Dana Joaquin

ECE 3512: Signals – Continuous and Discrete

Department of Electrical and Computer Engineering, Temple University, Philadelphia, PA 19121

# Problem Statement

The purpose of this assignment is to see how the number of samples can affect the accuracy of calculating the estimated mean and variance of a particular signal or set of data. For this assignment, we are to compute the estimated mean and variance of a uniformly distributed PDF of a signal containing random values from 0 to 1. It is known that the mean, µ, should be 0.5. For the variance value, that can be calculated using the variance formula for uniform distribution, and knowing the values range from 0 to 1, the variance for this random signal:

Equation 1. Variance Equation for Uniform Distribution

# Approach and Results

For the first part of the assignment, a random signal containing values from 0 to 1 with a length of 105­ elements was made using the MATLAB function “rand.” The signal’s estimated mean was computed using a coded function “compute\_mean” which does the calculations using the Equation 2. This function required the input arguments: the random signal vector and the length of the signal value.

Equation 2. Estimated Mean Formula

A variance function, “compute\_var” was written to compute the estimated variance of the random signal using Equation 3. The input arguments are the random signal vector, estimated mean vector that was computed from the compute\_mean function, and the length of the vector value.

Equation 2. Estimated Variance Formula

Once the estimated mean and variance vectors are made, the error vector was then made using these following equations:

Equation 3. Formula for calculating mean error

Equation 4. Formula for calculating variance error

Each element had their true value subtracted to create the error mean and error variance vectors.The resulting vectors were then plotted separately using the MATLAB function semilogx to have a log-based horizontal axis, resulting in figures Figure 1a and Figure 1b. Figure 1a is the mean error result and Figure 1b is the variance error result.

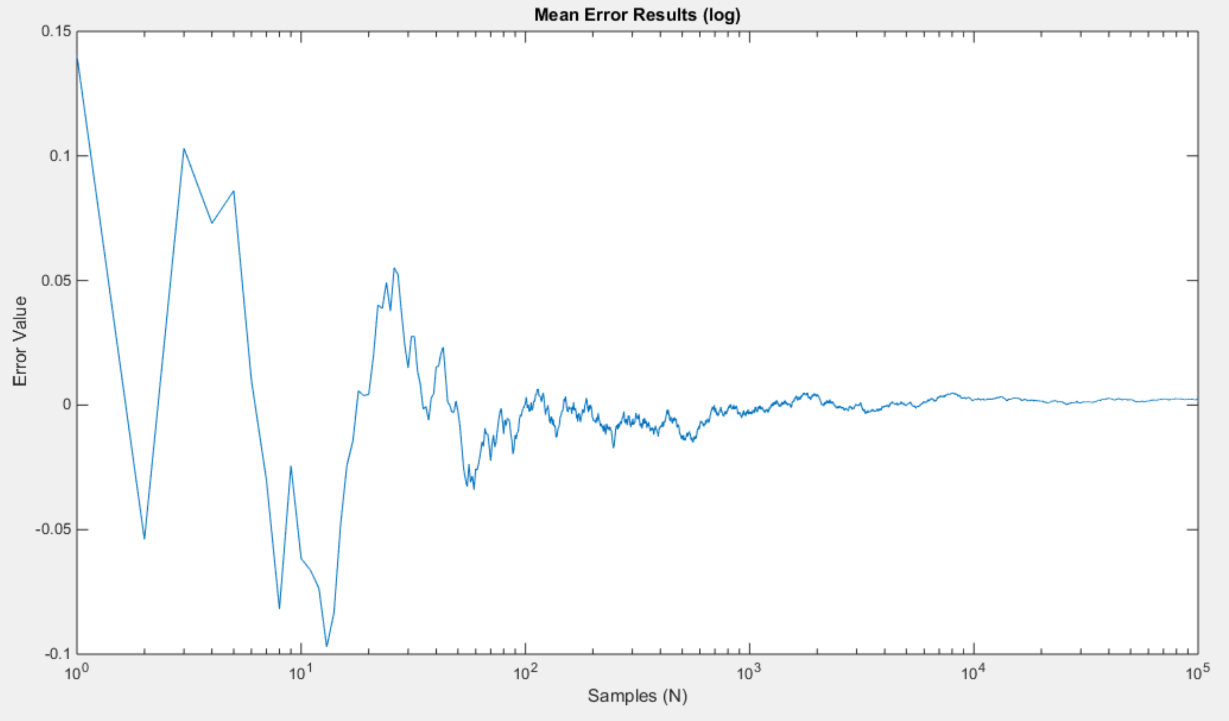


Figure 1a. Error mean calculation plot

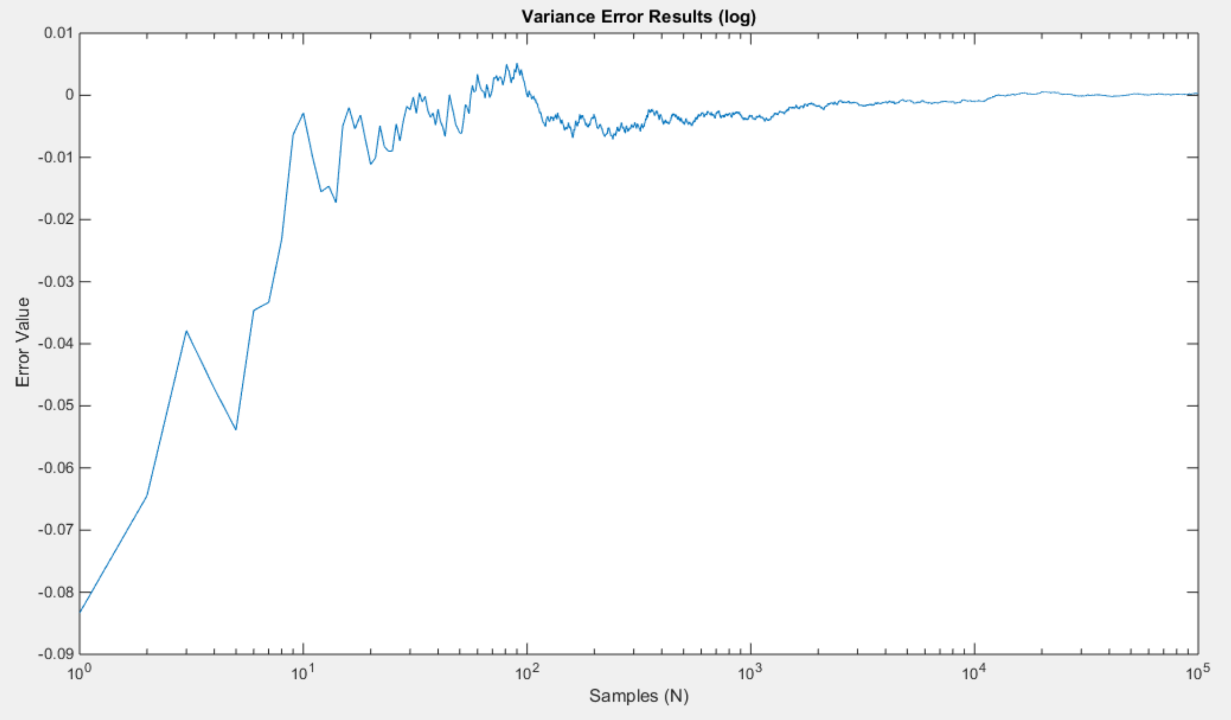


Figure 1b. Error variance calculation plot

Part 2 of this assignment asked compare the estimated PDF of the same type of signal used in part 1 with its actual PDF (uniform distribution) and calculate and plot the mean-squared error between the estimated and actual PDF. This part was done on 3 random signals with lengths of 10, 1e3 and 1e6.

First, the histogram of the PDF of the signal was generated using MATLAB’s histogram function. The bins value from the histogram was used to create a vector that contains the true PDF values that has the same values as the histogram. This was done by coding the function “unidist” to create this vector, taking in the arguments of the true uniform distribution value and the length of desired vector. The estimated PDF values came from the data values of the histogram. The true value vector was plotted on top of the histogram to compare it with the estimated PDF values. For the mean-squared error calculations, the same mean-squared function in computer assignment 4 was used. It was slightly altered for this assignment to take in vector values rather than scalar values. This function does the calculations using the means-squared error equation:

Equation 5. Mean-squared Error Equation

This algorithm was used for each signal, generating figures, Figure 2a, Figure 2b, and Figure 2c.

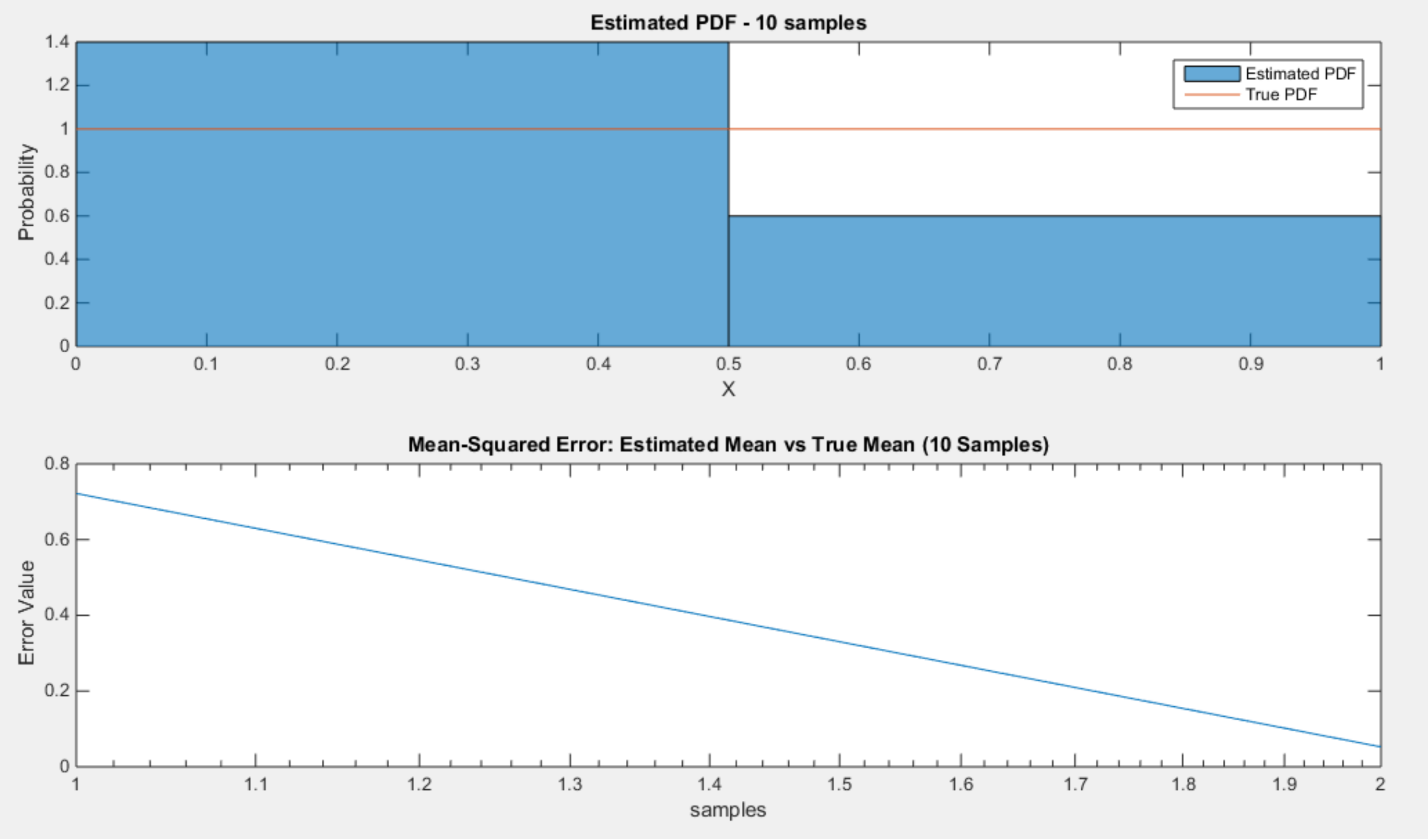


Figure 2a. Top plot is estimated and actual PDF of random signal N=10. Bottom plot is the mean-squared error plot between the estimated and true PDF.

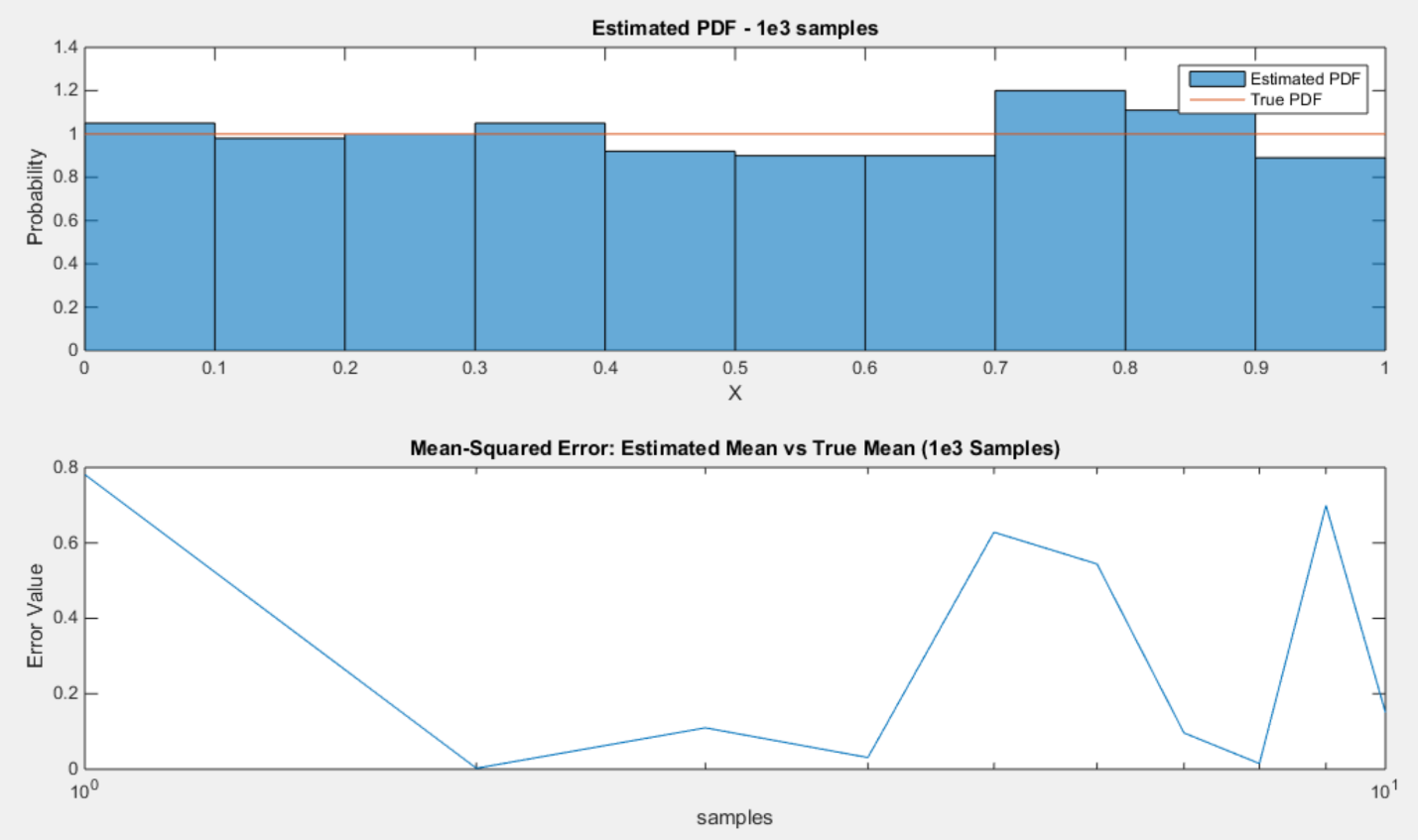


Figure 2b. Top plot is estimated and actual PDF of random signal N=10. Bottom plot is the mean-squared error plot between the estimated and true PDF.

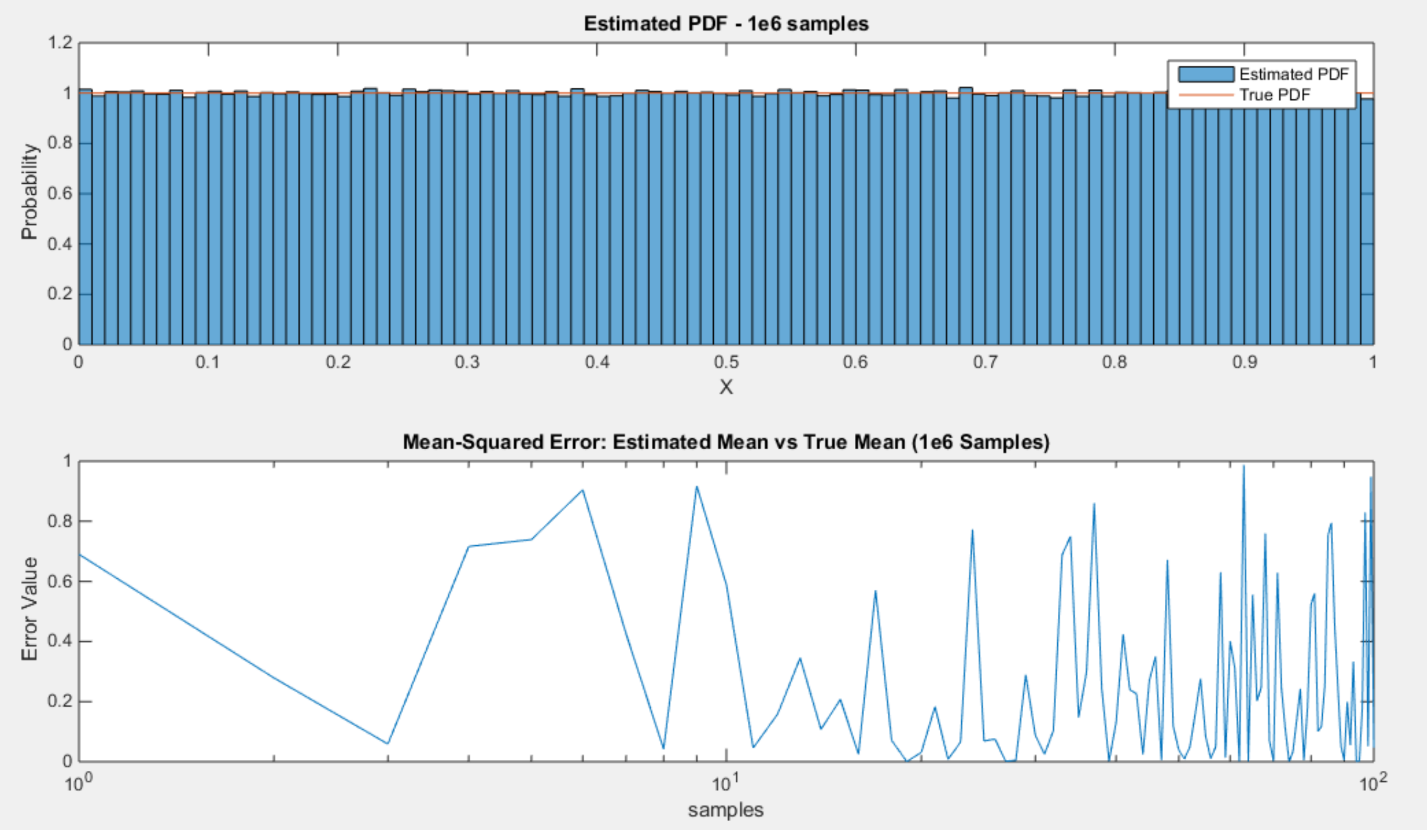


Figure 2c. Top plot is estimated and actual PDF of random signal N=10. Bottom plot is the mean-squared error plot between the estimated and true PDF.

# MATLAB Code

|  |
| --- |
| % Computer Assignment 6  % Dana Joaquin    function ca06    clear; clc; clf  close all    %------------------------- Question 1 -------------------------    mu = 0.5; % expected mean  sigma2 = 1/12; % expected variance  N = 1e5; % # of points  M = 1; % dimension of matrix  x = 1:N; % x-scale  siggie = rand(M, N); % randommmmm signallll!!! :D  mu1 = compute\_mean(N, siggie); % computing estimated mean of signal  errormean = mu1-mu; % calculating error vector of mean  var1 = compute\_var(N, siggie, mu1); % computing estimated variance  errorvar = var1-sigma2; % calculating error vector of variance    % plotting results  % mean  figure(1)  semilogx(errormean) % log plot  title('Mean Error Results (log)')  xlabel('Samples (N)')  ylabel('Error Value')    % variance  figure(2)  semilogx(errorvar) % log plot  title('Variance Error Results (log)')  xlabel('Samples (N)')  ylabel('Error Value')  %---------------------------------------------------------------      %------------------------- Question 2 --------------------------    % parameters  M = 1;  N = [10 1e3 1e6];  binz = 100;  valrange = [0 1];  truval = 1/(valrange(2)-valrange(1));    % generating 3 random signals  sig1 = rand(M, N(1)); % length of 10  sig2 = rand(M, N(2)); % length of 1000  sig3 = rand(M, N(3)); % length of 1e6    % plotting sig1 PDF results  figure(3)  subplot(2, 1, 1)  histpdf = histogram(sig1, 'Normalization', 'pdf'); %estimated PDF  title('Estimated PDF - 10 samples')  xlabel('X')  ylabel('Probability')  binz = histpdf.NumBins; % # of samples  estsig = histpdf.Data; % storing estimated PDF values in vector  trusig = unidist(binz, truval); % creating same size vector as estsig,  % contains expected PDF  t = linspace(0,1,binz); % x-scale vector  hold on  plot(t, trusig) % plot true uniform distribution vector  legend('Estimated PDF', 'True PDF')    subplot(2, 1, 2)  mse = compute\_mse(estsig, trusig, binz); % computing MSE: true vs. estimated  semilogx(mse)  title('Mean-Squared Error: Estimated Mean vs True Mean (10 Samples)')  xlabel('samples')  ylabel('Error Value')    % plotting sig2 PDF results  figure(4)  subplot(2, 1, 1)  histpdf = histogram(sig2, 'Normalization', 'pdf'); %estimated  title('Estimated PDF - 1e3 samples')  xlabel('X')  ylabel('Probability')  binz = histpdf.NumBins; % # of samples  estsig = histpdf.Data; % storing estimated PDF values in vector  trusig = unidist(binz, truval); % creating uniform distribution vector  % that's length of binz  t = linspace(0,1,binz); % creating x-scale vector  hold on  plot(t, trusig) % plotting uniform distribution value  legend('Estimated PDF', 'True PDF')    subplot(2, 1, 2)  mse = compute\_mse(estsig, trusig, binz); % computing MSE: true vs. estimated  semilogx(mse) % plotting MSE  title('Mean-Squared Error: Estimated Mean vs True Mean (1e3 Samples)')  xlabel('samples')  ylabel('Error Value')    figure(5)  subplot(2, 1, 1)  histpdf = histogram(sig3, 'Normalization', 'pdf'); %estimated PDF  title('Estimated PDF - 1e6 samples')  xlabel('X')  ylabel('Probability')  binz = histpdf.NumBins; % # of samples  estsig = histpdf.Data; % storing estimated PDF values into vector  trusig = unidist(binz, truval); % creating vector of uniform distribution values  % that's the length of binz  t = linspace(0,1,binz); % x-scale vector  hold on  plot(t, trusig) % plotting expected uniform distribution  legend('Estimated PDF', 'True PDF')    subplot(2, 1, 2)  mse = compute\_mse(estsig, trusig, binz); % calculating MSE  semilogx(mse) % plotting MSE  title('Mean-Squared Error: Estimated Mean vs True Mean (1e6 Samples)')  xlabel('samples')  ylabel('Error Value')    %---------------------------------------------------------------  end    % function name: compute\_mean  % coded by Dana Joaquin  % input argument(s):  % (1) n: length of array  % (2) val: uniform distribution value  % output argument(s):  % (1) unicorn: vector containing uniform distribution values  % objective:  % This function creates a vector that is n-long where  % every element contains the uniform distribution value, val.  %  function unicorn = unidist(n, val)  v = zeros(1, n);  for k=1:n  v(k) = val;  end    unicorn = v;  end      % function name: compute\_var  % coded by Dana Joaquin  % input argument(s):  % (1) nsize: size of the vector  % (2) rsig: signal data (array)  % (3) meansig: vector containing mean values of signal  % output argument(s):  % (1) so\_var: vector containing the variance of the signal  % objective:  % Takes the signal data and computes the estimated variance  % sample by sample until the end of the data array. Does this  % using this formula:  %  % u = estimated mean value  %  % estimated \_n-1\_  % variance [N] = 1 \ 2  % --- / ( (X[n]-u) )  % N -----  % n=0  %  function so\_var = compute\_var(nsize, rsig, meansig)    for i=1:nsize  a = rsig(1:i);  b = meansig(1:i);  n = length(rsig(1:i));  k = 1/n;  varv = k\*sum((a-b).^2);  so\_var(i)=varv;  end    end    % function name: compute\_mean  % coded by Dana Joaquin  % input argument(s):  % (1) nsize:  % (2) rsig: signal data (array)  % output argument(s):  % (1) so\_rude: vector containing the mean of the signal  % objective:  % Takes the signal data and computes the estimated mean  % sample by sample until the end. Does this by implementing  % this estimated mean formula:  %  % estimated \_n-1\_  % mean[N] = 1 \  % --- / X[n]  % N -----  % n=0  %  function so\_rude = compute\_mean(nsize, rsig)    for i=1:nsize  mew = (1/i)\*sum(rsig(1:i));  so\_rude(i) = mew;  end    end      % function name: compute\_mse  % coded by Dana Joaquin  % input argument(s):  % (1) sig\_a: signal data (array)  % (2) sig\_b: signal data (array)  % (3) length\_vect: length of signal data (array)  % output argument(s):  % (1) mse: vector value of mean-squared error  % objective:  % This function computes the mean-squared value of two  % data arrays brought in by sig\_a and sig\_b.  % Calculates the mean-squared value by using the formula:  %  % n : # of samples (both sig\_a and sig\_b have to be same size)  % Y-hat : predicted values (sig\_a is true val. arrays)  % Y : true values (sig\_b is predicted val. arrays)  %  % \_\_n\_\_  % MSE = 1 \ \_ 2  % --- / (Y - Y)  % n ----- i i  % i = 1  %  function mse\_val = compute\_mse(sig\_a, sig\_b, sig\_length)    n = sig\_length;  for i=1:n  data\_array(i) = (sig\_b(i) - sig\_a(i))^2;  end    mse\_val = data\_array;    end |

# Conclusions

Based on the results, the more samples computed, the more likely the estimated mean/variance will converge to the true mean/variance of the signal data. This is shown in the figures 1a and 1b where the error starts to converge to zero as the number of samples increased. This is expected because as you let an event continue for an infinite amount of time, it should converge to its true parameter values. However, this is not the case for real life situations because most of the time, the true mean and variance are not known and the appropriate amount of samples is not known as well. This is where engineers and statisticians have to decide how many samples they need in order to make their estimated parameter values as accurate and reliable as much as possible. The amount of samples is the challenge because although making an event run for an infinite amount of time will result in the true values, in reality, one cannot process and collect an infinite amount of data. How this is compensated is that engineers and statisticians choose an appropriate amount of samples and from that, they can compute a range where the parameters will most likely be in.

For question 2, the mean-squared error showed how much error there is between the estimated and true values. Comparing the mean-squared error plot and the PDF, where there is about 0 error, the true value and estimated value are almost the same. Both plots agreed with each other, therefore making the results reliable. Since the generated signal was a random uniformly distributed PDF, it would make sense that the true PDF would be a uniform distribution. In real life, collected data may not have a definite distribution shape, making it hard to decide what distribution fits with it the best. This is where the mean-squared error can help with that, it can be used to see and compare what distribution plot fits best with the collected data, and the one with the least amount of error is the better fit.