DETECTION OF DTMF KEY BY GOERTZEL ALGORITHM AND ACCURACY INSPECTION

LAB PROJECT

EECE - 312

DIGITAL SIGNAL PROCESSING LAB

GROUP - 04

Group Members:

- Lt. A M Saiman – 201816154
- Lt. Saleh Ahmed – 201816155
- Lt. A M Tahmidur Rahman – 201816158
- Fardin Shahab – 202016008
- Syed Nazmus Sakib – 202016051
- Galib-ul-Alim – 202016057

Submitted to:

Lec. Dibaloke Chanda

Lec Jarin Tasnim

Abstract:

Dual-Tone Multi-frequency (DTMF) is the Standardized term in the telecom industry, where if anyone press a key two tones are generated simultaneously. The DTMF tone detection is an important block in several embedded applications, which works for low-power devices. In general, DTMF is detected using filters banks b telephone companies. But in our project, we have generated and identified DTMF with the aid of The Goertzel algorithm. It is a somewhat well-known algorithm in tone recognition systems. This effective realization of a partial Fourier transform is able to evaluate selected spectral bin by a very simple algorithm. All the procedures and results of the project are analyzed using the MATLAB software.

Objective:

- DTMF generation and detection through Goertzel Algorithm
- Observe detection accuracy over various stages of noise

Introduction:

Dual-tone multi-frequency signaling (**DTMF**) is a telecommunication signaling system using the voice-frequency band over telephone lines between telephone equipment and other communications devices and switching centers. DTMF was first developed in the Bell System in the United States, and became known under the trademark **Touch-Tone** for use in push-button telephones supplied to telephone customers, starting in 1963. DTMF is standardized as ITU-T Recommendation Q.23. It is also known in the UK as *MF4*.

The Touch-Tone system using a telephone keypad gradually replaced the use of rotary dial and has become the industry standard for landline and mobile service. Other multi frequency systems are used for internal signaling within the telephone network

Prior to the development of DTMF, telephone numbers were dialed by users with a loop-disconnect (LD) signaling, more commonly known as pulse dialing (dial pulse, DP) in the United States. It functions by interrupting the current in the local loop between the telephone exchange and the calling party 's telephone at a precise rate with a switch in the telephone that is operated by the rotary dial as it spins back to its rest position after having been rotated to each desired number. The exchange equipment responds to the dial pulses either directly by operating relays or by storing the number in a digit register that records the dialed number. The physical distance for which this type of dialing was possible was restricted by electrical distortions and was possible only on direct metallic links between end points of a line. Placing calls over longer distances required either operator assistance or provision of special subscriber trunk dialing equipment. Operators used an earlier type of multi-frequency signaling.

Multi-frequency signaling (**MF**) is a group of signaling methods that use a mixture of two pure tone (pure sine wave) sounds. Various MF signaling protocols were devised by the Bell System and CCITT. The earliest of these were for in-band signaling between switching centers, where long distance telephone operators used a 16-digit keypad to input the next portion of the destination telephone number in order to contact the next downstream long-distance telephone operator. This semi-automated signaling and switching proved successful in both speed and cost effectiveness. Based on this prior success with using MF by specialists to establish long-distance telephone calls, dual-tone multi-frequency signaling was developed for end-user signaling without the assistance of operators.

The DTMF system uses a set of eight audio frequencies transmitted in pairs to represent 16 signals, represented by the ten digits, the letters A to D, and the symbols # and *. As the signals are audible tones in the voice frequency range, they can be transmitted through electrical repeaters and amplifiers, and over radio and microwave links, thus eliminating the need for intermediate operators on long-distance circuits.

AT&T described the product as "a method for pushbutton signaling from customer stations using the voice transmission path." In order to prevent consumer telephones from interfering with the MF-based routing and switching between telephone switching centers, DTMF frequencies differ from all of the pre-existing MF signaling protocols between switching centers: MF/R1, R2, CCS4, CCS5, and others that were later replaced by SS7 digital signaling. DTMF was known throughout the Bell System by the trademark *Touch-Tone*. The term was first used by AT&T in commerce on July 5, 1960, and was introduced to the public on November 18, 1963, when the first push-button telephone was made available to the public. As a parent company of Bell Systems, AT&T held the trademark from September 4, 1962, to March 13, 1984. It is standardized by ITU-T Recommendation Q.23. In the UK, it is also known as MF4.

Other vendors of compatible telephone equipment called the Touch-Tone feature *tone dialing* or *DTMF*. Automatic Electric (GTE) referred to it as "Touch-calling" in their marketing. Other trade names such as *Digitone* were used by the Northern Electrical Company in Canada.

As a method of in-band sampling, DTMF signals were also used by cable television broadcasters as cue tones to indicate the start and stop times of local commercial insertion points during station breaks for the benefit of cable companies. Until out-of-band signaling equipment was developed in the 1990s, fast, unacknowledged DTMF tone sequences could be heard during the commercial breaks of cable channels in the United States and elsewhere. Previously, terrestrial television stations used DTMF tones to control remote transmitters. In IP telephony, DTMF signals can also be delivered as either in-band or out-of-band tones, or even as a part of signaling protocols, as long as both endpoints agree on a common approach to adopt.

DTMF was originally decoded by tuned filter banks. By the end of the 20th century, digital signal processing became the predominant technology for decoding. DTMF decoding algorithms typically use the Goertzel Algorithm. As DTMF signaling is often transmitted in-band with voice or other audio signals present simultaneously, the DTMF signal definition includes strict limits for timing (minimum duration and interdigit spacing), frequency deviations, harmonics, and amplitude relation of the two components with respect to each other (*twist*)

The **Goertzel algorithm** is a technique in Digital Signal Processing (DSP) for efficient evaluation of the individual terms of the Discrete Fourier Transform (DFT). It is useful in certain practical applications, such as recognition of Dual Tone Mukti-Frequency (DTMF) signaling tones produced by the push buttons of the keypad of a traditional analog telephone. The algorithm was first described by Gerald Goertzel in 1958.

Like the DFT, the Goertzel algorithm analyses one selectable frequency component from a dicrete signaling. Unlike direct DFT calculations, the Goertzel algorithm applies a single real-valued coefficient at each iteration, using real-valued arithmetic for real-valued input sequences. For covering a full spectrum, the Goertzel algorithm has a higher order of complexity than Fast Fourier Transform (FFT) algorithms, but for computing a small number of selected frequency components, it is more numerically

efficient. The simple structure of the Goertzel algorithm makes it well suited to small processors and embedded applications.

The Goertzel algorithm can also be used "in reverse" as a sinusoid synthesis function, which requires only 1 multiplication and 1 subtraction per generated sample.
The main calculation in the Goertzel algorithm has the form of a digital filter, and for this reason the algorithm is often called a *Goertzel filter*. The filter operates on an input sequence **x[n]** in a cascade of two stages with a parameter w_0, giving the frequency to be analyzed, normalized to radians per sample.

The first stage calculates an intermediate sequence, **s[n]:**

$s[n] = x[n] + 2\cos(w_0) * s[n-1] - s[n-2]$ ..... (1)

The second stage applies the following filter to **s[n],** producing output sequence **y[n]:**

$y[n] = s[n] - e^{-jw_0} * s[n-1]$ ..... (2)

The first filter stage can be observed to be a second-order IIR filter with a direct form structure. This particular structure has the property that its internal state variables equal the past output values from that stage. Input values **x[n]** for **< 0** are presumed all equal to 0. To establish the initial filter state so that evaluation can begin at sample **x[0],** the filter states are assigned initial values **s[-2] = s[-1] = 0.** To avoid aliasing hazards, frequency w_0 is often restricted to the range 0 to π (Nyquist–Shannon sampling theorem); using a value outside this range is not meaningless, but is equivalent to using an aliased frequency inside this range, since the exponential function is periodic with a period of 2π in w_0.

The second-stage filter can be observed to be a FIR filter, since its calculations do not use any of its past outputs.

Z-transform methods can be applied to study the properties of the filter cascade. The Z transform of the first filter stage given in equation (1) is

$$\frac{S(z)}{X(z)} = \frac{1}{1 - 2\cos(w_0)\, z^{-1} + z^{-2}}$$

$$= \frac{1}{(1 - e^{+jw_0}\, z^{-1})(1 - e^{-jw_0}\, z^{-1})} \quad \text{..... (3)}$$

The Z transform of the second filter stage given in equation (2) is:

$$\frac{Y(z)}{X(z)} = 1 - e^{-jw_0}\, z^{-1} \quad \text{..... (4)}$$

The combined transfer function of the cascade of the two filter stages is then:

$$\frac{S(z)}{X(z)}\frac{Y(z)}{S(z)} = \frac{Y(z)}{X(z)} = \frac{1 - e^{-jw_0}\, z^{-1}}{(1 - e^{+jw_0}\, z^{-1})(1 - e^{-jw_0}\, z^{-1})}$$

$$= \frac{1}{1 - e^{+jw_0}\, z^{-1}} \quad \text{..... (5)}$$

This can be transformed back to an equivalent time-domain sequence, and the terms unrolled back to the first input term at index **n = 0:**

$y[n] = x[n] + e^{+jw_0}\, y[n-1]$

$= \sum_{k=-\infty}^{n} x[k]\, e^{+jw_0(n-k)}$

$= e^{jw_0 n} \sum_{k=0}^{n} x[k]\, e^{-jw_0 k}$

In DSP, for aperiodic signal we use DTFT to analyze the given signal in frequency domain. More specifically DTFT gives us the flexibility to convert the domain and look from a different perspective. However, DTFT takes all the values from negative infinity to positive infinity to transform which consumes huge time and memory. That is why DFT was invented to make the transformation more time efficient. DFT almost works similar as DTFT, the only difference here is it take a finite number of samples that makes the calculation far easier. The transform length always needs to be chosen with respect to the desired accuracy of the frequency resolution. The computational complexity of the DFT increases quadratically with the number of samples/frequencies, and thus in practice we use the Fast Fourier Transform algorithm (FFT), whose computational complexity is linear-logarithmic. When the task is to identify the modulus and/or phase of a single or of just a few of the frequency components, even FFT cannot offer significant advantage, because it always computes all the frequency components, most of which are discarded.

In such situations, methods specialized in computing a subset of output frequencies can be exploited with great benefit. Goertzel's algorithm is a method that calculates the DFT by converting it into a digital filtering problem. Besides it deals with single frequencies separately. Unlike DFT, Goertzel does not require all the samples in a specific range. It can be customized in a simple way. In short, Goertzel algorithm is a better version of DFT which is even faster than FFT.

Working Procedure:

- **DTMF generation:**
  We have generated by using the international standard frequency chart for DTMF. The chat is given below:

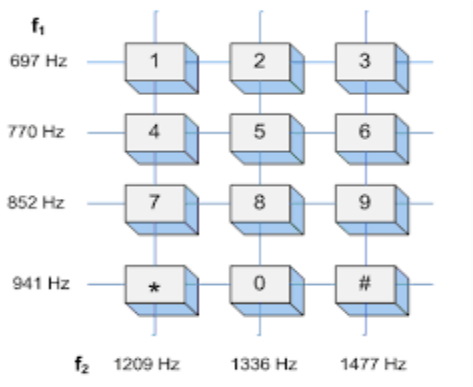

Fig 1: DTMF frequency chart

According to the chart we can see that there are 10 numbers from 0 to 9; and 2 unique symbols "*" and "#". Each of these has a unique set of low frequency and high frequency. So, to generate a DTMF signal at first, we have assigned all of the keys as a variable and manually set their high and low frequencies.

```
% Function for Key 1
function pushbutton1_Callback(hObject, eventdata, handles)
flow=697; fhigh=1209; % Standard frequencies for key 1
textstring= get(handles.detbox,'string');
textstring= strcat(textstring,'1');
set(handles.detbox,'string',textstring) % Displays 1 in the Detection Box
update % Reads the file update.m

% Function for Key 2
function pushbutton2_Callback(hObject, eventdata, handles)
flow=697; fhigh=1336; % Standard frequencies for key 2
textstring= get(handles.detbox,'string');
textstring= strcat(textstring,'2');
set(handles.detbox,'string',textstring) % Displays 2 in the Detection Box
update % Reads the file update.m

% Function for Key 3
function pushbutton3_Callback(hObject, eventdata, handles)
flow=697; fhigh=1477; % Standard frequencies for key 3
textstring= get(handles.detbox,'string');
textstring= strcat(textstring,'3');
set(handles.detbox,'string',textstring) % Displays 3 in the Detection Box
update % Reads the file update.m
```

Fig 2: Key frequency assign

Here we have declared a function for every key. Then we have declared their high and low frequencies. Then we have put the key in the box and when we press the key the detection box will detect what key we have pressed. Similarly we have generated all the keys and made a GUI.

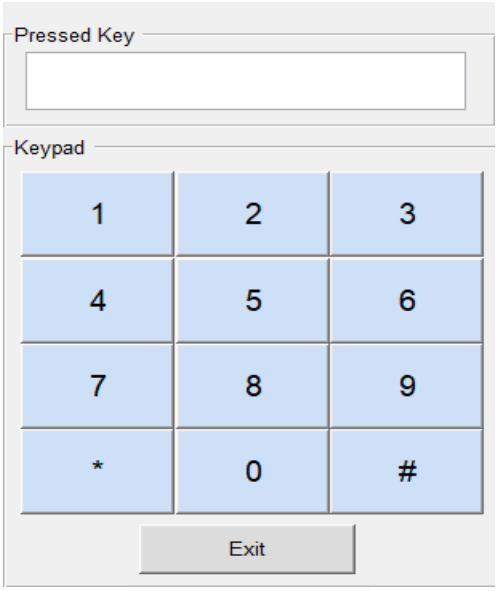

Fig 3: GUI keypad

When a button is pressed, the box named ‘pressed key’ shows the key we have pressed. Then a time domain graph is created by using the corresponding high and low frequency of the pressed key. Below the time domain plot, we have added another window in the GUI that shows the frequency domain plot of the corresponding key by using “periodogram ()” function. And thus, a DTMF signal has been generated.

Fig 4: Pressed key

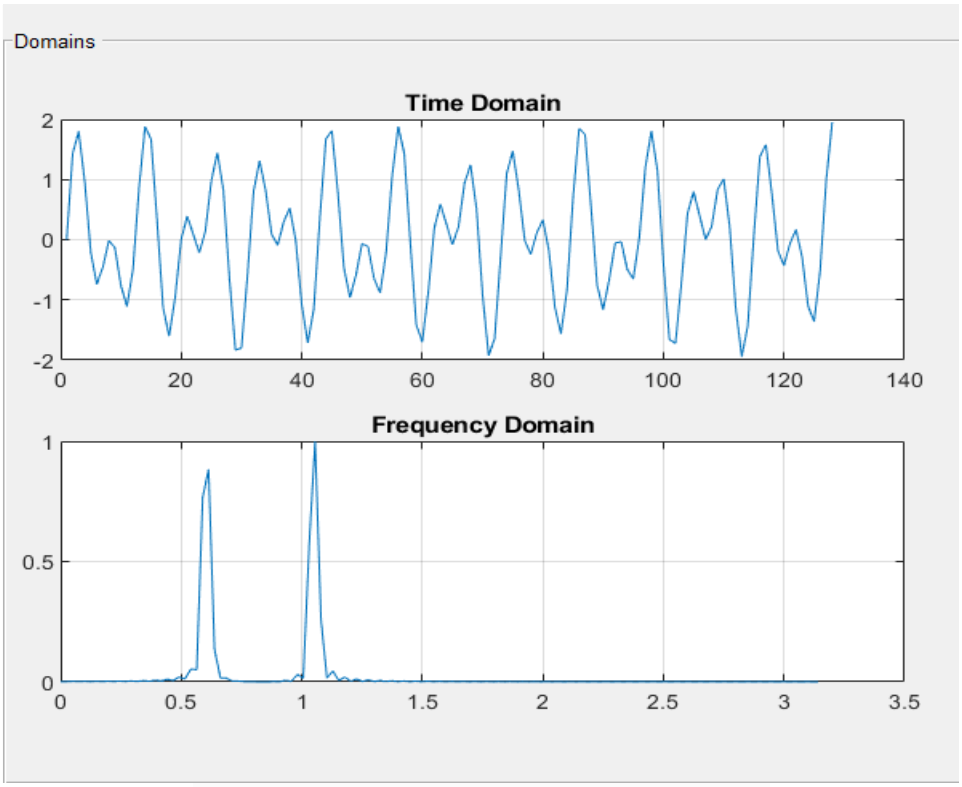

Fig 5: Different domain plots of pressed key

- **DTMF detection:**
  To detect the DTMF we have used the time domain plot and the frequency domain plot generated. After generation, the DTMF signal is observed by the detection or decoder unit. In telephony, DTMF signal is generally detected using filter banks. But we have used Goertzel Algorithm. We have found out the weights of input frequency from the time domain plot of the pressed key by using this algorithm. We could have used DFT or FFT but we have given preference to Goertzel because this Algorithm can do the same work as DFT or FFT but by using a very lesser span of the signal. As it takes a particular number of values of frequencies to decode and discards unnecessary frequency from calculation, it offers a much better detection and decoding speed than FFT.

  In our project, at first, we are detecting the necessary frequencies. Then the frequencies are being sampled to make a discrete value array of the weight of the frequencies in the frequency domain. This work is done by the algorithm. Then the discrete frequency domain array is plot.

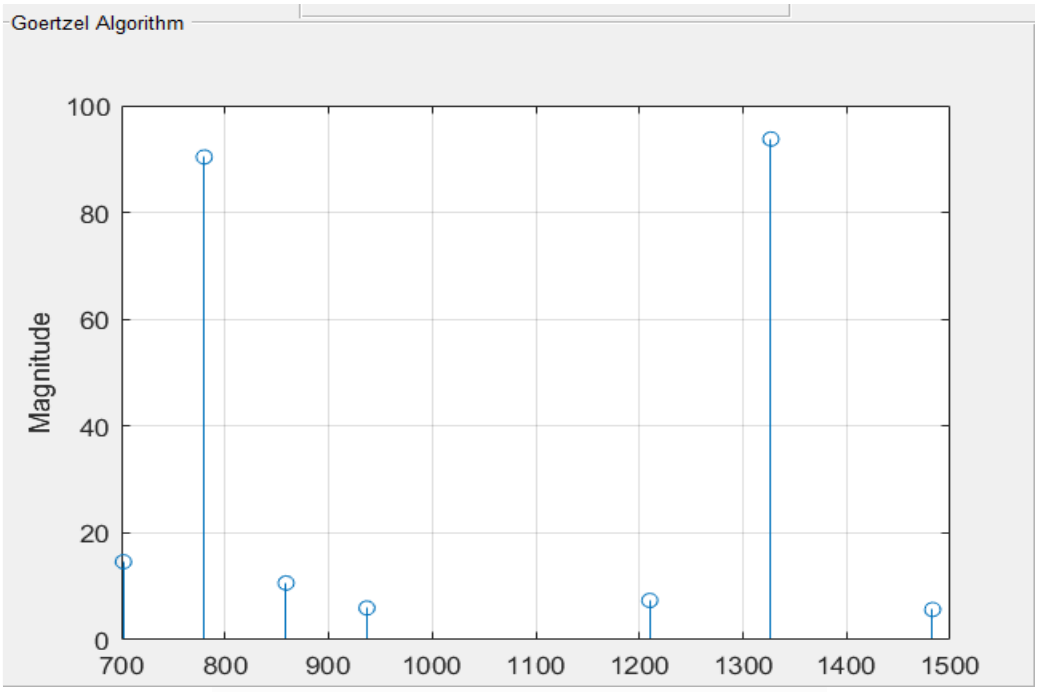

Fig 6 : Discrete plot of detected frequency array

Then we are taking the highest and second highest value from the discrete value array and comparing it with the 7 frequency values from the DTMF chart. After comparison we are able to detect the pressed key.

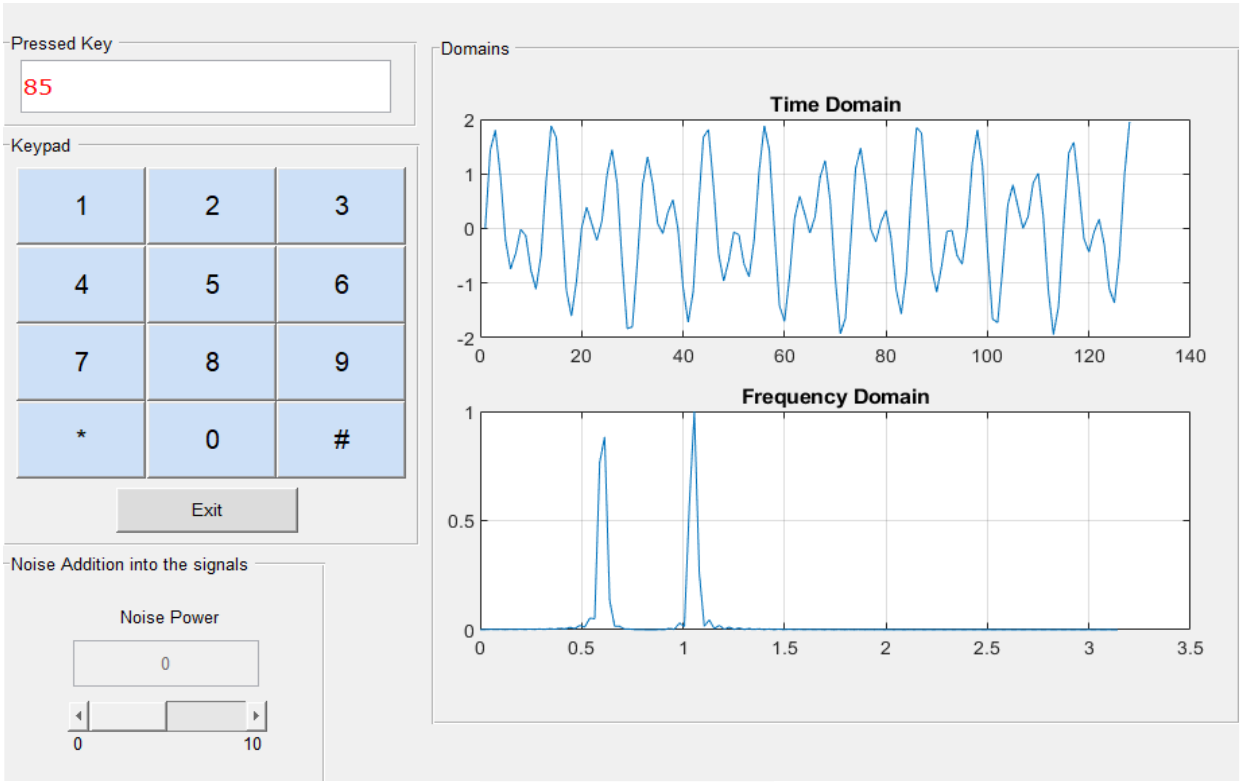

Fig 7: The whole input

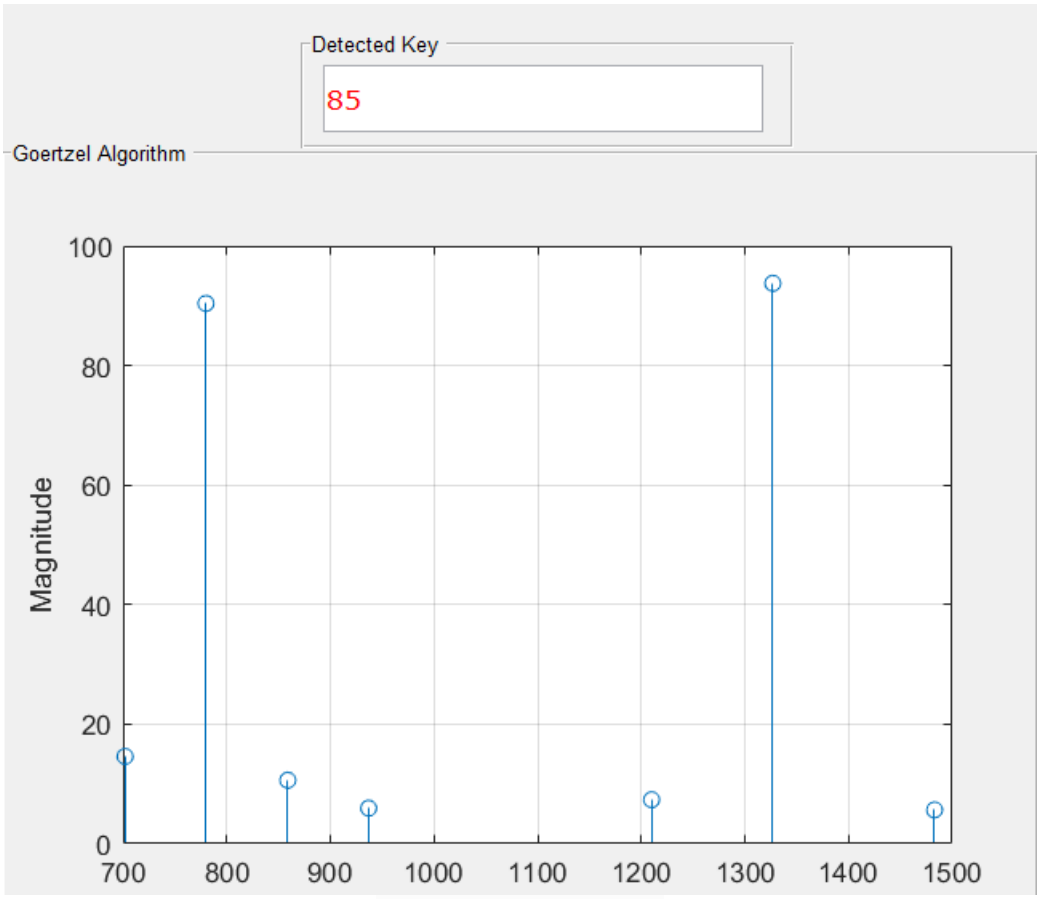

Fig 8: Detected output

- **DTMF detection with Noise:**
  For our final act, we have added random noise to the generated DTMF. We have kept the noise factor as a dial in our GUI.

```
axes(handles.goertzel)
f = [697 770 852 941 1209 1336 1477];
Fs = 8000;
N = 205;
n=0:N-1;
w= noisepower;
np= noisepower*randn(1,N);
x = sum(sin(2*pi*[flow;fhigh]*(n)/Fs))+np;
k = round(f/Fs*N);
xd = goertzel(x,k+1);
ef = round(k*Fs/N);
stem(ef,abs(xd))
xlabel('Frequency');
ylabel('Magnitude');
grid on;
```

Fig 9: Random Noise generate

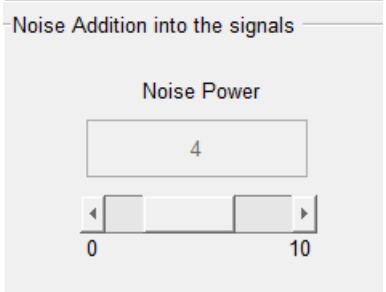

Fig 10: Noise Dial

As a result, we can increase and decrease the value of noise without restarting the interface. Then we have used the same detection procedure as before to detect the sent DTMF from the noisy signal received.

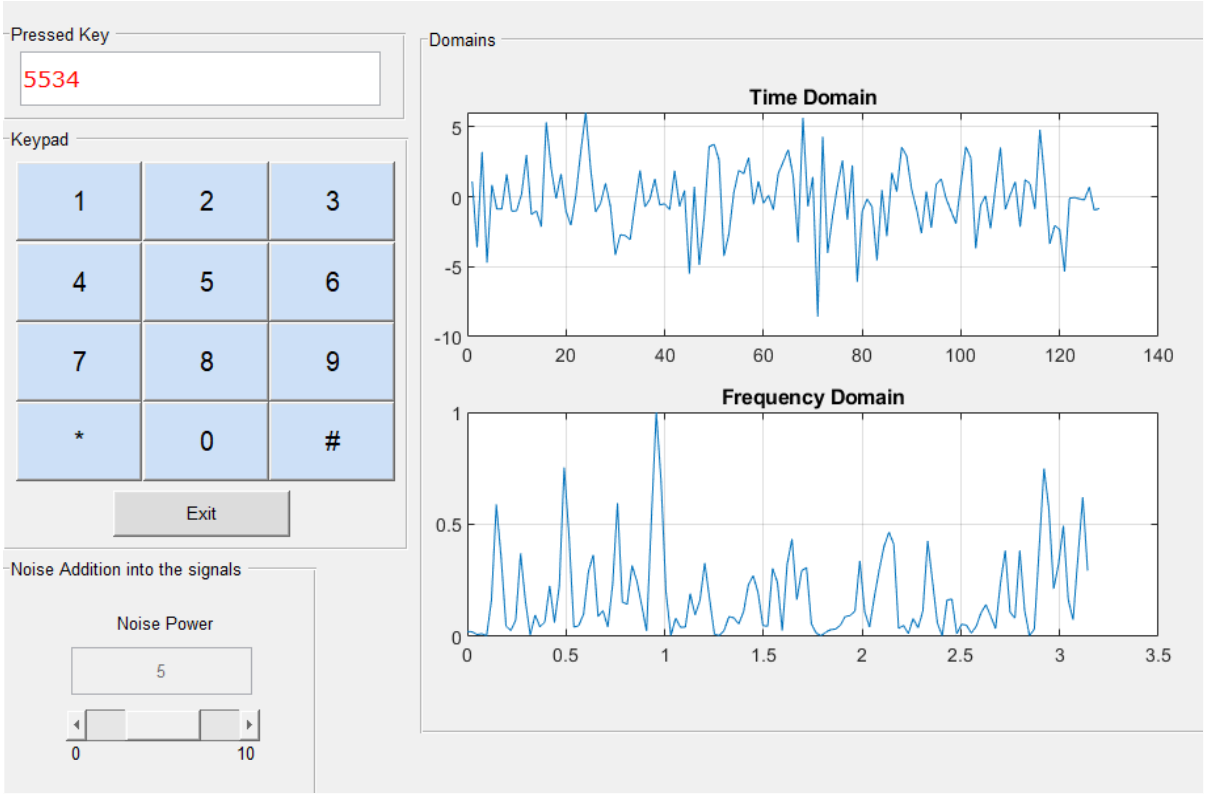

Fig 11: Input with noise

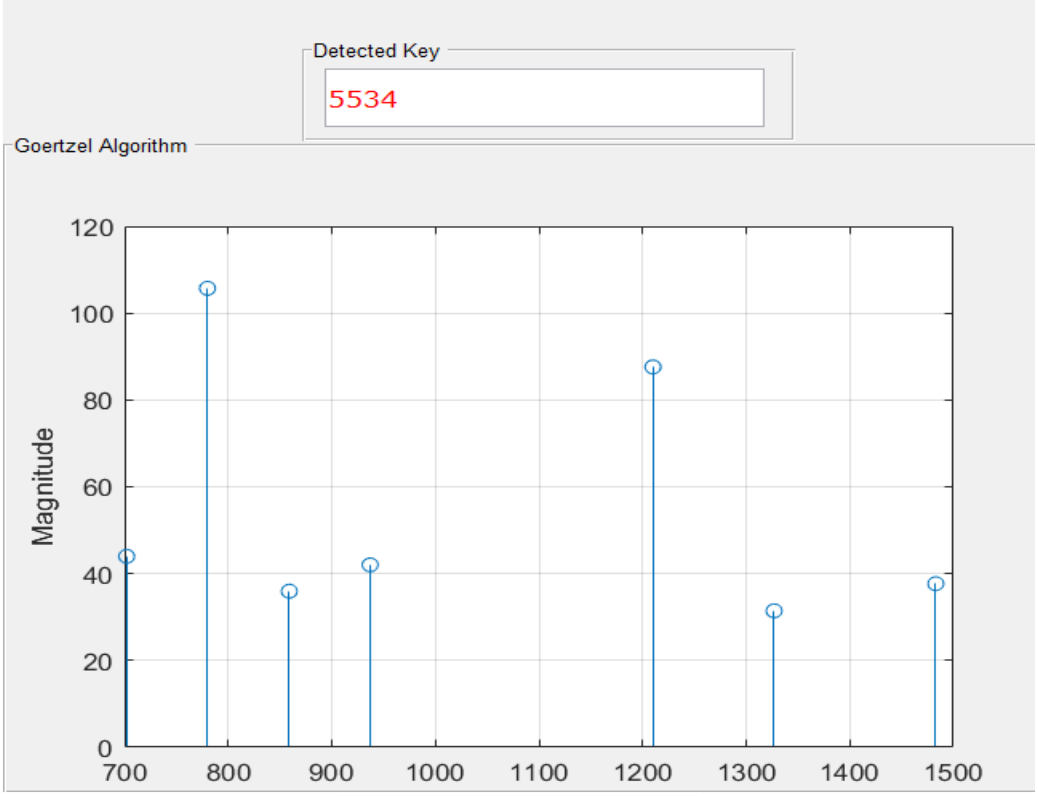

Fig 12: detection with noise

Result Analysis:

- **Normal signal:**
  For the signals without noise, we can observe that by using the Goertzel algorithm we can perfectly detect the input signal.
- **Noisy signal:**
  In case of noisy signal, detection accuracy is inversely proportional to the noise factor. The more we add noise the lesser the accuracy gets.

  When we add less amount of noise, we can see that the detector is upholding its capability.

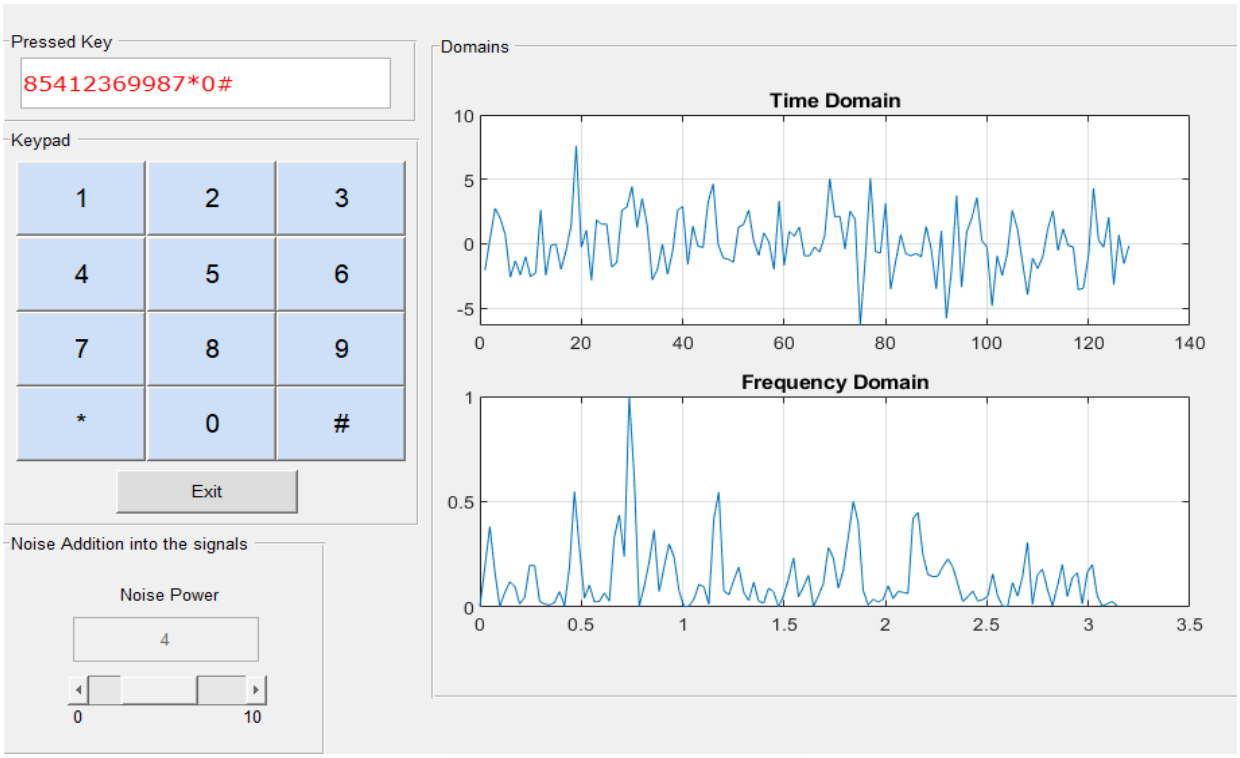

Fig 13: input with less noise

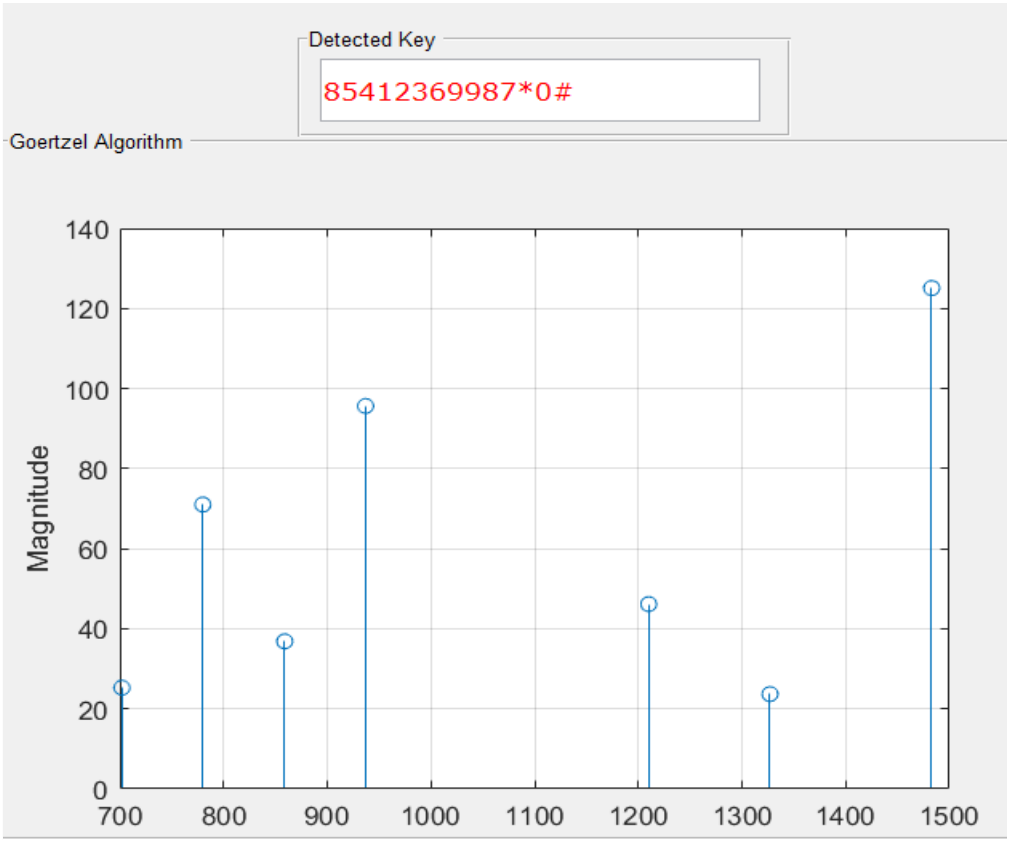

Fig 14: detection with less noise

But here with a significant amount of noise, the capability is seen to be gradually decaying resulting in false or no detection.

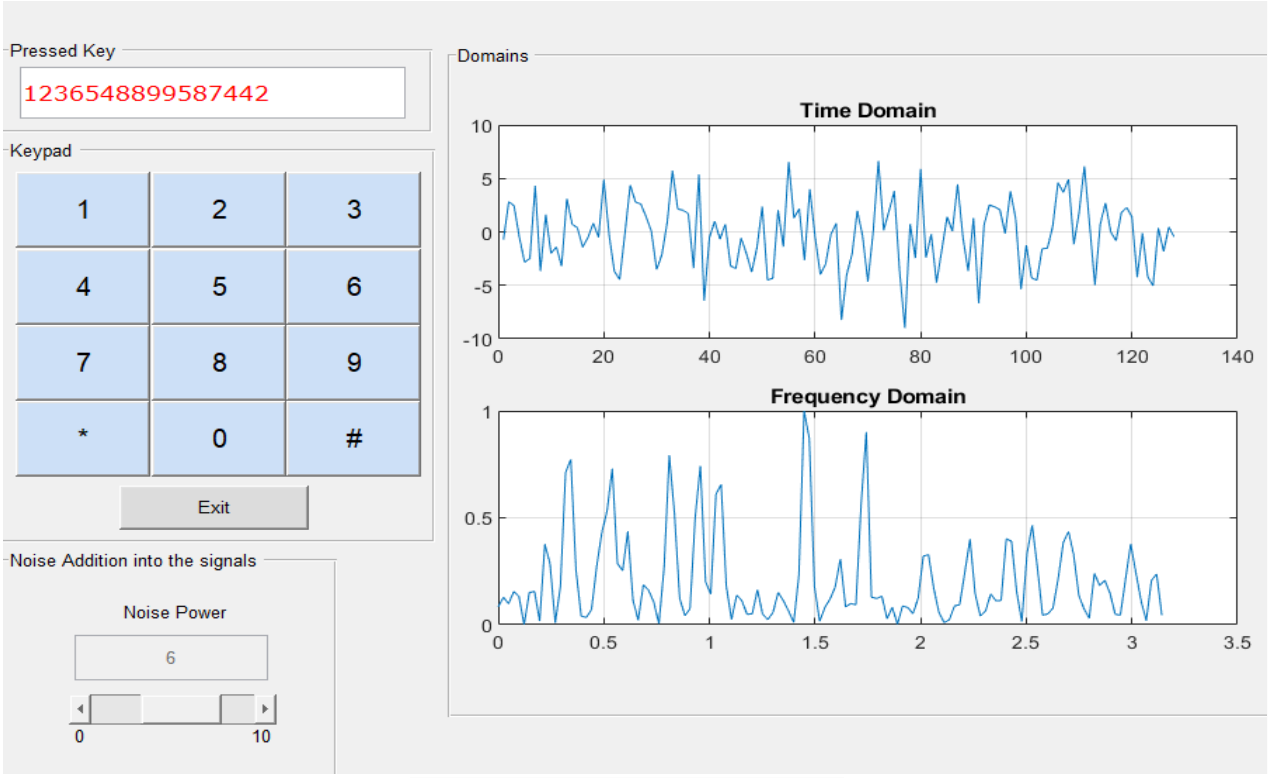

Fig 15: input with moderate noise

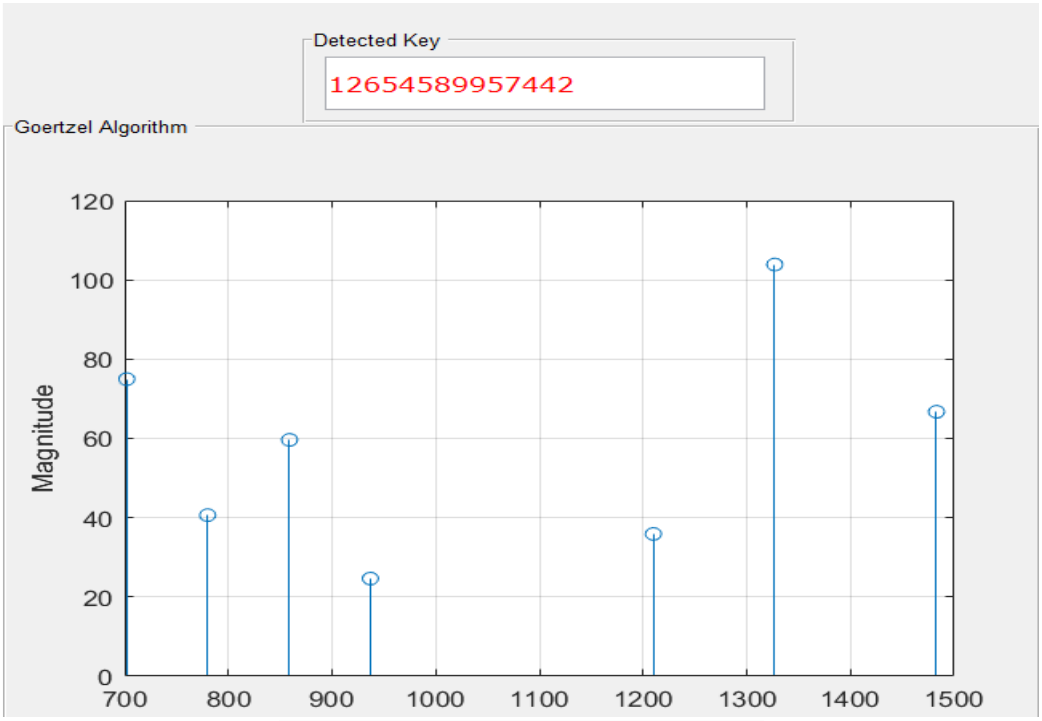

Fig 16: detection with moderate noise

Fig 17: input with high noise

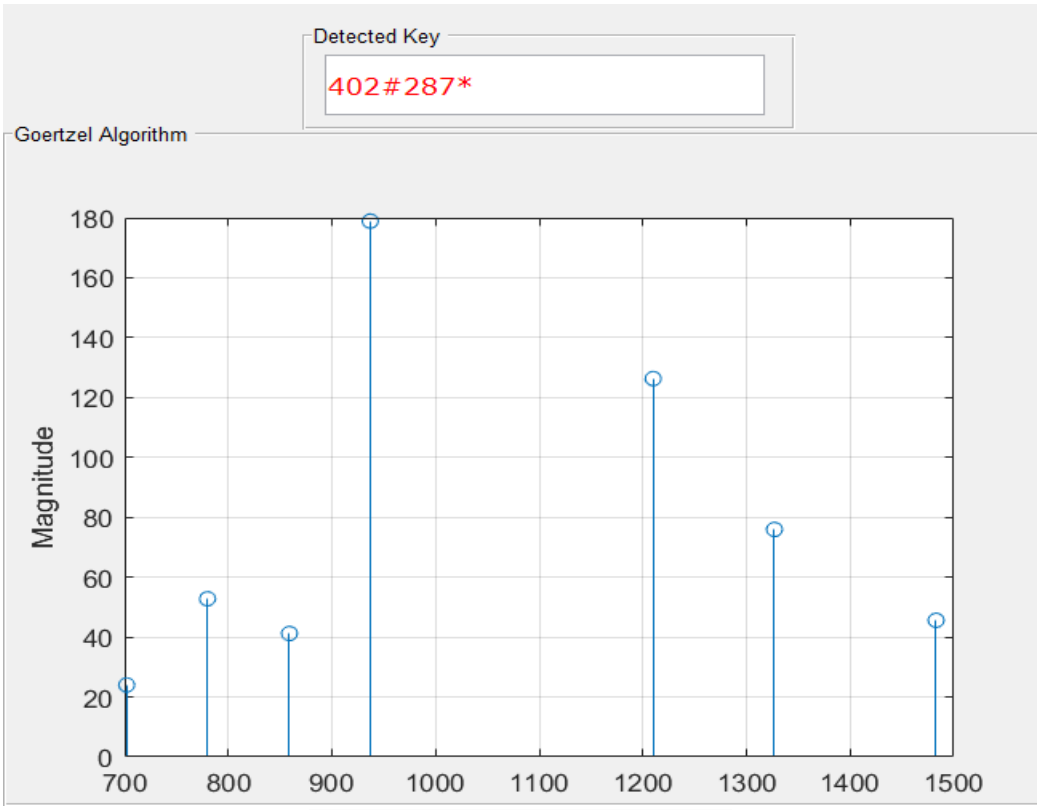

Fig 18: detection with high noise

Conclusion:

In conclusion, it can be shown that the algorithm can carry out the detection with the maximum level of accuracy and at a substantially faster rate. But the algorithm is viable up to a certain level of noise.We have determined that this noise stage is up to 5 factor units. After this stage of noise, the detection accuracy drastically falls down. And since it is random noise, the deviation is also unpredictable. To counter this issue, we may add extra noise reduction mechanisms to filter the original signal and have detection with higher accuracy.