9060 |
| 100 | 0.9050 | 0.9050 | 0.9080 |
| 1000 | 0.9180 | 0.9150 | 0.9150 |

The above table maps the fraction of trials for which the true channel coefficient Ctrue (true mean) fell within the confidence interval. A row for each value of n and columns for each of the 3 cases. It can be inferred from the results that, for the case 3 where gaussian samples with unknown variance is used to estimate, the performance of the simulation is the best since the width of the upper & lower bounds of the confidence intervals was broader than the case 1 & 2 (Refer to plots) thereby increasing the chance for the True channel value to fall within the bounds. If we were to draw a comparison between the performance of simulation for each of the cases it is as follows,

It is understood that case 2 achieves the least performance because the assumed variance value is not the true variance but the estimated variance.

**Plots obtained:**

N = 10

Chart, line chart

Description automatically generated

**N =100**

Chart

Description automatically generated

**Inference from the plots**:

For the case with N = 10, since the sample size is commendably small, there is a large variation in the range of the count of Channel values falling with in the bound of the estimated channel values. [0.9150, 0.8860, 0.9060]

But for the case where N =100, the haphazardness in the count is considerably reduced and estimation for all the 3 cases yielded nearly same results. [0.9050, 0.9050, 0.9080]

**Conclusion**:

This project gave me a clear understanding of implementation of Monte-Carlo simulations for confidence interval estimation in the practical scenarios like communication. I learnt how to use confidence intervals to quantify the accuracy of channel estimation. The key takeaway of the simulation was, as we keep increasing the sample sizes of the simulation, the performance keeps getting better.

**Appendix:**

Source code:

sigma\_N = 20;

COriginal = 10;

My\_name = 'Mahathi Lanka';

confidence\_value = 0.9;

alpha = 1 - confidence\_value;

function [seed] = vowelcount(My\_name)

a = 0;

x = upper(My\_name);

for i=1:length(x)

if(x(i) == 'A')

a= a + 16;

elseif(x (i) == 'E')

a = a + 64;

elseif(x(i) == 'I')

a= a + 256;

elseif(x(i) == 'O')

a = a + 512;

elseif(x (i) == 'U')

a = a + 1024;

end

end

seed = a;

end

%User input for num of samples, trails & x-axis of the plots

n\_sample = input('number of samples: ');

m\_trials = input('number of trials: ');

trial\_size = input('Enter the x\_axis length:');

constants;

%initializing sample mean for random variable X to zeroes which changes according to the input

Sm\_X = zeros(1, m\_trials);

Sd\_X = zeros(1, m\_trials);

% Seed value for 'Mahathi Lanka' = 16+16+256+16+16=320

val = vowelcount(My\_name);

display(val);

% initializing rng settings to twister

rng(val, "twister");

Randomfunc = rng;

%% Calculating for m trials

for i=1:m\_trials

%Generating random values for N

rand\_number = randn(1, n\_sample);

%disp(randomVal\_N)

X = sigma\_N\*rand\_number + COriginal; %realization of X

Sm\_X(i) = mean(X); %Estimation

Sd\_X(i) = std(X);

end

%% Case1 known variance sigmaN

%confidence interval is 90%, therefore 1- alpha/2 = 0.95

% delta = norminv(0.95, 0, 20/sqroot(n\_sample))

d\_1 = norminv(1-alpha/2, 0, sigma\_N/sqrt(n\_sample));

% clculating upper and lower bounds

CI\_1\_upper = Sm\_X + d\_1;

CI\_1\_lower = Sm\_X - d\_1;

% number of times CTrue lies within CI for Case 1

count\_1 = 0;

for i=1:m\_trials

if((CI\_1\_lower(i)<=COriginal) && (COriginal<=CI\_1\_upper(i)))

count\_1 = count\_1 + 1;

end

end

disp(count\_1/m\_trials)

%% Case2 unkown variance. Sample Variance estimator.yalpha/2 computed and verfied using Table 6.3 in Gubner = 1.645

yalphaby2 = sqrt(2)\*erfinv(confidence\_value);

% since variance is unknown, variance is estimated

d\_2 = (Sd\_X\*yalphaby2)/sqrt(n\_sample);

%calculating upper and lower bounds

CI\_2\_upper = Sm\_X + d\_2;

CI\_2\_lower = Sm\_X - d\_2;

% number of times originalC's lies within CI

count\_2 = 0;

for i=1:m\_trials

if(CI\_2\_lower(i)<=COriginal && COriginal<=CI\_2\_upper(i))

count\_2 = count\_2 + 1;

end

end

disp(count\_2/m\_trials)

%% Case 3 unknown variance and samples are gaussian and the values obtained are 1.833 & 1.60

%for gaussian samples, y alpha/2 is calculated using t student distribution

yalphaby2\_2 = tinv(1-alpha/2, n\_sample-1);

%using sample variance

d\_3 = (Sd\_X\*yalphaby2\_2)/sqrt(n\_sample);

%computing upper and lower bounds

CI\_3\_upper = Sm\_X + d\_3;

CI\_3\_lower = Sm\_X - d\_3;

% number of times Original C'S lies within CI

count\_3 = 0;

for i=1:m\_trials

if(CI\_3\_lower(i)<=COriginal && COriginal<=CI\_3\_upper(i))

count\_3 = count\_3 + 1;

end

end

disp(count\_3/m\_trials)

%% Plot 3 different cases

figure();

subplot(3, 1, 1);

p1=plot(1:trial\_size, COriginal\*ones(1,trial\_size));

l1='Original Channel';

hold on

p2=plot(1:trial\_size, Sm\_X(1:trial\_size),'--');

l2='Channel new';

p3=plot(1:trial\_size, CI\_1\_upper(1:trial\_size),'-.^');

l3='CI Upper bound';

p4=plot(1:trial\_size, CI\_1\_lower(1:trial\_size),'-.v');

l4='CI Lower bound';

hold off

legend([p1 p2 p3 p4], {l1, l2, l3, l4});

xlabel(' number of Trials');

title('case 1 with Known variance');

subplot(3, 1, 2);

p1=plot(1:trial\_size, COriginal\*ones(1,trial\_size));

l1='Original Channel';

hold on

p2=plot(1:trial\_size, Sm\_X(1:trial\_size), '--');

l2='Channel new ';

p3=plot(1:trial\_size, CI\_2\_upper(1:trial\_size), '-.^');

l3='CI Upper bound';

p4=plot(1:trial\_size, CI\_2\_lower(1:trial\_size), '-.v');

l4='CI Lower bound';

hold off

legend([p1 p2 p3 p4], {l1, l2, l3, l4});

xlabel('number of Trials');

title('case 2 with Unknown variance');

subplot(3, 1, 3);

p1=plot(1:trial\_size, COriginal\*ones(1,trial\_size));

l1='Original Channel';

hold on

p2=plot(1:trial\_size, Sm\_X(1:trial\_size), '--');

l2='new Channel';

p3=plot(1:trial\_size, CI\_3\_upper(1:trial\_size), '-.^');

l3='CI Upper bound';

p4=plot(1:trial\_size, CI\_3\_lower(1:trial\_size), '-.v');

l4='CI Lower bound';

hold off

legend([p1 p2 p3 p4], {l1, l2, l3, l4});

xlabel('number of Trials');

title('case 3 with Unnown variance & Guassian Samples');

**References:**

[2] www.mathworks.com

[1]Probability and Random Processes for Electrical and Computer Engineers – John A. Gubner