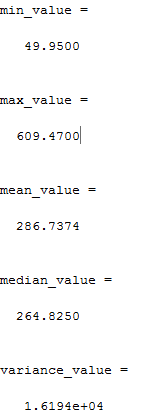
**Harish Boggarapu**

**ECE 3522**

**Computer Assignment #1**

Data 1: Google Stock price data values



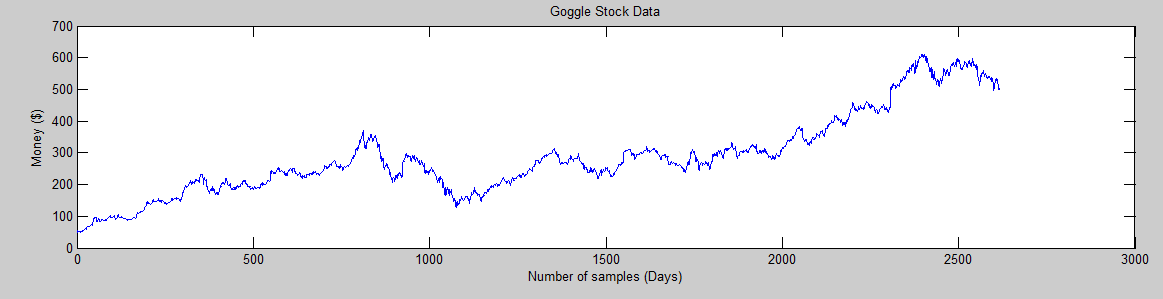


Figure 1: Plot of Google stock prices in MATLAB

Figures 3-6 illustrate the plots of mean for different combinations of frame durations and window durations.

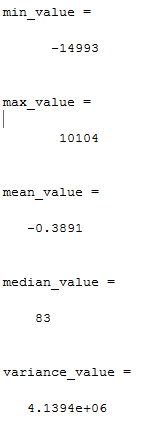
Figures 7-10 illustrate the plots of variance for different combinations of frame durations and window durations.

|  |  |
| --- | --- |
| Figure 3 | Figure 4 |

|  |  |
| --- | --- |
| Figure 5 | Figure 6 |
| Figure 7 | Figure 8 | |

|  |  |
| --- | --- |
| **Figure 9** | **Figure 10** |

Data 2: Speech signal values



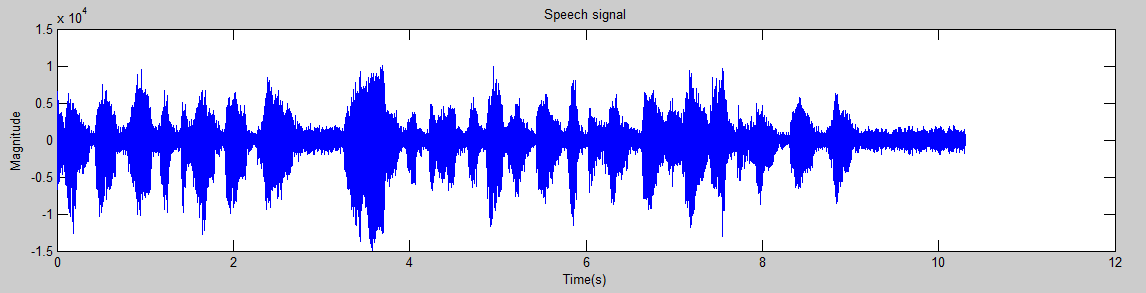


Figure 2: Plot of speech signal in MATLAB

Figures 11-13 illustrate the plots of mean for different combinations of frame durations and window durations.

Figures 13-16 illustrate the plots of variance for different combinations of frame durations and window durations.

|  |  |
| --- | --- |
| Figure 11 | Figure 12 |

|  |  |  |
| --- | --- | --- |
| Figure 13 | | Figure 14 |
| Figure 15 Figure 16 | |

Analysis:

The plots of the mean and variance of both data’s are showed that at lower window and frame parameters, the graphs were much smoother compared to the higher window and frame combinations. At higher window and frame parameters, the graphs sharp edges.

MATLAB code:

|  |
| --- |
| %% PART 1 DATA 1  close all;clear;  filename = 'google\_v00.xlsx';  % read excel file in matlab to perform and only colsing values  A = xlsread(filename,'E:E');  % plottting the data  figure(1)  subplot(2,1,1)  plot(A);  xlabel('Number of samples (Days)');ylabel('Money ($)');title('Goggle Stock Data');  % find min, max, mean, median, and variance  min\_value = min(A)  max\_value = max(A)  mean\_value = mean(A)  median\_value = median(A)  new\_file = A - mean\_value;  L = length(new\_file);  variance\_value = sum((new\_file).^2)/(L-1) |

|  |
| --- |
| %% PART 1 DATA 2  close all;clear;  fp = fopen('rec\_01\_speech.raw', 'r');  f = fread(fp, inf, 'int16');  fclose(fp);  Fs = 8000;  N = length(f);  t = (0:N-1)\*(1/Fs); % time vector  % plotting the audio signal  figure(2);  plot(t,f);  xlabel('Time(s)');ylabel('Magnitude');title('Speech signal');  % finding min, max, mean, median, and variance  min\_value = min(f)  max\_value = max(f)  mean\_value = mean(f)  median\_value = median(f)  new\_file = f - mean\_value;  L = length(new\_file);  variance\_value = sum((new\_file).^2)/(L-1) |

|  |
| --- |
| %% PART 2 DATA 1  %define frame duration and window duration  M = [ 1, 7, 14, 30];  N = [ 7, 30];    filename = 'google\_v00.xlsx';  % read excel file in matlab  f = xlsread(filename,'E:E');    %create a matrix to store mean values    mea\_n = zeros(length(M), length(N), length(f));  variance = zeros(length(M), length(N), length(f));  % 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');  h2 = figure('name', 'variance plot', 'numbertitle', 'off');  for n = 1:length(N)  % call a function to compute the rms vector  mea\_n(m,n,:) = compute\_mean(f, M(m), N(n));  variance(m,n,:) = compute\_variance(f, M(m), N(n));  % label the plot:  % include information 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 contour  plot(squeeze(mea\_n(m,n,:)));  % label the axes  title(str);  xlabel('time (Days)');  ylabel('mean');  figure(h2);  str = sprintf('frame duration: %d window duration = %d', M(m), N(n));  subplot(1 + length(N), 1, n);  % plot the mean contour  plot(squeeze(variance(m,n,:)));  % label the axes  title(str);  xlabel('time (Days)');  ylabel('variance');  end  % plot the signal:  % this is the last plot on the page  figure(h1);  subplot(1 + length(N), 1, n+1);  plot(f);  % label the axes  title('Google Stock Data');  xlabel('time (Days)');  ylabel('Money ($)');  figure(h2);  subplot(1 + length(N), 1, n+1);  plot(f);  % label the axes  title('Google Stock Data');  xlabel('time (Days)');  ylabel('Money ($)');  end |

|  |
| --- |
| %% PART 2 DATA 2  close all;  clear;clc    %define frame duration and window duration  M = [ 40, 80, 160];  N = [ 160, 240];    fp = fopen('rec\_01\_speech.raw', 'r');  f = fread(fp, inf, 'int16');  fclose(fp);    %create a matrix to store mean values |

|  |
| --- |
| mea\_n = zeros(length(M), length(N), length(f));  variance = zeros(length(M), length(N), length(f));  % 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');  h2 = figure('name', 'variance plot', 'numbertitle', 'off');  for n = 1:length(N)  % call a function to compute the rms vector  mea\_n(m,n,:) = compute\_mean(f, M(m), N(n));  variance(m,n,:) = compute\_variance(f, M(m), N(n));  % label the plot:  % include information 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 contour  plot(squeeze(mea\_n(m,n,:)));  % label the axes  title(str);  xlabel('time');  ylabel('mean');  figure(h2);  str = sprintf('frame duration: %d window duration = %d', M(m), N(n));  subplot(1 + length(N), 1, n);  % plot the mean contour  plot(squeeze(variance(m,n,:)));  % label the axes  title(str);  xlabel('time');  ylabel('variance');  end  % plot the signal:  % this is the last plot on the page  figure(h1);  subplot(1 + length(N), 1, n+1);  plot(f);  % label the axes  title('Input Signal');  xlabel('time');  ylabel('amplitude');  figure(h2);  subplot(1 + length(N), 1, n+1);  plot(f);  % label the axes  title('Input Signal');  xlabel('time');  ylabel('amplitude');  end |

|  |
| --- |
| %function to compute mean value  function sig\_mean = compute\_mean(sig\_a, fdur\_a, wdur\_a)  % declare local variables  sig\_wbuf = zeros(1, wdur\_a);  num\_samples = length(sig\_a);  num\_frames = round(num\_samples / fdur\_a);  sig\_mean = zeros(length(sig\_a), 1);  % loop over the entire signal |

|  |
| --- |
| for i = 1 : num\_frames  % generate pointers to the beginning and end of the frame/window:  % These pointers control how we move through the signal.  % For example, left and right refer to the beginning and end of  % a frame/window.  n\_frame\_left = (i - 1) \* fdur\_a + 1;  n\_frame\_right = n\_frame\_left + fdur\_a - 1;  n\_frame\_center = (n\_frame\_right + n\_frame\_left) / 2;  n\_win\_left = floor(n\_frame\_center - (wdur\_a / 2) + 1);  n\_win\_right = n\_win\_left + wdur\_a - 1;    % zero-stuff the data if necessary:  % 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\_win\_left < 0) || (n\_win\_right > num\_samples))  sig\_wbuf = zeros(1, wdur\_a);  end    % transfer the data to this buffer:  % note that this is really expensive computationally  %  j = 1;  index = n\_win\_left;  while (j <= wdur\_a)    % transfer the signal to the processing buffer:  % note that we must make sure we do not exceed the  % limits of the signal for the first and/or last frame  %  if ((index >= 1) && (index <= num\_samples))  sig\_wbuf(j) = sig\_a(index);  end    % increment the counters  %  index = index + 1;  j = j + 1;  end    % compute the rms:  % square the signal. divide it by the number of samples used and sum  % the result to build the value for that frame  %  mea\_n = sum(sig\_wbuf)/length(sig\_wbuf);    % assign the rms value to the output signal:  % note that we write fdur\_a values  %  for j = 1 : fdur\_a  index = n\_frame\_left + j - 1;  if ((index > 0) && (index <= num\_samples))  sig\_mean(index) = mea\_n;  end  end  end  end |

|  |
| --- |
| %function to compute variance value  function sig\_variance = compute\_variance(sig\_a, fdur\_a, wdur\_a)  % declare local variables  sig\_wbuf = zeros(1, wdur\_a);  num\_samples = length(sig\_a);  num\_frames = round(num\_samples / fdur\_a);  sig\_variance = zeros(length(sig\_a), 1);  % loop over the entire signal  for i = 1 : num\_frames  % generate pointers to the beginning and end of the frame/window:  % These pointers control how we move through the signal.  % For example, left and right refer to the beginning and end of  % a frame/window.  n\_frame\_left = (i - 1) \* fdur\_a + 1;  n\_frame\_right = n\_frame\_left + fdur\_a - 1;  n\_frame\_center = (n\_frame\_right + n\_frame\_left) / 2;  n\_win\_left = floor(n\_frame\_center - (wdur\_a / 2) + 1);  n\_win\_right = n\_win\_left + wdur\_a - 1;    % zero-stuff the data if necessary:  % 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\_win\_left < 0) || (n\_win\_right > num\_samples))  sig\_wbuf = zeros(1, wdur\_a);  end    % transfer the data to this buffer:  % note that this is really expensive computationally  %  j = 1;  index = n\_win\_left;  while (j <= wdur\_a)    % transfer the signal to the processing buffer:  % note that we must make sure we do not exceed the  % limits of the signal for the first and/or last frame  %  if ((index >= 1) && (index <= num\_samples))  sig\_wbuf(j) = sig\_a(index);  end    % increment the counters  %  index = index + 1;  j = j + 1;  end    % compute the rms:  % square the signal. divide it by the number of samples used and sum  % the result to build the value for that frame  %  variance = std(sig\_wbuf).^2;    % assign the rms value to the output signal:  % note that we write fdur\_a values  %  for j = 1 : fdur\_a  index = n\_frame\_left + j - 1;  if ((index > 0) && (index <= num\_samples))  sig\_variance(index) = variance;  end  end  end  end  %function to compute variance value  function sig\_variance = compute\_variance(sig\_a, fdur\_a, wdur\_a)  % declare local variables  sig\_wbuf = zeros(1, wdur\_a);  num\_samples = length(sig\_a);  num\_frames = round(num\_samples / fdur\_a);  sig\_variance = zeros(length(sig\_a), 1);  % loop over the entire signal  for i = 1 : num\_frames  % generate pointers to the beginning and end of the frame/window:  % These pointers control how we move through the signal.  % For example, left and right refer to the beginning and end of  % a frame/window.  n\_frame\_left = (i - 1) \* fdur\_a + 1;  n\_frame\_right = n\_frame\_left + fdur\_a - 1;  n\_frame\_center = (n\_frame\_right + n\_frame\_left) / 2;  n\_win\_left = floor(n\_frame\_center - (wdur\_a / 2) + 1);  n\_win\_right = n\_win\_left + wdur\_a - 1;    % zero-stuff the data if necessary:  % 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\_win\_left < 0) || (n\_win\_right > num\_samples))  sig\_wbuf = zeros(1, wdur\_a);  end    % transfer the data to this buffer:  % note that this is really expensive computationally  %  j = 1;  index = n\_win\_left;  while (j <= wdur\_a)    % transfer the signal to the processing buffer:  % note that we must make sure we do not exceed the  % limits of the signal for the first and/or last frame  %  if ((index >= 1) && (index <= num\_samples))  sig\_wbuf(j) = sig\_a(index);  end    % increment the counters  %  index = index + 1;  j = j + 1;  end    % compute the rms:  % square the signal. divide it by the number of samples used and sum  % the result to build the value for that frame  %  variance = std(sig\_wbuf).^2;    % assign the rms value to the output signal:  % note that we write fdur\_a values  %  for j = 1 : fdur\_a  index = n\_frame\_left + j - 1;  if ((index > 0) && (index <= num\_samples))  sig\_variance(index) = variance;  end  end  end  end |