Computer Assignment (CA) No. 8:   
Central LiMIT Theorem

Truc Le

ECE 3512: Signals – Continuous and Discrete

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

# Problem Statement

For this computer assignment, we would have to create a function that generated a sum of n random variables. And this function would be iterated a total of 10,000 times for each trial. The trials would then be compared to each other to see how the compare to each other as well as a Gaussian pdf approximation. After the three trials of various numbers of random variables were completed, they were analyzed and compared. At last, the random numbers were again generated through the Box-Muller transform method and was compared to the uniformly distributed random numbers for accuracy as well as computational time.

# Approach and Results

First, we would have to write function to generate sums of 1, 10 and 100 random variables. These vectors were then processed for a mean and variance get a Gaussian plot later on. Using Matlab’s histcounts function, a histogram was received for the values of amplitude. The Gaussians were then created using makedist and then looped through to compute the root mean square error as shown in Figure 1.

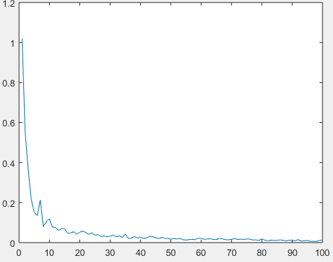


Figure 1: RMS Error for the Sums of One RV

From figure 1, we could see that the RMS error approaches zero over time as the difference between the two plots decreases towards the far end. If we repeated for a sum of 10 random variables we should expect that it will look less flat.

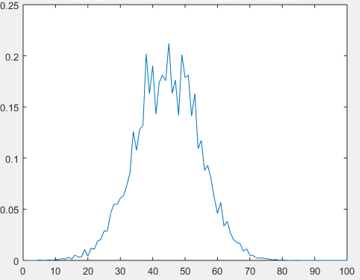


Figure 2: RMS Error for Sums of 10 RV

As we expected, the plot looks less flat. This is about what we would expect from the introduction of more random variables as the central limit theorem works. Next, we try the sums of 100 random variables.



Figure 3: RMS Error for Sums of 100 RV

Now that with 100 Random variables we see that the plot looks similar like a Gaussian. It is because that the root mean squared, RMS errors have peaks like so indicates that the predicted Gaussian and the histogram values have less in common than anticipated. Figure 4 to Figure 6 are the distribution plotted against the Gaussian fits that were calculated.

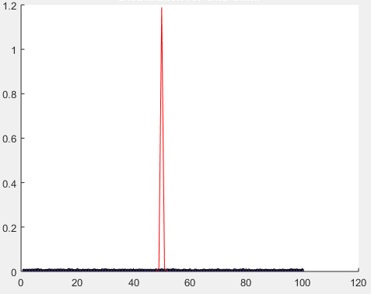


Figure 4: Gaussian fit for Sums of 1 RV

From the above figure, we can see the noisy blue line towards 0 is our normalized histogram of the sums of 1 random variable. This uniform-like behavior is expected as the sum of one number doesn’t have a lot of variance. The fitted Gaussian plot is extraordinarily peaked as it has to contain a lot of the area under the curve. This amplitude should decay with later fits and more random variables as shown in Fig. 5.

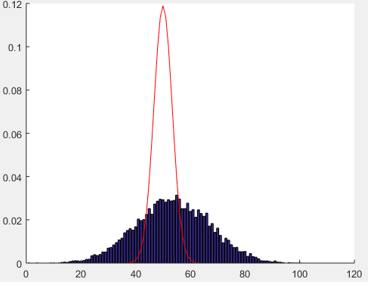


Figure 5: Gaussian fit for Sums of 10 RVs

From the above figure, with a sum of ten random variables we can see that the amplitude has decayed by a factor of approximately 10. We could noticed that the histogram looks less like a uniform distribution and more like a Gaussian, but it couldn’t be compare to a perfect Gaussian fit.

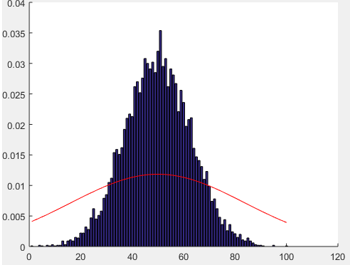


Figure 6: Gaussian fit for Sums of 100 RVs

From Figure 6, now with the sum of 100 random variables, it shows us that the peak of the histogram surpasses the Gaussian fit. While the mean of all of these plots has remained the same, the variances has been increasing with a higher number of random variables introduced. Increasing the amount of samples within the standard deviation and would therefore affecting the variance. We then had to use the Box Muller method to generate Normal Random Numbers. The same processes were done to this data as well to get a histogram and RMS error shown in Figure 7.

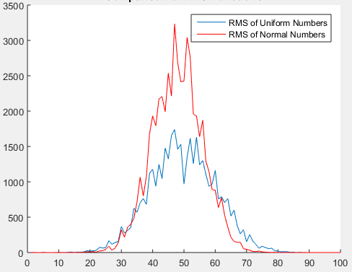


Figure 7: Comparison of RMS Functions

The RMS error for the uniformly generated numbers is less than the error of normally generated numbers. The computational time on both of these was also different. The Uniform numbers took about 0.17 seconds, while the Normal numbers took 0.25 seconds on average. When the sum of normal numbers was involving 1 to 5 we would see a reduction in the difference between the errors as well as the computational time.

# MATLAB Code

clear; clc; clf; close all;

n = [1, 10, 100];

N = 10000;

range = [-1, 1];

nbins = 100;

%Zero Vectors

rms1 = zeros(1, nbins);

rms2 = zeros(1, nbins);

rms3 = zeros(1, nbins);

%Running Sum Function

S1 = CA8(n(1), N, range);

S2 = CA8(n(2), N, range);

S3 = CA8(n(3), N, range);

%For a Sum of 1, 10 and 100 Random Numbers Respectively

mu1 = mean(S1);

sig1 = var(S1);

mu2 = mean(S2);

sig2 = var(S2);

mu3 = mean(S3);

sig3 = var(S3);

%Distributions

hist1 = histcounts(S1, nbins);

hist2 = histcounts(S2, nbins);

hist3 = histcounts(S3, nbins);

%Gaussians

x = ((-nbins/2)+1):1:(nbins/2);

gaus1 = pdf(makedist('Normal', 'mu', mu1, 'sigma', sig1), x);

gaus2 = pdf(makedist('Normal', 'mu', mu2, 'sigma', sig2), x);

gaus3 = pdf(makedist('Normal', 'mu', mu3, 'sigma', sig3), x);

%RMS Error Calculations

for a = 1:1:nbins

rms1(a) = 1/a \* (gaus1(a) - hist1(a))^2;

rms2(a) = 1/a \* (gaus2(a) - hist2(a))^2;

rms3(a) = 1/a \* (gaus3(a) - hist3(a))^2;

end

%Box-Muller Transform

tic;

BMmu = 0;

BMsig = 1/12;

BMrms = zeros(1, nbins);

BM = (normrnd(BMmu, BMsig, [1, N]))/2 + 0.5;

BMhist = histcounts(BM, nbins);

t = ((-nbins/2)+1):1:(nbins/2);

BMgaus = pdf(makedist('Normal', 'mu', BMmu, 'sigma', BMsig), t);

for a = 1:1:nbins

BMrms(a) = 1/a \* (BMgaus(a) - BMhist(a))^2;

end

toc;

figure(1);

hold on;

bar(BMhist./nbins);

plot(BMgaus, 'r');

hold off;

figure(2);

hold on;

plot(rms2);

plot(BMrms, 'r');

title('Comparison of RMS Functions');

legend('RMS of Uniform Numbers', 'RMS of Normal Numbers');

hold off;

%%%Plotting the RMS

figure(1);

plot(rms1./N);

title('RMS Error for Sums of 1 Random Variable');

figure(2);

plot(rms2./N);

title('RMS Error for Sums of 10 Random Variables');

figure(3);

plot(rms3./N);

title('RMS Error for Sums of 100 Random Variables');

figure(4);

hold on;

bar(hist1./N);

title('Distribution for One Sum:');

plot(gaus1, 'r');

hold off;

figure(5);

hold on;

bar(hist2./N);

title('Distribution for Ten Sums:');

plot(gaus2, 'r');

hold off;

figure(6);

hold on;

bar(hist3./N);

title('Distribution for One Hundred Sums:');

plot(gaus3, 'r');

hold off;

figure(7);

hold on;

plot(rms2);

plot(BMrms, 'r');

title('Comparison of RMS Functions');

legend('RMS of Uniform Numbers', 'RMS of Normal Numbers');

hold off;

function [S] = CA8(n, N, range)

%CA8 Takes n random variables and generates N

%Takes n random variables and generates N random numbers between a

%specified range. Returns the Sum, Mean and Variance

scale = range(2) - range(1);

S = zeros(1, N);

for i = 1:1:N

S(1, i) = sum(range(1) + scale .\* rand(1, n));

end

end

# Conclusions

As a result, the introduction of more random variables gets the plot closer to a Gaussian fit. This could also be visualized by convolving many uniform distributions together. Two constants net a linear relationship, which then net a parabolic relationship and the curves keep getting more and more defined until we ultimately get a true Gaussian fit when we convolve infinite uniform distributions together. We’ve also seen that uniform random numbers can be churned out more quickly than normal random numbers can be. Most likely because the process involved in generating normal random numbers has to take a lot more into account.