[Tyler Olivieri]

ECE 3512: Stochastic Processing in Signals and Systems

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

# Problem Statement

The central limit theorem states that a sum of independent random variables will approach a normal distribution. This concept will be explored with visualization and an error calculation between an ideal Gaussian and the pdf of the summed random variables. The mean and variance will be computed for each pdf generated. A different method will explore the same concept but with a different computational method called the box muller transform. The computation time and error will be explored.

# Approach and Results

Figure 1, Figure 2, and Figure 4 shows the Gaussian fit with n number of random variable summed. It can be observed that as n increases, the pdf approaches the Gaussian distribution. In Figure 5, it can be seen the MSE decreases as N increases. This states that pdf approaches the Gaussian fit as n increases. It can also be noted that the mean and variance of the pdf increases linearly with n. The variance will be the variance of the distribution being summed \*n. The variance can be seen to be 100 times bigger at n = 100 in Figure 6. The mean for the uniform distribution being used is 0, so the mean for all n is about 0. The axis is scaled so it appears to be a lot error and variation from zero but it is basically 0 for all n. This can be observed in Figure 7.

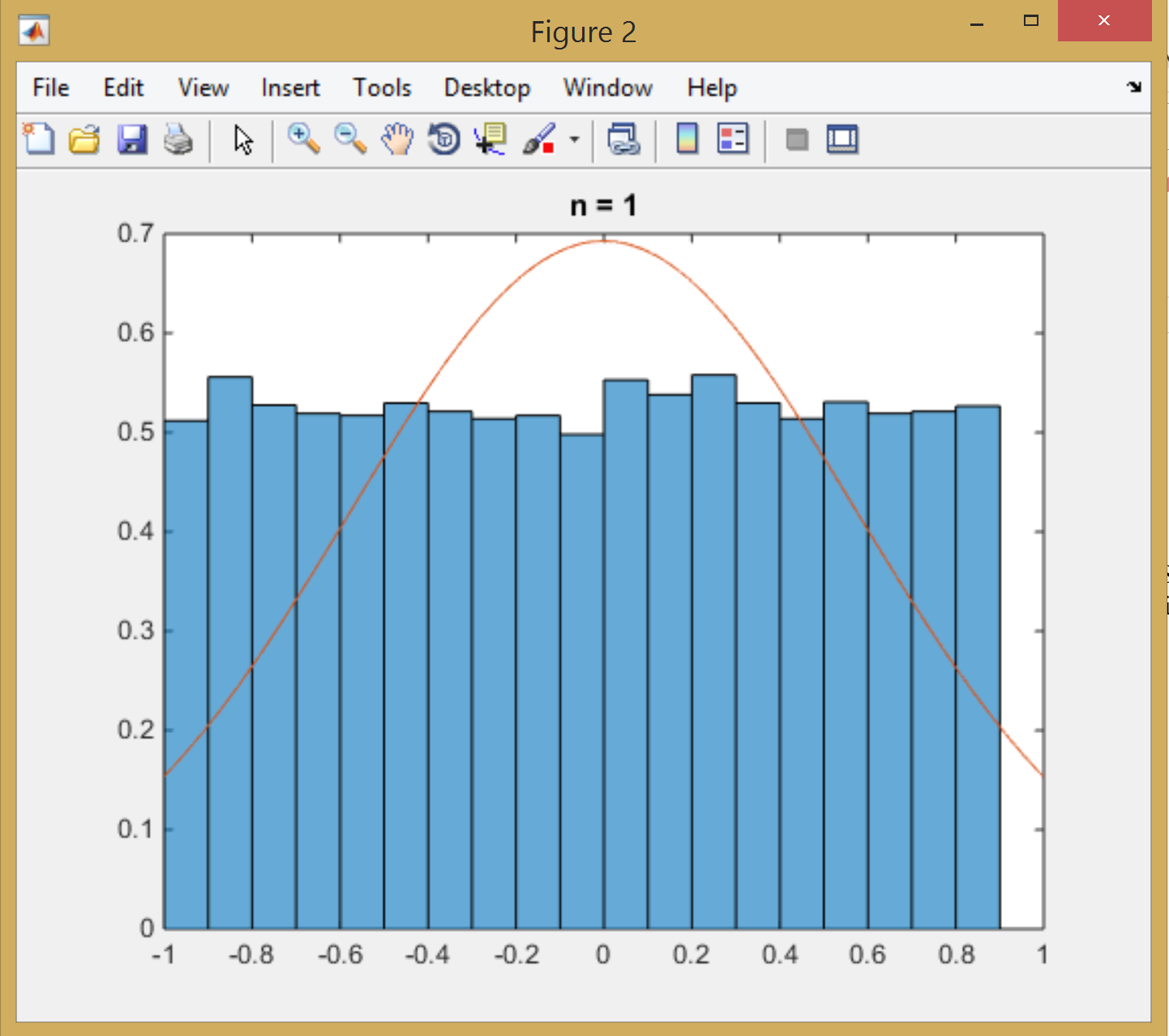


Figure 1: Gaussian and rvsum, n= 1

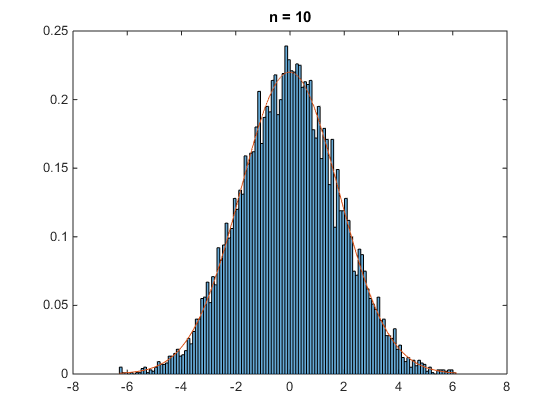


Figure 2: Gaussian fit with 10 RV summed

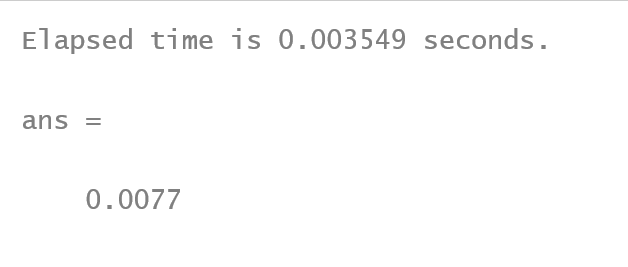


Figure 3: time elapsed and MSE to generate 10 RV sums

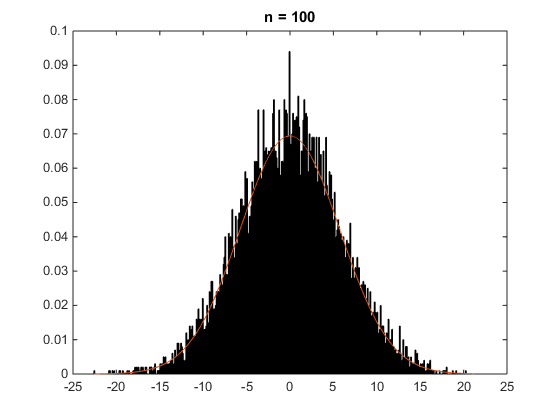


Figure 4: Gaussian fit with RV sum of n= 100

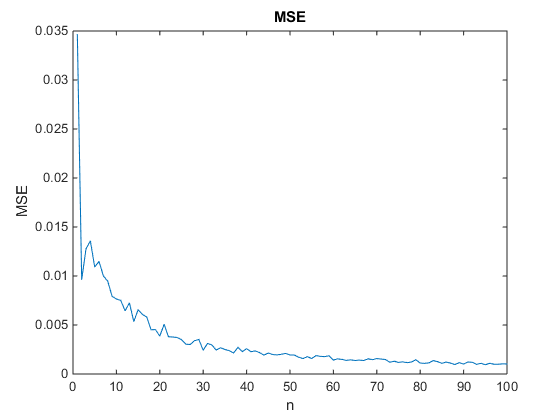


Figure 5: MSE for n= [1:1:100]

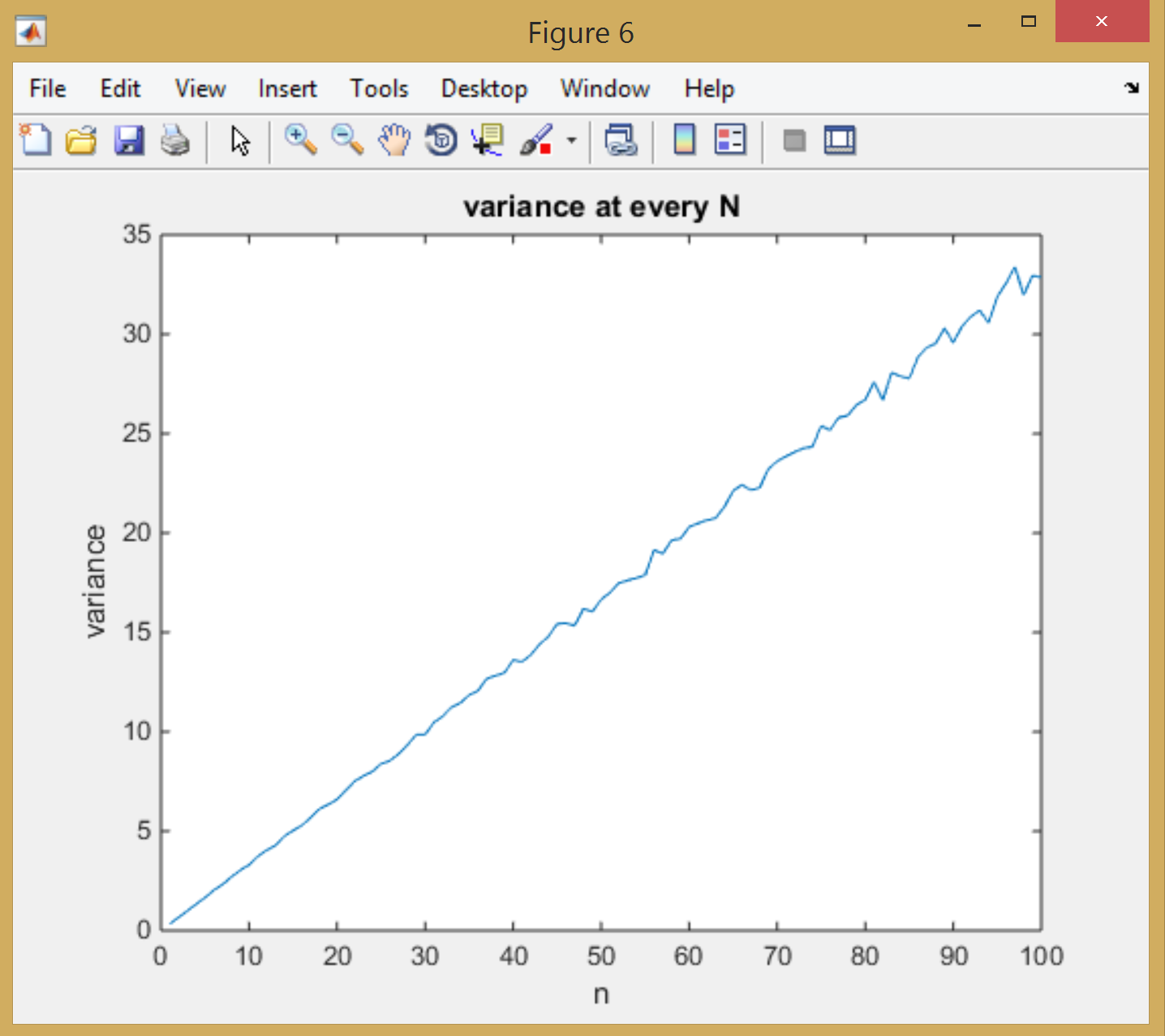


Figure 6: variance for every n

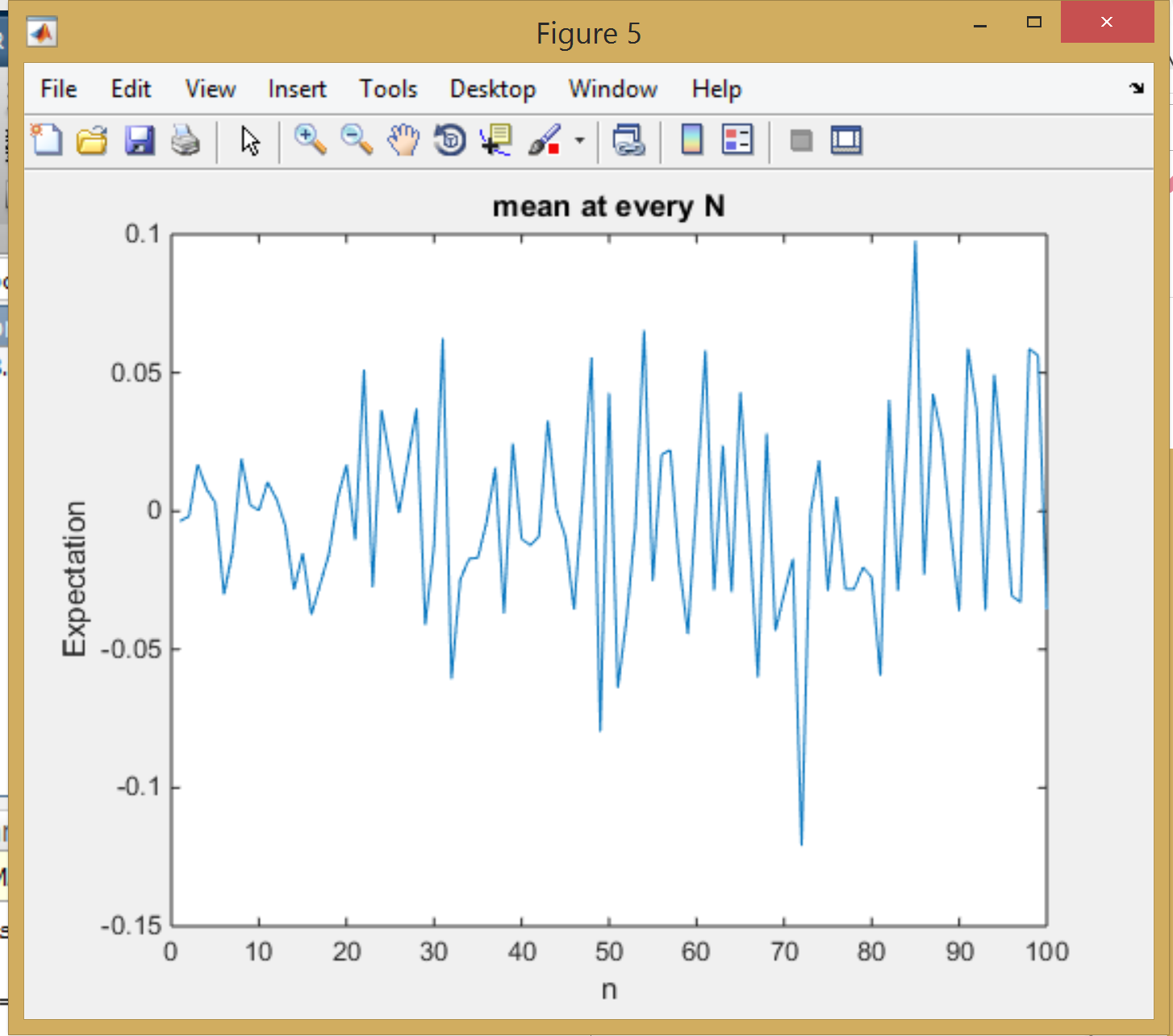


Figure 7: mean for all n

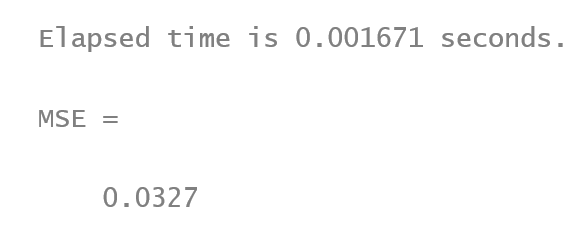


Figure 8: time to generate 10 RVsum using box muller and MSE

Figure 3 and Figure 8 shows the comparison of the time and MSE for n = 10. The box muller was faster but had higher MSE. The MSE difference is fairly small. At n = 6 a comparable computation time can be observed below but the MSE increases in the box muller transform.

Elapsed time is 0.001584 seconds.

ans =0.0090

First method n = 6

Elapsed time is 0.001023 seconds.

MSE = 0.0313

Box Muller n = 6

# MATLAB Code

The first function rvsum takes inputs of N- number of points in a vector, n- number of vectors to be generated, min and max- the min and max of distribution. The inputs are used to generate a uniform distribution with the input parameters with the function rand. The generated random variables are then summed and set as the output of the function. The second function rmsFIT takes the input of a vector (in this case the summed random variables) and the the number of vectors and outputs an MSE fit to a normal distribution. This is accomplished by calculating the mean and variance of the input vector, and creating a distribution with the fitdist function which takes the mean, standard deviation, and input vector. Now, a pdf is created of the input vector. The pdf will have less bins then the number of points created by the fitdist function, so the normal distribution will be downsampled with a matlab function downsample. Then the MSE is calculated between the two vectors of equal size and a plot is made for n=1,10,100 specifically for this CA.

% Tyler Olivieri  
% function to sum uniform distribution random variables  
% inputs  
% N - total number of points/samples  
% n - number of vectors to be generated//number of RV  
% min - min of uniform distibution  
% max - max of uniform distribution  
  
function s = rvsum(n,N,min,max)  
%test case  
% n = 10;  
% N = 10000;  
% min = -1;  
% max = 1;  
pointsinVector = round(N/n);  
x = min + (max-min)\*rand(N,1,n);  
if(n > 1)  
 s = sum(x,3);  
else  
 s = x;  
end

function [ R2 ] = rmsFIT( X, n)  
%Returns mean squared error between a data set and a Gaussian Distribution  
%of the data set.  
%Also plots the gaussian overlayed with a pdf for n=1, n=10, n=100,  
%where n is the number of uniformly distributed RVs being summed together.  
%if not experimenting with Central Limit Theorem, make n = 1.  
%X is the data set  
%NBIN represents how many bins to place in the histogram  
  
%find mean and variance  
mean\_x = mean(X);  
var\_x = var(X);  
  
X = sort(X);  
  
%define the gaussian  
pd = fitdist(X, 'Normal');  
gaus = pdf(pd, X);  
  
figure(1)  
edges = min(X):.1:max(X);  
% if(n>30)  
% edges = min(X):.5:max(X);  
% end  
h = histogram(X, edges, 'Normalization', 'pdf');  
title('all ns')  
%correlate the histogram and the gaussian distribution  
  
if((length(gaus)/length(h.Values))>1)  
 P = floor(length(gaus)/length(h.Values));  
 gaus\_avg = downsample(gaus, P);  
 for i = 1:length(h.Values)  
 dif\_square(i) = (h.Values(i)-gaus\_avg(i))^2;  
 end  
%compute the mean squared value  
 R2 = sum(dif\_square)/length(h.Values);  
  
elseif((length(gaus)/length(h.Values))<1)  
 P = floor(length(h.Values)/length(gaus));  
 gaus\_avg = downsample(h.Values, P);  
 for i = 1:length(gaus)  
 dif\_square(i) = (gaus(i)-gaus\_avg(i))^2;  
 end  
%compute the mean squared value  
 R2 = sum(dif\_square)/length(gaus);  
else  
 P = 1;  
end  
  
if(n == 1)  
 figure(2)  
 histogram(X, edges, 'Normalization', 'pdf');  
 hold on  
 plot(X',gaus)  
 title('n = 1')  
 hold off  
  
 elseif(n == 10)  
  
 figure(3)  
 histogram(X, edges, 'Normalization', 'pdf');  
 hold on  
 plot(X',gaus)  
 title('n = 10')  
 hold off  
  
 elseif(n == 100)  
 figure(4)  
 histogram(X, edges, 'Normalization', 'pdf');  
 hold on  
 plot(X',gaus)  
 title('n = 100')  
 hold off  
  
end  
  
  
end

In our main program we set the min max and number of points in each vector we want in our random distribution and loop the previous function while changing n. The sum of the random variables is calculated, the mean, variance and mse compared to a normal distribution. The tic and toc are used for timers. And plots are made at the end.

%Tyler Olivieri  
clc;clear;  
%  
min1 = -1;  
max1 = 1;  
N = 10000;  
%  
  
for n = 1:100  
 tic  
 x = rvsum(n,N,min1,max1);  
 if(n== 10)  
 toc  
 end  
 MSE(n) = rmsFIT(x,n);  
 E(n) = mean(x);  
 variance(n) = var(x);  
end  
%plot  
mse\_space = linspace(1,n,n);  
figure(50);  
plot(mse\_space,MSE)  
title('MSE')  
xlabel('n')  
ylabel('MSE')  
  
MSE(10)

n = 10;  
N = 10000;  
tic  
u1 = rand(N,1);  
u2 = rand(N,1);  
z = sqrt(-2\*log(u1)).\*cos(2\*pi\*u2);  
toc  
MSE = rmsFIT(z,n)

Finally, the box muller transform is used with n =10 by calculated to random vectors of uniform distribution and using the formula z = sqrt(-2\*log(u1)).\*cos(2\*pi\*u2); The MSE is calculated with the function rmsFIT.

# Conclusions

The sum of uniformly generated random variables will approach a Gaussian distribution. This can again be seen in Figure 1, Figure 2, Figure 4, and the MSE in Figure 5. There were problems found if N is set to be the total number of points in the distribution. As the variance grows when summing the random variables, there will be more range with the same number of points. This will create many empty bins and messed with the calculation of MSE. The variance and mean increases linearly. We expect the MSE to decrease as n increases because we are comparing the distribution to the normal. The box muller transform was also found to be much faster and I would expect it to stay faster as n increases. The time was calculated just to generate the sum of random variables. The MSE was higher but is still comparable. As n was decreased the computation time become comparable as well as the MSE.