

Department of Electrical & Electronics Engineering

**Abdullah Gül University**

**COCHLEAR IMPLANT SIMULATION PROJECT WITH FIR & IIR FILTER BANK PROJECT**

**EE3001 DSP Lab 6 & Lab 7**

**Submitted on:**

**Submitted by:**

**Oğuzhan Alasulu**

|  |  |  |
| --- | --- | --- |
|  |  | |
|  |  |  | |

**Lab Instructors: Oğuzhan Ayyıldız**

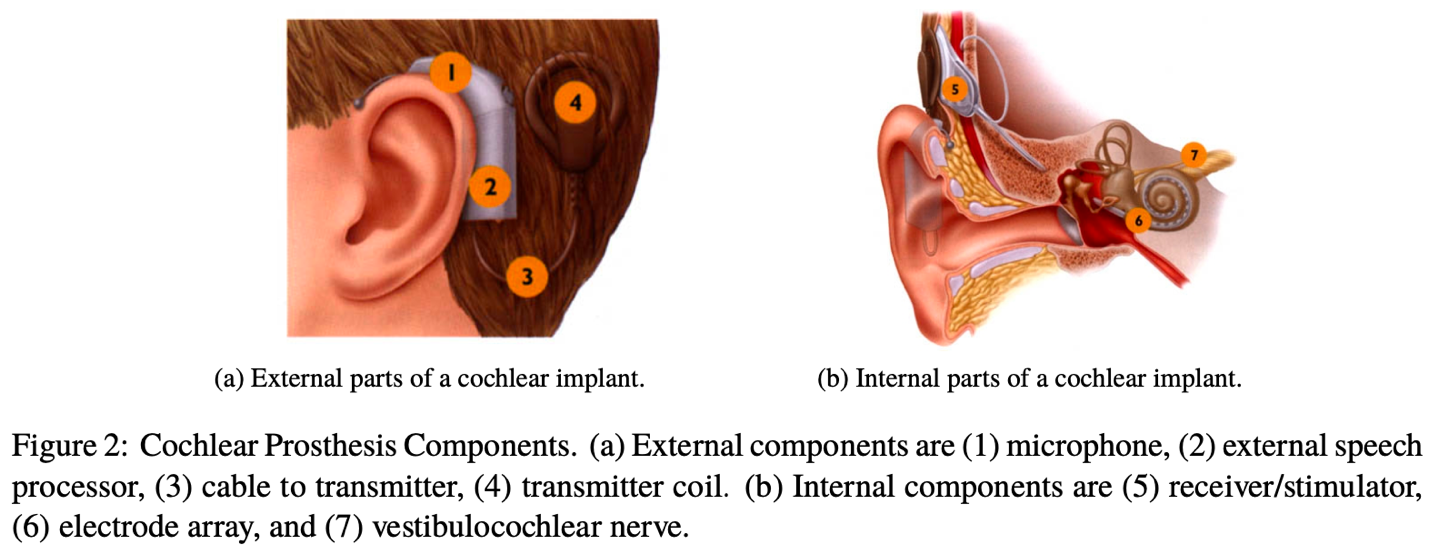
**Grade: / 100**

**Introduction**

A Cochlear Implant (CI) technology has been used for deafness through electrically stimulating auditory nerve fibers. The purpose of this lab project is to simulate the speech processing function of a CI. Comprehending the internal workings of a CI device would be useful to evaluate the hearing process. The human ear contains three parts: outer, middle, and inner, as illustrated in Fig. 1a.

A diagram of a human ear

Description automatically generated



In normal hearing process of a human, the outer ear captures sound, which will be transformed into mechanical vibrations by the middle ear. These vibrations are going into the cochlea, a fluid-filled coiled tube in the inner ear. The basilar membrane, which separates this tube, has different mechanical parts, to build the hearing process more sensitive to high frequencies at the base and low frequencies near the tip (see Fig. 1b). The movement of the fluid creates the frequency data of the acoustic signal. Small hair cells with stereocilia, linked with the membrane, turn when the membrane is displaced. Turning these hair cells triggers neurons that transmits signals to the brain, carrying acoustic information. Deafness mostly caused by a trouble in this process, regularly due to damaged hair cells. A cochlear implant avoids this damage by triggering the auditory nerves with electrical patterns interpreted by speech processing. Figures 2a and 3 demonstrate the external parts of a cochlear implant and the diagram of the major signal processing blocks.

**A diagram of a computer scheme

Description automatically generated**

**Background**

**Cochlear Implant Speech Processing**

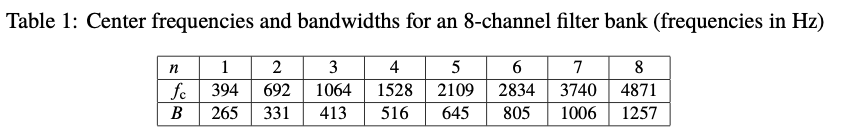
The external parts of a cochlear implant consist of a microphone, a speech processor, a transmitter/receiver, and an electrode array, as shown in Fig. 2a. The function of speech processor could be simulated using MATLAB. Cochlear speech processing includes four parts: pre-emphasis, many band-pass filters (BPFs) in a filter bank structure, envelope detection, and acquiring bipolar pulse. Figure 3 explains the signal flow through these steps. The last step, bipolar pulse generation, produces the signal directed to the current stimulator and next to the electrode array, triggering the nerves. Nevertheless, this current stimulation step will not be simulated in the MATLAB. In its place, the fourth step will be envelope modulation and summation of the channels, which permits the processed sound to be hearable.

**Pre-emphasis Filter**

A pre-emphasis filter is used to increase high frequencies and reduce low frequencies before upcoming processing steps. This improves the low-energy, high-frequency assets in a speech signal related with the high-energy, low-frequency sounds. A simple FIR filter, such as a first-difference filter, can accomplish this. It's significant to indicate that a first-difference filter would eliminate DC from the input speech waveform. Since the low-frequency cutoff for human hearing is 20 Hz, and generally computer microphones and speakers cannot record or produce again low frequencies like that.

**Bandpass Filter Bank**

At this step in speech processing, the pre-emphasized sound waves are separated into parallel channels for every narrow frequency bands. Table 1 lists the center frequency and bandwidth values for an 8-channel filter bank. Each channel employs a bandpass filter (BPF) to separate a small range of frequency components from the sound signal, those within its pass band. The pass band of each BPF ranges from to , where is meaning the width of pass band. For example, the seventh channel in the 8-channel system presented in Table 1 has a BPF that accepts frequencies from 3237 Hz to 4243 Hz.



While designing the cochlear implant (CI) filter bank, the main attention is defining the number of frequency channels and the passband width for every channel. Using too few channels do no result understandable audio in processed sound, when several channels increase computational load. The selection of center frequencies and bandwidths must support the human auditory interval, which is constructed on the frequency-to-place mapping of the cochlea. In conclusion, the bandwidths () of the filter banks are spaced in logarithm manner, meaning that the difference between log() values are constant, or equivalently, the ratio between the bandwidths of channels is constant.

**Envelope Detection**

Determining the envelope of each sub-band signal is important. Each BPF output is exhibited as a narrowband signal, involving a varying envelope signal that modulates a high-frequency sinusoid at the center frequency of the sub-band channel. This model denotes the narrowband signal as an amplitude modulation (AM) signal. Envelope extraction contains two steps: full-wave rectification and low-pass filtering. Full-wave rectification takes the magnitude of the sub-band signal, and low-pass filtering then smooths the signal, attenuating the high-frequency parts of the sinusoid at the center frequency.

**Envelope Modulation**

The external signal processor for the cochlear implant produces the envelope signals for each channel and transmits signals to the internal receiver. The receiver employs the envelopes to trigger auditory nerve fibers at special locations throughout the electrode array. This stimulation is completed through using amplitude-modulated signals, conveying output at each electrode place. Since the cochlea maps locations to different frequencies, each frequency band in the filter bank matches with a specific electrode position. For example, the first channel (the lowest frequency) in the filter bank corresponds to the deepest electrode place. In converse to that, the highest-frequency channel corresponds to the most fundamental electrode place, delivering the highest frequency stimulation.

Signal processing blocks cannot reproduce the interaction between the electrodes and auditory nerve fibers that construct the various frequencies that we perceive. In this simulation, the slowly varying envelope signal from each channel is used to rebuild a signal at the center frequency of that channel. Especially, the envelope for the n-th channel modulates a sinusoid at the center frequency of the n-th channel. These sinusoids from all channels are then added to generate a speech signal that approaches what a CI user would hear.

**Projects**

**LAB 6 PART OF CODES EXPLANATION:**

1. **Lowpass Filter Design**

In clause (a), different lowpass filters (LPFs) for each channel has been ordered, with the passband boundaries demonstrated by the bandwidths of bandpass filter (BPF).

lpfLength = 128; % Length of the LPF

lpfs = zeros(numChannels, lpfLength); % Storing the empty channels

for k = 1:numChannels

passbandEdge = B(k) / 2 / (Fs / 2); % Passband edge frequency (half of the BPF bandwidth)

lpfs(k, :) = fir1(lpfLength-1, passbandEdge);

end

* **lpfLength = 128:** Identifies the length of the lowpass filter.
* **lpfs = zeros(numChannels, lpfLength):** Storing a space for the low pass filters for each channel. The for loop repeats in every channel to design low pass filers.
* **passbandEdge =**  The normalized cutoff frequency for each low pass filter is determined with considering the bandwidth of the band pass filter.
* **lpfs(k, :) = fir1(lpfLength-1, passbandEdge):** Designs the low pass filter using the Hamming-sinc method with the stated passband limits.

1. **DC-Notch Filter Design**

In clause (b), a DC notch filter is designed to remove the DC component presented by the magnitude operation.

a\_values = [0.95, 0.98, 0.99, 0.995];

figure;

hold on;

for a = a\_values

[h\_notch, w] = freqz([1 -1], [1 -a], 1024, Fs);

plot(w, abs(h\_notch));

end

title('Frequency Response of DC-notch Filter for different "a" values');

xlabel('Frequency (Hz)');

ylabel('Magnitude');

legend('a = 0.95', 'a = 0.98', 'a = 0.99', 'a = 0.995');

hold off;

* **a\_values = [0.95, 0.98, 0.99, 0.995]:** Describes the values of the pole location to be tried.
* **for a = a\_values:** Repeats over each value of **a** to plot the frequency response.
* **freqz([1 -1], [1 -a], 1024, ):** Calculates the frequency response of the notch filter.
* **plot(w, abs(h\_notch)):** Plotting the magnitude of the frequency response.

1. **Determining the Pole for Desired Notch Bandwidth**

In clause (c), we define the proper value of such that the notch filter has a bandwidth of 100 Hz or smaller at 0.9 magnitude.

notch\_bandwidth = 100 / (Fs / 2);

a = 0.99; % Initial guess for "a"

for a\_candidate = 0.95:0.0001:0.9999

[h\_notch, w] = freqz([1 -1], [1 -a\_candidate], 1024, Fs);

notch\_bandwidth\_actual = sum(abs(h\_notch) >= 0.9) / length(w) \* (Fs / 2);

if notch\_bandwidth\_actual <= 100

a = a\_candidate;

break;

end

end

* **notch\_bandwidth = 100 /** : Normalizing the notch bandwidth.
* The **for** loop repeats over possible values of **a** to find the one that gets the wanted notch bandwidth.
* **freqz([1 -1], [1 -a\_candidate], 1024, ):** Computes the frequency response for the current candidate a.
* **notch\_bandwidth\_actual = sum(abs(h\_notch) >= 0.9) / length(w) \* ( / 2):** Computes the notch bandwidth.
* **If notch\_bandwidth\_actual <= 100,** the loop breaks and **a** is set to the present candidate.

1. **Plotting Frequency Responses**

In clause (d), we select one channel and plot the frequency responses of the Hamming-sinc low pass filters, the DC-notch filter, and the cascade of both filters.

channelToPlot = 1; % Pick one channel

[h\_lpf, w\_lpf] = freqz(lpfs(channelToPlot, :), 1, 1024, Fs); % Hamming-sinc LPF

[h\_cascade, w\_cascade] = freqz(conv(lpfs(channelToPlot, :), [1 -1]), [1 -a], 1024, Fs); % Cascade system (LPF followed by the DC-notch filter)

figure;

subplot(3,1,1);

plot(w\_lpf, abs(h\_lpf));

title('Frequency Response of Hamming-sinc LPF');

xlabel('Frequency (Hz)');

ylabel('Magnitude');

subplot(3,1,2);

plot(w, abs(h\_notch));

title(['Frequency Response of DC-notch Filter with a = ' num2str(a)]);

xlabel('Frequency (Hz)');

ylabel('Magnitude');

subplot(3,1,3);

plot(w\_cascade, abs(h\_cascade));

title('Frequency Response of Cascade System (LPF followed by DC-notch Filter)');

xlabel('Frequency (Hz)');

ylabel('Magnitude');

* **channelToPlot = 1:** Chooses the first channel for plotting.
* **freqz(lpfs(channelToPlot, :), 1, 1024, ):** Calculating the frequency response of the low pass filter for the required channel.
* **freqz(conv(lpfs(channelToPlot, :), [1 -1]), [1 -a], 1024, ):** Calculating the frequency response of the cascade system.
* Plots the frequency responses for the low pass filters, DC-notch filter, and the cascade system in separate subplots.

A screenshot of a screen

Description automatically generated

A group of blue and white sound waves

Description automatically generated

A pre-emphasis filter is implemented to the speech signal to boost the high frequencies. The pre-emphasis filter is a high-pass filter with the difference equation:

where = 0.97 is the pre-emphasis coefficient. The frequency response of that filter is:

A graph of a graph of a number of different values

Description automatically generated with medium confidence

The shows the frequency response of DC-notch Filter for different “” values. In that way, we can investigate mathematical explanations of DC-notch filter and frequency response of that. Firstly, let’s look at DC-notch filter.

* **DC Notch Filter:** The filter eliminates DC component (0) Hz by creating a zero at . The transfer function of the DC-notch filter is given by:

Where controls the bandwidth of the notch. The closer ” to 1 means narrower notch. In next step, let’s look at frequency response

* **Frequency Response:** The shows DC-notch filter with different values of The frequency response can be calculated with:

A graph of a graph

Description automatically generated

shows the value of ‘’ that can be explained with the bandwidth of This can be found with trying possible values of and measuring the bandwidth where magnitude response is above 0.9

A graph of frequency response

Description automatically generated

Fig. 8 consist of three plots which are:

* **Frequency Response of Hamming-sinc LPF:** Displaying the frequency response of the low-pass filter with using the Hamming window.
* **Frequency Response of DC-Notch Filter:** Displaying the frequency response of DC-Notch filter with different values.
* **Frequency Response of Cascade System:** Displaying the combined frequency response when low pass filters followed by DC-Notch filters.

In next step, let’s examine the mathematical explanation of plots:

* **Low Pass Filter:** Using theHamming window, its frequency response is given by
* **Cascade System:** The combined system response is the product of individual responses of the filters

.

A screen shot of a graph

Description automatically generated

The spectrogram (first plot in ) shows the STFT (Short-Time Fourier Transform) which interpret the frequency content of the signal versus time. Second plot which is the power spectrum shows the power spectral density of the audio output.

**LAB 7 PART OF THE CODES EXPLANATION:**

1. **Lowpass Filter Design**

% Designing a different LPF for each channel

lpfLength = 128; % Length of the LPF

lpfs = cell(numChannels, 1); % Preallocating the channels

for k = 1:numChannels

passbandEdge = B(k) / 2 / (fs / 2); % Implementaion of passband limit frequencies and filters (half of the BPF bandwidth)

[b, a] = butter(4, passbandEdge, 'low');

lpfs{k} = {b, a};

end

In this part, a Butterworth low pass filter is designed for every channel. The passband edge of every low pass filter is explained with the bandwidth of the band pass filters. A lower pass band boundary is used for band pass filters. The low passes are used as 4-th order Butterworth filter using the **‘butter’** function.

1. **Notch Filtering for DC removal**

a\_values = [0.95, 0.98, 0.99, 0.995];

% Magnitude Responses Figures

figure;

hold on;

for a = a\_values

[h\_notch, w] = freqz([1 -1], [1 -a], 1024, fs);

plot(w, abs(h\_notch));

end

title('Frequency Response of DC-notch Filter for different "a" values');

xlabel('Frequency (Hz)');

ylabel('Magnitude');

legend('a = 0.95', 'a = 0.98', 'a = 0.99', 'a = 0.995');

hold off;

In this part, notch filters values of "**a**" are selected and their frequency responses are displayed. The notch filter is applied to employing an IIR system function with a zero at the unit circle and a pole close by. The freqz function is implemented to calculate the frequency response of every notch filter.

The loop repeats different "**a**" value, and looking for values, the magnitude response of the notch filter is exhibited. This delivers changing "a" parameter.

1. **Determining Notch Filter Pole Value**

for a\_candidate = 0.95:0.0001:0.9999

[h\_notch, w] = freqz([1 -1], [1 -a\_candidate], 1024, fs);

notch\_bandwidth\_actual = sum(abs(h\_notch) >= 0.9) / length(w) \* (fs / 2);

if notch\_bandwidth\_actual <= 100

a = a\_candidate;

break;

end

end

The code repeats every interval of "" values to find the specified notch bandwidth constraint (<= 100 Hz). For each "" candidate, the frequency response of the notch filter is calculated using **freqz**, and the notch bandwidth is designed.

The loop runs until an appropriate "" value is found that corresponds the bandwidth limitation. When it is found, this value will be used to the variable "" for further use in the notch filter design.

1. **Frequency Response Comparison**

[h\_lpf, w\_lpf] = freqz(lpfs{channelToPlot}{1}, lpfs{channelToPlot}{2}, 1024, fs); % Butterworth LPF

[h\_cascade, w\_cascade] = freqz(conv(lpfs{channelToPlot}{1}, [1 -1]), conv(lpfs{channelToPlot}{2}, [1 -a]), 1024, fs); % Cascade system (LPF followed by the DC-notch filter)  
A screenshot of a screen

Description automatically generated  
A group of blue and white sound waves

Description automatically generated

The same pre-emphasis filter in previous lab project is implemented to the speech signal for improving high frequencies. The pre-emphasis filter is a high-pass filter with the difference equation where = 0.97 is the pre-emphasis coefficient:

The frequency response of that filter is:

A graph of a graph showing a number of different values

Description automatically generated

The DC-Notch filter is removing the DC component in the system:

The value of shows the width of the notch. Different values of are tested to find an appropriate value for notch bandwidth which would be equal or smaller than 100Hz. The transfer function in time domain is:

For better comprehension in the concept, let’s compare the FIR and IIR differences on the figures, bandpass filters and envelope extractions briefly. In the Lab 7 part, we use IIR domain operations.

**Bandpass Filter (FIR):** Designed using Hamming-sinc method

where is the hamming window and are normalized cutoff frequencies.

**Bandpass Filter (IIR):** Designed using the Butterworth method

**Envelope Extraction (FIR):** Designed using another FIR filter (Low pass filter)

**Envelope Extraction (IIR):** Designed using another FIR filter (Low pass filter)

A screen shot of a graph

Description automatically generated

The final value of that corresponds the notch bandwidth is used to display the frequency response of DC-Notch filter. The notch bandwidth is iterated as the frequency range where the magnitude response is greater than or equal to 1. The formula is common for both versions for various values. (0.95, 0.98, 0.99, 0.995)

where is the depth and width of the notch at 0 Hz.

A screen shot of a graph

Description automatically generated

* **Low-pass Filter Response:** The butterworth low-pass filter implemented to the envelope. The transfer of the low-pass filter:

Where and are the polynomials of Butterworth design.

* **DC-Notch Filter Response:** The frequency response of the DC-notch filter
* **Cascade System Response:** The frequency response of the cascade low pass filter with DC-Notch filter is the convolution of their impulse responses.

A screen shot of a graph

Description automatically generated

This part is the same with FIR project wart

**References**

* McClellan, Schafer, and Yoder, *DSP First, 2e*, ISBN 0-13-065562-7. 1 McClellan, Schafer and Yoder, *Signal Processing First*. Prentice Hall, Upper Saddle River, NJ 07458. ©2015 Pearson Education, Inc
* P. T. Bhatti and J. H. McClellan, “A Cochlear Implant Signal Processing Lab: Exploration of a Problem-Based Learning Exercise,” *IEEE Transactions on Education,* Vol. 54, No. 4, pp. 628–636, 2011. DOI: 10.1109/TE.2010.2103317
* Luo, J. H. McClellan and P. T. Bhatti, “Introductory signal processing labs based on filterbank applications,” *2011 IEEE Digital Signal Processing Workshop and IEEE Signal Processing Education Workshop (DSP/SPE),* Sedona, AZ, pp. 90–94, 2011. DOI: 10.1109/DSP-SPE.2011.5739192
* C. Loizou, “Introduction to Cochlear Implants,” *IEEE Engineering in Medicine and Biology Magazine,* Vol. 18, No. 1, pp. 32–42, 1999. DOI: 10.1109/51.740962.
* P. C. Loizou, “Mimicking the human ear,” *IEEE Signal Processing Magazine,* Vol. 15, No. 5, pp. 101–130, 1998. DOI: 10.1109/79.708543
* P. C. Loizou, et. al., “On the number of channels needed to understand speech,” *J. Acoustical Soc. of America,* vol. 106, pp, 2097–2103, 1999.
* In reality “The amplitudes of the sinusoids are computed by estimating the root-mean- square (rms) energy of the envelopes every 4 msecs.” Loizou ref.