[Robert Irwin]

ECE 3512: Stochastics

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

# Problem Statement

In this assignment we will be experimenting with the central limit theorem. The central limit theorem states that if you add enough uniformly distributed, multivariate independent random variables, the sum will be normally distributed. We will be testing this hypothesis by generating sets of uniformly distributed, multivariate independent random variables using Matlab’s rand() function. The hypothesis will be tested both visually, (fitting a Gaussian to a pdf of the set) and using the error between the pdf and the Gaussian fit. Then we will experiment with the Muller method of uniform distribution to normal distribution transformation, and compare it to the first method with respect to quickness and mean squared error.

# Approach and Results

To generate the uniformly distributed, multivariate independent random variables a user defined function was created in Matlab. The function takes parameters, n, N, and R, where n represents how many vectors of random variables are generated, N represents how many samples in each vector, and R is a vector that defines the range of the random number generator. This function adds up all vectors and outputs the sum, which we hope is Gaussian.

The output of this function is then passed to another user defined function called rmsFIT(). This function takes any input signal X, and an indexing variable n. The indexing variable decides whether or not to plot the data with the Gaussian fit. The function plots the pdf and the Gaussian fit when n=1,10,100. This function also outputs the mean squared error between the pdf and the Gaussian fit, the mean of the signal being passed in and the variance of the signal being passed in.

Below are the figures of the pdfs and the Gaussian fit for n=1,n=10, and n=100.



*Figure 1: n = 1*

**

*Figure 2: n = 10*

**

*Figure 3: n = 100*

As seen above, when n gets larger the Gaussian fits the pdf much better. This is illustrated below in a plot of the mean squared error as a function of n.



*Figure 4: MSE as a Function of n*

Figure 4 agrees with the visual results seen in Figures 1-3. All of these results are consistent with the Central Limit Theorem. We see when n is small, i.e. 1, the mean squared error is fairly high. This makes sense because the mean squared error is calculated between a uniform distribution and a Gaussian distribution.

The Central Limit Theorem also states that if the range of the random variables being generated is constant, both the mean and variance should increases linearly as a function of n. This result is seen in the following plot.



*Figure 5: Mean and Variance*

Now we will look at the Muller approach and compare these to the results from the method above when n = 10. The results were as follows:



As we can see, the Muller method is both faster, and more accurate than the first method used in the testing of the Central Limit Theorem.

Then we experimented with the value of n to see if we could ever get comparable results between the two methods. We found that if n = 5, the time elapsed in both methods is similar, and the mean squared errors converge. The results are shown below.



# MATLAB Code

The results above were obtained using the following Matlab code.



*Figure 6: Main Script.*

The only thing to discuss in this code is the fact the we run the functions udmirv() and rmsFIT 100 times, and each time the mean squared error, mean, and variance, are stored in a variable that is indexed according to the current value of n.. The rest of this code deals with plotting, which will not be discussed.

It is also important to note that n can be set equal to 10 and R can be changed to [0 1] for part 4 of this assignment.

Next we look into udmirv(), which creates our normally distributed set by summing up “i” uniform distributions.



*Figure 7: Function udmirv()*

In this code we are generating n, Nx1 uniformly distributed variables using Matlab’s rand() function. These vectors are then added together to produce the output S.

Now we will look into the function rmsFIT().



*Figure 8: Function rmsFIT()*

In the first part of the function we find the mean and variance of the variable X, which is the signal input to the function. Then we sort the data from lowest to highest. This was done because through experimentation, it was found the Gaussian fit was better if the data was sorted first. Then we create the Gaussian fit using fitdist() and pdf(). After that is completed, we define an edges vector that is used in the construction of the pdf. This vector is defined as the minimum of the input signal to the maximum of the input signal with bin sizes of .1. Then we compute the pmf using the histogram function.



*Figure 8: Correlating the pdf and the Gaussian Fit*

In order to compute the mean squared error, the vectors that contain the pdf values and the Gaussian fit values have to be the same size. Inherently, this is not the case. Therefore, we have to correlate the two sets. Checking which is larger does this. If we divide the lengths and compare the quotient to 1 we can tell which is larger. This then dictates which of the vectors we have to down sample. The downsample() function in Matlab does this very well. Before we call this function, we have to tell it how to down sample. Passing the variable P, which is how many times the smaller function goes into the larger function does this. For example, if P = 2, gaus\_avg will be equal to every other value of the longer vector, which makes the vectors the same length. Then the mean squared error is computed and outputted back to the main workspace.



*Figure 9: Plotting the pdf and Gaussian Fit*

Now we will look at the code for the Muller Method.



*Figure 10: Muller Method*

The first thing done is generating the two independent uniformly distributed random variables. Then we pass these to Muller’s function, which is ­ . This Z is then multiplied by the standard deviation of Z and added to the mean of Z. This essentially moves the distribution with the number of times n it is run. The five multiplied by the mean and variance of Z can be explained if we look back to Figure 5. This figure shows that the mean and variance grow linearly, as a function of n, where the slope of the line is equal to n. Once Z is computed, we pass it to rmsFIT() which has already been explained.

# Conclusions

 In this experiment we were able to verify the Central Limit Theorem. It is clear that if we add up an infinite number of uniformly distributed sets, the result would be purely Gaussian. This is evident by the way the mean squared error goes to zero as n gets large. We also see that the mean and variance of the Gaussian will be equal to the mean and variance of the uniformly distributed set, multiplied by the number of uniformly distributed sets added together to form the Gaussian.