Dana Joaquin

ECE 3512: Signals – Continuous and Discrete

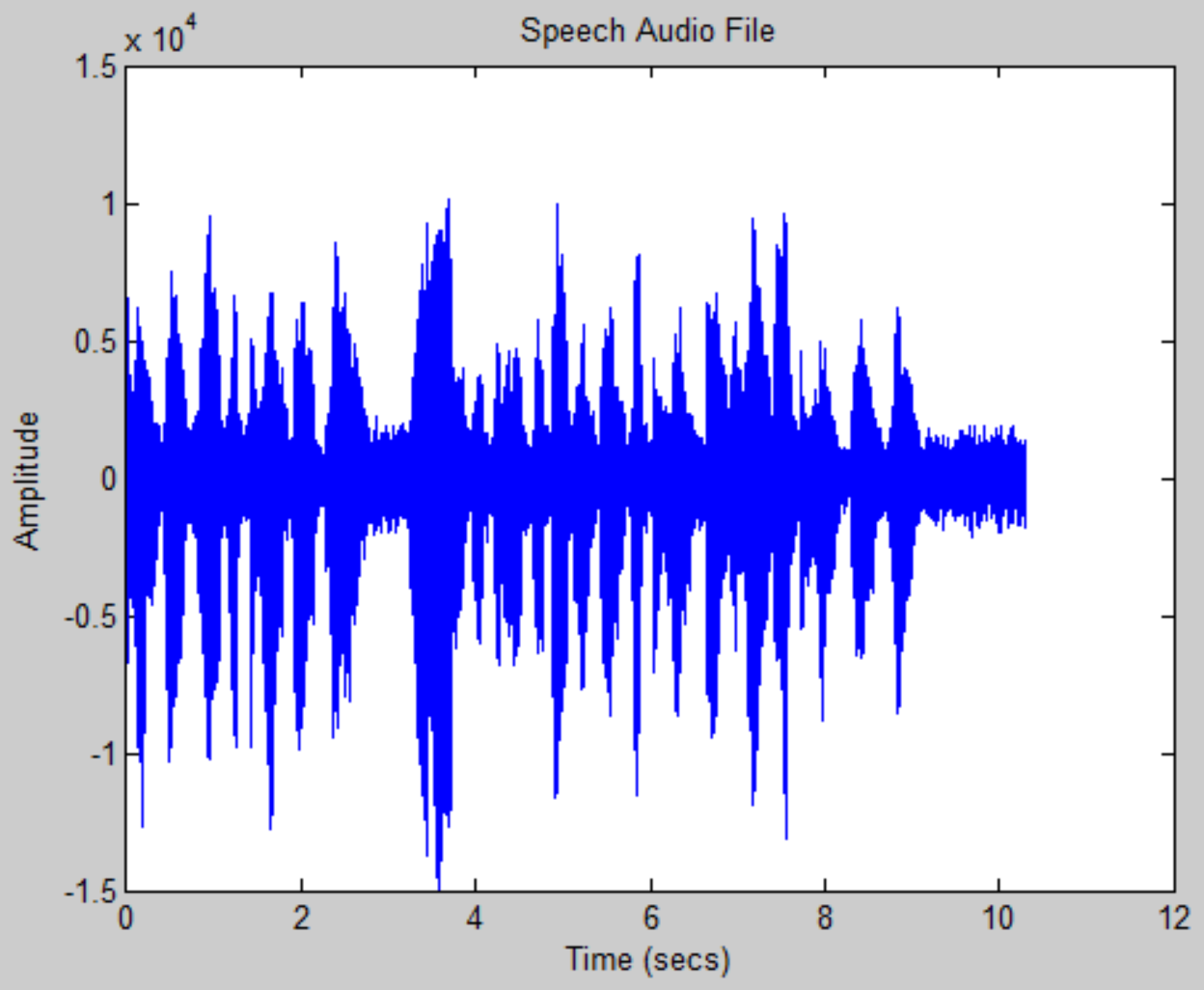
Department of Electrical and Computer Engineering, Temple University, Philadelphia, PA 19121

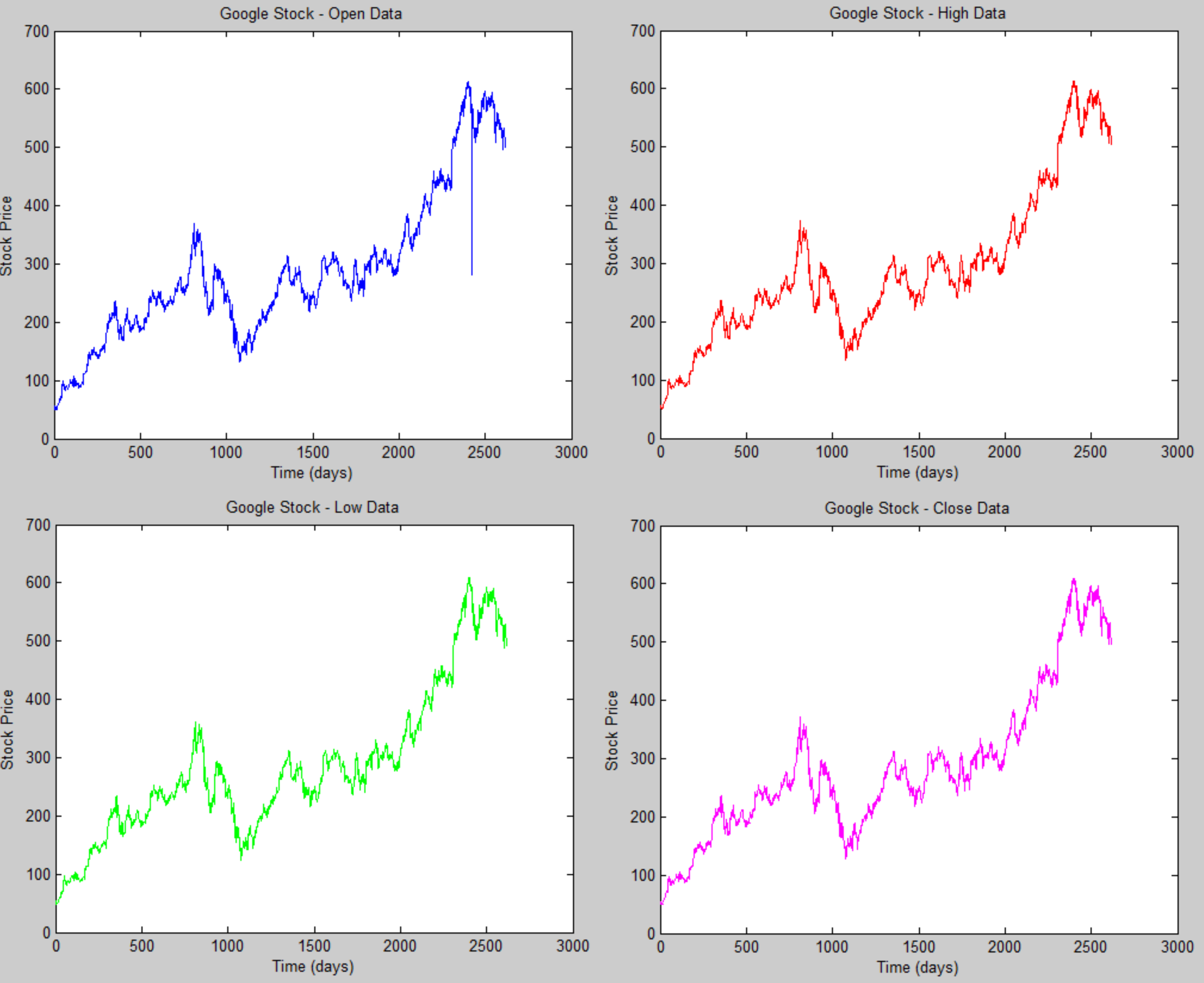
# Problem Statement

Two signals, an audio speech signal and Google stock prices data (provided in the assignment) are to be plotted and analyzed for their global stats using MATLAB. Both signals are to be plotted using MATLAB and once plotted, their minimum, maximum, mean, median and variance values are computed. Both signals are then further analyzed using windows and frames at different values to see how different resolutions alter the data display, as well as finding their mean and variances for each window and frame.

# Approach and Results

The first part of this assignment is to import the audio file and stock prices file onto MATLAB, read the files and plot them as is. Figures 1a shows the plot for the audio speech file and Figure 1b is the Google stock prices plot for its four sets of data: open, high, low and close.

  
Figure 1a. A plot of the audio speech file in respect to time.

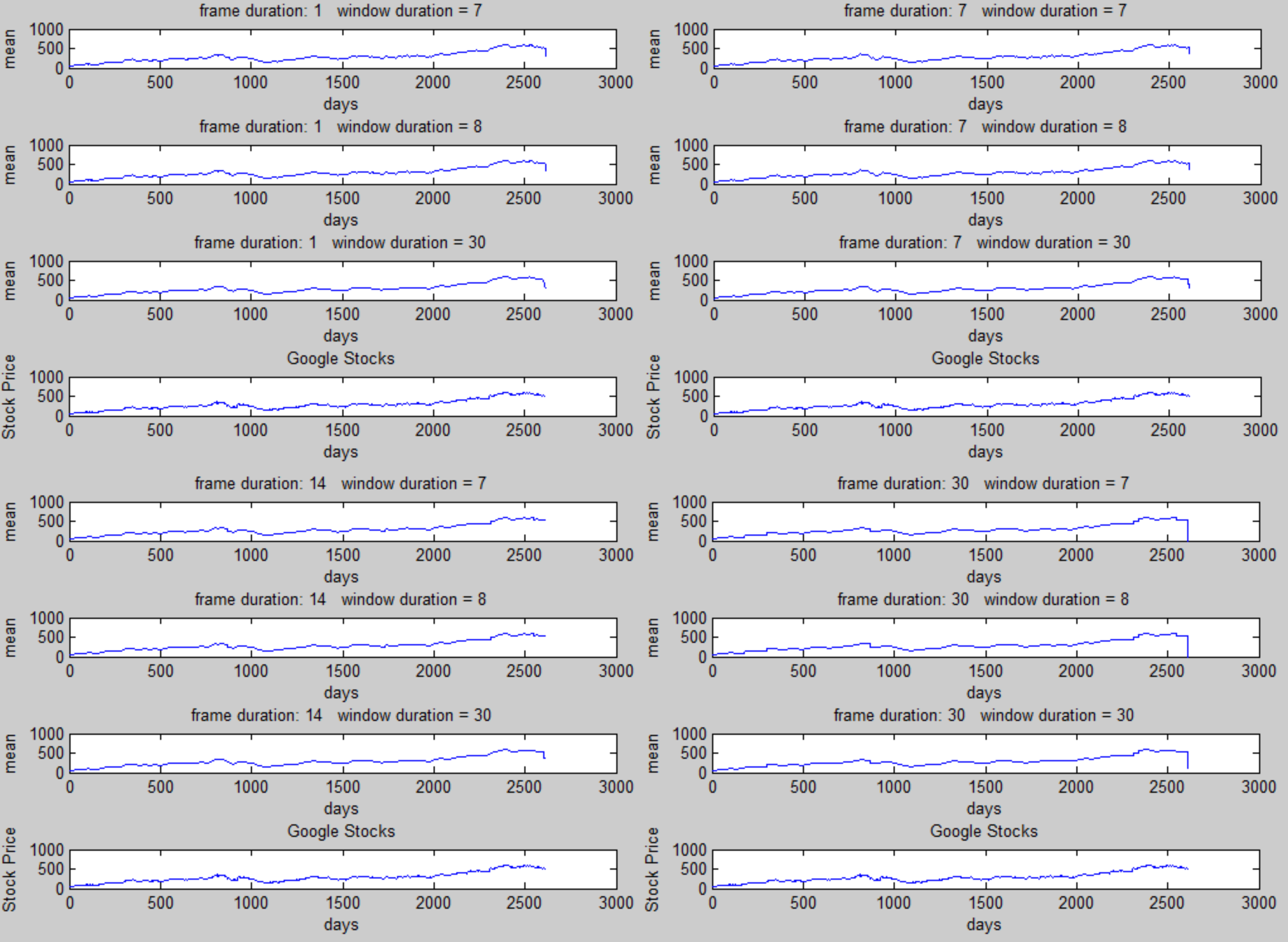
  
Figure 1b. The 4 plots of Google Stock prices data from 2004 to 2014: open data (top left), high data (top right), low data (bottom left), and close data (bottom right).

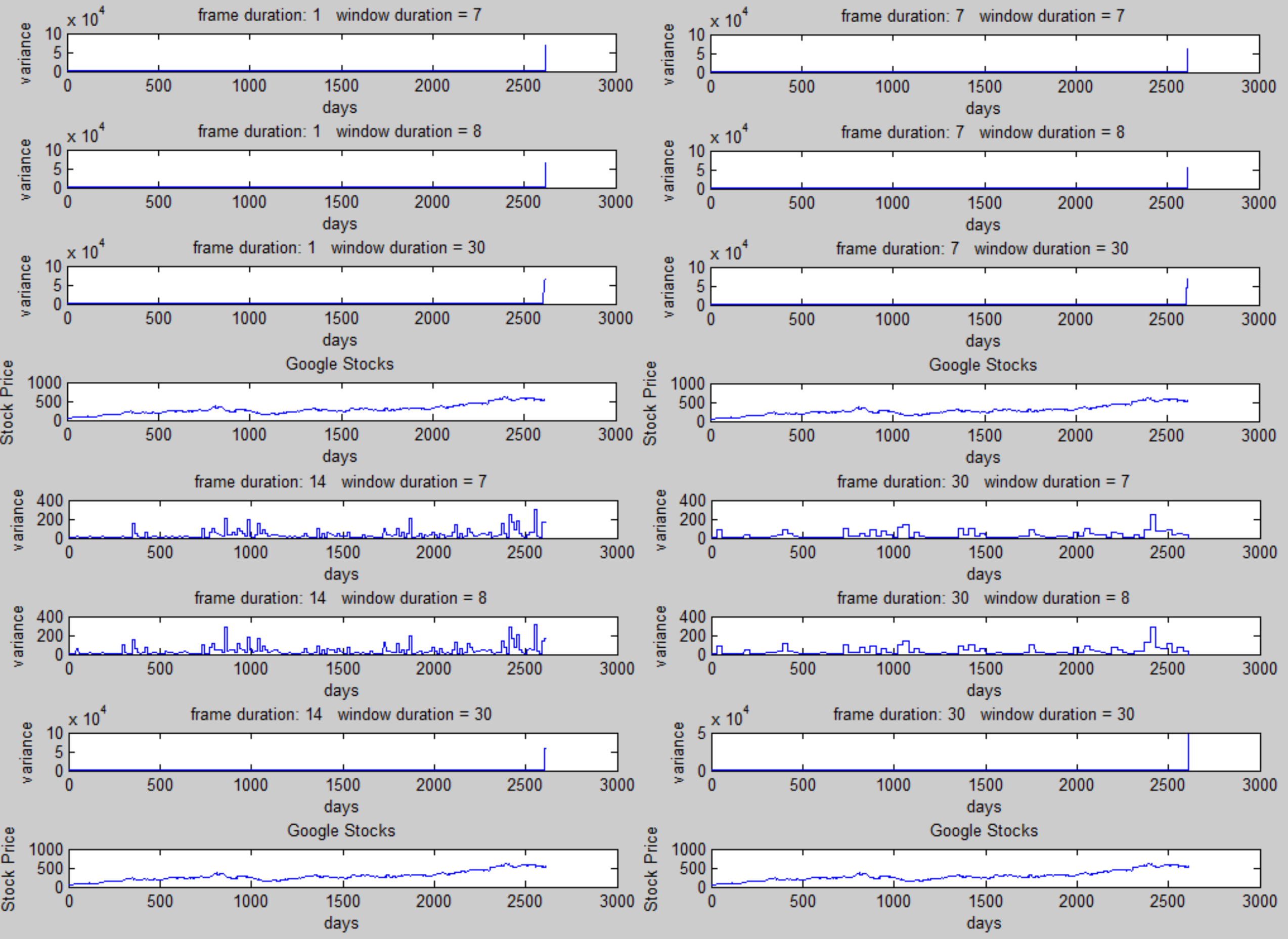
Question two of this assignment was to compute the minimum, maximum, mean, median and variance values of the audio speech file and the Google stock prices. The signals were imported into MATLAB the same way they were in question 1. These global data values were found using their respective MATLAB functions. Table I lists these values.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table I. Global Data of the Audio Speech File and Google Stock Prices** | | | | | |
|  | **Audio Speech File** | **Google Stock Prices** | | | |
| **Open** | **High** | **Low** | **Close** |
| **Minimum** | -14993 | 49.5 | 50.8200 | 47.9300 | 49.9500 |
| **Maximum** | 10104 | 612.7900 | 613.8300 | 608.6900 | 609.4700 |
| **Mean** | -0.3891 | 286.7918 | 289.6464 | 283.8242 | 286.7374 |
| **Median** | 83 | 265.0650 | 267.9800 | 262.3100 | 264.8250 |
| **Variance** | 4.1394e06 | 1.6198e04 | 1.6345e04 | 1.6030e04 | 1.6194e04 |

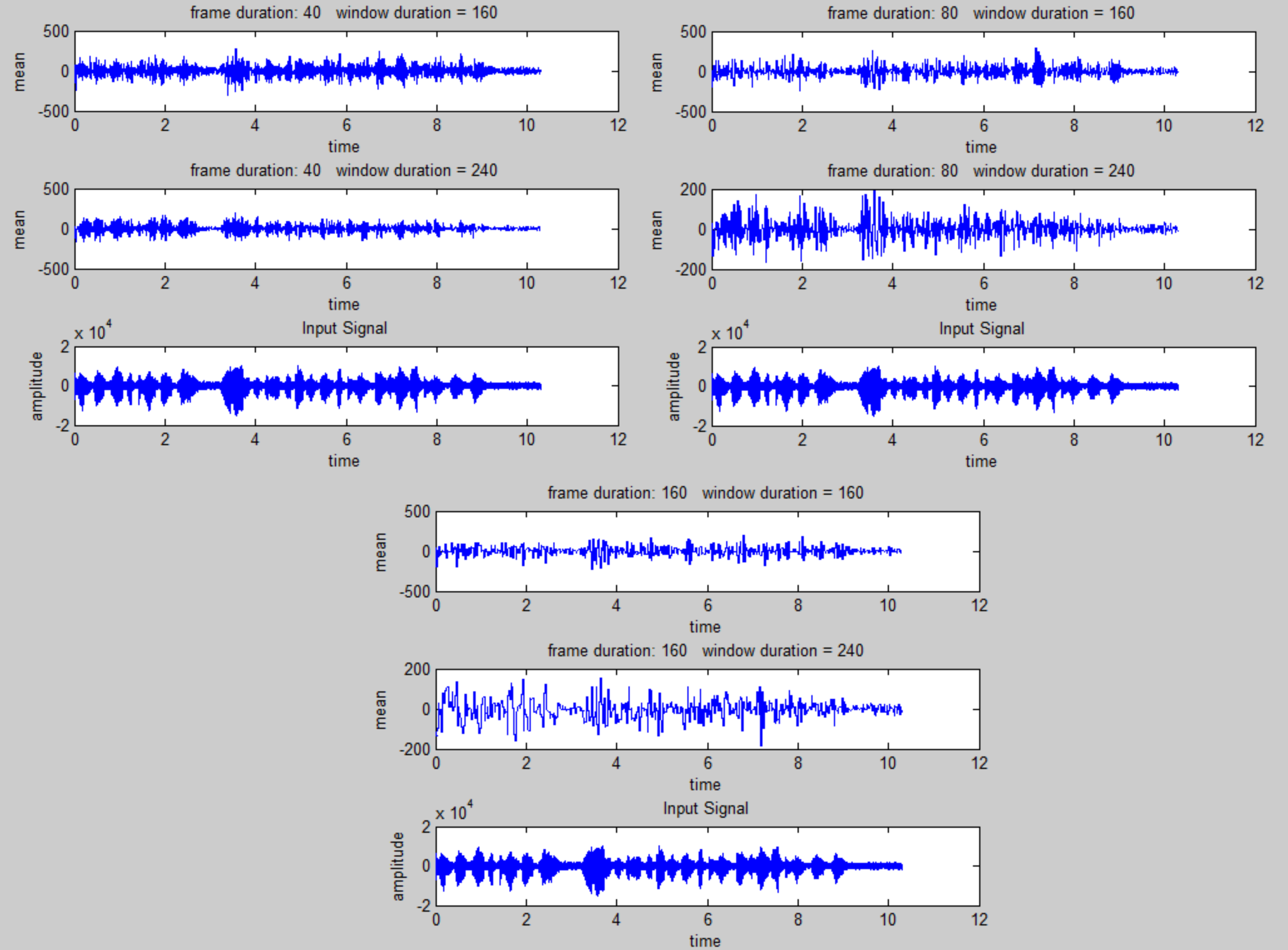
For the last question of this assignment, the close data of the Google stock prices and the speech audio file were analyzed with different windows and frame values. The window values the stock prices were analyzed under were N = 7 and 30 days and the frame values were M = 1, 7, 14, and 30 days. For the audio file it was analyzed at window values N = 160 and 240 msecs and frame values of M = 40, 80, and 160 msecs. The stock prices window and frame results for the mean and variance is shown in Figure 2a and 2b. Figure 3a shows the window and frame results of the audio signal’s mean and Figure 3b shows the window and frame results of its variance.

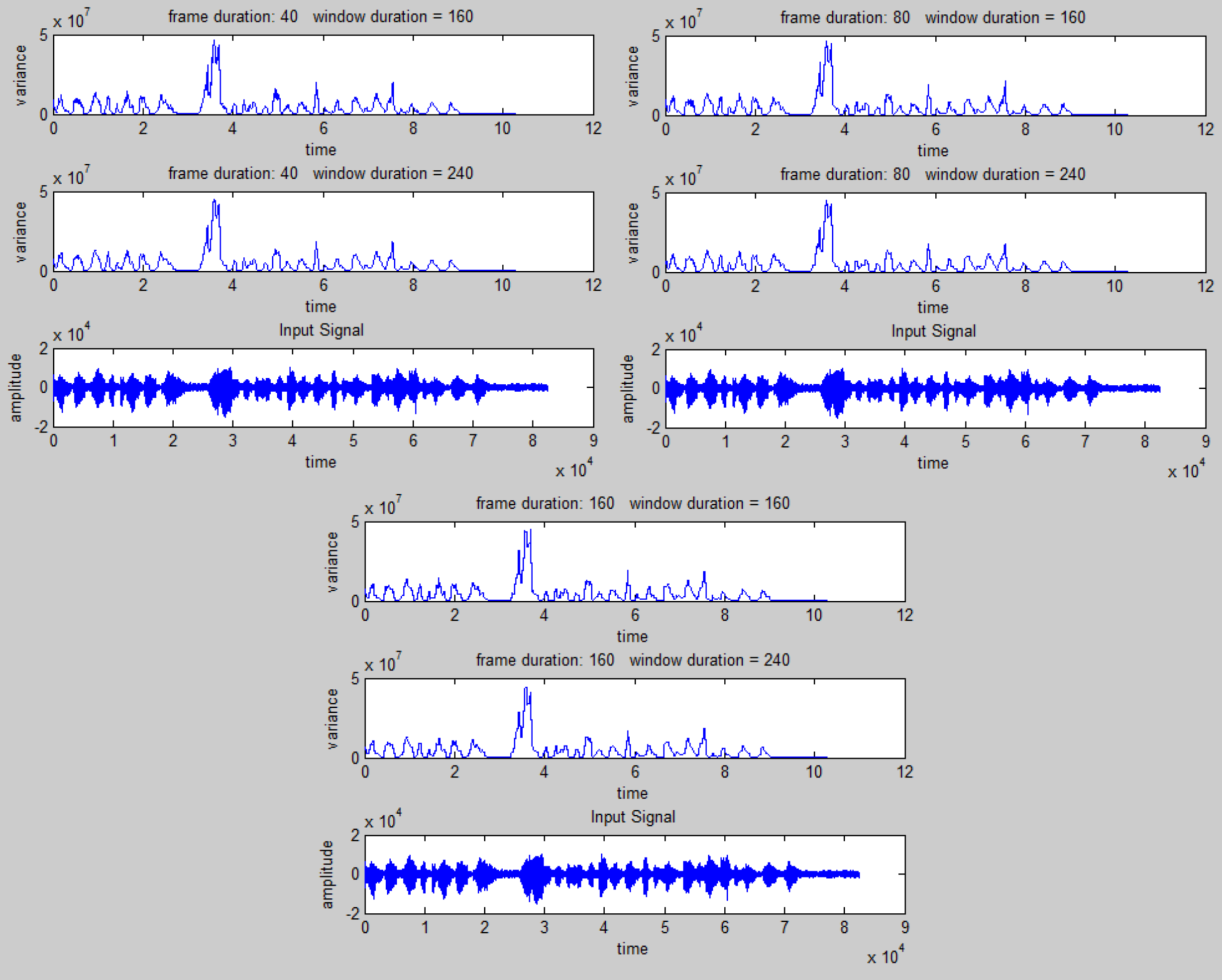
In Figure 2a, it makes sense that the mean plot increased as time (days) progresses because that is what happened shown in Figure 1b, the actual plots of the Google stock prices. The mean was computed for each individual window size, and since the stock prices kept increasing, so did the mean. For Figure 2b, the variance plots varied depending on the frame and window values were used. When the window size was larger than the frame size, the variance plots resulted in being about 0, signifying that the stock prices during those times were exact and there were barely any stock price changes. This is essentially true since the stock market prices gradually increased and the prices were about the same day after day. However, for the plots with a small frame size and large frame size, it would hover around 0 and immediately become nonzero at the last data value, saying there was only a major change from day one to present day. This is known to be false based on the plotted data of the Google stock prices in Figure 1b. As the frame size got bigger and the window sizes were smaller or not significantly larger than the frame size, the variance plot showed non-zero values. This is expected and more reliable data since in the actual Google stock price plot shown in Figure 1b, once can see that the stock prices fluctuated.

  
Figure 2a. The mean of the Google stock prices close data plotted at different windows and frames values.

  
Figure 2b. The variance of the Google stock prices close data plotted at different windows and frames values.

Variance is defined as taking the square of the difference of each data point from the mean and averaging them. Since this involves squaring values (therefore, no negative values), it has a strong resemblance to the energy plot of a signal since it involves squaring the absolute values of the data points for a time series. Comparing the mean of the signal, shown in Figure 3a, to its original plot, Figure 1a, the amplitude of the signal decreases when it is averaged. Averaging the signal is supposed to help remove the additive noise when the signal is being transmitted or received. Figure 3a displays the mean of the audio signal at various frame and window values. As stated before, the bigger the window size is compared to the frame size, the more averaged and smoothed out the plot becomes. This trend is prominent when comparing the plots for when window size is 240. The more the frame size and window size are closer in value, the more noise appears in the plotted signal.

  
Figure 3a. The mean plot of the audio signal plotted at different windows and frames values.

  
Figure 3b. The variance plot of the audio signal plotted at different windows and frames values.

# MATLAB Code

Question 1

The main functions used to import and read the audio file into MATLAB were fopen, which opens the file, and fread, while reads the file, into the variable “sig” as shown in the MATLAB code in Figure 4. The audio file is known to be sampled at 8 kHz, therefore, the time axis to plot the signal had to be adjusted. This was done by creating the variable “time,” making it a vector the length of the audio file, divided by 8 kHz. Once that was done, the signal was plotted using the plot function on MATLAB.

For the excel spreadsheet file containing the Google stock prices, the xlsread function was used. In order to read the four columns of data on the spreadsheet: open, high, low and close, the variables, “op,” “hi,” “lo,” and “cl” were used to designate what columns on the spreadsheet should be read, respectively. The data were then read into their own respective variables and all four of them were individually plotted.

|  |
| --- |
| %% CA Question 1 part a  %plot audio signal  clear; clc  close all    %reading in raw speech file  filename = 'rec\_01\_speech.raw';  fileID = fopen(filename);  sig = fread(fileID, inf, 'int16');  % always good to close the file  fclose(fileID);  time = 0:length(sig)-1;  time = time/(8e3);  time = time';    % plotting audio signal  plot(time,sig)  title('Speech Audio File')  ylabel('Amplitude')  xlabel('Time (secs)')    %% Question 1 Part b  % plot google stock prices  clear; clc  close all    %reading xlsx file  op = 'B:B'; %column B identifier  hi = 'C:C'; %column C identifier  lo = 'D:D'; %column D identifier  cl = 'E:E'; %column E identifier    %load google stock price data  open = xlsread('google\_v00.xlsx',1,op);  high = xlsread('google\_v00.xlsx',1,hi);  low = xlsread('google\_v00.xlsx',1,lo);  cloze = xlsread('google\_v00.xlsx',1,cl);    figure(1)  plot(open)  xlabel('Time (days)')  ylabel('Stock Price')  title('Google Stock - Open Data')  figure(2)  plot(high, 'r')  xlabel('Time (days)')  ylabel('Stock Price')  title('Google Stock - High Data')  figure(3)  plot(low, 'g')  xlabel('Time (days)')  ylabel('Stock Price')  title('Google Stock - Low Data')  figure(4)  plot(cloze, 'm')  xlabel('Time (days)')  ylabel('Stock Price')  title('Google Stock - Close Data') |
| Figure 4. MATLAB code for Question 1. |

Question 2

The same method and code to read to spreadsheet and audio file in question 1 were used for Question 2 to compute the global stats of both files: their minimum, maximum, mean, median and variance values.

The MATLAB functions: min, max, mean, median and var were used to find these values for both files and each value was stored in their respected variables, as seen in Figure 5. Overall, this question was done and answered by a sequence of function calls.

|  |
| --- |
| %% CA Question 2  clear;clc;  close all    op = 'B:B'; %column B identifier  hi = 'C:C'; %column C identifier  lo = 'D:D'; %column D identifier  cl = 'E:E'; %column E identifier    % load google stock price data  open = xlsread('google\_v00.xlsx',1,op);  high = xlsread('google\_v00.xlsx',1,hi);  low = xlsread('google\_v00.xlsx',1,lo);  cloze = xlsread('google\_v00.xlsx',1,cl);    % load raw audio data  a=fopen('rec\_01\_speech.raw');%create signal identifier  speech=fread(a,inf,'short'); %open signal as short  Fs=8e3; %sampling frequency  fclose(a);    % compute global stats of signals: minimum,  % maximum, mean, median, variance    % google stock price stats  open\_min = min(open)  open\_max = max(open)  open\_mean = mean(open)  open\_median = median(open)  open\_var = var(open)    high\_min = min(high)  high\_max = max(high)  high\_mean = mean(high)  high\_median = median(high)  high\_var = var(high)    low\_min = min(low)  low\_max = max(low)  low\_mean = mean(low)  low\_median = median(low)  low\_var = var(low)    cloze\_min = min(cloze)  cloze\_max = max(cloze)  cloze\_mean = mean(cloze)  cloze\_median = median(cloze)  cloze\_var = var(cloze)    % audio stats  speech\_min = min(speech)  speech\_max = max(speech)  speech\_mean = mean(speech)  speech\_median = median(speech)  speech\_var = var(speech) |
| Figure 5. MATLAB code for Question 2. |

Question 3

|  |
| --- |
| function mean\_value=CA1\_p3\_audiomean\_v01  close all;  % 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 = [ 40, 80, 160 ];  N = [ 160, 240 ];    filename = 'rec\_01\_speech.raw';  fp = fopen(filename,'r');  sig = fread(fp,inf,'int16');  fclose(fp);    % setting up time scale for plots  time = 0:length(sig)-1;  time = time/(8e3);  time = time';    % create a matrix to store the output  mean\_value = zeros(length(M), length(N), length(sig));    % loop over the a set of frame/window combinations.  for m = 1:length(M)  % set up a plotting window and label it  h1 = figure('name', 'mean plot', 'numbertitle', 'off');  for n = 1:length(N)  % call a function to compute mean of signal  mean\_value(m,n,:) = compute\_mean(sig, M(m), N(n));  % label the plot:  % include info about the parameters for each plot  figure(h1);  str = sprintf('frame duration: %d window duration = %d', M(m), N(n));  subplot( 1+length(N), 1, n );  % plot the mean  plot(time,squeeze(mean\_value(m,n,:)));  title(str);  xlabel('time');  ylabel('mean');  end    % plot the signal:  figure(h1);  subplot( 1+length(N),1,n+1);  plot(time, sig);  title('Input Signal');  xlabel('time');  ylabel('amplitude');  end  mean\_out = squeeze(mean\_value);    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)  %  % This algorithm computes the mean 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 = (i - 1) \* fdur\_a + (fdur\_a / 2);  n\_left = n\_center - (wdur\_a / 2);  n\_right = 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);  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 = n\_center + (j - 1) - (fdur\_a/2);  if ((index > 0) && (index <= num\_samples))  mean\_full(index) = mean\_val;  end  end  end    end |
| Figure 6. The general MATLAB code algorithm used to compute the mean and variance for the audio file and the close data of the Google Stock prices for Question 3. This code was provided from recitation 2 from Signals. |

This code was originally coded to compute the rms of the signal at different windows and frames values. What was generally changed in this code was changing the compute\_rms function in this code to compute the mean and variance of the signal, using their respected MATLAB functions. In addition, the M and N vector values were changed to the desired values stated in the computer assignment. Before modifying the code, a good understanding, testing it and running through the code was needed in order to properly use it to answer Question 3.

How the mean/variance values were computed was that the signal was looped over for each set of frames and windows by a for loop statement, which called the compute\_mean/var function to do the actual calculations. In that coded function, the matrix of mean values was created based on the passed in values for the frame and windows, scanning through the signal. Once that was done, the newly constructed matrix is passed out of the function, which is then plotted and displayed in the main function of the program.

# Conclusions

Variance tends to look similar to the energy plot of a signal based on the fact that computing both the energy and variance of a signal is similar since it involves squaring and averaging for a time series signal. Averaging and computing the mean of a signal tends to remove the noise that is added on to the signal when it was transmitted. For representation, the difference between the values found in Question 2 and the plotted results in Question 3 is that the mean and variance values were found all throughout the data for Question 2, while the mean and variance were found at specific intervals of both files. Both sets of calculated data are correct since they analyzed the same two files, it is just that they computed the data looking at different aspects of the two files. This is prominent in figures 2a, 2b, 3a and 3b when the windows and frames changed. Each graph looks similar in terms of shape, but they look different because some appear smoother than others, while some look blocky. With windows and frames, it is preferred to have a window size bigger than the frame size because the plotted data will appear smoother due to the overlapping of the adjacent windows, making the data more accurate. However, when the frame size is larger (or the window size is smaller) the windows where the signal is being analyzed, it will not overlap with the adjacent windows, leaving gaps in between. Those gaps do not get analyzed, making the graph appear choppy due to the inaccuracies. Although, it should be stated that having too big of a window size is bad too because it will average the values too much, losing the specific details that may be needed for analyzing. Therefore, the window and frame size always varies depending on what exactly is desired.