Digital Signal Processing Laboratory

EXPERIMENT-2

Shorya Sharma
19EE01017

Time domain analysis of signal

1) Short-term magnitude (stm):

Definition:

Short term magnitude gives the sum of absolute values of all the samples present in the input sequence.

Mathematical expression:

Let the given input signal is x[n] consisting of N samples. If the input signal is divided into M blocks each consisting of L samples, then the short-term magnitude of mth block (ym) is given by

ym = $\sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{L}} |\boldsymbol{xm}[\boldsymbol{i}]|$ where xm

denotes the mth block of input signal x[n].

Significance:

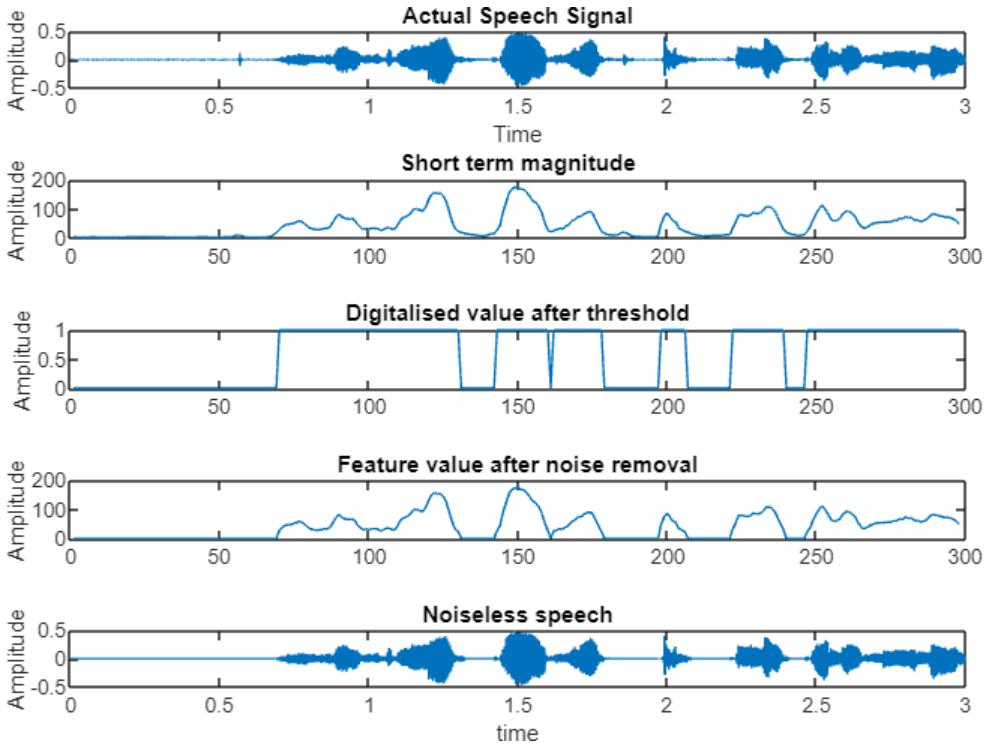
If the absolute values of the samples of a block is high, then the short-term magnitude value of the block is high. The voiced portions have higher amplitudes when compared to the unvoiced portions. Thus, short term magnitude can be used to separate the voiced and the unvoiced portions.

MATLAB CODE:

```
clc;
clear all;
close all;
[y,Fs] = audioread('Male1Vivek.wav');
n = length(y);
for i=1:n
    audio(i) = (y(i,1)+y(i,2))/2;
end
dt = 1/Fs;
tt = dt*(0:n-1);
t = tt(1:3*Fs);
len = length(t);
req = audio(1:len);
blocksize = 30*(10^(-3))*Fs;
blockshift = 10*(10^(-3))*Fs;
subplot(5,1,1);
plot(t,req);
title('Actual Speech Signal');
xlabel('Time');
ylabel('Amplitude');
i=1;
ind = 1;
while i+blocksize-1 <= len
    start = i;
    ending = i+blocksize-1;
    v = sg_stm(req,start,ending);
    val(ind)=v;
    ind = ind+1;
    i = i+blockshift;
end
for i=1:length(val)
    if(val(i)>20)
        new_val(i)=1;
    else
        new_val(i)=0;
    end
end
subplot(5,1,2);
plot(val);
title('Short term magnitude');
ylabel('Amplitude');
subplot(5,1,3);
plot(new_val);
title('Digitalised value after threshold');
ylabel('Amplitude');
no_noise_val = val.*new_val;
subplot(5,1,4);
plot(no_noise_val);
title('Feature value after noise removal');
ylabel('Amplitude');
new_speech = zeros(1,len);
k=1;
i=1;
while i+blocksize-1<=len
    start = i;
    ending = i+blocksize-1;
    if(new_val(k)==1)
        new_speech(start:ending)=req(start:ending);
```

```
    end
    k = k+1;
    i = i+blockshift;
end
subplot(5,1,5);
plot(t,new_speech);
title('Noiseless speech');
xlabel('time');
ylabel('Amplitude');
function[y] = sg_stm(audio,i,j)
sum=0;
for k=i:j
    sum = sum + abs(audio(k));
end
y = sum;
end
```

Output:



2) Short-term energy (ste):

Definition:

Short term energy is equal to the sum of squares of all the sample values present in the input sequence.

Mathematical expression:

Let the given input signal is x[n] consisting of N samples. If the input signal is divided into M blocks each consisting of L samples, then the short term energy of mth block (ym) is given by

$$y_m = \sum_{i=1}^{L} (x_m[i])^2$$

where xm denotes the mth block of input signal x[n].

Significance:

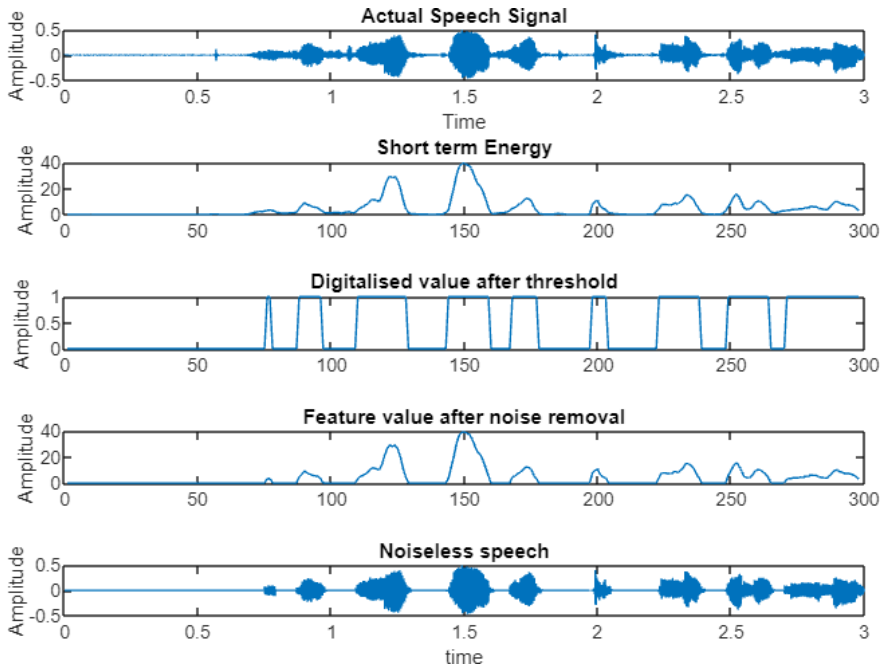
If the amplitudes of the samples in a sequence is high, then the energy of the sequence is also high. The voiced portions have higher magnitudes when compared to the unvoiced portions. Thus, this feature can be used to separate the voiced and the unvoiced portions of a speech signal.

MATLAB CODE:

```
clc;
clear all;
close all;
[y,Fs] = audioread('Male1Vivek.wav');
n = length(y);
for i=1:n
    audio(i) = (y(i,1)+y(i,2))/2;
end
dt = 1/Fs;
tt = dt*(0:n-1);
t = tt(1:3*Fs);
len = length(t);
req = audio(1:len);
blocksize = 30*(10^(-3))*Fs;
blockshift = 10*(10^(-3))*Fs;
subplot(5,1,1);
plot(t,req);
title('Actual Speech Signal');
xlabel('Time');
ylabel('Amplitude');
i=1;
ind = 1;
while i+blocksize-1 <= len
    start = i;
    ending = i+blocksize-1;
    v = sg_ste(req,start,ending);
    val(ind)=v;
    ind = ind+1;
    i = i+blockshift;
end
for i=1:length(val)

    if(val(i)>3)
        new_val(i)=1;
    else
        new_val(i)=0;
    end
end
subplot(5,1,2);
plot(val);
title('Short term Energy');
ylabel('Amplitude');
subplot(5,1,3);
plot(new_val);
title('Digitalised value after threshold');
ylabel('Amplitude');
no_noise_val = val.*new_val;
subplot(5,1,4);
plot(no_noise_val);
title('Feature value after noise removal');
ylabel('Amplitude');
new_speech = zeros(1,len);
k=1;
i=1;
while i+blocksize-1<=len
    start = i;
    ending = i+blocksize-1;
    if(new_val(k)==1)
```

```
        new_speech(start:ending)=req(start:ending);
    end
    k = k+1;
    i = i+blockshift;
end
subplot(5,1,5);
plot(t,new_speech);
title('Noiseless speech');
xlabel('time');
ylabel('Amplitude');
function[y] = sg_ste(audio,i,j)
sum=0;
for k=i:j
    sum = sum + (audio(k)^2);
end
y = sum;
end
```

Output:



3) Sample entropy (sament)

Definition: Entropy provides a measure of the average amount of information conveyed by the output of a random trial.

Mathematical expression: Given a discrete time random variable X, with possible outcomes x1, x2, x3, …xn which occur with probabilties P(x1), P(x2), P(x3), … P(xn) ; the entropy is given by

Entropy (H(X)) = $\sum -\boldsymbol{P}(\boldsymbol{xi})\mathbf{log}\,(\boldsymbol{P}(\boldsymbol{xi}))$

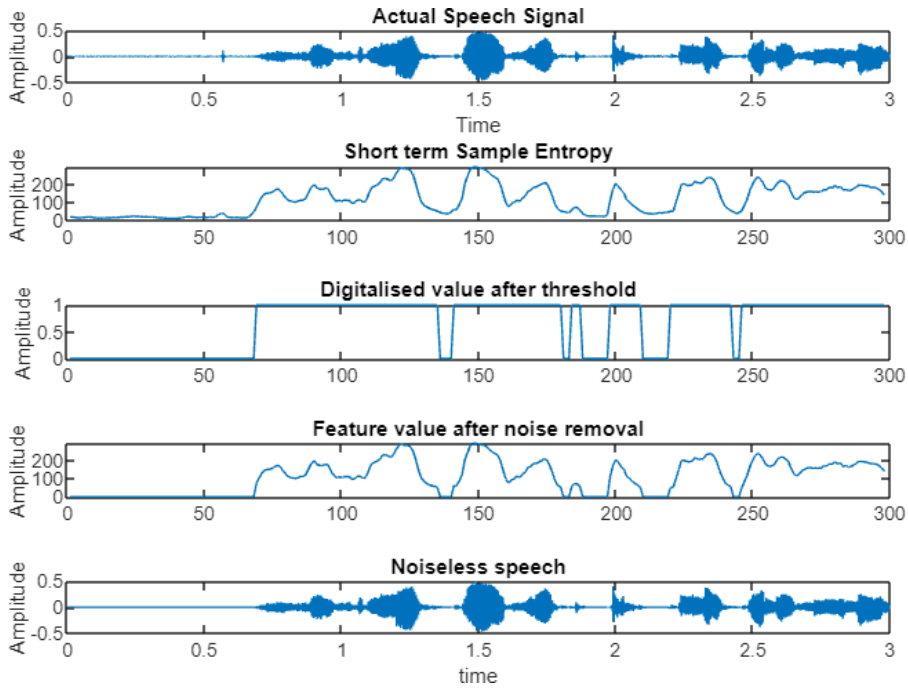
Significance: If the probability of occurrence of a sample is more, then the amount of information it contains will be less. Entropy is the average amount of information present in the signal. So, if entropy is more it means that the information the signal contains is also more. Here, voiced portions have larger entropy since they convey more information.

MATLAB CODE:

```
clc;
clear all;
close all;
[y,Fs] = audioread('Male1Vivek.wav');
n = length(y);
for i=1:n
    audio(i) = (y(i,1)+y(i,2))/2;
end
dt = 1/Fs;
tt = dt*(0:n-1);
t = tt(1:3*Fs);
len = length(t);
req = audio(1:len);
blocksize = 30*(10^(-3))*Fs;
blockshift = 10*(10^(-3))*Fs;
subplot(5,1,1);
plot(t,req);
title('Actual Speech Signal');
xlabel('Time');
ylabel('Amplitude');
i=1;
ind = 1;
while i+blocksize-1 <= len
    start = i;
    ending = i+blocksize-1;
    v = sg_sament(req,start,ending);
    val(ind)=v;
    ind = ind+1;
    i = i+blockshift;
end
```

```
for i=1:length(val)
    if(val(i)>50)
        new_val(i)=1;
    else
        new_val(i)=0;
    end
end
subplot(5,1,2);
plot(val);
title('Short term Sample Entropy');
ylabel('Amplitude');
subplot(5,1,3);
plot(new_val);
title('Digitalised value after threshold');
ylabel('Amplitude');
no_noise_val = val.*new_val;
subplot(5,1,4);
plot(no_noise_val);
title('Feature value after noise removal');
ylabel('Amplitude');
new_speech = zeros(1,len);
k=1;
i=1;
while i+blocksize-1<=len
    start = i;
    ending = i+blocksize-1;
    if(new_val(k)==1)
        new speech(start:ending)=req(start:ending);
    end
    k = k+1;
    i = i+blockshift;
end
subplot(5,1,5);
plot(t,new_speech);
title('Noiseless speech');
xlabel('time');
ylabel('Amplitude');
function[y] = sg_sament(audio,i,j)
m = audio(i:j);
m = abs(m);
sum=0;
for q=1:length(m)
    if(m(q)~=0)
        sum = sum -(m(q)*log(m(q)));
    end
end
y = sum;
end
```

Output:



4) short-term zerocrossing rate (stzcr)

Definition: Short term zero crossing rate would give the number of zero crossings present in a given input sequence. Any point is said to be a zero-crossing point, if the sign of sample value changes at that particular point.

Mathematical expression: For an input signal x[n] with N samples, there is a zero crossing at location i if x[i-1] × x[i+1] <= 0. We will compute the total number of zero crossings in the array by traversing the array.

Significance: Noise portions of the signal has more number of zero crossings when compared to the voiced portions. Thus, short term zero crossing rate would enable us to separate the voiced and unvoiced portions in a speech signal.

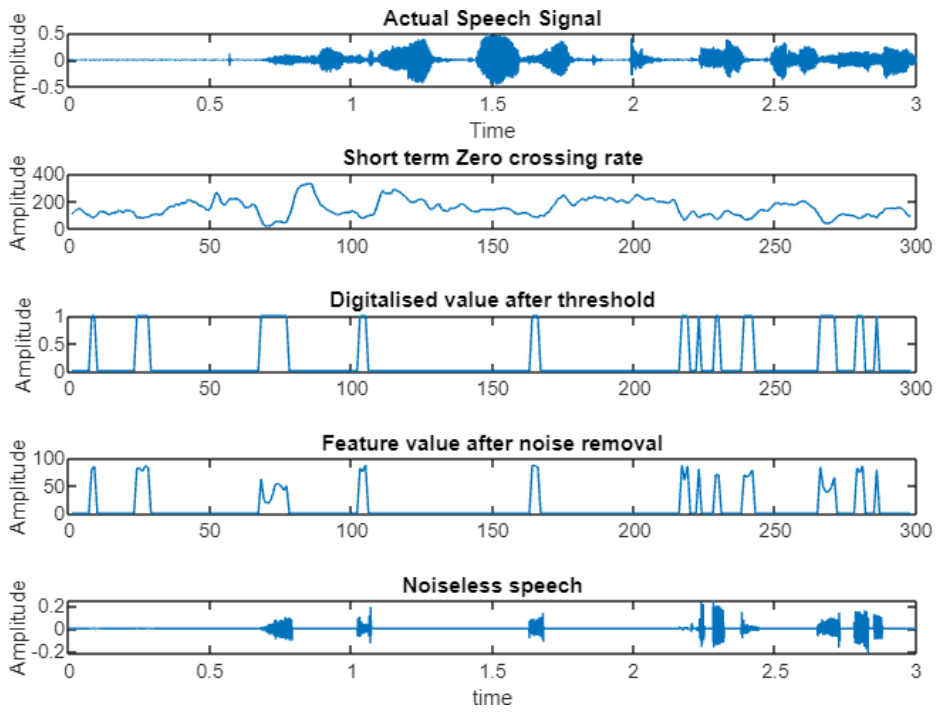
MATLAB CODE:

```
clc;
clear all;
close all;
[y,Fs] = audioread('Male1Vivek.wav');
n = length(y);
for i=1:n
    audio(i) = (y(i,1)+y(i,2))/2;
end
dt = 1/Fs;
tt = dt*(0:n-1);
t = tt(1:3*Fs);
len = length(t);
req = audio(1:len);
blocksize = 30*(10^(-3))*Fs;
blockshift = 10*(10^(-3))*Fs;
subplot(5,1,1);
plot(t,req);
title('Actual Speech Signal');
xlabel('Time');
ylabel('Amplitude');
i=1;
ind = 1;
while i+blocksize-1 <= len
    start = i;
    ending = i+blocksize-1;
    v = sg_stzcr(req,start,ending);
    val(ind)=v;
    ind = ind+1;
    i = i+blockshift;
end
```

```
for i=1:length(val)
    if(val(i)<90)
        new_val(i)=1;
    else
        new_val(i)=0;
    end
end
subplot(5,1,2);
plot(val);
title('Short term Zero crossing rate');
ylabel('Amplitude');
subplot(5,1,3);
plot(new_val);
title('Digitalised value after threshold');
ylabel('Amplitude');
no_noise_val = val.*new_val;
subplot(5,1,4);
plot(no_noise_val);
title('Feature value after noise removal');
ylabel('Amplitude');
new_speech = zeros(1,len);
k=1;
i=1;
```

```
while i+blocksize-1<=len
    start = i;
    ending = i+blocksize-1;
    if(new_val(k)==1)
        new_speech(start:ending)=req(start:ending);
    end
    k = k+1;
    i = i+blockshift;
end
subplot(5,1,5);
plot(t,new_speech);
title('Noiseless speech');
xlabel('time');
ylabel('Amplitude');
function[y] = sg_stzcr(audio,i,j)
count=0;
for k=i+1:j-1
    if((audio(k-1)*audio(k+1)<=0))
        count = count+1;
    end
end
y = count;
end
```

Output:



5) short-term autocorrelation function (stacf) with peak amplitude and its location, decaying rate, first zerocrossing location and first 3 ACF values

Definition: Autocorrelation is defined as the correlation of the signal with a delayed copy of itself. It measures the relationship between a variable's current value and its past values. Autocorrelation is an even function and it has peak value at 0. Here, the peak amplitude refers to the local maximum after the first zero crossing point.

Mathematical expression: Autocorrelation of a sequence x[n] is given by the sequence $\boldsymbol{rxx}$[n] = $\sum \boldsymbol{x}[\boldsymbol{k}]\boldsymbol{x}[\boldsymbol{k}+\boldsymbol{n}]$,where n=0, ±1, ±2, …

Significance: Autocorrelation would tell us how well the present value of the series is related with its past values. So, if we get a high autocorrelation at a specific lag, it may be the case that the signal tends to repeat itself after that amount of lag. Here, we can observe that the first peak autocorrelation value can be used as an appropriate feature to distinguish between voiced and unvoiced portions.

MATLAB CODE:

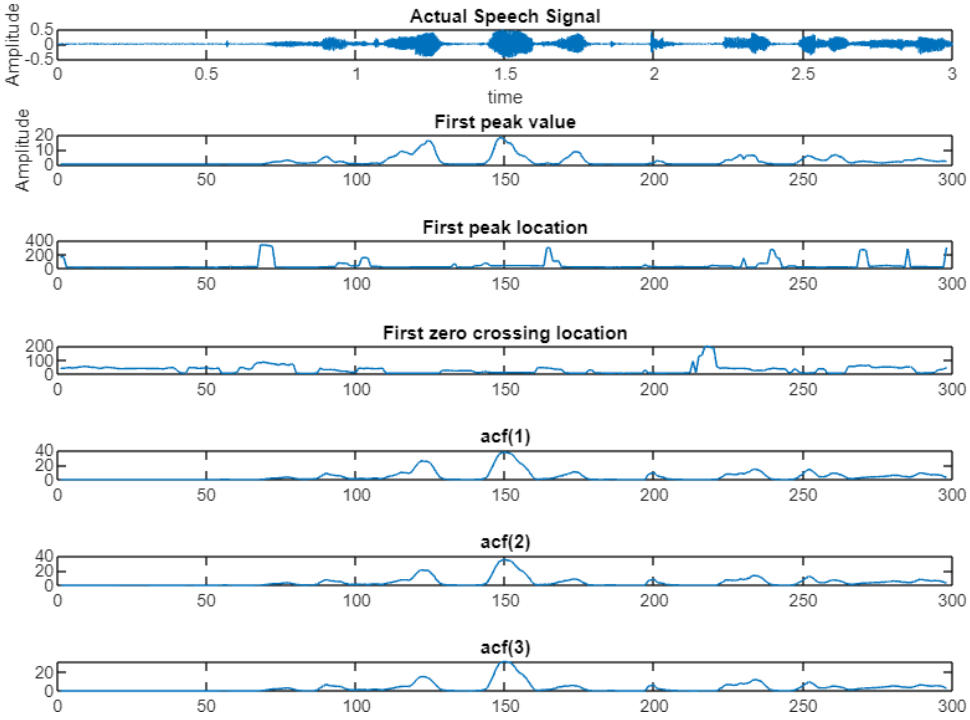
```
clc;
clear all;
close all;
[y,Fs] = audioread('Male1Vivek.wav');
n = length(y);
audio = zeros(1,n);
for i=1:n
    audio(i) = (y(i,1)+y(i,2))/2;
end
dt = 1/Fs;
tt = dt*(0:n-1);
t = tt(1:3*Fs);
len = length(t);
req = audio(1:len);
blocksize = 30*(10^(-3))*Fs;
blockshift = 10*(10^(-3))*Fs;
i=1;
k = 0;
while i+blocksize-1<=len
    k=k+1;
    i= i+blockshift;
end
peak_val = zeros(1,k);
peak_loc = zeros(1,k);
fzcl = zeros(1,k);
acf1 = zeros(1,k);
acf2 = zeros(1,k);
acf3 = zeros(1,k);
ind=1;
i=1;
```

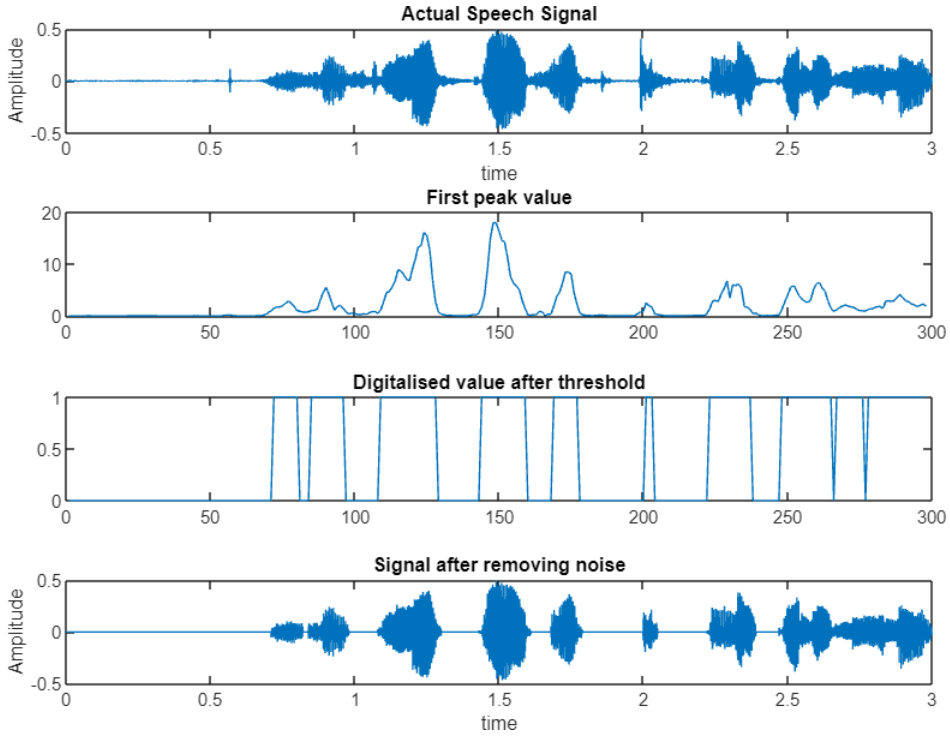
```
while i+blocksize-1 <= len
    start = i;
    ending = i+blocksize-1;
    [k1,k2,k3,k4,k5,k6] = sg_stacf(req,start,ending);
    peak_val(ind)=k1;
    peak_loc(ind)=k2;
    fzcl(ind)=k3;
    acf1(ind)=k4;
    acf2(ind)=k5;
    acf3(ind)=k6;
    ind = ind+1;
    i = i+ blockshift;
end
subplot(7,1,1);
plot(t,req);
title('Actual Speech Signal');
xlabel('time');
ylabel('Amplitude');
subplot(7,1,2);
plot(peak_val);
title('First peak value');
ylabel('Amplitude');
subplot(7,1,3);
plot(peak_loc);
title('First peak location');
subplot(7,1,4);
plot(fzcl);
```

```
title('First zero crossing location');
subplot(7,1,5);
plot(acf1);
title('acf(1)');
subplot(7,1,6);
plot(acf2);
title('acf(2)');
subplot(7,1,7);
plot(acf3);
title('acf(3)');
figure;
subplot(4,1,1);
plot(t,req);
title('Actual Speech Signal');
xlabel('time');
ylabel('Amplitude');
subplot(4,1,2);
plot(peak_val);
title('First peak value');
new_speech = zeros(1,len);
for i=1:length(peak_val)
    if(peak_val(i)>1)
        new_val(i)=1;
    else
        new_val(i)=0;
    end
end
k=1;
i=1;
```

```
while i+blocksize-1<=len
    start = i;
    ending = i+blocksize-1;
    if(new_val(k)==1)
        new_speech(start:ending)=req(start:ending);
    end
    k = k+1;
    i = i+blockshift;
end
subplot(4,1,3);
plot(new_val);
title('Digitalised value after threshold');
subplot(4,1,4);
plot(t,new_speech);
title('Signal after removing noise');
xlabel('time');
ylabel('Amplitude');

function[peak_val,peak_loc,fzcl,acf1,acf2,acf3] = sg_stacf(audio,i,j)
m = audio(i:j);
n = length(m);
z = zeros(1,n);
for i=1:n
    for j=1:n-i
        z(i) = z(i)+ (m(j)*m(j+i));
    end
end
peak_val=0;
peak_loc=0;
fzcl=0;
acf1=0;
acf2=0;
acf3=0;
%First peak amplitude
for i=2:n-1
    if(z(i)>0 && z(i)>z(i-1) && z(i)>z(i+1))
        peak_val = z(i);
        peak_loc = i;
        break;
    end
end

%First zero crossing location
for i=2:n-1
    if(z(i-1)*z(i+1)<=0)
        fzcl = i;
        break;
    end
end

%First three ACF values
acf1 = z(2);
acf2 = z(3);
acf3 = z(4);
end
```

Output:





6) signal statistics (signalsat) including the mean (MEAN), variance (VAR), skewness (SKEW), and kurtosis (KUR) of the signal

MEAN:

Definition: Mean is defined as the average or the expected value of the signal.

Mathematical expression: Let the given input signal is x[n] consisting of N samples. If the input signal is divided into M blocks each consisting of L samples, then the short term mean of mth block (ym) is given by

ym = $\sum \boldsymbol{xm}[\boldsymbol{i}] \,/\, \boldsymbol{L}$

where xm denotes the mth block of input signal x[n].

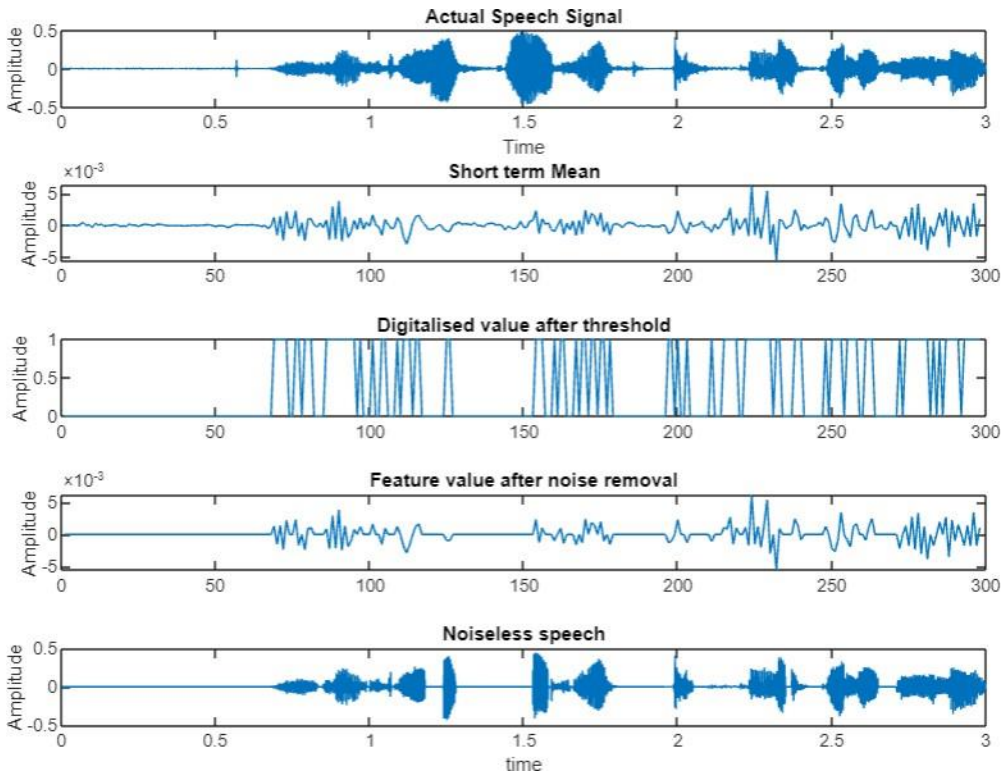
Significance: If the amplitude of the signal is high then the average value would be high. Amplitude of voiced portions is high and the noise portions have lower amplitudes. Therefore, the voiced portions have higher mean and thus it can be used to differentiate between the voiced and unvoiced portions.

MATLAB CODE:

```
clc;
clear all;
close all;
[y,Fs] = audioread('Male1Vivek.wav');
n = length(y);
for i=1:n
    audio(i) = (y(i,1)+y(i,2))/2;
end
dt = 1/Fs;
tt = dt*(0:n-1);
t = tt(1:3*Fs);
len = length(t);
req = audio(1:len);
blocksize = 30*(10^(-3))*Fs;
blockshift = 10*(10^(-3))*Fs;
subplot(5,1,1);
plot(t,req);
title('Actual Speech Signal');
xlabel('Time');
ylabel('Amplitude');
i=1;
ind = 1;
while i+blocksize-1 <= len
    start = i;
    ending = i+blocksize-1;
    v = sg_mean(req,start,ending);
    val(ind)=v;
    ind = ind+1;
    i = i+blockshift;
end
```

```
for i=1:length(val)
    if(abs(val(i))>0.0008)
        new_val(i)=1;
    else
        new_val(i)=0;
    end
end
subplot(5,1,2);
plot(val);
title('Short term Mean');
ylabel('Amplitude');
subplot(5,1,3);
plot(new_val);
title('Digitalised value after threshold');
ylabel('Amplitude');
no_noise_val = val.*new_val;
subplot(5,1,4);
plot(no_noise_val);
title('Feature value after noise removal');
ylabel('Amplitude');
new_speech = zeros(1,len);
k=1;
i=1;
while i+blocksize-1<=len
    start = i;
    ending = i+blocksize-1;
    if(new_val(k)==1)
        new_speech(start:ending)=req(start:ending);
    end

    k = k+1;
    i = i+blockshift;
end
subplot(5,1,5);
plot(t,new_speech);
title('Noiseless speech');
xlabel('time');
ylabel('Amplitude');
function[y] = sg_mean(audio,i,j)
mat = audio(i:j);
n = length(mat);
y = sum(mat)/n;
end
```

Output:



VARIANCE:

Definition: Variance is the second central moment and it describes the spread or dispersion of the signal around the mean.

Mathematical expression: Let the given input signal is x[n] consisting of N samples. If the input signal is divided into M blocks each consisting of L samples and, then the variance of mth block (ym) with mean μ is given by ym = ∑ $(\boldsymbol{xm}[\boldsymbol{i}]-\mu)^{\wedge}\boldsymbol{2}/L$.

Where, xm denotes the mth block of input signal x[n].

Significance: If the variance of the signal is more, then it means the signal is more deviated from its mean. The voiced portions vary more from their mean when compared to the unvoiced portions. The magnitude of noise signal is very less and its deviation from the mean is also less. Thus, the variance of noised portions is less. This property can be used to separate the voiced and unvoiced portions of the speech signal.

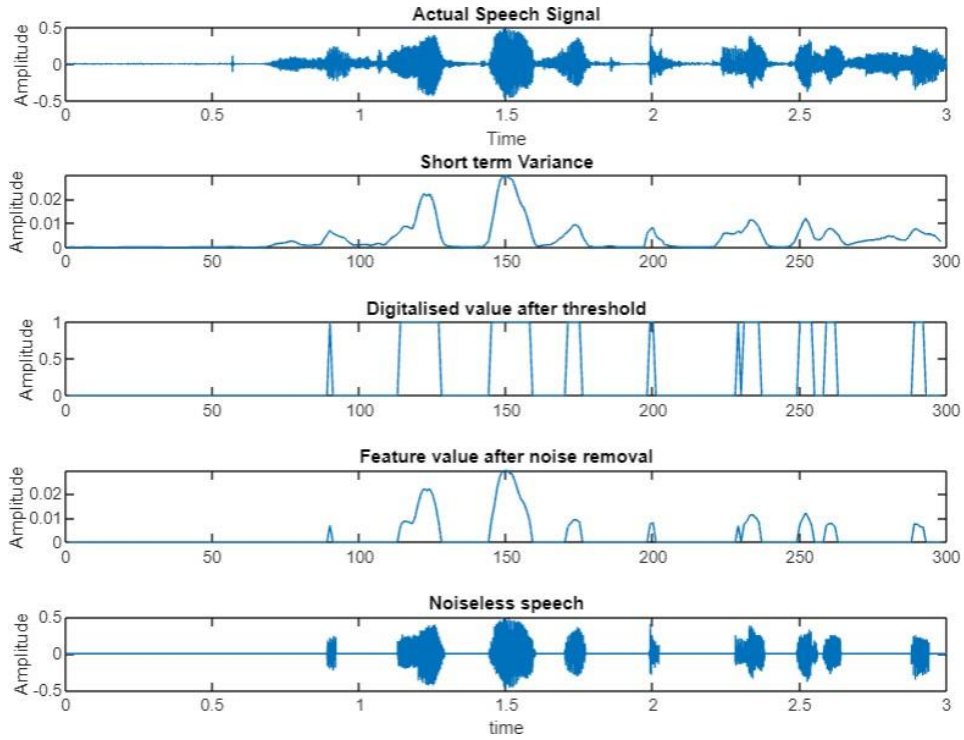
MATLAB CODE:

```
clc;
clear all;
close all;
[y,Fs] = audioread('Male1Vivek.wav');
n = length(y);
for i=1:n
    audio(i) = (y(i,1)+y(i,2))/2;
end
dt = 1/Fs;
tt = dt*(0:n-1);
t = tt(1:3*Fs);
len = length(t);
req = audio(1:len);
blocksize = 30*(10^(-3))*Fs;
blockshift = 10*(10^(-3))*Fs;
subplot(5,1,1);
plot(t,req);
title('Actual Speech Signal');
xlabel('Time');
ylabel('Amplitude');
i=1;
ind = 1;
while i+blocksize-1 <= len
    start = i;
    ending = i+blocksize-1;
    v = sg_var(req,start,ending);
    val(ind)=v;
    ind = ind+1;
    i = i+blockshift;
end
```

```
for i=1:length(val)
    if(val(i)>0.006)
        new_val(i)=1;
    else
        new_val(i)=0;
    end
end
subplot(5,1,2);
plot(val);
title('Short term Variance');
ylabel('Amplitude');
subplot(5,1,3);
plot(new_val);
title('Digitalised value after threshold');
ylabel('Amplitude');
no_noise_val = val.*new_val;
subplot(5,1,4);
plot(no_noise_val);
title('Feature value after noise removal');
ylabel('Amplitude');
new_speech = zeros(1,len);
k=1;
i=1;
while i+blocksize-1<=len
    start = i;
    ending = i+blocksize-1;
    if(new_val(k)==1)
        new_speech(start:ending)=req(start:ending);
```

```
    end
    k = k+1;
    i = i+blockshift;
end
subplot(5,1,5);
plot(t,new_speech);
title('Noiseless speech');
xlabel('time');
ylabel('Amplitude');
function[v] = sg_var(audio,i,j)
mat = audio(i:j);
m=0;
v=0;
n = length(mat);
m = sum(mat)/n;
for l=1:n
    v= v +((mat(l)-m)^2);
end
v=v/n;
end
```

Output:



SKEWNESS:

Definition: It is defined as the third central moment and describes about the measure of asymmetry of the distribution about its mean.

Mathematical expression: Let the given input signal is x[n] consisting of N samples. If the input signal is divided into M blocks each consisting of L samples, then the short term mean of mth block (ym) with mean μ and standard deviation σ is given by

ym = (∑ ($\boldsymbol{xm}[\boldsymbol{i}]$−μ) ^**3**)/(σ*L).

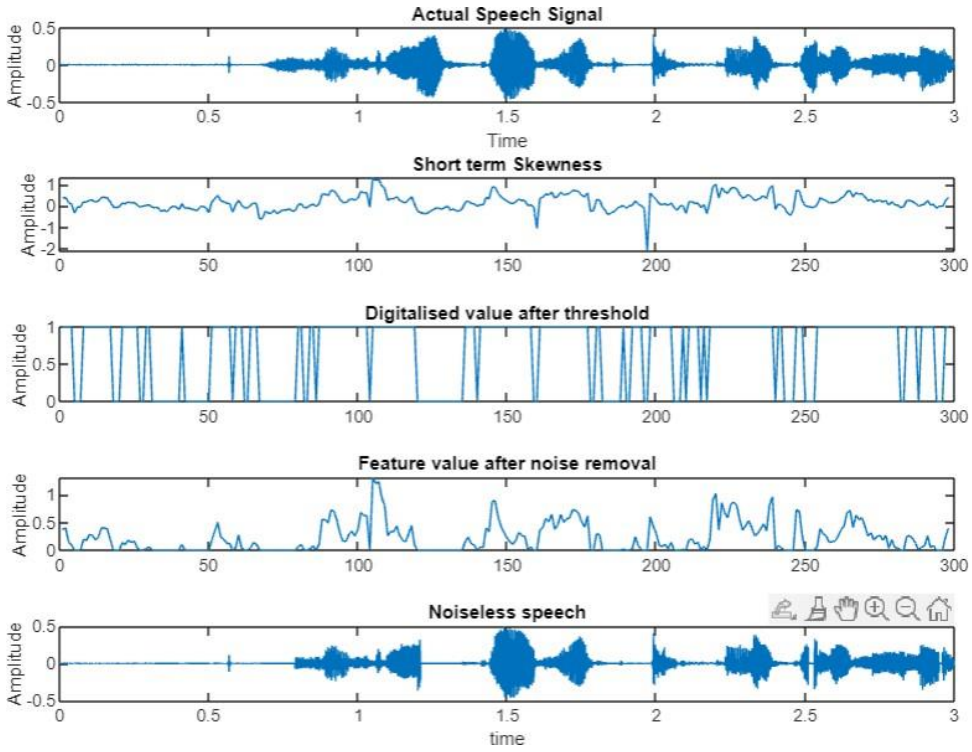
where xm denotes the mth block of input signal x[n].

MATLAB CODE:

```
clc;
clear all;
close all;
[y,Fs] = audioread('Male1Vivek.wav');
n = length(y);
for i=1:n
    audio(i) = (y(i,1)+y(i,2))/2;
end
dt = 1/Fs;
tt = dt*(0:n-1);
t = tt(1:3*Fs);
len = length(t);
req = audio(1:len);
blocksize = 30*(10^(-3))*Fs;
blockshift = 10*(10^(-3))*Fs;
subplot(5,1,1);
plot(t,req);
title('Actual Speech Signal');
xlabel('Time');
ylabel('Amplitude');
i=1;
ind = 1;
while i+blocksize-1 <= len
    start = i;
    ending = i+blocksize-1;
    v = sg_skewness(req,start,ending);
    val(ind)=v;
    ind = ind+1;
    i = i+blockshift;
end
```

```
for i=1:length(val)
    if(val(i)>0.005)
        new_val(i)=1;
    else
        new_val(i)=0;
    end
end
subplot(5,1,2);
plot(val);
title('Short term Skewness');
ylabel('Amplitude');
subplot(5,1,3);
plot(new_val);
title('Digitalised value after threshold');
ylabel('Amplitude');
no_noise_val = val.*new_val;
subplot(5,1,4);
plot(no_noise_val);
title('Feature value after noise removal');
ylabel('Amplitude');
new_speech = zeros(1,len);
k=1;
i=1;
while i+blocksize-1<=len
    start = i;
    ending = i+blocksize-1;
    if(new_val(k)==1)
        new_speech(start:ending)=req(start:ending);
    end

    k = k+1;
    i = i+blockshift;
end
subplot(5,1,5);
plot(t,new_speech);
title('Noiseless speech');
xlabel('time');
ylabel('Amplitude');
function[s] = sg_skewness(audio,i,j)
mat = audio(i:j);
m=0;
v=0;
s=0;
n = length(mat);
m = sum(mat)/n;
for l=1:n
    v= v +((mat(l)-m)^2);
    s= s +((mat(l)-m)^3);
end
v=v/n;
s=s/(n*power(v,1.5));
end
```

Output:



KURTOSIS:

Definition: It is defined as the fourth central moment and describes about the degree of peakedness of the signal

Mathematical expression: Let the given input signal is x[n] consisting of N samples. If the input signal is divided into M blocks each consisting of L samples, then the short term mean of mth block (ym) with mean μ and standard deviation σ is given by

ym = (∑ (($\boldsymbol{xm}[\boldsymbol{i}]$−μ)/ σ)^4)/L

Where, xm denotes the mth block of input signal x[n].

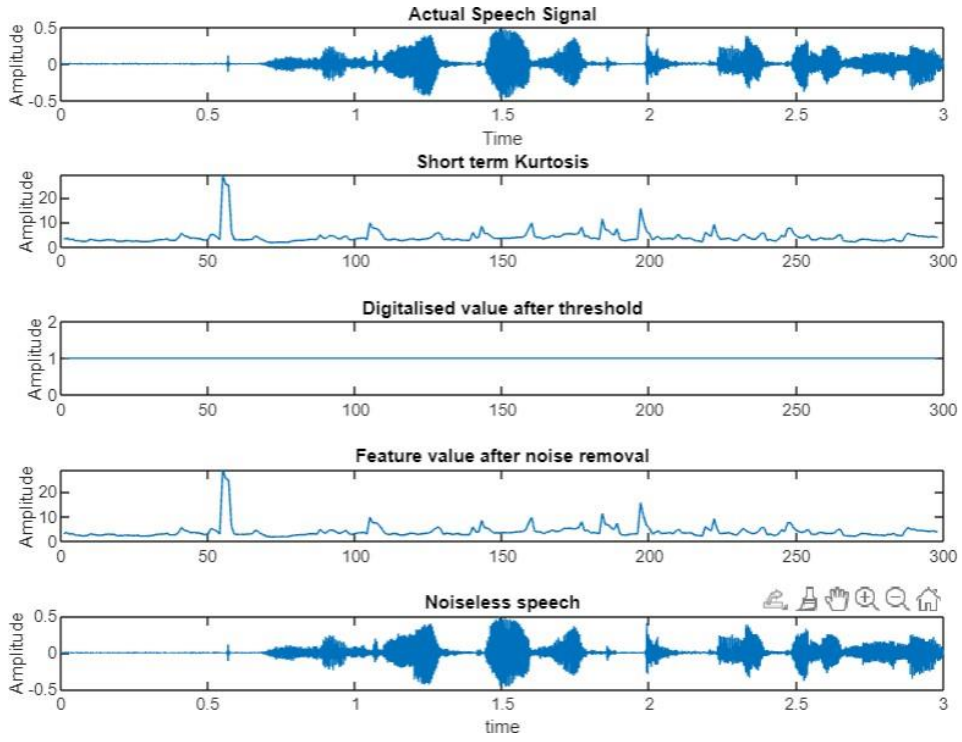
MATLAB CODE:

```
clc;
clear all;
close all;
[y,Fs] = audioread('Male1Vivek.wav');
n = length(y);
for i=1:n
    audio(i) = (y(i,1)+y(i,2))/2;
end
dt = 1/Fs;
tt = dt*(0:n-1);
t = tt(1:3*Fs);
len = length(t);
req = audio(1:len);
blocksize = 30*(10^(-3))*Fs;
blockshift = 10*(10^(-3))*Fs;
subplot(5,1,1);
plot(t,req);
title('Actual Speech Signal');
xlabel('Time');
ylabel('Amplitude');
i=1;
ind = 1;
while i+blocksize-1 <= len
    start = i;
    ending = i+blocksize-1;
    v = sg_kurtosis(req,start,ending);
    val(ind)=v;
    ind = ind+1;
    i = i+blockshift;
end
```

```
for i=1:length(val)
    if(val(i)>0.005)
        new_val(i)=1;
    else
        new_val(i)=0;
    end
end
subplot(5,1,2);
plot(val);
title('Short term Kurtosis');
ylabel('Amplitude');
subplot(5,1,3);
plot(new_val);
title('Digitalised value after threshold');
ylabel('Amplitude');
no_noise_val = val.*new_val;
subplot(5,1,4);
plot(no_noise_val);
title('Feature value after noise removal');
ylabel('Amplitude');
new_speech = zeros(1,len);
k=1;
i=1;
while i+blocksize-1<=len
    start = i;
    ending = i+blocksize-1;
    if(new_val(k)==1)
        new_speech(start:ending)=req(start:ending);
    end
```

```
    end
    k = k+1;
    i = i+blockshift;
end
subplot(5,1,5);
plot(t,new_speech);
title('Noiseless speech');
xlabel('time');
ylabel('Amplitude');
function[k] = sg_kurtosis(audio,i,j)
mat = audio(i:j);
m=0;
v=0;
k=0;
n = length(mat);
m = sum(mat)/n;
for l=1:n
    v= v +((mat(l)-m)^2);
    k= k +((mat(l)-m)^4);
end
v=v/n;
k=k/(n*power(v,2));
end
```

Output:



7) Normalized cross-correlation coefficient (nccc)

Definition: Cross correlation of two signals is the measure of similarity of the two signals as a function of displacement of one relative to the other.

Normalized cross correlation of two signals is also the cross correlation of the signals, but it can be performed between the two signals with different value ranges.

Mathematical expression : Cross correlation of sequence x1[n] and x2[n] with standard deviations σ1 and σ2 is given by the sequence $\boldsymbol{rx1x2}$ [n] = ∑ $\boldsymbol{x1}$ [$\boldsymbol{k}$]$\boldsymbol{x2}$[$\boldsymbol{k}$ + $\boldsymbol{n}$] .where, n=0, ±1, ±2, … Normalized cross correlation is given by the sequence $\rho x1x2$ [n] = $\boldsymbol{rx1\ x2}$ [$\mathbf{n}$] $\sigma\mathbf{1}\ \sigma\mathbf{2}$

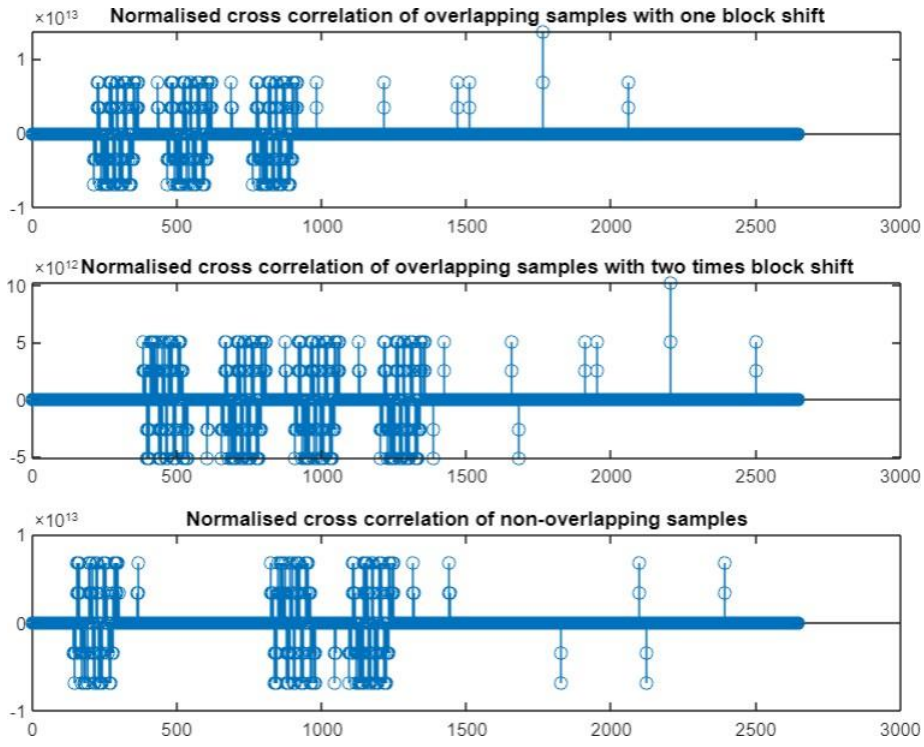
Significance : If the cross correlation of two signals is high, it indicates that there is similarity between the two signals. This is a most important property used to determine the signal that is transmitted using the received signal.

MATLAB CODE:

```
clc;
clear all;
close all;
[y,Fs] = audioread('Female3Likhitha.wav');
n = length(y);
audio = zeros(1,n);
for i=1:n
    audio(i) = (y(i,1)+y(i,2))/2;
end
dt = 1/Fs;
tt = dt*(0:n-1);
t = tt(1:3*Fs);
len = length(t);
req = audio(1:len);
blocksize = 30*(10^(-3))*Fs;
blockshift = 10*(10^(-3))*Fs;
i=1;
j=i+blocksize;
p1 = i+blockshift;
q1 = p1+blocksize;
y1 = sg_nccc(req,i,j,p1,q1);
subplot(3,1,1);
stem(y1);
title('Normalised cross correlation of overlapping samples with one block shift');
p2 = i+(2*blockshift);
q2 = p2+blocksize;
y2 = sg_nccc(req,i,j,p2,q2);
subplot(3,1,2);
stem(y2);
title('Normalised cross correlation of overlapping samples with two times block shift');
p3 = i+(3*blockshift);
```

```
q3 = p3+blocksize;
y3 = sg_nccc(req,i,j,p3,q3);
subplot(3,1,3);
stem(y3);
title('Normalised cross correlation of non-overlapping samples');
function[y] = sg_nccc(audio,i,j,p,q)
mat1 = audio(i:j);
mat2 = audio(p:q);
val = xcorr(mat1,mat2);
v1 = sg_var(mat1,1,length(mat1));
v2 = sg_var(mat2,1,length(mat2));
std1 = sqrt(v1);
std2 = sqrt(v2);
y = val/(v1*v2);
end
function[v] = sg_var(audio,i,j)
mat = audio(i:j);
m=0;
v=0;
n = length(mat);
m = sum(mat)/n;
for l=1:n
    v= v +((mat(l)-m)^2);
end
v=v/n;
end
```

Output:



8) number of turning points (ntp)

Definition: Any point is said to be a turning point if its value is either greater than or less than the adjacent values. Number of turning points returns the count of the turning points in an input sequence.

Mathematical expression: For a given sequence x[n],with N samples, we say there is a turning point at location i if (x[i]>x[i+1] and x[i]<x[i-1]) or if (x[i]<x[i+1] and x[i]<x[i-1]) Where 1< i <N.

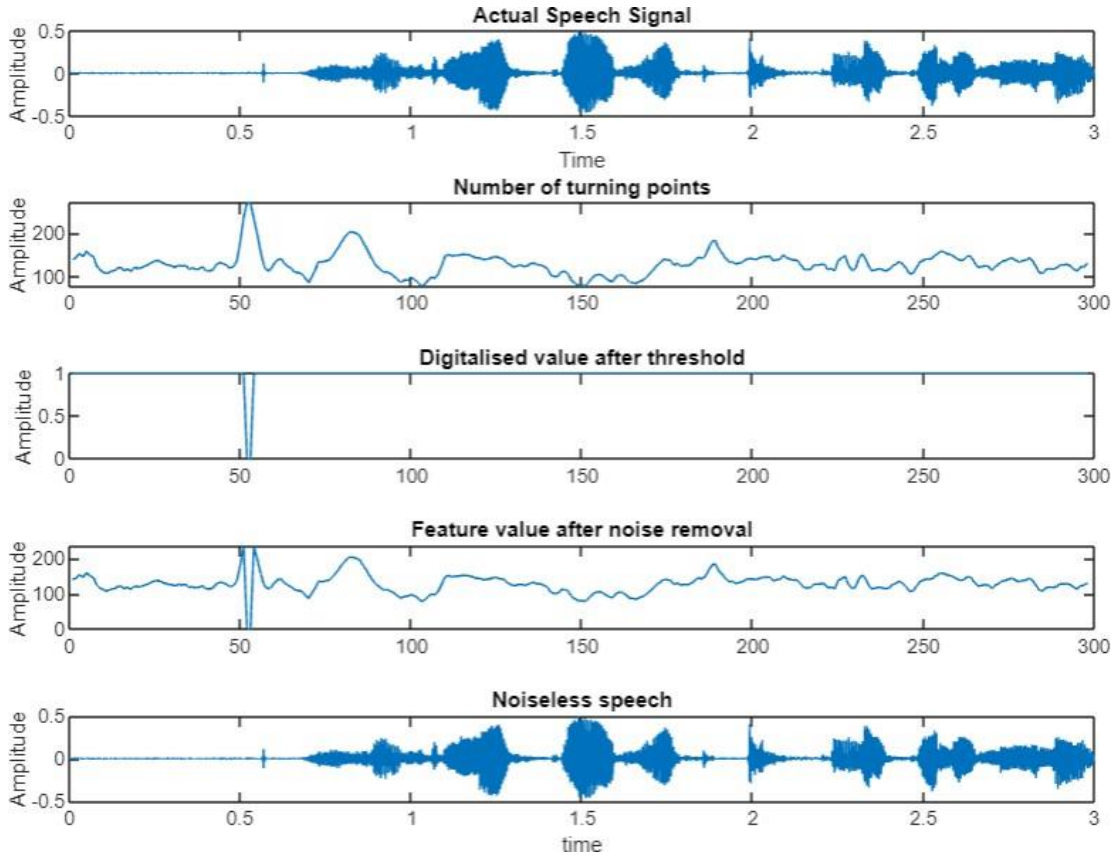
Significance: Number of turning points in a signal gives the measure of randomness of the signal. If the number of turning points are more, then it is indicative that the signal is more random. Speech is also a random in nature. Thus, the number of turning points in voiced and unvoiced portions wouldn't vary much.

MATLAB CODE:

```
clc;
clear all;
close all;
[y,Fs] = audioread('Male1Vivek.wav');
n = length(y);
for i=1:n
    audio(i) = (y(i,1)+y(i,2))/2;
end
dt = 1/Fs;
tt = dt*(0:n-1);
t = tt(1:3*Fs);
len = length(t);
req = audio(1:len);
blocksize = 30*(10^(-3))*Fs;
blockshift = 10*(10^(-3))*Fs;
subplot(5,1,1);
plot(t,req);
title('Actual Speech Signal');
xlabel('Time');
ylabel('Amplitude');
i=1;
ind = 1;
while i+blocksize-1 <= len
    start = i;
    ending = i+blocksize-1;
    v = sg_ntp(req,start,ending);
    val(ind)=v;
    ind = ind+1;
    i = i+blockshift;
end
```

```
for i=1:length(val)
    if(val(i)<260)
        new_val(i)=1;
    else
        new_val(i)=0;
    end
end
subplot(5,1,2);
plot(val);
title('Number of turning points');
ylabel('Amplitude');
subplot(5,1,3);
plot(new_val);
title('Digitalised value after threshold');
ylabel('Amplitude');
no_noise_val = val.*new_val;
subplot(5,1,4);
plot(no_noise_val);
title('Feature value after noise removal');
ylabel('Amplitude');
new_speech = zeros(1,len);
k=1;
i=1;
while i+blocksize-1<=len
    start = i;
    ending = i+blocksize-1;
    if(new_val(k)==1)
        new_speech(start:ending)=req(start:ending);
    end

    k = k+1;
    i = i+blockshift;
end
subplot(5,1,5);
plot(t,new_speech);
title('Noiseless speech');
xlabel('time');
ylabel('Amplitude');
function[y] = sg_ntp(audio,i,j)
m = audio(i:j);
count=0;
len = length(m);
for k=2:len-1
    if((m(k)>m(k-1) && m(k)>m(k+1))||(m(k)<m(k-1) && m(k)<m(k+1)))
        count=count+1;
    end
end
y = count;
end
```

Output:


Actual Speech Signal
Amplitude
Time
Number of turning points
Amplitude
Digitalised value after threshold
Amplitude
Feature value after noise removal
Amplitude
Noiseless speech
Amplitude
time