Dana Joaquin

ECE 3512: Signals – Continuous and Discrete

Department of Electrical and Computer Engineering, Temple University, Philadelphia, PA 19121

# Problem Statement

The purpose of this assignment is for students to plot the PDF of a particular data using different types of distributions. There are different ways to represent and plot data, again, it just depends on the preference and the type of information one wants to see or display. This assignment will allow students to plot a normal distribution for both the Google stock and audio file, in addition to figuring out a better distribution plot for the Google stock by using the concept mean-squared error.

# Approach and Results

The assignment asked for the histogram and normal distribution plot for both the Google stock and audio signal data. The overall approach to accomplishing this task is making an MATLAB function made entirely of function calls to plot and display the desired results for both sets of data. The main MATLAB functions used to do this were the histogram and fitdist functions because they were used in the program to get the needed data to plot the distributions.

The MATLAB histogram function was used to create the histogram plots for both the Google stock and audio file, the blue plot in figure 1a and 1b. The edges for the histogram were changed to make the histogram have the same number of bins as there were samples in the signal. Although this approach would create some empty bins/holes in the histogram, it was needed in order to compute the distributions. To plot the normal/Gaussian distribution for both sets of data, two functions were made to do this task: compute\_fit and normalfit. Compute\_fit is a function that takes in three arguments: the signal’s array of data, a vector that’s the same length as the signal, and the type of distribution desired (this is a string). What compute fit actually does is that it takes in those arguments and compares the string argument to check if the user is asking for a normal distribution using an if-else statement. Once there is a match, compute\_fit calls the function normalfit to actually compute the normal distribution of the signal, sending over the signal and vector arrays. Normalfit calculates the data’s normal distribution using the equation:

Equation 1: Normal/Gaussian Distribution

µ is the mean of the data and σ is the standard deviation of the data. These parameters were found by using the fitdist MATLAB function. These values were then implemented into Equation 1 to calculate the normal distribution for the signal for each value of x (which is the vector that is the same length as the signal.) The values were saved in the variable “normalfitval” which returned the array of values to the main function.

This distribution looked fine for the audio file, shown in figure 1b. Even though the histogram and normal distribution differ in amplitude, the general shapes of the two distributions appear to match. The amplitude difference is most likely due to the noise in the audio file. For the Google stock file, the normal distribution does not seem to fit the data as well because it does not account for the second small bump that appears in the histogram around the [500 600] x values. The other distributions that were used to obtain a better fit are: Poisson and gamma. A separate function, just like the normalfit one, was made for the Poisson and gamma distribution. The functions follow the same general algorithm as the, normalfit function. For the gamma distribution, the equation that was used is:

Equation 2: Gamma distribution

Parameters a and b were found using the fitdist function, similar to how fitdist was used to find the parameters for the normal distribution. For the Poisson distribution, the values were calculated using the MATLAB function poisspdf, which takes in the signal data and the mean of that signal.

In order to compare these distributions to see which one fits better with the signal’s histogram plot, a mean-squared error function was computed which was done by writing the function compute\_mse. Although there is already a MATLAB function to do this, it was better to make a custom one since I could see exactly how the function works. The mean-squared error equation as implemented into this compute\_mse function:

n = the number of samples in the signal

Y-hat = predicted values of signal (for this case, the distribution values)

Y = true values (for this case, the histogram values)

This function outputted the MSE values for each distribution used for both signals. For the Google stock, three distributions were used and for the audio, only the normal distribution was used. Table I lists these MSE values.

|  |  |
| --- | --- |
| **Google Stock Prices (Close) MSE** | |
| **Distribution** | **MSE Value** |
| **Normal** | 4.6584e-07 |
| **Gamma** | 4.0216e-07 |
| **Poisson** | 1.3698e-06 |
| **Audio File MSE** | |
| **Normal** | 5.7444e-10 |

Comparing the MSE results and the plotted results of the Google stock data, it makes sense that the gamma distribution has a better fit than the other two. The gamma distribution was able to fit the probabilities in [100 250] range of the histogram better since it has a sharper increase and that is where histogram had higher probability values. The normal distribution did not represent this part of the data that well and the bell curve appeared more around the [250 350] range. For the Poisson distribution, it had the highest MSE value and it shows in figure 1a. It did not fit the histogram data as well as the other two, but it was still able to represent the high peaks that appeared around the 250 values. Even though it still provides an answer, the Poisson fit is not the best one to choose from.

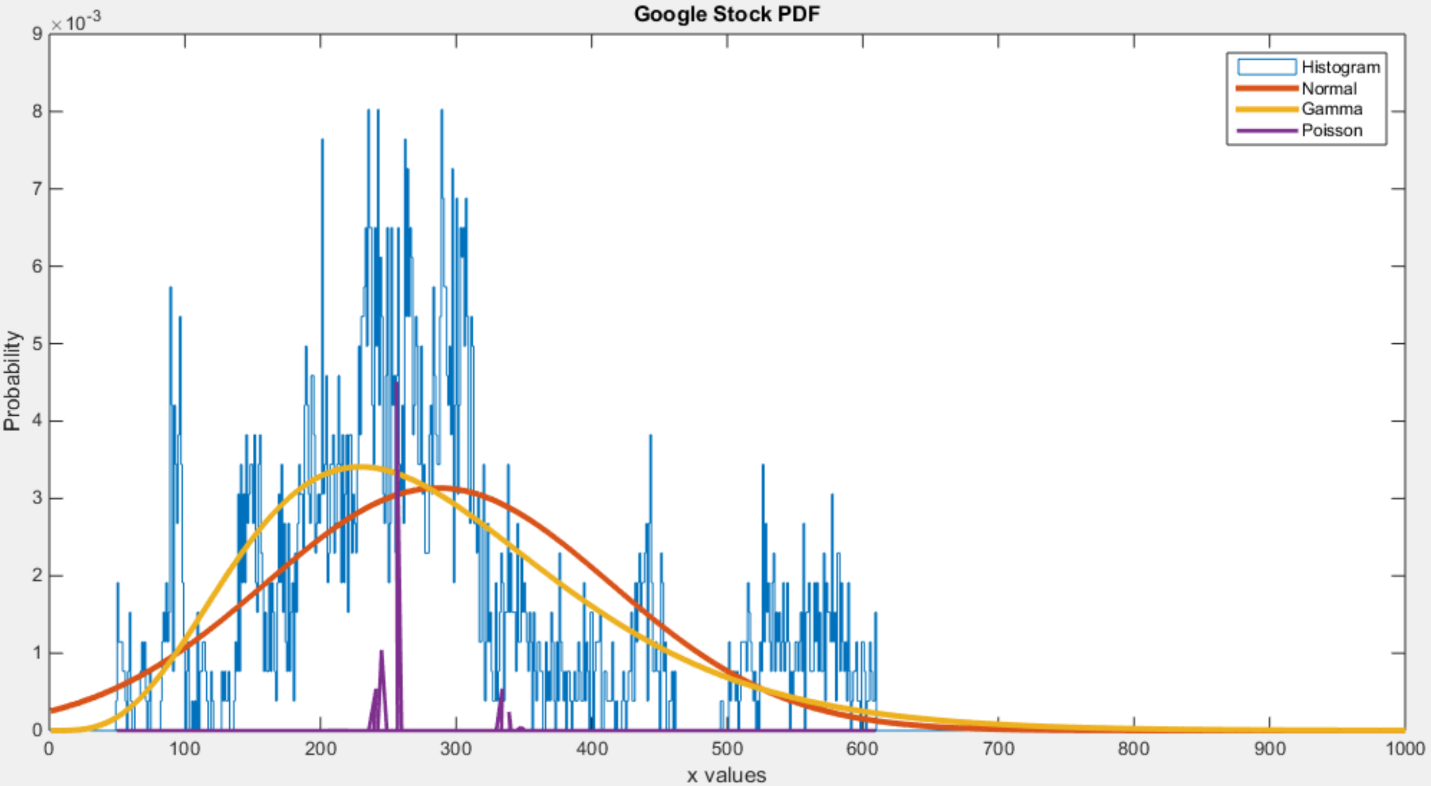


Figure 1a. Google Stock Close PDF Histogram with Normal, Poisson and Gamma distributions

For the audio file, the normal distribution was a good fit for the audio file, having an MSE value of 5.7444e-10. This make sense since the amplitude values in the audio file consisted of both negative and positive values, therefore the audio file’s mean would be closer towards zero, which is shown in the histogram, having a bell-like curve shape to it already. The normal distribution was able to pick this up really well and showed that the mean and most of the x values closer to zero had higher probabilities than those not close to zero.

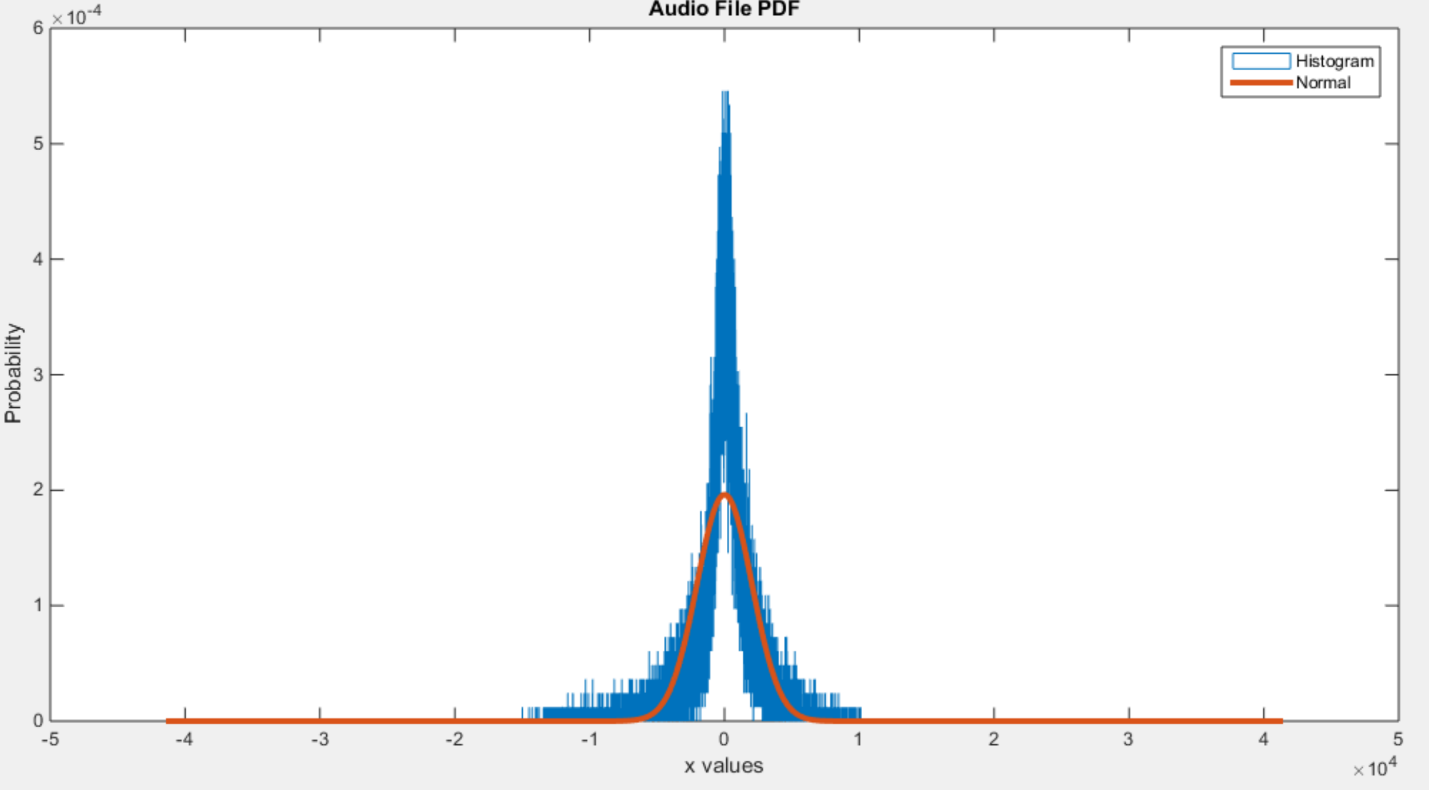


Figure 1b. Audio file PDF histogram with Normal distribution

# MATLAB Code

MATLAB code for Google stock close prices

|  |
| --- |
| % computer assignment 4 v02  % Dana Joaquin    function fitdatdist = ca4\_v02  clear; clc  close all    % distribution data for google stock file  % opening and reading google stock file  filename = 'google\_v00.xlsx';  cl = 'E:E'; %column E identifier  signaldata = xlsread(filename,1,cl);  datalength = length(signaldata);  time = 1:length(signaldata); %length of signal (array)    % % testing compute\_mse function  % test\_mse = compute\_mse(signaldata, signaldata, datalength)    % compute normal distribution of data  fittype = 'Normal';  a = compute\_fit(signaldata, time, fittype);    % compute gamma distribution of data  fittype = 'Gamma';  b = compute\_fit(signaldata, time, fittype);    % compute poisson distribution of data  fittype = 'Poisson';  c = compute\_fit(signaldata, time, fittype);    figure(1)  % compute pdf histogram of data  hist\_data = histogram(signaldata, ...  'DisplayStyle', 'stairs', 'Normalization', 'pdf');  so\_edgy = 0:1:length(signaldata);  hist\_data.BinEdges = so\_edgy;  hold on  plot(time, a) % plotting normal dist.  hold on  plot(time, b) % plotting gamma dist.  hold on  plot(signaldata, c) % plotting poisson dist.  legend('Histogram', 'Normal', 'Gamma', 'Poisson')  title('Google Stock PDF')  xlabel('x values')  xlim([0 1000])  ylabel('Probability')    % computing stock mse - normal  histvalues = hist\_data.Values;  stock\_mse\_normal = compute\_mse(histvalues, a, datalength);  stock\_mse\_gamma = compute\_mse(histvalues, b, datalength);  stock\_mse\_poisson = compute\_mse(histvalues, c, datalength);    % display mse calculations    display(stock\_mse\_normal); % normal  display(stock\_mse\_gamma); % gamma  display(stock\_mse\_poisson); % poisson    % ---------------------------------------------------------    % distribution data for audio file  % opening and reading audio file  filename = 'rec\_01\_speech.raw';  fileID = fopen(filename);  signaldata = fread(fileID, inf, 'int16');  fclose(fileID);  datalength = length(signaldata);  time = -length(signaldata)/2:1:(length(signaldata)/2)-1;    % % compute normal distribution of data  fittype = 'Normal';  a = compute\_fit(signaldata, time, fittype);    figure(2)  % % compute pdf histogram of data  hist\_data = histogram(signaldata, ...  'DisplayStyle', 'stairs', 'Normalization', 'pdf');  oh\_so\_edgy = -length(signaldata)/2:1:length(signaldata)/2;  hist\_data.BinEdges = oh\_so\_edgy;  hold on  plot(time, a) % plotting normal dist.  legend('Histogram', 'Normal')  title('Audio File PDF')  xlabel('x values')  ylabel('Probability')    % computing audio mse - normal  histvalues = hist\_data.Values;  audio\_mse\_normal = compute\_mse(histvalues, a, datalength);    % display audio mse data  display(audio\_mse\_normal);    end    % function name: compute\_fit  % coded by Dana Joaquin  % input argument(s):  % (1) siggie: signal data array (array)  % (2) siglength: length of signal (array)  % (3) fittype: distribution fit type (string)  % output argument(s):  % (1) befit: returns an array of values of the distribution plot  % objective:  % This function is to compute the distribution of an array  % of data based on the distribution type the user inputs (fittype).  % This compares the string in fittype in an if-else statement  % to call the corresponding function to compute the distribution.  % note(s):  % 2/13/2015 - Dana Joaquin  % For computer assignment 4 purposes, this function will only  % contain the distributions needed to accomplish this assignment.  % I may add more of the other distributions to make this function  % more complete.  % Currently has Normal, Gamma and Poisson  function befit = compute\_fit(siggie, siglength, fittype)    if strcmp(fittype, 'Normal') == 1 % check if fittype is normal dist  befit = normalfit(siggie, siglength);  elseif strcmp(fittype, 'Gamma') == 1  befit = gammafit(siggie, siglength); % check if fittype is gamma  elseif strcmp(fittype, 'Poisson') == 1  befit = poissonfit(siggie, siglength);  end    end    % function name: normalfit  % coded by Dana Joaquin  % input argument(s):  % (1) sig: signal data (array)  % (2) length\_vect: length of signal data (array)  % output argument(s):  % (1) normfitval: array of values of the normal distribution  % objective:  % This function uses the fitdist MATLAB function to compute the  % mean and standard deviaton (std dev) of the data.  % (Note: I am aware that MATLAB has functions to compute those  % 2 values, but I used fitdist because of preference.)  % Uses the mean and std dev values to compute the normal dist.  % of the signal. (saves it all in an array the length of signal)  % The data is calculated using the normal distrbution equation:  %  % u = mean  % sigma = std. dev.  % -(x-u)^2  % ---------  % 1 2(sigma)^2  % f(x|u, sigma) = --------------- \* e  % sigma\*sqrt(2pi)  %  function normfitval = normalfit(sig, length\_vect)  % get parameter values of Normal dist.  fit = fitdist(sig, 'Normal');    meanval = fit.mu; % mean  stddev = fit.sigma; % standard deviation  multf = 1/(stddev\*sqrt(2\*pi));  expf = (-(length\_vect-meanval).^2)./(2\*stddev^2);    normfitval = multf.\*exp(expf);  end    % function name: gammafit  % coded by Dana Joaquin  % input argument(s):  % (1) sig: signal data (array)  % (2) length\_vect: length of signal data (array)  % output argument(s):  % (1) gammaval: array of values of the gamma distribution  % objective:  % This function uses the fitdist MATLAB function to compute the  % a (aval) and b (bval) for the gamma distribution computation.  % Uses the a and b values to compute the gamma dist.  % of the signal. (saves it all in an array the length of signal)  % The data is calculated using the gamma distribution equation:  %  % 1 (a-1) (-x/b)  % f(x|a, b) = --------------- \* x \* e x > 0  % (b^a)gamma(a)  %  % Note: this function will not work on the audio file since  % the audio file has values in the negative region  function gammaval = gammafit(sig, length\_vect)  % get parameter values of Gamma dist.  fit = fitdist(sig, 'Gamma');    aval = fit.a;  bval = fit.b;  multf = 1/((bval^aval)\*gamma(aval));  expf = -(length\_vect)./bval;    %calculate gamma dist. values  gammaval = multf.\*(length\_vect.^(aval-1)).\*exp(expf);  end    % function name: poissonfit  % coded by Dana Joaquin  % input argument(s):  % (1) sig: signal data (array)  % (2) length\_vect: length of signal data (array)  % output argument(s):  % (1) poissonval: array of values of the Poisson distribution  % objective:  % This function uses the fitdist MATLAB function to compute the  % mean of the signal data.  % Uses the mean to compute the Poisson dist. of the signal.  % The Poisson dist. is computed using the poisspdf MATLAB  % function to get the values.  %  function poissonval = poissonfit(sig, length\_vect)  %get parameter value of poisson dist.  fit = fitdist(sig, 'Poisson');    meanval = fit.lambda;  poissonval = poisspdf(sig, meanval);    end    % function name: compute\_mse  % coded by Dana Joaquin  % input argument(s):  % (1) sig\_a: signal data (array)  % (2) sig\_b: signal data (array)  % (3) length\_vect: length of signal data (array)  % output argument(s):  % (1) mse: scalar value of mean-squared error  % objective:  % This function computes the mean-squared value of two  % data arrays brought in by sig\_a and sig\_b.  % Calculates the mean-squared value by using the formula:  %  % n : length of vector (both sig\_a and sig\_b have to be same size)  % Y-hat : predicted values (sig\_a is true val. arrays)  % Y : true values (sig\_b is predicted val. arrays)  %  % \_\_n\_\_  % MSE = 1 \ \_ 2  % --- / (Y - Y)  % n ----- i i  % i = 1  %  function mse\_val = compute\_mse(sig\_a, sig\_b, sig\_length)    n = sig\_length;  for i=1:n  data\_array(i) = (sig\_b(i) - sig\_a(i))^2;  end    mse = (1/n)\*(sum(data\_array));  mse\_val = mse;    end |

# Conclusions

There is no wrong answer in regards to what distribution would display the probabilities of a particular set of data, but there are always better answers that would do it. This assignment emphasizes that idea that data can be presented any way, just that there are better ways to do it. Both the Google stock and the audio file’s normal distributions were computed and although it was a good fit for the audio file, it was not for the Google data. After implementing two different types of distributions, Poisson and gamma, it shows that all three distributions provided the same type of data, but just showed it differently. The least appropriate one was the Poisson while the best one was the gamma distribution. There are different types of distributions and each type is suited better for particular types of data and for the Google case, Poisson did not do. To further verify which distribution was better, the mean-squared error was calculated to back-up that conclusion. Based on the data, the gamma distribution fit better than the normal distribution for the Google stock prices. It is reassuring to have quantitative results to help decide what fit is more appropriate for certain sets of data rather than just going by aesthetics since the gamma and normal distributions look almost similar.