Devin Trejo

**ECE 3522: Stochastic Processes in Signals and Systems**

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

# Problem Statement

Today we will take random numbers with lengths varying from 1 to 1 million points and compare their actual means/variances to their expected means/variances. The purpose of the assignment is to show that in the real world expected values are never actually observed. When we use our statistical models we are only finding an estimate to what the true values should be.

To begin we have our random array and compute its actual mean and variance. Our data’s magnitude will only range from 0 to 1. MatLab’s built in random number generator ideal will generate an array where each value has a uniform probability of occurring. The uniform distribution is defined by the following:



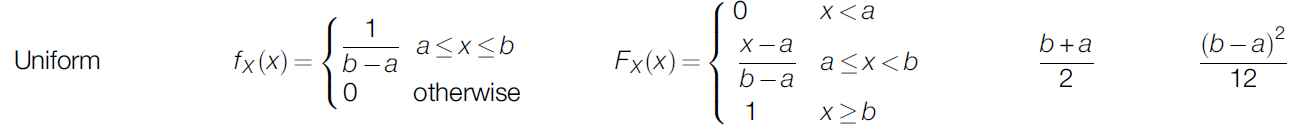


Figure : Uniform Distribution

Source: < [1, p. 149]>

Given our random variable range of [0,1] we expect the following values for the mean and variance:

|  |  |  |
| --- | --- | --- |
|  |  | (Equation ) |
|  |  | (Equation ) |

Now that we know our expected values, we will compare them to the values actually observed when MatLab’s creates random arrays ranging in size.

# Approach and Results

We first will see the actual mean and variance for our data sets as we range from 1 to 1 million.

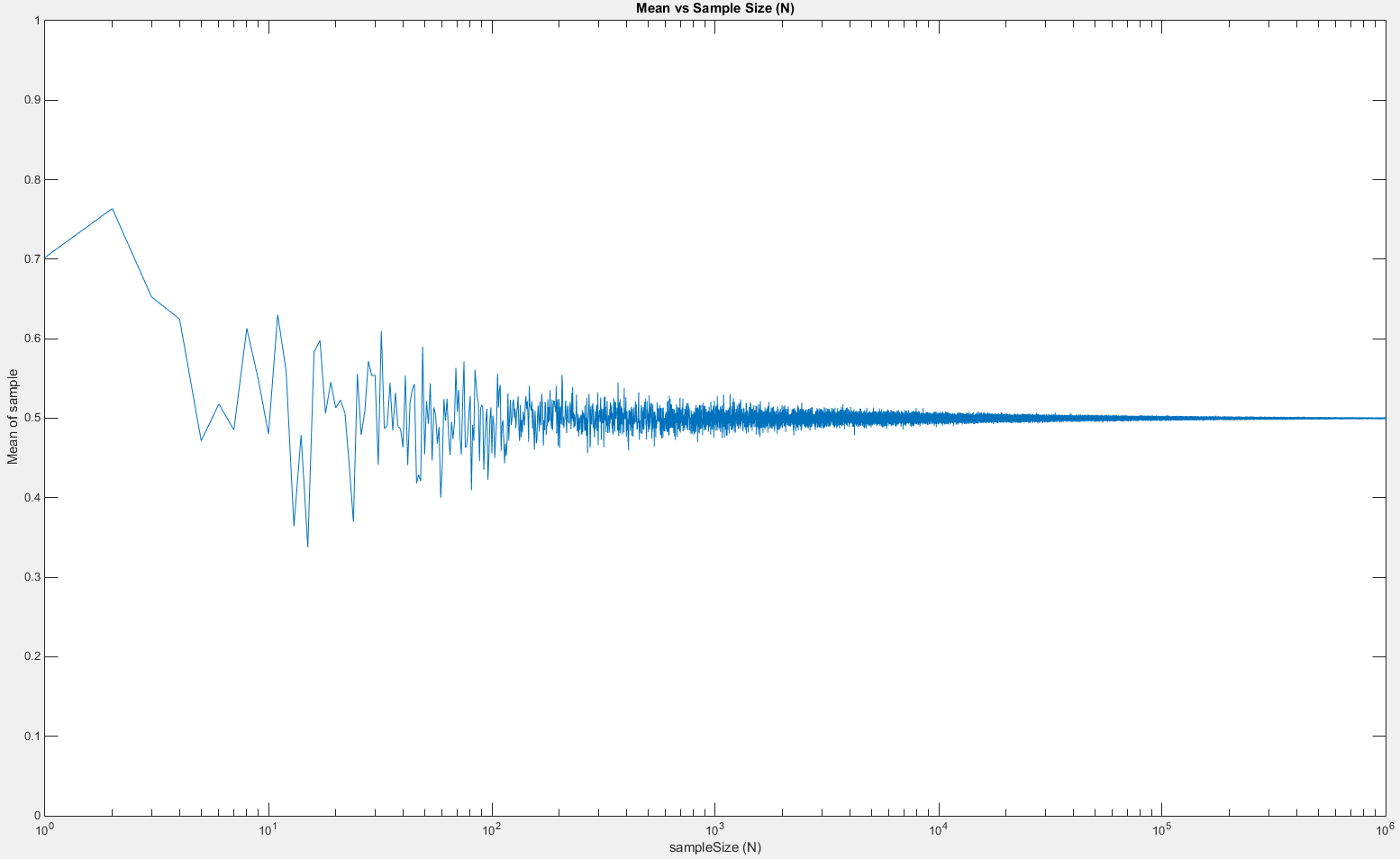


Figure 2: Actual mean of our data sets.

As is seen the Figure 2 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.

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 (Figure 1) do not correlate to our data until it reaches a sustainable size.

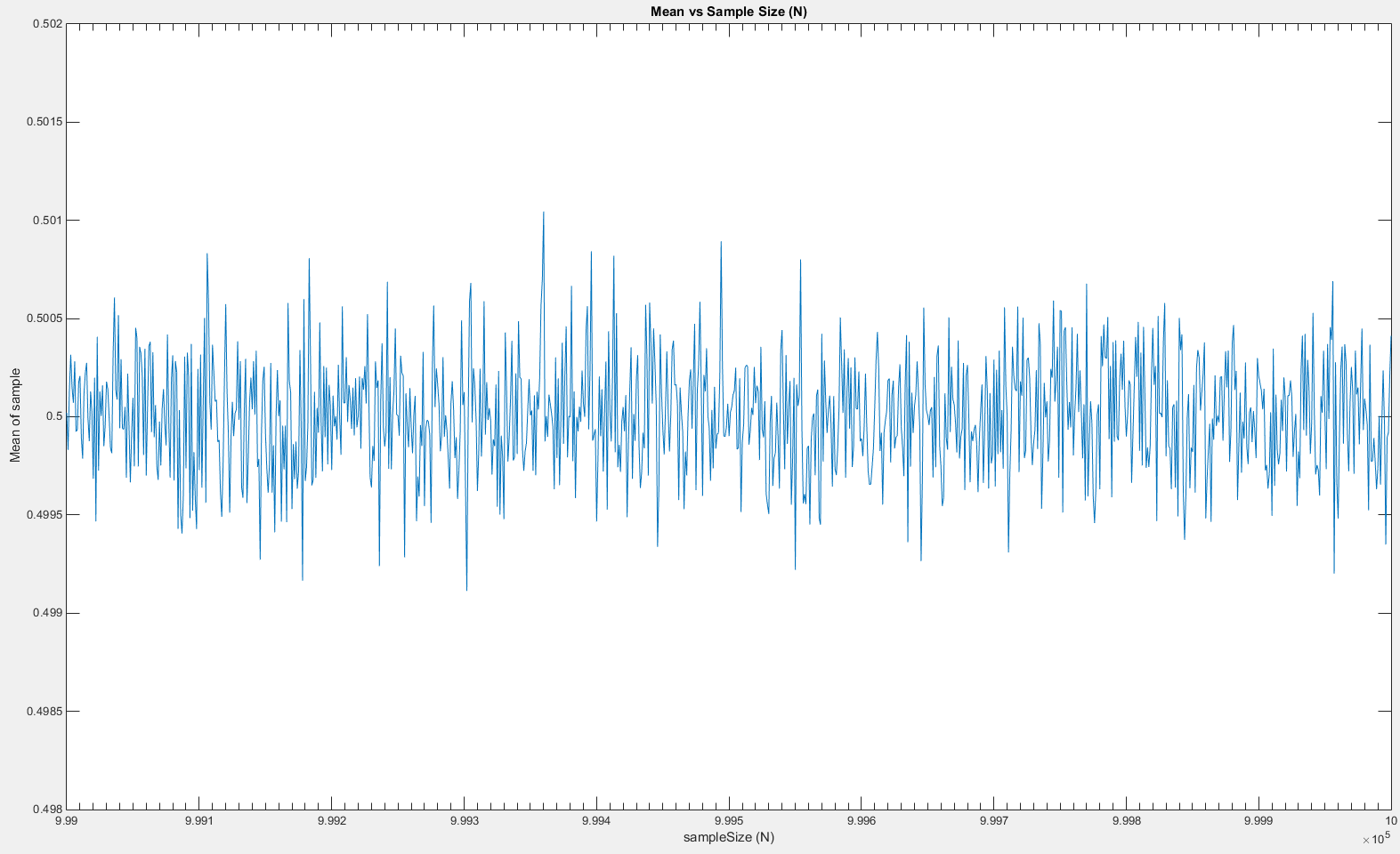


Figure : Actual mean of our data sets (x = [999,999 1,000,000]).

|  |  |
| --- | --- |
| Figure 4: Actual variance of our data sets. | Figure 5: Actual variance for the last 20 points. |

The characteristics seen with the mean will continue when looking at variance as well. In (Equation 2 we found the expected value of our variance to be . 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 (Figure 5).

We can find the error of our mean/variance when compared to the true values and plot the results. Upon glancing it appears that once our sameSize passes 30,000 the error is negligible.

|  |  |  |
| --- | --- | --- |
|  |  | (Equations 3/) |

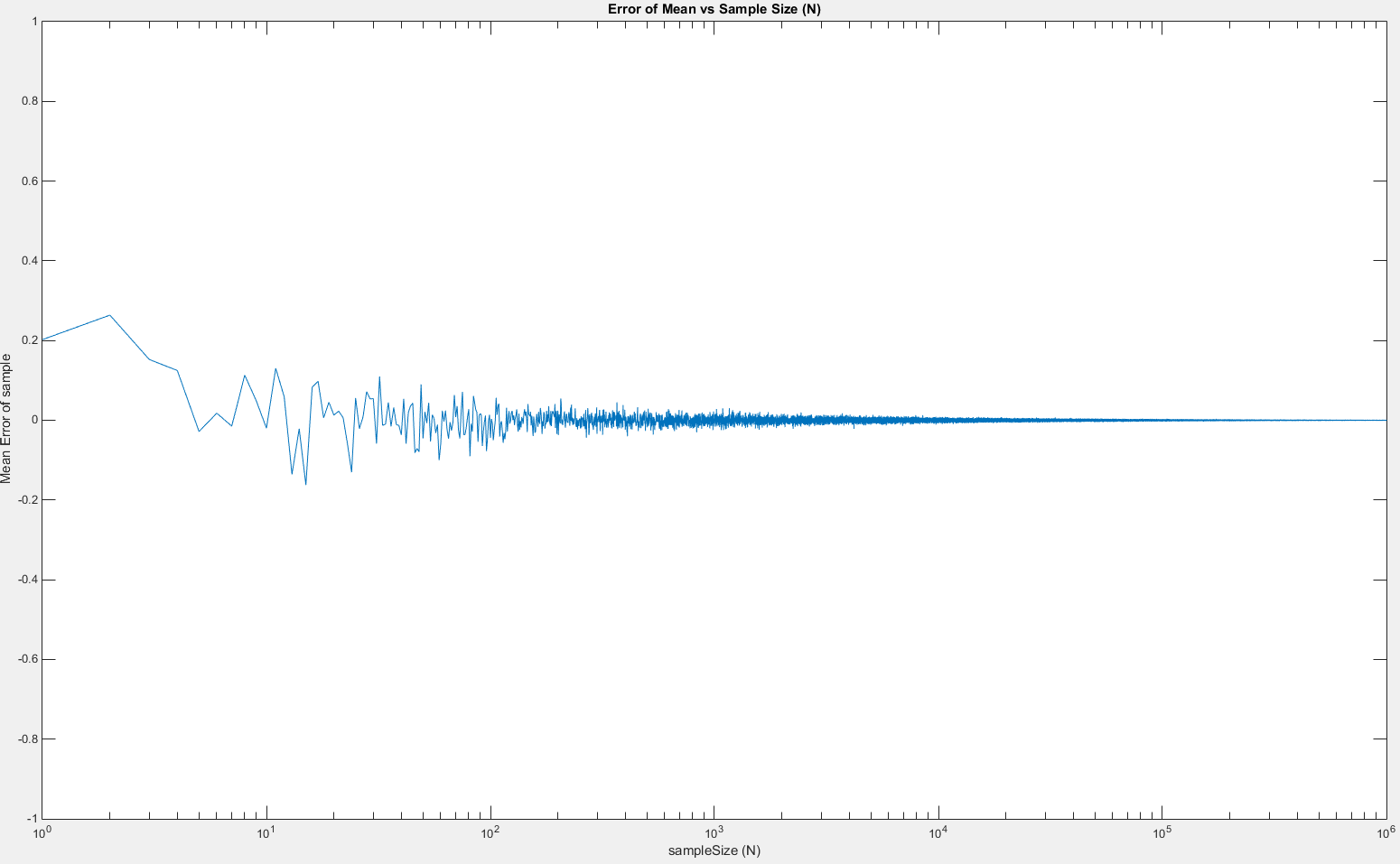


Figure : Error of observed mean compared to expected value.

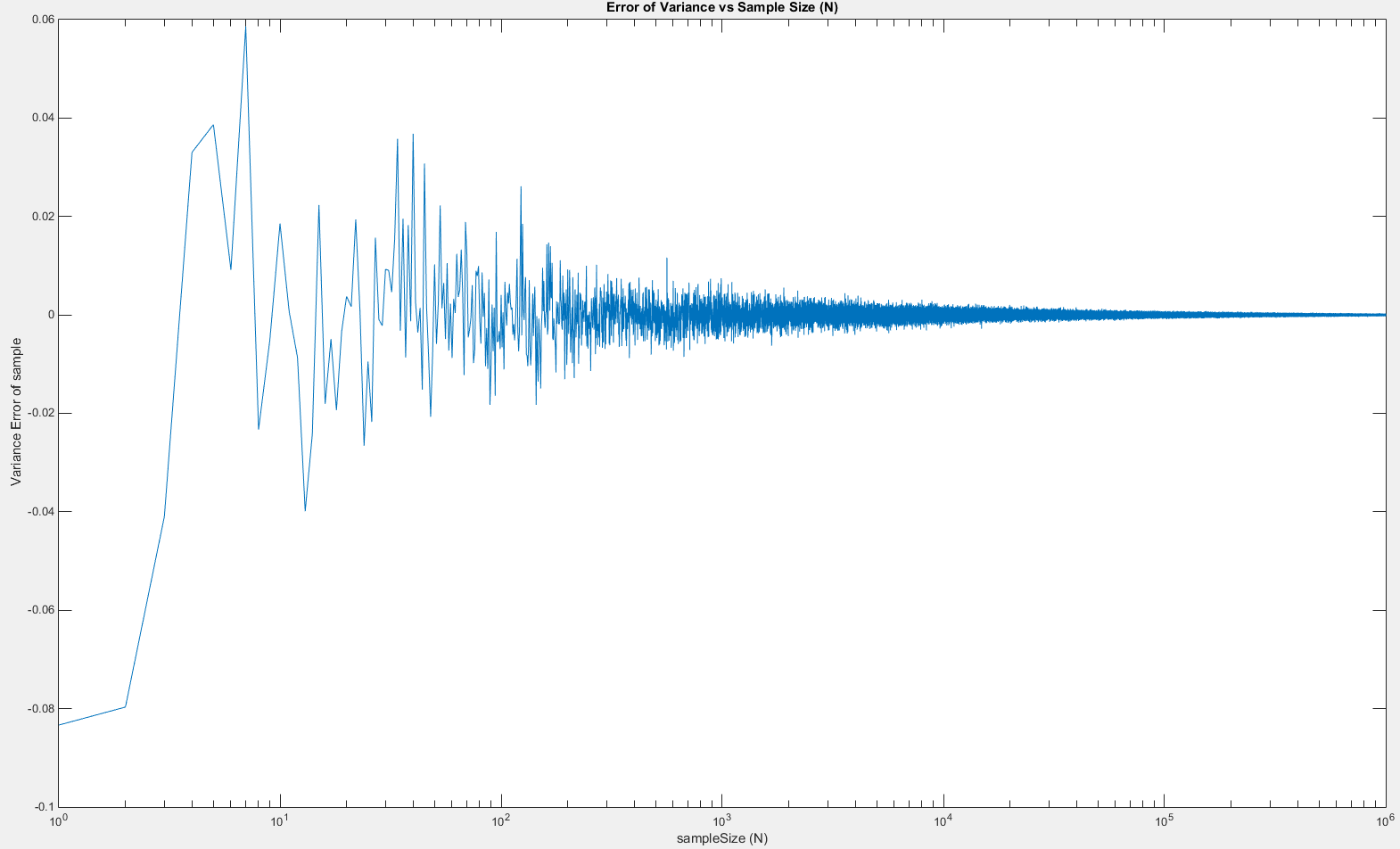


Figure : Error of observed variance compared to expected value.

Lastly we define the pdf of a uniform distribution as follows (from Figure 1).

|  |  |  |
| --- | --- | --- |
|  |  | (Equation ) |

The probability of any value in our sample array to occurring is equal. We can plot the mean squared error of our PDF how as we increase our sample size.

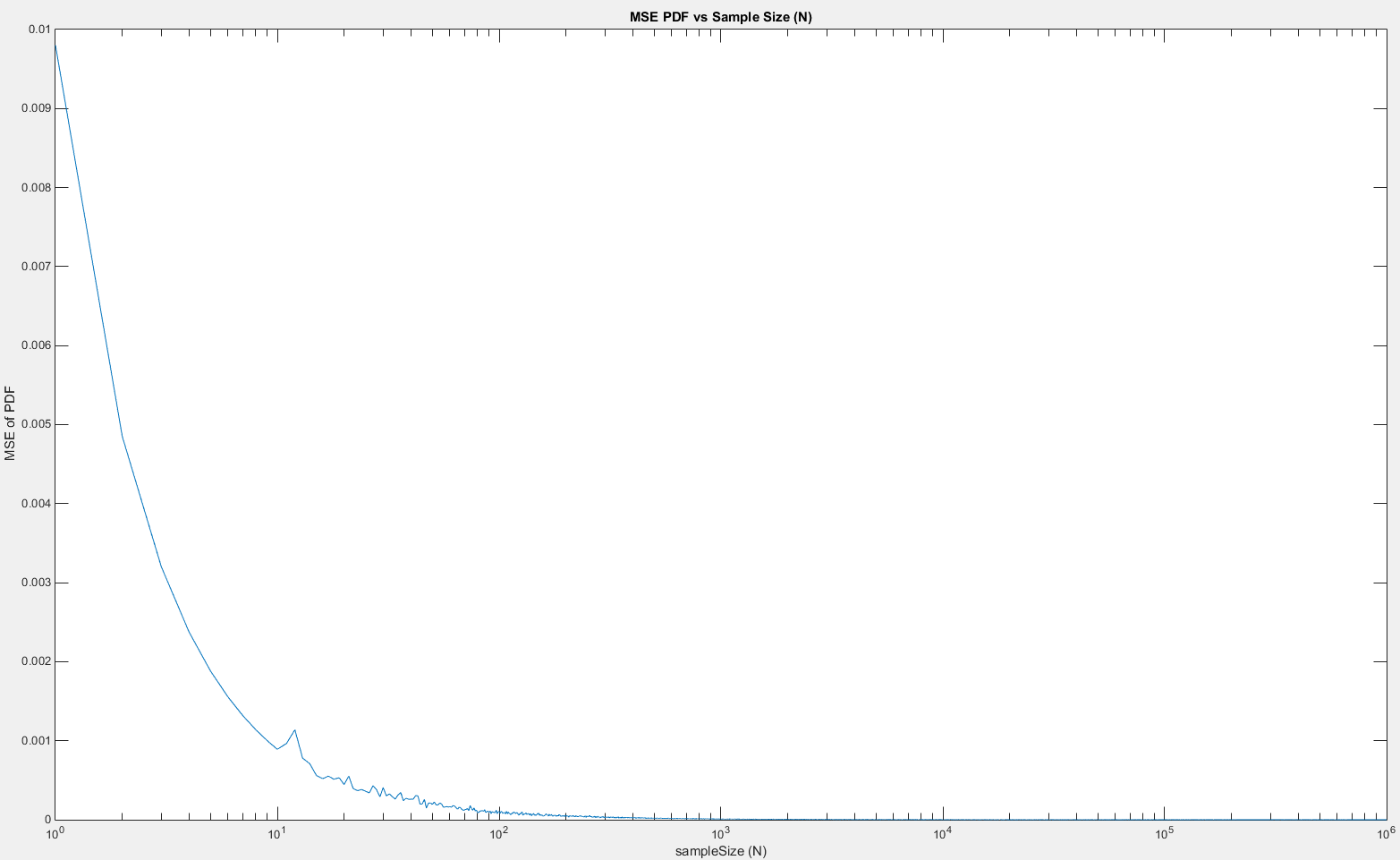


Figure : MSE of observed PDF compared to expected PDF.

As is presented in Figure 8 we observe again how the PDF error decreases as we increase our same size. We can take three different samples sizes to observe how the pdf reaches the expected value.

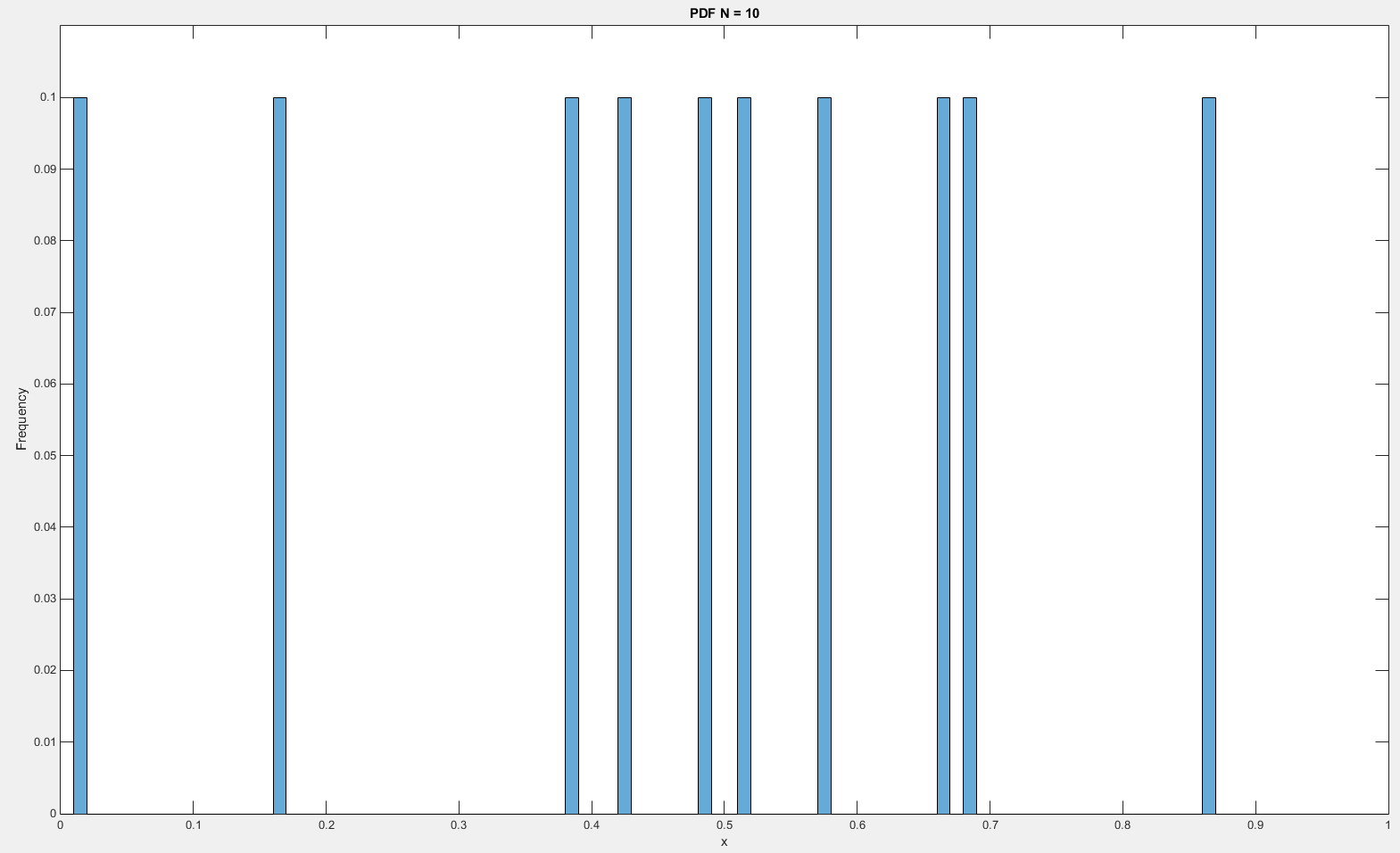


Figure : PDF when sampleSize = 10

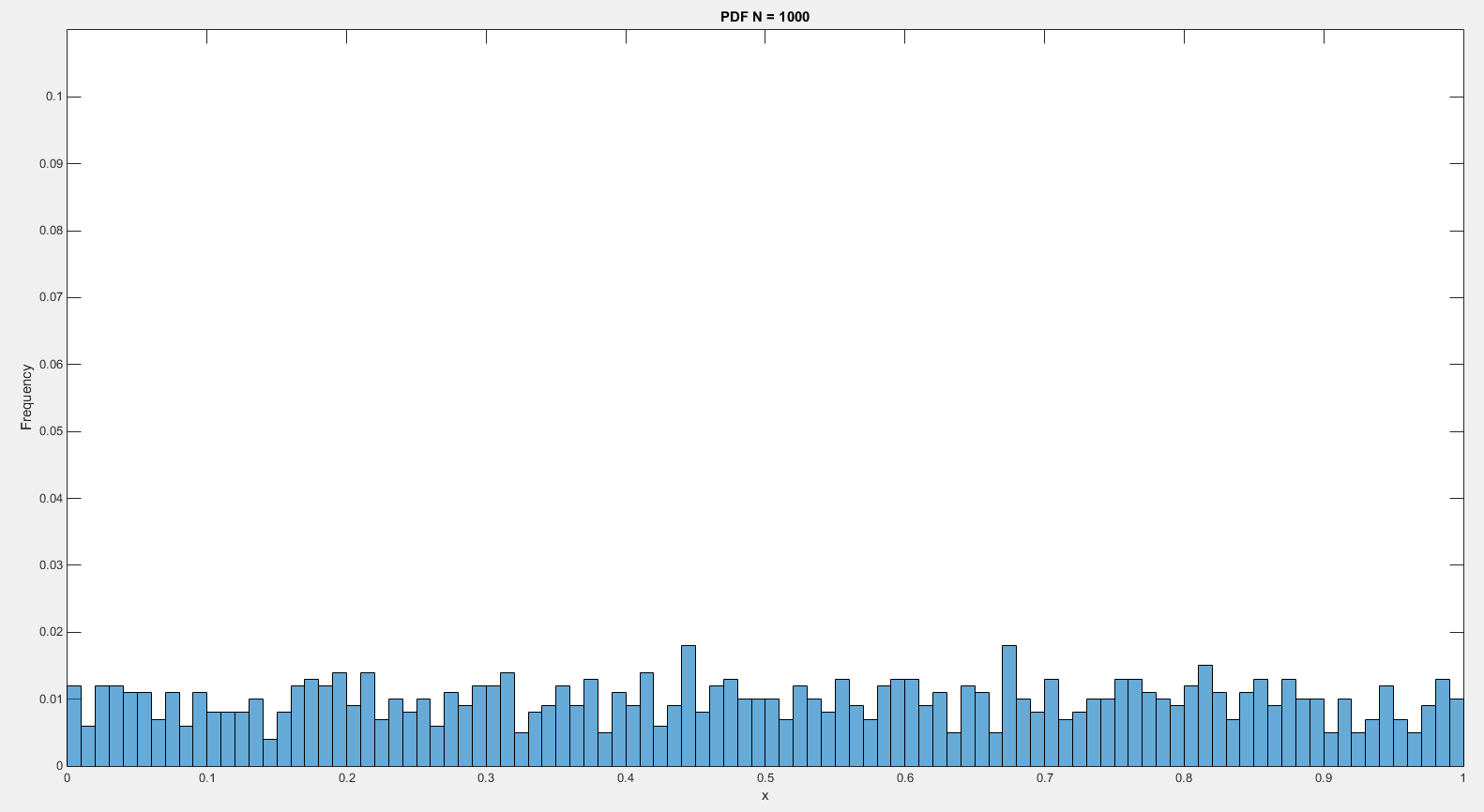


Figure : PDF when sampleSize = 1000

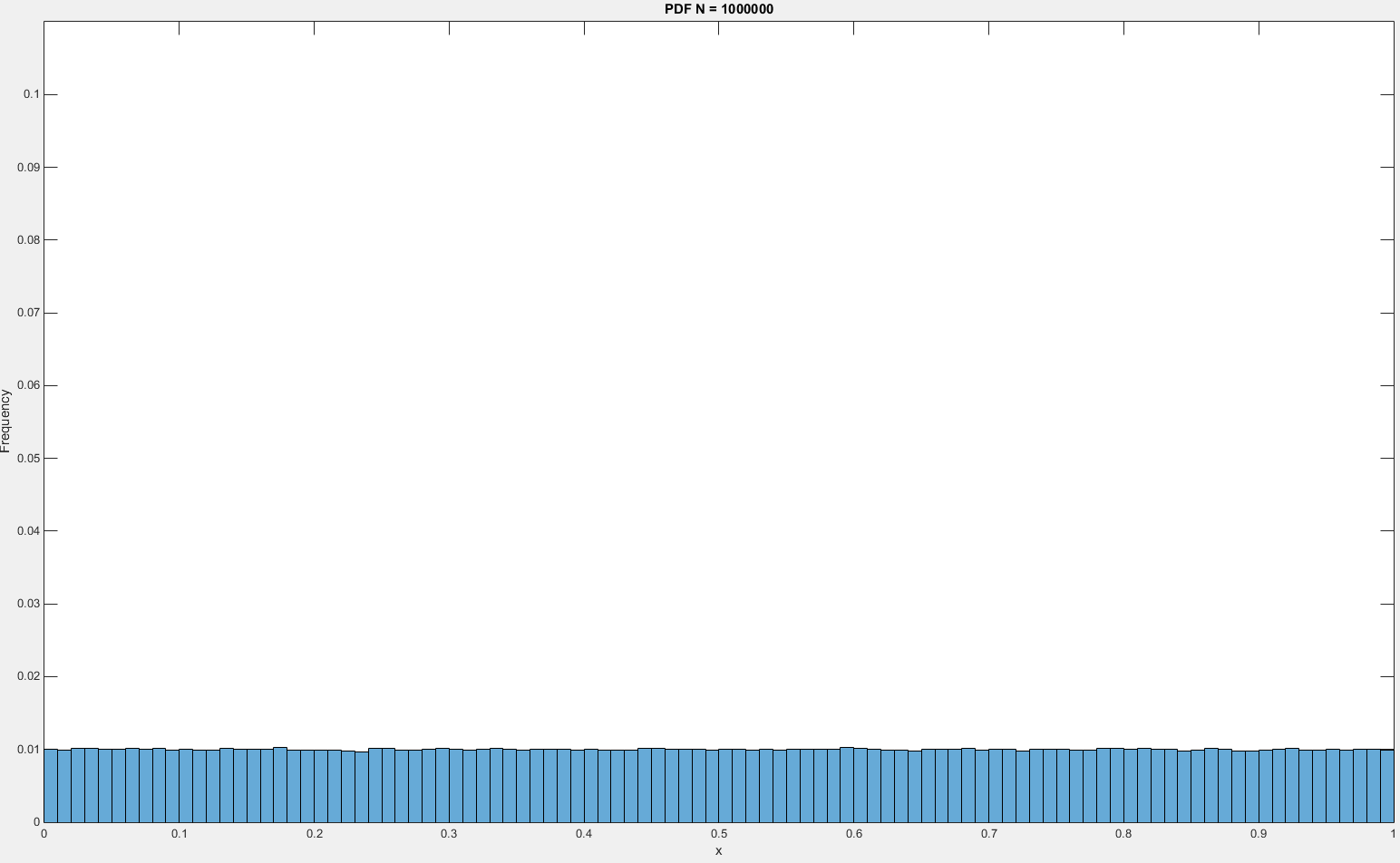


Figure : PDF when sampleSize = 1,000,000

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);

1 We setup our static variables, and set aside memory for the 1 million points we will compute the mean, variance, and mean/variance error. The histogram function ‘histcounts’ requires bin bounds which are defined by the variable bounds whose range is defined by our upper and lower value ranges.

% 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

2 Next we go into our main for-loop. Note the for-loop will loop through all million points defined in the assignment outline (may take several hours to run). We first print the variable counter ‘n’ to the console to keep track of where in our loop we currently reside. Now we create a new random variable set that increases depending on our array size ‘n’. Our values range from 0 to 1 so we must follow the guidelines seen in MatLab’s documentation ([LINK](http://www.mathworks.com/help/matlab/math/floating-point-numbers-within-specific-range.html)).

Using MatLab’s built in functions we find the mean, variance and the mean/variance error to our pre-defined expected values. Finally we find the histogram in order to compare it to the true pdf of a set of random variables ranging from 0 to 1.

%%

clc; close all;

disp('Done processing.');

%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)')

3 Once our main loop has finished computing all values necessary for the 1 million points we plot our results. We use the semilogx to plot our x values on a log scale. This plotting procedure allows us to clearly see the changes to our values along all 1 million points.

%% 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

4 Lastly we want to focus on the pdf for just a specific set of number sets. Specifically we let our random variable array equal N = 101, 103, and 106. We set a constant y-axis for comparison purposes.

# Conclusions

The take away from the assignment is that our ideal cases do not always represent the data observed. Given a small sample size we saw that our expected mean and variance were far from the actual values. If we were to design a system that referenced the expected values, we would end up with bad output. 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.

The estimates we have been using in the previous assignments for the mean and variance need to be noted that they are just estimates. It is an important factor to take into account since if a system of truly random anything can occur. 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, in the end we should expect everyone to observe the same results where we approach an expected mean and variance. That is the definition of a uniform distribution.

# References

|  |  |
| --- | --- |
| [1] | O. C. Ibe, Fundamentals of Applied Probability and Random Processes, vol. II, Elsevier’s Science & Technology, 2014. |