Alexander Smith

ECE 3522: STOCHASTIC PROCESSES IN SIGNALS AND SYSTEMS

**COMPUTER ASSIGNMENT (CA) NO. 3: VARIANCE**

Department of Electrical and Computer Engineering, Temple University, Philadelphia, PA 19122

# Problem Statement

We are given two files: one .xls and one .raw file. The first file’s data set represents Google’s stock price since its inception. The second file’s data is speech sampled at 8 kHz. We are asked to compute and plot the variance of each signal in three different ways:

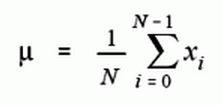
### as the total numerical value of the signal(considering all of its samples), plotted as a horizontal line

### the variance of sample 1 to N, with N going up to the last last sample, overlaid with the numerical plot.

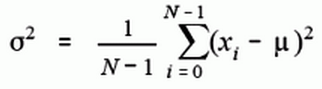
### using a frame-window approach, frame of 1 and window of 30 for the stock, and frame of 10 window of 30 for the speech file

The first task can easily be done using the *var* function or the equations below. I did both in MATLAB to make sure the function computed the same value as the equation. The second task is similar to the first except a for loop must be used to perform the calculation for the increasing amount of the samples being used. I modified the old framing and windowing RMS code from Recitation 2 of ECE 3512 to accomplish the third variance task. I replaced the RMS part of the code with the following math in MATLAB.

Equation for the mean of a signal:



Equation for the variance of a signal, the standard deviation squared:



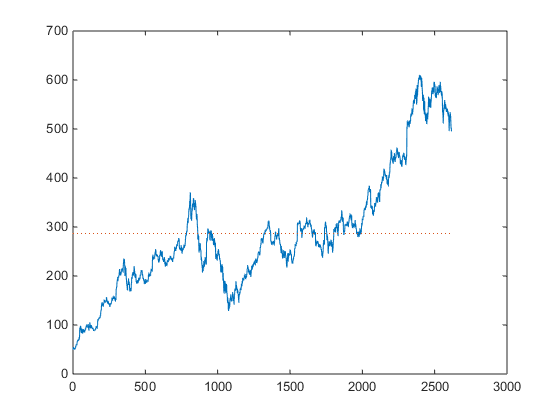


Figure . Closing stock input signal with its mean line

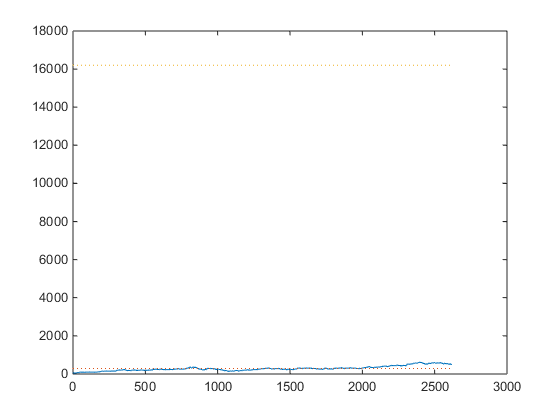


Figure . Stock variance line overlaid with original signal and mean line

Stock variance = 1.6194.46687231580E+4

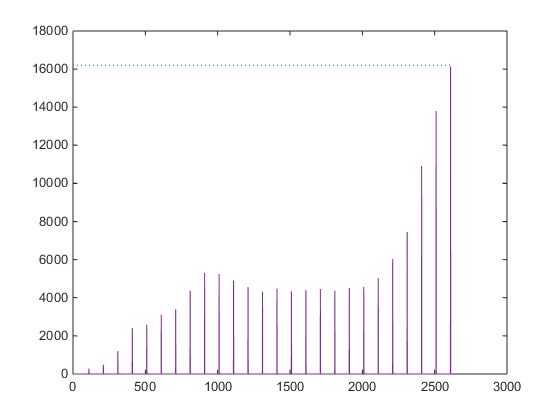


Figure 3. Variance as a function of ‘N’ incremented by 100. Partial variance (purple) converges to total data variance value (blue dots) as more of the stock data samples are considered.

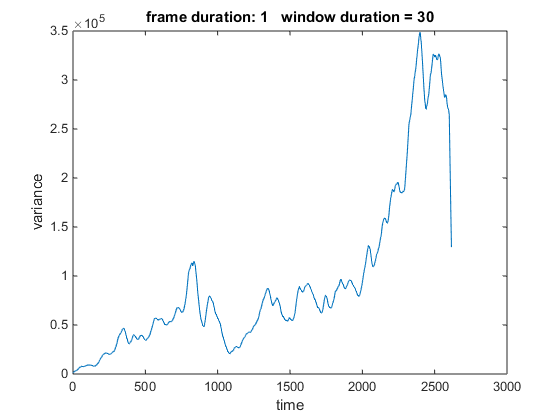


Figure . Plot from frame-window approach plot for the stock data variance

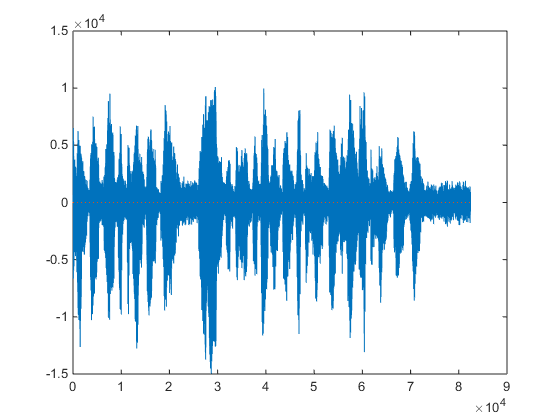


Figure . Audio input signal with its mean line (approximately zero)

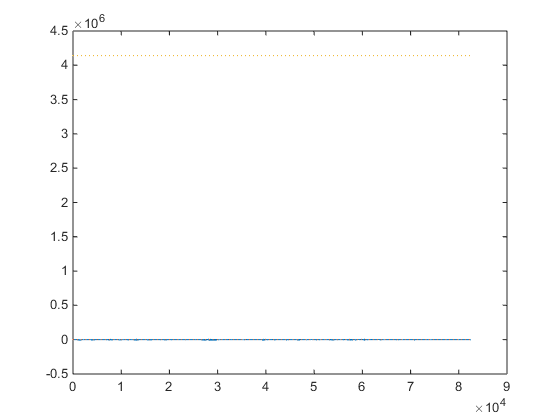


Figure . Variance line (top) overlaid speech signal and mean (bot.).

Speech variance = 4.139362775439662e+06

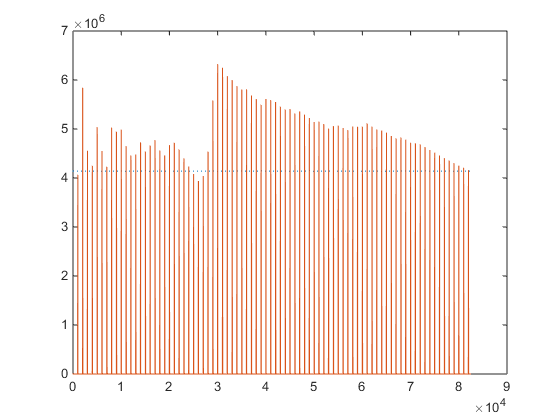


Figure 7. Variance as a function of ‘N’. Partial variance (red) converges to total data variance value (blue) as more of the speech file data samples are considered.

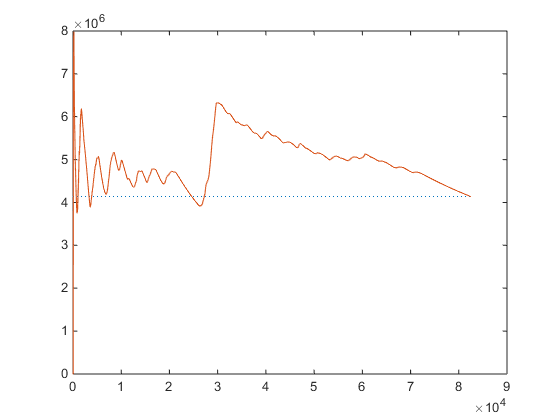


Figure . Variance as fn. N again but with every sample considered.

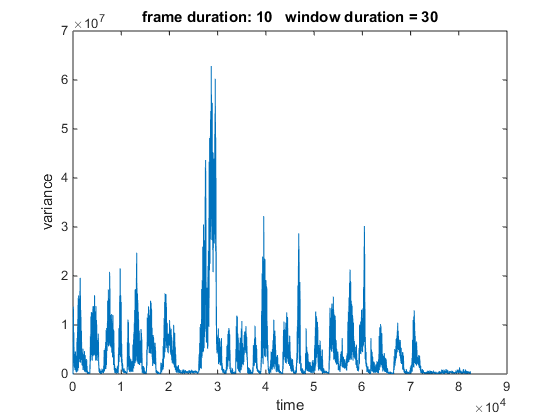


Figure 9. Plot from frame-window approach plot for speech file variance

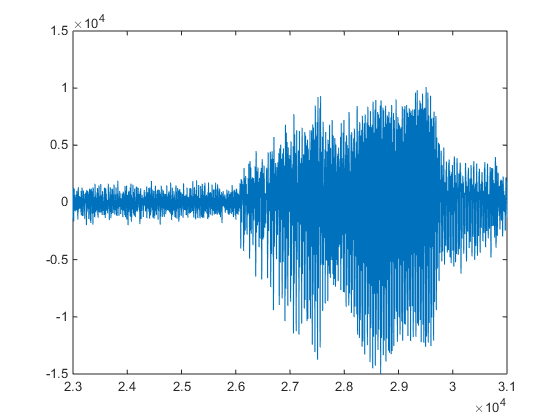


Figure . Closer look at the speech signal, around time 2.5-3, corresponding to the steepest domain section of the function of N variance plot.

The results of this assignment show that the variance of signals is the power of the signal. The stock signal’s overall shape could be seen its function of N variance plot. The speech signal’s negative part makes it a little difficult to see the same variance behavior as seen for the stock, but it can still be recognized.

# MATLAB Code

%% Pb 1 - Variance Plotted as a hor. line

clear;close all;clc;

% Stock Data

%

stock = xlsread('google\_v00.xlsx');

sig = stock(:,4);

%

%---------------------

%{

% Speech Signal

fp = fopen('rec\_01\_speech.raw','r');

sig = fread(fp,inf,'int16');

fclose(fp);

%}

%---------------------

plot(sig);

hold on

mean\_s = mean(sig)

plot(ones(size(sig))\*mean\_s,':');

%{

vari = 0;

for n=1:(length(sig)) %this loop does same thing as var fn

vari = vari + (sig(n)-mean\_s).^2;

end

vari=vari/(n-1)

%}

vari=var(sig)

figure(2)

plot(ones(size(sig))\*vari,':')

%% Pb 2 - Variance as a function of N.

%{

- Where N is the furthest point in the sample space of the data used to

calculate variance. Fn takes variance using sample 0-N of the data. N goes

from 10 to the last # sample of the data, sample #2616 for the stocks

%}

Nval = 1:length(sig);

vval = (1:length(sig))\*0;

for N=10:(length(sig))

%{

mean\_s = mean( sig(1:N) );

vari=0;

for n=1:N %this loop does same thing as var fn

vari = vari + (sig(n)-mean\_s).^2;

end

%}

%vval(N)=vari/(n-1);

vval(N)=var(sig(1:N));

end

hold on

plot(Nval,vval);

%% Investigate raw area w/ steep var change

%{

pcosig = sig(2.3e4:3.1e4;)

a=linspace(2.3e4,3.1e4,3.3e4-2.5e4+1);

plot(a,pcosig);

%}

% Pb 3 - Frame and Window Approach

function vari=rec02\_v05

% close open sessions

%

close all;

%

% For Speech RAW file:

% 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 = [ 10 ];

N = [ 30 ];

% 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.

%

fp = fopen('rec\_01\_speech.raw','r');

sig = fread(fp,inf,'int16');

% close the file:

% It is never a good idea to leave open files since they consume operating

% system resources and can get corrupted occassionally.

%

fclose(fp);

%

%{

% For Stock data

stock = xlsread('google\_v00.xlsx');

sig = stock(:,4);

M = [ 1 ];

N = [ 30 ];

%}

% create a matrix to store the output

%

vari = 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', 'vari plot', 'numbertitle', 'off');

for n = 1:length(N)

% call a function to compute the vari vector

% \* note that "vari" is the name of the matrix we made above, not

% the matlab vari function

%

vari(m,n,:) = compute\_vari(sig, 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 vari contour

%

plot(squeeze(vari(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);

plot( 1+length(N),1,n+1);

plot(sig);

% label the axes

%

title('Input Signal');

xlabel('time');

ylabel('amplitude');

%}

end

vari\_out = squeeze(vari);

% exit gracefully

%

end

% function: compute\_vari

%

% arguments:

% sig\_a: the input signal (input)

% fdur\_a: the frame duration in samples (input)

% wdur\_a: the window duration in samples (input)

%

% return:

% vari: a vector of vari values (output)

%

% Note that this function returns the vari 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.

%

%

function vari\_full = compute\_vari(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);

vari\_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 ((round(index) > 0) && (round(index) <= num\_samples))

sig\_wbuf(j) = sig\_a(round(index));

end

end

% square the signal. divide it by the number of samples used and sum

% the result to build the value for that frame

%

mean\_wbuf = (sig\_wbuf)/wdur\_a;

vari = ( 1/(wdur\_a-1) ) \* sum((sig\_wbuf-mean\_wbuf).^2);

% assign the vari 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))

vari\_full(index) = vari;

end

end

end

% exit gracefully

%

end

# Conclusions