Rachel King

ECE 3512: Signals – Continuous and Discrete

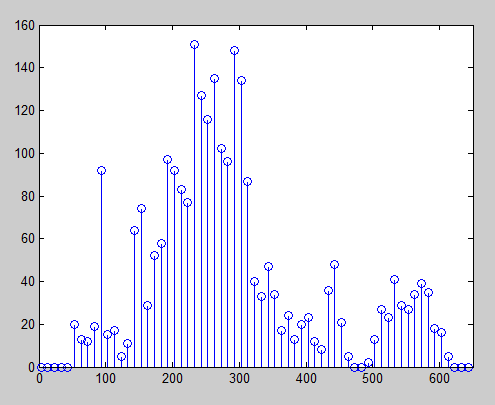
Department of Electrical and Computer Engineering, Temple University, Philadelphia, PA 1912

# Problem Statement

The purpose of this assignment is to explore concepts touched upon in chapter 4, which involved working with various types of distributions. Here, we will revisit the histograms we created in Computer Assignment 2, overlaying them with various types of fit (which MATLAB makes quite easy for us). Additionally, we will be evaluating numerically which fit is best for both the Google stock price data and the audio signal by evaluating the mean squared error. The main obstacle in this lab is determining which fit is best for each signal.

# Approach and Results

*Google Stock Prices*



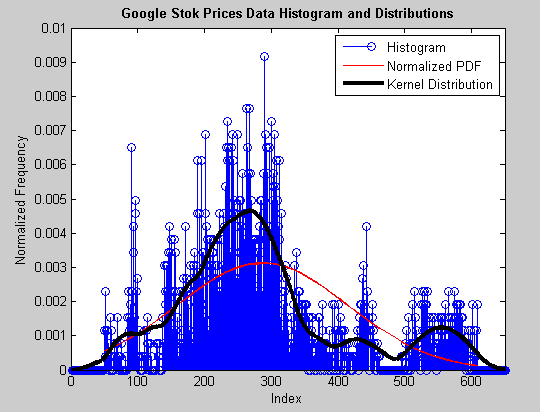
*Figure 1 – Histogram, Google Stock Prices*

First, we must make a histogram of the Google Stock prices, as that was not part of assignment 2. We simply used the histogram function and the following code:

binranges = [-32767:10:32767];

h = hist(sig,binranges);

Then, we just made a stem plot of ‘h’. We had to zoom in on our plot in order to analyze it.



*Figure 2 - Normalized PDF, Gaussian Fit, and Kernel Distribution*

Now, we normalize the histogram, shown in figure 2, so that it is a PDF. It now will sum to one! Here’s where the fun happens. We start by fitting the distribution with a Gaussian curve (because after all, everything is Gaussian eventually) as per instructions. In figure 2, the Gaussian curve is the red line…it really does not fit the distribution at all; it doesn’t respond to the rise and fall of peaks that the distribution of the google stock prices embodies. Here is how we accomplished creating this plot:

mn = mean(sig);

vr = nanvar(sig);

sqr = sqrt(vr);

norm = normpdf(sig, mn, sqr);

plot (sig, norm, 'r')

In this script ‘sig’ is the array in which we store the values of the stock prices. We find the mean, variance, and square root of the signal, and then feed it into prewritten MATLAB function ‘normpdf’.

We had to find a fit that does indeed work, as the Gaussian model failed. To do this, we honestly just guessed and checked…it was not very scientific. We settled on the ‘Kernel’ fit, which seemed to mimic the rise and fall of the pdf nicely. Here is how we achieved that result:

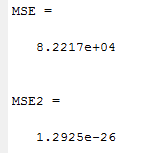
x\_values = -32767:1:32767;

pd = fitdist(sig,'Kernel','Kernel','epanechnikov')

y = pdf(pd,x\_values);

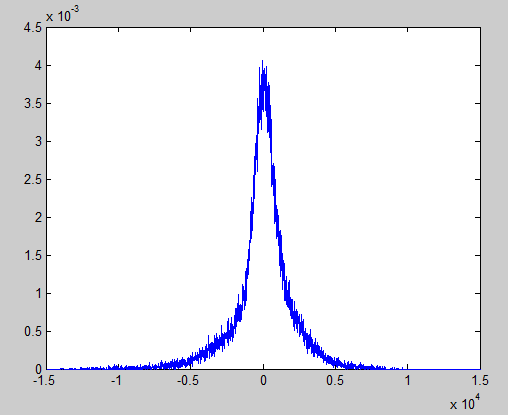
This script used the MATLAB function ‘fitdist’ to choose the appropriate distribution.

As far as the mean squared error goes, the following answers were our result:



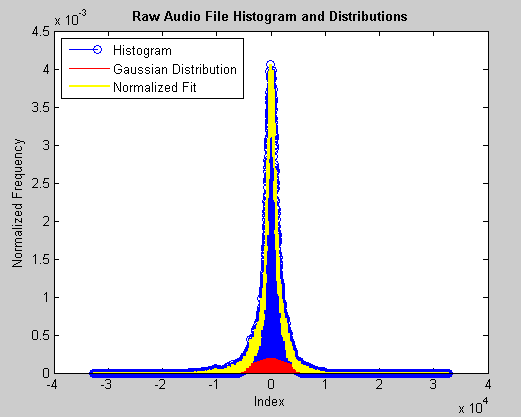
MSE is the mean-squared error from the Gaussian curve, and MSE2 was the mean-squared error from the kernel fit.

*Audio File*



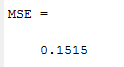
*Figure 3 - Histogram, Audio Signal*

The normalized histogram, shown in figure 3, is the same as in computer assignment 2. After studying what a Gaussian fit is, it becomes much clearer that the Gaussian model may in fact be perfect for the audio file.



*Figure 4 – PDF, Gaussian Distribution, Normalized Fit*

The yellow line shows the normalized Gaussian fit. As you can see, it’s almost perfectly matched with the shape of the histogram curve. The meat squared error is as follows:



We found this with the following code:  
MSE = mean((y - mn).^2)

Where mn is the mean of the signal and y is the normalized Gaussian fit.

As you can see, the mean squared error is very small, and therefore seems to be an excellent fit for this signal.

# MATLAB Code

function CA4\_part2

clear all; clc;

[sig, txt, raw] = xlsread('google\_v00.xlsx', 1);

sig = sig(:,4);

% load the data:

% It is 16-bit sampled data, so we must load it as short integers.

%

[z,q] = size(sig);

mn = mean(sig);

vr = nanvar(sig);

sqr = sqrt(vr);

norm = normpdf(sig, mn, sqr);

xbins = (-32767:1:32767);

x\_values = -32767:1:32767;

pd = fitdist(sig,'Kernel','Kernel','epanechnikov')

k = mean(pd)

p = mean(norm)

y = pdf(pd,x\_values);

figure(1)

h = hist(sig, xbins)

h = h/z

stem(xbins, h, 'b')

xlim([0, 650]);

hold on

plot (sig, norm, 'r')

hold on

plot(x\_values, y, 'k', 'linewidth', 3)

legend('Histogram', 'Normalized PDF', 'Kernel Distribution', 'location', 'northeast');

title('Google Stok Prices Data Histogram and Distributions', 'fontweight', 'bold');

ylabel('Normalized Frequency');

xlabel('Index');

MSE = mean((p-mn).^2)

MSE2 = mean((k -mn).^2)

end

# Conclusions

Ultimately, this assignment definitely taught us quite a bit about picking the appropriate fit for given data. The results were pretty straight forward; different signals have different distributions and can therefore be approximated by different models. The audio signal was modelled quite nicely by the Gaussian model, which makes since because its amplitudes were a little bit more random. Additionally, many of the amplitudes were 0, so it makes sense that it would be centered about 0. The google stock prices, however, were a little bit trickier; they were less random, and therefore needed an approximation that was more specific. We concluded that the ‘Kernel’ fit worked nicely. Fitting data is useful to us because it allows us to use a much smoother curve than our data if we wanted to apply our signal to something. Basically, it eliminates noise through estimation.