Dana Joaquin

ECE 3512: Signals – Continuous and Discrete

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

# Problem Statement

This computer assignment is to introduce new prediction functions: linear regression, probability density function (PDF) and cumulative distribution function (CDF). Using the Google stock prices provided from the last computer assignment, the linear regression line of the close data is to be computed and compared to the actual data and its mean data. For the audio file that was used in the last assignment, two histograms are to be plotted: one for the PDF and one for the CDF of the audio file. This is all to be done on MATLAB.

# Approach and Results

**Question 1**

To plot the original plot of the stock prices and the mean, the code for the first computer assignment was used. For the linear regression line, a separate function was written to compute it. The resource used to calculate for it was this: <http://onlinestatbook.com/2/regression/intro.html>. Based on that website, the values needed to compute the stock price’s linear regression are the stock prices: mean of the x values, mean of the y values, standard deviation of the x values, the standard deviation of the y values and the stock prices’ correlation value, specifically Pearson’s correlation, ‘r’. These values are used to compute the line’s slope and y-intercept. Referring to the website, they can be calculated as so:

These equations were implemented into the function to create the linear regression line for the Google stock prices for the “close” data. The results are shown in Figure 1, comparing all 3 plots together. It makes sense that the linear regression line is a positive slope since the Google stock prices generally kept increasing as the days progressed.

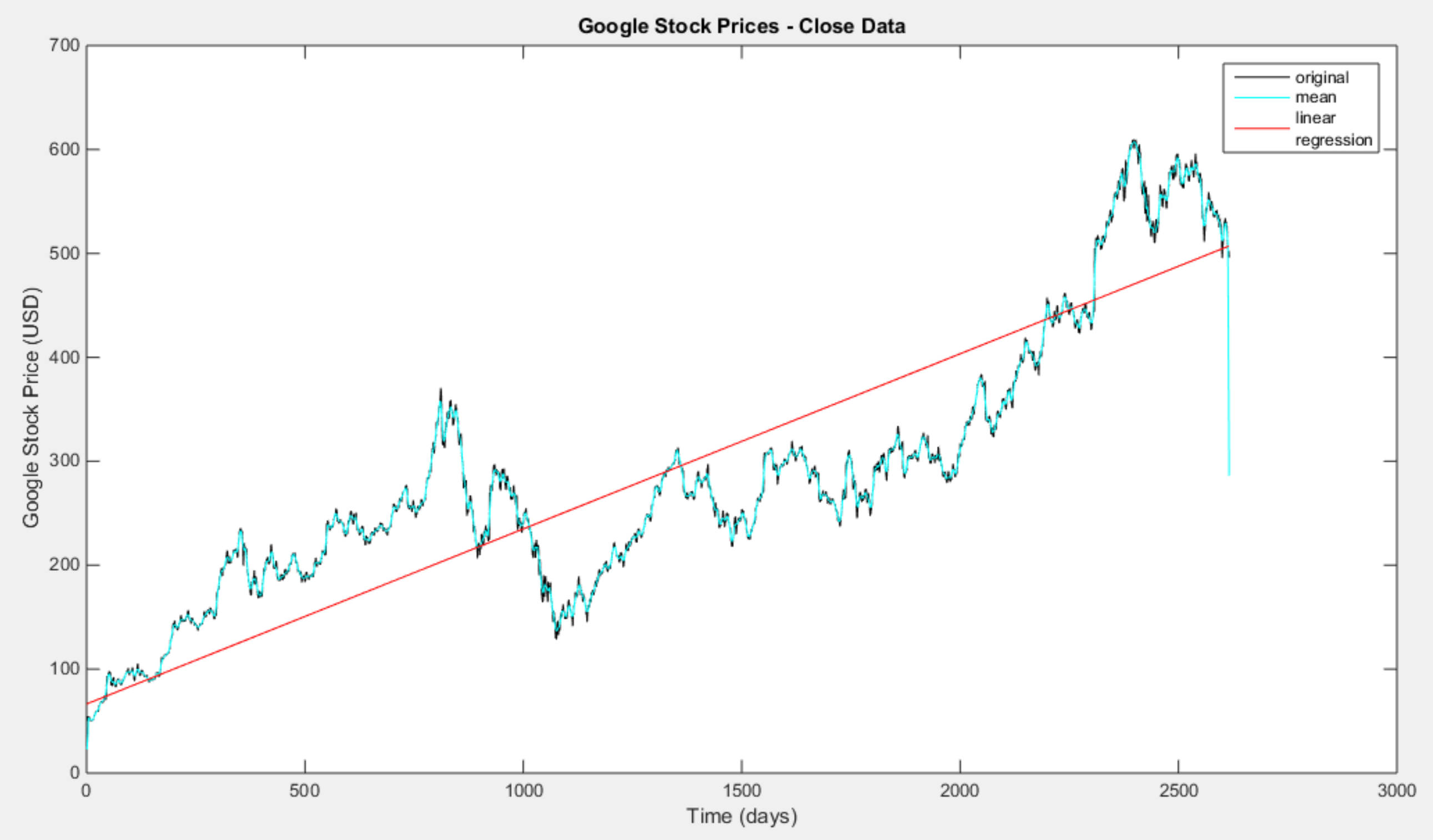


Figure 1. The original close stock price data, its means and its linear regression plotted together.

**Question 2**

The main function used to solve second part of this assignment was MATLAB’s histogram function, which is written to plot the PDF and CDF of its inputted argument. The bin size was made 10 and its range was from -32767 to +32767, stated by the problem. The samples in each bin were normalized, which is an option that can be done using the histogram function.

Overall, the resulted plots make sense since the probability of all the sample points in the sample space must equal to one. The PDF displays the probabilities of all possible sample points in the sample space, shown in Figure 2a. All probabilities have to be less than 1 and the sum of all probabilities must equal 1. 1Since the CDF is the integral of the PDF, the integral should equal to 1 based on the fact that the sum of the all the probabilities should be 1, as seen in Figure 2b.

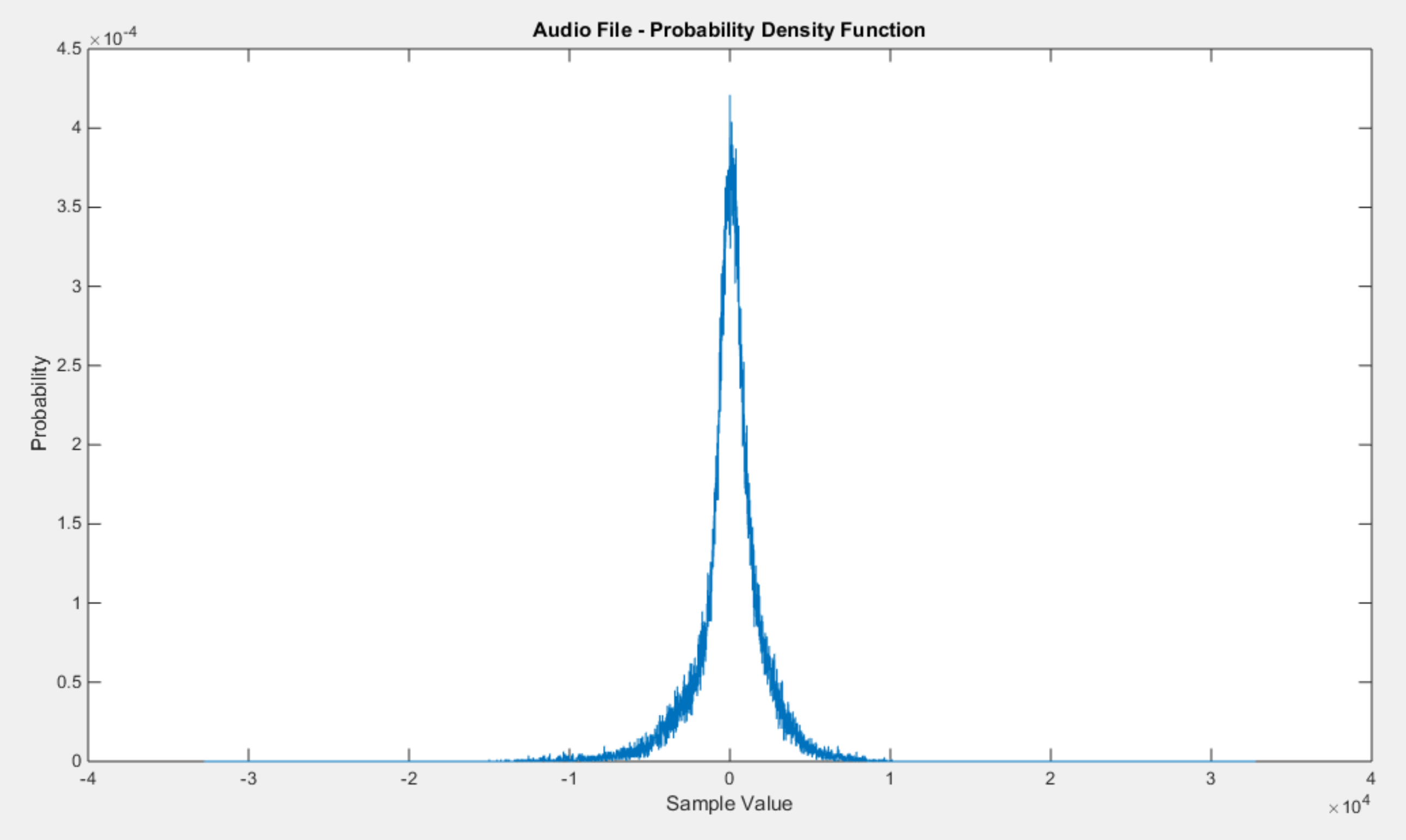


Figure 2a. PDF of the audio file. The smaller sample values, have a higher probability than the larger sample values. (Where the sample values are |x|).

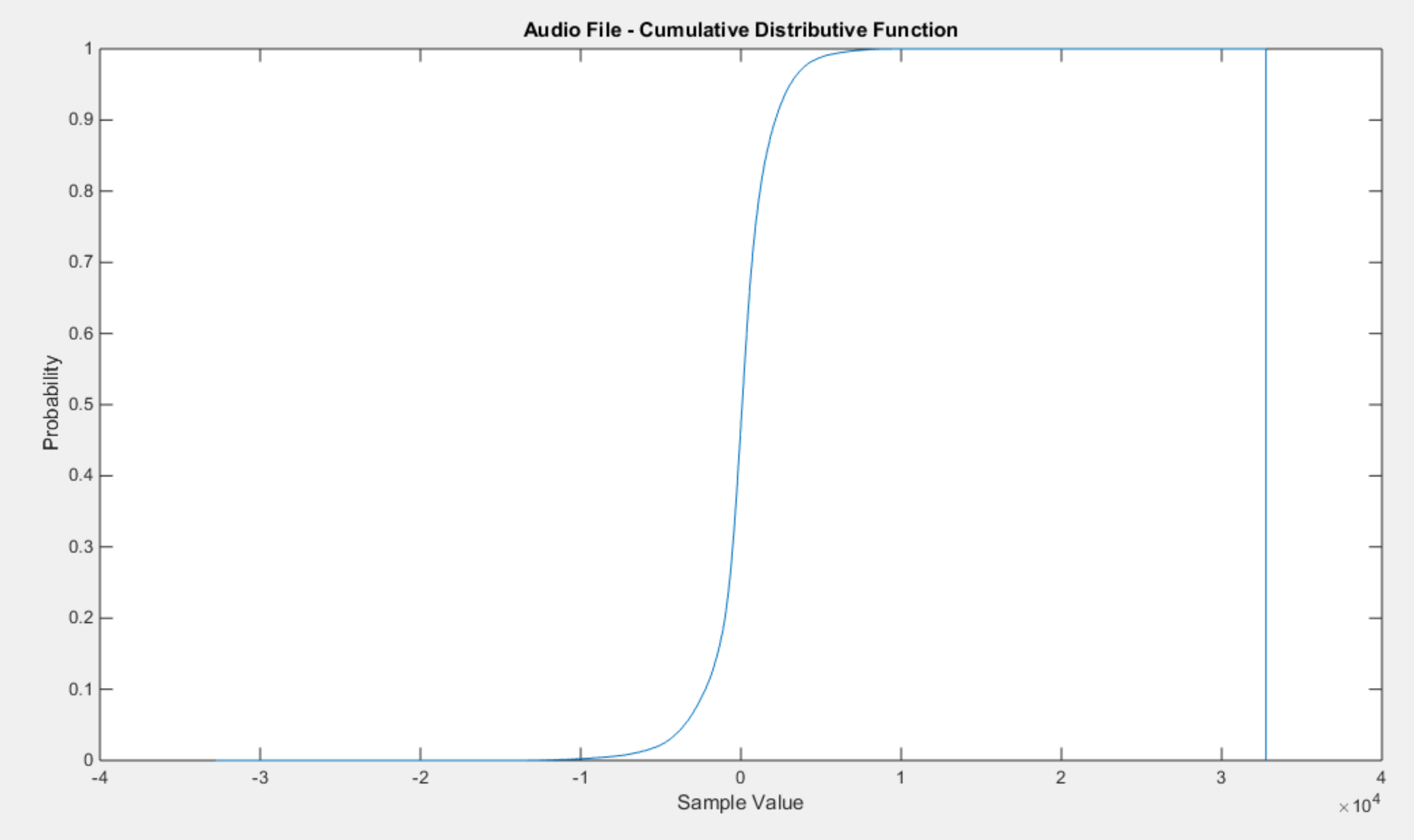


Figure 2b. The CDF of the audio file. The CDF is the integral of the PDF, therefore it makes sense the integral equals to 1.

# MATLAB Code

**Question 1 MATLAB Code**

|  |
| --- |
| % This function specifically finds the mean of the signal  % for each frame and window value.    %  % SIGNAL: =================================================================  % FRAMES: |---.---|---.---|---.---|---.---|---.---|---.---|---.---|---.---|  % WINDOWS: <---------------> <--------------->  % <---------------> <--------------->  % <---------------> <--------------->  %    % declare a function  %  function mean\_value=CA2\_p1\_stock\_v00    % close open sessions  %  close all;    % define two key parameters:  % M: frame duration in samples - how often we compute an output  % N: window duration in samples - how much data we use in each computation  %  M = 1;  N = 7;    % open the file:  % We assume the data is in a file "rec\_01\_speech.raw". This should be  % parameterized, but we hardcode it here to keep things simple.  %  filename = 'google\_v00.xlsx';  cl = 'E:E'; %column E identifier  cloze = xlsread(filename,1,cl);    % create a matrix to store the output  %  mean\_v = zeros(length(M), length(N), length(cloze));    h1 = figure('name', 'mean plot', 'numbertitle', 'off');  figure(h1);  plot(cloze, 'k');  hold on    % loop over the set of frame/window combinations.  %  for m = 1:length(M)    % set up a plotting window and label it  %  for n = 1:length(N)    % call a function to compute mean of signal  %  mean\_v(m,n,:) = compute\_mean(cloze, M(m), N(n));  % plot the mean  plot(squeeze(mean\_v(m,n,:)), 'c');  % labels  title('Google Stock Prices - Close Data')  xlabel('Time (days)')  ylabel('Google Stock Price (USD)')  end  end  mean\_out = squeeze(mean\_v);    bfl = compute\_linereg(cloze);    end    % function name: linear\_reg  % coded by Dana Joaquin  % function: takes in a signal/array of values and finds the  % linear regression line by implementing the slope  % and y-intercept equations desribed below  % reference: http://onlinestatbook.com/2/regression/intro.html  %  % notation:  % r : Pearson's correlation  % mx : mean of x values of signal/array  % my : mean of y values of signal/array  % sx : standard deviation of x values of signal/array  % sy : standard deviation of y values of signal/array  %  % slope (m) of linear regression line:  % m = r \* (sy/sx)  %  % intercept (b) of linear regression line  % b = my - m\*mx  function linear\_reg = compute\_linereg(siggie)    % getting x-axis vector  a = length(siggie);  days = 0:a-1;  % finding the mean values for the x and y values of whole signal  meanx = mean(days);  meany = mean(siggie);  % finding standard deviation of x and y values  xstand = sqrt(var(days));  ystand = sqrt(var(siggie));  % finding Pearson's correlation 'r' of signal  r = corr(siggie);  % calculating the slope of line  mx = r.\*(ystand./xstand);  % calculating y-intercept of line  intcept = meany - meanx.\*mx;  % creating a vector of values of the line  linear\_reg = zeros(size(days));  % need to make a for loop to make array of values to plot the line  for i=1:length(days)  slope = mx.\*(i-1); % must start with slope of 0 for y-intercept  if i == 1  linear\_reg(i) = intcept; %1st index contains y-intercept  else  linear\_reg(i) = intcept + slope;  end  end    plot(days, linear\_reg, 'r')  hold on  end    % function: compute\_mean  %  % arguments:  % sig\_a: the input signal (input)  % fdur\_a: the frame duration in samples (input)  % wdur\_a: the window duration in samples (input)  %  % return:  % mean: a vector of mean values (output)  %  % Note that this function returns the mean counter as a sampled data  % signal that is the same length as the input signal. This is wasteful  % of memory, but makes it easy to produce a time-aligned plot.  %  % This algorithm computes the sum of squares for wdur\_a samples.  %  function mean\_full = compute\_mean(sig\_a, fdur\_a, wdur\_a)    % declare local variables  %  sig\_wbuf = zeros(1, wdur\_a);  num\_samples = length(sig\_a);  num\_frames = 1+round(num\_samples / fdur\_a);  mean\_full = zeros(length(sig\_a),1);    % loop over the entire signal  %  for i = 1:num\_frames    % generate the pointers for how we will move through the data signal.  % the center tells us where our frame is located and the ptr and right  % indicate the reach of our window around that frame  %  n\_center = round((i - 1) \* fdur\_a + (fdur\_a / 2)); % 1st iteration, n\_center = 0.5  n\_left = round(n\_center - (wdur\_a / 2));  n\_right = round(n\_left + wdur\_a - 1);    % when the pointers exceed the index of the input data we won't be  % adding enough samples to fill the full window. to solve this zero  % stuffing will occur to ensure the buffer is always full of the same  % number of samples  %  if( (n\_left < 0) || (n\_right > num\_samples) )  sig\_wbuf = zeros(1, wdur\_a);  end    % transfer the data to this buffer:  % note that this is really expensive computationally  %  for j = 1:wdur\_a  index = n\_left + (j - 1);  if ((index > 0) && (index <= num\_samples))  sig\_wbuf(j) = sig\_a(index); % source of error  end  end    % compute mean value of signal  mean\_val = mean(sig\_wbuf);    % assign the mean value to the output signal:  % note that we write fdur\_a values  %  for j = 1:fdur\_a  index = round(n\_center + (j - 1) - (fdur\_a/2));  if ((index > 0) && (index <= num\_samples))  mean\_full(index) = mean\_val;  end  end  end  end |

The same MATLAB code in the first computer assignment was used to plot the original Google close stock prices and its mean. A new function was written (linear\_reg) to compute and plot the linear regression line of the stock prices. This was done by implementing the equations from the reference stated in the previous section.

How the linear\_reg function does this is that it takes in the signal and computes the mean of its x values and its y values, using the mean MATLAB function. The x matrix was created by taking the length of the signal and recreating a new matrix that is the same size as the signal. In this function, the variable “days” holds this matrix. Next the standard deviation of both the x and y matrix were computed, by using the var MATLAB function and taking the square root of that value since the square root of the variance gives the standard deviation. The last value needed to calculate the slope and y-intercept of the linear regression line, the corr MATLAB function was used to find the correlation of the stock price file. All these values were used to calculate the slope and y-intercept of the line. Next, the vector containing the y-values of the linear regression line had to be made. This was done using a for loop, looping from i = 1 to the length of the stock price file. In the first iteration, when x = 0, it is the y-intercept, therefore the first index contained the y-intercept value. The rest of the iterations, all of the other memory blocks contained the slope at that value x value plus the y-intercept. Normally in programming, the first index is 0, but in MATLAB, it is 1. To make sure to calculate the correct slope, each time the slope was calculated for the next i value, 1 was subtracted from it. This was how I was able to get only the y-intercept into the first index.

**Question 2 MATLAB Code**

The MATLAB function “histogram” was the main function used to answer question 2 of this assignment. This function was written to plot the CDF and PDF of the argument.

For a general run-through of the code, the speech file was first open and read into the variable “siggie” and then closed immediately after.

The problem stated that the histogram should be centered at -5 with a range from -32767 to +32767 with bin sizes of 10. The function horzcat was used to concatenate the left edge and right edge together, still keeping -5 the center value.

The histograms were normalized by adding the argument ‘Normalization’ to the histogram function. For better visualization, the display was set to ‘stairs’.

To differentiate which histogram plotted the CDF or PDF, their arguments were added to the histogram function.

|  |
| --- |
| %% Computer Assignment 2 Problem 2  % Dana Joaquin    clear; clc  close all    % opening and closing the audio file  filename = 'rec\_01\_speech.raw';  fileID = fopen(filename);  siggie = fread(fileID, inf, 'int16');  fclose(fileID);    % specify edges for histogram  edger = -5:10:32767;  edgel = -32767:10:-5;  so\_edgy = horzcat(edge, edger);    % PDF of audio file  figure(1)  histo\_pdf = histogram(siggie, 'Normalization', ...  'pdf', 'DisplayStyle', 'stairs');  histo\_pdf.BinEdges = so\_edgy;  title('Audio File - Probability Density Function')  xlabel('Sample Value')  ylabel('Probability')    % cdf of audio file  figure(2)  histo\_cdf = histogram(siggie, 'Normalization', ...  'cdf', 'DisplayStyle', 'stairs');  histo\_cdf.BinEdges = so\_edgy;  title('Audio File - Cumulative Distributive ...   Function')  ylabel('Probability')  xlabel('Sample Value') |

# Conclusions

Linear regression is a probability method to model a dependable variable with respect to an expected/independent variable. For this assignment, the expected/independent variable were the days in question 1. The time passed is independent, however, the stock prices were not and changed based on the day, which is the dependable variable. Since the Google stock prices only had one expected/independent variable, the plotted line is known as simple linear regression. Linear regression answers the question “Given x, what is the probability of y?” It helps predict potential trends, for example, in the stock market. For question 1, it is known that the Google stock prices increased as time progressed because the data was given beforehand. Since this is the case, the linear regression line should do the same, which is did. It showed that as time progressed, Google stocks kept increasing, giving a positive slope.

For question 2, it focused on the probability density and cumulative distributive functions. PDF plots the probability values for every sample point in the sample space. For the audio file, it shows that the amplitude of the audio file are most likely to be smaller in amplitude rather than higher in amplitude. The amplitude values are with respect to the volume level of the audio file. Therefore, the PDF is saying is that the audio clip is more likely to be low in volume rather high. This makes sense since the audio file was generally not that loud, it had a lot of breaks in between words, and it was silent more towards the end of the audio clip. The CDF is the integral of all the PDF values, and as the limits of integration for the CDF increased, so did the integrals value. This is because it sums up all the values of the PDF, which are the sample points’ probability values. Summing all these values up results with a 1, which make sense due to the fact that the total probability of a sample space is 1.