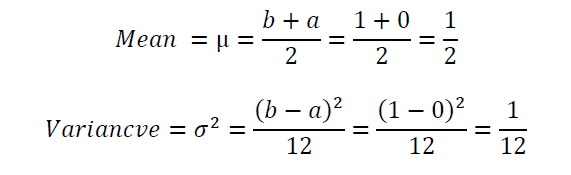
[Haiming Xu]

ECE 3512: Signals – Continuous and Discrete

Department of Electrical and Computer Engineering, Temple University, Philadelphia, PA 19122

# Problem Statement

Use MATLAB’s random number generator and generate uniformly distributed random numbers on the range [0,1]. We all agree that the mean, *μ*, should be 0.5, and the variance, σ2, should be 1/12. Using function.



Generate N random numbers, denoted by the signal x[n]. Estimate the mean using N data points. Compute the error of these estimates, then plot  and  for N=[1, 106]. Use a log base 10 scale for the horizontal axis (n – the number of points) and a linear scale for the error. Estimate the pdf of x[n] using a bin size of 0.01 (100 bins for the range [0,1]). Plot the mean-squared error between the measured distribution and the actual distribution Estimate the pdf for N = [1, 106], and plot MSE[N] using the same linear/log scale as above. Explain your findings. Plot the pdfs for N = 101, 103, and 106 and compare and contrast them. As we rarely know the parameters of these distributions or the functional forms of the distributions. We have to estimate these from the data and hope that our estimates are sufficiently accurate.

# Approach and Results

At first, MatLab’s random number generator appears to have biases that cause certain values to have more occurrences in our data array even though we expect a uniform distribution. Our equation defining uniform distribution do not correlate to our data until it reaches a sustainable size.

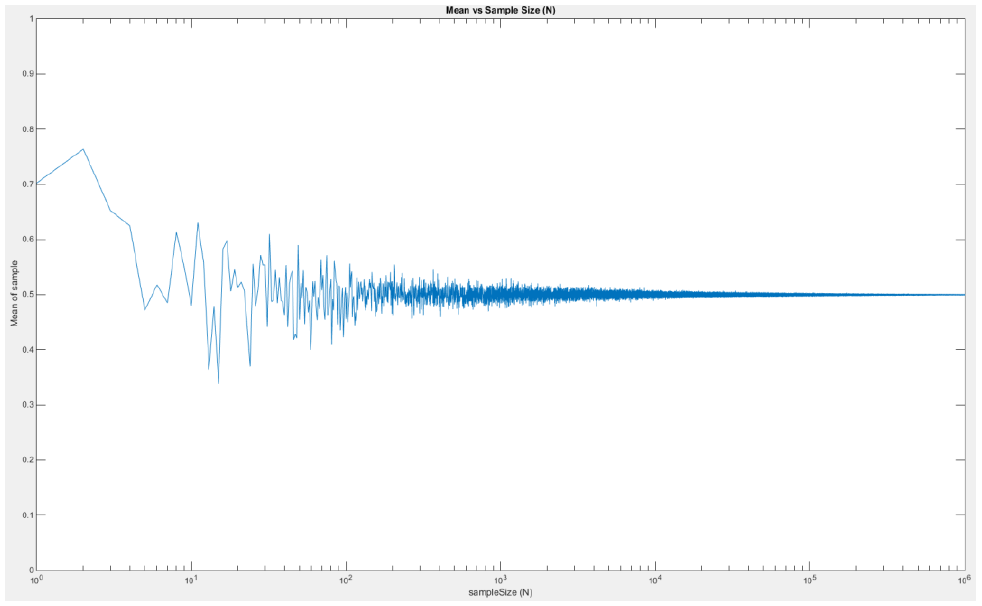


Figure 1: Actual mean of our data sets.

As is seen the Figure 1 when we increased our sample size closer to 1 million the mean of our data sets settled to the expected value of 0.5. In order first data set when N = 1 we see that our value was ~0.7. The next ten sample were all less that our original value which corresponds to a decreasing mean. The oscillation in our mean curve will continue occurring forever. If we zoom in to the last 1000 points in our plot (Figure 3) we observe our oscillation continues but we have successfully converged to our expected value.



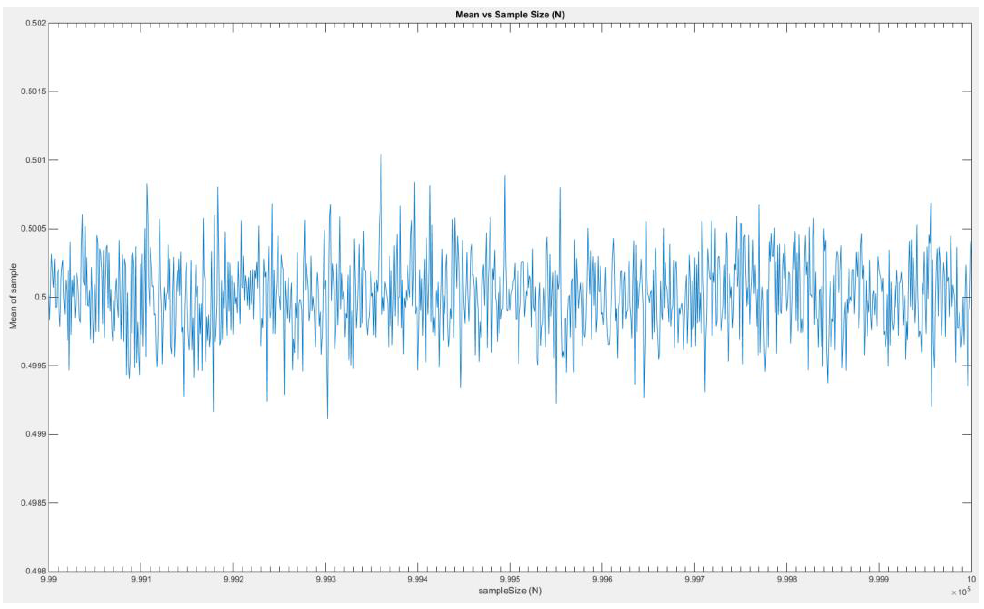


Figure 3: Actual mean of our data sets (x = [999,999 1,000,000]).

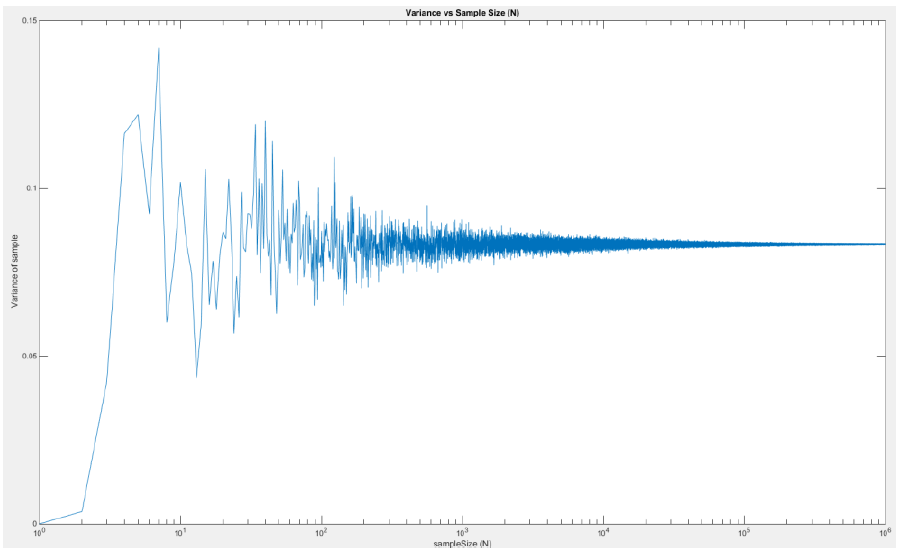
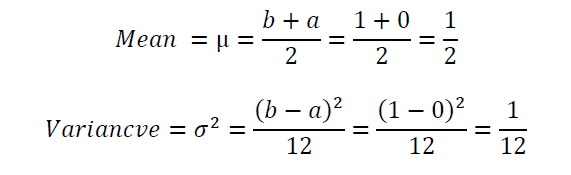
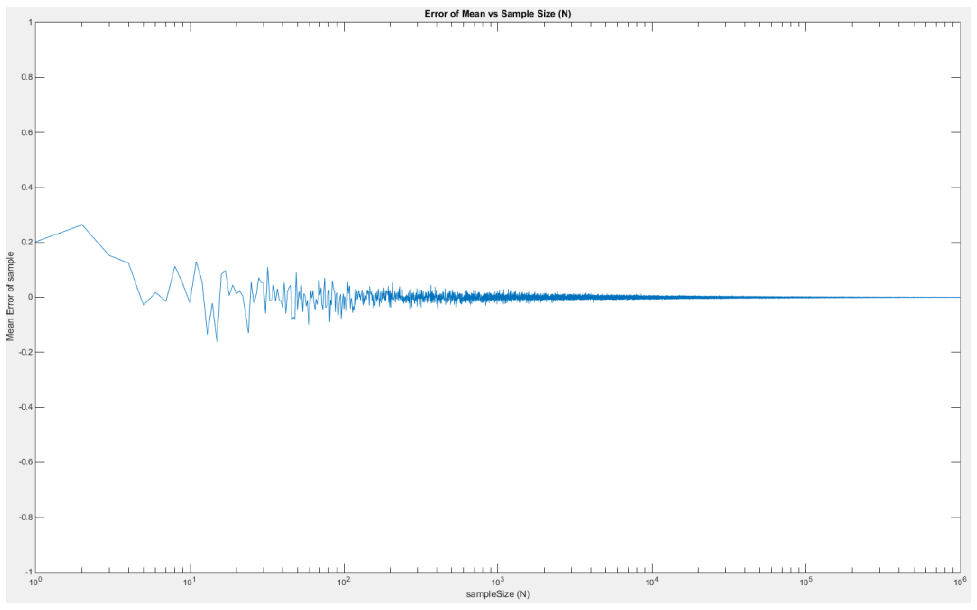


Figure 4: Actual variance of our data sets.

The characteristics seen with the mean will continue when looking at variance as well. We found the expected value of our variance to be 1/12= 0.083.

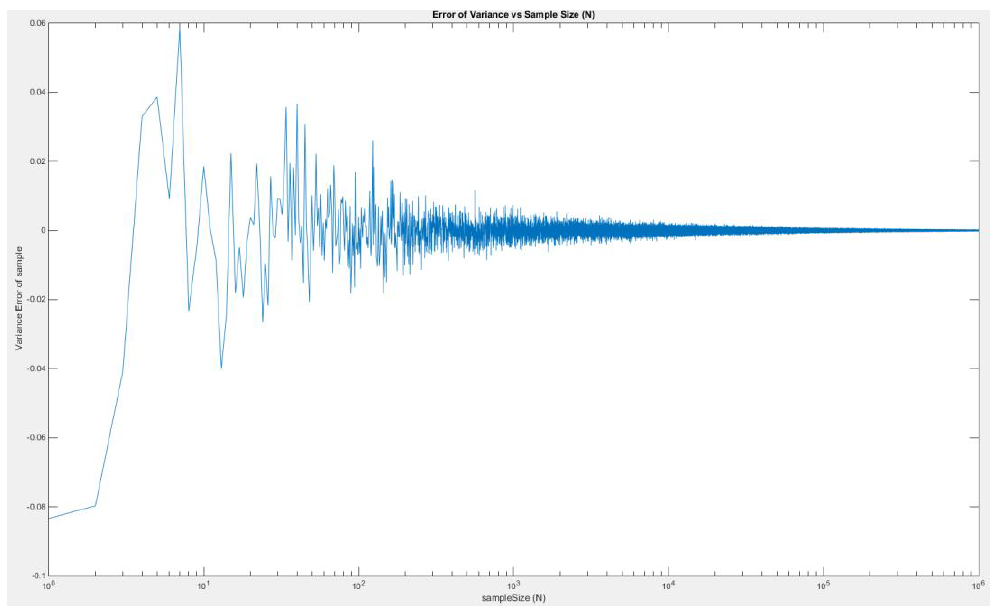


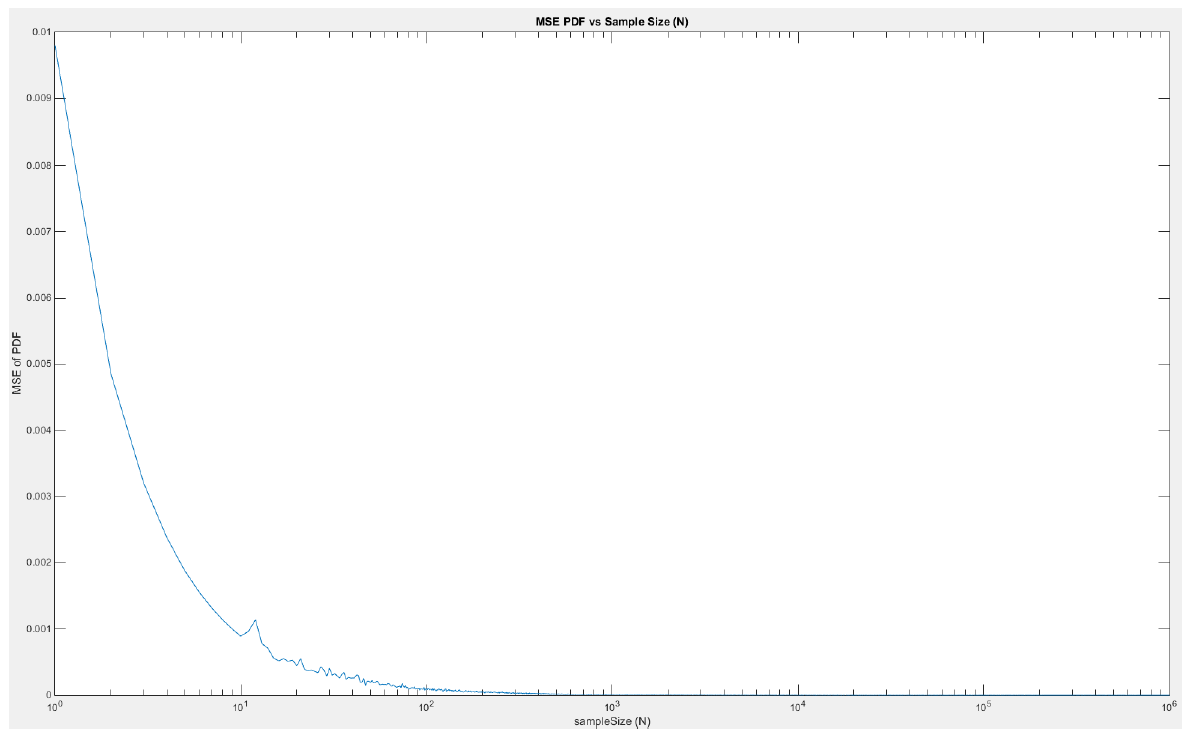
Like the mean we see how as we increase the sampleSize closer to our maximum of 1 million the variance converges to the true variance. The oscillation around the true variance still occurs



The error of our mean or variance when compared to the true values and plot the results. It appears that once our sameSize passes 30,000 the error is negligible.

.





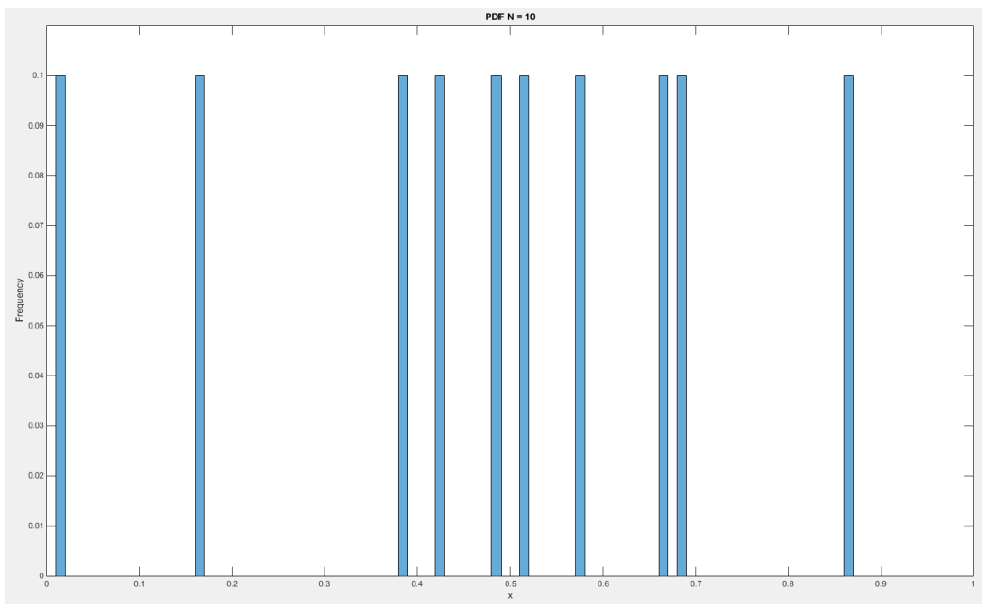


Figure 9: PDF when sampleSize = 10

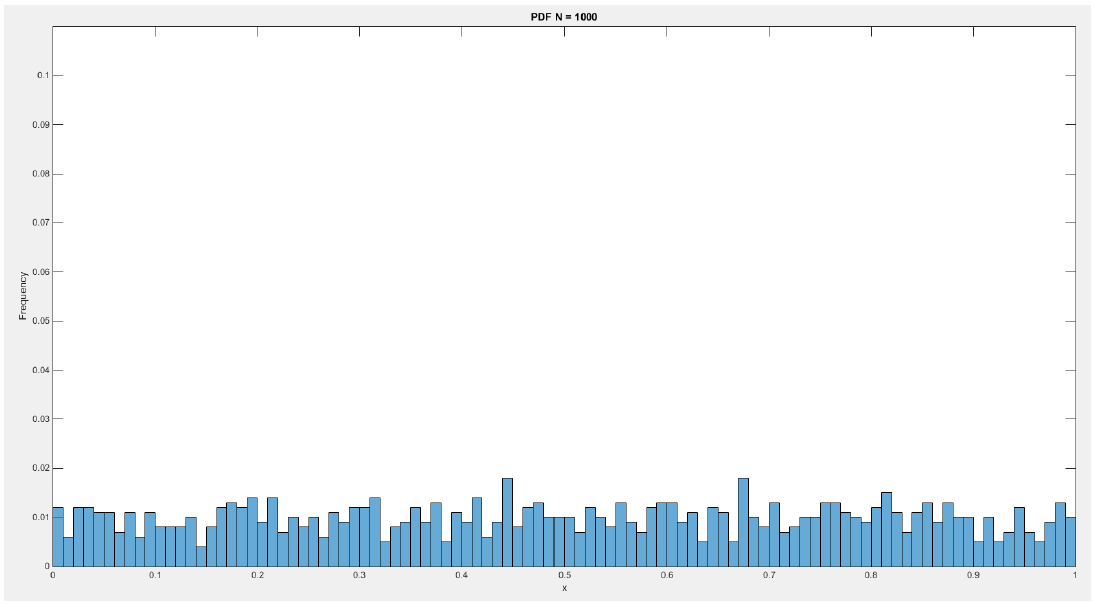
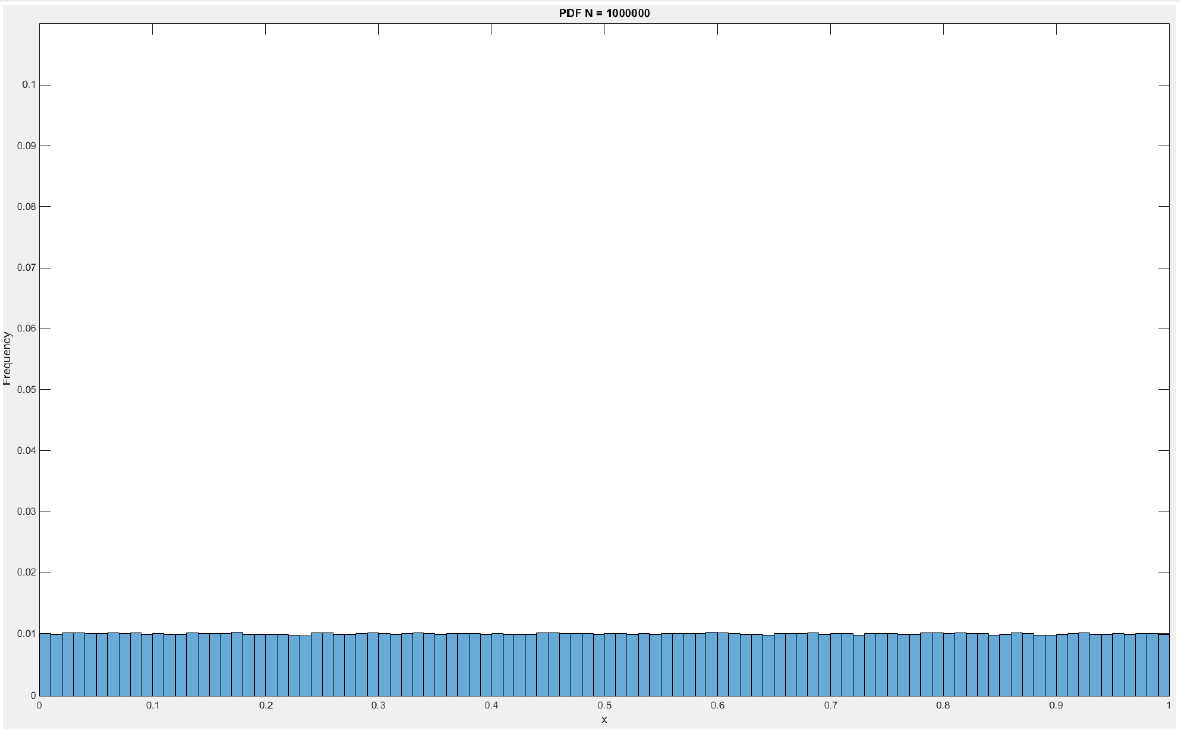
 

Figure 10: PDF when sampleSize = 1000 Figure 11: PDF when sampleSize = 1,000,000

Since we have a bin size of 0.01 the frequency that one bin occurs should be 0.01. As is observed in the plots above, when our sample size was 10 the frequency that each bin occurred had a frequency of 0.1. That frequency is far from the expected value of 0.01. However, as we increased our sample size (N) closer to a million our probability of each bin approached 0.01.

# MATLAB Code

%%

clear;

clc;

% How many points to evaluate through

uBound = 10^6;

% Lower and Upper bound of random Number Generation

lRange = 0;

uRange = 1;

% Excepted Mean/Variance

eMean = 0.5;

eVar = 1/12;

% Make a zero vector for our samArray

samMean = zeros(uBound, 1);

samVar = zeros(uBound, 1);

meanErr = zeros(uBound, 1);

varErr = zeros(uBound, 1);

% Histogram Bounds

bounds = [lRange:0.01:uRange];

numBins = length(bounds);

% For loop

for n=1:uBound

% Progress Bar

n

% Generate Random Array

samArray = lRange +(uRange-lRange)\*rand(n,1);

% Find mean and Variance

samMean(n) = mean(samArray);

samVar(n) = var(samArray);

meanErr(n) = samMean(n) - eMean;

varErr(n) = samVar(n) - eVar;

% Estimate the pdf

h\_samArray = histcounts(samArray, bounds, 'Normalization', 'pdf');

% Find MSE for the Kernel on GoogleClose

for k = 1:numBins-1

xt(k) = (h\_samArray(k)-1)^2;

end

MSE\_samArray(n) = sum(xt)/numBins;

end

%%

clc; close all;

disp('Done processing.');

the section of Plot all the things

figure('name','[ECE 3522] Class Assignment [6]');

semilogx(samMean); ylim([0 1]);

xlabel('sampleSize (N)')

ylabel('Mean of sample');

title('Mean vs Sample Size (N)')

figure('name','[ECE 3522] Class Assignment [6]');

semilogx(samVar);

xlabel('sampleSize (N)')

ylabel('Variance of sample');

title('Variance vs Sample Size (N)')

figure('name','[ECE 3522] Class Assignment [6]');

semilogx(meanErr);

ylim([-1 1]);

xlabel('sampleSize (N)')

ylabel('Mean Error of sample');

title('Error of Mean vs Sample Size (N)')

figure('name','[ECE 3522] Class Assignment [6]');

semilogx(varErr);

xlabel('sampleSize (N)')

ylabel('Variance Error of sample');

title('Error of Variance vs Sample Size (N)')

figure('name','[ECE 3522] Class Assignment [6]');

semilogx(MSE\_samArray);

xlabel('sampleSize (N)')

ylabel('MSE of PDF');

title('MSE PDF vs Sample Size (N)')

the section of Specific pdf Processing

% For loop 2

for n=[10^1 10^3 10^6]

% Progress Bar

n

% Generate Random Array

samArray = lRange +(uRange-lRange)\*rand(n,1);

% Find mean and Variance

% Estimate the pdf

figure('name','[ECE 3522] Class Assignment [6]');

h\_samArray = histogram(samArray, bounds, 'Normalization', 'probability');

title(sprintf('PDF N = %d', n));

xlabel('x');

ylabel('Frequency');

ylim([0 0.11]);

end

# Conclusions

The take away from the assignment is that our ideal cases do not always represent the data observed. Only when our sample size grew to significant sizes did we observe our actual data approach the expected values. We can therefore say that we can make better predictions on data as our sample size grows. Having too little data can make bad inferences on what we can expect next. In our MatLab’s analysis we created random variables 1 million samples in length. The observations I made in my plots are unique since every computer will have randomly chosen different values. For just a few samples my computer may have a bias to create certain numbers more often than someone else’s. However, he definition of a uniform distribution is that we should expect everyone to observe the same results where we approach an expected mean and variance.