William O’Mullan

ECE 3522: Stochastic Processing

Department of Electrical and Computer Engineering, Temple University, Philadelphia, PA 1912

# Problem Statement

In this computer assignment we revisited the concepts of mean and variance, but this time applied to a selection of truly random numbers as opposed to a data set that contains information such as the Google closing stock prices or the speech recording. We were required to progressively find the mean and variance for a data set that was comprised of uniformly distributed random numbers in the range of with different numbers of data points within the set. Additionally this assignment goes over how the number of data points we have affects our results by having us compare the error for our estimated means, variances, and PDFs with their exact value for a uniform distribution.

# Approach and Results

We know that the mean for a uniform distribution can be estimated with the equation:

For a uniform distribution in the range of we know the exact value for the mean is:

We also know that the variance for a uniform distribution can be estimated by:

This can be simplified to:

Where is the right bound of the range and is the left bound of the range. For our distribution the variance just comes out to:

We can find the error of any estimated value and its actual value with the equation:

Substituting in our mean and variance values for we can find the error between our estimations and their exact values for a continuous uniform distribution over the same range.

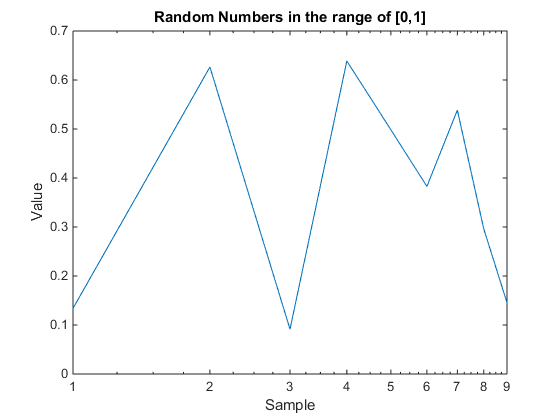
Additionally for the second part of this assignment we had to estimate the PDF of our random variables and compare it to the actual PDF determined by:

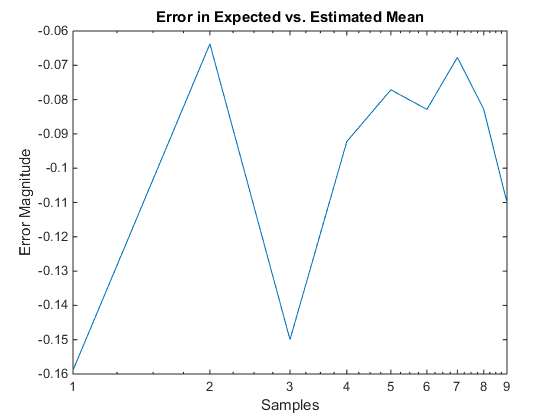
Where is the total number of samples since each sample is equally likely in an ideal continuous uniform distribution. To determine the accuracy of our estimated PDF we needed to find the mean squared error between our estimated and actual values by using the equation:

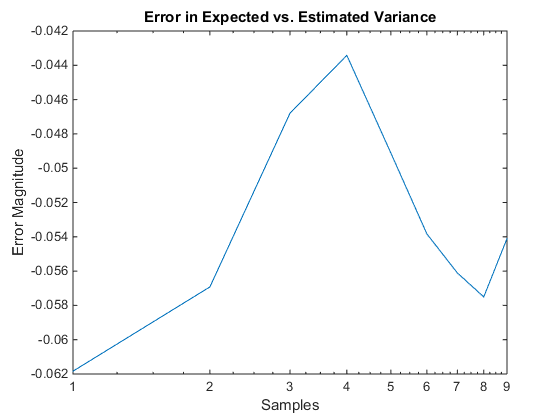
All of our calculations were performed progressively as the number of samples increased from one sample to the maximum number of points in the data set.

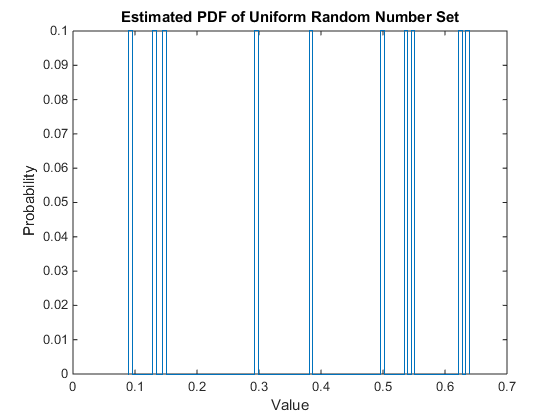
# MATLAB Code

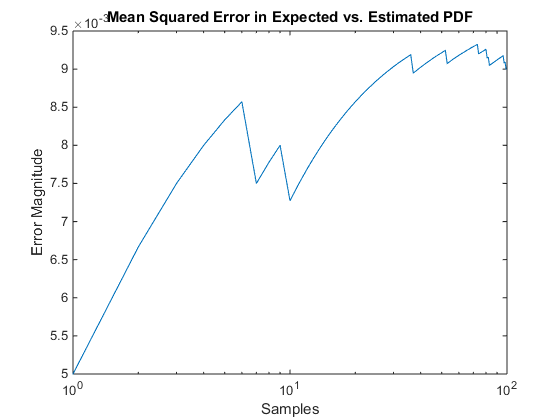
% Computer Assignment no. 6  
% Stochastic Processing  
% Bill O'Mullan  
  
function Stochastics\_ca\_6  
  
% Set up different range values as required in part two  
for A = [10, 1e3, 1e4]  
  
 % Generate uniformly distributed random numbers in the range [0,1]  
 rand\_num = generate\_rand('uniform',[1,A]);  
 figure(A-1);clf;  
 xaxis = 0:1:A-1;  
 semilogx(xaxis,rand\_num);  
 title('Random Numbers in the range of [0,1]')  
 xlabel('Sample');  
 ylabel('Value');  
  
 % Estimate the mean of the random number set we just generated  
 mu\_est = mean\_est(rand\_num);  
  
 % Estimate the variance of the random number set we just generated  
 sigma\_est = var\_est(rand\_num,mu\_est);  
  
 % Find the error in both the mean and the variance arrays  
 mu\_expect = 0.5;  
 sigma\_expect = 0.0833;  
 err\_mu = find\_error(mu\_est,mu\_expect);  
 err\_sigma = find\_error(sigma\_est,sigma\_expect);  
  
 % Plot the error with respect to the size of the array of random  
 % numbers in a log base ten scale  
 figure(A);clf;  
 x\_axis = 0:1:length(err\_mu)-1;  
 semilogx(x\_axis,err\_mu);  
 title('Error in Expected vs. Estimated Mean');  
 xlabel('Samples');  
 ylabel('Error Magnitude');  
  
 figure(A+1);clf;  
 x\_axis = 0:1:length(err\_sigma)-1;  
 semilogx(x\_axis,err\_sigma);  
 title('Error in Expected vs. Estimated Variance');  
 xlabel('Samples');  
 ylabel('Error Magnitude');  
  
 % Estimate the PDF using the histogram function with a bin size of 0.01  
 pdf\_est = get\_hist(rand\_num,A);  
  
 % Determine the expected y value for the uniform distribution PDF  
 % starting at zero and ending at one containing length(rand\_num)  
 % values  
 pdf\_exp = 1/length(rand\_num);  
  
 % Find the mean squared error in the pdf of the random values  
 mse = find\_mse(pdf\_est,pdf\_exp);  
  
 % Plot the MSE with respect to the number of bins in the PDF of the  
 % random numbers in a log base ten scale  
 figure(A+3);clf;  
 x\_axis = 0:1:length(mse)-1;  
 semilogx(x\_axis,mse);  
 title('Mean Squared Error in Expected vs. Estimated PDF');  
 xlabel('Samples');  
 ylabel('Error Magnitude');  
  
end  
  
end  
  
% This function generates an array of random numbers in the range of [0,1]  
% based on the input size and the type of distribution desired  
%  
% Input(s): size, random number generator type  
% Output(s): array of random numbers  
function x = generate\_rand(type,size)  
  
if(strcmp(type,'Uniform') || strcmp(type,'uniform'))  
  
 x = rand(size);  
  
elseif(strcmp(type,'Normal') || strcmp(type,'normal'))  
  
 x = randn(size);  
  
end  
  
end  
  
% This function progressively estimates the mean of a set uniformly  
% distributed numbers based on the equation:  
%  
% 1 N-1  
% mu = --- \* sum(x[n])  
% N n=0  
%  
% Input(s): random number array  
% Output(s): estimated mean  
function x = mean\_est(array)  
  
% Determine number of samples, N  
N = length(array);  
  
% Set up empty array for answer population  
x = zeros(1,N);  
  
% Estimate mean for progressively increasing numbers of samples of the  
% signal  
for n = 1:N  
  
 % Determine coefficient, 1/n  
 coeff = 1/n;  
  
 % Estimate mean up to n samples  
 x(n) = coeff\*sum(array(1:n));  
  
end  
  
end  
  
% This function progressively estimates the variance of a set uniformly  
% distributed numbers based on the equation:  
%  
% 1 N-1  
% sigma = --- \* sum{(x[n]-mu)^2}  
% N n=0  
%  
% Input(s): random number array, mean  
% Output(s): estimated variance  
function x = var\_est(array,mean)  
  
% Determine number of samples, N  
N = length(array);  
  
% Set up empty array for answer population  
x = zeros(1,N);  
  
% Estimate variance for progressively increasing numbers of samples of the  
% signal  
for n = 1:N  
  
 % Determine coefficient, 1/N  
 coeff = 1/n;  
  
 % Determine the argument, x[n]-mu  
 arg01 = array(1:n)-mean(1:n);  
  
 % Square the argument, (x[n]-mu)^2  
 arg02 = arg01.^2;  
  
 % Estimate the variance and output from function  
 x(n) = coeff.\*sum(arg02);  
  
end  
  
end  
  
% This function progressively finds the error between an estimated value  
% and the expected value using the equation:  
%  
% Err = est\_val - exp\_val  
%  
% Input(s): estimated value, expected value  
% Output(s): error between two values  
function x = find\_error(est,expect)  
  
% Set up empty array for answer population  
x = zeros(1,length(est));  
  
% Proceed through array of estimated values and compute error for each with  
% respect to expected value  
for n = 1:length(est)  
 x(n) = est(n)-expect;  
end  
  
end  
  
% This function finds the estimate of the probability distribution function  
% using the histogram function and a bin size of 0.01  
%  
% Input(s): data set, range upper limit  
% Output(s): pdf (graphical representation), pdf data  
function x = get\_hist(data,A)  
  
% Set the number of bins to be used as 100  
nbins = 100;  
  
% Create histogram and plot it on figure three  
figure(A+2);clf;  
hist = histogram(data, nbins);  
hist.Normalization = 'probability';  
hist.DisplayStyle = 'stairs';  
title('Estimated PDF of Uniform Random Number Set')  
xlabel('Value');  
ylabel('Probability');  
  
% Output y values of histogram to main function  
x = hist.Values;  
  
end  
  
% This function progressively finds the mean squared error between an  
% estimated value and the expected value using the equation:  
%  
% 1 B-1  
% Mse = --- \* sum{(est\_val(n) - exp\_val)^2}  
% B n=0  
%  
% Input(s): estimated value, expected value  
% Output(s): error between two values  
function x = find\_mse(est,expect)  
  
% Determine number of bins, B  
B = length(est);  
  
% Set up empty array for answer population  
x = zeros(1,B);  
  
% Proceed through array of estimated values and compute error for each with  
% respect to expected value  
for n = 1:B  
  
 % Find the coefficient, 1/B  
 coeff = 1/n;  
  
 % Find the first argument, est\_val(n) - exp\_val  
 arg01 = est(1:n) - expect;  
  
 % Find the second argument, arg01.^2  
 arg02 = arg01.^2;  
  
 % Estimate the MSE and output from function  
 x(n) = coeff\*sum(arg02);  
  
end  
  
end

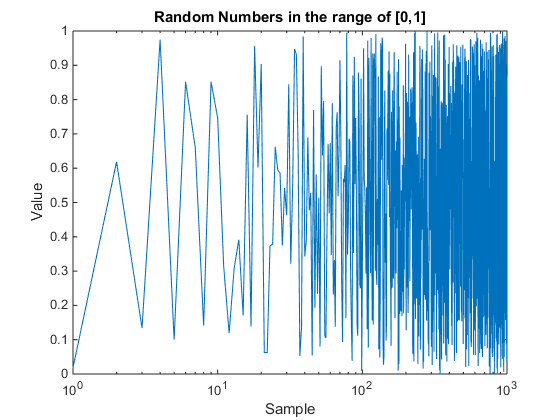


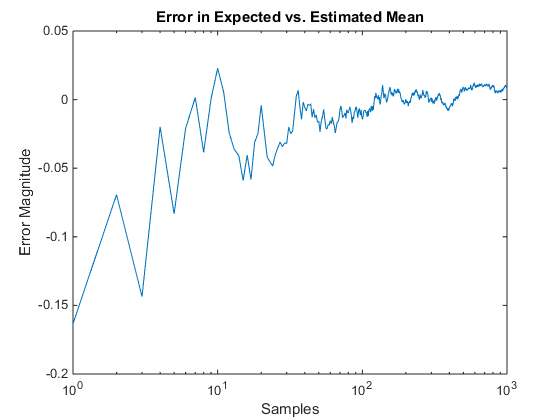


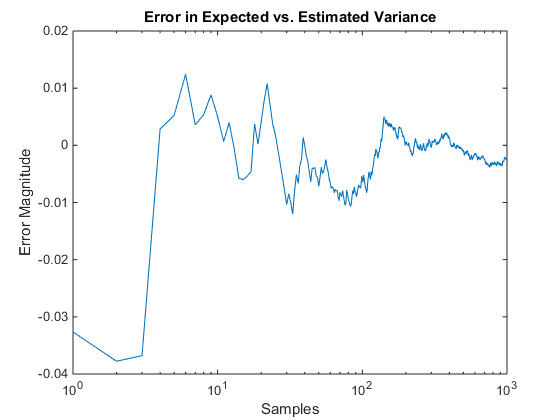


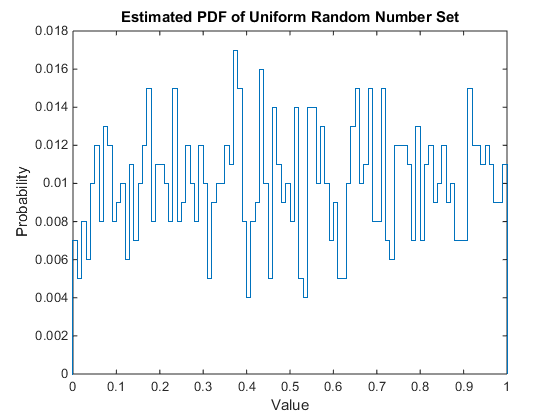


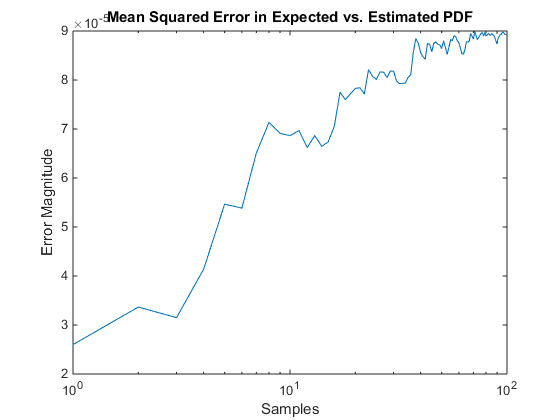


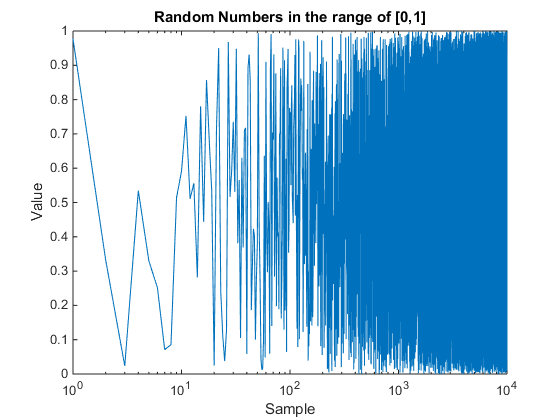


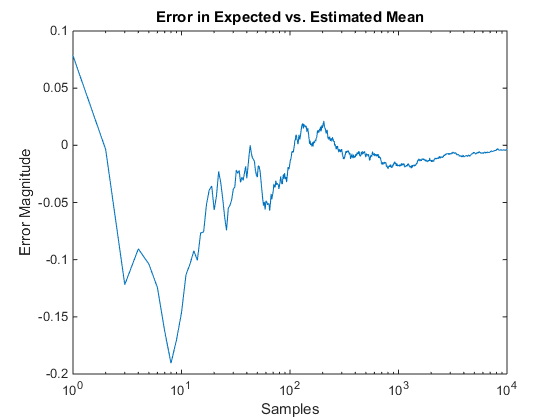


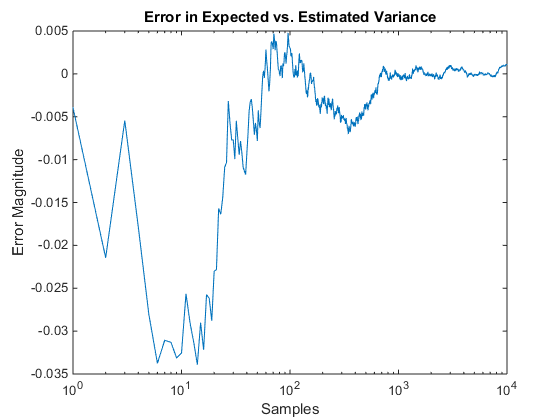


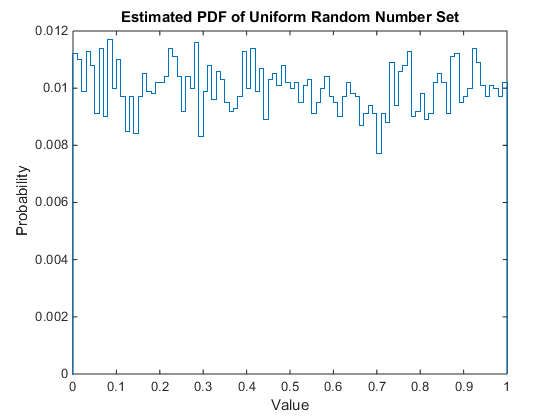


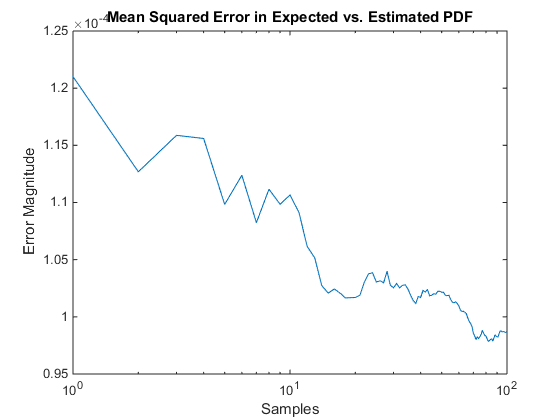












[*Published with MATLAB® R2014b*](http://www.mathworks.com/products/matlab)

# Conclusions

The overall conclusion that we can draw from this is that the more data points you have in a set the better. This is especially clear given the difference in error for the values found when we only used ten data points as opposed to the error from our calculations with ten thousand. For the graphs in which we used ten thousand samples, we can clearly see the error converge to zero as the amount of samples we calculated with increased. We can also see the difference in the PDFs for all the different number of data points. For the ten data point graph, the PDF barely looks like a uniform distribution whereas it starts to converge to the real PDF as the number of data points we use in our calculation increases.

A more practical example for this observation would be in the sampling rate of an incoming signal. A sampling rate that is too small will produce inconsistencies with the actual data we are trying to transmit due to the fact that there is not enough data points in the set to let us produce. Likewise a sampling rate that is too large will capture the noise of the signal and skew our mean and variance off making us see information that is not actually there. A sampling rate that is just right will produce enough data points that we can accurately model the statistical distribution of the incoming signal such that we can decode its message accurately.